National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

Visual Representation Learning for Document Image Recognition

Ph.D. Dissertation

Georgios Retsinas

Dipl.-Ing. in Electrical and Computer Engineering, NTUA

Supervisor: Prof. Petros Maragos

Athens, February 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων, Ελεγχου και Ρομποτικης
Εργαστηριο Ορασης Υπολογιστων, Επικοινωνιας Λογου και
Επεξεργασιας Σηματων

Εκμάθηση Οπτικών Αναπαραστάσεων για Αναγνώριση σε Εικόνες Εγγράφων

Διδακτορικη Διατριβη

του

Γεώργιου Ρετσινά

Διπλωματούχου Ηλεκτρολόγου Μηχανικού &
Μηχανικού Υπολογιστών Ε.Μ.Π.

Επιβλέπων: Πέτρος Μαραγκός, Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Σημάτων, Ελέγχου και Ρομποτικής

Εκμάθηση Οπτικών Αναπαραστάσεων για Αναγνώριση σε Εικόνες Εγγράφων

Διδακτορική Διατριβή

του

Γεώργιου Ρετσινά

Διπλωματούχου Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών Ε.Μ.Π.

Συμβουλευτική Επιτροπή: Πέτρος Μαραγκός, Καθηγητής
Βασίλειος Γάτος, Ερευνητής Α΄
Κωνσταντίνος Τζαφέστας, Αναπλ. Καθηγητής

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 27/02/2020.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....	.....
Πέτρος Μαραγκός	Βασίλειος Γάτος	Κωνσταντίνος Τζαφέστας	Γεράσιμος Ποταμιάνος
Καθηγητής	Ερευνητής Α΄	Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής
Ε.Μ.Π.	ΕΚΕΦΕ Δημόκριτος	Ε.Μ.Π.	Πανεπιστήμιο Θεσσαλίας

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Γκούμας	Βασίλειος Κατσούρος	Στέφανος Κόλλιας
Επίκουρος Καθηγητής	Ερευνητής Α΄	Καθηγητής
Ε.Μ.Π.	Ερευνητικό Κέντρο ΑΘΗΝΑ	Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2020

(Υπογραφή)

.....
Γεώργιος Ρετσινάς
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright ©–All rights reserved Γεώργιος Ρετσινάς, 2020.
Με επιφύλαξη παντός δικαιώματος.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ΄ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν το συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διδακτορικής διατριβής κλείνει ένα σημαντικό κεφάλαιο της ζωής μου, μετά από πέντε χρόνια γεμάτα ενδιαφέρουσες προκλήσεις. Η πορεία μου ως υποψήφιος διδάκτορας άρχισε από το ερευνητικό κέντρο «Δημόκριτος», για τέσσερα χρόνια, και τελείωσε στο Εθνικό Μετσόβιο Πολυτεχνείο και το εργαστηρίου Ρομποτικής και Αυτοματισμού, με το οποίο έχω πολυετή συνεργασία.

Θα ήθελα καταρχάς να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή Πέτρο Μαραγκό για την άψογη συνεργασία και την ανεκτίμητη καθοδήγησή του όλα αυτά τα χρόνια και τον Βασίλειο Γάτο για την ευκαιρία που μου έδωσε να εκπονήσω ένα μεγάλο μέρος της διατριβής μου στον ερευνητικό κέντρο ΕΚΕΦΕ 'Δημόκριτος'. Επίσης θα ήθελα να ευχαριστήσω τον Γεώργιο Γκούμα για τη σύντομη συνεργασία μας, καθώς και τα υπόλοιπα μέλη της επταμελούς επιτροπής για τις συμβουλές τους.

Θα ήθελα ακόμα να ευχαριστήσω την ομάδα του εργαστηρίου Υπολογιστικής Νοημοσύνης του Δημόκριτου, και συγκεκριμένα τους Γιώργο Λουλούδη, Νίκο Σταματόπουλο και Γιώργο Σφήκα, για την πολύτιμη καθοδήγηση που προσέφεραν, συνεισφέροντας τα μέγιστα στην ακαδημαϊκή μου πορεία. Επιπλέον, δεν γίνεται να παραλείψω από τις ευχαριστίες μου τα μέλη του εργαστηρίου Ρομποτικής και Αυτοματισμού, με τα οποία είχα άψογη συνεργασία και επικοινωνία, ενώ είναι και η ιδανική παρέα για όλες τις ώρες. Πολύτιμη βοήθεια και στήριξη προσέφεραν και οι συνάδελφοι, αλλά πρωτίστως πολύ καλοί φίλοι, Αθηνά, Δημοσθένης και Βασίλης.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου που με στήριξε καθ' όλη την διάρκεια των σπουδών μου αλλά και την κοπέλα μου και τους φίλους μου που προσέφεραν την απαραίτητη περίσπαση από τα ακαδημαϊκά μου καθήκοντα.

Εκτεταμένη Περίληψη

Η ανάλυση και αναγνώριση εικόνων εγγραφών αποτελεί έναν σημαντικό τομέα ερευνάς που συνδυάζει του τομείς της Όρασης Υπολογιστών και Μηχανικής Μάθησης και έχει άμεση επίδραση στον κλάδο των ανθρωπιστικών σπουδών, καθώς είναι δυνατό να αποκτηθεί αυτοματοποιημένη πρόσβαση στην περικλείουσα πληροφορία μεγάλου όγκου ιστορικών εγγραφών. Στη παρούσα διδακτορική διατριβή, επικεντρωνόμαστε στην εξαγωγή και εκμάθηση οπτικών αναπαραστάσεων με σκοπό τον επιτυχή εντοπισμό και την αναγνώριση κειμένου σε χειρόγραφα έγγραφα. Κύριος γνώμονας των αναπτυχθέντων μεθοδολογιών είναι η δημιουργία αποδοτικών συστημάτων με ελαχιστοποιημένες υπολογιστικές απαιτήσεις. Στην πορεία της διατριβής, καταπιανόμαστε με προβλήματα κλιμακούμενης δυσκολίας και απαιτήσεων με τελικό στόχο ένα αποδοτικό σύστημα εντοπισμού λέξεων, εστιάζοντας στην βελτιστοποίηση των οπτικών αναπαραστάσεων από λέξεις. Συγκεκριμένα εξετάζουμε τεχνικές εξαγωγής χαρακτηριστικών, αλλά και τρόπους βελτιστοποίησης της απόδοσής τους, δεδομένου της ιδιομορφίας των εικόνων κειμένου. Κλασσικές τεχνικές εξαγωγής χαρακτηριστικών έρχονται σε αντιδιαστολή με την χρήση τεχνικών βαθιάς μάθησης ως μεθόδους για την παραγωγή αποδοτικών οπτικών αναπαραστάσεων. Ιδιαίτερη σημασία δίνεται στην μελέτη συνελικτικών νευρωνικών δικτύων και κάτω από ποίες συνθήκες είναι ικανά να παράγουν τέτοιες συμπαγείς αναπαραστάσεις, ικανές να γενικεύουν και εκτός του συνόλου εκπαίδευσης. Έχοντας μελετήσει διεξοδικά το πρόβλημα εξαγωγής οπτικών αναπαραστάσεων λέξεων, προτείναμε προσεγγίσεις για την εφαρμογή εντοπισμού λέξεων σε επίπεδο γραμμής για την αποφυγή της κατάτμησης ενός κειμένου σε λέξεις (η κατάτμηση σε γραμμές είναι ένα αρκετά μελετημένο πρόβλημα στην βιβλιογραφία), ενώ προτείναμε και έναν πρωτότυπο τρόπο να συνενώσουμε ένα σύστημα αναγνώρισης και εντοπισμού λέξεων. Τέλος, καταπιαστήκαμε με το γενικό και ευρύτερο θέμα της συμπίεσης νευρωνικών δικτύων, καθώς κατά την μελέτη μας σε τεχνικές βαθιάς μάθησης συχνά συναντήσαμε πολύπλοκα μοντέλα εκατομμυρίων παραμέτρων, ιδιαίτερα απαιτητικά σε πόρους. Σε αυτή την κατεύθυνση προτείναμε δυο εναλλακτικές προσεγγίσεις συμπίεσης, μια με χρήση κοινών βαρών και μια με αραιοποίηση των βαρών, που αξιολογήθηκαν σε δημοφιλείς βάσεις κατηγοριοποίησης εικόνων αλλά και σε εφαρμογές κειμένων. Ακολουθεί μια σύντομη περιγραφή των σημαντικότερων συνεισφορών της παρούσας διατριβής.

Αναγνώριση Χαρακτήρων

Ο πρώτος στόχος της παρούσας δουλείας είναι η εφαρμογή πρωτότυπων μεθόδων για την εξαγωγή χαρακτηριστικών στο πρόβλημα της αναγνώρισης χαρακτήρων. Παρόλο που είναι εξαιρετικά δύσκολο να χωρίσουμε χειρόγραφο κείμενο σε χαρακτήρες, η αποτίμηση οπτικών χαρακτηριστικών στο πρόβλημα της αναγνώρισης χαρακτήρων αποτελεί μια πολύ χρήσιμη ένδειξη των ικανοτήτων της εκάστοτε μεθόδου. Εξετάσαμε δύο κατευθύνσεις: 1) την εξαγωγή χαρακτηριστικών που βασίζονται σε κατευθυντικές προβολές και 2) την δημιουργία ενός συστήματος για τον διαχωρισμό κλάσεων χαρακτήρων με την βοήθεια του χρήστη (δίνοντας έμφαση στην ελαχιστοποίηση της αλληλεπίδρασης του χρήστη). Στην συνέχεια θα αναλύσου-

με την πρότερη δουλεία που θα αποτελέσει βάση και για εφαρμογές εντοπισμού λέξεων.

Projections of Oriented Gradients

Για την εξαγωγή εύρωστων και αξιόπιστων οπτικών αναπαραστάσεων προτείναμε ένα νέο περιγραφητή που βασίζεται στις προβολές κατευθυντικών παραγώγων και τιτλοφορείται Projections of Oriented Gradients (POGs). Στόχος μας ήταν η ενσωμάτωση διαδεδομένων ιδεών, που είχαν αποδειχτεί πρακτικές σε ευρεία γκάμα προβλημάτων, όπως η χρήση των κατευθυντικών παραγώγων για την εξαγωγή χαρακτηριστικών με κύριο χαρακτηριστικό παράδειγμα τους περιγραφητές Histogram of Oriented Gradients (HOGs) [37]. Σε αντίθεση με τα HOGs, τα POGs εφαρμόζουν μια ολική κωδικοποίηση με βάση προβολικούς μετασχηματισμούς (Radon), αντί να κατατμήσουν χωρικά την εικόνα. Στην συνέχεια, αναλύουμε τα βήματα για την εξαγωγή του προτεινόμενου περιγραφητή και παρουσιάζουμε τα πειραματικά αποτελέσματα σε πληθώρα βάσεων μεμονωμένων χαρακτήρων

Τα βήματα για την εξαγωγή του προτεινόμενου περιγραφητή είναι:

- **Κατεύθυνση Παραγώγου:** Αρχικά αναπαριστούμε τις διαφορετικές κατευθύνσεις της παραγώγου με ένα σύνολο δυαδικών εικόνων. Κάθε τέτοια εικόνα αντιστοιχεί σε ένα εύρος γωνιών. Για τον σκοπό αυτό, υπολογίζουμε τις κατευθυντικές παραγώγους G_x ανδ G_y μιας εικόνας $I(x, y)$ χρησιμοποιώντας τους πυρήνες συνέλιξης $[-1\,0\,1]$ και $[-1\,0\,1]^T$. Η γωνία $\angle G$ υπολογίζεται για το διάστημα $[0, 180^\circ)$ (μη προσημασμένη παράγωγος) ως:

$$\angle G(x, y) = arctan\Big(\frac{G_y(x, y)}{G_x(x, y)}\Big) \tag{1}$$

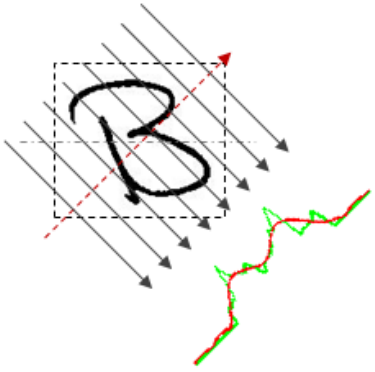
  Επειδή οι αρχικές εικόνες θεωρούνται δυαδικές, χρησιμοποιώντας και τους παραπάνω διακριτούς πυρήνες συνέλιξης, έχουμε 3 πιθανές τιμές για τις κατευθυντικές παραγώγους ($\{-1, 0, 1\}$) και συνεπώς έχουμε μόνο τέσσερεις πιθανές κατευθύνσεις στο διάστημα $[0, 180^\circ)$ (0°, 45°, 90° και 135°).

- **Προβολική Κωδικοποίηση:** Κάθε δυαδική εικόνα κατεύθυνσης θα κωδικοποιηθεί με ολιστικό τρόπο, χρησιμοποιώντας προβολές και συγκεκριμένα τον μετ/σμο Radon. Επιλέγουμε n_θ προβολές, δειγματοληπτημένες κάθε 180° / n_θ (οι γωνίες προβολής είναι $\theta_k = k\frac{180^\circ}{n_\theta}$, $k \in [0, n_\theta - 1]$). Συνεπώς κάθε εικόνα αναπαριστάται από ένα σύνολο μονοδιάστατων προβολών μήκους K. Οι προβολές αυτές περιέχουν και υψίσυχνη πληροφορία που αντιστοιχεί σε θόρυβο, ενώ οι χαμηλόσυχνη αντιστοιχεί στις αλλαγές του περιγράμματος που χαρακτηρίζει κάθε γράμμα. Για τον λόγο αυτό κωδικοποιούμε κάθε προβολή με τους πρώτους n_c συντελεστές Fourier της. Η διαδικασία αυτή παρουσιάζεται στο Σχήμα 1.

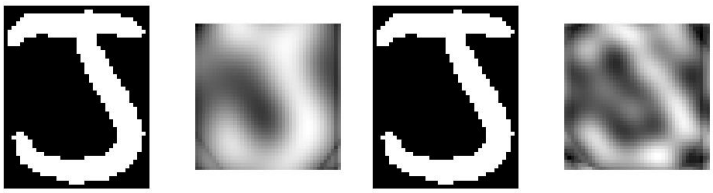
- **Τελικός Περιγραφητής:**

  Τελικά, κάθε εικόνα αναπαριστάται με συνένωση των επιμέρους συντελεστών που περιγράφουν τις προβολές, δημιουργώντας έτσι ένα διάνυσμα χαρακτηριστικών μεγέθους: 5 (αρχική εικόνα + 4 εικόνες κατεύθυνσης) $\times$ n_θ (αριθμός προβολών ανά εικόνα) $\times$ $2n_c$ (αριθμός συντελεστών Fourier - n_c μιγαδικοί συντελεστές).

Αξίζει να αναφέρουμε ότι το 2ο βήμα της παραπάνω διαδικασίας μπορεί να αναστραφεί με χρήση των αντίστροφων μετασχηματισμών Radon και Fourier και συνεπώς μπορούμε να απεικονίσουμε τι ακριβώς κωδικοποιεί η προτεινόμενη μεθοδολογία, όπως φαίνεται στο Σχήμα 2.



Σχήμα 1: Απεικόνιση προβολής ($\theta = 45°$) και της εξομάλυνσής της με χαμηλής συχνότητας συντελεστές.



(α′) $n_c = 2$ (β′) $n_c = 4$

Σχήμα 2: Παραδείγματα οπτικοποίησης της κωδικοποιημένης πληροφορίας ($n_\theta = 6$)

Τα αποτελέσματα αναγνώρισης σε τρεις διαφορετικές βάσεις μεμονωμένων χαρακτήρων παρουσιάζονται στον Πίνακα 1 (Αγγλικά - 52 κλάσεις [89], Ελληνικά - 56 κλάσεις [178], Πολυτονικά Ελληνικά - 125 κλάσεις [55]). Για την κατηγοριοποίηση εφαρμόζουμε την μέθοδο μηχανικής μάθησης Support Vector Machines (SVMs)[34]. Τα αποτελέσματα δείχνουν ότι με μόνο 3 συντελεστές ανά προβολή και 6 προβολές ανά εικόνα μπορούμε να περιγράψουμε επαρκώς εικόνες χαρακτήρων και να πετύχουμε καλύτερα αποτελέσματα από κλασσικές τεχνικές εξαγωγής χαρακτηριστικών.

	AdWin[54]	HOG[37]	Vam1[179]	Vam2[179]	POG3
Uppercase (26 cl.)	85.30%	94.66%	93.78%	95.90%	96.04%
Lowercase (26 cl.)	83.70%	91.17%	89.79%	93.50%	93.68%
All (52 cl.)	69.90%	81.68%	78.42%	85.11%	82.73%

(α) CEDAR dataset [89] - Αγγλικά - 52 κλάσεις

AdWin[54]	HOG[37]	Vam1[179]	Vam2[179]	POG3
87.97%	94.02%	92.53%	95.63%	96.14%

(β) CIL dataset [178] - Ελληνικά - 56 κλάσεις

AdWin[54]	HOG[37]	POG3	POG6
97.71%	98.37%	98.49%	98.60%

(γ) GRPOLY-DB dataset [55] - Πολυτονικά Ελληνικά - 125 κλάσεις

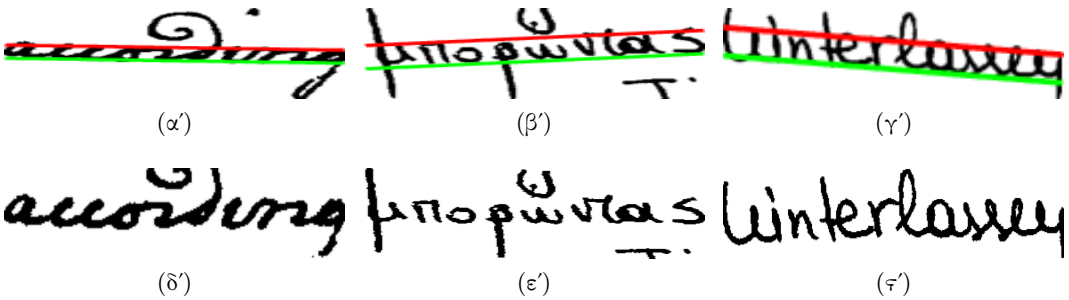
Πίνακας 1: Αποτελέσματα αναγνώρισης για τρεις διαφορετικές βάσεις. Η προτεινομένη μεθοδολογία συμβολίζεται με POG3 και POG6, που αντιστοιχούν σε 3 και 6 (περισσότερη λεπτομέρεια) συντελεστές ανά προβολή αντίστοιχα (6 προβολές ανά εικόνα).

Εντοπισμός Λέξεων

Όπως αναφέραμε, ο βασικός στόχος της παρούσας διατριβής είναι η εύρεση αποδοτικών και αξιόπιστων αναπαραστάσεων για εικόνες κειμένου. Μια εφαρμογή που είναι ιδιαιτέρα χρήσιμη και εναρμονίζεται απόλυτα με την ανάγκη να αξιολογήσουμε αναπαραστάσεις εικόνων από λέξεις κειμένου είναι ο εντοπισμός λέξεων. Ο εντοπισμός λέξεων ορίζεται ως ένα πρόβλημα ταξινόμησης όπου δίνοντας μια εικόνα- ή κείμενο-ερώτηση (query), πρέπει να επιστραφούν μια ταξινομημένη λίστα με τις πιο σχετικές εικόνες. Συνεπώς μια συχνή κατηγοριοποίηση των τεχνικών εντοπισμού λέξεων βασίζεται στην φύση της ερώτησης: εάν είναι εικόνα ονομάζεται Query-by-Example (QbE) (στην ουσία είναι πρόβλημα ταιριάσματος προτύπου), ενώ αν είναι συμβολοσειρά του προς αναζήτηση κειμένου ονομάζεται Query-by-String (QbS). Μια άλλη κατηγοριοποίηση που θα μας απασχολήσει είναι εάν έχουν χρησιμοποιηθεί ή όχι τεχνικές μηχανικής μάθησης, ή αλλιώς αν έχουμε κάποιο εκπαιδευμένο μοντέλο ή χρησιμοποιούμε κάποια κλασσική τεχνική Όρασης Υπολογιστών για την εξαγωγή χαρακτηριστικών και την μετέπειτα σύγκριση τους. Να σημειώσουμε ότι για την εφαρμογή του QbS σεναρίου απαιτείται η εκμάθηση κάποιου μοντέλου αναγνώρισης χαρακτήρων, όπως είναι λογικό. Αρχικά, θα παρουσιάσουμε τις τεχνικές που αναπτύξαμε για το QbE σενάριο βασιζόμενοι στον POG περιγραφητή και στην συνέχεια θα αναλύσουμε σε βάθος σύγχρονες τεχνικές βαθιάς μάθησης και για τα τα δύο σενάριο ερώτησης (QbS και QbE).

Κατάτμηση και Αναπαράσταση Λέξεων

Η επέκταση των περιγραφητών POG στο πρόβλημα του εντοπισμού μπορεί να επιτευχθεί με την άμεση εφαρμογή τους στις εικόνες λέξεων. Βέβαια, υπάρχουν δυο σημαντικά προβλήματα που πρέπει να αντιμετωπίσουμε για τα καλύτερα δυνατά αποτελέσματα: 1) οι περιγραφήτες δεν είναι ανεξάρτητοι από μετακινήσεις και περιστροφές και 2) σε εικόνες λέξεων υπάρχει πολύ μεγαλύτερη πληροφορία στον οριζόντιο άξονα (πολλοί χαρακτήρες) και οι προβολική προσεγγίσει των POGs υποθέτει ισοκατανεμημένη πληροφορία. Το πρώτο πρόβλημα το αντιμετωπίζουμε βρίσκοντας την κύρια ζώνη της λέξης, αγνοώντας προεκτάσεις χαρακτήρων (acenders/descenders). Για το σκοπό αυτό εφαρμόζουμε μια τεχνική εύρωστης παλινδρόμησης που βρίσκει την βέλτιστη ευθεία αγνοώντας πιθανές 'εκτροπές' τιμές (outliers). Με τον τρόπο αυτό, εντοπίζουμε και διορθώνουμε περιστροφές (κλίση της ευθείας) και μετατοπίσεις (θέση της ευθείας), όπως φαίνεται στο Σχήμα 3.



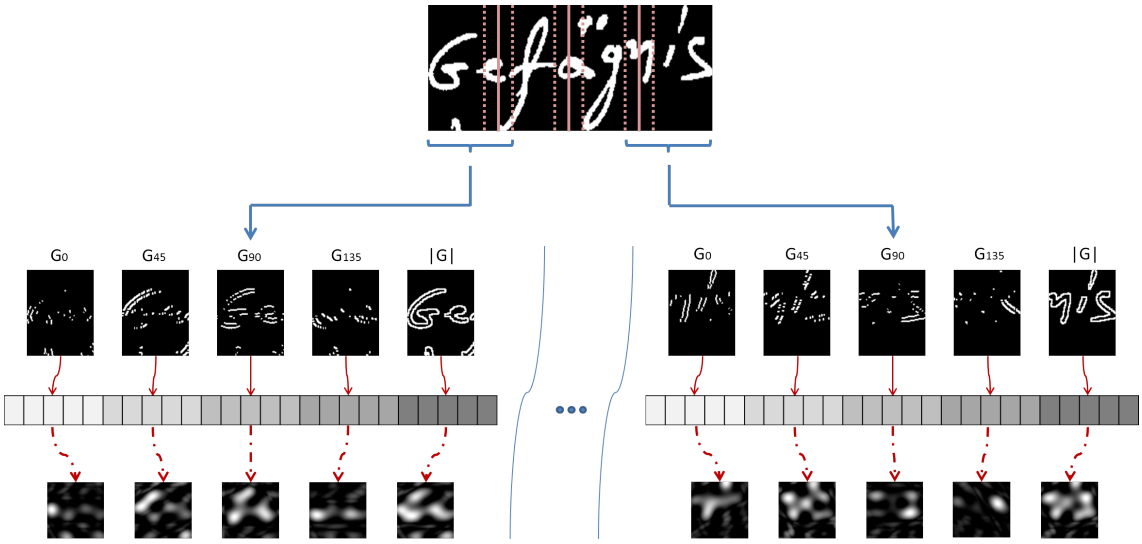
(α′) (β′) (γ′)

(δ′) (ε′) (ϝ′)

Σχήμα 3: Κανονικοποίηση εικόνων λέξεων, όπου (α),(β),(γ) αναπαριστούν τις αρχικές εικόνες μαζί με την ευρεθείσα κύρια ζώνη ως ευθείες, ενώ (δ),(ε),(ζ) αναπαριστούν τις κανονικοποιημένες εικόνες

Στην συνέχεια, καταπιανόμαστε με το δεύτερο πρόβλημα, αυτό της ανισοκατανομής της ουσιαστικής πληροφορίας μεταξύ των δύο αξόνων. Το πρόβλημα αυτό το αντιμετωπίζουμε με την οριζόντια κατάτμηση της εικόνας σε ζώνες (zoning). Με τον τρόπο αυτό, κάθε

ζώνη/τμήμα θα περιέχει πιο ισοκατανεμημένη πληροφορία για την άμεση εξαγωγή των POG διανυσμάτων. Τα τμήματα που δημιουργούνται είναι επικαλυπτόμενα και ο αριθμός των τμημάτων επιλέγεται να είναι τέσσερα, ως η μέση τιμή χαρακτήρων ανά λέξη στις βάσεις που χρησιμοποιούμε. Ο τελικός περιγραφητής της εικόνας λέξης θα είναι η συνένωση των περιγραφητών για τα επιμέρους τμήματα, όπως φαίνεται στο Σχήμα 4. Για την βελτιστοποίηση της απόδοσης των POGs, χρησιμοποιούμε παραλλαγή τους που εφαρμόζεται σε γκρίζες εικόνες (αντί για δυαδικές), όπου οι εικόνες κατεύθυνσης δημιουργούνται σύμφωνα με το πλάτος της κατευθυντικής παραγώγου σε κάθε pixel (soft binning).



Σχήμα 4: Επισκόπηση της εξαγωγής POGs διανυσμάτων για κατατμημένη εικόνα σε 4 επικαλυπτόμενα τμήματα. Συμπεριλαμβάνονται και οι οπτικοποιήσεις των κωδικοποιημένων εικόνων, όπως είδαμε σε προηγούμενη ενότητα.

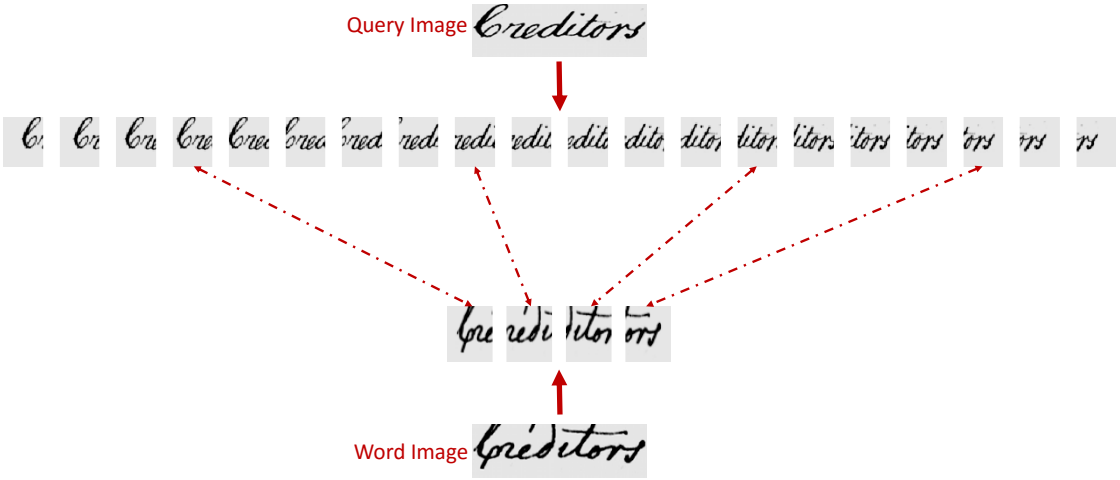
Σύγκριση Λέξεων ως Ακολουθία Περιγραφητών

Όπως προαναφέραμε, για την βέλτιστη εφαρμογή τυπικών περιγραφητών εμφάνισης (όπως τα προτεινόμενα POGs), πρέπει να αντιμετωπίσουμε πληθώρα ποικιλομορφιών (που σχετίζονται με τον τρόπο γραφής). Πάρα την αρχική προσέγγιση που προτείναμε με την κανονικοποίηση και κατάτμηση των εικόνων, η ανάλυση των λαθών έδειξε ότι έχουμε σημαντικά περιθώρια βελτίωσης. Προς αυτή την κατεύθυνση ορίζουμε το ταίριασμα λέξεων ως ευθυγράμμιση (alignement) ακολουθιών τοπικών περιγραφητών, εξαχθέντων σε οριζόντιες ζώνες. Το πλεονέκτημα της διαχείρισης ακολουθιών έγκειται στο γεγονός ότι μπορούμε να αντιμετωπίσουμε μετατοπίσεις εσωτερικά τις λέξεις, δηλαδή μεταξύ των χαρακτήρων που την αποτελούν. Παρόμοιες προσεγγίσεις έχουν εμφανιστεί συχνά στην βιβλιογραφία (συνήθως βασιζόμενες σε Dynamic Time Warping (DTW), π.χ. [140]), αλλά είναι πολύ αργές στην πράξη. Το μειονέκτημα αυτών των τεχνικών είναι το μεγάλο υπολογιστικό τους κόστος σε σύγκριση με τον υπολογισμό μιας απόστασης μεταξύ ολιστικών αναπαραστάσεων. Έχοντας σαν στόχο την ελαχιστοποίηση των υπολογιστικών πόρων, τόσο χρονικών όσο και χωρικών, εφαρμόζουμε μια πιο σύνθετη λογική και αναπτύσσουμε έναν αλγόριθμο ευθυγράμμισης/ταιριάσματος, κατάλληλο για την παρούσα μοντελοποίηση. Διακρίνουμε τρία βήματα:

1. Κανονικοποίηση των εικόνων. Υλοποιείται σύμφωνα με την προαναφερθείσα λογική και αντιμετωπίζει προβλήματα περιστροφής και κατακόρυφης μετατόπισης.

2. Εξαγωγή ακολουθιών από περιγραφητές ύστερα από οριζόντια κατάτμηση της εικόνας. Διακρίνουμε δύο περιπτώσεις: εικόνα βάσης, όπου χωρίζεται σε n_w ($n_w = 4$) τμήματα και εικόνα ερωτήματος (query), όπου διαλέγουμε έναν πολύ πιο πυκνό διαμερισμό ίσο με $n_w \times n_d$ ($n_d = 5$). Η άνιση κατάτμηση έχει δυο σκοπούς: πρώτον στην επιτάχυνση της διαδικασίας ταιριάσματος, όπως θα δούμε στην συνέχεια, και δεύτερον στην μείωση των αποθηκευτικών απαιτήσεων της μεθόδου, αφού η λεπτομερής κατάτμηση θα συμβαίνει κατά την υποβολή του ερωτήματος και μόνο στην εικόνα ερωτήματος. Στο Σχήμα 5 απεικονίζουμε ένα παράδειγμα της παραπάνω λογικής.

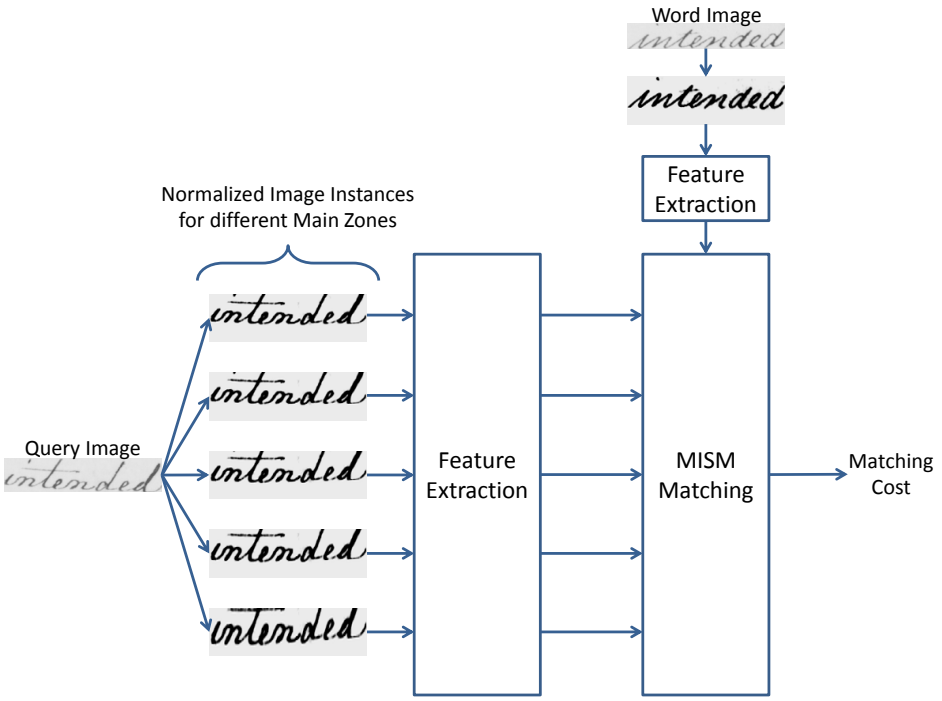
3. Έχοντας τις 2 ακολουθίες διαφορετικού μεγέθους, θέλουμε να βρούμε μια μοναδική αντιστοίχιση κάθε τμήματος από την εικόνα βάσης με ένα τμήμα από την πυκνή ακολουθία της εικόνας ερωτήματος, όπως φαίνεται και στο Σχήμα 5. Για την σωστή αντιστοίχηση υπάρχουν περιορισμοί: να υπάρχει ακολουθιακή αντιστοίχηση, δηλαδή δεν επιτρέπεται να ταιριάξουμε τμήματα εκτός σειράς (π.χ. το 1ο με το 10ο και το 2ο με το 5ο), ενώ θέλουμε τα ταιριάσματα να ισαπέχουν, όσο είναι αυτό δυνατόν. Τα παραπάνω μπορούν να μοντελοποιηθούν αποδοτικά σε ένα πρόβλημα δυναμικού προγραμματισμού και τον αλγόριθμο επίλυσής του τον ονομάζουμε Selective Matching (SM) (με πολυπλοκότητα $O(n_w^2 n_d^2)$).



Σχήμα 5: Απεικόνιση της κατάτμησης και της διαδικασίας ταιριάσματος για την προτεινόμενη μεθοδολογία.

Για να επιταχύνουμε και άλλο την διαδικασία, επιλέγουμε μια στρατηγική δυο βημάτων για την σωστή ταξινόμηση των σχετικών εικόνων: 1) χρήση ολιστικών αναπαραστάσεων POG σε όλη την εικόνα και πρότερη ταξινόμηση με βάση τις αναπαραστάσεις αυτές (πολύ γρήγορη διαδικασία - αρκεί η ευκλείδεια απόσταση μεταξύ τους) 2) εφαρμογή της ακολουθιακής λογικής σε ένα μικρό ποσοστό (10%) των ήδη ταξινομημένων εικόνων για μια πιο ακριβής επαναταξινόμηση.

Τέλος, επανεξετάσαμε και το βήμα της κανονικοποίησης, όπου μια λάθος εκτίμηση θα επηρεάσει σοβαρά όλη την προτεινόμενη μεθοδολογία (π.χ. λάθος εκτίμηση της περιστροφής). Για τον λόγο αυτό, προτείνουμε την εξαγωγή πολλών εκδόσεων κανονικοποίησης της αρχικής εικόνας, μεταβάλλοντας μια υπερ-παράμετρο του βήματος της κανονικοποίησης. Αυτή η λειτουργία πολλών εκδόσεων, εφαρμόζεται μόνο στην εικόνα ερωτήματος για τους ίδιους λόγους που προαναφέραμε για την αραιότερη κατάτμηση (μείωση χωρικών απαιτήσεων). Για την εφαρμογή του προτεινόμενου αλγόριθμου ταιριάσματος για το πρόβλημα αυτό, δημιουργήσαμε μια παραλλαγή του όπου έχει γραμμική εξάρτηση ως προς το πλήθος των διαφορετικών εκδόσεων



Σχήμα 6: Σύνοψη της μεθόδου Multi-Instance Selective Matching.

(Multi-Instance Selective Matching (MISM)). Η σύνοψη της μεθόδου αυτής απεικονίζεται στο Σχήμα 6.

	Bentham14		Modern14	
Method	**P@5**	**MAP**	**P@5**	**MAP**
Kovalchuk et al. [99]	73.8	52.4	58.8	33.8
⋆ Almazan et al.** [9]	72.4	51.3	70.6	**52.3**
Howe [86]	71.8	46.2	56.9	27.8
Zagoris et al. [189]	52.5	34.1	-	-
Zagoris et al. [190]	62.3	39.3	-	-
Aldavert et al. [6]	62.9	46.5	61.9	38.9
Sfikas et al. [158]	76.4	53.6	56.0	32.1
Retsinas et al. [144]	77.1	57.7	61.3	35.5
Zagoris et al. [191]	78.8	60.0	-	-
Holistic-mPOG+Eucl	77.5	60.4	52.3	29.7
PSeq-mPOG+SM	85.1	70.2	71.4	46.8
PSeq-mPOG+MISM	**85.5**	**71.1**	**73.5**	49.1

Πίνακας 2: Πειραματικά αποτελέσματα για τις βάσεις Bentham και Modern του διαγωνισμού ICFHR 2014 [134]. (**): μέθοδος μηχανικής μάθησης εκπαιδευμένη σε άλλη βάση (⋆) : νικητής διαγωνισμού.

Η πειραματική αξιολόγηση της μεθόδου και των παραλλαγών της συνοψίζεται στους Πίνακες 2 και 3, όπου συγκρινόμαστε με πληθώρα σύγχρονων τεχνικών. Αναλυτικά, χρησιμοποιούμε τρεις παραλλαγές: 1)Holistic-mPOG+Eucl - εξαγωγή ενός ενιαιου διανύσματος χαρακτηριστικών ανά εικόνα και σύγκρισή τους με την ευκλείδεια απόσταση, 2)PSeq-mPOG+SM - η προτεινόμενη μεθοδολογία σύγκριση ακολουθιών και 3)PSeq-mPOG+MISM - η επαυξημένη παραλλαγή της προτεινόμενης μεθοδολογίας για διαφορετικές κανονικοποιήσεις. Από

τα πειραματικά αποτελέσματα μπορούμε να εξάγουμε δυο κύρια συμπεράσματα. Πρώτον, η προτεινομένη ακολουθιακή μέθοδος (και η παραλλαγή της που δίνει σταθερή αύξηση στην απόδοση) ξεπερνάει κατά πολύ όλες τις άλλες τεχνικές που δεν εφαρμόζουν τεχνικές μηχανικής μάθησης. Δεύτερον, εμφανίζουμε αποτελέσματα πολύ κοντινά σε ορισμένες τεχνικές που χρησιμοποιούν εκπαιδευμένα μοντέλα, αλλά εν γένει δεν μπορούμε να ανταγωνιστούμε τέτοιες τεχνικές, καθώς μπορούν να μάθουν την τεράστια ποικιλία γραφικών χαρακτήρων που είναι πολύ δύσκολο να μοντελοποιηθεί με απλές τεχνικές Όρασης Υπολογιστών. Για τον λόγο αυτό στις επόμενες ενότητες θα μελετήσουμε τεχνικές εκπαίδευσης, είτε μη επιβλεπόμενες είτε επιβλεπόμενες, με σκοπό να βελτιστοποιήσουμε την απόδοση των συστημάτων μας.

Method	**P@5**	**MAP**
⋆ PRG [137]	46.0	42.4
CVC [137]	34.3	30.0
Zagoris et al. [189]	22.4	19.3
Zagoris et al. [190]	26.8	21.7
Almazan et al.** [9]	41.7	36.3
Sfikas et al. [158]	47.0	41.5
Retsinas et al. [144]	48.7	44.5
Zagoris et al. [191]	50.1	44.0
Holistic-mPOG +Eucl	48.1	44.2
PSeq-mPOG+SM	59.5	56.4
PSeq-mPOG+MISM	**61.6**	**58.4**

Method	**Botany16**	**Konzils16**
CVCDAG⋆ [135]	75.8	77.9
⋆ PRG⋆ [135]	**89.7**	**96.1**
QTOB⋆ [135]	55.0	82.2
TAU [135]	50.6	71.1
Retsinas et al. [144]	46.7	56.5
Sfikas et al. [158]	46.5	59.9
Holistic-mPOG+Eucl	53.2	64.2
PSeq-mPOG+SM	57.0	71.1
PSeq-mPOG+MISM	58.3	76.2

Πίνακας 3: Πειραματικά αποτελέσματα για τις βάσεις Bentham του διαγωνισμού ICDAR 2015 [137] (αριστερά) και τις βάσεις Botany και Konzilsprotokolle του διαγωνισμού ICFHR 2016 [135](δεξιά - μόνο MAP μετρική). (**): μέθοδος μηχανικής μάθησης εκπαιδευμένη σε άλλη βάση (⋆) : νικητής διαγωνισμού.

Μη-γραμμική Μείωση Διάστασης των Αναπαραστάσεων Λέξεων

Προτού στραφούμε σε τεχνικές βαθιάς μάθησης, μελετήσαμε κατά πόσο σύγχρονες τεχνικές μη-επιβλεπόμενης μάθησης, μπορούν να συμπιέσουν αναπαραστάσεις λέξεων αλλά και να συνεισφέρουν στην καλύτερη κατανόηση της κατανομής τους στον χώρο, μέσω της απλοποιημένης προβολής τους σε χώρους πολύ μικρών διαστάσεων. Ιδανικά, θα θέλαμε τέτοιες αναπαραστάσεις μικρής διάστασης να βελτιώσουν και την απόδοση του εντοπισμού λέξεων, καθώς η ευκλείδειας απόσταση σε χώρους υψηλής διάστασης δεν θεωρείται κατάλληλη (curse of dimensionality). Για τον σκοπό αυτό χρησιμοποιήσαμε την διαδεδομένη τεχνική μείωσης διάστασης t-distributed Stochastic Neighbor Embedding (t-SNE) [182] ,που ανήκει στην κατηγορία των manifold embedding τεχνικών. Η t-SNE μέθοδος είναι ιδιαίτερα δημοφιλής για την οπτικοποιήση κατανομών, εξαιτίας της ευρωστίας που επιδεικνύει ακόμα και σε πολύ μικρές διαστάσεις (π.χ. 2). Στην ουσία προσπαθεί να ελαχιστοποιήσει την απόκλιση της ομοιότητας (ανά ζεύγη) των σημείων του αρχικό χώρου δεδομένων με αυτών του μειωμένου. Συνεπώς, ορίζεται ως η ελαχιστοποίηση της Kullback-Leibler απόκλισης:

$$C(\mathbf{Y}) = KL(P||Q(\mathbf{Y})) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

όπου p_{ij} και q_{ij} συμβολίζουν τις ομοιότητες για τα αρχικά σημεία και τα τελικά σημεία, αντίστοιχα:

$$
\begin{aligned}
p_{j|i} &= \frac{\exp(-d(x_i, x_j)^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-d(x_i, x_k)^2/2\sigma_i^2)}, \quad p_{i|i} = 0 \\
p_{ij} &= \frac{p_{j|i} + p_{i|j}}{2N} \\
q_{ij} &= \frac{(1 + \|y_i - y_j\|^2)^{-1}}{Z}, \quad q_{ii} = 0 \\
Z &= \sum_k \sum_{l\neq k} (1 + \|y_k - y_l\|^2)^{-1}
\end{aligned}
$$

Η ελαχιστοποίηση επιτυγχάνεται με κάποιο επαναληπτικό αλγόριθμο gradient descent.

Παρόλο που η εφαρμογή μια τέτοιας τεχνικής μείωσης διάστασης είναι απλοϊκή στα πλαίσια των αναπαραστάσεων λέξεων, υπάρχει ένα πολύ σημαντικό πρόβλημα: Η εκπαίδευση των νέων χαμηλοδιάστατων διανυσμάτων γίνεται πάνω στα δεδομένα που έχουμε ήδη, τις εικόνες της βάσης, και η εφαρμογή της μεθόδου σε μια νέα εικόνα ερώτηση (query image) δεν είναι προφανής (η μέθοδος δεν είναι παραμετρική).

Για τον σκοπό αυτό προτείναμε μια νέα μέθοδο ενσωμάτωσης άγνωστων διανυσμάτων (Out-of-Vocabulary (OOV)) στον χαμηλοδιάστατο χώρο του t-SNE με μικρό υπολογιστικό κόστος. Η προτεινόμενη μέθοδος βασίζεται στον ορισμό του t-SNE, και συγκεκριμένα αναζητάμε το y διάνυσμα δεδομένου ενός διανύσματος εισόδου x. Θεωρούμε ήδη υπολογισμένα τα ζεύγη αρχικού/τελικού χώρου $\{x_i, y_i\}$. Υπολογίζουμε το y, με χρήση gradient descent, ως εξής:

$$
\begin{aligned}
y^{t+1} &= y^t - \alpha \frac{\partial C(y^t)}{\partial y^t} \\
\frac{\partial C(y)}{\partial y} &= 4 \sum_i [p(x, x_i) - q(y, y_i)] s(y, y_i)(y - y_i)
\end{aligned}
$$

όπου:

$$
\begin{aligned}
p(x|x_i) &= \frac{\exp(-\|x - x_i\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_k - x_i\|^2/2\sigma_i^2)} \\
p(x, x_i) &= \frac{p(x|x_i) + p(x_i|x)}{2N}, \quad p(x_i, x_i) = 0 \\
s(y, y_i) &= (1 + \|y - y_i\|^2)^{-1} \\
q(y, y_i) &= s(y, y_i) / \sum_k \sum_{l\neq k} s(y_k, y_l)
\end{aligned}
$$

Το βασικό πρόβλημα της παραπάνω προσέγγισης είναι ο ρυθμός σύγκλισης που ελέγχεται από την υπερ-παράμετρο α. Για τον λόγο αυτό εισάγουμε έναν προσαρμοστικό βήμα α ως εξής:

$$
\alpha(y^t) = \left[4 \sum_i p(x, x_i) s(y^t, y_i)\right]^{-1} \geq 0
$$

και άρα ο τελικός επαναληπτικός αλγόριθμος βασίζεται στην σχέση:

$$
y^{t+1} = y^t - \frac{\sum_i [p(x, x_i) - q(y^t, y_i)] s(y^t, y_i)(y^t - y_i)}{\sum_i p(x, x_i) s(y^t, y_i)}
$$

Το βήμα αυτό αποδεικνύεται βέλτιστο (line search strategy) εάν αγνοήσουμε την επίδραση του όρου $\sum_i q(y^t, y_i)s(y^t, y_i)(y - y_i)$.

Έστω d_x και d_y η αρχική και τελική διάσταση, N το πλήθος των σημείων της βάσης και K οι επαναλήψεις για να συγκλίνει η επαναληπτική μέθοδος. Η υπολογιστική πολυπλοκότητα της προτεινόμενης μεθόδου είναι $O(N(d_x + kd_y))$ και εφόσον απαιτούνται ελάχιστες επαναλήψεις ($\tilde{5}$) και επιλέγουμε πολύ μικρές διαστάσεις (2-5), το έξτρα κόστος, σε σχέση με τον υπολογισμό της ευκλείδειας απόστασης των αρχικών σημείων, είναι αμελητέο.

Ως προς το πειραματικό μέρος, θεωρούμε τρεις τεχνικές εξαγωγής ενός ενιαίου διανύσματος χαρακτηριστικών από εικόνες: 1) BoVW[6] , 2)POG[144] και 3)ZAH[158]. Για σύγκριση παραθέτουμε τις εξής εναλλακτικές ενσωμάτωσης της εικόνας ερωτήματος: 1) Closed Form Solution (CFS) $y^\star = \frac{\sum_i p(x,x_i)y_i}{\sum_i p(x,x_i)}$, 2) την παραμετροποίηση του μειωμένου χώρου σύμφωνα με την μέθοδο [57] (διακρίνουμε 2 περιπτώσεις, ανάλογα με το ποσοστό των σημείων που χρησιμοποιούνται: 90% ή 100%). Επίσης αναφέρουμε το ποσοστό επιτυχίας για απευθείας σύγκριση των δεδομένων στον αρχικό χώρο (No Embedding). Τα αποτελέσματα συνοψίζονται στον Πίνακα 4, όπου χρησιμοποιούμε μόνο 3 διαστάσεις στον τελικό χώρο. Παρατηρούμε ότι όχι μόνο η προτεινόμενη μέθοδος υπερτερεί σημαντικά των εναλλακτικών, αλλά και ότι η απόδοση του συστήματος εντοπισμού λέξεων βελτιώνεται σε σχέση με τον αρχικό χώρο, ακόμα και αν έχουμε μόλις 3 διαστάσεις.

		GW20[6]		**Bentham14**[134]		**Modern14**[134]	
Descriptor	**OOS approach**	MAP	P@5	MAP	P@5	MAP	P@5
BoVW	No Embedding	72.30	91.59	55.29	74.50	28.93	50.60
	CFS	80.10	82.22	48.42	49.50	19.67	23.00
	Parametric-90	82.59	88.58	35.93	32.25	12.81	13.00
	Parametric-100	85.13	92.30	38.13	35.12	13.62	15.33
	Proposed	**85.18**	**92.42**	**63.57**	**78.00**	**33.93**	**53.87**
POG	No Embedding	62.49	**85.85**	66.01	**82.25**	36.83	61.80
	CFS	68.63	74.51	56.09	53.69	29.81	28.60
	Parametric-90	70.38	79.82	55.67	57.12	35.21	47.73
	Parametric-100	74.15	85.04	60.77	64.31	43.64	57.13
	Proposed	**74.19**	85.14	**70.43**	80.94	**48.21**	**65.80**
ZAH	No Embedding	61.19	86.40	**53.49**	75.69	33.29	56.33
	CFS	70.69	76.45	37.41	44.06	24.65	30.07
	Parametric-90	74.10	84.44	28.51	33.56	30.30	43.33
	Parametric-100	76.77	89.13	34.95	42.62	33.08	47.87
	Proposed	**76.82**	**89.23**	53.17	**76.75**	**38.84**	**59.93**

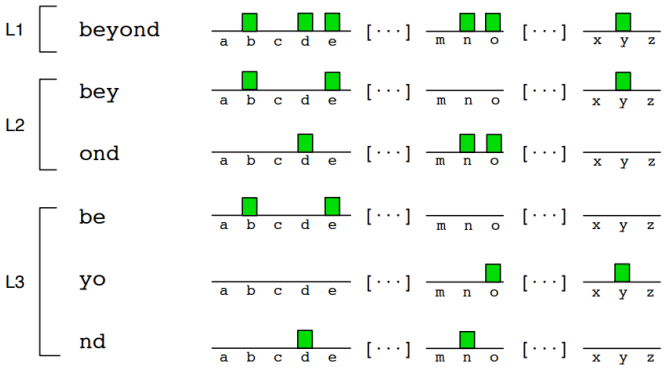
Πίνακας 4: Πειραματικά αποτελέσματα (με χρήση MAP και P@5) για $d_y = 3$

Εντοπισμός Λέξεων και Βαθιά Μάθηση

Η χρήση τεχνικών βαθιάς μάθησης, έχει φέρει την επανάσταση στο τομέα της όρασης υπολογιστών, παράγοντας αποτελέσματα που υπερτερούν σημαντικά των τυπικών μεθόδων εξαγωγής χαρακτηριστικών. Ακολουθώντας την τάση αυτή, η χρήση των βαθιών νευρωνικών δικτύων στο τομέα της επεξεργασίας και αναγνώρισης κειμένων έχει συνεισφέρει στην δημιουργία πολύ αποδοτικών και αξιόπιστων συστημάτων. Στην ενότητα αυτή, θα περιγράψουμε την αρχική μελέτη σε τέτοιες βαθιές αρχιτεκτονικές δικτύων και την μετέπειτα συνεισφορά μας στον τομέα αυτό με καινοτόμες ιδέες εφαρμογής τέτοιων αρχιτεκτονικών.

Εξερεύνηση Βαθιών Νευρωνικών Δικτύων για το Πρόβλημα του Εντοπισμού Λέξεων

Για το πρόβλημα του εντοπισμού λέξεων μπορούμε να διακρίνουμε δυο πολύ σημαντικές δουλειές που έχουν διαμορφώσει τις σύγχρονες προσεγγίσεις του προβλήματος. Η πρώτη σχετίζεται με την ανάθεση ενός διανύσματος ιδιοτήτων (attribute vector) ως στόχο εκπαίδευσης για κάθε λέξη [9]. Η ύπαρξη ενός συγκεκριμένου χαρακτήρα στην λέξη ορίζεται ως μια δυαδική ιδιότητα και συνεπώς δημιουργείται ένα δυαδικό ιστόγραμμα εμφάνισης χαρακτήρων. Για την ενσωμάτωση χωρικής πληροφορίας (που βρίσκεται ο κάθε χαρακτήρας), η λέξη χωρίζεται σε επιμέρους τμήματα ίδιου μήκους ανά επίπεδα (πυραμιδωτή στρατηγική), όπως φαίνεται στο Σχήμα 7. Το τελικό διάνυσμα ιδιοτήτων είναι η συνένωση των επιμέρους ιστογραμμάτων και ονομάζεται Pyramid Histogram of Characters (PHOC). Αυτή η προσέγγιση είναι πολύ πιο εύρωστη από την θεώρηση ότι κάθε διαφορετική λέξη αντιστοιχεί σε μια νέα κλάση, καθώς πολλές λέξεις μοιράζονται κοινές 'ιδιότητες'. Με τον τρόπο αυτό μπορούμε να αναπαραστήσουμε λέξεις, τις οποίες δεν έχουμε στο σύνολο εκπαίδευσης.



Σχήμα 7: Πυραμιδωτή στρατηγική (3 επίπεδα) για την εξαγωγή του PHOC ιστογράμματος [9].

Η παραπάνω δυαδική αναπαράσταση κειμενικής πληροφορίας συνδυάστηκε με μια τυπική αρχιτεκτονική VGG προσφέροντας πολύ καλύτερα αποτελέσματα από την τότε βιβλιογραφία. Η αρχιτεκτονική αυτή ονομάστηκε PHOCNet [166] και διαφοροποιήθηκε σε σχέση με την VGG αρχιτεκτονική σε δύο σημεία: υποστηρίζει δυαδικές αναπαραστάσεις στην έξοδό της (χρήση sigmoid) και υποστηρίζει εικόνες οποιοδήποτε μεγέθους στην είσοδό της σύμφωνα με την χρήση ενός πυραμιδωτού επιπέδου Spatial Pyramidal Pooling (SPP) [76], αμέσως μετά τα συνελικτικά επίπεδα.

Με βάση την αρχιτεκτονική αυτή, που θα είναι το σημείο αναφοράς για τεχνικές βαθιάς μάθησης στο πρόβλημα του εντοπισμού λέξεων στην παρούσα διατριβή, εξετάσαμε πληθώρα παραμέτρων και παραλλαγών που συνοψίζουμε ακολούθως.

- Επίδραση του μεγέθους της εικόνας (προκαθορισμένο ή αυθαίρετο μέγεθος).
- Αναγκαιότητα της υπολογιστικά ακριβής SPP δομής. Έλεγχος αποτελεσματικότητας μιας zoning δομής, αντί για πυραμιδωτή, με ενθαρρυντικά αποτελέσματα.
- Επιβεβαίωση αποτελεσματικότητας των τεχνικών δυναμικής επαύξησης δεδομένων που προσομοιώνουν μεταβολές σε εικόνες κειμένου (π.χ. αφινικοί μετ/σμοι). Επίδραση της ισοκατανομής των κλάσεων. Συγκεκριμένα, εξετάστηκε αν χρειάζεται να έχουμε ίδιο πλήθος από εικόνες της ίδιας λέξης.

- Χρήση εξόδων από ενδιάμεσα επίπεδα του νευρωνικού (πριν της τελικής εκτίμησης του PHOC) για εξαγωγή χαρακτηριστικών (deep features) και μελέτη της ικανότητας γενίκευσης τους σε άγνωστες βάσεις με πολύ θετικά αποτελέσματα. Ενίσχυση της προσέγγισης αυτής με χρήση διανυσμάτων μειωμένων διαστάσεων σύμφωνα με την λογική του t-SNE που περιγράψαμε στην προηγούμενη ενότητα.

- Χρήση πολλαπλών όμοιων νευρωνικών δικτύων (ensemble of nets) για την βελτίωση των αποτελεσμάτων. Κάθε φορά που εκπαιδεύουμε ένα νευρωνικό, συγκλίνουμε σε ένα διαφορετικό τοπικό βέλτιστο λόγω του πλήθους των παραμέτρων. Έτσι, κάθε τέτοια λύση έχει παρόμοια συνολική απόδοση, αλλά ευνοεί επιμέρους διαφορετικές κλάσεις κάθε φορά (π.χ. μαθαίνει καλύτερα τον χαρακτήρα 'α'). Αν συνδυάσουμε τέτοιες διαφορετικές λύσεις της ίδιας αρχιτεκτονικής, είναι δυνατόν να βελτιώσουμε την συνολική απόδοση. Παράλληλα, μελετήθηκαν τρόποι συνδυασμού των διαφορετικών δικτύων υπό το πρίσμα του προβλήματος εντοπισμού λέξεων.

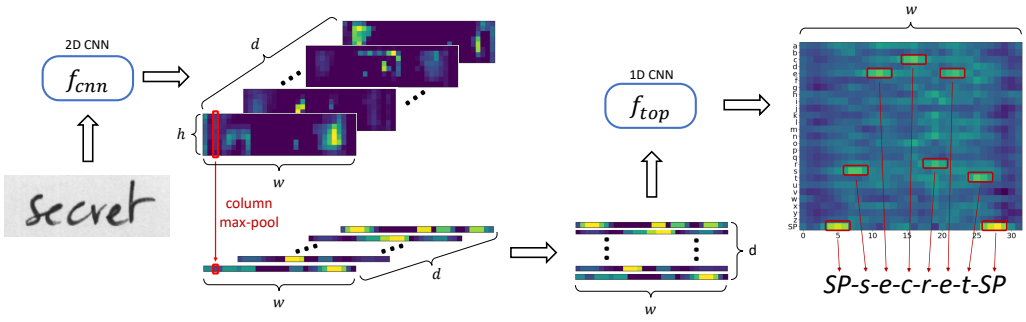
- Δημιουργία ενδιάμεσων αναπαραστάσεων ανεξάρτητων από την χωρική θέση του χαρακτήρα. Συγκεκριμένα, η χρήση του PHOC σαν τελικό στόχο εκπαίδευσης δημιουργεί χάρτες χαρακτηριστικών που συνδυάζουν την πληροφορία του ίδιου του χαρακτήρα μαζί με την χωρική του πληροφορία (π.χ. το 'α' του αριστερότερου τμήματος του επιπέδου 3 έχει άλλη αναπαράσταση από το 'α' σε κάποιο άλλο τμήμα ή επίπεδο). Αυτό το φαινόμενο που οφείλεται στην φύση του PHOC οδηγεί σε πλεονασμό πληροφορίας. Αντιμετωπίσαμε το πρόβλημα αυτό με ένα καινοτόμο επίπεδο εκπαίδευσης στην έξοδο του νευρωνικού και στην συνέχεια μελετήσαμε κατά πόσο μπορούμε να συμπιέσουμε τέτοιες ενδιάμεσες αναπαραστάσεις, μικραίνοντας σημαντικά το πλήθος των καναλιών του αντίστοιχου συνελικτικού επιπέδου (χρήση στρατηγικής επανεκπαίδευσης).

Κοινή Εφαρμογή Αναγνώρισης και Εντοπισμού Λέξεων

Μέχρι τώρα, στην βιβλιογραφία, η αναγνώριση κειμένου και ο εντοπισμός λέξεων αντιμετωπίζονται ως εντελώς διαφορετικά προβλήματα. Παρότι έχουν διαφορετικό τελικό στόχο (και εφαρμογή), είναι συγγενικά προβλήματα που έχουν έναν κοινό προσανατολισμό: την εξαγωγή διακριτικών χαρακτηριστικών, ικανών να διαχωρίσουν χαρακτήρες. Για τον λόγο αυτό, θεωρούμε ότι οι δυο αυτές διακριτές εφαρμογές μπορούν να συνδυαστούν ιδανικά σε κάποιο ενδιάμεσο επίπεδο μιας βαθιάς αρχιτεκτονικής, από την οπτική γωνία της εξαγωγής κοινών χαρακτηριστικών. Να σημειώσουμε ότι υπάρχουν τεχνικές που αντιμετωπίζουν το πρόβλημα του εντοπισμού λέξεων σε γραμμές κειμένου ως περιορισμένη αποκωδικοποίηση του αποτελέσματος αναγνώρισης (π.χ. σαν γράφος από πιθανούς χαρακτήρες και μεταβάσεις) ως προς την συμβολοσειρά προς αναζήτηση (π.χ. [52]). Βέβαια, τέτοιες προσεγγίσεις προϋποθέτουν πλήρη αναγνώριση πρώτα και δεν στοχεύουν στην χρήση ενδιάμεσων διανυσμάτων χαρακτηριστικών.

Η τυπική αρχιτεκτονική για αναγνώριση κειμένων υποθέτει RNNs (και συγκεκριμένα μια παραλλαγή τους, τα Long-Short Term Memory Netoworks - LSTMs) και στην έξοδο ένα επίπεδο Connectionist Temporal Classification (CTC)[63], το οποίο λειτουργεί σαν συνάρτηση ελαχιστοποίησης υπολογίζοντας τις πιθανές ευθυγραμμίσεις της ακολουθίας εξόδου του νευρωνικού (από πιθανότητες εμφάνισης των χαρακτήρων) και της ακολουθίας χαρακτήρων της συμβολοσειράς στόχου. Αρχικά, για την απλοποίηση της προαναφερθείσας τυπικής διάταξης αναγνώρισης, αποτελούμενη από ένα συνελικτικό τμήμα (CNN backbone), ένα ακολουθιακό τμήμα (LSTMs) και τέλος το επίπεδο CTC, αντικαθιστούμε τα υπολογιστικά απαιτητικά LSTMs με τα πλήρως παραλληλοποιήσιμα μονοδιάστατα CNNs, όπως φαίνεται στο Σχήμα 8.

Βέβαια, για να εξάγουμε μια ενιαία αναπαράσταση για κάθε λέξη επιστρατεύουμε την λογική των Sequence to Sequence (Seq2Seq) τεχνικών, που έχουν γνωρίσει μεγάλη επιτυ-



Σχήμα 8: Προτεινόμενη αρχιτεκτονική αναγνώρισης με χρήση 1-d CNNs και CTC.

χία σε εφαρμογές μετάφρασης, αλλά έχουν εφαρμοστέι ελάχιστα σε εφαρμογές αναγνώρισης κειμένων [168]. Τέτοιες τεχνικές χρησιμοποιούν δυο δομικά στοιχεία: τον κωδικοποιητή (Encoder), που μετατρέπει μια ακολουθία σε ένα ενιαίο διάνυσμα χαρακτηριστικών, και τον αποκωδικοποιητή (Decoder), που με την σειρά του μετατρέπει το ενιαίο διάνυσμα χαρακτηριστικών σε μια νέα ακολουθία. Λόγω της φύσης της προαναφερθείσας Seq2Seq προσέγγισης, η ενδιάμεση αναπαράσταση είναι ιδανική για την αναπαράσταση λέξεων και σε αντίθεση με κλασσικά διανύσματα ιδιοτήτων (π.χ. PHOC), η αναπαράσταση αυτή μπορεί να αποκωδικοποιήθει πλήρως σε μια συμβολοσειρά. Συνεπώς μπορούμε να εφαρμόσουμε τέτοιες αναπαραστάσεις και για αναγνώριση και για εντοπισμό λέξεων.

Ακολουθούν εν συντομία σημαντικές πτυχές της παραπάνω προσέγγισης. Η αρχιτεκτονική αυτή περιγράφεται στο Σχήμα 9 και την ονομάσαμε WSRNet.

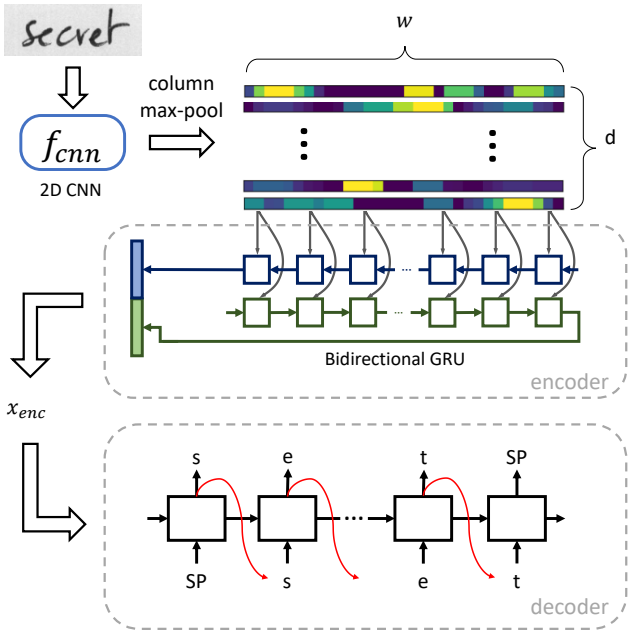
- Οι Encoder και Decoder υλοποιούνται με GRU μονάδες (μια υπολογιστικά αποδοτικότερη παραλλαγή των LSTMs). Δεν χρησιμοποιούμε Attention κλάδο στην παρούσα Seq2Seq προσέγγιση, καθώς υπονομεύει την ικανότητα της ενδιάμεσης αναπαράστασης.

- Εκπαιδεύουμε μαζί (multitask loss) τα υποσυστήματα των CTC και Seq2Seq, τα οποία μοιράζονται τα αρχικά συνελικτικά επίπεδα. Ο κλάδος CTC έχει βοηθητικό ρόλο για την επιτάχυνση της σύγκλισης του όλου συστήματος, εξάγοντας αποτελεσματικά χαρακτηριστικά στην έξοδο του CNN σε λίγες μόνο εποχές.

- Η εφαρμογή αναγνώρισης ή εντοπισμού λέξεων για το QbE σενάριο είναι προφανής με βάση την προτεινόμενη αρχιτεκτονική. Εντούτοις, η εφαρμογή του QbS σεναρίου δεν είναι εφικτή με το παρών σύστημα. Για τον λόγο αυτό προσθέτουμε έναν ακόμα κωδικοποιητή (Character Encoder), που μετατρέπει την τελική συμβολοσειρά στόχο στην ενδιάμεση αναπαράσταση. Ο Character Encoder μαζί με τον Decoder του Seq2Seq τμήματος, ορίζουν έναν Autoencoder, αφού η είσοδος και η έξοδος-στόχος είναι η ίδια συμβολοσειρά. Για την περαιτέρω βελτίωση του Autoencoder μπορούμε να εκπαιδευσούμε παράλληλα, συμβολοσειρές λέξεων που δεν ανήκουν στο σύνολο εκπαίδευσης (το οποίο είναι πολύ περιοριστικό, καθώς έχουμε ζεύγη εικόνων-συμβολοσειρών).

- Για την ενίσχυση της αναγνώρισης μπορούμε να χρησιμοποιήσουμε στατιστικά μοντέλα από εξωτερικό λεξικό (beam search variation).

- Μετατροπή της εξαχθείσας αναπαράστασης σε δυαδική μορφή. Η χρήση δυαδικών αναπαραστάσεων βελτιώνει κατά πολύ τις χρονικές (σύγκριση με bit-wise πράξεις: xnor & bincount) και χωρικές απαιτήσεις. Για τον σκοπό αυτό χρησιμοποιούμε την μέθοδο STE [18] κατά την εκπαίδευση.



Σχήμα 9: Αρχιτεκτονική βασισμένη στην Seq2Seq λογική.

- Χρήση forced alignment (FA) λογικής για τον εντοπισμό λέξεων. Αναμένουμε βελτίωση αποτελεσμάτων, αλλά με σημαντική αύξηση του χρόνου εντοπισμού.

Τα αποτελέσματα της προτεινόμενης μεθοδολογίας (μαζί με παραλλαγές της) παρουσιάζονται στον Πίνακα 5 και συγκρίνονται με σύγχρονες τεχνικές τόσο στην αναγνώριση λέξεων όσο και στον εντοπισμό τους (QbS & QbE). Για την αναγνώριση χρησιμοποιήθηκαν οι μετρικές CER και WER, ενώ για τον εντοπισμό η μετρική MAP. Η αποκωδικοποίηση (αναγνώριση) επιτελείται με δύο τρόπους: greedy και beam search. Η πειραματική αποτίμηση έγινε στην απαιτητική βάση IAM[123], αποτελούμενη από 657 διαφορετικούς γραφείς. Παρατηρούμε ότι παρά την απλότητα του συστήματος αναγνώρισης, έχουμε συγκρίσιμα αποτελέσματα με την βιβλιογραφία, ακόμα και με δυαδικές (binarized) αναπαραστάσεις, ενώ τα αποτελέσματα στον εντοπισμό λέξεων υπερτερούν σημαντικά σε σύγκριση με αντίστοιχες δουλειές στην βιβλιογραφία. Αξίζει να σημειώσουμε ότι το προτεινόμενο δίκτυο είναι αρκετά μικρότερο από την πλειονότητα των προταθέντων δικτύων για το πρόβλημα του εντοπισμού λέξεων.

Εντοπισμός Λέξεων σε Επίπεδο Γραμμής Κειμένου

Τέλος, μελετήσαμε το πρόβλημα εντοπισμού μια συγκεκριμένης λέξης-ερώτησης σε ολόκληρες γραμμές κειμένου. Σε αντίθεση με την επικρατούσα νόρμα για το πρόβλημα αυτό, που βασίζεται στην περιορισμένη αποκωδικοποίηση (constrained decoding) με βάση τεχνικές αναγνώρισης (π.χ. [52]), αντιμετωπίζουμε το πρόβλημα από την οπτική γωνία των εσωτερικών χαρακτηριστικών/αναπαραστάσεων.

Η βασική επιχειρηματολογία πίσω από την προτεινόμενη προσέγγιση είναι ότι η χρήση συνελικτικών επιπέδων δεν περιορίζεται σε εικόνες λέξεων, αλλά μπορούν να εμφανιστούν αυτούσια σε εικόνες γραμμών. Συνεπώς, μπορούμε να κάνουμε μια αναζήτηση προτύπου (template matching) στον χώρο της εξόδου των συνελικτικών δικτύων. Η λειτουργία αυτή χτίζεται πάνω σε μια παραλλαγή του PHOCNet[167]. Συγκεκριμένα, χρησιμοποιούμε 3 νευρωνικά δίκτυα:

Method	CER	WER
Sueiras et al. [168]	8.8	23.8
Wigington et al. [184]	6.07	19.07
Krishnan et al. [101]	6.34	16.19
Dutta et al. [44]	4.88	12.61
Proposed Models		
WSRNet+LM	5.0	13.9
WSRNet+LM (binarized)	5.7	15.4
WSRNet+LM+beam	4.8	13.6
WSRNet+LM+beam (binarized)	5.1	15.0

Method	QbE	QbS
Attributes+KCSR [9]	55.73	73.72
PHOCNet [166]	72.51	82.97
HWNet [102]	80.61	-
Triplet-CNN [185]	81.58	89.49
PHOCNet-TPP [167]	82.74	93.42
DeepEmbed [100]	84.25	91.58
Zoning Ensemble PHOCNet [148]	87.48	-
End2End Embed [101]	89.07	91.26
DeepEmbed [101]	90.38	94.04
Synth+DeepEmbed [101]	-	95.09
HWNetV2[103]	90.65	-
Proposed Models		
WSRNet+LM	92.05	95.67
WSRNet+LM (binarized)	91.31	93.69
WSRNet+LM+FA	-	96.27
WSRNet+LM+FA (binarized)	-	96.33

Πίνακας 5: Αποτίμηση για το πρόβλημα της αναγνώρισης (αριστερά) και εντοπισμού (δεξιά). Σε όλες της προτεινόμενες παραλλαγές έγινε χρήση εξωτερικού λεξικού για την εκπαίδευση του autoencoder (LM).

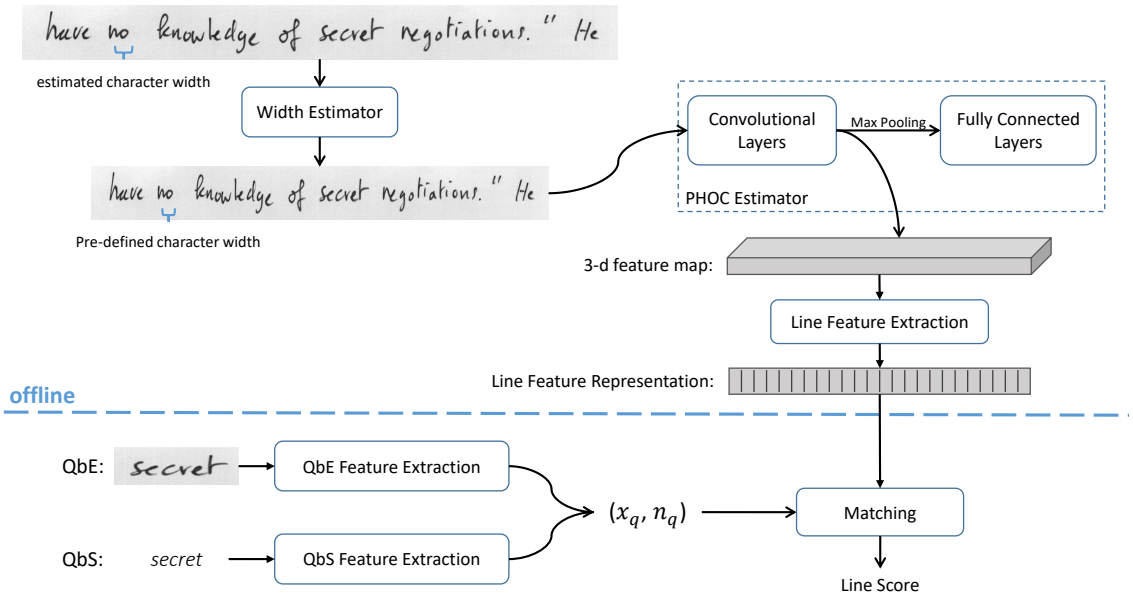
1. *Δίκτυο Εκτίμησης Πλάτους Χαρακτήρα:* Για την επιτάχυνση της αναζήτησης σε μια γραμμή, θέλουμε να κανονικοποιήσουμε το πλάτος τόσο των λέξεων όσο και των γραμμών έτσι ώστε να αντιστοιχούν σε ένα προκαθορισμένο πλάτος ανά χαρακτήρα. Έτσι, θα έχουμε χάρτες χαρακτηριστικών ανεξάρτητους από κλίμακα. Το συνελικτικό δίκτυο αυτό εκπαιδεύεται σε λέξεις έτσι ώστε να βρίσκει το μέσο πλάτος χαρακτήρα. Η ακρίβεια της εκτίμησης του πλάτους δεν είναι πρωτεύουσας σημασίας και αρκεί να μην υπάρχουν έντονες αποκλίσεις (π.χ. μεγαλύτερες από 10 pixels).

2. *Δίκτυο Εκτίμησης PHOC:* Χρησιμοποιούμε μια παραλλαγή του PHOCnet, αλλά εν τέλει αξιοποιούμε μόνο τα συνελικτικά επίπεδα που παράγουν τους χάρτες χαρακτηριστικών. Η εκπαίδευση γίνεται σε επίπεδο λέξεων. Τα τελικά πλήρως συνδεδεμένα (fully connected) επίπεδα χρησιμοποιούνται μόνο κατά την εκπαίδευση. Για την καλύτερη προσαρμογή σε εικόνες γραμμών κειμένου, δεν χρησιμοποιούμε απομονωμένες λέξεις, αλλά επιλέγουμε ένα ευρύτερο παράθυρο, που περιέχει και γειτονικές λέξεις (προσομοιώνουμε γειτονικό θόρυβο).

3. *Δίκτυο Κοινής Προβολής:* Η σύγκριση σε επίπεδο χαρακτηριστικών για το QbE σενάριο είναι προφανής, αλλά είναι αδύνατον να πραγματοποιήσουμε το QbS σενάριο. Για τον λόγο αυτό, έχουμε και ένα επιπλέον δίκτυο, αποτελούμενο μόνο από πλήρως συνδεδεμένα (fully connected) επίπεδα, που μετασχηματίζει μια PHOC αναπαράσταση στον χώρο χαρακτηριστικών. Βασική προϋπόθεση του μετασχηματισμού είναι να διατηρηθεί η κατανομή του αρχικού χώρου των PHOC, έτσι ώστε οι λέξεις να παραμείνουν διακριτές.

Μετά την εφαρμογή των συνελικτικών δικτύων σε γραμμές κειμένου, εξάγουμε μια ακολουθία χαρακτηριστικών για κάθε γραμμή, έπειτα από κατάτμηση σύμφωνα με προκαθορισμένο πλάτος (σχετίζεται με το πλάτος εκτίμησης του μέσου χαρακτήρα). Συνεπώς κάθε ερώτημα/query αντιστοιχεί σε συγκεκριμένο πλήθος τμημάτων της γραμμής, ανάλογα με το πλάτος του. Το προτεινόμενο σύστημα συνοψίζεται στο Σχήμα 10.

Η σύγκριση της προτεινόμενης μεθόδους με αντίστοιχες τεχνικές που εκτελούν εντοπισμό λέξεων σε επίπεδο γραμμής παρουσιάζεται στον Πίνακα 6 για την διαδεδομένη βάση IAM. Διακρίνουμε δυο σενάρια για την βάση αυτή, IAMDB1 ανδ IAMDB2, με βάση την ύπαρξη



Σχήμα 10: Σύνοψη της προτεινόμενης μεθοδολογίας για εντοπισμό λέξεων σε επίπεδο γραμμής. Το τελικό διάνυσμα χαρακτηριστικών του query συμβολίζεται με x_q και αντιστοιχεί σε n_q τμήματα των χαρακτηριστικών γραμμής.

λέξεων στο σύνολο αναζήτησης που δεν υπάρχουν στο σύνολο εκπαίδευσης. Αν υπάρχουν τέτοιες OOV λέξεις δεν μπορούμε να εφαρμόσουμε την QbE παραλλαγή, όπως στο IAMDB2. Επιπλέον διακρίνουμε και τοπική (local) και ολική ταξινόμηση (global) των ανακτηθέντων γραμμών (ταξινόμηση ανά query ή ταξινόμηση για όλα τα queries), όπως γίνεται συνήθως στην βιβλιογραφία για τέτοιες τεχνικές [50]. Τα αποτελέσματα αποδεικνύουν την αποτελεσματικότητα της μεθόδου, παρά την εντελώς διαφορετική προσέγγιση του προβλήματος, ενώ αξίζει να επισημάνουμε ότι είναι η πρώτη προσπάθεια στην βιβλιογραφία για QbE σε γραμμές κειμένου.

	IAMDB1		IAMDB2
methods	local	global	global
Fisher et al. [50]	68.92	47.75	-
Fisher et al. [48]	-	55.05	36.00
Toselli et al. [176]	-	-	61.03
Toselli et al. [173]	-	-	72.00
Frinken et al. [53]	-	-	76.00
proposed QbS	88.73	83.15	75.31
proposed QbE	84.25	73.16	-

Πίνακας 6: MAP (%) αποτίμηση σε δύο διαφορετικά σενάρια (IAMDB1 και IAMDB2) της βάσης IAM ($n_{quant} = 3$).

Συμπίεση Βαθιών Νευρωνικών Δικτύων

Όπως διαπιστώσαμε στην πράξη, τα σύγχρονα νευρωνικά δίκτυα αποτελούνται από εκατομμύρια παραμέτρους, δημιουργώντας υπολογιστικές απαιτήσεις πού υποστηρίζονται μόνο από σύγχρονο εξοπλισμό (GPUs), ενώ είναι αδύνατον να εφαρμοστούν σε μικρο-υπολογιστικές μονάδες, όπως σε κινητά κ.λ.π. Στην πράξη βέβαια ένα μικρό υποσύνολό τους προσφέρει στο τελικό αποτέλεσμα (εντούτοις η περίσσεια παραμέτρων βοηθάει στην γρήγορη εκπαίδευση). Για τον λόγο αυτό η επιστημονική κοινότητα ασχολείται εκτενώς με το πρόβλημα συμπίεσης των βαθιών νευρωνικών δικτύων. Προς αυτή την κατεύθυνση αναπτύξαμε δυο καινοτόμες προσεγγίσεις: 1) επαναχρησιμοποίηση βαρών 2) αραιοποίηση βαρών. Για την αξιολόγησή τους χρησιμοποιήσαμε τις εξαιρετικά διαδεδομένες βάσεις κατηγοριοποίησης εικόνων CIFAR [104] και ImageNet [154], έτσι ώστε να είναι εφικτή η σύγκριση με σχετικές τεχνικές της βιβλιογραφίας.

Συμπίεση Μοντέλων με Χρήση Κοινών Βαρών

Ο πλεονασμών των παραμέτρων μπορεί να αντιμετωπιστεί αποτελεσματικά με την επαναχρησιμοποίηση των ίδιων βαρών. Έτσι, δημιουργήσαμε ένα νέο συνελικτικό επίπεδο, το οποίο χωρίζει την είσοδο του επιπέδου σε τμήματα (κατά τον άξονα των καναλιών) και εφαρμόζει ακολουθιακά συνελίξεις σε κάθε τμήμα, με κοινά βάρη. Ονομάσαμε το επίπεδο αυτό Channel-wise Recurrent Convolution (CRC) και επί της ουσίας προσομοιώνει την λειτουργία των RNNs με χρήση συνελίξεων, όπως φαίνεται και από τον παρακάτω ορισμό:

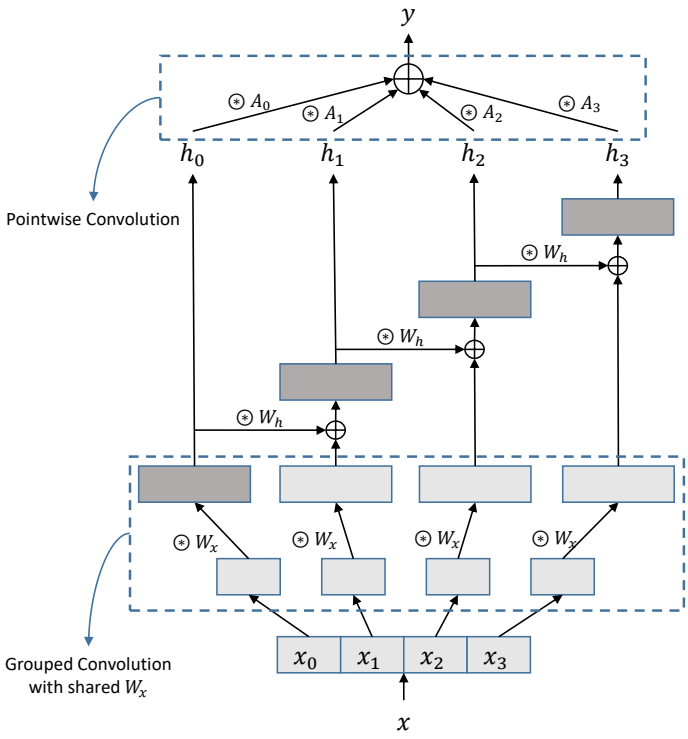
$$\mathbf{h}_i = \begin{cases} \sigma(\mathbf{x}_i * \mathbf{W_x} + \mathbf{b}), i = 0 \\ \sigma(\mathbf{x_i} * \mathbf{W_x} + \mathbf{h}_{i-1} * \mathbf{W_h} + \mathbf{b}), \forall i \in [1, d-1]. \end{cases}$$

,όπου $\{\mathbf{x_i}\}$ και $\{\mathbf{h_i}\}$ τα τμήματα εισόδου και εξόδου αντίστοιχα, $\mathbf{W_x}$ και $\mathbf{W_h}$ τα κοινά συνελικτικά βάρη και $\sigma()$ μια μη-γραμμική συνάρτηση ενεργοποίησης. Η έξοδος του CRC είναι η συνένωση των τμημάτων εξόδου $\{\mathbf{h_i}\}$.

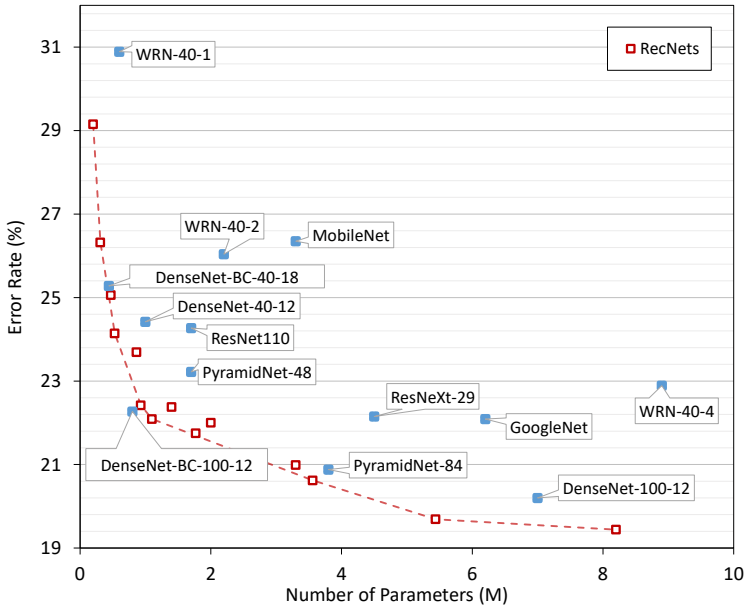
Με το προτεινόμενο επίπεδο, μπορούμε να προσομοιώσουμε πλατεία επίπεδα. Για υπολογιστική απλότητα, κάθε τέτοιο επίπεδο ακολουθείται από ένα απλό συνελικτικό επίπεδο συμπίεσης μοναδιαίου πυρήνα (1×1). Η χρήση αυτών των δύο επιπέδων απεικονίζεται στο Σχήμα 11. Επαναλαμβάνοντας αυτή τη δομή χτίζουμε μια νέα αρχιτεκτονική δικτύων, την οποία ονομάζουμε RecNets. Έχοντας μελετήσει την επίδραση των παραλλαγών της προτεινόμενης αρχιτεκτονικής (π.χ. χρήση διάφορων συναρτήσεων ενεργοποίησης), παρουσιάζουμε την αποτελεσματικότητα των RecNets στο Σχήμα 12 σε σύγκριση με πληθώρα σύγχρονων αρχιτεκτονικών. Η καμπύλη απόδοσης των RecNets δημιουργήθηκε με την εφαρμογή διαφορετικών υπερ-παραμέτρων (πλάτους επιπέδων, d). Παρατηρούμε ότι η προτεινόμενη αρχιτεκτονική παρουσιάζει πολύ καλά ποσοστά αναγνώρισης ακόμα και υπό μεγάλο ποσοστό συμπίεσης. Βέβαια, ένα μεγάλο μειονέκτημα της μεθόδου, στην πράξη, είναι η ακολουθιακή φύση της που ισοδυναμεί με μη-αποτελεσματικές υλοποιήσεις σε σχέση με τυπικά συνελικτικά επίπεδα (η εφαρμογή των RecNets στην ImageNet είναι εξαιρετικά χρονοβόρα).

Συμπίεση Μοντέλων με Χρήση Αραιών Βαρών

Μια κλασσική τεχνική συμπίεσης δικτύων, εντελώς διαφορετική από την δημιουργία συμπαγών αρχιτεκτονικών της προηγούμενης υπο-ενότητας, είναι η περικοπή μεμονωμένων βαρών ή και ολόκληρων νευρώνων (pruning). Ακολουθώντας αυτή την κατηγορία τεχνικών και συγκεκριμένα την περικοπή μεμονωμένων βαρών, μοντελοποιημένη ως πρόβλημα αραιοποίησης

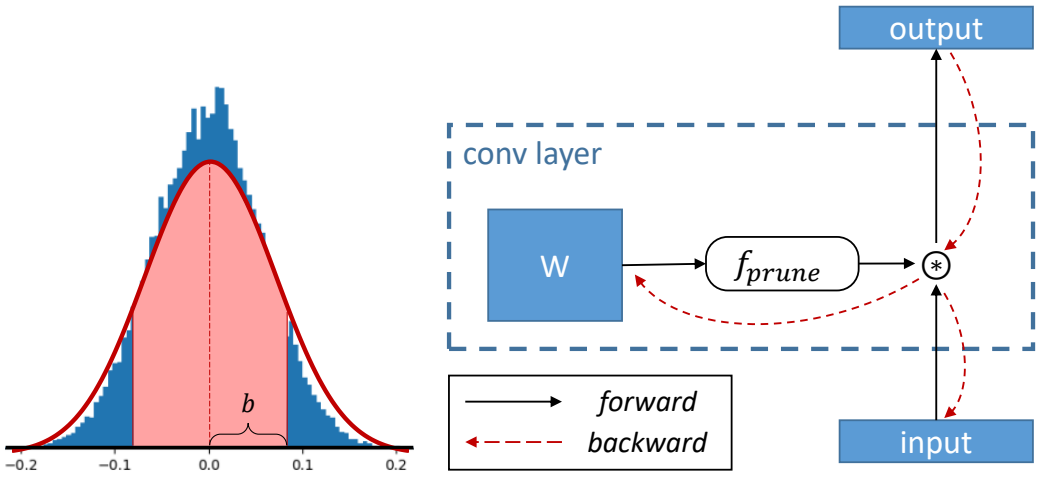


Σχήμα 11: Οπτικοποίηση της ακολουθιακής λογικής για την δημιουργία ενός CRC επιπέδου ($d = 4$ τμήματα), ακολουθούμενο από ένα συνελικτικό επίπεδο συμπίεσης.



Σχήμα 12: Σύγκριση με σύγχρονες αρχιτεκτονικές για την βάση CIFAR-100.

(sparsification), προτείνουμε μια καινοτόμα συνάρτηση απώλειας (loss function) που ελέγχει το συνολικό ποσοστό μη-μηδενικών βαρών.



(α) (β)

Σχήμα 13: (α) Υπέρθεση Γκαουσιανής κατανομής πάνω από την πραγματική κατανομή των βαρών (layer2_1 / ResNet50). (β) Χρήση STE για την περικοπή βαρών.

Για την περικοπή των βαρών χρησιμοποιούμε μια απλή συνάρτηση κατωφλίου:

$$f_{prune}(w;b) = \begin{cases} 0, & \text{εάν } |w| < b \\ w, & \text{αλλιώς} \end{cases}$$

Για τον έλεγχο του ποσοστού μη-μηδενικών βαρών αρκεί να ορίσουμε καταλλήλως το κατώφλι b. Η επιλογή του κατωφλιού, σύμφωνα με ένα ποσοστό αραιότητας s, μπορεί να προσεγγιστεί είτε με κάποια μέθοδο δυαδικής αναζήτησης, είτε μέσω διαστημάτων εμπιστοσύνης, εάν υποθέσουμε γκαουσιανή κατανομή βαρών. Η δεύτερη εναλλακτική έχει ιδιαίτερο ενδιαφέρον για δύο λόγους:

- η υπόθεση γκαουσιανής κατανομής προσεγγίζει επαρκώς την πραγματική κατανομή των βαρών (βλέπε Σχήμα 13(α))
- το ποσοστό των μηδενικών στοιχειών μπορεί να εκφραστεί αναλυτικά ως προς το κατώφλι b:

$$s = \text{erf}(\frac{b}{\sigma\sqrt{2}})$$

Το βασικό πρόβλημα για την εφαρμογή της παραπάνω λογικής σε κάθε βήμα εκπαίδευσης είναι ότι μετά την περικοπή βαρών, η κατανομή δεν διατηρεί την γκαουσιανή μορφή της. Η διατήρηση της μορφής της κατανομής επιτυγχάνεται με την ενημέρωση όλων των βαρών σε κάθε βήμα, ακόμα και αν δεν συνεισφέρουν, δηλαδή έχουν περικοπεί. Η λειτουργία αυτή ισοδυναμεί με έναν Straight-Through Estimator (STE) και περιγράφεται στο Σχήμα 13(β).

Εφόσον διατηρούμε την υπόθεση κατανομής με χρήση του STE, μπορούμε να περιγράψουμε το ποσοστό αιραιότητας κάθε επίπεδο σύμφωνα με την συνάρτηση λάθους erf(). Για όλο το δίκτυο (N επίπεδα), επιθυμούμε να μεγιστοποιήσουμε την συνολική αραιότητα ή αλλιώς:

$$L_s(\{b_i\}) = \frac{1}{N}\sum_i^N (1 - s_i)$$

$$= 1 - \frac{1}{N}\sum_i^N \text{erf}\Big(\frac{b_i}{\sigma_i\sqrt{2}}\Big)$$

Η παραπάνω συνάρτηση μπορεί να χρησιμοποιηθεί σαν συνάρτηση απώλειας (loss function) για την εκπαίδευση αραιών δικτύων και θα χρησιμοποιείται σαν επιπλέον όρος, μαζί με την συνάρτηση απώλειας L που σχετίζεται με το εκάστοτε πρόβλημα (και ελέγχει τις παραμέτρους $\{\mathbf{W}_i\}$ του μοντέλου). Συνεπώς, έχουμε να ελαχιστοποιήσουμε την εξής συνάρτηση:

$$L(\{\mathbf{W}_i\}, \{b_i\}) + \lambda L_s(\{b_i\})$$

Σύμφωνα με την παραπάνω μοντελοποίηση του προβλήματος τα κατώφλια $\{b_i\}$ (ένα για κάθε επίπεδο) είναι παράμετροι προς εκπαίδευση μαζί με τα υπόλοιπα βάρη $\{\mathbf{W}_i\}$. Αξίζει να σημειωθεί ότι είναι εύκολο να επεκτείνουμε την αναπτυχθείσα λογική σε προβλήματα περιορισμένων πόρων (budget-constrained), παραλλάσσοντας κατάλληλα τον επιπλέον όρο αραιότητας. Η αποτίμηση της προτεινόμενης μεθόδου, σε σύγκριση με σχετικές τεχνικές της βιβλιογραφίας, παρουσιάζεται στον Πίνακα 7.

Method	#Params (Million)	Top-1 Acc. (%)	Top-5 Acc. (%)
AlexNet reference	61.10	56.58	79.88
LWC [71]	6.71	57.20	80.30
L-OBS [42]	6.71	56.89	79.99
DNS [68]	3.48	56.91	80.01
Constraint-aware [30]	2.97	54.84	-
ADMM [196]	2.90	-	80.20
FDNP [114]	2.90	56.84	80.02
BA-FDNP [114]	2.70	56.82	79.96
Sparsity Loss	2.65	56.62	79.64

Πίνακας 7: Σύγκριση τεχνικών περικοπής μεμονωμένων βαρών για την αρχιτεκτονική AlexNet [105] και την βάση ImageNet.

Abstract

Document Analysis and Recognition is a prominent research area which combines the fields of Computer Vision and Machine Learning and has a great impact to humanitarian studies, by unraveling information stored in collections of historical documents all over the world. In this PhD thesis, we focus on extracting and learning visual representations capable of successfully detecting and recognizing text in handwritten documents. The main intention behind the developed methodologies, presented in thesis, is the creation of efficient systems with minimal computational requirements, aiming towards real-time applications. During the thesis, we tackle document-related problems of increasing difficulty, while the main goal is the development of a effective word detection approach by focusing on the improvement of the extracted visual representation of text. Specifically we explore feature extraction techniques along with possible improvement modifications, based on the specific characteristics of text images (possible text deformations e.t.c). Typical handcrafted feature extraction methods are compared to generating visual representations either from manifold embedding techniques or from deep learning approaches, which both show superior performance. An important part of this thesis is the study of Convolutional Neural Networks (CNNs) for the word detection problem along with their generalization capability, i.e.if it is possible to generate transferable and discriminative deep features. To this end, we propose several modified architectures in order to create compact, yet well-performing, features. Furthermore, we present a novel deep learning approach that combines both spotting and recognition tasks, leading to superior performance, while we also tackle the problem of line-level spotting from deep features viewpoint. Finally, we address the more generic neural network compression problem, which is not limited to document-related tasks. Specifically, we design two different approaches for model compression, both achieving significant compression according to size-accuracy trade-off on different datasets and settings, including image classification and keyword spotting tasks.

Keywords: Document Analysis and Recognition, Keyword Spotting, Deep Learning, Neural Network Compression

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Document Analysis and Recognition

Document analysis and recognition is an active area of research for over four decades. Digitization and understanding of documents is of great interest both to the computer vision and the humanities community. Over the years, several methods have been developed for processing a document image with the aim of acquiring the underlying text, aiming to automatically transcribe the existing text into an electronic format. Such a feat would enable easier access to information stored on document images, especially historical documents, since there is a vast number of untranscribed manuscripts. The analysis and recognition of diverse historical documents, in a large scale, could provide great insight to historical events by systematically mining information by different sources, a goal of significant importance to cultural heritage.

Towards building a successful text recognition system, which can be viewed as the ultimate goal of the document analysis field, many assisting subtasks can be defined (e.g. denoising, document segmentation into lines/words/characters e.t.c.). As one would suspect, document processing relies on both computer vision and machine learning approaches, usually lending ideas from more generic tasks of the computer vision field, such as object detection. In this section, we briefly summarize the most prominent tasks of the document analysis and recognition field.

1.1.1 Layout Analysis

The majority of documents consists of structured elements, such as paragraphs, lines, words e.t.c. Correctly extracting these elements would greatly assist the upcoming text recognition step, since the problem complexity can be significantly reduced.

A typical pre-processing step of a document analysis pipeline, in the previous decade, was to binarize the document image before processing it any further. Document binarization goal is to assign pixels corresponding to character as foreground (1) and every other pixel, non text related, as background (0), resulting into a binary image. The intuition is similar to extracting perfect edges/sketch, where the depicting object could be represented by the least possible discriminative information, without any confusing noise. However, this is a hard-decision step, which may lead to significant information loss and was dropped as a step in recent approaches.

The segmentation of a document to entities have proven to be a pivotal step towards well-performing recognition. Typical entities are: main text, titles, comments, marginalia, signatures, figures, stamps e.t.c. A useful categorization of text entities are (in descending

order): pages (if the manuscript is multi-page), paragraphs, lines, words and characters. Detecting the above entities can helps us to unveil the structure of the document. However, concerning text recognition, line/word/character segmentation is essential for fast and accurate systems. Such entities constrain the recognition problem to a sequential decoding of characters, or even a simple classification of character entities.

Handwritten documents can be very complex and character segmentation is almost impossible (consecutive characters are often merged into bigrams, trigrams e.t.c.) , while word segmentation is error-prone up to this date. Line entities were the easiest to correctly segment and many techniques were developed over the course of the years towards accurately detecting lines (from Hough transform to CNNs). Note that the segmentation step should provide almost perfect results in order to avoid error-propagation to the text recognition step.

Moreover, tables are important document structures, often encountered in historical records, which enclose important information in structured form (e.g. birth/death records). To this end, table detection and segmentation of table entities are crucial to a variety of tasks.

1.1.2 Text Recognition

Text recognition can be rightfully considered as the most crucial task of the document processing field and the existence of many different writing styles (between different people or even between eras) and languages add a notable complexity to the problem.

The predecessor of modern text recognition was Optical Character Recognition (OCR) which was a very popular area of research during the 90s (Neural Networks for computer vision tasks were first evaluated at character classification). OCR typically assumes a character segmentation step, which is feasible only for typewritten text. Even though OCR and typewritten text is of low practical and research interest at this day and age, it describes the evolution of computer vision and machine learning over the years.

Modern approaches focus on Handwritten Text Recognition (HTR), which is typically applied on text-line images and aim to fully decode the text into a sequence of characters, assuming a well-performing line segmentation step. Handwritten text recognition and voice recognition were developed simultaneously and ideas were often exchanged between these two fields. The backbone of state-of-the-art recognition systems is a recurrent module capturing the sequential nature of text. Initially Hidden Markov Models were employed, while the last decade Recurrent Neural Networks (RNNs, LSTMs e.t.c.) are at the spotlight with exceptional performance. As expected, recognition relies on machine learning techniques and the main problem was defining a loss which can correlate the sequence of extracted feature vectors to the character sequence, which is the target. This was successfully addressed by the seminal work of Graves et al. [65], proposing the Connectionist Temporal Classification (CTC) loss, efficiently implemented by dynamic programming. Notably, CTC used to train a bidirectional recurrent neural network (for more details see[65]) before the recent bloom of deep learning.

Even though trainable models, such as RNNs, learn implicitly the language model existing in the training corpus, one can assist the final decoding by explicitly impose external language models on the (viterbi-like) decoding process. External language models can be in the form of n-grams or even word-grams, exported from an existing corpus of the target language. Such extra information, drawn from language modeling, leads to noteworthy improvement since it can correct possible errors of the visual features decoding. A simple intuitive explanation of the decoding process, referred to as *Beam Search*, can be

described as follows: Beam-search explores a (characters' probability) graph by expanding the most promising node, at each time-step, considering a limited set of k alternative paths. The k hyper-parameter is called beam width. The paths are selected in order to maximize the visual character probability along with the probability of the existence of the decoded consecutive characters according to the external language model.

1.1.3 Keyword Spotting (KWS)

An alternative to the handwritten text recognition approach is keyword spotting (KWS) which can be defined as the task aiming to retrieve specific words of interest in a document collection without the need of transcribing every single word on the collection given a specific keyword as a query. Searching a specific word in a set of documents is the most common task in practice in order to filter the vast amount of information stored in large collections of manuscripts.

A keyword spotting system returns a ranked list of the word images with respect to the degree of similarity with the query. KWS is categorized into two taxonomies with respect to the query: 1) Query by String (QbS), where the query is a string of characters. 2) Query by Example (QbE), where the query is a template image. The former category is strictly correlated to machine learning techniques since visual features should correspond to characters. Specifically, one straightforward solution to the problem is the complete automatic transcription of the document using an HTR system, followed by a substring search on the HTR output. Such simplified solution is not optimal, given that crucial information would be lost if we only use the decoded characters. Another, most sophisticated approach, is to store the graph of character transitions and perform a constrained decoding over the graph. The latter category is a case of template matching and can be resolved by classical computer vision techniques (e.g. Bag of Visual Words). By interpreting keyword spotting as a template matching problem, one can focus on generating discriminative visual features which lead to efficient word representations. If the whole document is stored as a set of features, one can draw the query image, following the QbE paradigm, and perform language independent keyword spotting. This last observation is the main intuition behind a significant part of this thesis, while taking into consideration the time and storage restriction in order to scale the application of a KWS system on large document collections.

1.1.4 Other Document-Related Tasks

Document processing is not constrained only on layout analysis and recognition/detection. Some examples of other popular document related tasks are:

Document Forensics The goals of the document forensics field include: writer identification, detection of forged/fake documents, authentication of documents and automatic signature identification and verification. The importance of such tasks is evident and the automatic forgery detection and authentication would greatly assist the automation of processing and verifying legal forms. Implementation-wise, forensics methods require the discovery of small patterns, through the utilization of machine learning approaches, that differentiate various writers or an authentic document with possible forgeries.

Document Classification and Document Understanding Often companies have large collections of a vast variety of different document forms (legal documents, invoices

e.t.c.) which are difficult to handle. Document understanding aims to digitize, classify, and extract knowledge (such as form fields, text passages, tables, and graphs) from thousands of documents. It helps to organize and store knowledge graphs and other extracted data for easy search, query and monitoring.

1.2 Contributions and Thesis Structure

The contribution of this thesis is mainly focused on discriminative visual representation methods, while the keyword spotting task is the prominent paradigm for evaluating the developed methods. The presented methods are addressing incrementally more difficult problems – from character classification to line-level KWS – and our main effort is shifted from handcrafted descriptors to compact Convolutional Neural Networks. Towards exploring compact neural networks, we also developed two methodologies for model compression, independent of the recognition/spotting tasks.

In the following, we present a brief description of the upcoming chapters, along with the respective contributions:

- **Chapter 2: Theoretical Background and Related Work** Deep Learning plays a major role in this thesis and thus the theoretical background of neural networks is presented, including architectures and optimization schemes. Next, we establish the current status quo of handwritten recognition and keyword spotting by highlighting possible categorizations and existing state-of-the-art approaches.

- **Chapter 3: Character Recognition** First, the simplest problem of document recognition is considered: *character classification*. Even though it has low practical value (we already noted that character segmentation to handwritten documents is not feasible), character classification is ideal for the evaluation of a new feature extraction method that we developed, called Projection of Oriented Gradients (POGs). Moreover, we address the problem of label absence for the case of character recognition over typewritten documents. Since there is no prior information about the existing characters (number of classes, language e.t.c.), we propose a user-assisted system for label extraction with minimized user interaction.

- **Chapter 4: Learning-Free KWS** This chapter corresponds to our earlier work, before the adoption of deep-learning as the backbone of our KWS systems. Considering only the Query by Example case, we explore KWS as a template matching problem. Therefore, since no training is required, our efforts are focused on the feature extraction step and the subsequent matching step between image descriptors. To this end, we analyze the most common variations of handwritten text-images and we efficiently address them at preprocessing (remove global rotations and translations), feature-extraction (use POG as a robust descriptor to small deformations) and matching (inner translations between characters). The developed methods outperform all existing learning-free approaches.

- **Chapter 5: Manifold Learning on Word Representations** Non-linear dimensionality reduction techniques, such as manifold learning, can effectively project the initial high-dimensional descriptors into a low-dimensional space, where euclidean distance is much more intuitive and effective. Such dimensionality reduction can be beneficial to word representations, generated by learning-free KWS methods, which typically are descriptors of high-dimensionality. However, concerning KWS, queries

are out-of-sample data and their projection to the low-dimensional space is not trivial, especially when time constraints are important. To overcome this problem, we propose a solution to the out-of-sample problem for the t-SNE [182] manifold learning technique, generating projections with minor computational overhead and significant retrieval improvement.

- **Chapter 6: Word Representations using Deep Features** Following the unprecedented success of Convolutional Neural Networks (CNNs) to a vast variety of tasks, we explore several deep learning aspects (architecture, training strategies) under the prism of keyword spotting, aiming to a CNN-based robust word image representation. Having performed an extensive exploration based on a state-of-the-art neural network PHOCNet [166], we reconsider both architectural and training-related aspects in order to build efficient and well performing neural networks capable of producing compact feature vectors. Specifically, we employ an ensemble strategy to further increase the performance of existing KWS networks, while we attempt to untangle the character class information from its spatial information (i.e. the position of the character in a word from the actual character class) in order to promote more compact representations.

  More importantly, we develop a deep learning approach that combines both word recognition and word spotting, achieving state-of-the-art results in both tasks. This approach lends from sequence recognition ideas and consists of two modules: the Encoder and The Decoder. Encoder module encodes the input image, treated as a sequence, into a fixed-sized vector. This vector is then decoded by the Decoder module into the target sequence of characters, i.e. the corresponding word text. The intermediate fixed-sized representation can be used as an effective word representation, enabling word spotting applications as a by-product. Simultaneously, this representation can be inherently decoded by the Decoder module.

  Finally, we address the problem of line-level keyword spotting which is governed by constrained recognition approaches. Contrary to prior works, we tackle tis problem from a feature-based viewpoint, searching for a specific feature template over a sequence of such features, representing a text-line. According to the proposed approach, we can efficiently perform both QbE and QbS searches over text-lines. Key components for the success of our method are two additional neural networks: the *character width estimator* which serves as a scale detector in order to assert scale invariance and the *common space encoder* which projects both visual features and attributes into a common space, enabling both QbE and QbS alternatives.

- **Chapter 6: Neural Network Compression**

  An occurring theme during the exploration of deep learning techniques was the demanding computational requirements of recent deep architectures (e.g. VGG [160]), both in number of parameters and inference time. Nevertheless, neural network compression is a very active research area, not limited to document analysis but of broader interest. Since the advent of AlexNet [105], the first deep CNN to advance the state-of-the-art in a highly challenging image classification task, building deeper and wider neural networks has been the main research direction in order to further improve performance in a variety of visual recognition tasks. While such requirements can easily be met by server-class computing systems, this is not the case for less powerful mobile or embedded devices, thus limiting the deployment of state-of-the-art models in many real-life applications. Therefore, the need for compact

architectures and compressed models is evident.

Towards model compression, we develop two vastly different approaches, evaluated on the image classification task in order to be comparable with other compression methods in the literature. First, we propose a novel compact architecture based on the concept of sharing weights across different layers, employing a recurrent formulation akin to RNNs. Next, we explore the weight pruning alternative. We propose an adaptive sparsity loss, enabling budget constrained optimization, while retaining the same convergence with the initial dense network. This approach achieves state-of-the-art results across different settings and datasets, including the compression of a keyword spotting network, presented at the previous chapter.

- **Chapter 7: Conclusions** This final chapter of the thesis summarizes the conclusions of the conducted research under description. In addition, we highlight the most important research directions, according to our understanding of the existing literature, that were not yet adequately explored and can assist or complement the existing work.

Chapter 2

Theoretical Background and Related Work

In this chapter we briefly present the theoretical background, mainly focusing on neural networks which play a major role in the second half of this thesis. Next, we describe notable literature works in the area of text recognition and keyword spotting, both closely related to the main theme of this thesis: descriptive text image representations.

2.1 Neural Networks

2.1.1 From Feature Extraction to Deep Learning

Computer vision tasks, up until the recent resurgence of Neural Networks, were governed by a distinct methodology outline summarized by the following steps: 1) feature extraction and 2) machine learning. The majority of existing approaches over a plethora of tasks (e.g. object detection, template matching, e.t.c) heavily relied on the feature extraction step, since the goal was to extract as much information possible from an image into a compact descriptor. These handcrafted feature representations were then fed to a task-related machine learning approach (e.g. Support Vector Machines [34] for image classification). Statistical properties, shape descriptors or texture descriptors were successfully used as features in computer vision. Usually, data were scarce and resources limited, thus the features should be highly discriminative in order to enhance the method's generalization. Discriminative representation was enhanced by unsupervised learning, such as Principal Component Analysis (PCA), Guassian Mixture Models (GMMs) or manifold embedding. An indicative, highly popular, methodology that was prominent over a vast variety of computer vision tasks, involved the detection of interest points over different scales and the extraction of local descriptors around these points ([119, 14]). As local descriptors, gradient orientation was encoded into fixed-sized histograms . Each image was the represented by a set of different local descriptors. Concerning matching or detection applications, a transformation between similar descriptors was sought, trying to retain topological properties of each image (e.g. allowing only affine transformations). Classification applications required a supervised learning step (e.g. SVMs) and a fixed-sized representation for the entire image. To this end, the set of local descriptors was organized into a histogram based on the Bag of Visual Words approach [108].

Until recently, machine learning techniques, such as SVMs and NNs, were only emploed as the last step after meticulously crafting the feature extraction step. However, the abundance of labeled data and the computational capabilities of state-of-the-art hardware

equipment (first and foremost GPUs) led to a unprecedented surge of neural networks utilization over every possible computer vision task during the last decade. Notably, existing resources enable the efficient training of end-to-end *deep neural networks*, consisting of hundreds, even millions, of layers. Since the advent of AlexNet [105] (2012), the computer vision and machine learning community shifted towards using Deep Neural Networks (DNNs) as optimal feature extractors. In fact, the vast majority of the recent continouisly expanding computer vision literature relies on DNNs for almost every possible sub-task. Since deep learning will also serve as the core component for a major portion of this thesis, we briefly analyze several important aspects of building and optimizing neural networks.

2.1.2 A Brief History of Neural Networks

The concept of neural networks was firstly introduced at 1943 by McCulloch and Pitts. The McCulloch-Pitts neuron is also known as linear threshold gate and consisted by a linear transform topped by a step activation function. The Perceptron, an enhancement over the McCulloch-Pitts Neuron, proposed by Rosenblatt at 1958, introducing the idea of trainable weights along with an appropriate training algorithm for binary classification. In order to extend the neuron concept to classify non-linearly separable classes, multiple neurons are stacked to build a multi-layer feed forward network (typical use case was the XOR problem). Intermediate layers between input and output, where referred to as *hidden layers*. The Perceptron and a multi-layer network example are depicted at Figures 2.1 (a) and (b), respectively.

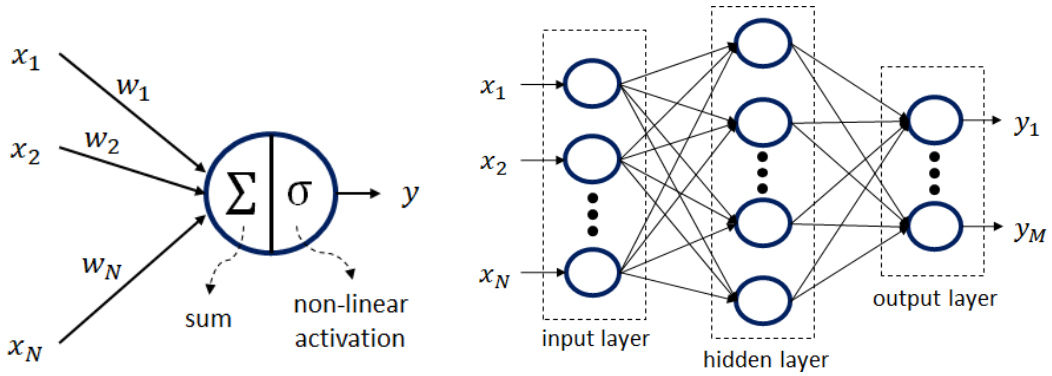


Figure 2.1: Visalization of (a) the Percepton Neuron, consisted of a dot product and a non-linear activation, and (b) an example of multi-layer feed forward neural network, consisted of multiple neurons and 3 layers: input, one hidden and output.

Since the formulation of multi-layer feed forward networks, leaps have been made towards complex structures of neurons, leading to two types of neural networks that will emerge multiple times in this thesis: *Recurrent Neural Networks (RNNs)* and *Convolutional Neural Networks (CNNs)*.

2.1.3 Recurrent Neural Networks

Sequence modeling is a prominent area of research, with notable results in Speech Recognition and Natural Language Processing. Sequences of data cannot be modeled by typical neural network formulation and therefore an alternation is required. To this end, Recurrent Neural Networks (RNNs) were introduced. Assuming the $\{\mathbf{x}_i\}$ segments form the

input sequence of an RNN and the $\{\mathbf{h}_i\}$ segments comprise its hidden state, we get the following recurrent formulation:

$$\mathbf{h}_i = \sigma(f_h(\mathbf{h}_{i-1}) + f_x(\mathbf{x}_i)) \tag{2.1}$$

where $\sigma()$ is a nonlinear activation function and $f_h()$, $f_x()$ are the transformation functions for the hidden state and input respectively. The transformations $f_h()$ and $f_x()$, are linear and can be formulated by a weight matrix and a bias, e.g. $f_h(\mathbf{x}) = \mathbf{W_h x} + \mathbf{b_h}$. Note that at each step of the recurrent formulation of Equation 2.1, the same weights, comprising the functions $f_h()$ and $f_x()$, are used. Thus, an unrolled version of RNNs can be viewed as a typical NN with shared weights. Optimizing an RNN can be performed by a gradient descent algorithm, as done with simple multi-layered NNs, even though RNNs are considered sensitive to optimization related problems, such as vanishing gradients (the optimization scheme will be described in detail in an upcoming subsection).

Apart from vanilla RNNs, there exist several variants. Specifically, we distinguish the following cases:

- *Multi-layer RNN:* In practice, RNNs have several layers, i.e. the output sequence of the first RNN is fed as an input sequence to the second RNN, etc. Each layer is described by the recurrent relation of Eq. (2.1) and uses a pair of shared parameter tensors $(\mathbf{W_x}^k, \mathbf{W_h}^k)$, where k is the layer's identifier.

- *Bi-directional RNN:* A major issue with typical RNNs is that they learn representations from previous time steps: $\mathbf{r_i} = f_r(\mathbf{r_{i-1}}, \mathbf{x_i})$. However, it may be helpful to incorporate information from future time steps: $\mathbf{l_i} = f_l(\mathbf{l_{i+1}}, \mathbf{x_i})$. Bi-directional RNNs combine both information flows, rightward $\mathbf{r_i}$ and leftward $\mathbf{l_i}$, at each time step i: $\mathbf{h_i} = f_c(\mathbf{r_i}, \mathbf{l_i})$. Sometimes, you might have to learn representations from future time steps to better understand the context and eliminate ambiguity.

- *Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) Networks:* A gated recurrent unit has an update gate, which decides on how much of information from the hidden state should be let through, and a reset gate, which decides on how much of information from the hidden state should be discarded [33]. GRU has been proposed as a cost-efficient alternative to Long Short Term Memory (LSTM) Networks [83] and both networks aim to retain information from distant time steps, without vanishing it through time. LSTMs and GRUs are preferred in the majority of recent sequence-related applications.

One notable application of recurrent networks in recent literature is Sequence to Sequence (Seq2Seq) models [169], where a sequence is encoded into a fixed-sized representation and then decoded into a new sequence. Seq2Seq is comprised by two distinct recurrent networks (usually LSTMs or GRUs), an encoder and a decoder (and recently an extra attention module [13]).

2.1.4 Convolutional Neural Networks

An important aspect of neural networks resurgence is the introduction of Convolution Neural Networks, which can effectively handle images. Image related problems had always been challenging, since spatial context should be captured. Traditionally, spatial filtering was performed by convolution with a handcrafted kernel, designed to capture specific patterns (e.g. edges). Convolutional networks introduced trainable filters which can generate

discriminative feature maps, optimized with respect to the task at hand. This approach revolutionized the computer vision domain, pushing aside suboptimal handcrafted features. CNNs are mostly comprised by stacked convolutional layers, which perform the convolution operation: $\mathbf{Y} = \mathbf{X} * \mathbf{W}$, where $\mathbf{X}$ and $\mathbf{Y}$ are the input and output 3-D tensors respectively, while $\mathbf{W}$ is the 4-D kerneled weight tensor. Specifically, using the typical spatia (2-D) cross-corellation operation ($\star$), the convolution operation ($*$) is defined as:

$$\mathbf{Y}[m] = \sum_{n=1}^{C_{in}} \mathbf{X}[n] \star \mathbf{W}[m,n] \, , m = 1, \ldots, C_{out}$$
$$(\mathbf{Y} \in \mathbb{R}^{C_{out} \times H \times W} \, , \mathbf{X} \in \mathbb{R}^{C_{in} \times H \times W} \, , \mathbf{W} \in \mathbb{R}^{C_{in} \times C_{out} \times k_H \times k_W})$$

The spatial dimensions $H \times W$ and $k_H \times k_W$ correspond to the the feature map and the kernel size, respectively, while C_{in} and C_{out} correspond to the number of 2-D feature maps on the input and on the output of the convolution (usually called channels in the CNN literature). Essentially, convolution layer transforms an image feature map to another, taking into account contextual information about each pixel's neighborhood. Note that convolution is a linear operation, replacing the linear projection of traditional NNs. Layers close to the input generate low-level features, such as edges, while layers close to the output generate high-level features of complex shape and texture, e.g. a nose or an eye for face detection. Convolutions with large kernels are extremely costly, thus multiple stacked convolutions of small kernel (3×3) are preferred [105, 160] (retaining large receptive field). Finally, the stacked layers in successive order is not a necessary architectural requirement. On the contrary, the majority of recent architectures contains complex information flows of multiple paths (e.g. GoogleNet [170], ResNet [77], DenseNet [87] e.t.c.).

2.1.5 Categories of Layers

Next, we briefly describe a variety of layers, often met in recent architectures.

Linear/Convolutional Layer:

At the heart of NNs lies a simple idea: alternate between linear transformations and non-linearities to build complex functions. The linear transformation can be one of the following:

- *Fully connected layer:* a simple linear projection performed by a 2-d tensor ($C_{in} \times C_{out}$).
- *Convolutional layer:* performs a convolution operation of kernel $k_H \times k_W$ for every input/output channel pair, resulting to a 4-d tensor of size $C_{in} \times C_{out} \times k_H \times k_W$.

Activation Layer:

A non-linear function following every linear transformation. These non-linearities greatly contribute to the representational capabilities of NNs. Typical activation functions are: tanh, sigmoid, ReLU (ramp function).

Downsampling:

Typical CNN architectures increase the number of channels after a group of layers. The generated feature maps contain abstract encodings of the underlying information and thus

there is no need for per pixel representation, which adds unnecessary overhead. Instead a down-sampling operation should be performed. Such operations are referred to as pooling operations, where neighborhoods of pixels are replaced by a single one (e.g. pooling over 2×2 neighborhood with stride 2). Obvious choices for pooling operations is the maximum or the average over a group of neighboring pixels. One computationally cheap alternative is to perform strided convolutions without any pooling operation.

Batch Normalization:

The constraint of a specific range of values over the layer's input/output may be beneficial (e.g. avoid extreme values, which may affect the gradients). Batch normalization (BN) [92] offers such a constraint by computing the running mean value and standard deviation, updated at each batch by a momentum scheme. The input is then normalized to approximate a normal distribution of zero-mean and unitary deviation. Batch normalization helps the convergence of the network's optimization.

Dropout:

One main problem of NNs is overfitting, i.e. learning exclusively the training dataset without the ability to generalize. This was addressed by data augmentation and the introduction of random noise into the network. A form of such noise is the random zeroing of channels, dubbed dropout [161]. Dropout essentially assists the creation of multiple "paths" of information through different channels and avoids correlating a neuron/channel with a specific input sample, thus enhancing generalization.

Finally, we should note that every task-related differentiable function can be used as a building block for NNs, e.g. an interpolation function implementing affine transformations over images.

2.1.6 Optimization

We have already defined, architecture-wise, a variety of neural networks, but the main question remains: *how do we train such networks?* Every NN realizes a nonlinear parametric function $\hat{y} = f_\theta(x)$, where θ is the set of all the weights comprising the network. Training such parametric function, with respect to the weights of the model, requires the following steps:

- Define a training set of input samples x_i and output targets y_i.
- Define a loss function $L(\hat{y}_i, y_i)$, which quantifies the proximity of the prediction $\hat{y}_i$ to the requested target y_i. The loss function is task dependent and its minimization corresponds to having perfect predictions, i.e. $\hat{y}_i = y_i$.
- Select an optimizing algorithm to perform $\theta^\star = \text{argmin}_\theta J(\theta)$, where $J(\theta) = \sum_i L(f_\theta(x_i), y_i)$. Due to the complexity of the underlying function, there is no analytic solution for the aforementioned optimizing scheme. Therefore, iterative gradient-based algorithms are employed in order to gradually minimize the overall loss (e.g. gradient descent).

Loss Function

The selection of the loss function is critical for the effectiveness of the trained model. Two main properties should be kept in mind:

- The loss function should reflect the task's goal, e.g. Mean Squared Error (MSE) for regression or Cross Entropy (CE) for classification. Losses could be complex and consist of multiple terms when considering multi-task problems.
- The function should differentiable since a gradient-based optimization scheme is considered.

Backpropagation

The main problem with training neural networks is their complex structure of stacked layers. Computing the gradient of each parameter with respect to the loss function relies on a simple yet powerful concept: the chain rule. Iterative gradient-based optimizations schemes can be summarized by the following two steps: 1) Compute Gradients and 2) Update Weights. These two steps are performed iteratively until convergence is guaranteed[1]. Upon convergence the gradient would be close to zero, which translates to detecting an optima. Nonetheless this optimization scheme cannot guarantee that the optima, discovered by the iterative algorithm, is global.

Compute Gradients: When training deep architectures, consisting of many stacked layers, the gradient computation at each layer is not straightforward. The solution is to perform the gradients computation layer-wise, starting from the loss function and moving backwards towards the input layer, hence the backpropagation term of this approach. Let's consider the example of an intermediate fully-connected layer described by $y = \sigma(\mathbf{w}^T\mathbf{x})$, where $\mathbf{x}$ is the k-dimensional input, y is the single dimensional output, $\mathbf{w}$ is the layer's weights and $\sigma()$ is a non-linear activation. Bias is omitted and output is one-dimensional for simplifying the analysis, but their extensions are straightforward. Given the gradient at the output $\vartheta J/\vartheta y$, we want to compute the gradients $\vartheta J/\vartheta \mathbf{w}$ and $\vartheta J/\vartheta \mathbf{x}$ in order to update the weights and propagate the gradient error to previous layers, respectively. The first one, using the chain rule, can be expressed as:

$$\frac{\vartheta J}{\vartheta \mathbf{w}} = \frac{\vartheta J}{\vartheta y}\frac{\vartheta \sigma(\mathbf{w}^T\mathbf{x})}{\vartheta(\mathbf{w}^T\mathbf{x})}\frac{\vartheta(\mathbf{w}^T\mathbf{x})}{\vartheta \mathbf{w}} = \frac{\vartheta J}{\vartheta y}\sigma'(\mathbf{w}^T\mathbf{x})\mathbf{x} \tag{2.2}$$

while the second one:

$$\frac{\vartheta J}{\vartheta \mathbf{x}} = \frac{\vartheta J}{\vartheta y}\frac{\vartheta \sigma(\mathbf{w}^T\mathbf{x})}{\vartheta(\mathbf{w}^T\mathbf{x})}\frac{\vartheta(\mathbf{w}^T\mathbf{x})}{\vartheta \mathbf{x}} = \frac{\vartheta J}{\vartheta y}\sigma'(\mathbf{w}^T\mathbf{x})\mathbf{w} \tag{2.3}$$

The above utilization of gradient chain rule, can be used sequentially from the network's output to its input, in order to calculate every parameter gradient w.r.t. the objective loss function.

Update Weights: Vanilla Gradient descent computes the gradient score of the objective function with respect to the parameters θ of the entire training dataset and uses the following update rule: $\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta)$. The hyper-parameter η controls the convergence rate. Small value may lead to slow convergence, while large values lead to overshooting and divergence. The above formulation is impractical, since calculating the gradients over the entire dataset can introduce significant computational overhead. To address this problem, the classical Stochastic Gradient Descent (SGD) optimization algorithm was used. SGD

[1] stopping criteria may vary and are usually empirical

performs a parameter update for each training sample. An entire iteration of every dataset sample is referred to as *epoch*. The pairs of inputs/targets are fed to the SGD in a different sequence at each epoch (hence the Stochastic term in SGD). In practice, the mini-batch alternative of SGD is used, where the gradients are computed over batches of samples. Additionally, SGD is usually accompanied by a momentum strategy, where a "history" of previous gradients is used in order to avoid steep differences in the gradient direction by the current batch. This way, extreme oscillation as well as the risk of getting stuck in a local minimum are reduced. Note that both learning rate and momentum should be empirically set by the user.

An alternative popular optimization algorithm, akin to SGD, is Adaptive Moment Estimation [97], referred to as Adam. Adam is a method that computes adaptive learning rate for each parameter, an attempt to decrease the user-defined hyper-parameters compared to SGD. Adam keeps an exponentially decaying average of past gradients, similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface. Due to its formulation, Adam is more robust when sparse targets are considered.

2.2 Handwritten Text Recognition

The majority of document-related applications include a text recognition step. Ideally, the ultimate goal of document analysis is the full transcription of a document image. Handwritten documents pose numerous challenges (e.g. complex layouts and vast variety of writing styles), making text recognition an active area of research. Most text recognition approaches operate on text lines (or even text words), assuming a successful line segmentation step was proceeded [111, 117]. This assumption simplifies the initial complexity of characters' spatial detection, since a text image can be processed in a sequential way, recognizing character by character. The sequential viewpoint of text recognition shares many similarities with speech recognition, a classical problem of signal processing and machine learning. In fact many text recognition approaches were influenced by their counterparts in speech recognition and vice versa.

2.2.1 From HMMs to NNs

Following the speech recognition paradigm, first attempts towards line-level text recognition were governed by the use of Hidden Markov Models [98, 24, 175, 131, 45]. Typically, simple statistical features were extracted column-wise from the initial text-line image, forming a sequence of features which was then fed to an HMM. Due to the simplicity of the features, a preprocessing step is necessary to alleviate possible spatial variations [130], e.g. translation, rotation (skew) and slant. According to the HMM formulation, each character is described by a set of hidden states. The number of these states differs from character to character, e.g. "i" has fewer states than "w". Finally, the recognized text is generated by the Viterbi decoding algorithm.

The sequential nature of text-line recognition makes recurrent networks perfect candidates. Nevertheless, the choice of an appropriate loss is not straightforward. Specifically, the loss function should capture the dynamics of different possible alignments between the output sequence and the target sequence of characters. This was addressed by Graves et al. at his seminal work "Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks" [63], where the Connectionist Temporal Classification (CTC) was firstly introduced. Connectionist Temporal Classification is

a segmentation-free and alignment-free algorithm for sequence-to-sequence transduction, which can serve as the loss function in recognition problems. The final recognition is performed by a Viterbi-like decoding process, named CTC decoding, that takes into account the CTC properties (e.g. the extra blank character used to separate consecutive same characters) [65, 66, 62]. Recently, text recognition was also approached as a Sequence to Sequence problem, providing an alternative to CTC backbone [124].

2.2.2 Language Models and Decoding

Simply taking into account the visual cues is not enough for text recognition. In fact, even humans require some language context in order to clarify possible ambiguities between similar characters, e.g. "o" and "a". To this end, a post-processing step is applied after the visual recognition, which tries to incorporate existing language models (lending from NLP field of research). The simplest form of a language model is the occurrence frequency of characters' n-grams, given a specific corpora, serving as a prior over the existence of specific n-grams, generated by the visual recognition step. Language models can be extended to n-grams of entire words. Decoding using a language model can assist both HMM and NN approaches (e.g. [174] and [65], respectively). Core of such decoding processes is dynamic programming, akin to the Viterbi decoding, seeking to overcome the problem of exponential combinations of characters. Beam search algorithms are often used in practice, retaining alternative versions (up to pre-defined maximum number) of the possible decodings at each step [127, 174, 64, 90, 157]. In order to further assist beam search, prefix trees are used to "predict" upcoming characters and words [127, 174, 157].

2.2.3 Metrics

The performance of HTR methods is recorder in terms of Character Error Rate (CER) and Word Error Rate (WER). Their definition is straightforward:

- **CER:** the percentage of misclassified characters.
- **WER:** the percentage of misclassified words. Each text is segmented to words. Even if one characters is wrong, the word is considered misclassified.

Note that these two metrics are not linearly correlated. For example consider that only one character is misclassified and its corresponding word is wrong. If its neighboring character is also misclassified (typical bi-gram related issue), the CER will increase while WER will remain the same.

2.3 Keyword Spotting

A considerable portion of this thesis is focused on the keyword spotting application, a practical alternative to handwritten text recognition. Therefore, in this section, we will thoroughly explore the different taxonomies of KWS and present the existing state-of-the-art approaches, as well as the datasets and metrics often met in the literature. We also present, in more detail, two works ([9], [166]), which will serve as the backbone to many of our developed methods. As we have already mentioned, the keyword query which needs to be detected can be either a text string, referred to as *query by string (QbS)* keyword spotting ([50], [177], [52], [152]), or an example image, referred to as *query by example (QbE)* keyword spotting ([141], [99], [144], [158], [7]).

Keyword spotting techniques are also divided into two main categories based on the considered search space: *Segmentation-Free* and *Segmentation-Based* approaches. Segmentation-free approaches aim to locate the query instances on the whole document without the involvement of a segmentation step ([152], [7]), whereas segmentation-based approaches assume that a segmentation step has preceded. Segmentation-based techniques can be further divided with respect to the level of segmentation. Word-based techniques assume that words are already segmented and focus on comparing word images ([144], [9],[166]). These techniques are usually applied after an initial segmentation-like step that aims at fast retrieval of candidate regions of words as in [99]. On the other hand, line-based techniques assume that a line-segmentation was previously applied and the goal is to perform word spotting on line level ([50], [52]). A advantage of segmentation-based approaches compared to segmentation-free is the computational cost of processing specific regions of the documents (located words) rather than the whole document.

Initially on this thesis, we will assume that the segmented words are provided and we will focus on extracting discriminative information from the already segmented images, as this is the case in several recent works [9],[166]. Word segmentation is not addressed in this thesis, but it is a well-studied subject and existing techniques, such as [118] and [99], perform sufficiently well and are cost-effective. Typical word segmentation methods, which require to find a unique region for each existing word, have not proven to be very accurate. However, a word segmentation technique could provide candidate word regions, that may be overlapping, and aims to reduce significantly the search space on the document. These candidate regions do not necessarily have a unique correspondence to the existing words in the document, since for each actual word in the document there could be multiple possible candidate regions.

A different taxonomy of the keyword spotting methods is related to the existence of a training phase. Techniques that involve a training step belong to the *learning-based* category whereas methods that work directly on the collection without any prior knowledge except from the query belong to the *learning-free* category.

Typically, this taxonomy is closely related to QbS and QbE approaches. Learning-based techniques are required for QbS approaches in order to pre-train character models, e.g. HMM models [50], and eventually associate the visual information to characters. These character models can be concatenated in order to form the query string, i.e a word model. Learning-free techniques are usually associated with QbE approaches, which resemble template matching. However, there are approaches that perform QbE word spotting by using pre-trained models ([9], [166]). One of the major goals of this thesis is to explore the capabilities of a learning-free keyword spotting system, since the existence of a training set is rare, mainly due to the difficulty on the annotation of document collections, the existence of several languages as well as the variability of historical documents of different eras.

Keyword spotting in handwritten documents is considered as a challenging task due to the variability of different writing styles. This challenge is even more noticeable in learning-free approaches since it is expected that out of a single word instance (query) the KWS method should be able to simulate all possible variations and eventually return all relevant appearances of the word across the document collection.

A common system for a segmentation-based QbE keyword spotting task usually consists of the following steps. First, a preprocessing (normalization) step is applied to each already segmented word image, where several common variations (skew, slant e.t.c.) are absorbed leading to word images of reduced variability. At a next step, feature extraction is performed in order to create a compact representation of each normalized image. The

final step involves the calculation of a similarity between the searched keyword and word images belonging to the document collection, based on features produced at the previous step. The output of such a system is a ranked list of all the words in the collection which reflects the similarity of each one to the provided query, i.e. words very similar to the query appear at the top of the list. It is evident that the selection of a similarity measure is heavily dependent on the nature of the previously extracted features. In this article, segmented words belonging to the document collection will be referred as word images whereas word samples which are used as queries will be referred as query images.

A taxonomy of the QbE keyword spotting methods can be defined based on the output of the feature extraction step. In more detail, methods can be grouped into two main categories: *methods that produce a fixed feature vector* [99], [6], [144], [158], [7], [9] and *methods that produce a set of features* [86], [141], [190]. The matching procedure is directly dependent on the choice of the feature representation. In more detail, methods belonging to the first category use a simple distance/similarity measure for the matching step (e.g. Euclidean distance). On the contrary, methods that are grouped to the second category require a more sophisticated algorithm for matching.

Methods belonging to the first category include the work of Kovalchuk et al. [99] who proposed the use of the Histogram of Oriented Gradients (HOG) together with the Local Binary Pattern (LBP) descriptors, using binarized images as input, rescaled to a fixed size. The ranking is performed using the Euclidean distance after the application of a dimensionality reduction operation. This method reports very good results despite its simplicity and is ranked second in the overall ranking of the ICFHR 2014 competition [134], while achieving the best performance in the historical dataset. Similarly, in our previous work [144], POG descriptor is used both as a global descriptor and as a local applied to zoned segments of the image. A simple fusion of the Euclidean distance of these two approaches (global and local) achieved the best published results on the historical dataset of ICHFR 2014 competition. In [158], a pre-trained Deep Convolutional Network (DCN) is used on a zoning of the word images for feature extraction. The DCN was trained on an independent set of typewritten characters. Consequently this DCN utilization can be seen as a procedure that produces discriminative features with respect to (typewritten) character images. Aldavert et al. [6], used Spatial Pyramids and Bag of Visual Words (BoVW) representations based on HOG local descriptors, which are very popular techniques in the object recognition area of research, reporting encouraging results. Instead of HOG descriptors, Rothacker et al. [137] constructed Spatial Pyramids using SIFT as local descriptors. The final ranking of the words is performed using the Bray-Curtis distance. It should be noted that this method won the ICDAR 2015 keyword spotting competition [137]. A similar work was proposed by Rusiñol et al. [137]. The authors made use of Integral Histogram of Gradients as local descriptors using the Euclidean distance for producing the final ranking. The method took the second place in ICDAR 2015 keyword spotting competition.

Another approach on QbE keyword spotting was presented by Almazan et al. [7]. In this work, machine learning techniques were used in order to generate robust descriptors. Specifically, the authors proposed a training phase using Support Vector Machines (SVM) over possible translations of the HOG-based word representation. The SVM step provides significant improvement on the performance of the method at the expense of increased time requirements on the online phase of the algorithm (query time). Even though a training phase is involved, this approach can be considered unsupervised since the training is performed on an augmentation set of the initial query image (no training data is necessary except for the query image). Furthermore, the authors made use of Product Quantization

with the aim to minimize time and memory requirements.

The most characteristic method of the second category (producing a set of features) concerns the work of Rath and Manmatha [141]. In this work, the authors proposed the description of each word image using a sequence of simple geometric (statistical) features, computed at each column of the image, followed by the application of dynamic time warping (DTW) between pairs of sequences for the production of the final ranking. The computation of the DTW score (due to the complexity of the algorithm) increases the processing time for the creation of the final ranking list. In a more recent work, Zagoris et al. [190] presented an approach which shares ideas with classical keypoint detection techniques, referred to as Document-Specific Local Features. The focus of this work is given on finding keypoints appropriate for text images and a gradient-based descriptor is extracted at each keypoint. The matching is performed by comparing descriptors as well as their spatial correlation. The matching stage may cause significant overhead for a large set of keypoints, leading to slow retrieval time. Howe [86], proposed a flexible inkball (template) model which allows deformed template matching. With the term inkball model, Howe referred to a set of connected nodes that describe the word images, where nodes correspond to points belonging to text strokes. Due to this formulation, all the computational effort is on matching procedure, which is performed with an iterative energy minimization algorithm following a two step procedure, query models are fitted to the target words and each target word is transformed to a model for reverse verification.

One of the advantages for choosing a fixed feature vector for the representation of a word image, is the ability to apply a simple similarity measure (e.g. Euclidean distance) for the retrieval phase, thus leading to reduced time requirements. Consequently, methods belonging to this category can be straightforwardly applied to large collections of documents. On the other hand, approaches which use a set of features (such as sequences of features [141] or graphs of inner structure [86]), are considered more robust to variations of the writing style at the expense of larger time requirements for retrieval due to the increased complexity of the appropriate similarity measure.

2.3.1 Datasets and Metrics

The performance of the word spotting methods is recorded in terms of the Precision at Top 5 Retrieved words (P@5) as well as the Mean Average Precision (MAP). Specifically the aforementioned metrics are defined as follows:

- **P@5:** Precision at top 5 retrieved words. P@k measure is defined as:

$$\text{P@k} = \frac{|\{\text{relevant words}\} \cap \{\text{k retrieved words}\}|}{|\{\text{k retrieved words}\}|} \tag{2.4}$$

- **MAP:** Mean Average Precision. The MAP for a set of queries is the mean of the average precision scores for each query. The Average Precision for a query is defined as:

$$\text{AP} = \frac{\sum_{k=1}^{n}(P(k) \times \text{rel}(k))}{|\{\text{relevant words}\}|} \tag{2.5}$$

  where rel(k) takes the value 1 if a word is relevant at rank k and the value 0 otherwise.

We briefly describe the set of considered KWS datasets for the case of word-level segmented images. Each dataset is accurately defined by the set of the word images, as

well as the query set. The QbE scenario requires the existence of query images, which is not always guaranteed.

The datasets, able to support the QbE scenario, are the following: **George Washington Dataset**[2]**:** This dataset is the well-known collection of writings of George Washington, consisting of 20 pages segmented into 4860 words. A standard partition or query selection is not available for GW dataset and subsequently the majority of the reported results are not comparable. To this end, we distinguish three different setups. The first one is used only for parameter selection whereas the other two for comparison. **GW1:** Fifteen words selected as in [158] and all instances of each of the 15 words were considered as queries (306 queries in total). **GW2:** Words with ten or more instances and three or more characters are selected as queries as in [6], resulting in 1847 queries in total. Several recent works on learning-free QbE KWS report results on this setup [191],[99],[6]. **GW3:** A fourfold cross validation setup is employed, using the exact same partitions as in [9]. Words with at least two instances in the test sets are selected as queries. This setup is adopted by the majority of the recent learning-based methods, since it is split to training and testing partitions. Table 2.1 reports the average number of queries and words over the four different partitions. We adopt this setup as the default one, reffered to as **GW**, when we evaluate the explored deep learning approaches.

IAM[3]**:** This challenging dataset consists of 115.320 words written by 657 writers. The official partition for writer independent text line recognition is selected and the queries are extracted according to [9], excluding stopwords and words that appear only once. IAM dataset is a popular choice for evaluating learning-based methods.

Bentham14: This dataset was part of the ICFHR 2014 H-KWS competition [134] and includes manuscripts in English written by Jeremy Bentham himself as well as by Bentham's secretarial staff. It consists of 10370 segmented word images from 50 document images and 320 query images.

Modern14: This dataset was also part of the ICFHR 2014 H-KWS competition [134] and includes handwritten documents written in four languages (English, French, German and Greek). It consists of 14754 segmented word images from 100 document images (25 for each language) and 300 image queries.

Bentham15: This dataset was part of the ICDAR 2015 KWS competition [137] and is also a part of the Bentham collection, although a larger and more challenging set is used than the one in ICFHR 2014 competition. This dataset consists of 1421 image queries and 13657 segmented word images from 70 document images.

Botany16: It was part of ICFHR 2016 H-KWS competition [135] and the test set consists of 20 documents written in English. A training set of documents is also provided for learning-based techniques.

Konzilsprotokolle16: This dataset was also part of ICFHR 2016 H-KWS competition [135] and the test set consists of 20 documents written in German. Similar to Botany dataset, there exists an extra training set of documents.

Concerning QbE scenario, we also should take into account whether the query image is considered relevant to itself. This information (denoted as the *qr* property), along with the number of queries and segmented words for each dataset are summarized in Table 2.1. Examples of segmented words from each collection are shown in Figure 2.2. Two different instances of the same word are presented in order to highlight the variations of the aforementioned collections.

[2] `http://www.fki.inf.unibe.ch/databases/iam-historical-document-database/washington-database`

[3] `http://www.fki.inf.unibe.ch/databases/iam-handwriting-database`

Table 2.1: Properties of the Datasets/Setups used for evaluation.

Dataset/Setup	#words	#queries	qr
GW1[158]	4860	306	yes
GW2[6]	4860	1847	no
GW3[9]	1215	901	no
IAM[9]	13752	4030	no
Bentham14[134]	10370	320	yes
Modern14[134]	14754	300	yes
Bentham15[137]	13657	1421	no
Botany16[135]	3230	150	yes
Konzils16[135]	3534	200	yes

(a) (b) (c)

(d) (e) (f)

Figure 2.2: Two indicative examples of the same word in each dataset: (a) George Washington, (b) IAM, (c) Bentham, (d) Modern, (e) Botany and (f) Konzilsprotokolle

2.3.2 Attribute-based Representations

Recent developments on the KWS are focused on the case of extracting a word representation, i.e. holistic approaches on segmented word images, by training a model to predict specific character-related attributes. Normally, training cannot be straightforwardly performed for KWS applications, without borrowing from HTR and training with respect to a sequential target of variable length. However, such an approach is simply a different evaluation of an HTR task. Keyword spotting, ideally, should be complimentary to text recognition. Towards this direction, Almazan et al. [9] proposed an fixed-length attribute-based representation for each word, enabling the application of machine learning techniques to KWS, independently of HTR. To this day, several similar attribute-based representations [185, 167, 59] have been proposed but we will not dwell on the details of these representations, since their main purpose is common.

Of course, if we do not take into account that a word is consisted of characters, one can define a unique label to each word. This would lead to a typical classification problem, even though there would be thousands different classes. However, two important problems arise with this formulation. First, the classifier would only learn to predict the known classes. Therefore, if a sample is presented to the classifier and does not correspond to one of the already known classes, presented to the classifier at training time, the classifier would falsely assign the sample into one of the existing classes. Closely related to the first problem, even if no unknown classes are requested, the existing classes cannot retain the underlying distribution of the words. Such distribution can be helpful, if for example, we want to find words with the same prefix. On the contrary, if the classes are defined by a set of attributes (e.g. the existence and the position of a set of characters), unseen word classes can be correctly described by estimating the appropriate attributes. Such approach is commonly referred to as zero-shot learning with attributes.

In the following, we will describe the attribute-based representation proposed by Almazan along with its initial application to keyword spotting and finally we will present its adaptation to neural networks, proposed by Sudholt et al. [166].

Pyramid Histogram of Characters (PHOC)

Almazan et al. [9], introduced a character attribute embedding which encodes word strings. The embedding is dubbed Pyramidal Histogram of Characters (PHOC) and essentially encodes the presence or absence of characters in certain sections of a string in a pyramidal fashion. The embedding is in fact a binary embedding, since it encodes the existence of specific characters. The embedding is defined by two factors: 1) The character set. i.e. which unigrams would be treated as attributes. It should be noted that bigrams were also considered. 2) The number of pyramid levels. Pyramidal structure is used in order to encode spatial information along with the presence or absence of characters. At the first level of the PHOC, all unigrams in the string are considered and the respective attribute is set to 1 if the corresponding unigram appears at least once in the word. In the following levels, the string is equally split into as many segments as the levels. For each segment, the same logic of the first level is applied. The PHOC rationale can be easily grasped by the visual overview of Figure 2.3.

Apart from the proposed label encoding, Almazan et al. [9] proposed a system in order to accurately estimating an attribute embedding from a word image. Specifically, a Fisher Vector descriptor is extracted from each word image. At the same time, the transcription of each word is encoded using a pyramidal histogram of characters (PHOC) descriptor, which is the attribute-based embedding of interest. Using the PHOC representation as

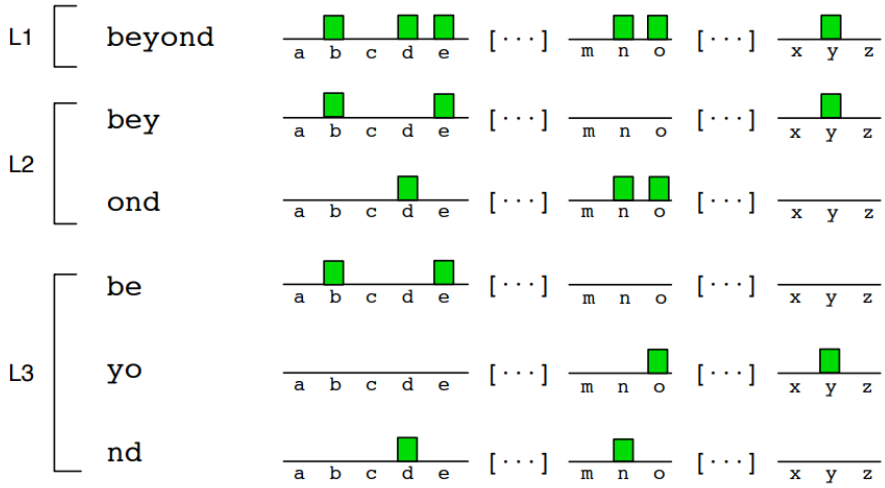


Figure 2.3: PHOC histogram of a word at 3 levels. The final PHOC histogram is the concatenation of the depicted histograms [9].

labeling, a SVM training is performed allowing the system to learn to discriminate the position of the characters (or bigrams of characters). A Canonical Correlation Analysis is performed in order to generate the common subspace between the PHOC and the estimated attributes. The final descriptor produced is a fixed-length feature vector which can be used on both QbE and QbS scenarios. The overview of the described system is depicted at Figure 2.4. For a more detailed description, we recommend the reader to see [9].

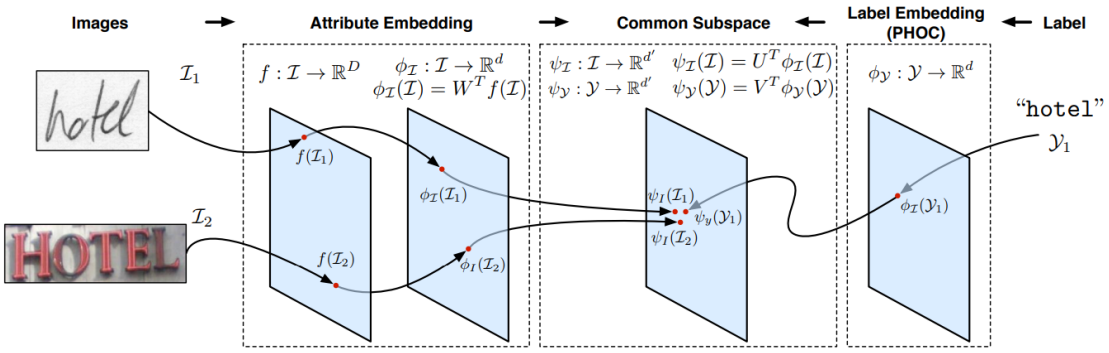


Figure 2.4: Overview of the system proposed by [9]

PHOCNet

Sudholt et al. combined the merits of the PHOC embedding with the representational power of Convolutional Neural Networks. To this end, the authors build a network akin to VGG-16 [160], trained to predict the PHOC embeddings. The network input is a word image, without any size restriction, and the corresponding label is the binary attributes of PHOC representation of the respective word string. In order to generate binary estimations, the network output is topped with a sigmoid activation and the binary cross entropy loss is selected.

One notable addition to the classical feed-forward CNN architecture is the Spatial Pyramid Pooling (SPP) layer [76] stacked between the convolution layers and the fully connected layers. The importance of this layer is to enable size-invariant inputs. Specifically, the output of the convolutional layers is dependent to the size of the input image. However, the upcoming fully connected layers require a fixed-sized vector. In order to vectorize the variable-sized convolutional feature map, a pyramidal pooling scheme is employed, where the feature map is split into disjoint segments along both height and width axis. While increasing the levels, the number of segments is also increased. Due to the nature of words, more segments are generated along width axis compared to height. Therefore, a fixed sized of segments is extracted, which depends on the number of levels and the number of segments at each level, and subsequently a max-pooling operation is performed at each level. The max-pooling operation reduces each segmented 3-d feature map of size $h \times w \times d$, where h is the height, w is the width and d is depth (corresponds to the number of channels of the last convolutional layer), into a d-sized vector. It should be noted that SPP produces very large vectorized representations, which may be impractical. To this end, the authors published a variation of the aforementioned network, where the pyramidal layer segments the feature map only horizontally. This variation is called Temporal Pyramid Pooling (TPP) [167]. The PHOCNet overview is depicted at Figure 6.12.

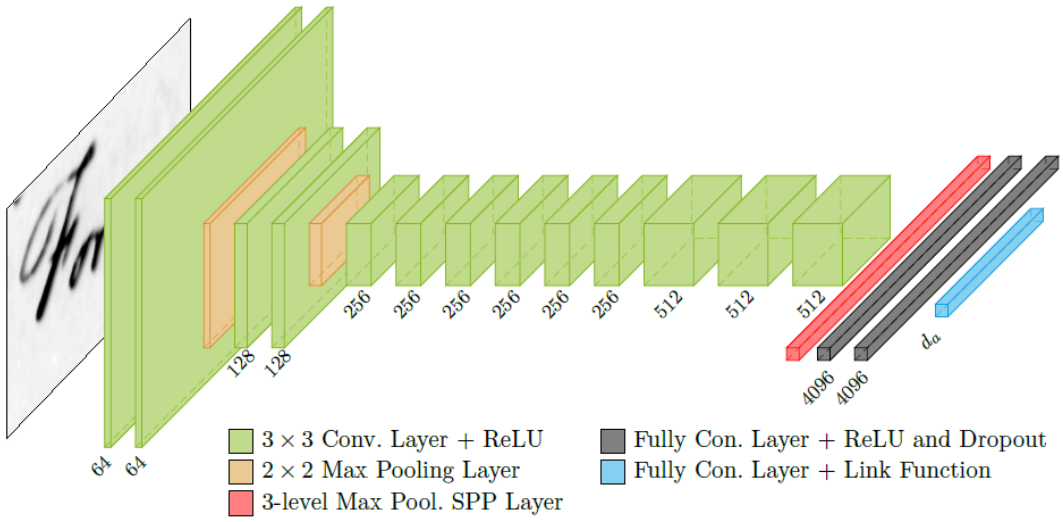


Figure 2.5: Visualization of the PHOCNet architecture: Green layers represent 3×3 convolutional layers with the respective number of filters shown underneath. Orange layers depict 2×2 max pooling layers with a stride of 2. The red layer indicates a 3- level SPP layer [166].

Chapter 3

Character Recognition

Optical Character Recognition (OCR) was a pivotal area of research for many years and the character classification task (sepcifically the MNIST dataset, consisted of handwritten digits) was the norm for validating new-developed computer vision and pattern recognition techniques.

The last decade the character classification problem is considered as a rather easy task, since segmented characters (typewritten or even handwritten) can be classified, almost perfectly, by classical machine learning techniques, such as Support Vector Machines. It is important to note that the majority of proprietary software for text recognition is still based on OCR approaches due to their simplicity, since they are oriented towards recognizing typewritten, industry-related documents, e.g. invoices.

Nevertheless, the character classification task can be useful to evaluate if a feature extraction method is appropriate for extracting descriptive text-related features. To this end, we developed a feature descriptor and evaluated it on several datasets of segmented characters. Moreover, we will describe an approach for recognizing the existing characters classes with the user's help when there is no information about the document collection (character set, language e.t.c.). Note that due to the users' involvement, we aim to minimize their interaction and avoid a tedious transcription process.

3.1 Projections of Oriented Gradients (POGs)

A widely used approach in isolated character recognition systems is to follow a two step schema: a) feature extraction and b) classification. Selection of an appropriate feature extraction method is of great importance for achieving high recognition performance. The main goal of feature extraction techniques is to extract representative information from the original image in a compact and robust way. In order to improve the effectiveness and efficiency of the subsequent classification stage, the selected features should minimize the within class pattern variability, while enhancing the separability of the classes.

Aiming towards a compact yet discriminate descriptor, we developed a novel feature extraction technique for character recognition. The proposed approach attempts to overcome the problem of high variation, mainly in handwriting, which is accentuated by the use of local features. In contrast to using a grid-based technique for local features extraction, a more global interpretation of the image is adopted. Specifically, the proposed feature extraction is a projection-based technique, which resembles the Radon transform [20]. In addition, an attempt is made to incorporate popular ideas from classical high-performing feature extraction methods, such as Histogram of Oriented Gradients (HOG)

[37], based on the fact that the gradient information is proven very useful for a variety of object recognition problems. Consequently, instead of features derived from local gradient, as in HOGs, a projection-based feature extraction for each gradient orientation, which is represented as a binary image, is adopted.

3.1.1 Feature Extraction

The proposed feature extraction technique, Projections of Oriented Gradients (POG), consists of two stages:

1. Computation of the oriented gradients of the image.
2. Extraction of a projection-based descriptor for each image gradient as well as for the initial image.

Before the computation of oriented gradients, a pre-processing step for noise reduction is applied. This is accomplished using a two-dimensional median filter (3×3 neighborhood) on the original image.

Gradient Orientation

This stage is essential for capturing informative details of the image, such as the change in the direction of the contour, through the representation of the gradient orientation as binary images.

The computation of the directional gradients G_x and G_y of the image $I(x, y)$, along x-axis (horizontal) and y-axis (vertical) respectively, is performed using the following filter kernels: $[-1\,0\,1]$ and $[-1\,0\,1]^T$. In order to compute the gradient orientation at each pixel, a transformation into polar coordinates is performed through Eq. 3.1,3.2. Only the orientation $\angle G$ is of interest and a wrapping is performed so that the orientation values lie on the interval $[0, 180°)$ instead of $[0, 360°)$ ("unsigned" gradient as in [37]).

$$|G(x, y)| = \sqrt{(G_x^2(x, y) + G_y^2(x, y))} \tag{3.1}$$

$$\angle G(x, y) = arctan\Big(\frac{G_y(x, y)}{G_x(x, y)}\Big) \tag{3.2}$$

The possible orientations, due to the selected gradient filter, are four (0°, 45°, 90° and 135°) and thus four binary images, that represent the gradient orientation, are constructed: $G_0 = (\angle G(x, y) = 0°)$, $G_{45} = (\angle G(x, y) = 45°)$, $G_{90} = (\angle G(x, y) = 90°)$ and $G_{135} = (\angle G(x, y) = 135°)$.

Projection-Based Descriptor

The basic concept of the projection-based feature extraction is to decompose the binary image into several projections under selected angles, imitating the Radon transform. The projections are n_θ in total, sampled every 180° / n_θ , i.e. the angles of the projections are $\theta_k = k\frac{180°}{n_\theta}$, $k \in [0, n_\theta - 1]$. Let $H \times W$ denote the size of the image, (x_i, y_i) denote every foreground pixel of the image $(i = 1, \ldots, N)$, (x_c, y_c) denote the center of the image and θ denote the chosen angle of projection. The projection p_i of each pixel into the selected angle vector is computed according to Eq. 3.3.

$$p_i = (x_i - x_c)\cos(\theta) + (y_i - y_c)\sin(\theta), \quad i \in [1, N] \tag{3.3}$$

Finally, in order to obtain a K-length vector that represents the projection, a soft binning technique (K bins) is applied for p_i in the interval:
$\left[-\frac{1}{2}\sqrt{H^2+W^2}, \frac{1}{2}\sqrt{H^2+W^2} \right]$.

Our goal is to eliminate the redundancy and the excessive variability of the image. However, the resulting matrix of the Radon transform usually is rather large to be a useful descriptor. Furthermore, possible variations of the same character may translate into slight distortion of the projections or even spikes due to noise. Thus, each projection subjects to a smoothing-like procedure in order to reduce unnecessary variations. The most descriptive information corresponds to smoothed regions of relatively high pixel concentration, or equivalently to the low frequency coefficients of the Fourier Transform of the projection. Therefore, after computing the Discrete Fourier Transform coefficients $c_j\,, j \in [0, K-1]$ of the projection vector, only the first n_c+1 are used to form the feature vector while the remaining are discarded. The aforementioned steps are depicted in Fig. 3.1 for a specific angle ($\theta = 45°$).

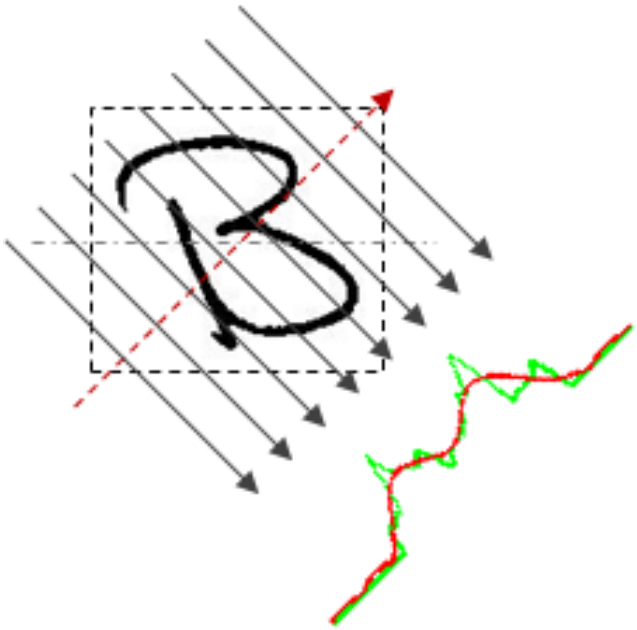


Figure 3.1: Visualization of a projection ($\theta = 45°$) and its smoothing (low-frequency components)

Subsequently, a normalization with regard to the number of pixels in each image is applied by dividing each Fourier coefficient by N (foreground pixels), or equivalently by c_0, while discarding c_0. The complex feature vector for each projection is computed according to the following equation.

$$f_j = c_j/N\,, \quad j = 1, \ldots, n_c \tag{3.4}$$

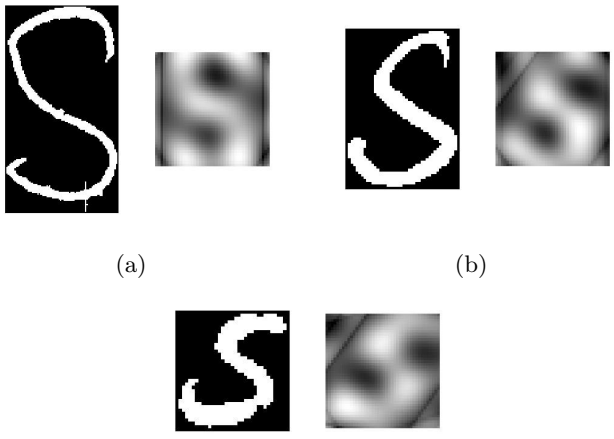
The final feature vector is a concatenation of the real and imaginary parts of the complex feature vector for each projection. It is clear that a selection of a subset of the Fourier coefficients results to projection length independence and, consequently, to image size independence.

It should be noted that the described procedure is equivalent to a circular low-pass filter in 2-D frequency domain via the Projection-Slice theorem [43]. Specifically, each of the

computed projections corresponds to a slice in the frequency domain via the Fourier transform of the projection, thus providing a more accurate sampling in low-frequencies comparing to two-dimensional Discrete Fourier Transform. Additionally, due to the applied lowpass filtering, only a few (n_θ) projections (angular sampling) suffice for representing the low-frequency domain.

Reconstruction And Visualization

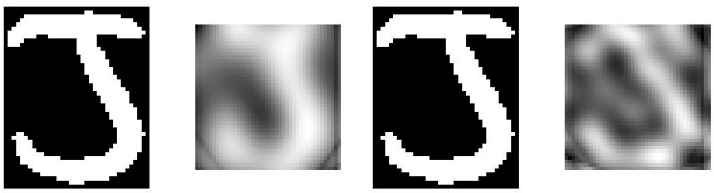
A useful property of the described method is the approximate reconstruction of the original image via the inverse Radon transform, after interpolating each projection to a specific length using the inverse Fourier transform. Therefore, a normalized approximation of the image is generated, which provides a visualization of the preserved information. Fig. 3.11 depicts the visualization of the described feature vector for three instances of the same class. It can be observed that the within-class variations seem to be reduced. Additionally, Fig. 3.3 shows the impact of the number of Fourier coefficients. It is obvious that an increase in the number of coefficients, corresponds to a more detailed representation, though the additional details may be undesirable (noise or redundant variations of the same class).



(a) (b)

(c)

Figure 3.2: Examples of reconstruction for different instances of the same class ($n_\theta = 6$ & $n_c = 2$)



(a) $n_c = 2$ (b) $n_c = 4$

Figure 3.3: Examples of reconstruction using different number of projection coefficients ($n_\theta = 6$)

Final Descriptor

The final feature vector is the concatenation of the projection-based descriptor applied in the initial binary image as well as in each of the four gradient orientation images (G_0,G_{45},G_{90} and G_{135}), as it is depicted in Fig. 3.4. Overall, the length of the descriptor is: 5 (images) $\times$ n_θ (projections) $\times$ $2n_c$ (Fourier coefficients). It should be noted that the proposed technique is not restricted to binary images since every substep can be easily generalized to grayscale images.

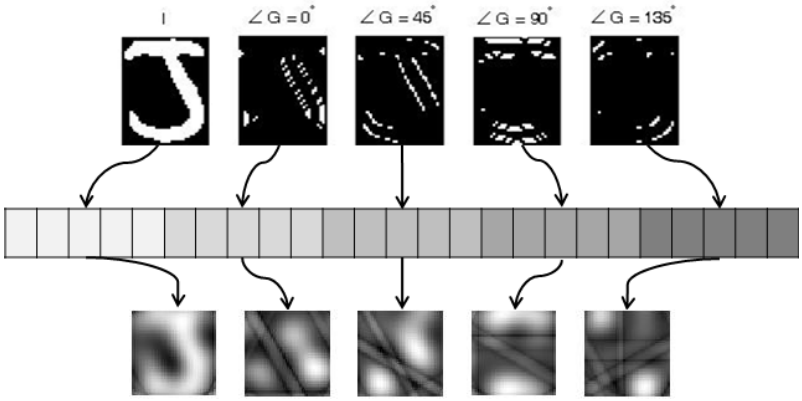


Figure 3.4: Final descriptor as the concatenation of features of the initial image and its oriented gradients and their corresponding visualizations

3.1.2 Experimental Evaluation

Data Setup

For our experiments two handwritten character databases were used, the CIL Database [178] and the CEDAR character Database CD-ROM-1 [89], as well as the database GRPOLY-DB [55] of machine printed Historical Greek characters.

The CIL database comprises samples of 56 Greek handwritten characters written by 125 Greek writers. Every writer contributed 5 samples of each letter, thus resulting to a database of 625 variations per letter and an overall of 35000 isolated and labeled characters. Due to the similarity of several classes, especially due to size invariance of the methodology, a total of 10 pairs of classes were merged, randomly selecting 625 characters from each pair, as in [179]. The resulting 46 classes consist of 28750 characters, among which 23000 characters were used for training and the remaining 5750 characters for testing.

The CEDAR database consists of samples of 52 English handwritten characters, where 19145 characters were used for training and 2183 characters for testing. For the CEDAR database, four scenarios are distinguished for our experiments: 1) 26 classes of uppercase characters, 2) 26 classes of lowercase characters, 3) 35 classes after merging similar classes, as in [179], and 4) 52 classes without merging

The GRPOLY-DB machine-printed dataset consists of isolated Greek polytonic characters extracted from the Hellenic parliament session proceedings. The Greek polytonic system has a variety of diacritic marks leading to a large number of classes. Only characters belonging to classes with at least 30 instances were selected, resulting to 125 classes in total. Two different scenarios were defined. According to the first scenario (SC-1), all instances were used (143051 instances) while at the second scenario (SC-2), only 30

randomly selected instances per class were used (3750 instances). In each scenario a 5-fold cross-validation was applied in order to evaluate the recognition rate.

Compared Methods

The state-of-the art techniques that were used in order to compare the efficiency of the proposed methodology are briefly described below:

- Adaptive zoning features method [54] (AdWin). According to this method, features are extracted after adjusting the position of every zone based on local pattern information, i.e. the maximization of the local pixel density around each zone. The classification for this feature descriptor is performed using a kNN classifier ($k = 3$).
- Histograms of Oriented Gradients method [37] (HOG). For each zone formed from a regular grid, a histogram of the local gradient orientation is constructed, where each bin corresponds to the frequency of a specific orientation in the zone. For binary images and the central difference gradient operator the orientations are limited, as it has been described previously. Multiclass SVM with RBF kernel is used as a classifier.
- Foreground sub-sampling method [179]. This method iteratively subdivides the image into four sub-images with approximately equal number of foreground pixels. The feature vector consists of the coordinates of the division points for a selected level of granularity (iteration steps). Further improvement to recognition rates is achieved by introducing a two-stage classification scheme (Vam2) comparing to a single step SVM classification (Vam1), where the information of different granularities is utilized.

3.1.3 Parameter Selection

The number of projections is not an essential parameter due to the lowpass filter equivalence of the image descriptor, thus, in order to avoid oversampling, we choose $n_\theta = 6$, or equivalently, we extract a projection every 30°.

Classification is performed using Support Vector Machines (SVM) [34], which is a machine learning technique for binary classification problems. The multiclass classification problem is reduced to multiple binary classification problems, applying the *one-versus-one* scheme. Specifically, a SVM is trained for each pair of classes and the classification is performed by voting.

In order to evaluate the efficiency of the descriptor while changing the number of the complex Fourier coefficients (n_c) , a linear SVM classifier is applied for the CEDAR (only for the merged scenario), CIL and GRPOLY-DB (only SC-2) datasets. The recognition results are depicted in Fig. 3.5. It is obvious that for both handwritten character databases the optimal choice is $n_c = 3$ (POG3). A further increase of the coefficients corresponds to an introduction of unnecessary details, hence increasing the within class variation and consequently hindering the classification step. For the machine-printed database, an increased number of complex coefficients is optimal ($n_c = 6$ - POG6), compared to the previous experiments. This is to be expected due to the large number of classes (125) as well as the minor differences of a variety of polytonic Greek diacritic marks (for example $\grave{\alpha}, \acute{\alpha}, \tilde{\alpha}$). Furthermore, the variance between each class is limited, thus increasing the coefficients does not correspond to a significant decrease in the recognition rate as in the CEDAR and CIL databases.

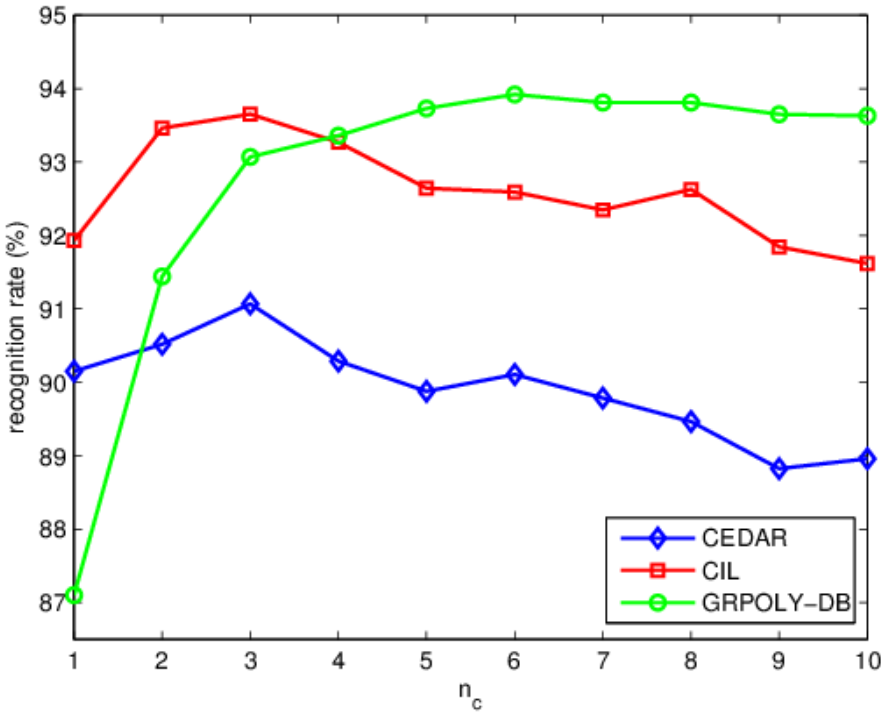


Figure 3.5: Recognition rate for different number of Fourier coefficients

The final classifier is a multi-class (one-vs-one) SVM with RBF (Radial Basis Function) kernel, aiming to further increase the performance. A grid search was performed in order to find the optimal values for the parameters γ and C using cross-validation. For our experiments, the selected values were 0.05 and 8 for γ and C respectively.

Classification Results

Table 3.1 shows the overall comparison results for the CEDAR database (4 scenarios), while Table 3.2 shows the recognition results for the CIL database. In both databases the proposed method (POG3) clearly outperforms the other techniques with only one classification step. Only the two-stage classification scheme (Vam2) in [179] performs better in the CEDAR database for the blended scenarios (both lowercase and uppercase characters), although has slightly worse performance in the remaining scenarios and in the CIL database. Nonetheless, the described technique in [179], as a feature descriptor along with the typical one-step classification (Vam1), displays a significant decrease in the recognition efficiency over the proposed one.

	AdWin	**HOG**	**Vam1**	**Vam2**	**POG3**
Uppercase (26 cl.)	85.30%	94.66%	93.78%	95.90%	96.04%
Lowercase (26 cl.)	83.70%	91.17%	89.79%	93.50%	93.68%
All (52 cl.)	69.90%	81.68%	78.42%	85.11%	82.73%
Merged (35 cl.)	82.46%	91.8%	90.70%	94.73%	93.59%

Table 3.1: Classification results for the CEDAR database

AdWin	HOG	Vam1	Vam2	POG3
87.97%	94.02%	92.53%	95.63%	96.14%

Table 3.2: Recognition results for the CIL database

	AdWin	HOG	POG3	POG6
SC-1	97.71%	98.37%	98.49%	98.60%
SC-2	88.69%	92.00%	93.67%	94.35%

Table 3.3: Recognition results for the GRPOLY-DB database

The classification results for the two scenarios of the GRPOLY-DB database are shown in Table 3.3. Specifically, we evaluate both the cases for $n_c = 3$ (POG3), which was the optimal for the other databases and the resulting descriptor is fairly small (180 features), and for $n_c = 6$ (POG6), which was chosen as the optimal for this database, even though it is a four times larger descriptor. As with the experimental results in the two previous databases, the proposed technique exhibits superior performance, while the POG6 descriptor shows better performance compared to POG3 (especially in SC-2, which is a more qualitative scenario) at the cost of the length of the feature vector.

Overall, we evaluated a novel image descriptor, based on radon transform, on the character classification problem. The experimental results support the claim that the descriptor extracts highly descriptive visual features for text images. In the following chapter, we will apply this descriptor on the more challenging task of keyword spotting in order to fully evaluate its descriptive capability.

3.2 Discovering Character Classes: A User-in-the-Loop Case

Even though character level classification produced noteworthy results (over 90% accuracy) it is rather impractical, since character segmentation has not been addressed effectively to this date. In practice, handwritten text usually mixes successive characters into bigrams, trigrams e.t.c. and therefore a vast number of context-dependent variations emerge. On the other hand, typewritten text can be segmented into characters with little effort, while the main hindrance is the visual noise usually seen at degraded historical documents.

Historical typewritten documents can be considered an important source of novel information for scholars in order to study recent history. Large collections of typewritten documents still remain unexploited since existing recognition techniques cannot be successfully applied to the cases of low-quality historical typewritten documents that suffer from several problems such as the existence of faint and degraded characters, broken or merged characters, stains, tears and punch holes, quality problems due to ageing. Moreover, a great part of these collections may survive only as a carbon copy of the original and this poses additional challenges concerning image quality.

Existing approaches related to the analysis and recognition of historical typewritten documents mainly focus on the enhancement of the degraded typewritten characters before using a conventional OCR engine [121, 2, 1, 61]. As it is reported in [1, 61], the recognition accuracy can be increased from 77.2% to 91.3% [1] or from 54.5% to 78.2% [61] when a suitable enhancement of text areas is applied. A new framework for recognizing particularly challenging collections of historical documents, such as typewritten documents of World War II is proposed in [155]. A recognition approach is proposed which can be

trained on specific document collections (e.g. produced by similar typewriters) with minimal user interaction. It is based on the combination of collection-independent domain knowledge (such as typography conventions) with human feedback in an iterative manner to gradually refine the system's understanding of the unique characteristics of the specific document collections.

We developed an approach based on the basic principle of [155] which mainly focuses on (i) having minimum but effective user-interaction, (ii) introducing a robust clustering scheme and (iii) involving a segmentation-free recognition module in order to overcome segmentation problems. User-interaction is a common practice in many OCR applications in order to correct the result and re-train the corresponding classifiers. In [25], a "carpet session" is involved in order to sort all detected characters in a way that similar characters appear close to each other. The user has to identify all errors and thus automatically approve all other characters as valid. To eliminate this tedious process, we propose to involve a sub-clustering procedure in order to have the user checking only a limited number of representative characters (e.g. 4) per character class. The exact re-arrangement of the clusters/classes is done via a semi-supervised clustering scheme which uses the provided information from the user as a rough labeling. The workflow of the proposed methodology is presented in Figure 3.6 and its components are described in detail in the upcoming subsections.

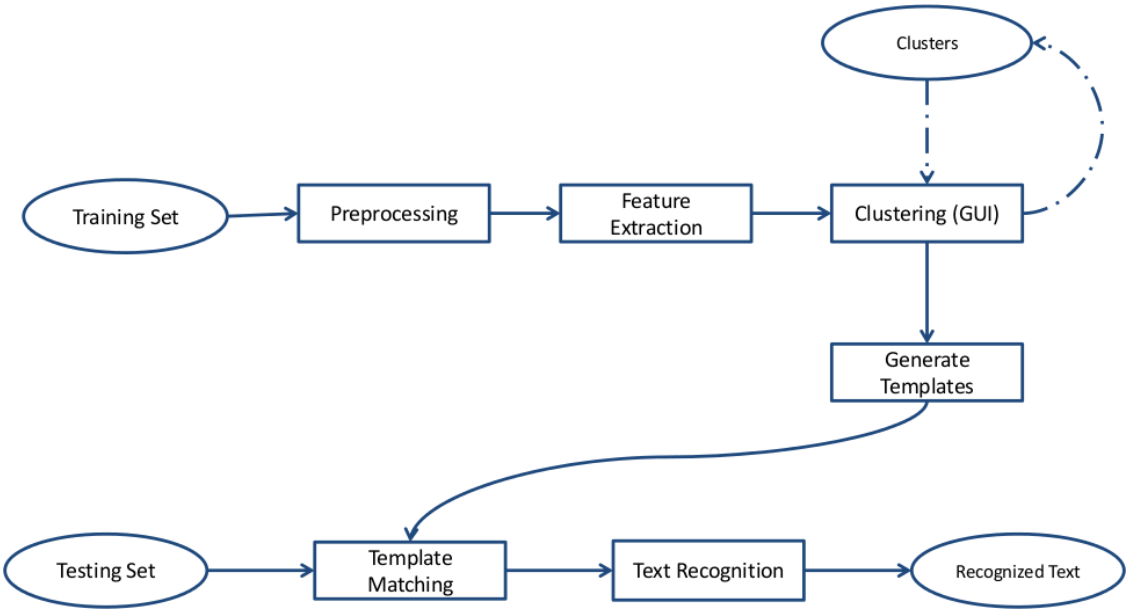


Figure 3.6: System Overview

3.2.1 Preprocessing

The first stage of the proposed system consists of several preprocessing steps in order to extract possible characters of the document as connected components. The first two steps are image filtering for noise reduction and binarization followed by a connected components extraction. The main goal of this stage is to extract components of the image, which correspond to characters with high certainty. Thus, there is no need to extract all possible characters, which in practice is a challenging task. It is sufficient to extract a satisfying large set of characters in order to perform clustering on them. Later,

the discarded characters can be retrieved at the recognition stage, if they correspond to a valid class, generated by the semi-supervised clustering. This approach can tolerate a more rough and simplified approximation of each step of the preprocessing stage. In order to extract a useful set of characters, capable of a meaningful clustering, a set of images, instead of only one, is preferred.

Alternating Sequential Filters

First, we apply a noise reduction technique, which is essential especially for the case of carbon-copied documents. Morphological *Alternating Sequential Filters (ASF)* [122] are selected for this purpose as they provide an effective smoothing of the image while preserving its topology, i.e. the characters shape. The ASF filtering is formulated as follows:

$$\Psi_n(g) = \beta_n(\alpha_n ... (\beta_2(\alpha_2(\beta_1(\alpha_1(g))))))\text{, } n = 1, 2, 3, ... \tag{3.5}$$

$$\alpha_r(g) = \rho^-(g \ominus rB|g), \tag{3.6}$$

$$\beta_r(g) = \rho^+(g \oplus rB|g) \tag{3.7}$$

where ρ^- is *Reconstruction Opening* and ρ^+ is *Reconstruction Closing*.

We choose to use ASF up to level 2, i.e. $n = 2$, while the initial structural element B is a disk of radius 1. The result of the ASF filtering over an image is depicted in Figure 3.7(b).

Image Binarization

The selected binarization method is Sauvola's approach for adaptive thresholding [156].However, as it was mentioned before, we only need to extract a satisfying number of connected components, corresponding to characters, and not a fine binarization result. A rough estimation of the foreground regions, where oversegmenting of a character binary mask is avoided, is preferable. Thus, Sauvola's parameters are fixed, using $k = 0.2$ and window size 50×50. The binarization of a selected document part is depicted in Figure 3.7(c). To ensure that punctuations are merged with the character's main body and thus forming a single connected component, a morphological dilation with a rectangular structural element (longer in y-axis, e.g. 1×3) is applied at the binarized image.

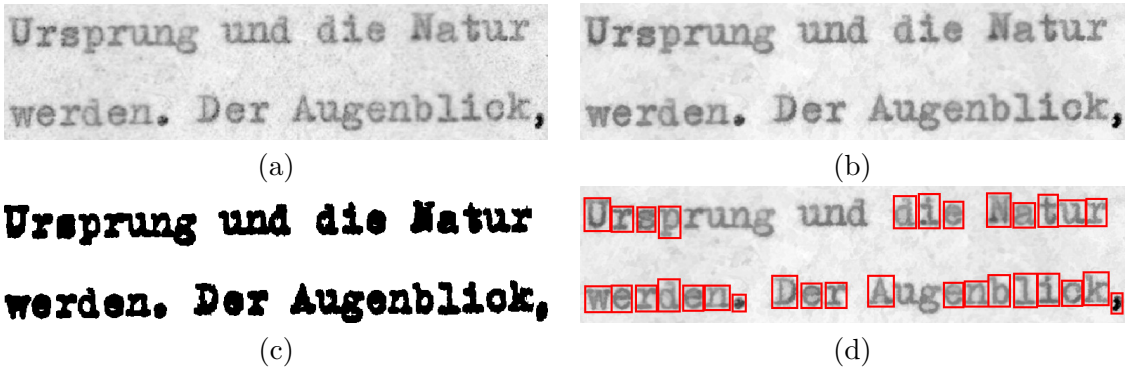


Figure 3.7: Summarization of the preprocessing steps. (a)Initial gray-scale part of a typewritten document (b)Result of the ASF filtering (c)Sauvola's binarization (d)Connected components selection

Connected Components Extraction

The final step of the preprocessing is the extraction of appropriate connected components (8-connectivity) that should be, most likely, characters. Although the connected components of the binarized image will vary greatly in size, a large number of the components correspond to single characters with similar size. Under this assumption we calculate the median width w_m and height h_m of the bounding boxes of all the connected components and we assume that all the possible characters are connected components with similar size. Thus, a connected component within a specified range of bounding box width and height (related to w_m,h_m) is accepted as a character candidate. The rest connected components (noise, pictures in the documents or group of characters) are discarded.

Finally, upon finding the bounding box of the selected connected components, we extract the character candidates over the gray-scale image (after ASF filtering), as shown in Figure 3.7(d).The set of the extracted character candidates is the input of the next stage in order to perform the semi-supervised clustering.

3.2.2 Character Clustering

In this section we will describe our approach for a user-assisted system for effective character clustering. The proposed semi-supervised clustering is performed on the feature descriptors of the character images, using Histogram of Oriented Gradients [37]. Our main focus is on the user perspective, trying to maintain a simple yet effective interaction. Specifically, we define a set of simple actions for the user to choose in order to assist in the improvement of the existing clustering. For this purpose, we developed a GUI which will be described in detail further on.

Feature Extraction

Instead of using the raw image of a character, a feature descriptor would be more convenient in terms of simplicity and speed for the upcoming clustering. Consequently for each character candidate a feature vector is extracted using Histograms of Oriented Gradients (HOG) [37]. Before using the HOG feature extraction technique, a uniform padding of the characters is applied in order to impose the same size in each character image. The (common) size of the final padded image corresponds to the maximum possible width/height amongst all the character candidates.

Histograms of Oriented Gradients (HOG) is a state-of-the-art feature extraction technique. HOG descriptors are widely used due to their effectiveness in recognition tasks, while retaining a simple and fast implementation. The main concept is to encode the direction of the gradient for a set of regions, which called cells, as a local histogram.The final descriptor is the concatenation of all the local histograms.

The computation of the directional gradients G_x and G_y of the image $I(x, y)$, along x-axis (horizontal) and y-axis (vertical) respectively, is performed using the following filter kernels: $[-1\,0\,1]$ and $[-1\,0\,1]^T$. In order to compute the gradient orientation at each pixel, a transformation into polar coordinates is performed through Eq. 3.8, 3.9. Only the orientation $\angle G$ is of interest and a wrapping is performed so that the orientation values lie on the interval $[0, 180°)$ instead of $[0, 360°)$ ("unsigned" gradient as in [37]).

$$|G(x, y)| = \sqrt{(G_x^2(x, y) + G_y^2(x, y))} \tag{3.8}$$

$$\angle G(x,y) = arctan\Big(\frac{G_y(x,y)}{G_x(x,y)}\Big) \tag{3.9}$$

Cells are extracted using a rectangular grid upon the image and within each cell a histogram of a set of orientations (bins) is constructed. For our implementation we selected a 5×7 grid, obtaining a total of 35 cells. The local histogram is computed by each pixel voting in the corresponding orientation bin with weight equal to its magnitude. The number of orientation bins is predefined and for our problem we chose 6 bins (30° range). Due to the simplicity of the problem no block-normalization is performed.

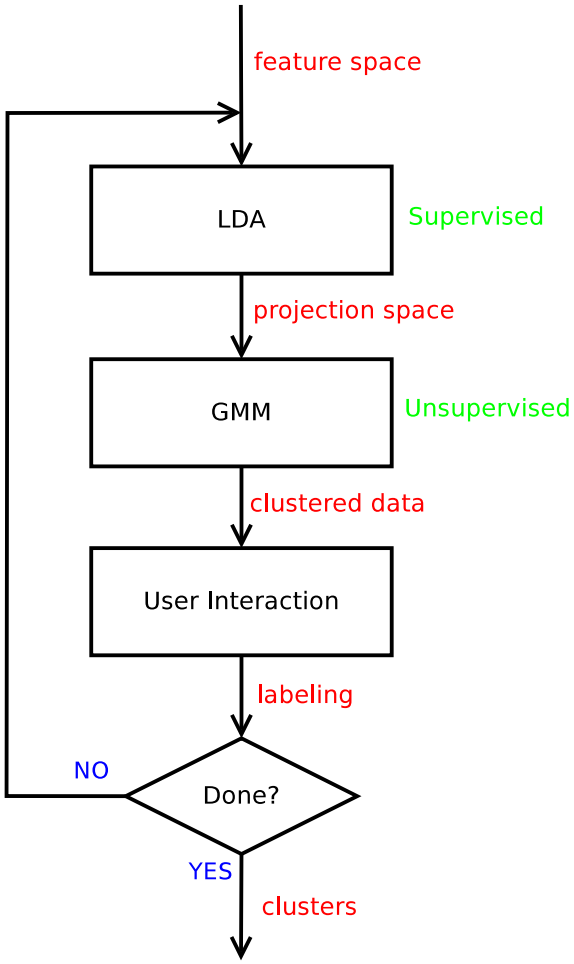


Figure 3.8: Overview of the proposed semi-supervised clustering approach.

Semi-Supervised Clustering

The proposed approach for the clustering of the extracted character candidates tries to minimize user interaction, assuming that if a cluster has a large number of instances of the appropriate character and relatively few other ("miss-classified") characters, there is no need for further refinement. This assumption is based on the fact that the aforementioned conditions are sufficient for extracting a pattern for the character.

The proposed semi-supervised clustering consists of three main components: Linear Discriminant Analysis (LDA), Gaussian Mixture Models (GMM) and User Interaction, as it is summarized in Figure 3.8.

Linear Discriminant Analysis Linear Discriminant Analysis (LDA) [150] is a supervised dimensionality reduction method, which projects the initial data to a subspace via a linear projection matrix. The projection is performed by optimizing the Fisher's criterion, maximizing the inter-classes variance while minimizing the intra-class variance. This is interpreted as trying to reduce the dimensionality while enhancing the separability of the classes. The aforementioned criterion can be formulated as a generalized eigenvalues problem and, therefore, can be solved analytically by using Singular Value Decomposition. The resulting subspace has dimensionality equal to the number of classes minus one.

The use of LDA imposes the user information about the character clustering upon the projected data, incorporating a supervised step to the proposed methodology. The whole concept is in a manner similar to must-link and cannot-link constraints [155], but in a more compact way. Furthermore, the projected data consists of significantly fewer dimensions simplifying the upcoming unsupervised clustering for two reasons. First, the complexity is reduced as it is proportional to the the dimensionality of the features and, second, more meaningful clustering results are generated, avoiding the so-called dimensionality curse.

Gaussian Mixture Model

After taking into account the information provided by the user (by applying the supervised LDA technique as it was mentioned previously), a re-assignment of the clusters is attempted. This is performed via an unsupervised clustering technique, such as the k-means clustering. The number of the clusters, which is a parameter for the clustering technique, is defined by the choices of the user (e.g. if the user chooses to create a new cluster or to delete one) and it is the same as the number of unique labels for the LDA step.

Instead of using k-means for the unsupervised clustering task, the more generic Gaussian Mixture Model (GMM) [150] method is preferred. In fact k-means is a deterministic special case of GMM. Specifically, each cluster in GMM is a gaussian distribution with an assigned probability. GMMs can simulate a wider variety of cluster shapes due to the covariance matrix (hyper-ellipsoid), which enables a more realistic interpretation of the data distribution, rather than using fixed-sized hyper-spheres for each cluster as k-means does.

Similar to k-means, the GMMs are generated by fitting them to the data via an iterative process which maximizes the overall likelihood of the model. The parameters of the model are estimated using the Expectation Maximization (EM) algorithm. EM finds a local maximum of the likelihoodl, which may result in varying resulting parameters for different initializations, as in k-means.

User Interaction Despite of how effective a clustering technique is, it will not provide satisfying results and in the absence of labeled data in order to use a supervised technique, the user should provide the necessary information about the quality of the clustering or a new/different labeling. Our purpose is to make the system provide highly representative information about existing clusters and also incorporate effectively all the information the user can give in an iterative manner as in Figure 3.8.

The representation of each cluster is done by performing a sub-clustering. The number of sub-clusters is fixed to 4, as it is shown to lead to a fair representation of the inner structure of each class. As in general clustering, GMMs are also used for sub-clustering for the aforementioned reasons. However, GMMs are used directly to the initial feature space, rather than the projected, because the projected is biased by the previous labeling.

Due to the dimension of the initial data and the reduced number of data in each cluster the generated covariance matrices, most likely, are not full-rank and therefore the diagonal approximation of the covariance matrix is preferred.

The interaction of the user is constrained to a set of specific actions, provided by a Graphical User Interface, which are described in detail in the following section.

Graphical User Interface

The user interacts with the clustering procedure though a Graphical User Interface (GUI), which provides all the necessary actions. An example of using the GUI is depicted in Figure 3.9, where all the important actions and the provided information are annotated. As it is observed in Figure 3.9, the layout of the GUI (for a selected cluster) is simple without providing overflowing and rather unnecessary details. The provided user-friendly actions and representation of the cluster, which will be described in detail below, encourage the user in contrast to the rather bothersome procedure of "carpet"-like approaches [25].

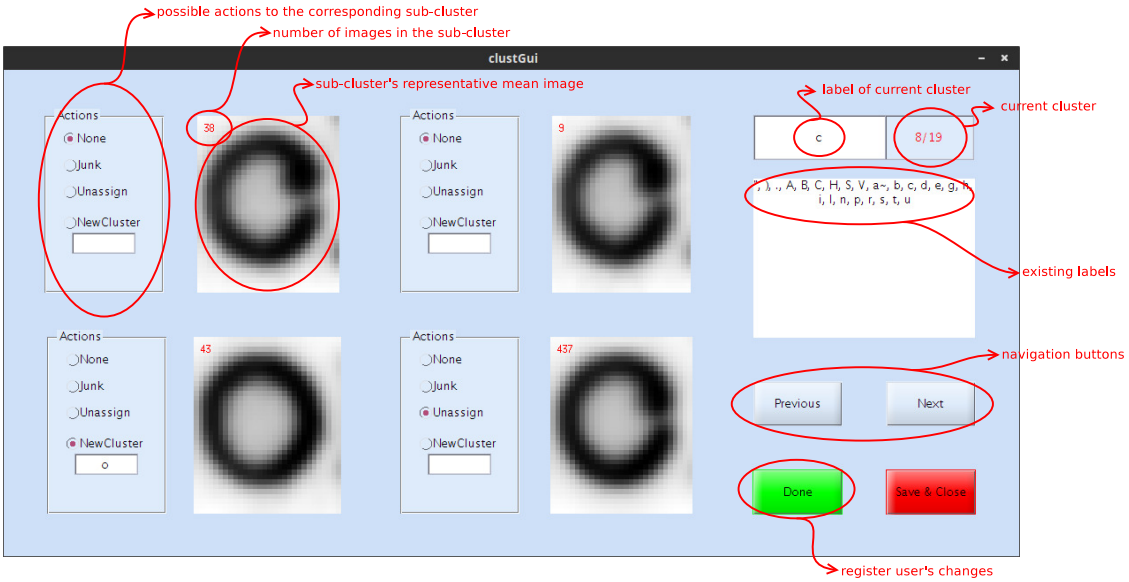


Figure 3.9: Graphical User Interface.

Sub-Clustering As mentioned before, a rough estimation of the inner structure of the class is given by sub-clustering each generated cluster. The GUI provides a mean image as a representative for every sub-cluster, as is shown in Figure 3.9. The mean images should be incomprehensible if many instances of different characters belong to the same sub-cluster, but will be rather clear if the majority of the instances comes from the same character. For clarity, the number of the associated character images is displayed for each sub-cluster in the top-left corner of the representative image, as it is marked in Figure 3.9. The purpose of providing the number of images in each sub-cluster is to measure the "strength" of the sub-cluster, i.e. how much does it contribute to the whole cluster.

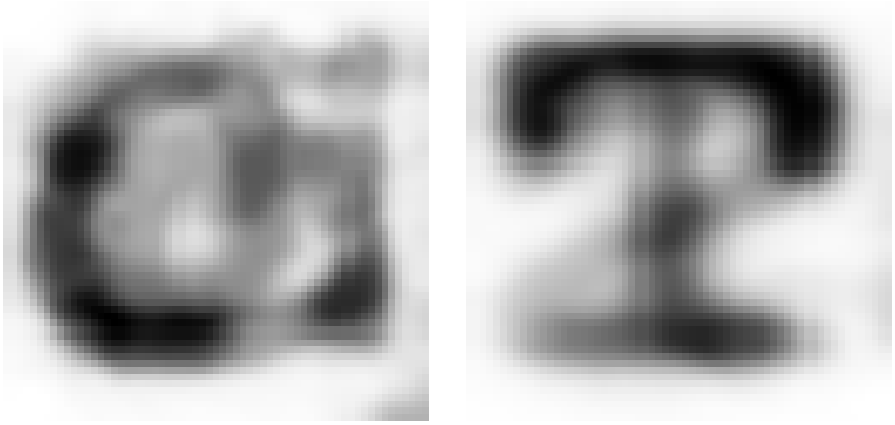
Actions The user can navigate through the existing classes and for each sub-cluster, four actions are possible:

- **None** No action. The sub-cluster is in correspondence with the current cluster and therefore no change is needed.

- **Junk** Delete every descriptor in this sub-cluster (mostly for extremely noisy set of images or over-segmented components of characters). This action is used for discarding a set of images that does not correspond to a valid character and may affect the clustering. A representative example is presented in Figure 3.10(a).

- **Unassign** Unassign the descriptors and expect a more satisfying clustering in a next iteration. It is possible that a sub-cluster consists of instances of more than on character and thus a reassignment of its contents is needed. This can be accomplished in the next iteration of the proposed system, after registering the user's actions. However, we do not want this "miss-labeling" to contribute to the supervised LDA, but only in the upcoming clustering. In other words, we want to keep this sub-cluster "hidden" in the LDA step. This is exactly what the "unassign" action does. An example of a sub-cluster consisting of the characters 'T' and '2' and should be unassigned is shown in Figure 3.10(b).

- **New Cluster** Consider the sub-cluster as a new cluster, thus assigning it a label. If the label already exists a merge is performed between the current sub-cluster and the appropriate cluster. This action is performed when the sub-cluster is clearly a different character from the corresponding cluster's character. Such cases are common when the number of labeled clusters is smaller than the different existing classes/characters, so the user can help the system "discover" new character classes or correct a bad clustering result.



(a) (b)

Figure 3.10: Example of the representative mean image for the case (a) *junk* action (b) *unassign* action

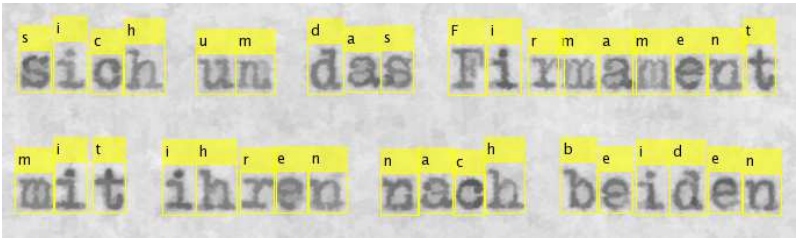
Summarizing, given the representative mean images of the sub-cluster, the user should decide which action is the most appropriate one for a better clustering and eventually unvarying sub-clusters. Such a scenario is depicted in Figure 3.9, where in the cluster 'c' are also instances of the character 'o' ('o' hasn't been "discovered" yet). So we choose the new-cluster action for the bottom-left sub-cluster, while assigning the label 'o', and the unassign action for the bottom-right sub-cluster, which seems to consist of both characters.

It should be noted that the user should identify each constructed cluster and mark it with the corresponding label. This labeling is essential for the upcoming recognition stage.

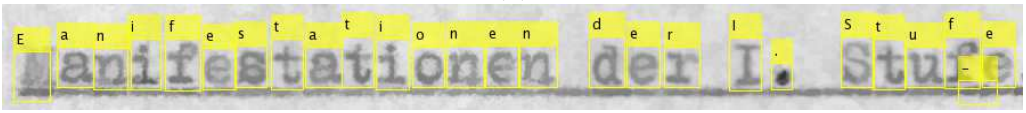
Cluster Initialization For the initialization of the clustering procedure the GUI provides two options: 1) a k-means clustering on the initial feature descriptors to a predefined number of clusters and 2) the usage of a clustering result from a previous use of the GUI. The second option apparently speed up the procedure and can be used to quickly adapt (with the least effort by the user) to slightly different font for a different set of images.

3.2.3 Optical Character Recognition

The final stage of the proposed system is the character recognition according to the generated character classes. This procedure is in fact independent of the previous steps, i.e. given the generated classes from a set of images the recognition scheme can be applied to a different set. The only prerequisite is that the training and the testing sets have the same font (a possible difference in scale can be easily computed using the median connected component size, as in the preprocessing stage). Additionally, similar to the preprocessing stage, an ASF filtering is applied to the images before recognition.



(a)



(b)

Figure 3.11: Examples of Optical Character Recognition.

Template Matching

Recognition is performed using template matching (a template corresponds to a generated class), thus providing a segmentation-free OCR methodology. The choice of template matching is related to our approach of constructing an effective system consisting of simple components. However, contrary to the rest of the system components, template matching is rather slow due to the convolution of each template with the document image.

Template Selection First, a template for each cluster is generated. We observed that for a sufficiently good clustering, which is proportional to the time devoted by the user during the semi-supervised clustering, a mean image of the raw character images is a valid choice. The mean image, given that it is generated from a sufficiently large set of images, is representative of the class, while noise is significantly reduced.

Normalized Cross-Correlation Having extracted the representative templates, the template matching is performed using normalized cross - correlation which is computed for every pixel via Eq.3.10 (I is the image, T is the template, $W_{u,v}$ is the set of pixels corresponding to a window equal to the template's size and centered at the pixel {u,v}, μ_I, σ_I

the local mean and standard deviation of the image corresponding to $W_{u,v}$ and μ_T, σ_T the mean and standard deviation of the template image). Normalized cross-correlation, denoted by c, takes values in the range $[-1, 1]$, where 1 denotes an exact match and -1 denotes complete dissimilarity. It is a widely used technique for template matching due to its contrast insensitive property; a useful property for detecting faint characters. This insensitivity is achieved by dividing the convolution with the local standard deviation of the image and the template's standard deviation. The normalized cross-correlation is implemented according to [95], using integral images for the computation of the local mean and deviation of the document image.

$$c(u,v) = \frac{1}{|W|} \frac{\sum_{x,y \in W_{u,v}} [I(x,y) - \mu_I][T(x-u, y-v) - \mu_T]}{\sigma_I \sigma_T}, \tag{3.10}$$

However, the aforementioned insensitivity is a hindrance in some cases, resulting to false detections, e.g. noise with high variance. Therefore, a semi-normalized approach is adopted. Specifically, we assign a threshold in standard deviation σ_{th}, such that the local deviation is constrained as given by Eq. 3.11.

$$\sigma'_I(x,y) = \begin{cases} \sigma_I(x,y), & if\ \sigma_I(x,y) > \sigma_{th} \\ \sigma_{th}, & if\ \sigma_I(x,y) \leq \sigma_{th} \end{cases} \tag{3.11}$$

We chose this threshold to be twice the median[1]of the local standard deviation of every background pixel, which is denoted by σ_b, i.e. $\sigma_{th} = 2\sigma_b$. We can find the median standard deviation of the background using Sauvola's binarization method with almost no extra computational cost, as the local mean and standard deviation have already been computed.

Character Recognition After performing the template matching, we obtain the normalized cross-correlation coefficients for every pixel and we want to find the possible positions/regions of the recognized character. In order to obtain a reasonable number of possible candidates for each template we discard locations where $c(x,y) < c_{th}$. Furthermore, we retain only the positions corresponding to local maxima by performing a dilation on c with a structural element of the size of the template. The threshold c_{th}, after experimentation, is set to 0.75 which assumes a fair similarity between the template and the corresponding region of the image.

Concerning the final recognition of the image characters, the issue of overlapping regions between different templates should be addressed. Generally, the best spatial arrangement for a set of points can be found by dynamic programming as a generalization of the best sequence problem (e.g. Viterbi algorithm). However, for our simplified case, where usually very small groups of character are overlapping significantly, this can be resolved by finding such each group and choose only a candidate template as a "winner". The winner is the character with the biggest similarity value c. The results of the character recognition can be shown in Figure 3.11. It should be noted that the presented segmentation-free method works well even for underlined text as it can be observed in Figure 3.11(b).

Page Recognition

The last step of the recognition stage, in the context of a complete OCR system, is the retrieval of the text from the document image. Given the labels of the recognized charac-

[1]the choice of median instead of mean is made in order to obtain a less sensitive value due to outliers.

ters and their positions, our final task is to align them into a readable text. This task is performed in a similar way to simple line detection techniques (e.g. using Hough transform) under the assumption that lines can be easily separated, which is a fair assumption for typewritten documents.

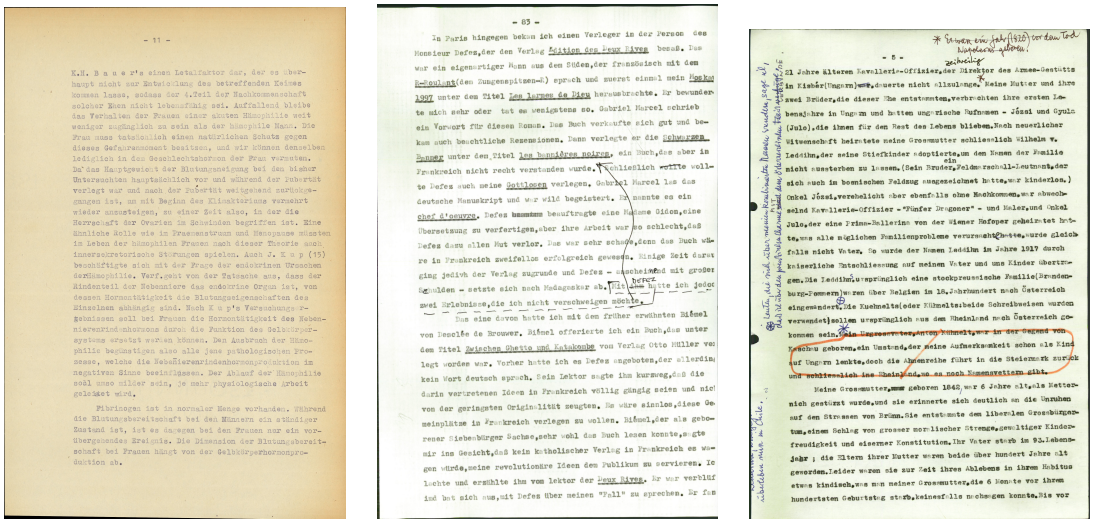


Figure 3.12: Examples of Dataset Images.

3.2.4 Experimental Evaluation

Having described the proposed system, we want to evaluate it in terms of recognition on a selected dataset. Recognition results serve as an indirect evaluation of the quality of the generated classes by the proposed clustering approach.

The dataset contains typewritten pages of technical/ scientific nature from the collection of the University of Innsbruck (in German). Some pages are originals (directly impacted by the typewriter keys and ink ribbon) and some are carbon copies (produced by the force of the typewriter keys through a carbon sheet behind the originals). There are occurrences of manual corrections and annotations in various places. All pages were scanned at 300dpi, the majority in color - the rest in grayscale. Some examples of documents featuring the variety of the dataset are presented in Figure 3.12. The database consists of 8 different sets of document pages, which may vary in font or level of noise, i.e. original or carbon copy.

The number of pages that are used for feature extraction and clustering are 10 and correspond to the training set. We have observed that in practice the choice of 10 pages is sufficient and produces ~ 15000 character images. An increase in the number of training image doesn't have any significant effect in the accuracy of the proposed system and slows noticeably the procedure, making the use of the GUI time-consuming due to unnecessary waiting between each iteration.

The evaluation of the system is performed by measuring the error rate of the recognized text, defined as the edit distance between the generated text and the groundtruth divided by the number of the characters in the groundtruth text. The spaces are discarded from every text in order to have a meaningful error rate since the proposed system does not count them (there isn't a template corresponding to space). Our proposed methodology is compared to the ABBYY FineReader11.0 [3], which is set to typewritten German in order to be consistent with the used database.

Initially we choose only one set and perform the proposed clustering scheme with the aforementioned GUI on 10 randomly chosen pages from this set. The initialization of the GUI clusters is performed using k-means clustering, where $k = 10$. We evaluate our OCR for the following cases: 1) the remaining set (*SetA*) 2) the rest of the sets (*SetB*). This is done in order to evaluate the dependency of the OCR to the font and the variability of the different sets. The resulting error rates are shown in Table 3.4. It is observed that the recognition error of the proposed system is significantly lower for the set used for training, mainly due to the same font, outperforming FineReader. However, in the more generic case of SetB, the Finereader preserves a similar error rate in contrast to our methodology which deteriorates noticeably. The main reason for that is the variability of the fonts in the sets. Considering user's interaction the whole procedure takes about 15 minutes with a random k-means initialization, which assumes no prior knowledge. This can be considered relatively fast, considering that the user assisted the system to "discover" as much existing characters as possible without any initial information.

	FineReader11	Proposed System
SetA	12.20%	6.00%
SetB	10.92%	16.38%

Table 3.4: Error Rate for Page Recognition

	FineReader	Proposed System	Retrained Proposed System	Unidentified Characters (#Clusters)
Set1	12.20%	6.00%	6.00%	0.65%(63)
Set2	15.77%	23.70%	18.90%	13.43%(62)
Set3	15.75%	16.18%	8.51%	1.53%(65)
Set4	9.62%	16.15%	11.27%	1.16%(67)
Set5	7.62%	18.51%	12.04%	1.31%(66)
Set6	7.84%	15.06%	10.43%	1.31%(69)
Set7	7.58%	15.73%	9.56%	1.28%(69)
Set8	4.41%	9.85%	6.17%	1.63%(64)
Total	10.92%	16.65%	11.62%	

Table 3.5: Error Rate for Page Recognition after Retraining each Set

To overcome the aforementioned problem, we try to adapt the clustering results to each set using the GUI initialized to a previous clustering in order to speed-up the procedure. Therefore we introduce a second scenario where having as initial clustering the one from the first experiment, we re-train our system to the current set (by choosing 10 training images). Using the aforementioned initialization, the user's interaction time is significantly reduced to about 5 minutes, since most of the clusters are found correctly instantly and only minor changes are required. The recognition results are presented in Table 3.5 along with the error rate without re-training (corresponding to the first experiment) and the percentage of unidentified characters of the groundtruth. As unidentified characters we refer to those which do not correspond to any of the generated classes (the total number of classes that our system produced in each set is also provided in parenthesis in Table 3.5). The overall recognition rate of our system is greatly improved, as expected. Individually

for each set, the proposed system outperforms FineReader only for two cases.

The main reason for the increased error rate in specific sets is the state of the documents, where many faint characters are present and template matching is not very suitable for these cases. Additionally, a reason which leads to a further deterioration of the recognition result is that the recognized text consists of as many character classes as the generated clusters, while the groundtruth text may have more classes (either they are not present in the training images or the system was unable to "discover" them mainly due to their few occurrences). This can be observed at the last column of Table 3.5. However, it should be noted that using a simple template-matching technique provides promising results after a brief period of user's assistance, even though no pre-trained models are used (e.g. FineReader).

In conclusion, the proposed system can effectively used for document recognition, when no information is provided about the documents, with a minimal involvement by the user. Given the simplicity of many of the system's substeps, the experimental results are considerably accurate. One important point, which is difficult to fully define, is the user's assistance strategy, since the used mean images are helpful but not ideal.

Chapter 4

Learning-Free KWS

In this chapter, we focus on feature extraction methods and the subsequent matching for the task of keyword spotting, given segmented word images. Specifically, we aim to tackle the QbE scenario without using any machine learning technique (learning-free). In the following, we will describe two holistic feature extraction approaches which generate a fixed-sized feature vector for each word image. Having established the effectiveness of holistic approaches, we will finally tackle the problem of KWS by trying to simultaneously utilize the advantages of such approaches along with the advantages of sequential techniques, inspired by Dynamic Time Warping (DTW). This final approach, which combines the merits of both categories, outperforms every existing learning-free approach while sustaining competitive time and storage requirements.

4.1 Word Image Representation using Horizontal Zoning

Words usually contain several characters; If we assume that each character can be drawn into a fixed-size square, the resulting word image would be contained into a rectangle, where width is significantly larger than height at the majority of the cases. To this end, if one has a well-performing feature extraction method, it would be preferable to break the image into smaller segments, approximating ideally the characters' squares.

The strategy of segmenting the image into zones, is called zoning. Concerning the case of word images, the image is divided along the horizontal axis into a fixed number of sub-images, i.e. horizontal zoning, trying to reproduce a draft character segmentation in order to apply the descriptor at hand to each sub-image. Obviously, this approach does not contains a character-segmentation procedure, but rather assumes that the size of the sub-images is similar to the size of a character for a large number of word instances. Neighboring segments can be overlapping. Considering holistic representations, i.e. vectors of fixed size, the number of zones is pre-defined and is constant for every word image.

In the following, we consider two different feature extraction processes which will be combined with a zoning technique:

- *Handcrafted Features:* We apply the POG descriptor of the previous chapter to the keyword spotting problem, since it presented superior performance to the character classification problem.

- *Deep Features:* Contrary to the aforementioned devised feature extraction method, we also consider to use the features generated by a Convolutional Neural Network (CNN), referred to as deep features. It should be noted that we treat the network

as an off-the-self feature extractor rather than a supervised approach and therefore it is presented in this chapter and compared to the learning-free POG method. The selected network, based on a rather simple architecture, is trained on typewritten characters and cannot be considered as an example of the representation power of machine learning, since typewritten characters are not ideal for training a system which aims to handwritten documents (in fact it can be considered as a case of overfitting, since the network learns that each character has only a very narrow set of characteristics, apart from noise). No further adaptation was performed. However, such an approach would serve as a useful baseline for the upcoming deep learning chapter.

4.1.1 Projections of Oriented Gradients

First, we attempt to build a KWS system based on Projection of Oriented Gradients (POG), which has been successfully used in character recognition and displayed robustness to handwritten character variations,as we have shown in the previous chapter. POG is a projection-based descriptor that tries to encode crucial information about the edges in a global manner, which can be briefly summarized by Figure 4.1.

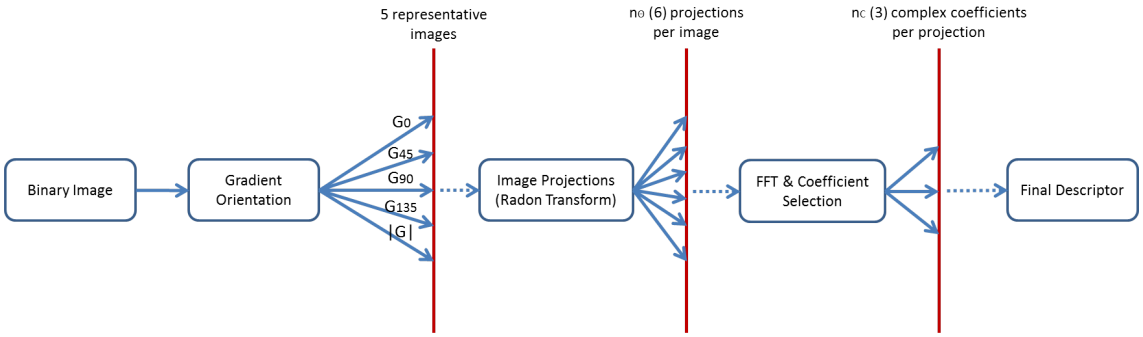


Figure 4.1: Overview of the Projections of Oriented Gradients. First, 5 representative images are generated from each binary image corresponding to the gradient orientation. Then, for each image, angular projections are extracted in a Radon-like procedure. Finally, each projection is encoded with a set of low-frequency complex coefficients of its FFT. The final descriptor is the concatenation of all the extracted coefficients.

Due to POG sensitivity to large affine deformations, we transform the word image before the feature extraction step using a preprocessing step, aiming to alleviate intense deformations of specific nature. The preprocessing step consists of image binarization, skew correction and height normalization. A normalization of the word image to a fixed size is not necessary, because the POG descriptor is size invariant.

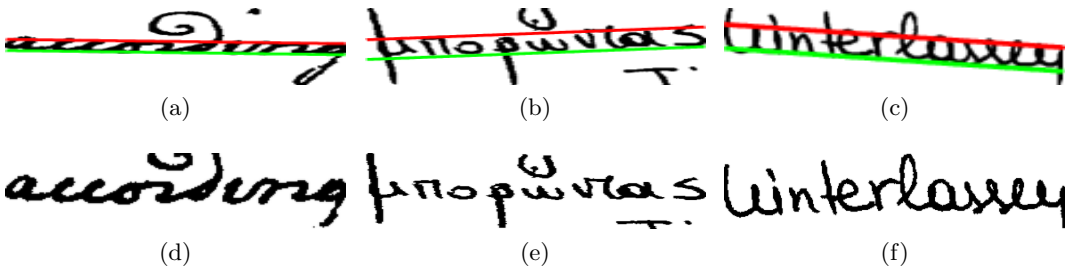
Due to the fact that the POG method is currently restricted to binary images, a *binarization* is imperative for non-binary word images. For the image binarization procedure, we choose Sauvola's method [156].

Furthermore, we employ a *robust regression* method in order to find the main zone of the word. The algorithm that we use for this task is a regression procedure based on iteratively re-weighted least squares, using a bi-square weighting function [84]. Essentially, the algorithm finds a set of inliers foreground pixels (and the corresponding outliers) that best describe the word as a line. The result of this procedure consists of a line model, i.e. the parameters a, b of the fitted line $y = ax + b$, as well as the corresponding width of the line, as a variance parameter. Therefore, the word's main zone is the area within the

resulting thickened line, which excludes outliers that, ideally, correspond to the ascenders and descenders of the word. Given the line model and its main zone (denoted by a baseline and an upperline as shown in Fig. 4.2(a),(b),(c)), we perform the following normalization steps:

- **Skew Correction:** Given the parameters of the fitted line, it is trivial to compute the rotation angle of the main zone and subsequently deskew the image.
- **Height Normalization:** We place the main zone in the center of the generated normalized image (y-axis), which will promote the global approach of the proposed feature extraction scheme. Additionally, in order to avoid extreme ascenders and descenders, that contain no useful information, we crop the image using a threshold of the vertical distance from the main zone. The threshold is dependent to the width of the main zone w_l, i.e. the normalized image has a margin of $1.4 \times w_l$ pixels under and over the main zone (see Fig. 4.2(d),(e),(f)).

Overall, the preprocessing step consists of simple and cost-effective stages in order to obtain normalized word images with respect to the spatial distribution of the word pixels into the image (centralized and deskewed words).



(a) (b) (c)

(d) (e) (f)

Figure 4.2: Word image normalization: (a),(b),(c) are the initial word images and their main zone marked with a baseline and a upperline and (d),(e),(f) are the normalized images after skew correction and height normalization

We decide to use $k = 4$ overlapping zones (20% overlap) as shown in Figure 4.8. The final descriptor of each word image is the concatenation of the k generated POG feature vectors. It should be noted that we can reconstruct approximately the original (representative) image from the POG descriptor via the inverse Radon transform (see Chapter 3), after interpolating each projection to a specific length using the inverse Fourier transform. Hence, a normalized approximation of the image is generated. Examples of this visualization are depicted at the bottom row of the Figure 4.8.

4.1.2 Zoning-Aggregated Hypercolumns

We use a deep Convolutional Neural Network (CNN) pretrained for a character classification task [93]. Instead of using the CNN for the task it was originally trained for, i.e. character classification, we use it as an off-the-shelf feature extractor. It has been shown that per-layer activations can act as efficient local descriptors [12, 72]. The pool of resulting convolutional features is aggregated to a single descriptor per word image. Aggregation is done by combining simple sum-pooling, that has been recently demonstrated to be an appropriate encoding technique for convolutional features [11]. We combine this simple aggregation model with a zoning scheme, suitable for word images, to create the

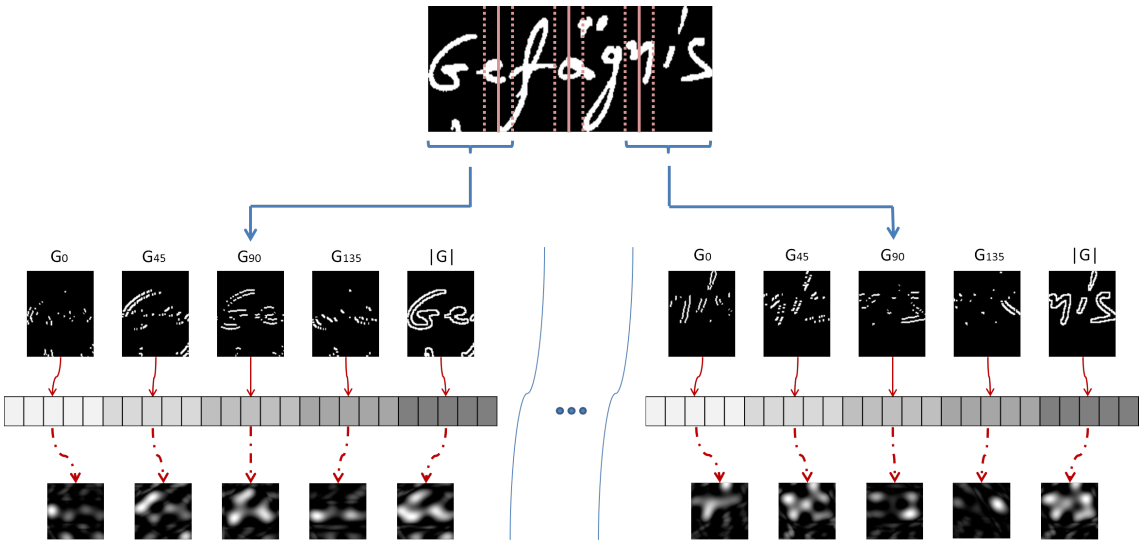


Figure 4.3: Overview of the Projections of Oriented Gradient feature extraction for the case of the k-segmented image approach. Each segmented part is used for extracting a conventional POG descriptor. The last row corresponds to the generated images after the reconstruction and visualizes/highlights the stored information in the proposed descriptor.

fixed-length word-level feature vector. We shall refer to this word-level descriptor as a "Zoning-Aggregated Hypercolumns" descriptor (ZAH). Querying with the proposed descriptor is performed by nearest-neighbour search in the (Euclidean) descriptor space.

As we have already mentioned, this approach is not, strictly speaking, learning-free, since it relies on an CNN. However, the CNN is rather crude and it is trained on a notably different set, as shown at Figure 4.4. Therefore, consider the CNN as an off-the-self feature extractor, which is not adapted to the specific task at hand (KWS on handwritten documents). This assumption is supported by the experimental results of this chapter, where the POG variation (handcrafted features) outperforms the ZAH approach. On the contrary, as we will present at the upcoming chapter, deep learning techniques, carefully trained, have proven to considerably outperform existing learning-free approaches. Summing it up, we consider the ZAH approach as a special case, much closer (and comparable) to learning-free approaches than deep-learning approaches.

Pre-processing We use a similar preprocessing step to the POG approach, consisting of skew correction and height normalization. It should be noted that main-zone detection assisted significantly the performance.

Setup of the pretrained CNN To compute the proposed descriptor, we make use of a pretrained deep Convolutional Neural Network. We have used the character classifier CNNs of [93]. In principle, we can use any off-the-shelf CNN that has been trained to classify character images to character classes. We have used the case-insensitive CNN character classifier and the bigram classifier of [93], to which we shall refer here as unigram CNN and bigram CNN. These nets have been trained on a total of $186k$ and $92k$ samples respectively, coming from various "street-view" sources (see fig. 4.4).

Feed-forward neural networks consist of a stack of layers, each one forwarding output to the layer that follows, until the output layer is reached. Consequently this applies also to convolutional networks, i.e. the nets we use. Input is pre-normalized so that the mean

of the input is zero and its standard deviation equal to one. The architecture of the CNNs we have used in this work is as follows. Input is a box of 24×24 pixels, containing a single unigram or bigram. The output of the models contains 37 classes in the case of the unigram model (26 latin characters, 10 digits and a class for space) and 604 classes in the case of the bigram model. Intermediate layers are either sets of convolutional or fully-connected layers. We have three sets of convolutional layers topped by maxout layers, and one set of fully-connected layers topped by a maxout layer. The convolutional layers have window side of 9, 9, 8 pixels respectively. We shall refer to the maxout layers for each set as $conv1$, $conv2$, $conv3$ and $fc1$ correspondingly. In maxout layers a point-wise maximum over underlying layers of the same resolution is taken. In each maxout layer the maximum value is taken over tuples of 2, 4, 4, 4 underlying channels, respectively for $conv1$, $conv2$, $conv3$, $fc1$. Each maxout layer applies this operation over a number of input channels such that the output channels are 48, 64, 128 total channels respectively for the first three maxouts and 37(604) for the last maxout layer of the unigram(bigram) model. The last layer $fc1$ is followed by softmax activation to convert its values to a probability vector of class responsibilities.

Figure 4.4: Samples of training images that the CNNs we use in this work were trained with.

Extracting Hypercolumns using the CNN In test mode, the input can vary to any size larger than the size at training time. We resize input word images so that their height h is equal to a fixed number close to the original CNN input box size (we used 30 as a default value), and width w is such that the aspect ratio of the image remains fixed. With appropriate zero-padding, the convolutional network gives output as a $30 \times w$ map. The amount of zero-padding necessary depends on the number and size of the receptive fields of the net's convolutional layers. We zero-pad inputs with 11 zero pixels per dimension. The resulting map is of size $h \times w \times d$, where d is the total number of channels for all net layers. For the unigram model, this would be $d = 48 + 64 + 128 + 37 = 277$ (we do not take into account maxout inputs). We shall refer to this per-pixel concatenation of layer activations into a single vector as a "hypercolumn", following [72]. In practice, we do not concatenate all layers; concatenating a subset of all layers tends to be more efficient (cf. [72], and sec. 6.4.3 in the current work).

Zoning The input is partitioned into Z vertical zones (fig. 4.5); we used $Z = 6$ in this work. Each zone overlaps its neighbouring zones by a width equal to a $\frac{1}{4Z}$ fraction of the total word width.

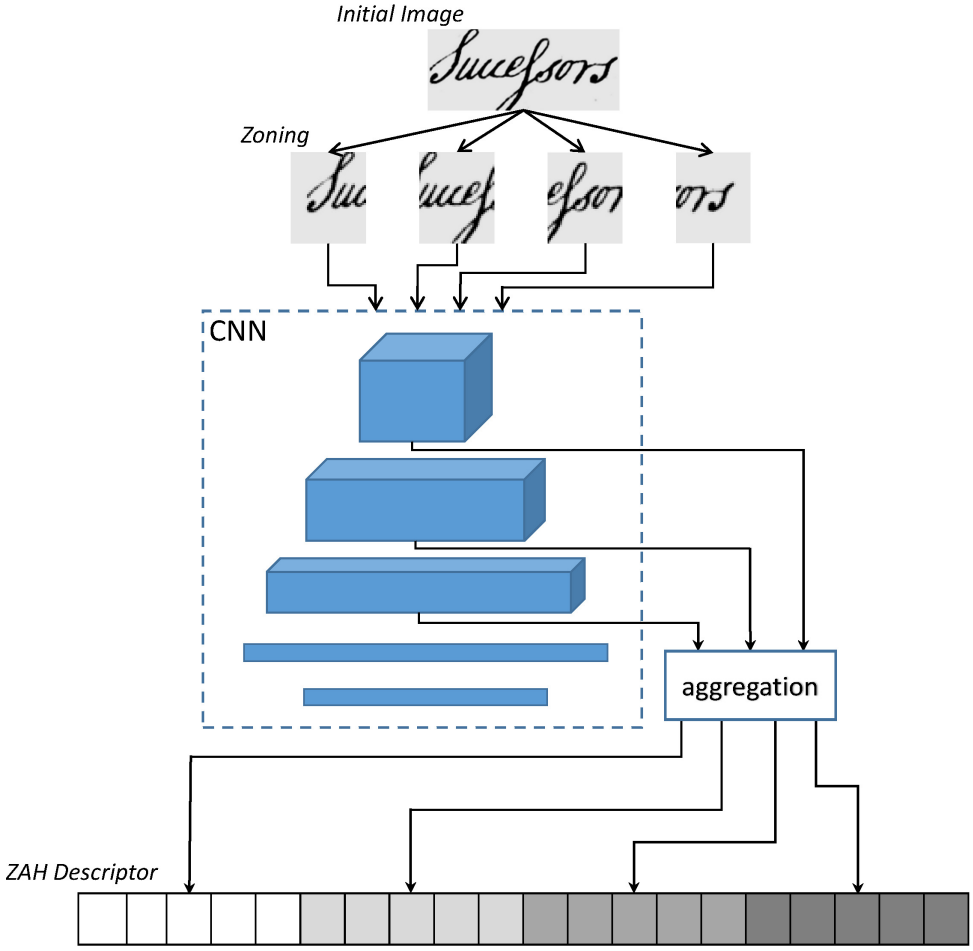


Figure 4.5: Processing pipeline to construct ZAH features. From top to bottom, we have: Input word image ; partition of image along its width into Z zones ($Z = 4$ for this example), with each zone covering an area of $h \times w_z$ pixels; extraction of pixel-level hypercolumns, for each zone a $h \times w_z \times d$, where d is hypercolumn dimensionality; aggregation of hypercolumns into a single $1 \times d$ hypercolumn per zone; concatenation of zone hypercolumns into a single $1 \times Zd$ vector.

Encoding Hypercolumns Preprocessing and passing each word zone through the CNN gives Z $h \times w_z \times d$ maps of hypercolumns, one for each zone. [1]. We proceed by encoding each of the zone hypercolumns to a single descriptor per zone. We used sum-pooling for aggregation, formally:

$$d_z = \sum_{x \in W_z} \sum_{y \in H} v_{xy} \tag{4.1}$$

where d_z is the descriptor of zone d, set W_z contains all possible width values of zone z, set H contains all possible height values, and v_{xy} is the hypercolumn computed for the pixel with xy coordinates. We have also experimented using a center prior with a weighted-sum version of sum-pooling [11], formally

$$d_z = \sum_{x \in W_z} \sum_{y \in H} v_{xy} \exp^{-\frac{1}{2}\lambda(y-y_0)^2} \tag{4.2}$$

[1] We actually feed the whole word image to the CNN and then split into zones, which is more efficient; the two practices are in other respects equivalent.

where λ is the precision (inverse variance) of the Gaussian kernel and y_0 is the height coordinate of the central horizontal axis of the word image window. Setting $\lambda = 0$ is equivalent to standard sum-pooling.

Zone-specific descriptors are then l2-normalized and concatenated to a single, fixed-length vector d:

$$d = [\frac{d_1}{||d_1||}, \frac{d_1}{||d_1||}, \cdots, \frac{d_Z}{||d_Z||}] \tag{4.3}$$

The full vector is itself subsequently l2-normalized.

It should be noted that after experimental evaluation, we conclude on using the conv2 and conv3 of the bigram model. For further details see [158]

4.2 Sequential Matching

Having explored two indicative holistic approaches, we can conclude that even though the feature extraction methods are well-performing, they are too sensitive to specific distortions. In this section, we revisit and address typical distortions found in word images, aiming to build a effective KWS system. Our system is based on carefully crafted preprocessing step, a modified/improved POG variant and a novel matching algorithm over sequences of features.

4.2.1 Contribution

The main goal of the proposed method is to handle the majority of challenges encountered in the QbE scenario, such as affine variations between images and different writing styles. At the same time, these variations should be treated in such a way that resource requirements are retained low. The proposed keyword spotting method consists of three main steps, 1) preprocessing, 2) feature extraction and 3) matching, which are described in detail in the following sections. These steps are strongly related and each one is designed to address a different set of common variations in word images that have been identified as crucial to the system's performance.

Specifically, we propose a novel descriptor, referred as mPOG, which encodes gradient orientation and exhibits robustness to small affine distortions. These variations are common when considering collections containing documents of different writing styles. One drawback of appearance descriptors is their sensitivity to rotation and translation. To this end, the proposed preprocessing step provides rotation and vertical translation invariance by detecting the main-zone of a word image. Furthermore, horizontal variation is addressed using a zoning scheme, resulting to a sequence of descriptors followed by an appropriate sequence matching algorithm. Finally, the sequence matching algorithm is modified in order to deal with multiple instances of the query image aiming to assist the error-prone preprocessing step. The proposed preprocessing step is dependent on the selection of specific parameters that may not perform equally well on different datasets. In order to overcome this shortcoming and avoid any errors at an early step, different instances of the normalized query image are generated for different preprocessing parameter values and the aforementioned problem is resolved using the proposed multiple instance sequence matching algorithm.

The proposed method not only achieves outstanding performance in terms of retrieval accuracy, as it will become evident in the experimental results, but is also cost-effective

with respect to time and memory requirements. One of our major concerns was the minimization of the resource requirements without affecting the method's performance, aiming towards a real-time KWS application. We distinguish two main directions in resource optimization: 1) storage of word images (document collection storage) and 2) retrieval time. The main computational advantages of the proposed method are highlighted below:

(i) Each word image is described by a short sequence of descriptors using a zoning technique (e.g. 4-8 zones), reducing the storage cost of the image representation. The produced descriptors are further compressed with the use of a dimensionality reduction technique (PCA).

(ii) Possible extra variations are considered only on the query image (denser zoning or multi-instance generation), avoiding the overhead of computing and storing extra information for each segmented word image of the document collection. Therefore, word images are represented with shorter sequences of descriptors compared to query images. To support this decision, an efficient sequence matching algorithm between sequences of uneven length is introduced using dynamic programming.

(iii) Retrieval time is further reduced by a re-ranking scheme on the retrieved words. Assuming a query image and a set of word images, words are first ranked with the use of a holistic descriptor and the Euclidean distance. Then, a subset of possibly relevant words is selected as a percentage of the initial words and the sequence-based algorithm is applied only on this subset.

4.2.2 Preprocessing

The preprocessing step consists of two main procedures: contrast and main-zone normalization. It is assumed that the input is a gray-scale word image.

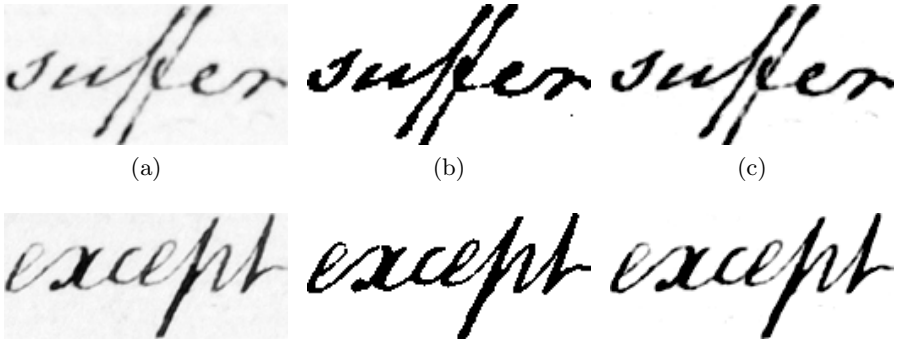
Contrast Normalization

Robustness to illumination changes is a key aspect of several descriptors, e.g. Histogram of Oriented Gradients [37], which is commonly addressed locally by performing an appropriate normalization on a group of neighboring descriptors. However, document images are simpler in this aspect, since the text is supposed to be black (or dark) colored letters on a white page. Based on this observation, many document image processing tasks (e.g. text-line segmentation, text recognition) rely on binarization as a preprocessing step. Nevertheless, binarization is error prone due to the hard assignment of a pixel to either foreground or background.

The aforementioned problems are surpassed by replacing the hard assignment of the binarization with a soft assignment scheme. In more detail, starting from Sauvola's binarization technique [156], a membership function is defined and applied to each pixel (see Equation 4.4) resulting to the contrast-normalized image I_{cn}. Each pixel with illumination value outside the interval $[a(x,y), b(x,y)]$ is considered either foreground or background. We define $a(x,y) = t(x,y) - 1.5s(x,y)$ and $b(x,y) = t(x,y) + 0.3s(x,y)^2$, where $I(x,y)$ is the initial image, $t(x,y)$ is the Sauvola's method threshold and $s(x,y)$ the standard deviation at each pixel. Mean value, standard deviation and the corresponding threshold are computed as in [156], assuming a neighborhood of interest which is specified by a window of fixed size.

$$I_{cn}(x,y) = \begin{cases} 0, & I(x,y) \leq a(x,y) \\ \frac{I(x,y)-a(x,y)}{b(x,y)-a(x,y)}, & a(x,y) < I(x,y) \leq b(x,y) \\ 1, & I(x,y) > b(x,y) \end{cases} \tag{4.4}$$

Examples of the contrast-normalization step are presented in Figure 4.6 in comparison with Sauvola's binarization, given the same parameters. It can be observed that the normalized images retain the useful relative contrast changes on the foreground while at the same time enhance the contrast between foreground and background pixels. Furthermore, despite the fact that binarization introduces boundary discontinuities, it is obvious that the proposed contrast-normalization step enhances the edge contrast without affecting the edge boundaries. This property has proven to be essential for the extraction of informative gradient-based descriptors, which is the next step of the proposed method.



(a) (b) (c)

(d) (e) (f)

Figure 4.6: Contrast-normalization examples: (a),(d) initial grayscale images (b),(e) Sauvola's binarization (c),(f) proposed contrast normalization.

Main-Zone Normalization

The goal of this procedure is to detect the main-zone of the word. Once the main-zone has been detected, the following normalizations can be straightforwardly applied: 1) slope correction (deskew) 2) vertical centralization 3) ascenders and descenders cropping.

Main-zone detection is often a crucial step in text image normalization. The main-zone of a text image (word or line) is defined by an upper and a lower boundary, often referred as upperline and baseline. Under the assumption that these boundaries are linear (or piecewise linear), regression techniques have been successfully applied [65],[144]. Alternatively, the upper and lower boundaries of the main zone can be arbitrarily detected using trained models as in [45] and [129].

In this work, the boundaries are considered linear and the main-zone detection is achieved by processing the horizontal projections which correspond to the distinct angles: $\theta \in [-8°, -7°, ..., 7°, 8°]$. Horizontal projections are generated using the Radon transform instead of rotating the image, which introduces an extra interpolation step, in order to reduce the computational cost. Given a horizontal projection $P_\theta(i)\,, i = 1, .., L$ corresponding to a specific angle θ, our goal is to find a lower and an upper bound (a and b, respectively) such that the sum of the enclosed projection values is maximized ($\sum_{i=a}^{b} P_\theta(i)$), while the height of the main zone ($b-a$) is minimized. This is achieved by maximizing the criterion of Equation 4.5, in which the contribution of the two terms is controlled by a regularization parameter r. The problem of finding the main zone is therefore formulated as a problem of finding the maximum contiguous subarray, which can be solved in linear time.

[2]The choice of a, b adjusts an interval $[a, b]$ with its endpoint b close to the Sauvola's threshold t and its length proportional to each pixel's standard deviation s.

The angle with the maximum value J_θ defines the slope of the main zone.

$$J_\theta = \max_{a,b} \left\{ \frac{\sum_{i=a}^{b} P_\theta(i)}{\sum_{i=1}^{L} P_\theta(i)} - r\frac{b-a}{L} \right\} \tag{4.5}$$

The robustness of the detection is further improved by:

(i) Replacing the length L with L' corresponding to the 95% of horizontal projection's information, strictly formulated as in Equation 4.6. The upper and lower boundary that define L' are referred as u and l, respectively.

$$L' = u - l \quad \text{s.t.} \quad \frac{\sum_{i=l}^{u} P_\theta(i)}{\sum_{i=1}^{L} P_\theta(i)} \approx 0.95 \tag{4.6}$$

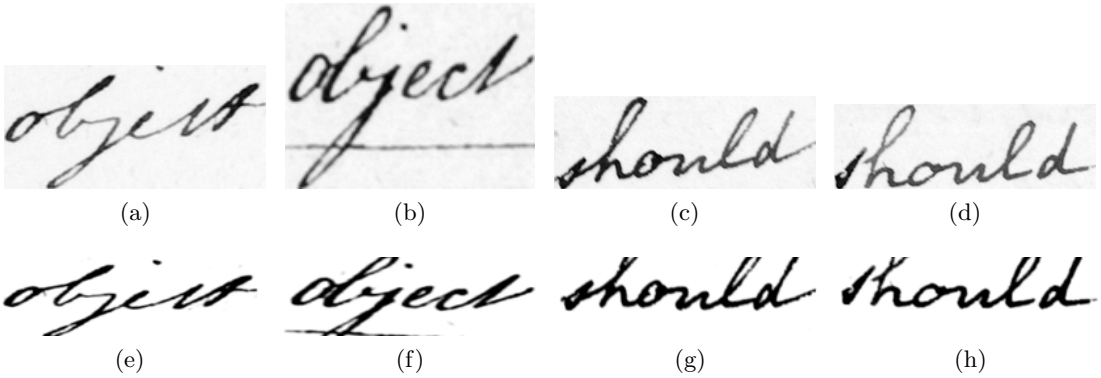
This choice ideally eliminates variations due to the segmentation of the words, which may result in the inclusion of parts of neighboring words.

(ii) Introducing an adaptive regularization parameter r which depends on the distribution of the horizontal projection as described in Equation 4.7. The reasoning behind the definition of r is that words with significantly large ascenders or descenders should obtain a higher value, thus leading to a narrower main zone.

$$r = \frac{\sum_{i=l}^{u} P_\theta^2(i)}{\left(\sum_{i=l}^{u} P_\theta(i)\right)^2} \tag{4.7}$$

After the detection of the main-zone, we proceed with skew correction using the slope θ of the detected main-zone, as well as vertical normalization of the image by moving the main zone at the center of the generated normalized image. Additionally, in order to avoid extreme ascenders and descenders, that contain no useful information, the images are cropped (or padded) to a fixed height which is proportional to the main-zone's height. Specifically, given the height of the main zone $h_m = b - a$, the resulting normalized image has a margin of $1.5 \times h_m$ pixels under and over the main zone, i.e. the resulting image has $4 \times h_m$ overall height. The results of the complete preprocessing step are depicted in Figure 5.2. It is evident that many of the undesired variations have been reduced. Although appearance descriptors (e.g HOG or mPOG) are robust to small (affine) deformations, they cannot cope with the large variations that are frequently encountered in word images (rotations and translations). Thus, it is imperative to use the proposed preprocessing step before the extraction of an appearance descriptor.

Experimentation indicates that the most critical factor affecting the performance of the proposed keyword spotting method concerns main-zone detection. Even though the proposed preprocessing increases the robustness of main-zone detection, the automatically generated regularization parameter r may result to a poor detection and affect the upcoming steps and consequently the overall system's performance. Instead of expecting to extract very robust main-zones, which ideally requires a machine learning algorithm and a training dataset [45],[129], one can assume a set of n_l different regularization parameters and consequently a set of n_l generated query image instances. Starting from the regularization parameter $r_0 = r$ of Equation 4.7, we define a set of n_l regularization parameters as $r_i = p_i r_0$, where p_i values are uniformly distributed in the interval $[0.6, 1.4]$, i.e $p_i = 0.6 + i\frac{0.8}{n_l}$ $(i = 1, \ldots, n_l)$. In other words, we propose an augmentation scheme in order to cope with possible variations that cannot be addressed effectively at the preprocessing step (see Figure 4.10). The retrieval task for the augmentation case (multiple query instances) is performed as it is described in Section 4.2.4.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.7: Different instances of the words "object" and "should", containing vertical size variation, before (a),(b),(c),(d) and after (e),(f),(g),(h).

4.2.3 Feature Extraction

The main objective of the feature extraction step is to efficiently encode useful and discriminative information of a word image. The proposed feature extraction consists of a zoning scheme along with the extraction of a local descriptor for each zone. The result of the aforementioned step corresponds to a sequence of local descriptors.

A modification of Projections of Oriented Gradients (POG), referred as mPOG, was chosen as the descriptor of the zoned image segments. The POG descriptor was first introduced as a descriptor for the character classification task in our previous work [143] (previous chapter) and later applied to keyword spotting with promising results, as described in [144] (previous section). POG descriptor is a projection based encoding and was conceived as an alternative to HOG descriptor. Essentially, they both encode the same initial information, which is the spatial distribution of strokes or, more precisely, gradient orientation. However, HOG extracts gradient orientation information over image segments, while POG encodes the same information on the whole image using projections.

The main difference of the proposed mPOG descriptor compared to our previous works [143],[144], is the extension of the descriptor in order to be applicable to grayscale images. The gradient information is represented using a set of intensity images, referred as *orientation images*, which correspond to different gradient orientations. This representation, depends not only on the gradient magnitude of each pixel but also on the deviation of the pixel's gradient orientation from the representation's central orientation. Subsequently, the proposed descriptor is constructed by encoding the projections of each orientation image. Furthermore for the efficient encoding of the generated projections their complex Fourier coefficients are used as features. The Fourier coefficients are represented by their absolute, real and imaginary values, while a $L2$-norm normalization is performed on each projection's encoding. On the contrary, the POG descriptor consists only of unnormalized real and imaginary values of the Fourier coefficients.

In the following subsections, the proposed mPOG descriptor is described, emphasizing on the modifications with respect to our previous method. Finally, the zoning procedure is presented in detail.

modified Projections of Oriented Gradients (mPOG)

The steps of the mPOG feature extraction are briefly described below:

1) Orientation Images

Gradient orientation, which describes edge orientation, has proven to be a very informative feature (e.g. HOG, SIFT e.t.c.); hence the proposed descriptor follows a similar concept by representing gradient orientation information through orientation images. Orientation images are defined as intensity images that represent the gradient information corresponding to a specific range of gradient orientations, as depicted in Figure 4.8.

The gradient information is represented by the gradient magnitude M as well as the gradient orientation Φ of the image, using polar coordinates, at each pixel (Equations 4.8,4.9). A wrapping is performed so that all orientation values lie on the interval $[0, 180^\circ)$.

$$M(x,y) = \sqrt{(I_x^2(x,y) + I_y^2(x,y))} \tag{4.8}$$

$$\Phi(x,y) = arctan\Big(\frac{I_y(x,y)}{I_x(x,y)}\Big) \tag{4.9}$$

An orientation image I_ϕ, given the central orientation/direction ϕ that characterizes it, is the result of a pixel-based function on the gradient magnitude M and the gradient orientation Φ. The orientation dependency of this function is modeled by a Gaussian weighting function w (see Equation 4.10) in order to represent how close an orientation is compared to ϕ. The ϕ value corresponds to the mean value μ while the acceptable orientation interval corresponds to the standard deviation σ of the Gaussian function. The resulting weights for each pixel are referred as orientation weights.

$$w(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{4.10}$$

Given a number of possible orientations n_ϕ, the orientation range σ_ϕ and the central orientation values ϕ_k are defined as in Equation 4.11. The resulting orientation images I_{ϕ_k} are generated by multiplying the orientation weights with the gradient magnitude of the image according to Equation 4.12.

$$\sigma_\phi = \frac{180^\circ}{n_\phi}, \quad \phi_k = k\sigma_\phi, \quad k \in [0, n_\phi - 1] \tag{4.11}$$

$$I_{\phi_k}(x,y) = w(\Phi(x,y);\phi_k,\sigma_\phi)M(x,y) \tag{4.12}$$

Orientation images correspond to a decomposition of the magnitude M with respect to the orientation ϕ, i.e. $\sum_{i=1}^{n_\phi} I_\phi \approx M$. The weighting function takes into consideration overlapping orientation ranges, aiming to generate orientation images with smoother strokes in case of edge orientation changes. On the contrary, our previous implementation of POG for binary images involved a hard assignment of pixels in orientation ranges which resulted in discontinuities.

2) Projections

Each orientation image is decomposed into several projections under selected angles by applying Radon transform. This projection-based approach is selected in order to represent (orientation) images in a more holistic manner compared to the HOG encoding (spatial cells). Assuming that the number of projections is n_θ, the projections angles $\{\theta_k\}$ are sampled every $180^\circ/n_\theta$:

$$\theta_k = k\frac{180^\circ}{n_\theta}, \quad k \in [0, n_\theta - 1] \tag{4.13}$$

3) FFT & Coefficient Selection

Each projection is simplified and eventually encoded by selecting only the low frequency components of the projection, which correspond to a smooth approximation of the projection and retain information about regions with high pixel concentration. Therefore, after computing the Discrete Fourier Transform coefficients $c_j\,, j \in [0, K-1]$ of the projection, only the first n_c are used to form the projection's descriptor, excluding c_0. Subsequently, each coefficient is normalized by c_0, which is equal to the total number of foreground pixels. The extracted feature vector of a projection is the concatenation of the real, imaginary and absolute values of the complex feature vector $f_j = c_j/c_0$, $j = 1, \ldots, n_c$.

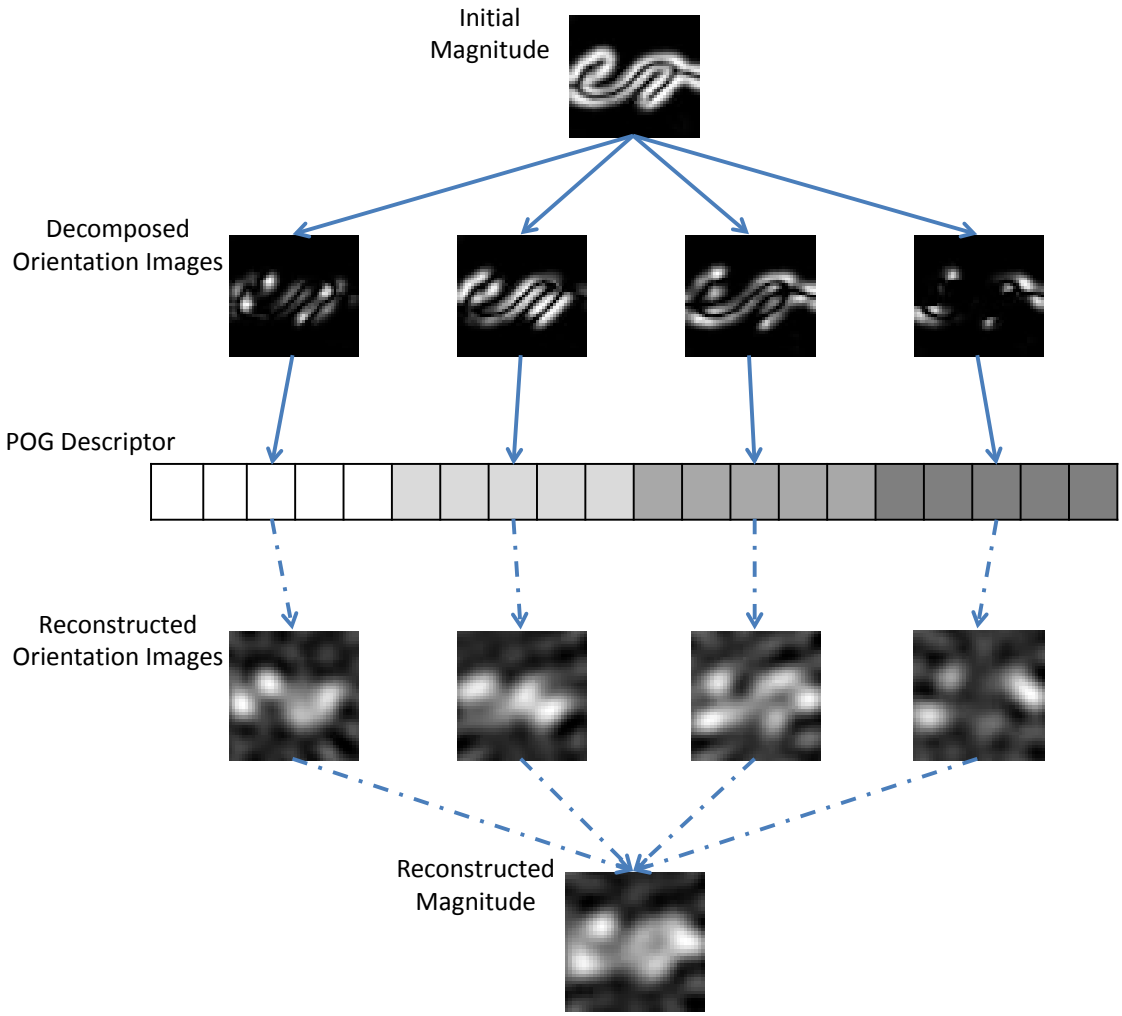


Figure 4.8: Overview of the proposed mPOG descriptor extraction on an image part and their corresponding reconstruction. The reconstructed orientation images visualize/highlight the stored information in each orientation.

4) Final Descriptor

The final descriptor is the concatenation of the generated coefficients of every projection and every orientation image. Overall, the length of the image descriptor is: n_ϕ (images) $\times$ n_θ (projections) $\times$ $3n_c$ (Fourier coefficients). Prior to the concatenation, a $L2$-norm normalization is performed to each projection's feature vector of coefficients.

One interesting property of the proposed descriptor is that each orientation image can be approximately reconstructed via the inverse Radon transform, after interpolating each

projection to a specific length using the inverse Fourier transform. As a result, a normalized approximation of the image is constructed which provides an informative and helpful visualization of the remaining information. The initial decomposed orientation images and the reconstructed orientation images from the extracted descriptor are presented in Figure 4.8. The reconstructed magnitude is visualized as the sum of reconstructed orientation images. We can observe that the descriptor preserves the useful edge information by displaying higher illumination values in the reconstructed images on stroke regions.

Zoning

It is observed (see the experimental results section) that the application of the mPOG descriptor on the entire word image (holistic mPOG) has encouraging performance. However, by keeping a fixed number of coefficients in each projection, we can only retain a uniform distribution of the edge information on each projected direction. Therefore, the usage of a holistic mPOG descriptor results in loss of useful information (e.g. the same amount of information is considered for both vertical and horizontal projections).

Motivated by the successful application of zoning techniques on KWS [144],[158], we propose to uniformly split the image along the x-axis into overlapping segments. In this way, a sequence of descriptors is created. As it was mentioned before, one of the main contributions of this work is the application of a different zoning scheme for query images and word images. We assume that possible translation variations on the horizontal axis (e.g. different spacing between letters or a horizontal displacement of the word) could be captured by denser sampling either the query or the word image and not necessarily both. Therefore, a simple zoning procedure is used on word images, assuming n_w zones, while, on the contrary, a denser zoning is performed on query images, assuming $n_w \times n_d$ zones, where n_d is an integer denoting the number of dense samples for each zone. The width of the zones is the same for both cases. The proposed uneven zoning aims to reduce processing and storage requirements for document collections by shifting the extra computational cost to the query. An example of the proposed uneven zoning procedure is shown in Figure 4.9.

In practice, simple zoning of the word images should be scarce, i.e $n_w \in [4, 8]$, which results in cost-effective matching. This observation is also in line with the fact that the mPOG descriptor performs very well on character level [143] and therefore n_w should take values close to the average number of characters in a word.

4.2.4 Matching of Descriptor Sequences

Single Query Matching

Given the word sequence (reference sequence) t_i, $i = 1, \ldots, n_w$ and the query sequence x_j, $j = 1, \ldots, n_w n_d$, our goal is to assign each t_i descriptor to a single x_j descriptor of the query sequence, as shown in Figure 4.9. The matching procedure results in a subset $S_m = \{x_{j_1}, x_{j_2}, ..., x_{j_{n_w}} : j_{i-1} < j_i\}$ of the query sequence descriptors that best match the reference sequence descriptors. The aforementioned matching approximates an ideal zoning scheme, where each selected query zone is optimally located with respect to the corresponding word zone alleviating possible horizontal translations.

Assuming that no inner (local) horizontal variations exist, the difference of the indexes of neighboring matches should be equal to the denser zoning parameter n_d. This strict assumption allows only global horizontal translation invariance. However, in order to capture local horizontal variations, a constraint on relative distance between neighboring

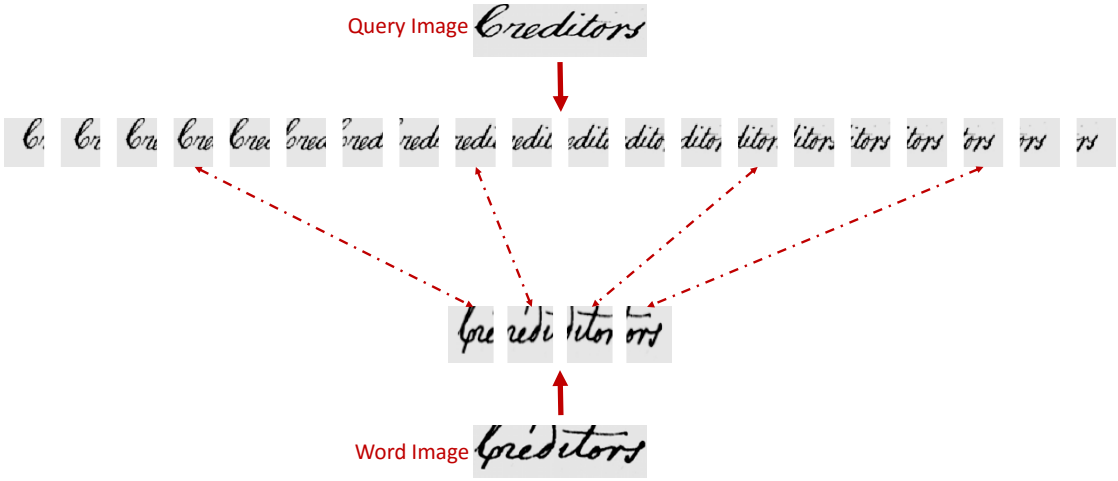


Figure 4.9: Visualization of the matching procedure between the different zoning schemes for a query and a word image ($n_w = 4$ & $n_d = 5$).

matches (at the query sequence) is imposed, i.e. between j_{i-1} and j_i. This constraint penalizes large variations in x-axis, assuming that the "proper" distance of consecutive matches is $j_i - j_{i-1} \approx n_d$. To this end, the quadratic penalty function is defined as:

$$w(k, l; a) = \begin{cases} 1 + p(k - l - a)^2, & \text{if} \quad k > l \\ \infty, & \text{otherwise} \end{cases} \tag{4.14}$$

where a penalty, which depends on a scaling parameter p and has values over 1, is determined by comparing the distance $k - l$ to the pre-specified reference distance a.

Due to the quadratic form of the penalty function the meaningful displacement is restricted to a neighborhood of radius b around the "proper" translation, as it is defined in Equation 4.15.

$$w(k, l; a, b) = \begin{cases} 1 + p(k - l - a)^2, & |k - l - a| < b \\ \infty, & \text{otherwise} \end{cases} \tag{4.15}$$

In practice, we assume $b = a/2$ and the penalty function will be used as $w(k, l; a) = w(k, l; a, a/2)$. The scaling parameter p is selected in such a way that the maximum penalty value is 1.2, i.e. $w(k, l; a, b) = 1.2$ when $|k - l - a| = b = a/2$. Thus, p is defined as: $p = 0.2(1/b^2) = 0.8(1/a^2)$.

The best matching subsequence problem can be formulated as a minimizing function over all possible ordered subsets S_m, while $|S_m| = n_w$, as shown in Equation 4.16.

$$score = \min_{S_m} \{ \|x_{j_1} - t_1\|_2 + \sum_{i=2}^{n_w} w(j_i, j_{i-1}; n_d) \|x_{j_i} - t_i\|_2 \} \tag{4.16}$$

The above formulation indicates that the matching procedure can be efficiently implemented using dynamic programming, avoiding to explore all possible ordered subsets S_m which has an exponential complexity. In more detail, given the distance matrix D, where $D_{ij} = \|x_j - t_i\|_2$ (D is of size $n_w \times n_w n_d$), the overall matching cost is computed by iteratively adding the best match of t_i to the best score so far based on Equation 4.16. A matching matrix M is used to store the solutions (best matchings) of the intermediate

subproblems. Each M_{ij} value denotes the best score considering the subproblem of matching the subsequence $t_1, t_2, ..., t_i$, while t_i is constrained to match with x_j. The matrix M is constructed step by step according to the update function of Equation 4.17, utilizing incrementally the score of smaller subsequences. The last row of M consists of possible matching scores for the whole sequence $\{t_i\}$ and thus the overall best matching score is obtained as the minimum value of this row. We refer to this algorithm, which is described in Algorithm 1, as Selective Matching (SM). It can be easily deduced by the description of the proposed algorithm that its time complexity is $O(n_w^2 n_d^2)$.

$$\begin{aligned} M[i,j] &= \min_{\Omega} \Big\{ M[i-1,k] + w(j,k;n_d) D[i,j] \Big\} \\ \Omega &= \Big\{ k \in [1, n_w n_d] \,,\, |k - j - n_d| < \frac{n_d}{2} \Big\} \end{aligned} \tag{4.17}$$

Algorithm 1 Selective Matching

```
procedure SelectiveMatching(D)
    Input: Distance Matrix, D (n_w × n_w n_d)
    Output: Best Matching Score, score
    for j = 1 to n_w n_d do
        M[1, j] = D[1, j]
    for i = 2 to n_w do
        for j = 1 to n_w n_d do
            Update M[i, j] using Equation 4.17
    return score = min_j{M(n_w, j)}
```

Matching of Multiple Instances

As it was mentioned on the preprocessing section 4.2.2, an augmentation scheme is proposed with respect to the main-zone in order to cope with variations that cannot be efficiently addressed at the preprocessing step. Therefore, we decided to propagate the possible variations generated by the main-zone detection step to the matching step. According to the proposed augmentation scheme, n_l different query instances are created through the use of a set of n_l different regularization parameters at main-zone detection (see Section 4.2.2).

A modified version of the SM algorithm, namely Multi-Instance Selective Matching (MISM), is proposed in order to properly handle the augmented set. The final matching set of descriptors may originate from different query instances in order to increase the robustness of the proposed method. The concept is essentially the same with the simple SM algorithm and therefore has a straightforward implementation using dynamic programming according to the Equation 4.18. Contrary to the SM algorithm, the matching matrix M as well as the distance matrix D have three dimensions $(n_w \times n_l \times n_w n_d)$ since the n_l different image instances form the second dimension of the matrices. The complexity of the new matching procedure is the multiplication of the simple SM algorithm's complexity with the number of different instances n_l, i.e. $O(n_l n_w^2 n_d^2)$. An overview of the MISM is depicted in Figure 4.10.

$$\begin{aligned} M[i,l,j] &= \min_{\Omega} \Big\{ M[i-1,m,k] + w(k,j;n_d) D[i,l,j] \Big\} \\ \Omega &= \Big\{ k \in [1, n_w n_d] \,,\, m \in [1, n_l] \,,\, |k - j - n_d| < \frac{n_d}{2} \Big\} \end{aligned} \tag{4.18}$$

It should be noted that the idea of multi-instance query generation/augmentation is not limited to main-zone detection and could be applied to any possible variation which cannot be addressed in a typical and efficient way at the preprocessing or the feature extraction step, e.g. local affine transformations on each segmented zone.

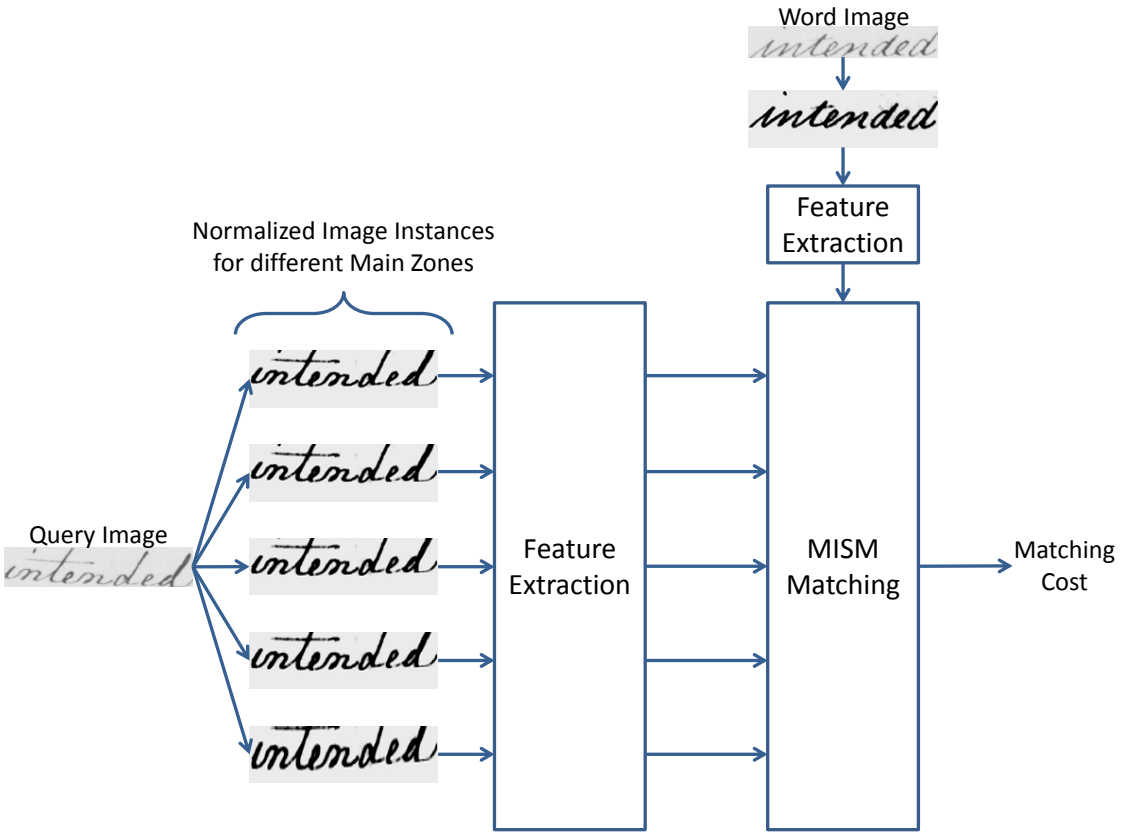


Figure 4.10: Overview of the Multi-Instance Selective Matching for the case of multiple main zone detection.

4.2.5 Experimental Setup

The proposed method applies on the task of segmentation-based learning-free QbE word spotting. Initially, a parameter tuning is performed on the widely used George Washington (GW) Dataset, consisting of three different setups (see Chapter 2) and finally an extensive evaluation of the proposed method is considered on the IAM dataset as well as on several competition datasets (ICFHR 2014 [134], ICDAR 2015 [137] and ICFHR 2016 [135]). A detailed description of the dataset is given at Chapter 2. We should highlight the fact that no further parameter exploration has been performed on the datasets. The values of the parameters have been fixed using the GW1 setup and were kept constant throughout the evaluation stage of the method on all other datasets/setups. Such an experimental scheme not only enables the proposed method to be directly comparable to the methods that participated on the KWS competitions but also confirms the generalization of the proposed method. The performance of the word spotting methods was recorded in terms of the Precision at Top 5 Retrieved words (P@5) as well as the Mean Average Precision (MAP).

4.2.6 Zoning Parameter Selection

We experimented on the impact of the zoning procedure to the system performance by changing the number of word segments n_w and dense query segments n_d using the GW1 setup. The parameters of the mPOG descriptor are fixed to $n_\phi = 4$, $n_\theta = 6$ and $n_c = 7$. For this experiment, only the case of SM is considered. The performance results are presented in Figure 4.11. It can be observed that increasing parameter n_d leads to better performance. This complies with our perception that the denser the sampling, the higher the probability of correctly localizing the corresponding segment. Since, both feature extraction and matching/retrieval time depend on the product $n_w \times n_d$, we concluded that the choice $n_w = 6$ and $n_d = 5$ provides results close to the best performance, while retaining low time requirements. These parameter values are considered as the default values for the remaining experimental section both for SM and MISM (augmentation scheme) approaches.

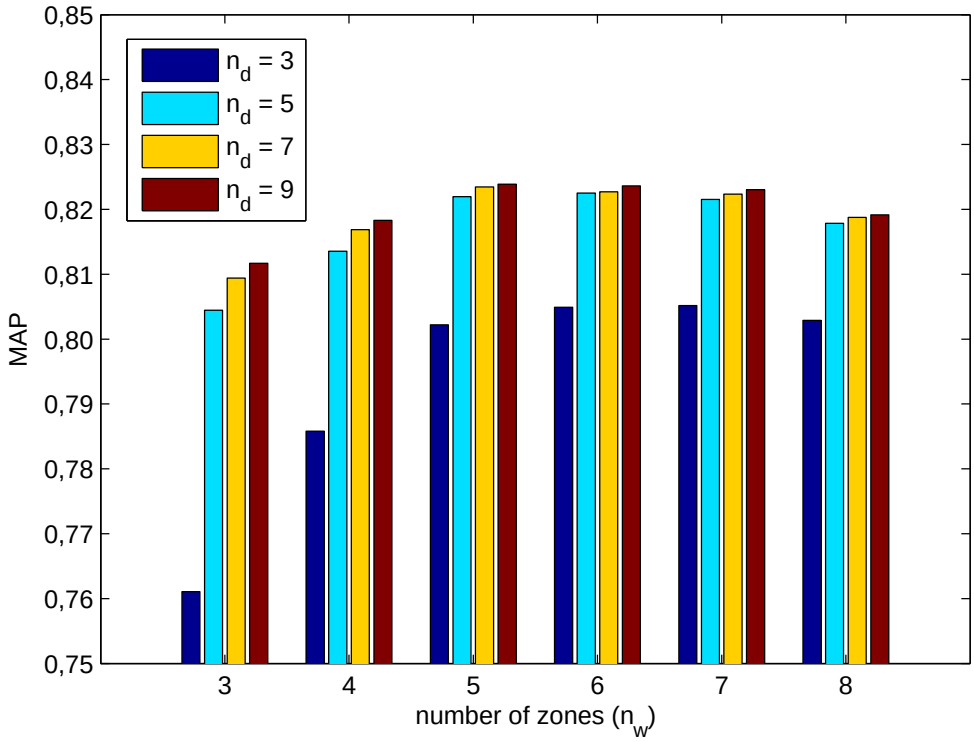


Figure 4.11: Performance evaluation of the mPOG descriptor and Selective Matching on GW1 for different zoning parameters.

4.2.7 Comparing Variations of the Proposed Method

In order to highlight the effectiveness of the proposed mPOG descriptor as well as the proposed zoning and sequence matching procedures, we evaluate the performance of the following variations of the proposed pipeline using the GW1 setup.

Holistic descriptor: The simplest version of the proposed *mPOG* descriptor is to apply it on the whole image as a global descriptor. An alternative descriptor is the widely-used *fHOG* (Felzenwalb's variation of Histograms of Oriented Gradients) [46], which serves as a competitive descriptor to the mPOG. These descriptors along with the Euclidean distance (denoted as *Holistic-mPOG+Eucl* and *Holistic-fHOG+Eucl*, respectively) provide

a baseline performance to be compared with the proposed sequence-based approach.

Sequence of descriptors: Sequence of descriptors, either *fHOG* or *mPOG*, is extracted from each word image using a zoning scheme. In order to further clarify the extraction process of descriptor sequences, we distinguish three zoning schemes. **(a) Seq**: standard zoning on both word and query images (n_w zones) **(b) DSeq**: dense zoning on both word and image queries ($n_w \times n_d$ zones) **(c) PSeq**: standard zoning on word images and dense zoning on query images (proposed). The proposed method (*PSeq-mPOG+MISM*) belongs to the third category and the compared variations based on uneven zoning are denoted as *PSeq-fHOG+SM*, *PSeq-mPOG+SM*, *PSeq-fHOG+MISM*. Due to the sequential nature of the proposed method, the widely used DTW matching algorithm is also evaluated on both uneven-zoned (*PSeq-fHOG+DTW* and *PSeq-mPOG+DTW*) and dense-zoned sequences (*DSeq-fHOG+DTW* and *DSeq-mPOG+DTW*) for comparison. Finally, the *Seq* and *DSeq* schemes are evaluated using only the Euclidean distance, aiming to underline the importance of a sequence matching algorithm on top of a zoning scheme.

The parameters of mPOG descriptor (n_ϕ, n_θ and n_c) are already defined in Section 4.2, resulting to a feature vector of 504 dimensions. Concerning the fHOG descriptors, a 6×6 cell grid is used with 5 possible (unsigned) orientations (720 dimensions). The parameters for both descriptors are selected using a grid search approach. The number of generated query instances, used in the MISM approach, is set to $n_l = 7$. Furthermore, after extracting the descriptor sequences of the images, PCA is performed over local descriptors to reduce the length of each descriptor to 60.

Method	**P@5**	**MAP**
Almazan et al. [7]	88.6	61.8
Retsinas et al. [144]	93.0	70.2
Sfikas et al. [158]	92.4	71.1
Holistic-fHOG + Eucl	90.3	65.8
Seq-fHOG + Eucl	89.5	64.3
DSeq-fHOG + Eucl	90.3	64.8
PSeq-fHOG + DTW	91.7	71.6
DSeq-fHOG + DTW	96.4	81.1
PSeq-fHOG + SM	96.3	81.5
PSeq-fHOG + MISM	96.1	83.2
Holistic-mPOG + Eucl	93.0	71.7
Seq-mPOG + Eucl	94.0	74.9
DSeq-mPOG + Eucl	91.7	69.3
PSeq-mPOG + DTW	89.3	71.6
DSeq-mPOG + DTW	96.5	82.2
PSeq-mPOG + SM	97.3	83.7
PSeq-mPOG + MISM	**98.0**	**87.7**

Table 4.1: Evaluation Metrics on GW1 Dataset

The performance on the GW1 setup of the aforementioned variations along with the state-of-the-art methods of Almazan et al. [7], Retsinas et al. [144] and Sfikas et al. [158], is presented in Table 4.1. The main observations are summarized below:

(i) Sequence-based approaches outperform holistic approaches. The gain in performance between Holistic-mPOG+Eucl and PSeq-mPOG+MISM (over 15% MAP)

highlights the effectiveness of the proposed method. Moreover, this gap in performance is also observed between PSeq-mPOG+MISM and the state-of-the-art holistic methods presented in [144] and [158], which have similar performance to the holistic mPOG descriptor. The necessity of a sequence matching algorithm is verified by comparing the performance of Euclidean matching on descriptor sequences (Seq-mPOG+Eucl, DSeq-mPOG+Eucl) with sequential matching approaches over descriptor sequences (DSeq-mPOG+DTW, PSeq-mPOG+SM and PSeq-mPOG+MISM). The latter category achieves a noticeable increase in performance, as it handles horizontal translations of the image. On the contrary, using the Euclidean distance on standard zoning (Seq), or even dense zoning (DSeq), may result in the accumulation of errors from miss-aligned zones.

(ii) mPOG outperforms fHOG. The mPOG descriptor consistently outperforms fHOG supporting our claim that the proposed descriptor is more robust.

(iii) Comparison with DTW. Even though DTW could be applied to uneven sequences, such an approach would not produce reliable scores when it is applied on the proposed uneven zoning. Specifically, distances concerning translated image segments would be accumulated into the score value. Therefore, due to the descriptor's sensitivity to translation, the calculated score would include many error terms related with the same image segment. Furthermore, when applying DTW either to dense (DSeq) or uneven (Pseq) sequences, an early mismatch in the unconstrained sequence alignment may propagate a significant error into the final score. On the contrary, the proposed matching algorithm (SM) is consistent with the formulation and constraints of the specific problem. The aforementioned analysis is verified by the experimental results.

(iv) Importance of augmentation. The proposed method PSeq-mPOG+MISM which includes the augmentation scheme for multiple main-zone detection, provides a noteworthy boost in performance compared to the PSeq-mPOG-SM version. The success of the PSeq-mPOG+MISM method emphasize the significance of a robust main-zone detection.

4.2.8 Time and Memory Requirements

A simple strategy for the reduction of the retrieval time is the application of a sequence matching approach on a subset of the word images. To this end, based on the successful application of the holistic descriptors, the Holistic-mPOG+Eucl version is considered as the first step of a re-ranking procedure comprising the following steps:

1) Perform a typical retrieval scheme using Euclidean distance on the holistic descriptor.

2) Select a subset of the best retrieved words, i.e. with the smallest Euclidean distance, and consequently re-calculate the matching cost of the reduced set using the SM (or MISM) algorithm on the descriptor sequences.

Table 4.2 presents the impact on the average retrieval time (per query) as well as the retrieval performance with respect to the percentage of selected words for re-ranking. This experiment was conducted only for the case of PSeq-mPOG+SM method. Note that using 0% of words is equivalent to using only the Holistic-mPOG+Eucl approach. It is clear that the retrieval time has linear dependence on the percentage of the selected subset. However, retrieval performance exhibits no significant changes for percentage values greater than 10%. At 10% we achieve a speed-up close to $\times 9$ and approximately 1% drop in MAP compared to applying the PSeq-mPOG+SM method directly on the entire word image collection. Hence, 10% is considered as the default value for the percentage of used words in the upcoming experiments.

Used Words (%)	Time (sec)	P@5(%)	MAP(%)
0	0.0062	93.0	71.7
5	0.0322	96.8	81.5
10	0.0584	96.9	82.2
15	0.0846	96.9	82.6
20	0.1109	97.1	82.9
25	0.1373	97.1	83.1
40	0.2145	97.2	83.5
60	0.3114	97.3	83.6
100	0.5237	97.3	83.7

Table 4.2: Average retrieval time (per query) - Performance trade-off for the proposed re-ranking procedure (using PSeq-mPOG+SM on GW1).

Time and memory requirements with respect to feature extraction (including the pre-processing step), retrieval time as well as storage requirements are presented in Table 4.3. The reported retrieval time per query refers to the re-ranking scheme, i.e. it consists of the retrieval time for comparing the holistic descriptors between the query and the words, as well as the retrieval time for comparing a query sequence with the top 10% relevant word sequences. It is worth mentioning that the feature extraction procedure was performed on parallel using 4 cores. As it has been already mentioned, one of the main contributions of this work is to shift the extra information (dense zoning) at query level in order to retain the memory requirements of storing a document collection low. This is indicated in Table 4.3, where each document ($\approx$ 250 words) requires only 0.5 MB including both PSeq-mPOG and Holistic-mPOG descriptors, whereas a dense sequence approach (DSeq) requires n_d times more storage. The retrieval time, which is the most important factor for a real-time KWS application, is noticeable low compared to the performance gain of the PSeq-mPOG+SM method. As it was expected, the multi-instance modification performs better at the cost of a slower retrieval response (the response difference is proportional to the number of generated instances). Nevertheless, the retrieval time of the proposed method is still sufficient low for a real-time KWS application.

Resource Requirements	PSeq+SM	PSeq+MISM
Extraction Time per Word	0.018 sec	0.018 sec
Extraction Time per Query	0.126 sec	0.659 sec
Memory per Document (after PCA)	525.8 KB	525.8 KB
Retrieval Time per Query	0.058 sec	0.294 sec

Table 4.3: Resource Requirements for sequential mPOG features on GW1

4.2.9 Experimental Evaluation on GW & IAM Datasets

Having concluded on the parameter values (Sections 4.2 & 4.4), we proceed to the evaluation of the proposed method using the remaining GW setups (GW2 and GW3) as well as the IAM dataset. In order to further explore the effectiveness of our method, we also report the performance of both descriptors (fHOG and mPOG) using the variations presented in Section 4.3. The experimental results on the GW2 dataset are presented in

Table 4.4 (a), while in Table 4.4 (b) we report the results on both GW3 and IAM datasets. Learning-based methods are also included in Table 4.4 (b), even though they cannot be directly compared to learning-free methods. We distinguish two evaluation scenarios for learning-based methods: 1) train and test data originate from the same dataset (denoted with *) and 2) train and test data originate from different datasets (denoted with **). The second scenario is used for a better comparison between learning-based and learning-free methods since no fine-tuning occurs. The main observations are presented below:

(a)

Method	**GW2**
Wang et al. [183]	17.5
Retsinas et al. [144]	39.1
Zagoris et al. [189]	40.1
Zagoris et al. [190]	40.5
Almazan et al. [7]	48.3
Sfikas et al. [158]	60.3
Kovalchuk et al. [99]	66.3
Zagoris et al. [191]	69.2
Aldavert et. al [6]	76.5
Holistic-fHOG+Eucl	65.1
DSeq-fHOG+DTW	75.2
PSeq-fHOG+SM	75.7
PSeq-fHOG+MISM	77.6
Holistic-mPOG+Eucl	66.3
DSeq-mPOG+DTW	76.8
PSeq-mPOG+SM	79.2
PSeq-mPOG+MISM	**81.1**

(b)

Method	**GW3**	**IAM**
Retsinas et al. [144]	37.0	15.2
Almazan et al. [7]	49.4	-
Sfikas et al. [158]	58.3	13.2
DTW [9]	60.6	12.3
FV [9]	62.7	15.7
Holistic-fHOG+Eucl	63.8	18.7
DSeq-fHOG+DTW	72.4	23.9
PSeq-fHOG+SM	72.7	23.3
PSeq-fHOG+MISM	74.5	24.7
Holistic-mPOG+Eucl	63.8	18.8
DSeq-mPOG +DTW	72.3	26.9
PSeq-mPOG+SM	74.0	27.6
PSeq-mPOG+MISM	77.1	28.1
Almazan et al.* [9]	93.0	55.7
Softmax CNN* [166]	78.2	48.7
Finetuned CNN* [159]	-	46.5
Sudholt et al.* [166]	**96.7**	72.5
Krisnan et al.* [102]	94.8	80.6
Krisnan et al.* [100]	94.4	**84.2**
Sudholt et al.** [166]	74.9	3.4

Table 4.4: MAP Evaluation on (a) GW2 and (b) GW3 & IAM datasets. (*) : train and test on the same dataset, (**) : train and test on different datasets.

(i) The proposed method outperforms all learning-free approaches. The proposed method (PSeq-mPOG+MISM) significantly outperforms any other learning-free method (including the examined variations) on all datasets/setups. Notably, the proposed method surpasses recent works as [191] and [6], reported at GW2 setup, by a significant extent (11.9% and 4.6%, respectively). In addition, the retrieval results on both tables support our previous claim that the proposed mPOG provides superior performance over the fHOG descriptor. Since both descriptors cannot cope with horizontal translations, they have similar performance as holistic descriptors. However, the robustness of the mPOG descriptor is evident when used as a local descriptor for sequence-based approaches.

(ii) Success of learning-based approaches. Learning-based techniques achieve

remarkable results, reporting a significant increase in performance compared to learning-free techniques. This increase in performance is expected due to the existence of a training phase, during which the generated model is adjusted to the existing writing styles. In order to achieve the reported performance, learning-based methods require a considerable amount of available training data (training sets consist of 3645 and 30226 word images for GW3 and IAM datasets, respectively, without including possible augmentations).

(iii) Generalization of learning-based approaches. An important observation is that the results reported in the literature are dataset-oriented, i.e. the generalization of the created models is not explored or supported. This fact implies that if a model is trained and tested on different datasets (e.g. trained on IAM and tested on GW dataset), it may lead to a significant drop in performance, as shown in [8]. This observation is also verified in our work using the PHOCNet system [166] (Sudholt et al.**) trained on the IAM dataset and tested on GW3 and vice versa. As it was expected, the model trained on the simple GW dataset shows very poor performance on the challenging IAM dataset. On the contrary, the model trained on the IAM dataset, which includes many writing styles, appears to be more robust. Nevertheless, the gap in performance when compared with the models trained in GW3 is significant (nearly 20%), while it performs slightly worse compared to the proposed method. In conclusion, even though learning-based approaches are very successful, their generalization to unseen data coming from different collections needs to be further explored.

4.2.10 Experimental Evaluation on Competition Datasets

We further explore the generalization and efficiency of the proposed method on several competition datasets. Tables 4.5, 4.6 and 4.7 report the results concerning the ICFHR14, ICDAR15 and ICFHR16 competition datasets, respectively. Besides the proposed method (PSeq-mPOG+MISM), we also evaluate Holistic-mPOG+Eucl and PSeq-mPOG+SM variations in order to highlight the increase in performance when considering sequence matching and augmentation procedures. The competitions' participants are reported at the first part of each table, annotating the competitions' winners using the $\star$ symbol on the left side of the method's name. State-of-the-art methods which used the competition datasets for measuring their performance are reported on the second part of each table. The main observations are summarized below:

(i) The proposed method outperforms all learning-free approaches. Holistic-mPOG+Eucl approach performs reasonably well for the majority of the datasets, producing results comparable to state-of-the-art learning-free techniques. Nevertheless, sequential approaches (PSeq-mPOG + SM and PSeq-mPOG + MISM) demonstrate an outstanding boost in performance on all datasets. For example, the performance increase on the Modern14 dataset is over 17% on both metrics. Another remark is that although the MISM approach provides a considerable gain in performance, this is not consistent among all datasets. The reason of this inconsistency is correlated with the degree of success of the main-zone detection for different datasets. Overall, the proposed method achieves the best performance among every published learning-free method by a notable extent.

(ii) Learning-based methods trained and tested on different datasets (competitions ICFHR14 and ICDAR15). The attribute-based method [9], proposed by Almazan et al., was the competition winner and still outperforms all existing KWS methods on the Modern14 dataset. Even though this method uses a training step, the training was performed on independent datasets that include similar writing styles to the competition's datasets (GW and IAM datasets were selected for the Bentham14 and Modern14

	Bentham14		Modern14	
Method	**P@5**	**MAP**	**P@5**	**MAP**
Kovalchuk et al. [99]	73.8	52.4	58.8	33.8
⋆ Almazan et al.⋆⋆ [9]	72.4	51.3	70.6	**52.3**
Howe [86]	71.8	46.2	56.9	27.8
Zagoris et al. [189]	52.5	34.1	-	-
Zagoris et al. [190]	62.3	39.3	-	-
Aldavert et al. [6]	62.9	46.5	61.9	38.9
Sfikas et al. [158]	76.4	53.6	56.0	32.1
Retsinas et al. [144]	77.1	57.7	61.3	35.5
Zagoris et al. [191]	78.8	60.0	-	-
Holistic-mPOG+Eucl	77.5	60.4	52.3	29.7
PSeq-mPOG+SM	85.1	70.2	71.4	46.8
PSeq-mPOG+MISM	**85.5**	**71.1**	**73.5**	49.1

Table 4.5: Evaluation Metrics for Bentham and Modern datasets on ICFHR14 Competition. (**) : train and test on different datasets. (⋆) : competition winner.

Method	**P@5**	**MAP**
⋆ PRG [137]	46.0	42.4
CVC [137]	34.3	30.0
Zagoris et al. [189]	22.4	19.3
Zagoris et al. [190]	26.8	21.7
Almazan et al.⋆⋆ [9]	41.7	36.3
Sfikas et al. [158]	47.0	41.5
Retsinas et al. [144]	48.7	44.5
Zagoris et al. [191]	50.1	44.0
Holistic-mPOG +Eucl	48.1	44.2
PSeq-mPOG+SM	59.5	56.4
PSeq-mPOG+MISM	**61.6**	**58.4**

Table 4.6: Evaluation Metrics for Bentham on ICDAR15 Competition dataset. (**) : train and test on different datasets. (⋆) : competition winner.

Method	**Botany16**	**Konzils16**
CVCDAG⋆ [135]	75.8	77.9
⋆ PRG⋆ [135]	**89.7**	**96.1**
QTOB⋆ [135]	55.0	82.2
TAU [135]	50.6	71.1
Retsinas et al. [144]	46.7	56.5
Sfikas et al. [158]	46.5	59.9
Holistic-mPOG+Eucl	53.2	64.2
PSeq-mPOG+SM	57.0	71.1
PSeq-mPOG+MISM	58.3	76.2

Table 4.7: MAP Evaluation on ICFHR16 Competition datasets. (*) : train and test on the same dataset. (⋆) : competition winner.

datasets, respectively). The involved training step proved to be advantageous in the case of the Modern14 dataset, which consists of different languages and writers. However, the proposed method surpasses the attribute-based method in terms of P@5 on the Modern14 dataset. Furthermore, it achieves a gain over 20% on both Bentham datasets, two datasets for which method [9] does not perform equally well.

(iii) Learning-based methods trained and tested on the same dataset (competition ICFHR16). Learning-based methods, trained and evaluated on the same collection, report superior performance over learning-free methods, as it can be observed in Table 4.7, which summarizes the results for the ICFHR16 competition's datasets. This competition allowed participants to use different amount of training data, as an attempt to examine the generalization of their approaches. In addition, the training data was not completely manually annotated, i.e. the word segmentation step is performed automatically. For further details of the competition setup see [135]. It should be noted that the competition winner (*PRG*), used all the available training data. Even though the learning-based methods that participated on this competition are not directly comparable to our learning-free approach, there are cases for which the proposed method achieves similar results to the participants' methods (QTOB on Botany16 and CVCDAG on Konzils16).

4.2.11 Error Analysis

Even though the proposed method outperforms the existing learning-free keyword-spotting techniques, there are cases for which it does not produce adequate results. Two indicative examples are shown in Figure 4.12. This erroneous behavior is related to the calculated cost between two sequences (a weighted summation of the distance of zoned segments). Therefore, images of different words with identical segments may have lower cost compared to images of the same word in different writing styles.

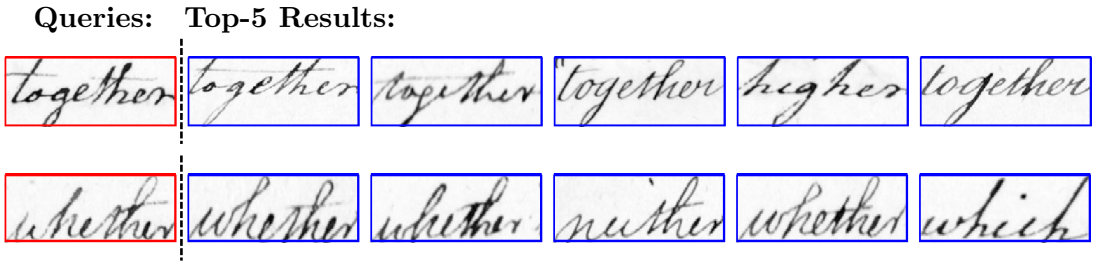


Figure 4.12: Two cases of erroneous retrieval: (top row) 80% P@5 (b) 60% P@5. Query is separated from the top 5 retrieved words with a dotted line. As it can be observed, PSeq-mPOG+MISM retrieves a set of words that have almost identical prefix or suffix.

Chapter 5

Manifold Embedding of Word Representations

Feature extraction is the most crucial step of a KWS method. Specifically, QbE KWS methods can be categorized, with respect to the extracted features, into (i) methods that extract a feature vector (descriptor) of fixed dimensionality for each word image and (ii) methods that extract a set of features for each word image. Methods of the former category, also called holistic word representations, attract a lot of interest due to their simplicity at the retrieval step. Such representations require a simple distance/similarity measure for the retrieval step (e.g. Euclidean distance) contrary to techniques belonging to the second category which require more complex matching algorithms (e.g. DTW sequential matching). Nevertheless, the majority of KWS methods that rely on holistic word representations generate high-dimensional descriptors (e.g. BoVW or CNN approaches). However, Euclidean distance on high-dimensional vectors is not a reliable metric for the generation of the retrieval list. This observation leads to the realization that a dimensionality reduction technique is essential in order to fully utilize the descriptive power of holistic representations. Preserving as much of the significant structure of the high-dimensional data as possible in the low-dimensional map is crucial and thus nonlinear dimensionality reduction techniques are required. An interesting property, which has proven to be effective, is to assume that the high-dimensional data lies on a manifold of significant lower intrinsic dimensionality. Thus, the computation of the low-dimensional map is equivalent to learning the underlying manifold. The generated nonlinear mapping is called manifold embedding ([171],[153],[15]).

Several manifold embedding methods have been reported in the literature aiming to generate a non-linear mapping which encodes high-dimensional data to a low-dimensional space without significantly affecting the local structure of the initial space. Unfolding successively the underlying manifold leads to many interesting and useful applications. An important and straightforward property of the new mapping is that the Euclidean distance is more meaningful, since the so called curse of dimensionality has been diminished. Notable manifold embedding techniques are Isomap [171], which creates an embedding based on geodesic distances, and Locally Linear Embedding (LLE) [153] as well as Laplacian Eigenmaps [15], which both assume the same local structure (linearity) for both the initial high-dimensional and the resulting low-dimensional space. Such techniques can be viewed as generalized eigenvector problems at adjacency matrices. The aforementioned techniques are sensitive to outliers as well as to the predefined dimensionality of the embedding space and consequently lead to the generation of low quality embeddings for the case of challenging datasets. On the contrary, t-SNE [182] has been extensively used on real

datasets, providing embeddings of high quality, even when the embedding dimensionality is lower than the intrinsic dimensionality of the underlying manifold.

The majority of the manifold embedding methods do not support the addition of a new sample to the already learnt embedding. This is referred as the **out-of-sample problem** for which many approaches have been proposed in the literature. Two main categories can be distinguished: parametric and non-parametric out-of-sample extensions. Parametric approaches assume that the learnt embedding can be modeled by a (non-linear) combination of the initial data along with a set of parameters [57], [180]. By estimating these parameters on the already extracted embedding, the out-of-sample extension is straightforward using the same model and the estimated parameters. The main disadvantage of such approaches is the assumption that the generated mapping can be efficiently represented by a (non-linear) model of ideally few parameters. On the other hand, non-parametric approaches usually exploit the geometric intuition of the local structure and the nature as well as specific characteristics of the selected manifold learning algorithm [19], [164], [28].

It should be noted that the KWS task, in essence a retrieval problem, is an ideal application of manifold embedding techniques due to the necessity of a credible distance metric between high-dimensional word image representations.

To the best of our knowledge, the only approach that utilizes manifold embedding for the task of KWS is the work of Sudholt et al. [165]. The authors of [165] proposed a variation of Isomap embedding for the case of Bag of Visual Words (BoVW) features. Although one can become aware of the efficiency of manifold embedding on the reduction of the descriptor's size without significantly affecting the retrieval performance, the presented system has some notable shortcomings mainly derived from the Isomap embedding, such as its sensitivity to the selection of the embedding dimensionality. Furthermore, the Isomap embedding requires the computation of geodesic distances, even for the out-of-sample scenario, which is a computational overhead for the retrieval step. In addition, although a significant reduction of the memory requirements has been achieved no consistent gain in retrieval performance was reported.

We tackle the KWS task by proposing a method [149] which relies on the well-known t-distributed Stochastic Neighbor Embedding (t-SNE) [182] due to its success on the dimensionality reduction task for a large variety of real datasets. The main hindrance for a t-SNE based KWS application is the out-of-sample extension, i.e. the addition of a new descriptor on the previously learnt embedding, i.e the embedding of the query representation. In fact, the majority of manifold learning approaches, including t-SNE, are non-parametric, meaning that no straightforward way exists to add a new descriptor to the embedding. In order to overcome this problem, we propose a novel out-of-sample extension to the t-SNE embedding. This extension enables us to utilize the t-SNE method and explore its efficiency for the KWS task.

5.1 t-SNE

The goal of t-SNE is to minimize the divergence between the pairwise similarity distributions of input points and the low-dimensional embedded points. The input points are denoted as $\{x_i\}$ and their corresponding embeddings are denoted as $\{y_i\}$, where $i = 1, \ldots, N$. The joint probability p_{ij} that measures the pairwise similarity between two points x_i and

x_j is denoted as follows:

$$p_{j|i} = \frac{\exp(-d(x_i, x_j)^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-d(x_i, x_k)^2/2\sigma_i^2)}, \quad p_{i|i} = 0 \tag{5.1}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{5.2}$$

For the rest of this work, the distance function $d(\cdot, \cdot)$ is considered to be Euclidean as in [182]. The standard deviation σ_i is computed according to a predefined perplexity which can be considered as the effective number of neighbors for each point x_i. The use of a predefined perplexity results to smaller values of σ_i for dense spaces around the point of interest (many neighboring points) and to larger values for sparse regions.

The pairwise similarities in the embedding space are modeled by a normalized Student's-t distribution with a single degree of freedom. This was the main contribution by the authors of the t-SNE [182] method compared to the SNE method [82], which used a normalized Gaussian distribution in accordance to the pairwise similarity of the initial space. The embedding similarity between two points y_i and y_j is defined as:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{Z}, \quad q_{ii} = 0 \tag{5.3}$$

$$Z = \sum_k \sum_{l\neq k} (1 + \|y_k - y_l\|^2)^{-1} \tag{5.4}$$

The choice of the Student's-t kernel prevents the crowding problem, as it is explained in [182], which favors embeddings whose points are gathered in the center of the space. The heavy-tailed Student's-t maps sufficiently well points that are far-apart even if the dimension of the embedding space is lower than the (unknown) intrinsic dimensionality of the existing manifold.

Given the definitions of pairwise similarity distributions for both the initial and the embedding space, the embedding $\mathbf{Y}$ is calculated by minimizing the Kullback-Leibler divergence:

$$C(\mathbf{Y}) = KL(P||Q(\mathbf{Y})) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{5.5}$$

The aforementioned minimization problem does not have an analytical solution. To this end, iterative methods are employed in order to find an embedding $\mathbf{Y}$ that (locally) minimizes the divergence. The problem is solved by a gradient descent method, whereas the gradient of the divergence for each point of the embedding space is computed as follows:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j\neq i} (p_{ij} - q_{ij}) q_{ij} Z (y_i - y_j) \tag{5.6}$$

where $Z = \sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}$.

5.2 Out-of-sample (OOS) Extension of t-SNE

As we noted earlier, under the manifold assumption, a retrieval step based on the euclidean distance should be more effective/meaningful compared to using euclidean distance on the high-dimensional raw data. A QbE retrieval on the embedding space requires an out-of-sample addition of the query to the previously learnt embedding and subsequently

a sorting of the euclidean distance between the generated query feature vector and the already embedded words features. It should be noted that out-of-sample embedding should be fast for KWS applications, because it would be computed on query (retrieval) time.

We assume a set of points x_i, which correspond to the descriptors of the word images for the KWS task, and their embeddings y_i as the result of the t-SNE optimization. Due to the non-parametric formulation of t-SNE an out-of-sample extension is not straightforward. Given a new point x in the initial space, our goal is to estimate its mapping y to the t-SNE embedding space. We define the following auxiliary functions in accordance to the t-SNE formulation:

$$p(x|x_i) = \frac{\exp(-\|x - x_i\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_k - x_i\|^2/2\sigma_i^2)} \tag{5.7}$$

$$p(x, x_i) = \frac{p(x|x_i) + p(x_i|x)}{2N}, \quad p(x_i, x_i) = 0 \tag{5.8}$$

$$s(y, y_i) = (1 + \|y - y_i\|^2)^{-1} \tag{5.9}$$

$$q(y, y_i) = s(y, y_i) / \sum_k \sum_{l \neq k} s(y_k, y_l) \tag{5.10}$$

A straightforward solution to the out-of-sample problem is to preserve the local structure of the initial space [153] which can be formulated as the minimization of the cost:

$$C_{oos}(y|x) = \sum_i w(x, x_i) \|y - y_i\|^2 \tag{5.11}$$

All previously learnt embeddings y_i are considered fixed, so we minimize over the sought embedding y. The function $w(x, x_i)$ is a pairwise similarity function (e.g. a Gaussian kernel) and in correspondence to t-SNE, the previously defined function $p(x, x_i)$ can be used. The above minimization has a closed form solution:

$$y^\star = \frac{\sum_i p(x, x_i) y_i}{\sum_i p(x, x_i)} \tag{5.12}$$

A drawback of this solution, as well as of the majority of the existing OOS methods, is that it provides a general approach for the OOS problem (i.e. locality preservation of the initial space) while ignoring crucial aspects of t-SNE success, namely the Student's-t distribution and the locality of the embedding space. Contrary to existing approaches, in order to address the out-of-sample problem, we examine the initial equations of t-SNE.

Proposed Gradient Descent Approach: The estimation of the new embedding y is computed iteratively by minimizing the t-SNE cost according to a gradient descent procedure:

$$y^{t+1} = y^t - \alpha \frac{\partial C(y^t)}{\partial y^t} \tag{5.13}$$

$$\frac{\partial C(y)}{\partial y} = 4 \sum_i [p(x, x_i) - q(y, y_i)] s(y, y_i)(y - y_i) \tag{5.14}$$

The main shortcoming of a gradient descent estimation is its convergence rate. If a fixed step size a is predefined, the convergence may be extremely slow. In order to avoid a slow convergence, we propose the use of the following adaptive step size:

$$\alpha(y^t) = \left[4 \sum_i p(x, x_i) s(y^t, y_i)\right]^{-1} \geq 0 \tag{5.15}$$

Therefore, the update equation for iteratively estimating the embedding y is:

$$y^{t+1} = y^t - \frac{\sum_i [p(x, x_i) - q(y^t, y_i)] s(y^t, y_i)(y^t - y_i)}{\sum_i p(x, x_i) s(y^t, y_i)} \tag{5.16}$$

It should be noted that the update equation (Eq. 5.16) can be derived from the solution of $\|\frac{\partial C(y)}{\partial y}\| = 0$ and thus it is equivalent to a fixed point iteration approach.

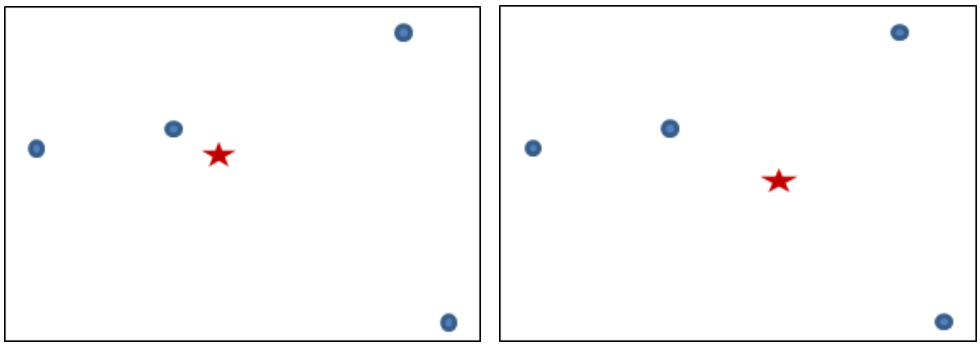
Aiming to further promote the simplicity of the update equation and the speed convergence, we choose to omit the terms of Eq. 5.16 referring to $q(y^t, y_i)$. The term $\sum_i q(y^t, y_i) s(y^t, y_i)(y - y_i)$ corresponds to the derivative of the normalizing term $Z = \sum_k \sum_{l \neq k} s(y_k, y_l)$ and it is responsible for keeping the new embedding y sufficiently apart from the embeddings y_i, as a repulsive force. Concerning retrieval applications, only the relative distances between the new embedding and the already embedded points are of interest and thus this repulsion property is not important. Consequently, for the rest of this work, the proposed out-of-sample embedding is approximated by minimizing the cost $C_r(y|x) = \sum_i p(x, x_i) log\big(p(x, x_i)/s(y, y_i)\big)$ which is performed by the following update equation:

$$y^{t+1} = \frac{\sum_i p(x, x_i) s(y^t, y_i) y_i}{\sum_i p(x, x_i) s(y^t, y_i)} \tag{5.17}$$

The adaptive step size $a(y)$ for the latest update equation can be easily proven to concede with the optimum step size for the line search strategy over the gradient descent algorithm. This means that the acquired step size of Eq. 5.15 at each iteration is the solution to the minimization problem:

$$\alpha(y^t) = \arg\min_{\alpha > 0} C_r\big(y^t - \alpha \frac{\partial C_r(y^t)}{\partial y^t}\big) \tag{5.18}$$

The aforementioned observation ensures significantly faster convergence compared to setting a predefined step size, which was empirically verified through experimentation. It was observed that the majority of out-of-sample embeddings converged under ten iterations with random initial y^0.



(a) (b)

Figure 5.1: Visualization of the proposed out-of-sample embedding in two dimensions for (a) using only the F_{attr} term (b) using both F_{attr} and F_{rep} terms. The red star is the final embedding and the blue circles are the embedding of its neighborhood. The repulsive term generates embeddings that are not close to theirs neighbor points in the embedding space.

Implementation Issues: The computation of $q(y, y_i)$ is straightforward at each iteration. However, $p(x, x_i)$ is calculated only once, before the iteration process, and involves

summations over all the pairwise Gaussian functions. To overcome this problem, we store the standard deviations σ_i and the partial sums $S_i = \sum_{k \neq i} exp(-\|x_k - x_i\|^2/2\sigma_i^2)$ as auxiliary variables generated during the t-SNE embedding. Having estimated the standard deviation σ for the unseen point x (using the predefined perplexity), we redefine the equations of $p(x, x_i)$ as follows:

$$p(x_i|x) = \frac{\exp(-\|x - x_i\|^2/2\sigma^2)}{\sum_{k=1}^{N} \exp(-\|x - x_i\|^2/2\sigma^2)} \tag{5.19}$$

$$p(x|x_i) = \frac{\exp(-\|x - x_i\|^2/2\sigma_i^2)}{S_i + \exp(-\|x - x_i\|^2/2\sigma_i^2)} \tag{5.20}$$

$$p(x, x_i) = \frac{p(x|x_i) + p(x_i|x)}{2N}, \quad p(x_i, x_i) = 0 \tag{5.21}$$

The above formulation requires only the distances of the new point x from the existing points x_i, i.e. O(N) computations.

Complexity: Given a set of N points $\{x_i\}$ of d_x dimensions and their embeddings $\{y_i\}$ of $d_y \ll d_x$ dimensions, the complexity of computing the embedding y of an out-of-sample point x is estimated as follows:

- $O(Nd_x)$ for computing the $p(x, x_i)$ pairwise similarities.
- $O(Nd_y)$ for updating y in each iteration.

Assuming k as the total number of iterations for convergence, the overall complexity is $O(N(d_x + kd_y))$. For small embedding dimension d_y and number of iterations ($d_y = 3 \,\&\, k = 10$), the computation of the pairwise similarities in the initial space, when d_x is large enough which is usually the case, governs the computation time ($d_x \gg kd_y$). This observation hints that the proposed OOS extension is only sligthly slower than the closed form solution approach of Eq. 5.12. It should be noted that the OOS embedding procedure should be fast for KWS applications, because it is computed during query (retrieval) time. Memory requirements correspond to storing the initial data points and their embeddings, as well as the standard deviations σ_i and sums S_i, i.e. $O(N(d_x + d_y + 2))$, which requires only $N \times (d_y + 2)$ more memory space compared to storing only the initial data points.

5.3 Experimental Evaluation

5.3.1 Experimental Setup

The proposed OOS extension of t-SNE embedding method is applied on QbE keyword spotting as a post-processing step after the extraction of fixed-sized descriptors. In order to highlight the efficiency of the manifold assumption and the capability of the proposed method, its evaluation is performed on three state-of-the-art descriptors. The performance of the KWS task was recorded in terms of the Precision at Top 5 Retrieved words (P@5) as well as the Mean Average Precision (MAP).

The workflow for the application of t-SNE embedding on KWS includes the following steps:

1. Extract the descriptors for each word image of the dataset.
2. Perform Principal Component Analysis (PCA) on the dataset descriptors. This is suggested before using t-SNE because it preserves the global structure of the

points/descriptors and reduces noise. For this work, the feature vector dimension after PCA is set to $d_{pca} = 400$ regardless the initial dimension of the descriptors (selected descriptors have a dimensionality over 400).

3. Perform t-SNE embedding on the descriptors. The accelerated version of t-SNE is selected, which uses tree-based structures [181]. Due to the fact that the t-SNE approach generates an embedding which corresponds to a local minimum of the t-SNE cost optimization problem, the process was repeated multiple (five) times with different (random) initialization. The embedding with the lowest cost was selected as the final embedding.

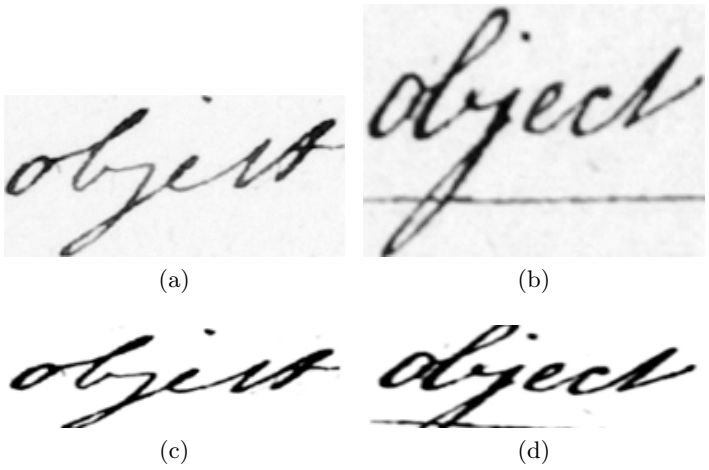
4. Compute the auxiliary values σ_i and S_i, which are required in the proposed out-of-sample extension.

5. Given a query image, compute the initial descriptor and perform PCA. The resulting descriptor is used as input (along with descriptors of the dataset, their corresponding embeddings and the auxiliary variables) to the out-of-sample estimation method.

6. Generate the retrieval list using Euclidean distance on the embedding space.

5.3.2 Preprocessing and Descriptors

Before we proceed with feature extraction, we apply a preprocessing step which consists of contrast and main-zone normalization. Contrast normalization is performed by replacing Sauvola's binarization hard assignment with a soft one. Main-zone normalization is based on detecting the main-zone in a way similar to [144]. After the detection of the main-zone, skew correction is performed using the slope of the detected main-zone, as well as a vertical normalization of the image by moving the main zone at the center of the generated normalized image. An example of the effect of the aforementioned preprocessing step is depicted in Figure 5.2.



(a) (b)

(c) (d)

Figure 5.2: Different instances of the same word before (a),(b) and after (c),(d) the preprocessing.

Three state-of-the-art holistic descriptors are selected, which are briefly described below:

BoVW: A Bag of Visual Words (BoVW) approach organized in Spatial Pyramids was implemented due to the established efficiency of such methods in keyword spotting [6]. Dense SIFT features at multiple scales were chosen as local descriptors and a codebook of 1024 entries for the histogram encoding. Spatial pyramids are employed to encode indirectly the spatial information as in [6].

POG: An image is segmented in three (overlapping) parts and each segment is encoded using the Projections of Oriented Gradients (POG) descriptors, which have shown to perform well in keyword spotting [144]. In this work, a slight modification of POG descriptor is used in order to be applied to gray-scale images.

ZAH: Zoning Aggregating Hypercolumns (ZAH) features are based on a pre-trained Deep Convolutional Network (DCN) [158]. The features are extracted from the output of the convolutional layers of a DCN, which was trained on an independent set of typewritten characters. The final descriptor is produced by the concatenation of the aggregated convolution responses over (six) image segments.

		GW20		**Bentham14**		**Modern14**	
Descriptor	**OOS approach**	MAP	P@5	MAP	P@5	MAP	P@5
BoVW	No Embedding	72.30	91.59	55.29	74.50	28.93	50.60
	CFS	80.10	82.22	48.42	49.50	19.67	23.00
	Parametric-90	82.59	88.58	35.93	32.25	12.81	13.00
	Parametric-100	85.13	92.30	38.13	35.12	13.62	15.33
	Proposed	**85.18**	**92.42**	**63.57**	**78.00**	**33.93**	**53.87**
POG	No Embedding	62.49	**85.85**	66.01	**82.25**	36.83	61.80
	CFS	68.63	74.51	56.09	53.69	29.81	28.60
	Parametric-90	70.38	79.82	55.67	57.12	35.21	47.73
	Parametric-100	74.15	85.04	60.77	64.31	43.64	57.13
	Proposed	**74.19**	85.14	**70.43**	80.94	**48.21**	**65.80**
ZAH	No Embedding	61.19	86.40	**53.49**	75.69	33.29	56.33
	CFS	70.69	76.45	37.41	44.06	24.65	30.07
	Parametric-90	74.10	84.44	28.51	33.56	30.30	43.33
	Parametric-100	76.77	89.13	34.95	42.62	33.08	47.87
	Proposed	**76.82**	**89.23**	53.17	**76.75**	**38.84**	**59.93**

Table 5.1: MAP and P@5 Evaluation on all Datasets for $d_y = 3$

5.3.3 Out-of-Sample Approaches

Concerning the efficiency of the proposed OOS method, the following OOS embedding methods have been considered for comparison:

CFS: Out-of-sample extension using Eq. 5.12. This approach assumes that the local structure of the embedding space is defined only by pairwise similarities of the initial points [153].

Parametric: A parametrization between the initial data and the produced embeddings is introduced according to [57], where the parametric form $y(x) = f_a(x)$ is assumed and a are the sought parameters. Non-linearity is introduced by Gaussian kernels of the form $k(x, x_i) = exp(-\|x - x_i\|^2/2\sigma_i^2)$. Thus, the parametrization is defined as $y(x) = \sum_i a_i k(x, x_i) / \sum_l k(x, x_l)$. The parameters are estimated in a least square manner: $A = K^+ Y$, where $[K]_{ij} = k(x_i, x_j) / \sum_l k(x_i, x_l)$. The parametric approach of [57] suggests

using only a set of landmark points, i.e. a subset of the initial points, which alleviates the computation overhead of inverting a matrix of size $N \times N$. However, in practice, selecting a subset of the initial space leads to poor performance. To highlight this behavior, 90% of the points are randomly selected in order to estimate the parameter matrix A. It should be stressed that 90% is a very high percentage of points kept, which yields no significant computational acceleration. As a result, we distinguish two variations, **parametric-100** and **parametric-90**, where 100% (all) and 90% of the points are used as landmark points, respectively.

Proposed: Out-of-sample extension using gradient descent (Eq. 5.17) based on the initial t-SNE cost function. The proposed step size is adaptive and optimal according to the line search strategy, which guarantees fast convergence. The maximum number of iterations is set to $N_{max} = 15$, since the majority of the out-of-sample experiments achieve convergence under 10 iterations.

5.3.4 Performance Evaluation

To verify the efficiency of the proposed OOS extension, we apply the aforementioned OOS methods on all the descriptors and datasets for the case of $d_y = 3$ (embedding dimension). The results, in terms of MAP and P@5, are presented in Table 5.1. The *No Embedding* case corresponds to the absence of a manifold embedding step, i.e. the PCA generated descriptors are used. The main observations are summarized below:

- The proposed OOS extension performs significantly better compared to the other OOS methods, especially in the challenging Betham14 and Modern14 datasets.

- The parametric method shows similar performance only on the GW20 dataset, which is smaller and less challenging, while in the other two datasets its performance deteriorates significantly. This leads to the conclusion that the parametric approach of [57] cannot model the t-SNE embedding sufficiently well. Specifically, the *parametric-90* variation reports a considerable drop in performance, even though 90% of all points are used. This observation hints that the use of landmark points yields unreliable parameters for parametric OOS extension.

- Another important observation is that the performance may drop after the use of manifold embedding compared to the case of using the initial descriptors (No Embedding case). This drop in performance is mainly credited to t-SNE embedding of the word descriptors, rather than the OOS methods. It is possible that the selected embedding dimensionality is much lower than the intrinsic dimensionality of the underlying manifold and thus the generated embedding is not suitable. It should be noted that the intrinsic dimensionality depends on the descriptor and the dataset selection, since both define the initial space.

  Furthermore, we investigate the importance of the embedding dimension d_y. Tables 5.2, 5.3 and 5.4 summarize the results for different embedding dimensions concerning the GW20, Bentham14 and Modern14 datasets, respectively. The proposed OOS extension is used to obtain the low-dimensional embedding of the query image. In addition, the performance of state-of-the-art KWS methods are provided for comparison. The main observations are summarized below:

- A significant gain in performance, more than 10% in some cases, is observed for the majority of the datasets and the descriptors, when using the t-SNE embedding.

- It can be observed that only in few cases the overall gain is small. In addition, the efficiency increases along with the embedding dimension. This behavior hints towards a higher intrinsic manifold dimensionality.
- The presented KWS approach provides results that outperform the majority of state-of-the-art techniques without any fine-tuning (Aldavert et al. [6] performed fine-tuning on the GW20 dataset). A noteworthy observation is that these results have been reported using a very low embedding dimensionality ($d_y = 2, 3, 4, 5$), which highlights the efficiency of the t-SNE method.

Method	No Embedding	$\mathbf{d_y}$ 2	3	4	5
Kovalchuk [99]	66.30	-	-	-	-
Aldavert [6]	76.50	-	-	-	-
BoVW	72.30	84.12	85.35	85.62	86.23
POG	62.49	70.76	74.01	74.79	74.96
ZAH	61.19	74.85	78.51	78.43	79.54

Table 5.2: MAP Evaluation on GW20 Dataset

Method	No Embedding	$\mathbf{d_y}$ 2	3	4	5
Kovalchuk [134]	52.40	-	-	-	-
Almazan [134]	51.30	-	-	-	-
Howe [134]	46.20	-	-	-	-
fPOG [144]	57.70	-	-	-	-
Aldavert [6]	46.50	-	-	-	-
BoVW	55.29	61.02	62.38	64.82	64.81
POG	66.01	67.95	70.68	70.66	71.54
ZAH	53.49	53.30	54.46	53.83	54.07

Table 5.3: MAP Evaluation on Bentham14 Dataset

Method	No Embedding	$\mathbf{d_y}$ 2	3	4	5
Kovalchuk [134]	33.80	-	-	-	-
Almazan [134]	52.30	-	-	-	-
Howe [134]	27.80	-	-	-	-
fPOG [144]	35.50	-	-	-	-
Aldavert [6]	38.90	-	-	-	-
BoVW	28.93	34.09	34.56	35.64	36.29
POG	36.83	48.82	50.51	49.39	51.61
ZAH	33.29	39.30	40.02	39.50	40.65

Table 5.4: MAP Evaluation on Modern14 Dataset

Chapter 6

Word Representations using Deep Features

Up until now, we have only considered either learning-free methods or unsupervised approaches (manifold learning) in order to extract discriminative features. Such approaches cannot effectively tackle the prominent challenge of handwritten documents: the vast variety of writing styles. To this end, in this chapter, we rely on supervised learning, and specifically deep learning, in order to generate word representations independent of the writing style. Our ultimate goal is the appropriate/efficient utilization of deep neural networks for the task of keyword spotting and the validation of their generalization capability, i.e. if we can successfully apply the same network on different datasets without a re-training step.

Convolutional Neural Networks (CNNs) consist the backbone of every approach presented in this chapter. PHOCNet [166] serves as the initial reference network, which lends to several ideas through this chapter. CNNs can be used straightforwardly as optimal feature extractors (designed to optimize the task at hand) and the exploration of the generated deep features, extracted at different layers, is a major research direction of this thesis. During our initial exploration of the PHOCNet architecture, we progressively trace several critical aspects of deep learning, related both to the architecture selection and the training procedure, and how they affect the network's generalization. Following this analysis, we consider compressing specific CNN layers in order to generate compact deep features, avoiding the existing over-redundancy of state-of-the-art CNNs. Notably, we propose a novel approach to generate discriminative word representations by using a Seq2Seq word recognition system. By combining the recognition techniques with their spotting counterparts, we outperform existing spotting techniques, while the generated representations can be fully decoded into sequence of characters (i.e. word recognition). Finally, we utilize the recurring ideas of feature-based representations in order to apply keyword spotting over a complete line rather than pre-segmented words.

6.1 Deep Features: Extracting Features from CNNs

Recent deep learning approaches, addressing the KWS task, have similar traits: 1) an appropriate text-realistic augmentation 2) an off-the-self architecture with minor modifications if necessary (lending from the image classification success on Imagenet [154]) and 3) a text embedding technique, leading to fixed-sized representations as targets (e.g. PHOC [9]). An exemplar case of such an approach is PHOCNet [166], which will be

the center of interest of this exploratory study (for more details about the PHOCNet architecture see Chapter 2).

In essence, PHOCNet is used as a text embedding estimator. Both QbE and QbS paradigms are available by comparing the PHOC estimations, generated by the network. Nevertheless, features can be also acquired from previous hidden layers and not only the last layer that produces the text embeddings. It has however been noted that layers other than the output layer can be used to produce features. These features, corresponding to one (or more) hidden layers of the network, have been commonly known as deep features in the literature [96](or hypercolumns [73, 158], when more than one layer activations are combined). In other machine vision tasks, deep features have often led to superior results compared to the standard use of the network [96], since they are capable to capture more abstract traits of the input [193]. Our first goal is to explore which layer performs best as feature extractor, under the assumption of a language-independent, well-generalizing QbE KWS system. In order to further enhance the efficiency of the extracted high-dimensional features, we also utilize the manifold embedding approach of the previous chapter.

6.1.1 Analysis Method and Model Parameters

We assume a Query-by-Example (QbE) segmentation-based KWS scenario and the Mean Average Precision (MAP) evaluation metric. This means that all data -both training and test- are segmented word images, and queries are word images as well. After having trained the reference CNN, i.e. PHOCNet, the extracted Deep Features are defined simply as the activations of a hidden layer, when a specific word image input is provided. Given all word images, plus the query, we thus create descriptors –one for each word image– based on deep features. These descriptors are typically of high dimensionality, with the exact number of the latter depending on the number of neurons per hidden layer [1]. Hence, a dimensionality reduction method can be applied to reduce the dimensionality of the descriptor. The final descriptors can then be compared using some measure of dissimilarity, in order to provide the retrieved query list. In the following subsections we explain these steps in more detail.

6.1.2 Neural Network Architecture and Deep Features

PHOCnet is a standard feed-forward neural network. The word image to be processed is fed to the network input, with information flowing first through a number of alternating standard convolutional and max pooling layers. The size of all these layers depends on the size of the input image. The last convolutional layer is then fed to a spatial pyramid max pooling layer (SPP) [76]. The SPP layer (referred to as *spp*) produces a fixed-size output given a variable-size input, as it processes input from the previous layer after partitioning it into a hierarchy of grids of variable resolution $(4 \times 4, 2 \times 2, 1 \times 1)$. The SPP property of producing a fixed-size output regardless of the input, is in a way inherited by the whole model. In this manner, there is no need either to scale the input image to a fixed size or perform some manual zoning step afterwards. The output of the SPP is fed to two fully connected layers, coupled with ReLU non-linearities (referred to as $fc1$, $fc2$). The final fully connected layer generates the PHOC estimation. All layers between the input layer and the SPP layer are of variable size, as they depend on the input word image size. Therefore, we use *spp*, $fc1$ and $fc2$ to extract deep features (the features are acquired after the ReLU nonlinearity). We should note that as we get closer to the output layer,

[1] For example, the dimensionality of the features generated from layer *spp* (the closest to network input) is $10,752$.

the features are more adapted to the PHOC labels, which may affects generalization. For further details on PHOCNet architecture see Chapter 2 and [166].

6.1.3 Manifold Learning and Dissimilarity Measures

We use t-SNE as our non-linear manifold learning technique of choice. In t-SNE, as we described in the previous chapter, the goal is to minimize the divergence between pairwise similarity distributions of input points and the low-dimensional embedded points. The N input points are denoted as $\{x_i\}_{i=1}^{N}$ and their corresponding embeddings are denoted as $\{y_i\}_{i=1}^{N}$. The joint probability p_{ij} that measures the pairwise similarity between two points x_i and x_j is defined as $p_{ij} = (2N)^{-1}(p_{j|i} + p_{i|j})$, with $p_{j|i} \propto exp(-d(x_i, x_j)^2/2\sigma_i^2)$. A typical choice for $d(\cdot, \cdot)$ would be the Euclidean distance. In this analysis, we will experiment with other distances as well.

We perform comparisons on all extracted features using three different dissimilarity measures: The Euclidean distance (L2), the Bray-Curtis dissimilarity (BC), and the normalized Euclidean distance (L2-normal). As we employ manifold learning to reduce feature dimensionality, we use the chosen measure to learn the manifold by plugging it into the related similarity equation for p_{ij} (see previous subsection). Comparisons on the reduced space are always performed using the Euclidean space. BC is defined as $d_{BC}(a, b) = \frac{\sum_i |a_i - b_i|}{\sum_i a_i + b_i}$. It has originally been proposed as a measure of distance between histograms [10]. In practice, it can be used with any nonnegative-valued pairs of vectors [2].

The normalized L2 distance (L2-normal) is the Euclidean distance computed over L2-normalized versions of the original vectors. This is tantamount to what is referred to in the literature as cosine or dot-product distance. In other words, the angle between vectors is measured, regardless of the magnitudes of the compared vectors. Given normalized vectors, this distance is easy to compute, as it only involves computing a dot product. That is the reason why numerous authors seem to prefer it over the Euclidean distance (e.g. [9]), while its use can also lead to slightly improved performance.

6.1.4 Experimental Evaluation

For our experiments we have used the GW20, IAM, Bentham14 and Modern14 sets (see Chapter 2). We have first trained our model on the IAM database, and run tests over itself (cross-validated folds) as well as GW20, Bentham14, Modern14. The parameters of the test were (a) extracting deep features from different network layers (we compare $spp, fc1, fc2$) (b) applying t-SNE to produce low-dimensional embeddings or not (c) using different dissimilarity measures (we compare BC, L2, L2-normal). Prior to applying t-SNE, we first compute projections with PCA. As BC is suitable for use only with non-negative vectors [3], we do not apply PCA in that case. The PCA projection and the t-SNE embedding dimension was fixed in all cases to 400 and 4 respectively.

Table 6.1 contains the results of the IAM trained network, when evaluated on both IAM and GW, for different output feature layers and different dissimilarity measures. The main observations are:

- First, embedding with t-SNE gives in most cases a slight up to considerable boost.

[2] We must note that BC is not a distance metric in the strict mathematical sense, as it does not adhere to the triangle inequality [10]. However, for all practical intent, at least in the scope of the current application, this is not a problem.

[3] Preliminary tests of the BC on real-valued vectors have shown that performance deteriorates severely.

	GW20		
	BC	L2	L2-normal
spp	(76.3)80.4	(74.9)78.4	(77.6)81.4
fc1	(82.1)84.1	(75.6)81.8	(79.0)**84.4**
fc2	(80.2)82.4	(74.7)79.5	(78.3)81.9
output	(76.0)78.4	(72.7)73.4	(74.9)75.9

(a)

	IAM		
	BC	L2	L2-normal
spp	(59.5)63.6	(62.0)67.0	(68.4)72.8
fc1	(74.6)79.4	(70.0)75.6	(78.2)**81.5**
fc2	(75.5)80.0	(69.4)74.2	(78.03)81.2
output	(76.1)79.5	(73.5)74.4	(76.7)77.6

(b)

Table 6.1: MAP results of KWS trials over different test sets, different layers to extract deep features, and different dissimilarity measures. PHOCnet trained on IAM was used in all cases. Figures are results obtained without using t-SNE embeddings.

- Concerning the best layer choice, deep features consistently outperform the "output" of the NN.
- Regarding the choice of dissimilarity measure, L2 gives the worst results. Both Cosine (L2-normal) and BC achieve improved results.

In table 6.2 we present a list of the best performing deep features, and a comparison of their performance against extracting descriptors from the output layer. Note that in almost every case, deep features give the best performing descriptors. The difference between best layer and output in terms of MAP, can reach more than 17% (table 6.2, train on IAM/test on Modern14). A gain of $5 - 10\%$ performance on average seems to be the norm. We report only results over t-SNE embedded features, which have shown slightly better performance as we have seen on the previous results; however they are strongly correlated to the results obtained without applying t-SNE. Only the output of the $GW20$-trained net seems to perform better than deep features, with a negative difference (-1.3%). This can be explained due to the limited style variability of $GW20$, which leads to overfitted high results on itself and low results on other datasets. Bentham14 seems to be an exception, at least to a certain extent. This can be attributed to the similar style of these two datasets (for example, [9] had trained a model on GW20 and tested it on Bentham14 with considerable success [134]).

Interestingly, deep features give improved performance even in cases when using the model output would give almost *zero* MAP (for example, table 6.2, train on GW/test on IAM or Modern14). The case of training on GW20 is indeed *particularly noteworthy*: the same model gives figures close to 100% when tested and trained on different folds of the same database, but results become very inadequate when tested on input coming from a different database. This is evidently a case of overfitting. In our opinion, this result is quite alarming, as many recent works have focused on obtaining the best figure with training and testing on the same database [56], neglecting to report how the same model would fare if tested on a different set.

On the other hand, deep features comparatively show better performance, even when using a model that has overfit on a specific set. This validates the point that deep features

	Deep features	PHOCNet output	Boost
GW20	84.4($fc1$)	75.9	+8.5
IAM	81.5($fc1$)	77.6	+3.9
Bentham14	87.8($fc1$)	79.8	+8
Modern14	91.1($fc1$)	74.0	+17.1

(a)

	Deep features	PHOCNet output	Boost
GW20	95.3($fc2$)	96.6	−1.3
IAM	15.0(spp)	2.8	+12.2
Bentham14	72.0($fc2$)	64.5	+7.5
Modern14	16.4(spp)	8.4	+8.0

(b)

Table 6.2: Performance comparison (MAP%) of deep feature performance vs standard NN output. Results using models trained on IAM (a) and GW20 (b) are presented. The layers related to the best deep features are indicated, along with the boost compared to performance using standard NN output.

Method	Bentham14	Modern14
Kovalchuk et al. [99]	52.4	33.8
Almazán et al. [9]	51.3	52.3
Howe [86]	46.2	27.8
Retsinas et al. [144]	57.7	35.5
Sfikas et al. [158]	53.6	32.1
PHOCNet output	79.8	74.0
Deep Features	**87.8**	**91.1**

Table 6.3: Comparison versus state-of-the-art KWS methods. Results using the IAM-trained PHOCnet model are used to extract deep features.

are transferable and lead to less specific, more general features compared to the network output.

Another notable conclusion is that features obtained from the NN trained on IAM are considerably transferable. We attribute this fact to the rich variability of the IAM dataset, which contains data coming from hundreds of writers, compared to the much smaller and less variable GW20 set. Furthermore, in this case, the t-SNE embedding achieves a noteworthy gain, regardless the layer or the dataset used. Deep features extracted using the NN trained on IAM, outperform all other methods in the literature compared on the sets of ICFHR'14 by a significant margin (table 6.3). Note that Modern dataset contains instances of four different languages, while the network is trained on IAM, which contains only English.

Moreover, we can finally quantify the vast improvement of deep learning techniques compared to state-of-the-art techniques, which either are learning-free (e.g. POG [144]) or trained with traditional machine learning techniques (e.g. SVMs [9]). One should observe that feature transfer from typewritten characters, as done in ZAH approach, is not effective, a result similar to networks trained on GW20.

6.2 An Exploration on Architecture and Training Strategies

In this section, we aim to review some of the most crucial architectural and training choices made when training CNNs, assuming word-level segmented images. As baseline network we use PHOCNet, since we already have evaluated its efficiency and generalization. We examine the pros and cons of each choice, and whether employing a particular technique outweighs the drawbacks, if any. In particular, we explore different strategies of input image resizing and combining with different strategies of spatial pooling, and we argue that a non-pyramidal, zoning approach can be equally if not more useful while being less computationally expensive. We also validate the extent of augmentation's impact on performance. Along with the augmentation scheme, we evaluate the necessity of a class balancing scheme (given that text words are notably miss-balanced). Finally, we highlight the effectiveness of ensemble learning [22] under the KWS context.

6.2.1 Considered Network Architecture Choices

Input size One difference of the PHOCNet architecture versus the "very deep" VGG model [160] is the addition of the SPP layer (see discussion in Chapter 2) between the convolutional backbone and the network head. The SPP layer bears the special trait of transforming variable-size inputs into fixed-size outputs. In a sense, the whole model inherits this trait: the PHOCNet architecture indeed accepts variable-size word image inputs and produces outputs of the same size. This trait leads to the seemingly important advantage of not resizing the input image to a fixed size (as done in [160] for example, where all images are resized to 224×224 pixels). One could argue that not resizing the input is an obvious advantage, as resizing will change the aspect ratio and scale of the input. However, this advantage is not as clear as one may expect. We propose two different strategies concerning the input image, besides keeping the initial input size the same: a) resizing to a fixed height while keeping the aspect ratio of the original image, b) resizing to a fixed image size. It should be noted that a sufficiently deep network with a vast number of parameters and filters can model specific word text image structures (e.g. characters) at various aspect ratios in one class.

Spatial pooling strategy The original SPP layer considers a pyramidal hierarchy of layers, where each layer pools from progressively less divisions of the input (in our case, the output of the convolutional backbone). An alternative layout of pooling has been proposed in [167], where the input is partitioned only across the horizontal direction. This layer has been dubbed Temporal Pyramidal Pooling (TPP) layer.

Since the standard layer operation at all levels of the hierarchy is max-pooling, the output of the SPP will contain largely redundant information. For example, consider an input that is split into 4 equal-size horizontal zones on one hierarchical level and 2 equal-size horizontal zones on the next level. All the information on a coarser level already exists on a finer level. The redundancy of the pyramidal pooling output is our motivation to consider only a single finer level of pooling. Hence, this strategy is akin to zoning, where a pooling operation is applied on a non-hierarchical partition of the input into spatial zones. Replacing pyramidal pooling with single-layer pooling/zoning results in a significant reduction of the number of learning parameters. This leads to a smaller parameter search space and in principle to an easier optimization. In the current model, replacing pyramidal pooling translates to a reduction of up to 20 million parameters.

Nevertheless, we have to note that pyramidal descriptions can be very useful, under

a specific context. Bag of Visual Words (BoVW) approaches are one well-known example. In that case, a pyramidal / hierarchical scheme is used to construct a descriptor; the redundancy trait of this descriptor helps in enabling matching in different levels of coarseness, which in turn lends to the robustness of the descriptor. However, in the context of convnets, where they are used as a non-output, intermediate layer, the picture is slightly different. The linear transformation of the pyramidal output before the non-linearity of the first fully connected layer can be trivially transformed to an equivalent linear transformation where we consider a single layer / zoning output. Hence, we can argue that in this case, pyramidal pooling does not have as much usefulness, if any.

We also consider a model where we dispense altogether with zoning and pyramidal pooling techniques. The alternative would be to simply flatten the output of the convolutional backbone and use that as input to the fully-connected layers. However, since these pooling modules are necessary when working with variable-size input, this is applicable only when we choose to resize to a fixed-size image input.

6.2.2 Considered Network Training Choices

Training neural networks can lead to the undesirable event of overfitting to their training set. Neural networks are notorious for supposedly being "data-hungry", in the sense of requiring vast amounts of data during training. Dropout is one standard technique that can improve network generalization. Another standard practice in the literature is data augmentation, used to artificially create a bigger training set. Augmentation has been used with its class balancing variant.

Augmentation vs no augmentation Data augmentation (otherwise known as *data jittering*) [29] is a standard strategy employed for training neural networks when training examples are relatively scarce. The general idea is the application of a random transformation on each of the original data to produce new, artificial data. Typically a family of spatial transformations is employed to produce new data, with the choice of the family depending on the nature of the problem. In the work introducing PHOCNet [166], affine transformations have been used to produce warped word images. The affine parameters are constrained in order to have meaningful transformations considering text images.

Class balancing vs no class balancing In [166], a class balancing strategy is used to mitigate the negative effect of under-represented classes (meaning in this context, words) in the training data. This is originally combined with the affine-based augmentation scheme. In particular, under-represented classes are used to produce augmented examples much more often than over-represented ones, so that eventually all training classes are equal with respect the number of their respective data.

The class balancing scheme can be easily decoupled from augmentation procedure as follows. When choosing the next datum to be used with the net training algorithm, we select an image at random. Instead of considering uniform probability for this random selection, we adjust the probability distribution so that asymptotically all classes are equally represented during training.

The problem with this type of class balancing is that the model, due to its structure, does not learn the target classes directly. Instead, classes are only learned indirectly as a result of learning to estimate sets of attributes [5] (i.e. unigrams/bigrams per word image position). We therefore argue that class balancing, at least in the sense

previously described and employed in [166], is inadequate in terms of mitigating class under-representation.

6.2.3 Considered ensemble strategies

As solving for a convnet's optimal weights is treated as a local optimization task, different (random) initializations will in general lead to different local minima, and therefore different models. Combining a set of different models to obtain increased performance, known as *ensemble learning* is a strategy that has been employed for various pattern recognition tasks, including document image processing tasks [22, 21, 172]. The rationale behind ensembles is that different models will excel with respect to performance in different parts of the input space. Therefore, combining the models in some way, should lead to a better overall model. In the current work, we experiment with two different strategies for combining different trained convnets. The first strategy (*feature fusion*) consists of averaging over L convnet outputs given the same input word image. Concerning the second strategy (*late fusion*), we begin by computing L word spotting retrieval lists that correspond to the L convnets. Each retrieval list consists of the distance of the query example to the database instances, ordered in terms of increasing distance. Subsequently, we construct a single retrieval list, where we determine the position of each retrieved instance in the list by taking its minimum over the corresponding distances found on the L separate retrieval lists. We show that ensemble strategies, even when using a relatively low number of models L, can lead to significant accuracy improvement.

6.2.4 Experimental Evaluation

Experimental Setup

The keyword spotting experiments are evaluated on the challenging IAM dataset. Other datasets (for example, the George Washington dataset) may pose challenges on training due to their limited training samples, but the close to perfect existing results prove otherwise [167]. Moreover, the lack of generalization for networks trained on the George Washington dataset, as shown in the previous section, renders the case of exploring the aforementioned strategies on this specific dataset, a case of limited interest. We focus on QbE KWS scenario and therefore images in the testing set that correspond to stop words or appearing only once are excluded from the query set but are kept as distractors.

Given the outputs of the PHOCNet architecture for the images consisting the testing set, i.e. the PHOC estimations, the retrieval list is computed by nearest neighbor search using the cosine distance, which proved to be consistently well-performing at the previous section.

Training is performed assuming a binary logistic loss, as in [166]. Following [167], we use a batch size of 10 images. At this point, it is important to note that using batches of images of different sizes comes with the drawback of not fully utilizing GPU capabilities, as the batch cannot be described as a single 4-dimensional tensor. This results in slower training time compared to using same-sized images.

All networks are trained for $100,000$ iterations using Adam [97] with optimizer hyperparameters set to $\beta_1 = 0.9$, $\beta_2 = 0.99$ and weight decay $= 5 \cdot 10^{-5}$. An iteration corresponds to computing the gradients for a single batch and updating the weights accordingly. The initial learning rate is 10^{-4} and is divided by 10 after $60,000$ training iterations.

Training of a deep architecture will converge to, in general different, local optima that may differ slightly in terms of KWS performance. In order to better compare the architecture and training choices that we explore in this work, our networks are re-trained 5 times from different random initializations and their results are averaged.

Results & Discussion

Input Size and Spatial Pooling Strategy: First, we aim to explore the impact of both the input size and the spatial pooling strategy. We consider three different scenarios with respect to the input size: 1) initial image size 2) fixed height while keeping aspect ratio and 3) fixed-sized images. We use fixed sized images or even fixed height in order to speed-up training and inference, since resizing usually reduces the image size. The fixed height was selected as 50, whereas the fixed width was selected as 100. This choice was not further explored. We also evaluate two different spatial polling strategies: 1) 5-level Temporal Pyramidal Pooling (TPP): Uniformly segment the output of the last convolutional layer along the x-axis into i zones at perform max pooling at each zone. This is repeated for 5 pyramid levels ($i = 1, \ldots, 5$) from fine to coarse level. 2) Zoning: use only the finer level of TPP (5th level, i.e. 5 zones), discarding the pyramidal scheme.

We trained a network for each combination of the aforementioned scenarios and the KWS evaluation results are summarized on Figure 6.1. Two main observations are derived: *1)* Changing the aspect ratio of an image does not affect the result. The seemingly preferable choice of keeping the initial image size has no performance impact, even though it has greater training and inference time requirements. *2)* Pyramidal approach of spatial pooling has no advantage in performance over a simple zoning approach. However, TPP generates $\times 3$ more features than zoning to be fed to the first fully connected layer, which results to an excessive number of redundant parameters (over 20 million parameters in our case).

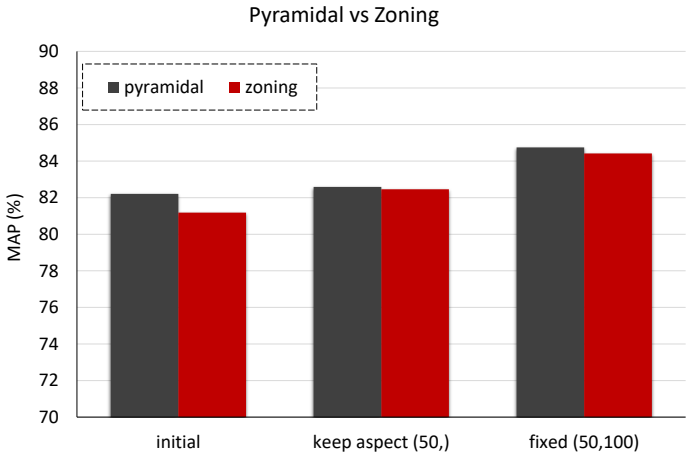


Figure 6.1: Performance of the trained models while different input size and spatial pooling strategies are considered.

These two observations can notably reduce the time of the training procedure. The impact on training time is presented in Table 6.4. It is evident that using fixed-sized images along with zoning drastically reduces training time.

For each spatial pooling scenario, we also report the evolution of MAP while training the network. These results are depicted in Figure 6.2 for both TPP and zoning scenarios. Both figures share a similar behavior on training convergence over the different input

approach	time (sec)
TPP, initial-size	0.295
TPP, fixed aspect ratio	0.176
TPP, fixed-size	0.064
Zoning, initial-size	0.274
Zoning, fixed aspect ratio	0.163
Zoning, fixed-size	0.051

Table 6.4: Training time per batch

size cases: resizing to a fixed-size provides a faster convergence compared to using the initial images. Furthermore, concerning convergence, we depict the MAP evolution for both spatial pooling strategies assuming fixed-sized images at Figure 6.3. The resulting evaluation curves are almost identical, supporting our claim that a pyramidal structure is redundant.

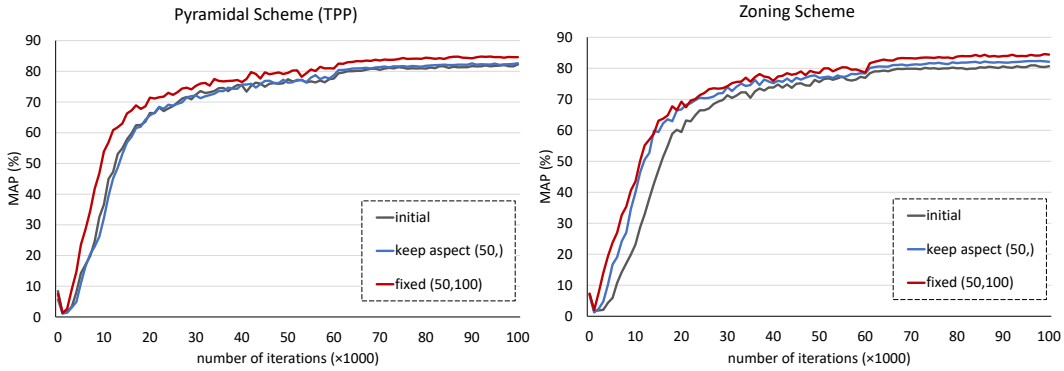


Figure 6.2: Evaluation curves of MAP performance during training under different input size

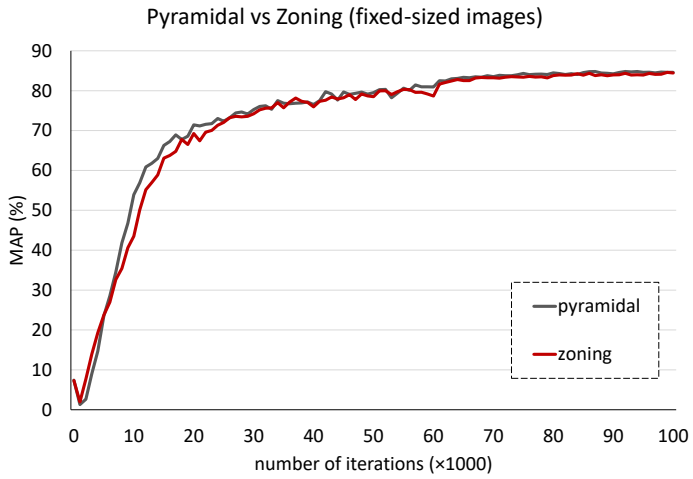


Figure 6.3: Comparison of evaluation curves during training for pyramidal and zoning schemes.

Up to this point, we have examined the effectiveness of the different spatial pooling schemes and have claimed that using fixed-size images does not affect the performance.

However, the spatial pooling layer was introduced in order to handle images of arbitrary size. Therefore, a follow-up question should be: is an adaptive spatial pooling layer necessary when the input images are of the same size? To answer this question we will compare the performance of a PHOCNet architecture with zoning layer against a PHOCNet architecture where the convolutional output is simply flattened. The second architecture is infeasible on our GPU due to memory limitations, since the output of the convolutional part has almost 150,000 dimensions. To overcome this problem we add a typical max pooling layer before the flattening operation. The aforementioned bare architecture gives 80.48% MAP, whereas using a zoning scheme gives 84.42% MAP. The difference in performance is significant, even though the bare network performs sufficiently well on this task. We speculate that this difference stems from the fact that using all the convolutional responses adds distractions, perhaps, unnecessarily enlarging the optimization search space. On the other hand, choosing a single response per filter and zone seems to suffice to describe the text structure, provided an adequate deep network.

Augmentation and Class-Balancing: Another important factor for training a deep CNN, such as PHOCNet, is the strategy of selecting and transforming the training images in order to be fed into the network. In the PHOCNet paper [166], there are two distinct choices regarding the training images: 1) data augmentation by (small) affine transformations and 2) class-balancing with respect to the unique word classes. We have trained models with or without utilizing these two strategies. For this experiment we assume zoning pooling at the output of convolution layers and fixed-sized input images. Figure 6.4 shows the resulting evaluation curves during the training process.

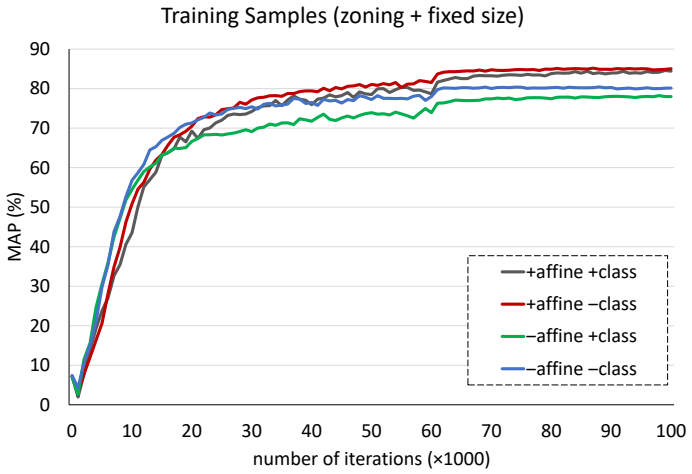


Figure 6.4: Evaluation curves under different training strategies with respect to affine augmentation and class balancing.

The results indicate that the PHOCNet architecture performs considerable well, even without any of these strategies. However, affine augmentation helps the network to generalize and produces a notable gain of 4% MAP. This was expected since small affine transformations appear frequently in text images are thus are a suitable for augmentation concerning KWS task. On the contrary, the class-balancing with respect to word classes has no impact when considering augmented images, while deteriorates the performance when no augmentation is involved. This behavior is seemingly surprising, but we should keep in mind that the network is trained with respect to PHOC attributes and not word classes.

Using Ensembles of Convnets: We train $L = 5$ diffent models of the same architecture starting from different initializations and use them in order to evaluate two different ensemble strategies: *feature fusion* and *late fusion*.

The evaluation of the ensemble variants are presented in Table 6.7, along with the evaluation of some baseline models from this work and the existing state-of-the-art approaches on IAM dataset. Baseline models *PHOC+bare-architecture* and *PHOC+zoning* are used as an indication of the capabilities of the PHOCNet architecture. Bare-architecture corresponds to discarding the spatial pooling layer and resembles the common CNN architecture. The *PHOC+zoning* is already on par with the best performing network so far [100] and is the network that we use for the ensemble approaches. The ensemble strategy utilizes the generated models and produces results that clearly outperform all state-of-the-art approaches. Even though both fusion techniques provide a significant boost in performance, late fusion performs better than feature fusion, as the former tries to select the best output out of the five while the latter combines the generated outputs into a single PHOC estimation.

Method	MAP(%)
PHOCNet [166]	72.51
Attribute SVM [9]	55.73
Krishnan et al. [100]	84.24
Wilkinson et al. [185]	81.58
Deep PHOCNet features [146]	81.50
PHOCNet-TPP [167]	83.38
PHOC + bare-architecture	80.48
PHOC + zoning	84.42
5-ensemble, feature fusion	86.94
5-ensemble, late fusion	**87.48**

Table 6.5: MAP Evaluation on IAM Dataset

6.3 Compressed Deep Features

Having explored adequately several aspect of a well-performing CNN designed for the task of KWS, we consider an alternative approach to build a KWS system based on deep learning [147]. Specifically, we aim to extract a deep, yet compact word descriptor, by appropriately training a modified VGG architecture (has similar traits with PHOCNet). The proposed descriptor is compact, in the sense that word instances can be represented as low-dimensional vectors following our technique; this result can be very advantageous in a large-scale KWS setting, where large document collections need to be efficiently indexed and subsequently searched.

Contrary to the majority of recent approaches, where each word is represented by a unique target class (class index or embedding), we assume that our target contains only the existence or not of unigrams and bigrams, resulting to a simple attribute-based representation. This approach cannot discriminate different words at target level, since many words contain the same characters, e.g. “dog” and “god”. To address this, the spatial information of each character should be added to the generated descriptor. The word descriptor is therefore extracted from the convolutional output of the final convolutional layer, which can be viewed as an intuitive discriminative feature map. Specifically, the

word descriptor is the concatenation of the max-pooled outputs after applying a zoning scheme on the convolutional output, i.e. splitting of the output into horizontal segments.

Following the aforementioned feature extraction procedure, the generated descriptor is dependent on the number of filters of the last convolutional layer, which is typically large; this shortcoming is addressed by the use of a compact descriptor for each zone. To the end of computing the proposed compact descriptor, we present a method that entails a novel, two-stage optimization scheme. According to the proposed optimization scheme, two separate neural network architectures are defined. This couple of neural networks, dubbed here the *extended* and *compact* network are trained sequentially, in the sense of using the former network's output as a pre-training initialization for the latter. We demonstrate that this scheme leads to superior results compared to obtaining deep features separately from either of the two networks.

Finally, we present a novel training strategy, which relies on the application of the fully connected layers on parts of the convolutional output and on the combination of different responses into a single output (the attribute-based target). The idea behind this approach is to train a model that can accurately detect unigrams (or bigrams) independently of the context, i.e. the word image, as it will explained in detail at the corresponding subsection.

We should note that the rationale behind this approach, is simple yet powerful: Neural networks are trained to optimize the respective loss. Let's consider the case of PHOCNet; the labels, i.e. the PHOC representation, demand that the network should distinguish between characters (i.e. "a" vs "b"), as well as between their respective position (i.e. second vs fourth letter). Without any extra constraint, the network does not learn to distinguish the letters and their position independently but it does it simultaneously. This is equivalent to introducing several filters which correspond to the PHOC attributes (i.e. "a" at the rightmost position). It is straightforward that such attributes require a high-dimensional feature representation due to their number. However, according to our approach, we restrain our network to detect specific characters and not their position. Their position is embedded in the final descriptor using the zoning technique.

6.3.1 Proposed Networks and Optimization Scheme

The proposed keyword spotting method assumes a set of annotated, word-level segmented images to be available for training. The case of interest is the QbE paradigm; the query, as well as the elements to be retrieved are segmented word images. Concerning the method itself, the backbone of the proposed processing pipeline is a pair of convolutional neural networks. These two networks are distinct, yet they are architecturally similar to a great extent. We shall dub the two networks, *extended network* and *compact network* and their differences will be described in detail at the upcoming sections. The architecture of both networks follows the template set by the PHOCNet model [166, 167]. However, contrary to PHOCNet and its variations, one of the main contributions of this work is to apply the fully connected layers on multiple max-pooled outputs from the convolutional part, rather than creating a unique, fixed-size, feature vector from the flattening operation and fed it to the fully connected part (such vectors contain spatial information which is being propagated to the fully connected part). This differentiation aims to simulate the existence or absence of characters, which is the target of our network.

The final word image descriptor is evaluated as a function of convolutional layer activations, following the paradigm of deep features [146]. As mentioned before, we use a word label that relies on encoding merely the existence of unigrams/bigrams, without directly taking into account their relative positions on the word. This is in contrast to what is done

by recent word embeddings ([9, 151, 185]). Given the aforementioned targets, we expect that the final convolutional layer will generate features closely related to the existence of the respective unigrams or bigrams. Concerning the testing procedure, we integrate the description with a spatial reference of attributes indirectly, by applying a zoning scheme. The final, fixed-sized feature vector is generated by a max-pooling operation over the segmented feature map into zones. An overview of the base neural network architecture can be examined in Figure 6.5.

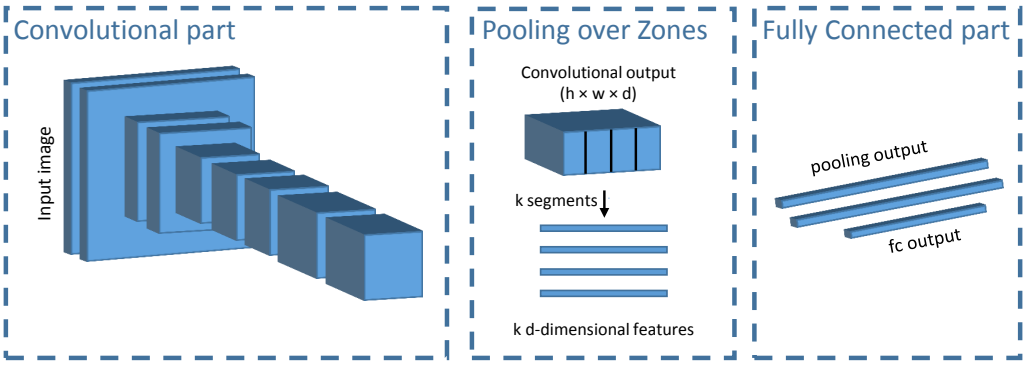


Figure 6.5: Neural network architecture overview. The base architecture, used for both employed neural networks in this work ("compact" and "extended") follows the template of a convolutional neural network comprising three componenents: a convolutional backbone (left), a pooling layer (center) and a fully-connected head (right). The pooling layer, applied on the last convolutional layer, is used as a deep feature extractor.

6.3.2 Word labels

According to the PHOCNet architecture, the network should learn the characters' representations (i.e. if a specific character exists in the image) along with their relative position (e.g. the character is in the first half of the image). This is imposed by the PHOC embedding, which encodes characters and their relative position. The PHOC formulation of relative positions requires filter responses that distinguish characters as well as their position and therefore a great number of such (convolutional) filters are required.

One straightforward simplification is to train the network in order to decide upon the existence or not of the possible characters. The word label for this approach would be a binary embedding (denoting the existence or not) of possible unigrams (lowercase letters and digits: 36 unigrams in total). An extension to the aforementioned embedding is to also include all possible bigrams to the word target (36 unigrams + 36 × 36 bigrams). Hence, the length of the last, sigmoid-activated layer is $36 \times 37 = 1337$ neurons long. This representation is expected to be sparse, since several bigrams do not exist for the English language (or the selected script language). The reasoning of such an alternative representation is that usually isolated characters differ visually when written alongside specific characters.

Both representations are useful for training filters that correspond to specific characters (or couples of characters). However, they cannot constitute a descriptor that could be used for effective word comparison, which would in turn be necessary for a KWS system. This holds especially when taking into account a description comprising only unigram attributes with no spatial reference. To this end, we use the response of the final convolutional layer as a deep feature. This will ideally encode non-spatial, character attribute information.

Concerning dealing with encoding spatial information, we integrate the obtained feature with a zoning scheme.

6.3.3 Network Architecture

We adopt a network consisting of three main components, namely:

- **Convolutional Part:** 2 convolutional layers $\rightarrow$ 2×2 max-pooling layer $\rightarrow$ 2 convolutional layers $\rightarrow$ 2×2 max-pooling layer $\rightarrow$ 9 convolutional layers.
- **Pooling/Flattening Part:** Use of a Temporal Pyramidal Pooling (TPP) [167] scheme on the convolutional output, without concatenating the resulting max-pooled outputs. This is explained in detail in the following subsections.
- **Fully Connected Part:** 3 fully connected layers resulting to a vector whose size is equal to the word's representation attributes (36 or 1337).

The size of all convolutional filters is set to 3×3. All non-pooling layers are topped by ReLU non-linearities, except for the output layer. The network is topped by sigmoid activation functions, with each layer variate representing a word image attribute class.

Each level l of the previous convolutional layer output is segmented into l horizontal zones. As a result, for n_l levels and a convolutional output of depth d, we get a set of $N = \sum_{l=1,\ldots,n_l} l = n_l(n_l + 1)/2$ different d-sized vectors. Compared to the standard Temporal Pyramidal Pooling (TPP) layer, an important tweak at this point is that we do not concatenate the N vectors into a single vector. Instead, each one of the N max-pooled, d-sized outputs is fed to the fully connected stack of layers separately.

The N vectors that are produced as the output of the modified TPP layer are fed to the fully connected layer and N output vectors are generated. On the output layer, the variates of each of these N sigmoid-activated vectors correspond to the word attributes previously described in subsection 6.3.2. The outputs of each zone segment, after the fully connected part, corresponding to a specific pyramid level should be combined in order to produce a valid attribute representation, since each zone contains a different set of unigrams/bigrams. Therefore, they can be combined using a union operation of the separate attribute representations, i.e. an element-wise max operation. Each pyramid level, after the aforementioned max-pooling, should provide a valid attribute representation of the word and under this assumption we employ an element-wise min operation over the different attribute representations, which relates to optimizing the precision of the representations (in other words, the min operation assists the correction of true negatives over the pyramid levels). Figure 6.6 visually depicts the training strategy, consisting of union/intersection operations, that we described above.

6.3.4 Extended and Compact Network

The two networks, extended and compact, differ in the layout of exactly one layer, which however makes for a considerable difference in practice. The layer that differs is the last convolutional layer, i.e. the layer that is used as input to the pyramidal pooling layer. As mentioned before, the word image representation is linear to the depth of the convolutional output which means that reducing the number of filters of the last convolutional layer results to compact image representations. Nevertheless, such reduction may significantly affect the network performance.

Our goal is to examine if we can accomplish acceptable performance on the QbE KWS scenario even if we significantly reduce the depth of the convolutional output, i.e. the

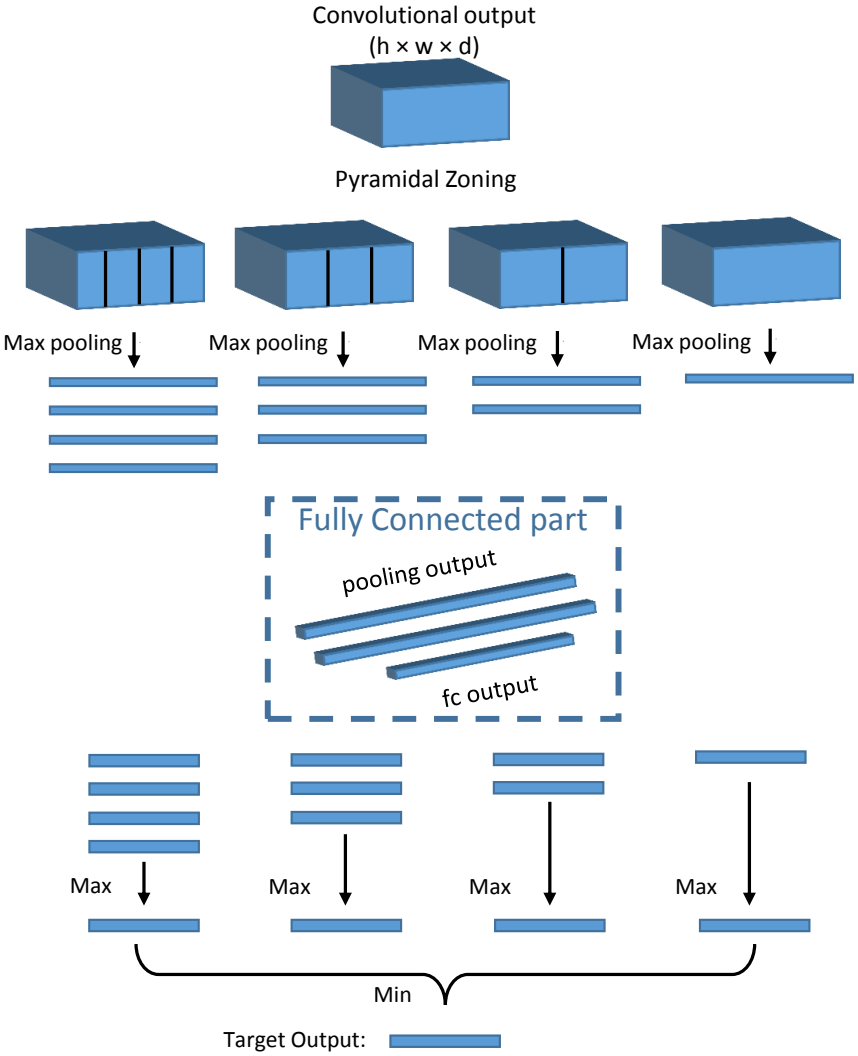


Figure 6.6: Proposed training strategy, which uses a modified TPP layer and generates multiple outputs for each image. Convolutional outputs are first pooled to N feature vectors (top, in this figure $N = 4 + 3 + 2 + 1 = 10$) using temporal max pooling. These are transformed to N attribute detector outputs (bottom), which are then combined with max and min operations to form a single attribute detector output.

number of filters. A compact representation is supported by the idea of learning filters that only detect characters and not their spatial information, since much less filter depth is required for this goal. It should be noted that the compact architecture not only provides compact feature representations, but also results to a significant reduction of the inference time.

6.3.5 Optimization scheme

An important observation is that the training of the compact model, assuming random initialization, converges to suboptimal solutions. This is not surprising, since the reduced last convolutional layer operates as bottleneck to the entire architecture and it is highly probable to have a unique solution with respect to its parameters. Such a unique solution is difficult to approximate while using a stochastic gradient approach to simultaneously optimize the whole set of parameters. Nevertheless it is safe to assume that the extended and the compact network are closely related regarding their weights, since the task remains the same. Compared to the compact network, the extended network can be trained effectively without a problem from randomly initialized weights.

Therefore, for training, the networks are used in the following manner. We use the available word images to first train the extended network. After the training of the extended network finishes, we keep all of the convolutional weights except for those of the last layer, and re-use them as pretraining weights for the compact network. Note that this is possible since the two networks share very similar weights with regard to all convolutional layers, save for the last one.

In detail, we apply a two-step training procedure, which ensures a fast convergence of the compact network:

1. **Initialization:** Use the pre-trained model of the extended network to initialize the convolutional layers except the last one, since it has a different number of filters. Train only the remaining layers/parameters, i.e. the last convolutional layer and the fully connected layers, assuming the other convolutional layers fixed. This approach will converge fast to a valid solution, which corresponds to a decent performing model.

2. **Fine-tuning:** Re-train the whole network, using the initialization of the first step. All convolutional layers are also trained in this step in order to better adapt to the problem.

The intuition behind the reason the proposed scheme works is summarized as follows. The extended network is easier to optimize, in the sense of reaching lower loss and higher accuracy figures comparatively fast. At the same time, the compact network is harder to optimize. Since the architectural difference between compact and extended network is the depth of the last convolutional layer, we can conclude that this feature forms an apparent optimization bottleneck. However, having two similar models comes with the advantage that the weight search space of the two networks is "similar". This is said in the sense that the compact and extended network differ only to a minor percent of weights, and practically all their convolutional backbone is identical. In practice, we have validated that the compact network trains better once a good initialization is used; this is provided by the (comparatively) easy-to-train extended network.

6.3.6 Experimental Evaluation

Similarly to the previous section, we evaluate our method using the IAM dataset, while focusing on QbE KWS scenario. Concerning our method, we assumed 5 levels of the modified TPP layer and 5 zones on the extraction of the deep descriptor. In Table 6.16, the numerical results for both the extended and the compact network are presented. There are three main observations:

- The attributed-based output used in this work is not capable of successfully discriminating words, as expected. On the contrary, the zoning scheme achieves very good results.
- Including the bigram information on the target vector provides a notable boost in performance.
- Compact network retains a good performance, even though it generates $32\times$ smaller descriptors. For the rest of the evaluation section, we consider unigrams+bigrams representation and the zoning scheme as the default parameters for our method.

It should be noted that if we train the compact network with randomly initialized weights, the model seems to get stuck to a local optima which performs at least 5% lower compared to using the proposed optimization scheme.

Approaches	Unigrams	Unigrams+Bigrams
Extended		
Zoning ($5 \times 512\,d$)	80.79	84.68
Output	60.43	73.77
Compact		
Zoning ($5 \times 16\,d$)	74.72	81.65
Output	52.06	56.95

Table 6.6: MAP (%) performance comparison of the extended and the compact network.

Table 6.7 reports the numerical results for the proposed method compared to several state-of-the-art methods. Both networks, extended and compact, achieve notable results that are on par with state-of-the-art results (extended network outperforms the reported methods). Concerning the compact descriptor, which is our case of interest, not only it achieves state-of-the-art results but also it has other merits as well. First, even though the proposed optimization scheme is a two-step process, integrating the training of two separate networks, word descriptor evaluation can be performed by a simple feed-forward on the compact network only. Hence, descriptor evaluation is comparably fast. Second, the resulting word image descriptor is compact, represented by a 80-dimension vector, which is almost 10 times smaller than the PHOC representation of [166]. This can be seen as a considerable advantage, especially in a large-scale spotting context.

Finally, the network formulation and training scheme that we presented enables the detection of characters on the image, which is not explored quantitative in the context of KWS. However, it is an advantage of our approach and could provide a cost-effective solution for segmentation-free OCR on document images. A visualization of the aforementioned idea is depicted on Figure 6.7, where the fully connected layers are used as convolutional layers with 1×1 filters and the initial images is transformed to a response map of the existing unigrams.

Method	MAP(%)
PHOCNet [166]	72.51
Attribute SVM [9]	55.73
Krishnan et al. [100]	84.24
Wilkinson et al. [185]	81.58
Deep PHOCNet features [146]	81.50
PHOCNet-TPP [167]	83.38
Extended Network	84.68
Compact Network	81.65

Table 6.7: MAP comparison on IAM Dataset using the extended and the compact networks trained on the unigrams+bigrams attribute representation.

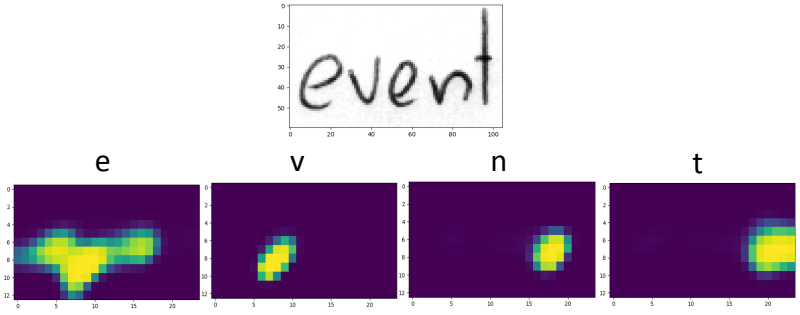


Figure 6.7: Response output maps of the word image "event", after the use of the fully connected part as convolutional with 1×1 filters. The only characters that have high confidence (> 0.8) on the output are the existing ones, "e", "v", "n" and "t". Only the responses of these unigram filters are depicted and we can clearly observe the spatial information about each unigram.

6.4 WSRNet: Joint Spotting and Recognition

Handwritten Text Recognition (HTR) and Keyword Spotting (KWS) are two tasks of central importance in the literature of document image processing, as we have already explained at Chapter 2. The two problems are closely connected, yet they are faced with different families of techniques. However, in this section, we present *Word Spotting and Recognition Network (WSRNet)*, a unified model that can tackle both handwriting recognition and keyword spotting, using the same neural network architecture. The proposed architecture fuses two components that are prevalent in Recurrent Neural Network-based methods, namely the Connectionist Temporal Classification (CTC)[63]-based training and the sequence-to-sequence (Seq2Seq) paradigm. The two components are combined into a single architecture through a suitable multi-task loss, where we show that both components are necessary for optimally efficient model training. Full word recognition is possible by either of the CTC or the Seq2Seq paths, while both the CTC and Seq2Seq network branches are used to backpropagate losses during training. Note that Seq2Seq architectures have been introduced to the HTR field of research very recently [168]. The option of keyword spotting is enabled by taking advantage of the Seq2Seq intermediate fixed sized encoding as a feature vector. Then KWS can be performed either by Example (QbE) simply by comparing feature vectors or by String (QbS) by employing an extra encoder module that translates query strings to the Seq2Seq intermediate representation space, or by forced

aligning the query to the decoder. Furthermore, we show that the Seq2Seq-based representation can be refined by binarizing it with an efficient straight-through estimator-based (STE) retraining scheme [18], hence producing a compact feature vector.

6.4.1 Proposed Architecture for Word Recognition

In this section, we describe in detail the proposed architecture for word recognition and spotting, first with a focus on the former application. Our architecture comprises two basic components, namely a CTC-based branch and a Seq2Seq branch. The CTC pipeline is inspired by typical CTC-based HTR systems applied in text-line level, as in [49] Contrary to these systems, which consist of a CNN for feature extraction topped by a RNN (usually LSTM), we replace the RNN module with an 1-D CNN. This architectural choice aims to simplify the pipeline and assist the training procedure, as we will explain in detail in the following sections. Over the CTC pipeline we added a Seq2Seq module [169], i.e. a Encoder/Decoder Recurrent network pair, which takes as input the output of the aforementioned CNN module, encodes the word information into a fixed-sized vector and consequently decodes it into a sequence of characters. In what follows, we describe these modules and their functionality in detail.

Convolutional backbone

The visual feature extraction task is performed by a CNN, dubbed as backbone network, which consists of multiple residual blocks along with ReLU nonlinearities. To assist network convergence, we add batch normalization (BN) before nonlinearities, and use dropout at each layer in order to improve generalization [17]. The feature map downsampling between groups of consecutive residual blocks is performed by max pooling. The CNN architecture is summarized on Table 6.8.

layer	building block
conv1	7×7, 64, stride 2
conv2_x	$\begin{bmatrix} 3 \times 3, & 64 \\ 3 \times 3, & 64 \end{bmatrix} \times 2$
maxpool_1	2×2, stride 2
conv3_x	$\begin{bmatrix} 3 \times 3, & 128 \\ 3 \times 3, & 128 \end{bmatrix} \times 4$
maxpool_2	2×2, stride 2
conv4_x	$\begin{bmatrix} 3 \times 3, & 256 \\ 3 \times 3, & 256 \end{bmatrix} \times 4$

Table 6.8: CNN backbone architecture: Convolutional stacks of residual blocks transform the input word image to a feature map, subsequently passed to the CTC Convolutional head.

CTC component

Typically, CTC-based approaches in the literature are using LSTMs on top of the feature map, extracted by the CNN backbone. In this work, contrary to the majority of existing

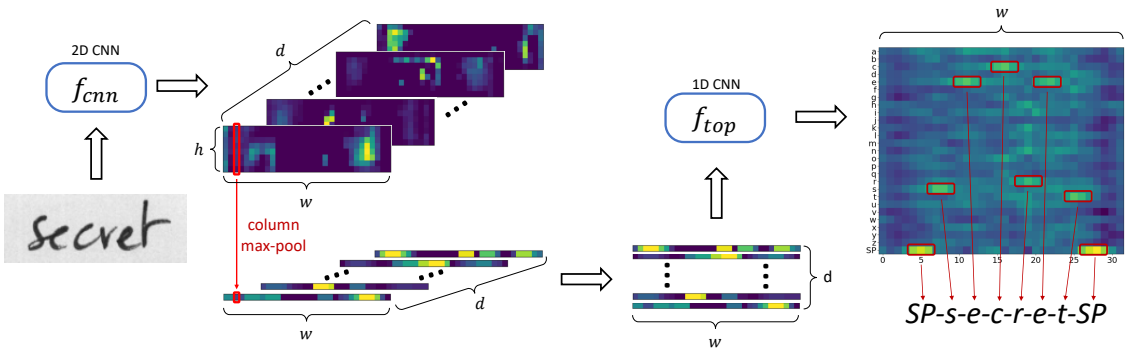


Figure 6.8: Proposed architecture CTC branch. The word image is processed through a 2D convolutional backbone, and fed to 1D convolutional head and CTC output layer. Omitting the "traditional" Recurrent layer before the CTC output results in easier model training.

CTC-based approaches, the last layers (CTC head) are not recurrent. Instead, we use a batch of three 1-D convolutional layers with kernel size equal to 7, along with BN, ReLU and dropout. Note that multiple 1-D convolutional layers are capable of encoding context-wise information, which is the major goal of recurrent networks. The gain of replacing recurrent networks with 1-D CNNs is two-fold. First, LSTMs are known to display convergence difficulties, while 1-D convolutions with BN can converge very fast. Second, 1-D convolutions can be fully parallelized and thus considerably improve training and inference time, as opposed to LSTMs.

To further promote simplicity we transform the output of the CNN backbone of size $h \times w \times d$ into a feature sequence of size $w \times d$ by column-wise max-pooling. The reasoning behind max-pooling is that we care only about the existence of features related to a character and not their spatial position. Note that if column-wise concatenation was performed, instead of max-pooling, the feature sequence would be of size $w \times hd$.

The CNN backbone topped by the CTC head is depicted in Figure 6.8. The output of the 1-D CNN is of size $w \times n_{classes}$, where $n_{classes}$ is the number of possible character classes. Applying the softmax function on the final output, we form a sequence of probability distribution over the possible characters which is then propagated into the CTC loss L_{CTC}. Given the trained system and an input image, a sequence of character probabilities is generated and it can be eventually transformed into the recognized character sequence by a CTC decoding procedure [62].

Sequence-to-Sequence Component

The second branch of the proposed network architecture involves a sequence-to-sequence component [169]. The main components of such a system are two recurrent neural networks, the *encoder*, which projects a sequence into a fixed-sized vector, and the *decoder*, which is responsible for decoding the encoded fixed-sized feature vector into the target sequence. The encoder network generates the fixed-sized vector by extracting the last hidden vector of the recurrent operation as a holistic representation. The decoder network, given a hidden vector and the previous element of the sequence, predicts the next element. Concerning the problem of handwriting recognition, the input sequence is the sequence of visual features, generated by the backbone CNN, as described at the previous section. The output sequence is, as expected, the target sequence of characters. Therefore, given

a input character and an intermediate feature representation (hidden vector), the decoder system should predict the next character. For our system, both the starting and ending tokens (starting and ending a word, respectively) are selected to be the same space token (SP), which naturally separates words.

The proposed Seq2Seq branch is visualized at Figure 6.9, for which our architectural choices are as follows:

Encoder: bi-directional GRU [33] of 3 layers with 256 hidden size. The $2 \cdot 3 \cdot 256$ dimensional output vector is then compressed into 512 dimensions by a linear transformation, which is the final output of the encoder and, consequently, the input of the decoder.

Decoder: one-directional GRU of 1 layer and 512 hidden size. The output at each step is transformed by a linear layer into $n_{classes}$ size in order to predict the next character. Let g_d be the GRU cell of the decoder module and $s = c_0 c_1 \cdots c_{K-1}$ the target string. Note that, according to our formulation, $c_0 = c_{K-1} = \text{SP}$, where SP stands for the "blank" token. Also, let x_{enc} the output of the encoder module. Each decoding step can be written as:

$$\begin{aligned} c_0 = \text{SP} \quad \& \quad h_0 = x_{enc} \\ c'_i,\, h_i = g_d(c_{i-1}, h_{i-1}) \end{aligned} \tag{6.1}$$

Concerning the use of attention [133], we have chosen against using it with our Seq2Seq component. State-of-the-art Seq2Seq models do use an attention module [133], which directly propagates information from the input sequence to the output sequence. However, such an assisting module would result in decreasing the significance of the intermediate vector between the encoder and the decoder. Specifically, for attention-based approaches, the character encoding information is mostly propagated through the attention module while the intermediate feature vector usually assists the spatial correspondence of the attention module. As we mentioned in the introduction, one of our main goals it to generate unique word representations. The intermediate feature vector is ideal for this task and thus adding an attention path would decrease the ability of generating discriminative representations.

The Seq2Seq component shares the same backbone with the CTC-based component. In particular, this is the convolutional path from the input word image up to and including the max-pooling result to a feature map of size $w \times d$ (see Figures 6.11,6.9). The full model is then trained with a multi-task loss, defined as a weighted sum of the loss for the two branches:

$$\begin{aligned} L(w_{cnn}, w_{top}, w_{S2S}) = \\ L_{CTC}(w_{cnn}, w_{top}) + \lambda L_{S2S}(w_{cnn}, w_{S2S}), \end{aligned} \tag{6.2}$$

where L_{CTC} and L_{S2S} are the CTC and the Seq2Seq loss functions respectively and hyper-parameter λ controls the contribution of each loss. L_{S2S} loss is the average cross entropy loss across all the per-character predictions produced by the decoder. The parameters of the backbone CNN w_{cnn} are jointly trained by both losses, while the parameters of the 1-D CNN w_{top} and the parameters of the Seq2Seq modules w_{S2S} are optimized by their corresponding losses.

In inference mode, this network branch offers an alternative to the CTC for decoding the input, by simply computing the Seq2Seq output sequence. In practice, joint training offers an improved decoding for either of the two inference options, compared to having trained the two branches separately.

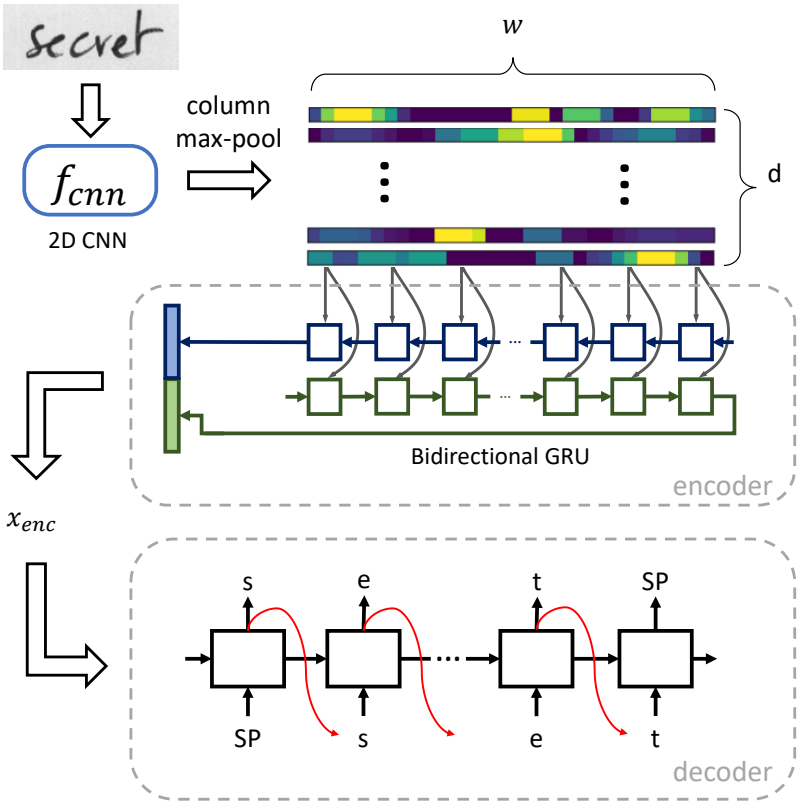


Figure 6.9: Proposed architecture Seq2Seq branch. The backbone is shared with the CTC branch (Fig. 6.8). The Seq2Seq branch is combined with the rest of the network through a multi-task loss.

6.4.2 Keyword spotting using the Seq2Seq encoding

Apart from evaluating the Seq2Seq approach on HTR, in this section we show that the proposed architecture can also be used to tackle either QbE or QbS keyword spotting. Furthermore, we explore further refining the resulting holistic word representation by binarizing and thus significantly compressing it. Regarding QbS KWS, we present two alternative schemes, one involving adding an autoencoder module to the main network, and the other involving using the technique of forced alignment on the Seq2Seq branch.

Query-by-Example with the encoder output as a word image representation

The idea behind the proposed word spotting extension is to utilize the architecture to generate descriptive holistic representations for each word image and the query. The intermediate feature vector of the Seq2Seq system, generated between the encoder and the decoder module, is ideal for this task, as it readily produces a fixed-sized descriptive word descriptor. Query-by-Example KWS is then straightforward, as it suffices to compare descriptors with a suitable distance measure.It is also important to note that the generated descriptor can be fully translated into the target character sequence through the decoder module, as opposed to attribute-based word representations, such as Pyramidal Histogram of Characters (PHOC).

Query-by-String with an Autoencoder Module

The existing system can straightforwardly perform QbE spotting by comparing the generated intermediate feature vectors. Nevertheless, the QbS variation cannot be executed with the current formulation. To this end, we add an extra encoder module which encodes the target sequence (i.e. in practice, the string query) into a feature vector which can be decoded with the existing decoder into the same sequence, effectively forming an autoencoder path. Therefore, with this autoencoder we construct a one-to-one function between character sequences and intermediate fixed-sized feature vectors.

The problem that arises from the addition of the extra character encoder module is the simultaneous training along with the visual encoder of the Seq2Seq component. If we constrain the output of the character encoder to be similar to the visual encoder output, e.g. using MSE loss, there is no guarantee that both encoding can be decoded successfully, since even a small euclidean divergence between the encoding may result two different decodings. We overcome this problem by randomly choosing to decode one of the two information flows at each iteration, while constraining them to be close to each other. More formally, let f_{cnn} be the backbone CNN, f_{enc}/f_{dec} the encoder/decoder of Seq2Seq and f_{cenc} the character encoder. Given an image I and its corresponding groundtruth string $s = c_0c_1 \dots c_K$, the two encodings can be written as follows:

$$x_{enc} = f_{enc}(f_{cnn}(I; w_{cnn}); w_{enc}) \tag{6.3}$$

$$x_{cenc} = f_{cenc}(s; w_{cenc}) \tag{6.4}$$

The loss for the Seq2Seq branch is then extended as:

$$L_{S2S}(s_{pred}, s; w_{cnn}, w_{S2S}) + d_{cos}(x_{enc}, x_{cenc}) \tag{6.5}$$

where $d_{cos}(\cdot, \cdot)$ stands for the cosine distance, and the predicted string s_{pred} is the decoding of either x_{enc} or x_{cenc}, randomly selected, as follows:

$$b \sim Bernoulli(0.5) \tag{6.6}$$

$$s_{pred} = f_{dec}(bx_{enc} + (1 - b)x_{cenc}; w_{dec}) \tag{6.7}$$

The aforementioned training scheme is summarized at Figure 6.10.

During model training, this extended loss (Eq. 6.5) replaces the corresponding term in the full model multi-task loss (Eq. 6.2). Aside from enabling QbS, the addition of the autoencoder module is beneficial implicitly also for the QbE scenario, as it effectively results in constraining representations of different word images depicting the same word around a unique representation.

An additional interesting consequence of adding the autoencoder module is that an *external* lexicon (*external* in the sense that it contains word transcripts that are not necessarily part of the training set, nor its words are paired with ground-truth images) can be used to aid training. So far, we took into account only the words exist in the training set, even though Seq2Seq system is capable of learning an implicit language model of valid consecutive characters. The intuition behind this variation is to assist the underlying implicit language model of the Seq2Seq system, by feeding it with valid words that may not exist in the training set. Implementation-wise, at each optimization iteration and after updating weights by backpropagating the multi-task loss w.r.t. the standard training set, we fine-tune the autoencoder path with words drawn from a lexicon.

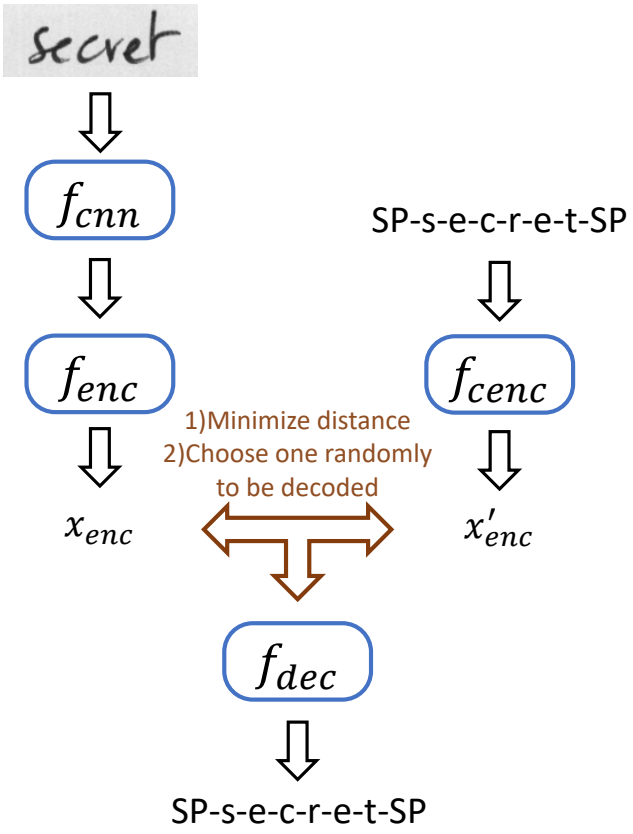


Figure 6.10: Autoencoder module that extends on the Seq2Seq component. By learning a mapping from a character sequence to the Seq2Seq intermediate representation, Query-by-String KWS is effectively enabled.

Query-by-String with Forced Alignment

Even though holistic representations of fixed size greatly simplify the upcoming matching step, we also consider the case of using the decoder model of the proposed Seq2Seq system as a scoring function for the representation x_{enc} constrained to produce a target word string s. This constrained decoding, usually referred to as forced alignment, is a popular QbS alternative based on character lattices [173], where a specific sequence of character is scored according to a pre-computed character graph (consisted of the probability of the character at each node, as well as the transition probability from one node to another). The alignment term refers to the case of several possible alignments of the sequence of features to the desired sequence of characters. However, following the decoder definition, the query-constrained scoring can be efficiently performed as the following equations suggest:

$$\begin{aligned} c_0 = \text{SP} \quad \& \quad h_0 = x_{enc} \\ c'_i,\, h_i = g_d(c_{i-1}, h_{i-1}), \quad i = 1, \ldots, K-1 \\ \text{score} = \sum_{j=1}^{K-1} L_{CE}(c'_j, c_j) \end{aligned} \tag{6.8}$$

Specifically, we assume that the input of the decoder is the requested query $s = c_0 c_1 \ldots c_{K-1}$ and thus it is straightforward to predict the next character, given the previous one and

the hidden vector computed this far. Consequently, the score is the average cross entropy loss of the predictions and thus if the score is low, the given query was in line with the word representation x_{enc}.

Implementation-wise, performing a forced alignment as described above is time-consuming and cannot be parallelized in order to fully utilize accelerators, e.g. GPUs. We can however overcome the computational cost by organizing multiple queries into a character trie: given a single intermediate feature, we decode over the character trie in a breadth-first traversal manner, making use of multiple nodes for a parallelized, fast implementation on the GPU.

Binary Word Representation

One noteworthy variation is the binarization of the intermediate feature vector, which can greatly reduce the storage requirements for storing large collections of documents. For example, if we have a collection comprised by 1,000 words (assuming perfect word segmentation), the proposed representation has a dimensionality of 512 with floating-point values, which amounts to 512×4 bytes and overall almost 2.5 GB for the whole collection. On the other hand, by assuming a binary 512-d representation, the overall storage requirements are only 64MB, namely $32\times$ less storage. The binarization of the word representation can be straightforwardly performed by a *sign* operation and we expect to generate binary descriptors that do not affect significantly the performance of KWS task, since the already used cosine distance would also work well under the signed vector, as the experimental results validate. Note that all the signed vectors have the same magnitude and therefore, since we care about comparing an image representation with a reference one, the cosine similarity can efficiently computed by XNOR and bincount operations. Nevertheless, such a binarization approach may significantly affect the decoding step and thus the decoding property, one of the crucial contributions of this work, is lost. To address this problem we proposed a training scheme for binarized vectors based on the Straight-Through Estimator [18].

The Straight-Through Estimator was initially motivated by the inability to train networks which include threshold operations, such as the Heaviside or the sign function. Such discontinuous functions are not differentiable and thus unsuitable for backpropagation. To overcome this problematic behavior of threshold operations, it was proposed [18] to assume that these hard thresholding functions, which are used for the forward pass, let the error to be propagated without any change through at the backward pass (i.e. straight through estimation), as if the function was $f(x) = x$ and consequently had an identity gradient.

Although STE naturally is essential for training binary CNNs [139], it is rather crude to be applied straightforwardly on the generated representations (we have tried such an approach, and the system could not converge). One main difference is that STE was applied on the network weights that are steadily updated, while we apply STE to a feature vector with considerable variations for each image/word at the same iteration. This problem was resolved by simulating a *tanh* activation based on the following observation:

$$x_{bin} = \text{sign}(x_{enc}) = \lim_{a \to \infty} \tanh(a x_{enc}) \tag{6.9}$$

Therefore, we distinguish two cases: In the forward pass, we simply use the binarized vector $\text{sign}(x_{enc})$ as input to the decoder; in the backward pass, we treat the signed vector as if a *tanh* operation was applied, namely $tanh(x_{enc})$, and compute the gradients and backpropagation error accordingly. Following the STE reasoning, we do not care about

defining the a value of Eq. 6.9, as long as we distinguish one hard operation at forward pass and a corresponding soft differentiable one for the backward.

6.4.3 Experimental Evaluation

In this section, we present experimental results across all HTR and KWS variations discussed so far. We have run numerical trials on the IAM dataset. As IAM is a large and multi-writer dataset, it is typically used as the standard benchmark of comparison for either handwriting recognition or keyword spotting methods alike.

In order to train our network, we have used the ADAM optimizer [163] (initial learning rate: 0.01 / number of epochs: 80/ cosine annealing scheduler). We have augmented the training set by performing random global affine transformations on available word images. Regarding image preprocessing, we apply zero padding of all inputs to a fixed size without affecting the aspect ratio (64×256). This simple preprocessing enables batch processing, which leads to faster and more efficient training. Concerning the multi-task loss weight λ in Eq. 6.2, we found that applying a larger weight to the Seq2Seq branch ($\lambda = 10$) was beneficial, as the CTC branch can be trained more easily, due to being completely non-recurrent. Regarding training of the Seq2Seq in particular, we employ the *teacher forcing scheme* [16]. Specifically, we randomly select the decoder input at step i to be either the predicted character of the previous step $i-1$ or the real character c_{i-1}. In this manner we avoid frequent error propagation from a miss-predicted character during step-by-step decoding.

First, we explore the word recognition task with the full set of existing characters, i.e. lowercase/uppercase letters, digits and punctuation, following the line-level recognition setting. The results are summarized at Table 6.9, considering both the CTC and the Seq2Seq approach. We distinguish the training scheme to *alone*, where each pipeline is trained separately, and *joint*, where both recognition flows are trained together as a multi-task problem (see Eq. 6.2). All the results are based on a greedy decoding procedure. Experimental results suggest that the Seq2Seq approach has convergence difficulties when trained alone. Nevertheless, when used alongside the CTC, Seq2Seq has a notable increase in performance for the exact same number of epochs. This is in line with our initial suggestion of a fast converging CTC pipeline, which helps the generation of meaningful visual features at the CNN output in only a few epochs. Moreover, we should highlight that Seq2Seq variation gives the best WER, while assists the CTC flow to have a minor improvement at WER too. At Table 6.9 we also report results for a state-of-the-art line-

setting	method	CER	WER
line	CTC [136]	6.2	20.2
word (alone)	CTC	6.6	17.4
	Seq2Seq	10.7	21.6
word (joint)	CTC	6.7	17.2
	Seq2Seq	6.9	16.5

Table 6.9: Recognition results using the proposed method (setting "word"), comparing use of the two branches (CTC or Seq2Seq). Results are also compared w.r.t training each branch separately ("alone") versus training w.r.t. the proposed loss. CTC-based results on line-level recognition are also reported for comparison.

level method (CNN+LSTM) with otherwise the same trial setting (training/test set and

character set) [136]. Line-level segmented images are benefitted with more context compared to word-level segmentations, which can be very helpful when employing an LSTM network. However, word-level recognition relies on a perfect word segmentation, which is ideal, and thus avoids errors by misaligned words. Even though trade-off is reflected by the results, we can conclude that the proposed architectures have close performance to a typical state-of-the-art line-level recognition system.

To further explore the capabilities of the proposed recognition system, we report the results of a more complex decoding scheme, the beam search algorithm, as shown at Table 6.10. Beam search over the CTC output is fairly common, especially for incorporating external Language Models (LMs). Note that no LM is used for the results. The beam search algorithm can be easily modified for the Seq2Seq variation and specifically the decoder module. Instead of having a unique decoding path, where we predict the next character, we have a set of beams, each one with a different ongoing hidden vector and a history of previous characters. A new beam is created by adding the top-k predicted characters at each step. The beams are sorted according to the summed probability of the added characters and pruned to a predefined beam width, in order to avoid exploding complexity. The results (we use jointly trained outputs and beam width = 5) indicate a similar consistent error decrease for recognition with either of the model branches (CTC, Seq2Seq) when using the beam search algorithm.

decoding	method	CER	WER
greedy	CTC	6.7	17.2
	Seq2Seq	6.9	16.5
beam search	CTC	6.6	16.9
	Seq2Seq	6.8	16.2

Table 6.10: Comparison of decoding schemes for word recognition. Beam search is compared to a greedy decoding scheme.

Having established the functionality of the recognition system, we focus on exploring the proposed variations for the keyword spotting task. Spotting approaches usually use a different character set, consisted only of lowercase letters and digits, and thus we follow the same setting, both for word spotting as well as the reported recognition results from now on. Table 6.11 contains the experimental results for both QbE and QbS, along with the recognition metrics for the decoder. The system under consideration comprises both the CTC and Seq2Seq branches, as well as the autoencoder module. Recognition is performed using the Seq2Seq branch. This system is dubbed as *WSRNet*, while the variation of adding an external Lexicon during training in order to learn an implicit Language Model is referred to as *WSRNet+LM*. Our lexicon consists of a concatenation of the LOB [94] and BROWN [51] corpora, from which we sample words according to their occurrence frequency. We also evaluate two distinct training schemes: we compare training from scratch the whole network versus using the previously trained modules for the recognition task as initialization then fine-tuning the autoencoder. The experimental results indicate that the external lexicon improves the decoding process by a considerable margin, while having a positive impact on KWS as well (especially for the case of training from scratch). Moreover, the initialization of the majority of the system weights with the previously trained recognition task leads to a noteworthy boost.

Next, we present trials concerning the binarization strategy of the proposed word

method	QbE	QbS	CER	WER
from scratch				
WSRNet	90.20	92.07	6.5	17.2
WSRNet+LM	90.87	93.16	5.9	15.8
fine-tuned				
WSRNet	92.09	95.23	5.1	14.1
WSRNet+LM	92.05	95.67	5.0	13.9

Table 6.11: Comparison of recognition and KWS results for incorporating a language model. Recognition on the Seq2Seq is considered, and QbS is performed by means of the proposed autoencoder module.

representation over the fine-tuned *WSRNet+LM* variation. Table 6.12 summarizes the effect of the binarization. Perhaps unsurprisingly, applying the *sign* operator on the intermediate feature vector and without retraining the network, with impressive spotting results (only $\sim 1\%$ drop in both metrics) and very poor recognition performance. However, by performing the retraining strategy with the STE, as proposed in subsection 6.4.2, we get a well-performing system over all tasks despite the information loss due to the binarization step, while at the same time obtaining a very compact descriptor (512-bit long).

method	QbE	QbS	CER	WER
w/o training	90.99	94.11	92.5	86.9
w/ training	91.31	93.69	5.7	15.4

Table 6.12: Report on QbE & QbS (MAP), as well as recognition for binarized intermediate feature vector

Moreover, we evaluated the efficiency of the proposed word representation with an alternative method. We have computed statistics over the edit distance between words in our lexicon and the corresponding statistics when using cosine distance on either the proposed *binary representation* or the *PHOC* representations (*level*5 unigrams). We used the Kullback-Leibler divergence (KL) to express the statistics' correlation between the reference edit distance and the considered representations, and found the two divergences to be 0.0224 (proposed), 0.0417 (PHOC). Hence, the proposed binary representation is closer to the desirable statistics, as expressed by edit distances between lexicon words, compared to the state-of-the-art PHOC descriptor.

We have also compared the two proposed QbS/KWS strategies. The system under consideration is the fine-tuned *WSRNet+LM* and we distinguish the use of the binarization strategy, as shown at Table 6.13. We compare QbS using the Autoencoder module (section 6.4.2) versus the Forced Alignment approach (section 6.4.2). Notably, the Forced Alignment approach not only increases spotting performance, which was expected, but also leads to a slightly better result when used with the compact binarized vector. Of course, the improvement is achieved at the cost of computational effort, since Autoencoder-based QbS relies on comparing fixed-length representations with a simple distance metric, and is thus very efficient.

Finally, we present a comparison of our method versus state-of-the-art methods for KWS and Word Recognition in tables 6.11 and 6.15 respectively. In KWS, we have the

method	binarization	MAP
Autoencoder module	No	95.67
	Yes	93.69
Forced Alignment	No	96.27
	Yes	96.33

Table 6.13: Comparison of different approaches for QbS KWS. The two methods for QbS integration are compared, while using or not the descriptor binarization scheme.

Method	CER	WER
Sueiras et al. [168]	8.8	23.8
Wigington et al. [184]	6.07	19.07
Krishnan et al. [101]	6.34	16.19
Dutta et al. [44]	4.88	12.61
Proposed Models		
WSRNet+LM	5.0	13.9
WSRNet+LM (binarized)	5.7	15.4
WSRNet+LM+beam	4.8	13.6
WSRNet+LM+beam (binarized)	5.1	15.0

Table 6.14: Comparison of the state of the art for word recognition versus variations of the proposed method. The greedy decoding variation versus beam search-based variation results are reported (+beam). All figures are CER and WER percentages computed for the IAM dataset.

best results while having a very compact descriptor at the same time. In recognition, note that while our results are marginally very close to the leading method of Dutta et al. [44], they use methods such as extensive data augmentation with local deformations and pretraining with synthetic datasets, none of which is part of the proposed pipeline. Also note that our network, including all its subcomponents, is considerably smaller compared to vast VGG-based models such as PHOCNet [166, 167].

6.5 Feature-based Extension to Line-Level Spotting

??

Lending from ideas discussed previously at this chapter, we present a keyword spotting method that is based on deep feature extraction and fast query to text line matching. The key idea of the proposed method is that the features extracted from the convolutional output of a PHOC estimation network, trained on word images, should be similar in the case when the input is a line image or even a document image. We should stress that the convolutional layers are not constrained by the size of the input image. As a result, the keyword search procedure can be addressed as a feature matching procedure, 1-d search when the input is a line image or 2-d when the input is a document image (segmentation-free approach). Based on the success of line segmentation methods [67], we assume line images as input since the matching procedure is significant simplified (1-d search).

Our processing pipeline consists of three components, all modeled as neural networks [145]: (a) the *Character Width Normalization*. With this component we can normalize inputs, rendering horizontal zones of deep features consummerate to one another,

Method	QbE	QbS
Attributes+KCSR [9]	55.73	73.72
PHOCNet [166]	72.51	82.97
HWNet [102]	80.61	-
Triplet-CNN [185]	81.58	89.49
PHOCNet-TPP [167]	82.74	93.42
DeepEmbed [100]	84.25	91.58
Zoning Ensemble PHOCNet [148]	87.48	-
End2End Embed [101]	89.07	91.26
DeepEmbed [101]	90.38	94.04
Synth+DeepEmbed [101]	-	95.09
HWNetV2[103]	90.65	-
Proposed Models		
WSRNet+LM	92.05	95.67
WSRNet+LM (binarized)	91.31	93.69
WSRNet+LM+FA	-	96.27
WSRNet+LM+FA (binarized)	-	96.33

Table 6.15: Comparison of the state of the art for keyword spotting versus variations of the proposed method. Autoencoder based vs Forced Alignment (+FA) QbS results are reported. Figures are MAP percentages computed on the IAM dataset.

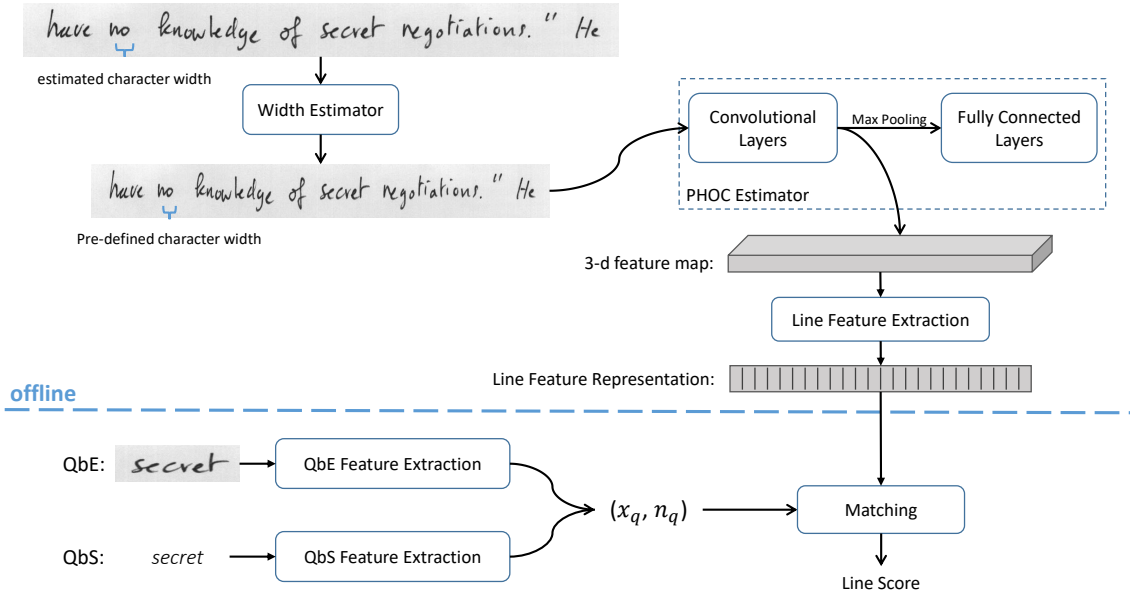


Figure 6.11: Line-level KWS system overview. Word images and string attributes are encoded into a common space, suitable for performing an efficient line matching approach on the sequence of convolutional features representing the line image. x_q is the final feature vector and n_q is the number of line segments extracted from the query.

and enabling fast query matching, since instances of the same word have similar width at the normalized line images. (b) the *Feature extraction*. This component is responsible for producing discriminative features for the query as well as the text lines of the document collection. Deep features are defined as the activations of an intermediate layer, produced

when feeding an image to the network. Such features have proven to be robust on training-test content disparity; for example they have been successfully applied to cross-language KWS tasks without the need for model refinement [146]. (c) the *Common Space Encoder*. This component enables the QbS option, i.e. using a string as a query instead of a word image. This is achieved by training two distinct networks that generate feature vectors in the same feature space, whether the input is an image (QbE) or a PHOC representation (QbS).

After having processed the query and the text lines, a step which is preformed offline, the query can be compared against the set of text lines on the fly using a fast matching procedure. Matching of a specific query with a single text line can be accomplished very efficiently due to two factors: (a) matched features are extracted by max-pooling over a set number of horizontal zones (b) the size of the pooling filter is fixed given the word length of the query, due to the Character Width Normalization component. Matched tokens are then sorted and returned to the user. The proposed model processing pipeline together with the query to text line matching can be reviewed in Fig. 6.11.

One interesting aspect of this work is that, all the aforementioned components focus on the simplest word-level image representation and collaborate in order to efficiently address the complex problem of line-level KWS.

6.5.1 Character Width Normalization

The existence of different writing styles as well as different scales leads to considerable image width differences, even for the same word. This variance, closely related to scale, complicates the line matching procedure, since we cannot pre-define the space that a specific word, the query word, occupies. Therefore, in order to build an efficient KWS system, we propose an image normalization scheme, where different instances of the same word have the same width. Key step towards such a normalization is the estimation of the average character width which is performed by a convolution neural network since it can be easily correlated with visual features (e.g. curvature of strokes).

The normalization step should be able to estimate the character width irrespective of the size of the input image (word or line). One simple and efficient way to perform the width estimation on both categories, word and line, is to extract a set of fixed-size patches from the images. The patches are randomly sampled from the image while their number is proportional to the size of the input image. The width estimation network $f_w()$, expects a fixed-sized image patch as input and provides an estimation of the character width for the specific patch. As the task at hand is a regression problem, the network f_w is trained using the mean squared error (MSE) loss function.

Even though the problem was clearly stated, the required labels for training the CNN, i.e. the character widths corresponding to image patches, are not known in advance. We follow a simple procedure for the assignment of the average character width (label) to each image patch. Given a word image and its transcription, the expected average character width (label) of every patch generated by this word image is the value w/n_c, where w is the image width and n_c is the number of characters of the word's transcription. Albeit the generated regression labels are not accurate, they suffice for this regression task since a few pixels difference will not affect the final result. In fact, the trained network provides an average loss of 3 pixels at test set, which is acceptable for the purpose of the normalization task.

Given an image I, a set of patches P_i, $i = 1, ..., n_p$ is generated and each patch has its own width estimation $w_i = f_w(P_i)$. It is possible that the system produces erroneous

character width estimations, especially for the case of patches not containing representative information (e.g. patch which mostly contains blank space). This issue is addressed by the assignment of the median value $w_c = median(\{w_i\})$ (more likely to correspond to the average character width) as the estimation for the whole image.

After estimating the average character width, we normalize the initial image to a fixed average character width w_{ref} (defined by the user). This normalization corresponds to a simple image resizing, for which the width is rescaled by a factor w_{ref}/w_c. The normalization process is depicted at Figure 6.11.

6.5.2 Feature Extraction

A neural network, trained to learn a semantic prediction task can be used to easily extract so-called deep features [195]. We use deep features to describe both text lines and queries since such features are known to outperform traditional feature extraction methods on various vision tasks. These features are extracted as activations of a neural network with a feed-forward architecture, tuned to a semantic attribute prediction task.

In particular, the task is predicting word-level attributes that correspond to the existence or absence of a unigram or bigram at a specific section of the word. To this end, we adapt on the PHOCNet architecture inspired by the VGG architecture [166, 167]. The input is a word image of arbitrary size, processed by a convolutional backbone that leads to a fully-connected head topped by sigmoid outputs. Each sigmoid output predicts a word semantic attribute, e.g. whether the letter 'b' exists on the first half of the word, or whether the bigram 'st' exists on the second third of the word, and so on.

The proposed PHOC estimator network bears two differences w.r.t. to the PHOCNet architecture. The first novel point is the application of adaptive pooling on a specific region of interest (ROI), which corresponds to a part of the initial input image. We perform training using word images that contain information from neighboring words, i.e. not a tight crop of the word image as the word segmentation ground truth indicates (see Figure 6.12). This training strategy tunes the neural network in such a way that it can efficiently generalize to inputs corresponding to entire line images. In addition, it enables the creation of more robust filters, not affected by the existence of neighboring image elements. To this end, a bounding box with the region of interest is also defined for the pooling operation as it is shown in Figure 6.12. Hence, the training input consists of pairs of images and bounding boxes.

The second novel point is related with the pooling scheme used to create a fixed-size bridge with the fully-connected head. We employ a horizontal zoning-based scheme, dividing the text-line or word input into fixed-size horizontal segments. This scheme follows the rationale of zoning techniques that have been widely used for keyword spotting [56]. The use of pyramidal pooling is also avoided, contrary to [166, 167]. We further simplify the above mentioned pooling scheme by completely ignoring the segmentation scheme over the convolutional map and instead, apply max-pooling over the entire map. This procedure will result to the creation of a feature vector whose size will be equal to the depth of the last convolutional layer. Furthermore, the application of a max-pooling operation over the entire convolutional map drastically reduces the size of the generated feature vector and therefore the number of parameters of the upcoming fully connected layer. The significance of this observation in the context of the proposed system, is that each word is represented using a collection of different filter responses instead of a sequence of feature vectors (zoning) or a set of sequences (pyramidal). The creation of a single feature vector simplifies the forthcoming matching procedure which can be accomplished

using a nearest neighbor algorithm. The proposed pooling strategy hence results to the creation of a simplified and compact feature descriptor, suitable for fast matching.

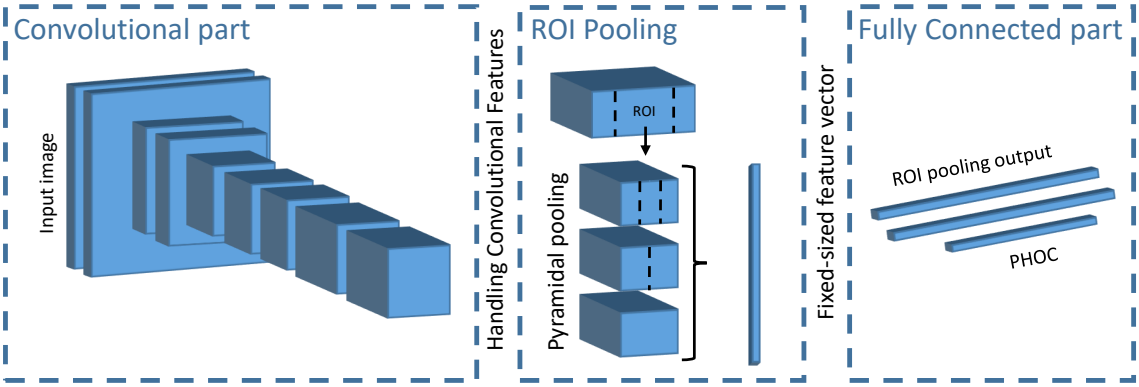


Figure 6.12: The PHOC estimation CNN consists of three distinct parts: 1) Convolutional 2) ROI Pooling and 3) Fully connected. Training is performed on an extended image containing a word instance along with the ROI bounding box of the word.

Given that f_c is the function consisting of the convolutional layers, f_p is the pooling operation over the ROI and f_{fc} is the fully connected part which results to the PHOC description, the PHOC estimation process can be formulated as follows:

$$x = f_p(f_c(I), b), \quad y = f_{fc}(x) \tag{6.10}$$

where I is the input image, b is the ROI bounding box, $x \in \mathbb{R}^{n_d}$ is the extracted deep feature vector and $y \in \mathbb{R}^{n_{phoc}}$ is the estimated PHOC. Note that n_d is an important hyper-parameter of the PHOC estimation network, since it defines the dimensionality of the extracted deep features.

6.5.3 Common Space Encoder

The PHOC estimation network, after discarding the fully connected layers, provides discriminative features, ideal for a line-based approach. Nevertheless, the discarded layers create the connection between the convolutional features and the PHOC representation which is mandatory for the QbS scenario. Therefore, an inverse function is required for the QbS scenario, i.e. a function that accepts a PHOC embedding (input) and produces a feature vector which is similar to the feature vector generated by the PHOC's estimator convolutional part (output).

One problem that may arise with this function is related with the fact that each input (PHOC representation) corresponds to a variety of outputs (word images of the same word written by different writers) and therefore the problem is ill-posed, since the function has to describe an one-to-many relation. This problem can be effectively solved by choosing to encode the PHOC embedding y as well as the convolutional feature vector x into a common space of fixed dimensionality n_e.

The encoding process is performed by two separate encoding functions, namely e_p (encodes the PHOC embeddings) and e_f (encodes the convolutional features), which are modeled by neural networks consisting of fully connected layers. Therefore, given a pair ($x \in \mathbb{R}^{n_d}$, $y \in \mathbb{R}^{n_{phoc}}$) the resulting $e_p(y) \in \mathbb{R}^{n_e}$ and $e_f(x) \in \mathbb{R}^{n_e}$ must be identical in the encoding space.

A critical aspect concerning the training of the encoding functions is the creation of an appropriate loss function. The obvious goal is to minimize the distance between $e_p(y)$ and $e_f(x)$, but such an approach may lead to trivial solutions (e.g. projection to the origin point). To address this problem, the training is performed using pairs of different words, i.e. (x_1, y_1) and (x_2, y_2), and the goal is the minimization of the encoding distance within the same word while preserving an appropriate distance between different words.

Given a distance $d(\cdot, \cdot)$, the inner distance i and cross distance c are defined as:

$$\begin{aligned} i_1 &= d(e_f(x_1), e_p(y_1)), & i_2 &= d(e_f(x_2), e_p(y_2)) \\ c_1 &= d(e_f(x_1), e_f(x_2)), & c_2 &= d(e_f(x_1), e_p(y_2)) \\ c_3 &= d(e_p(y_1), e_f(x_2)), & c_4 &= d(e_p(y_1), e_p(y_2)) \\ i &= (i_1 + i_2)/2 & & \quad (6.11) \\ c &= (c_1 + c_2 + c_3 + c_4)/4 & & \quad (6.12) \end{aligned}$$

The obvious choice is to minimize the inner distance, while at the same time maximize the cross distance, i.e. the loss would be $L = i - c$. However, our goal is to preserve a cross distance at least as big as the reference distance $m = d(y_1, y_2)$ in order to preserve the correlation of the initial PHOC representations. We call this reference distance as margin m. The margin term is crucial since the data distribution on the new space (with the help of the margin term) is expected to be similar to the data distribution on the PHOC space. In fact, the cross distance should be always greater than the inner distance by the aforementioned margin ($c > i + m$) in order to ensure separable classes of words. Therefore, the loss function is formulated as:

$$L = i + \max(0, i - c + m) \quad (6.13)$$

Implementation details: 1) The e_p consists of 4 fully connected layers with ReLU non-linearities, while e_f is a linear transformation. 2) The dimension of the encoding space is set to $n_e = 128$. 3) The cosine distance is selected as the distance function d. 4) The pairs (x, y) are selected such as their PHOC representations are not far apart, i.e. $d(y_1, y_2)$ is less than a threshold t. The latter assists convergence, since unconstrained selection of pairs may generate dissimilar pairs which are easy to separate.

6.5.4 Line Image Features and Matching

Following the definitions of the subsystems and their respective networks, this section contains a detailed description on how the aforementioned systems are cooperating in order to perform line based KWS.

We can distinguish two main steps: 1) Extract compact features from a line image and store them. This step is performed off-line in a KWS application. 2) Given a query (either image or string), extract appropriate features and compute a matching score between the specific keyword and the line (see Fig. 6.11).

Line Features Extraction

The extraction of line features consists of three steps presented in Figure 6.11, as part of the overall system:

1. Resize the input line image ($h_i \times w_i$) with respect to a pre-defined average character width w_{ref} using the Character Width Normalization component (see Section 3). The size of the resized image is $h \times w$.

2. Apply the convolutional part of the PHOC estimator network (f_c, see section 4) to the resized image produced in step 1. The output corresponds to a 3-d feature map, for which the height and width is proportional to the height and width of the resized line image ($h/4 \times w/4$ - the two max pooling layers of kernel and stride 2 of f_c downsample the image by 4) while its depth corresponds to the number of filters of the final convolution layer ($n_d = 512$).

3. Organize the aforementioned 3-d map ($h/4 \times w/4 \times n_d$) of visual features into a sequence of feature vectors ($n_l \times n_d$). The organization of the 3-d map into a sequence of feature vectors makes possible the efficient matching between a query word and a line image. The 3-d map is reduced by making use of local max-pooling performed on non-overlapping segments of size $h/4 \times w_{step}$, where $w_{step} = w_{ref}/4n_{quant}$. As a result, each line image is described by a short sequence of feature vectors reducing the storage cost as well as the matching time. This step is depicted at Figure 6.13.

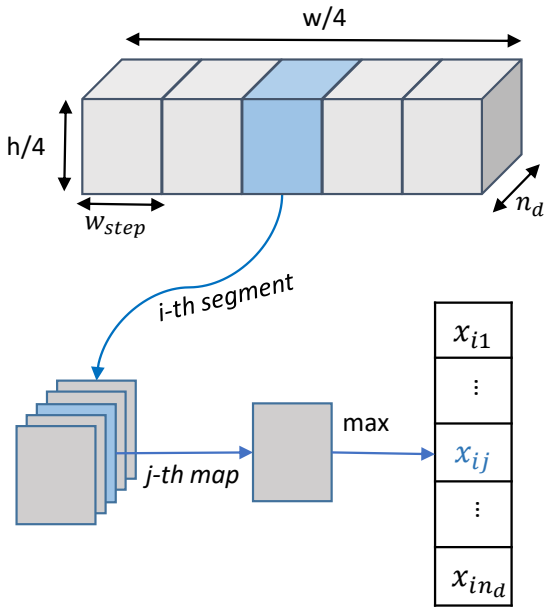


Figure 6.13: Extraction of line features from the 3-d convolutional feature map. Each line is represented by a sequence of n_l feature vectors of size n_d. Each feature vector occupies w_{ref}/n_{quant} width in the input line image.

QbE Feature Extraction

Given a query image I_q, we apply the following steps:

1. Estimate the average character width of the image and use it to resize the image accordingly (the resized image is denoted as I'_q and its size is $h_q \times w_q$).

2. Calculate the number of line segments n_q by dividing the resized image's width by the step width w_{step} which is defined on the line features extraction section.

3. Apply the convolutional part of the PHOC estimator network (f_c, see section 4). This leads to the creation of a 3-d feature map whose size is $h_q/4 \times w_q/4 \times n_d$.

Finally, extract a single feature vector by performing max-pooling for each channel (i.e. taking the maximum value over $h_q/4 \times w_q/4$ values). It should be noted that during this step no ROI is defined.

4. Project the produced feature vector into the common encoding space using the encoding function e_f (see Section 6.5.3).

→ The final feature vector x_q is generated as: $x_q = e_f(f_p(f_c(I'_q)))$

QbS Feature Extraction

Given a query string s_q, we apply the following steps:

1. Calculate the number of line segments n_q, by multiplying the number of query's characters with the character quantization parameter n_{quant} (each character should occupy n_{quant} line segments).
2. Compute the PHOC embedding of the query string s_q.
3. Encode the PHOC embedding into the common encoding space using the corresponding neural network e_p (see Section 6.5.3).

→ The final feature vector x_q is generated as: $x_q = e_p(PHOC(s_q))$.

Line Matching

It is assumed that line features are represented as a sequence of vectors $\{x_i\}$, $x_i \in \mathbb{R}^{n_d}$, $i = 1, \ldots, n_l$. The query (either image or string) is represented as a single feature vector $x_q \in \mathbb{R}^{n_e}$. A necessary parameter which should be known beforehand in order to perform the matching is the number of line segments n_q corresponding to the query width, i.e. how many line segments make up the query. The number n_q is calculated during the query feature extraction step (see Sections 6.2 and 6.3).

For each line, a set of scores is calculated using the cosine distance. In more detail, the scoring is performed using the cosine distance between the query feature vector and a feature vector created using a max-pooling procedure over consecutive n_q line features, i.e.:

$$\bar{x}_i[k] = \max_{j=0,\ldots,n_q-1} x_{i+j}[k],\ k = 1, \ldots, n_d - n_q + 1 \tag{6.14}$$

$$d[i] = 1 - cos(e_f(\bar{x}_i), x_q), \quad \bar{x}_i \in \mathbb{R}^{n_d}, x_q \in \mathbb{R}^{n_e} \tag{6.15}$$

The merge of consecutive n_q line features into a single feature vector, using a sliding window approach of max operations, is visualized at Figure 6.14. Since a set of feature vectors x_i can be extracted from a line image (the total number is equal to $n_l - n_q + 1$), the final score is the minimum among all scores calculated using the set of feature vectors $\overline{x}_i$:

$$line\_score = \min_{i=1,\ldots,n_l-n_q+1} d[i] \tag{6.16}$$

For the matching step it is imperative for the two feature vectors to have the same dimensions. To this end, each feature vector $\overline{x}_i$ is projected to the common encoding space using the network e_f. This task has been already performed for the query feature vector x_q during the feature extraction step. The simplicity of the matching procedure described above, is a considerable advantage of the proposed method, since it reduces the retrieval time.

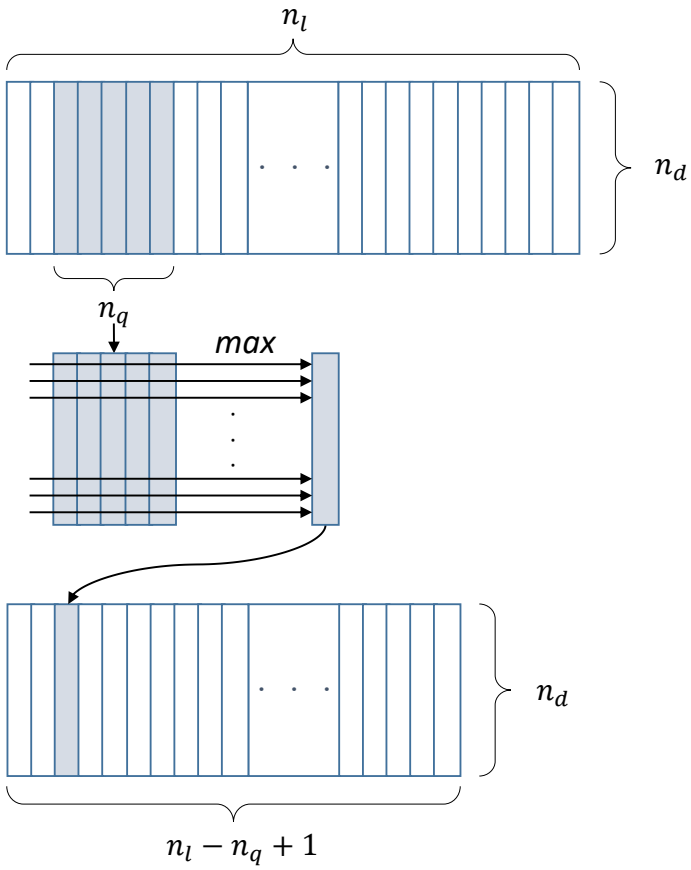


Figure 6.14: Line features transformation according to query size n_q. This step is essential in order to have comparable query and line features.

6.5.5 Experimental Results

Experimental Setup

The performance of the proposed system was evaluated on IAM dataset. In order to be comparable with the results reported in the bibliography, we follow the two most widely used setups of the IAM dataset. **IAMDB1**: 882 queries selected as in [50] using all non-stop words that appear at least once in the training set as well as the test set. **IAMDB2**: All non-stop words among the 4000 most frequent words that also occur in the training set are selected as queries as in [53], resulting in 3421 queries in total.

In addition, for measuring the performance of the proposed system we consider two possible scenarios as in [50]. In the first scenario (**local**), a local threshold is used for each keyword separately. Concerning the second scenario (**global**), a global threshold is used that is independent of the keyword. For a vocabulary of common keywords, local thresholds can be optimized at training stage. On the other hand, for arbitrary out-of-vocabulary keywords, a global threshold has to be applied.

Another critical aspect for KWS techniques is the definition of the character set. Word-based approaches assume only lowercase and numeric characters [9]. On the contrary, line-based approaches [53] assume a wider variety of possible characters, including capital letters as well as some special characters (e.g. '/', '-' e.t.c). This variety also exists in the aforementioned query lists and therefore, for our experiments, we adopt the same

Approaches	QbE-PHOC	QbE-cfeat	QbS-PHOC
TPP	83.98	82.15	93.01
Zoning	83.66	82.24	92.47
Entire	83.27	81.73	91.11

Table 6.16: MAP (%) performance evaluation of different pooling strategies on the word-segmented IAM dataset.

characters set (68 unique characters) and create 5-level PHOC embeddings with as many unigrams (1020 dimensional vectors).

The performance of the KWS methods is recorded in terms of the Mean Average Precision (MAP) since it is a retrieval problem. The retrieval list consists of the lines in the test set sorted according to their matching score with the query. A line that contains the requested query is considered as a hit.

Pooling strategies on PHOC estimation

One major difference of the proposed system with respect to PHOCNet is the usage of a PHOC estimation network with only one segment at the adaptive pooling layer, between the convolutional and the fully connected part. We experimented on the impact of the pooling operation to the system performance by evaluating three different strategies at the adaptive pooling layer: 1) Temporal Pyramidal Pooling - TPP (initial PHOCNet architecture [167]) 2) Zoning into 5 segments (without the pyramidal scheme) 3) Using max-pooling on the entire convolutional output (1 segment).

These strategies are evaluated on the segmented words as in [167] and the query list is extracted similarly to [9]. It should be stressed that the results are not directly comparable with the state-of-the-art word-level KWS techniques due to the fact that the characters set used in this work is significantly different (see Section 6.5.5). The results are presented at Table 6.16, where we distinguish the QbE scenario, both on PHOC level (QbE-PHOC) and on deep feature level (QbE-cfeat) and the QbS scenario on PHOC level (QbS-PHOC). As the experimental results indicate, the pooling strategy does not play significant role since the performance of the system is similar regardless the pooling scheme or the feature level. As a result, using max-pooling on the entire convolutional output simplifies the subsequent matching procedure while barely affecting the performance of the overall system.

Encoder Performance

A major contribution of this work is that it enables both QbS and QbE scenarios on deep features level, using an encoding neural network which projects convolutional features as well as PHOC embeddings into a common encoding space. Therefore, we can evaluate both QbE and QbS on the new encoding space. For this experiment we used the pooling strategy over the entire convolutional map. The word level KWS with encoder component results to: **81.36%** and **89.43%** MAP for QbE and QbS respectively. These results are similar to the KWS scenario without the encoder even though the feature dimension is compressed (from 1020 of PHOC and 512 of convolutional features to 128 of the encoded features), while feature-based QbS is enabled.

methods	local	global
QbS with ROI	88.73	83.15
QbS without ROI	87.13	79.14
QbE with ROI	84.25	73.16
QbE without ROI	83.01	71.58

Table 6.17: Impact of the ROI variation on the KWS system performance. MAP (%) results are reported at the line-level IAMDB1 setup ($n_{quant} = 3$).

Line level KWS

Having confirmed the effectiveness of the system's components we proceed to the evaluation of the whole proposed KWS system using the IAMDB1 and IAMDB2 setups. It is worth to note that, even if QbE scenario is not considered on line-level by existing methods, we evaluated it using the query list of IAMDB1 setup since the corresponding query images can be detected at the training set. If there are more than one instance for a query string the query image is selected randomly (the reported QbE results are the average MAP of 10 runs). However, the IAMDB2 setup cannot be used for this case since query strings do not necessarily exist on the training set.

First, we evaluate the impact of the ROI variation of feature extraction network as shown in Table 6.17 using the IAMDB1 setup. The results indicate that the usage of a training scheme with expanded word images, as done in the ROI variation, assist the network to generate features that are more similar to the extracted line features.

The experimental results of the proposed method along with several state-of-the-art methods, for both IAMDB1 and IAMDB2 setups, are reported at Table 6.18. The proposed system, concerning the QbS scenario, significantly outperforms the majority of the line-based KWS methods reported in the literature and it achieves comparable performance compared to the best system [53]. Morever, the experimental results indicate the validity and the effectiveness of the proposed method, which enables both QbS and QbE scenarios, since the QbE scenario achieves comparable results with the QbS scenario (especially for the local evaluation scenario). The gap in performance between QbS and QbE cases can be attributed to the fact that the QbE approach may generates a feature vector that lies on the margin of the word cluster (defined by the feature vectors of all the instances of the same word), compared to QbS features that are unique per word and by default are in the center of the word cluster.

Width normalization is a necessary step, in the sense that without it the proposed KWS system fails. Specifically, QbS cannot be performed at all without having an estimate of the character width, since only the query string is available as input. On the other hand, QbE is still applicable (as a template matching operation) but it is not meaningful to compare words/line segments with notable difference in the size of the characters. To prove this, we tested the proposed pipeline without the normalization step (IAMDB1-QbE) and the results were: 56.77% MAP (local scenario) and 25.84% (global scenario), while using the normalization gives us 84.25% and 73.16% respectively (see Table 3). As expected, without width normalization the proposed approach fails completely.

Finally, we should stress that the proposed method can be used for real-time applications due to its storage and time efficiency. For clarity, we report some indicative storage and time requirements on the IAM dataset: the line image features, which computed of-

fline, require 260KB storage (without any quantization), while it takes around 0.28msec to compare a query to a line. To put time-constraints into perspective, we consider applying the full pipeline of a PHOCNet-based model in order to extract PHOC embeddings for both the QbE and the QbS alternatives. However, one should apply the full PHOC estimation neural network and not only the convolutional layers in order to get PHOC estimations over a line (using a sliding window approach over the convolutional output for computational efficiency rather than applying a sliding window over the initial image). In particular, using the width estimator we get segments of the convolutional feature map that correspond to the query width. The PHOC estimations are generated by applying the fully connected layers of the PHOC estimator at each segment. Such an approach introduces a major computational overhead, since fully connected layers have numerous channels and consequently lead to a high dimensional PHOC representation. On the contrary, using the encoder, we can mitigate all the computational effort from all line segments to the query, which is significantly faster at the cost of a small drop in accuracy. To prove the importance of the encoder, we used the PHOC estimator along with the width estimator, as described above, and we get 76.83% MAP on the IAMDB2 setup (QbS senario), i.e. an increase of 1.52% (similar increase to word-level results of Table 1). However, even though this result outperforms SoA, the computational cost is significantly increased: it takes more than 10× time compared to the proposed approach.

	IAMDB1		IAMDB2
methods	local	global	global
Fisher et al. [50]	68.92	47.75	-
Fisher et al. [48]	-	55.05	36.00
Toselli et al. [176]	-	-	61.03
Toselli et al. [173]	-	-	72.00
Frinken et al. [53]	-	-	76.00
proposed QbS	88.73	83.15	75.31
proposed QbE	84.25	73.16	-

Table 6.18: MAP (%) evaluation on the line-level IAMDB1 and IAMDB2 setups ($n_{quant} = 3$).

Chapter 7

Neural Network Compression

Convolutional neural networks (CNNs) deliver state-of-the-art results in a variety of computer vision tasks, including image classification [105], image segmentation [115] and object recognition [142]. The general trend has been to design deeper and more sophisticated network architectures in order to build higher accuracy models [170, 162, 77]. This trend is also the case for the document analysis research field, while PHOCNet is an indicative example of well-performing yet computationally intense and over-parameterized neural network. In this thesis, we have already addressed this problem in a indirect manner by exploring task-related architectural choices which can alleviate the overhead of specific layers (see Sections 6.3 and 6.4). Nonetheless, we have not delved into this important topic of model compression, which is relevant to a plethora of computer vision and machine learning tasks. To this end, in this section, we propose two novel model compression techniques. In order to evaluate them, we focus on the generic image classification task, while at the end of the chapter we consider the keyword spotting case, by compressing the neural network presented at Section 6.4.

7.1 Related Work

Since the advent of AlexNet [105], the first deep CNN to advance the state-of-the-art in a highly challenging image classification task, building deeper and wider neural networks has been the main research direction in order to further improve performance in a variety of visual recognition tasks. For instance, top results in the image classification task of the ILSVRC2017 challenge were achieved using an ensemble of DualPathNets [31] with the number of layers ranging from 98 to 131 and the largest model requiring 304 MB of storage and 16 GFLOPs of computation. While such requirements can easily be met by server-class computing systems, this is not the case for less powerful mobile or embedded devices, thus limiting the deployment of state-of-the-art models in many real-life applications. The limitation of computational resources has inspired researches to follow two main directions: a) design compact, yet well-performing architectures and b) develop methods to compress large pre-trained models with minimal loss in accuracy.

7.1.1 Compact Architectures

Many architectural improvements have been devised to improve the cost efficiency of CNNs, since carefully designed building blocks lead to compact models with compelling accuracy. A characteristic example of such a design is a multi-branched convolution block: the input of the block is split into different (low-dimensional) tensors, each branch performs

a transformation on these tensors and the transformed tensors are merged forming the block's output. GoogLeNet [170] and FractalNet [107] are characteristic examples of such a design, while ResNet [77] can be seen as an extreme case where one branch is the identity transform.

Driven by the fact that convolutional layers typically dominate the inference time and largely contribute to the number of parameters in a network, many recent works propose the use of cheaper convolutional blocks, usually referred to as Bottleneck Blocks, which typically approximate the initial "costly" convolutional layer with a set of cheaper convolutions. Cheap convolutions include pointwise convolutions (1×1 kernel) as well as grouped convolutions (each filter is applied only to a subset of the input channels). A common practise is the *reduce-transform-expand* strategy, where reduction and re-expansion of a feature map is performed with pointwise convolutions while the transformation is a regular convolution (on the reduced feature maps). This strategy is widely adopted by many recent works in order to build deeper, yet cost-effective models [77, 192, 186, 69, 197]. Grouped convolutions are used as an alternative, or even complement, to the *reduce-transform-expand* strategy, since they drastically reduce the required parameters by applying filters only on distinct groups of the input tensor [91, 186, 197]. A special case of grouped convolutions is channel-wise convolutions in which the number of groups is equal to the number of channels, commonly referred to as depthwise separable convolutions when combined with a pointwise convolution. Depthwise separable convolutions are considered as a key-component of several recent compact architectures [32, 85]. Alternatively, some recent architectures, such as DenseNets [87] and DualPathNets [31], improve parameter efficiency by increasing the connectivity between different layers of the network. This connectivity promotes feature reuse and encourages feature exploration by combining features of different layers and receptive fields. Moreover, it alleviates the vanishing-gradient problem, assisting the training of well-performing deep models. While all of the aforementioned networks improve parameter efficiency in some manner, fast inference depends on the availability of highly-optimized implementations of their building blocks.

7.1.2 Model Compression

Besides designing new architectures, researchers have explored different ways of compressing pre-trained models. Driven by the observation that such models tend to be hugely over-parameterized [39], this family of techniques aims to exploit this redundancy in order to reduce the model size, while preserving the model's accuracy.

Network pruning, pioneered in the early development of neural networks [109], seeks to induce sparsity in a neural network by removing unimportant connections. Pruning neural networks has regained interest in recent years as a means to compress state-of-the-art deep neural networks and enable their deployment on resource-constrained devices. The main assumption behind this method is that DNNs are usually over-parameterized and, thus, we can obtain comparable accuracy using a subset of the network parameters. Depending on the granularity of parameters that are removed, pruning methods can be generally categorized as *unstructured* or *structured*. Unstructured pruning—also referred to as weight pruning—removes individual parameters (connections), generating a sparse model that preserves the high-dimensional features of the original network. On the other hand, structured pruning removes individual channels/filters (neurons) producing a thinner, denser model. From a theoretical standpoint, unstructured pruning can preserve more accuracy under coarser compression rates. It does, however, require sparse tensor representations and operations [110] to translate the memory and computational savings

to practical performance gains on commodity hardware. Nonetheless, many hardware accelerators for DNNs have been proposed with support for unstructured sparsity as their key design goal [128, 120, 198, 188, 194, 60, 199, 80, 38]. Most existing work on weight pruning attempts to sparsify a network either *indirectly* by adding a sparsity inducing penalty term in the learning process [125, 27, 58] or *directly* by removing weights from a fully-trained reference network based on a "saliency" ranking criterion [109, 74, 70, 68]. In the first case, heavily-tuned hyper-parameter settings are often required to obtain high sparsity levels. In the latter case, it is necessary to perform multiple prune-retrain cycles—commonly referred to as fine-tuning—to minimize accuracy loss, which is typically very time-consuming and can, thus, hinder their application on large-scale networks.

Quantization of neural networks, which dates back to the 1990s [47], uses low-precision fixed-point instead of high-precision floating-point formats to represent the weights and/or activations of a model in order to reduce its memory footprint both in terms of storage and working memory during inference [35, 139, 88, 26]. When both weights and activations are quantized [139], there is the added benefit that one can replace power-hungry floating-point operations with bitwise/integer operations. The state-of-the-art *HWGQ* method [26] has trained popular networks such as AlexNet [105], GoogLeNet [170] and ResNet [77] using 1-bit binary weights and 2-bit quantized activations with accuracy losses in the range of $5 - 9\%$.

Tensor decomposition [39, 40, 113] techniques exploit parameter redundancy to derive low-rank approximations of the weight tensors. Although tensor decomposition can easily provide execution time speedups for inference, these methods typically incur severe accuracy loss under high compression requirements. Note that the pruning techniques that we already mentioned, contrary to decomposition techniques, typically employ an extra step of fine tuning the remaining weights.

7.1.3 Distillation

A prominent technique related to model compression is knowledge distillation from one complex network to a smaller, and thus compressed, one [23, 81, 36, 132]. In the typical setting of model distillation, the initial network is called teacher network, similar to an oracle, and the target model is called student model. The student network may have an architecture that is shallower and/or thinner than the teacher network [81, 132] or use cheaper convolutional blocks [36]. Distillation is performed by constraining the output distribution of the two networks to be similar (i.e. with KL divergence loss), while training the student model (along with the task-specific loss). In other words, distillation is in fact a soft labeling approach, where the class probabilities produced by the complex teacher model are used as soft, and thus more informative, targets for training the compressed student model. The main reasoning behind distillation is that the complex, most likely over-redundant, model would better capture the target distribution (e.g. correlation between the label "cat" and the label "tiger"), whilst the compact network could not converge easily to such a generalizing solution. Nevertheless, by using the teacher network as an optimization assistant, one can converge to more generalizing solutions for the student network. Empirical results hint that distillation, in the majority of the cases, leads to higher performance of the student model compared to the same network trained directly on the dataset. Note that the distillation concept is not limited to model compression and can be used whenever knowledge transfer is required.

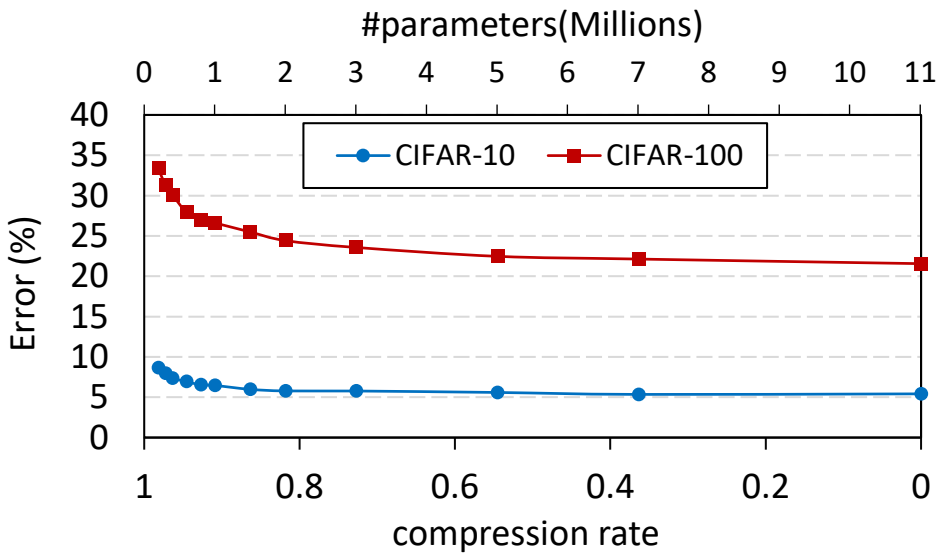


Figure 7.1: Error vs parameter budget trade-off curve for the CIFAR datasets. The initial network was WRN-16-8 [192].

7.1.4 Rethinking the Evaluation of Compression Techniques

A significant problem, often met at the literature, is the quantification of the compression results. A rather usual claim is: "We decrease the number of parameters by X%, while the accuracy drop is only Y%". Nevertheless, large networks are inherently over-redundant and we can significantly compress them without sacrificing accuracy by simply retraining a smaller network from scratch. It should be noted that compression capability depends both on the initial architecture and the selected dataset. Simply reporting the compression ratio vs accuracy loss trade-off is insufficient to validate the effectiveness of any compression technique, as the question that often arises is whether the same level of accuracy can be actually attained by training a thinner network with the same resource budget. In order to answer this question, we need to compare the accuracy of the compressed network to that of a thinner version of the original network with the same resource budget.

As an example, let us consider the number of parameters as our budget (analysis for flops is very similar). One can create a budget-constrained thinner model by using the same (reference) architecture and layer depth, while reducing the number of channels by the same percentage at each layer so that the size of the thinner dense model matches the size of the compressed model. If C'_{in}/C'_{out} are the number of input/output channels of a layer in the thin model, C_{in}/C_{out} are the initial input/output channels of the dense model and r_c is the overall compression ratio, then:

$$C'_{in} = \lfloor \sqrt{r_c}\, C_{in} \rfloor, \quad C'_{out} = \lfloor \sqrt{r_c}\, C_{out} \rfloor \tag{7.1}$$

Based on this analysis, we can build a trade-off curve reporting the error versus the parameter budget for a specific architecture and dataset, as shown in Figure 7.1. The plotted curve indicates that the WRN-16-8 network (Wide Resnet [192]) is over-parameterized for the fairly easy CIFAR datasets [104] and, thus, we can successfully train much thinner versions with considerably fewer parameters without sacrificing accuracy. For larger datasets and architectures, plotting such a curve can be rather impractical despite being insightful. Nevertheless, in order to objectively evaluate a compressed network, one should report the accuracy of an equivalent dense model with the same budget.

7.2 RecNets: Channel-wise Recurrent Convolutional Neural Networks

Our first attempt to tackle the model compression problem in an innovative way is to introduce a novel compact layer based on parameter sharing. Specifically, we present *RecNets*, a family of neural network architectures for computer vision tasks inspired by Recurrent Neural Networks (RNNs) [112]. Driven by the need for compact, yet accurate models, RecNets incorporate two important features that allow them to achieve this goal: (a) parameter sharing through a recurrent formulation of convolutional layers and (b) wide feature maps—which have been shown to improve network performance [192]. The recurrent logic embedded in the design of RecNets can be viewed as a different approach to the concept of cross-layer connectivity paths established by recent architectures such as DenseNets [87] and DualPathNets [31].

The main building block of RecNets comprises a *Channel-wise Recurrent Convolutional* (CRC) layer followed by a pointwise convolutional layer. A CRC layer differs from a typical convolutional layer in that it splits the input channels into disjoint segments and processes them in a recurrent fashion. In this way, a CRC layer reuses the same parameters for every segment, vastly reducing the number of parameters required to process the input. At the same time, the hidden state of the layer, generated by the recurrent process, captures information along the sequence of input channels, therefore enabling information flow between successive segments. The output of the CRC layer is the concatenation of the d generated hidden variables, which can be considered as a sequence generated by the underlying RNN. Therefore, CRC can approximate wide layers in a way similar to grouped convolutions [105], but retains a group connectivity through the recurrent formulation. The reduced representational power due to parameter sharing is replenished to some degree by the hidden state of the recurrent formulation.

7.2.1 Channel-wise Recurrent Convolutional Layers

In [31] the authors hint that residual networks [77] and their variations have a formulation similar to RNNs regarding the usage of successive layers. Driven by this observation, the main idea behind the RecNet architecture is to explicitly embed this recurrent logic in the network structure by replacing a typical convolutional layer by multiple successive smaller layers that form an RNN across the channel dimension. In short, we propose a recurrent formulation of a convolutional layer by splitting its input and output feature maps across the layer's channels into segments; the input segments form a sequence that is fed to an RNN, while the output sequence, generated by the RNN, consists of the output segments. We will henceforth refer to such layers as *Channel-wise Recurrent Convolutional* (CRC) layers.

Layer Definition

Let $\mathbf{x} \in \mathbb{R}^{C_{in} \times H \times W}$ be the input tensor and $\mathbf{h} \in \mathbb{R}^{C_{out} \times H \times W}$ be the output tensor of a CRC layer, where C_{in}, C_{out} are the number of input and output channels respectively and W, H the spatial dimensions of each individual channel. We split $\mathbf{x}$ and $\mathbf{h}$ into d segments across the channel dimension, i.e. $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{d-1})$ and $\mathbf{h} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{d-1})$ so that $\mathbf{x}_i \in \mathbb{R}^{S_{in} \times H \times W}$ and $\mathbf{h}_i \in \mathbb{R}^{S_{out} \times H \times W}$ for $i \in [0, d-1]$, where $S_{in} = C_{in}/d$ and $S_{out} = C_{out}/d$ are the input and output channels of each segment respectively. Assuming the $\{\mathbf{x}_i\}$ segments form the input sequence of an RNN and the $\{\mathbf{h}_i\}$ segments comprise

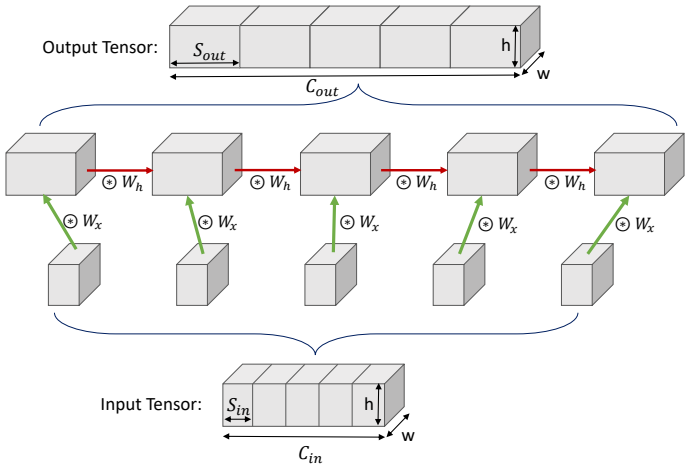


Figure 7.2: Visualization of a CRC layer with $d = 5$ segments. Input and output tensors are segmented across the channels' dimension into d segments, which form an input and output sequence, respectively. The output sequence is generated from the input sequence using the recurrent relation of Eq. (7.3).

its hidden state, we get the following recurrent formulation:

$$\mathbf{h}_i = \sigma(f_h(\mathbf{h}_{i-1}) + f_x(\mathbf{x}_i)), \tag{7.2}$$

where $\sigma()$ is a nonlinear activation function and $f_h()$, $f_x()$ are the transformation functions for the hidden state and input respectively. The output feature map y of the CRC layer is formed by the concatenation of the hidden state segments $\{\mathbf{h}_i\}$. A visualization of a CRC layer with $d = 5$ segments is given in Figure 7.2.

Let $\mathbf{W_h} \in \mathbb{R}^{S_{out} \times S_{out} \times k \times k}$ be the weight tensor for the hidden state, $\mathbf{W_x} \in \mathbb{R}^{S_{out} \times S_{in} \times k \times k}$ be the weight tensor for the input and $\mathbf{b} \in \mathbb{R}^{S_{out}}$ be the bias. Contrary to typical RNNs, we define every transformation of the input and hidden state as a convolutional layer with $k \times k$ filters so that Eq. (7.2) becomes:

$$\mathbf{h}_i = \begin{cases} \sigma(\mathbf{x}_i * \mathbf{W_x} + \mathbf{b}), i = 0 \\ \sigma(\mathbf{x_i} * \mathbf{W_x} + \mathbf{h}_{i-1} * \mathbf{W_h} + \mathbf{b}), \forall i \in [1, d-1]. \end{cases} \tag{7.3}$$

In this way the CRC layer simulates a typical convolutional layer with $C_{in} \cdot C_{out} \cdot k \cdot k$ parameters using only $(S_{in} + S_{out}) \cdot S_{out} \cdot k \cdot k = (C_{in} + C_{out}) \cdot C_{out} \cdot k \cdot k \cdot \frac{1}{d^2}$ parameters. The term $\frac{1}{d^2}$ can significantly reduce the layer's parameters when using an appropriate number of steps (e.g $d = 10$). To give an example, a CRC layer with 160 input and 640 output channels both split into 10 segments and has a total of 47,360 parameters (compared to 921,600 parameters for a typical 3×3 convolutional layer with the same input and output channels). This reduction is the result of re-using the same weights across each step $i = 0, \ldots, d-1$ and can lead to very compact architectures. The receptive field of the CRC layer depends on the number of recurrent steps d, therefore its representational power is tightly coupled to this hyper-parameter. By default, we use 3×3 convolution filters for processing both the input and hidden state of each step of the CRC layer. However, we will also explore the usage of 1×1 filters for either $\mathbf{W_h}$ or $\mathbf{W_x}$ in Section 7.2.4 to further reduce the number of parameters.

If we omit the $\mathbf{W_h}$ term from Eq. (7.3), which corresponds to the "history" of already processed feature maps, the CRC layer degenerates to a grouped convolution with shared

parameters across the groups, i.e. $\mathbf{h_i} = \sigma(\mathbf{x_i} * \mathbf{W_x})$, $i \in [0, d-1]$. However, the history term $\mathbf{h_{i-1}} * \mathbf{W_h}$ carries significant representational power, since it combines and encodes several previous feature maps, something that has been shown to strengthen feature propagation and encourage feature reuse [87].

The computational cost of a CRC layer in terms of FLOPs is $H \cdot W \cdot (C_{in} + C_{out}) \cdot C_{out} \cdot k \cdot k \cdot \frac{1}{d}$ (convolutions) $+ 2 \cdot H \cdot W \cdot C_{out}$ (additions), which is practically equivalent to applying two successive grouped convolutions of size $C_{in} \times C_{out} \times k \times k$ and $C_{out} \times C_{out} \times k \times k$ respectively, both using d groups. Even though the parameters are reduced by a quadratic term ($1/d^2$) and the computational cost is linearly reduced ($1/d$), the drawback of the proposed layer compared to typical (grouped) convolutional layers lies on the fact that the sequential nature of the recurrent formulation cannot be effectively parallelized.

In short, a CRC layer, which we will henceforth denote as CRC(S_{in}, S_{out}, d), is defined in terms of the following hyper-parameters: 1) S_{in}: number of input channels per segment, 2) S_{out}: number of output channels per segment and 3) d: number of segments (equivalent to layer's depth).

Non-Linearity

Another differentiation from the typical RNN formulation involves the non-linear function $\sigma()$ in Eq. (7.3). We distinguish three cases of interest that will be evaluated in Section 7.2.4:

1. *ReLU non-linearity*: The simple case of using the ReLU function, which is the most popular choice of non-linearity in CNNs.

2. *BN + ReLU non-linearity*: Use of a Batch Normalization (BN) [92] layer along with the ReLU function. To add a degree of freedom, the BN layer is not shared across the d steps, but instead we use d separate BN layers and drop the bias matrix b in Eq. (7.3). We denote each BN layer along with the ReLU non-linearity as $\sigma_i()$. This choice barely affects the number of parameters since the BN layer parameters are linear to the number of channels. Equation (7.3) now becomes:

$$\mathbf{h}_i = \begin{cases} \sigma_i(\mathbf{x_i} * \mathbf{W_x}), i = 0 \\ \sigma_i(\mathbf{x_i} * \mathbf{W_x} + \mathbf{h}_{i-1} * \mathbf{W_h}), \forall i \in [1, d-1]. \end{cases} \tag{7.4}$$

3. *Linear recursion*: An interesting case is the adoption of a linear recurrent formulation by dropping the $\sigma()$ term:

$$\mathbf{h}_i = \begin{cases} \mathbf{x_i} * \mathbf{W_x}, i = 0 \\ \mathbf{x_i} * \mathbf{W_x} + \mathbf{h}_{i-1} * \mathbf{W_h} + \mathbf{b}, \forall i \in [1, d-1]. \end{cases} \tag{7.5}$$

An important property of a linear recurrent formulation is that the output can be re-written without the intermediate hidden variables, as shown in Eq. (7.6), and thus the layer can be parallelized at the cost of pre-computing the convolved tensors of Eq. (7.6), where $\mathbf{W}^{*n} = \underbrace{\mathbf{W} * \mathbf{W} \cdots \mathbf{W}}_{\text{n times}}$.

$$\mathbf{h}_i = \sum_{j=0}^{i} \mathbf{x}_j * (\mathbf{W_x} * \mathbf{W_h}^{*(i-j)}) + \sum_{j=0}^{i} \mathbf{b} * (\mathbf{W_h}^{*j}) \tag{7.6}$$

Contrary to the previous cases where the non-linearity is applied at each recursion step, in this case we place a BN layer along with a ReLU non-linearity at the output of the CRC layer.

RNN Variants

A straightforward extension of the proposed RNN-based convolutional layer is the application of typical RNN variants. Specifically, we distinguish the following cases:

- *Multiple layered RNN:* In practice, RNNs have several layers, i.e. the output sequence of the first RNN is fed as an input sequence to the second RNN, etc. Each layer is described by the recurrent relation of Eq. (7.2) and uses a pair of shared parameter tensors $(\mathbf{W_x}^k, \mathbf{W_h}^k)$, where k is the layer's identifier.

- *Bi-directional RNN:* A major issue with typical RNNs is that they learn representations from previous time steps: $\mathbf{r_i} = f_r(\mathbf{r_{i-1}}, \mathbf{x_i})$. However, it may be helpful to incorporate information from future time steps: $\mathbf{l_i} = f_l(\mathbf{l_{i+1}}, \mathbf{x_i})$. Bi-directional RNNs combine both information flows, rightward $\mathbf{r_i}$ and leftward $\mathbf{l_i}$, at each time step i: $\mathbf{h_i} = f_c(\mathbf{r_i}, \mathbf{l_i})$. Sometimes, you might have to learn representations from future time steps to better understand the context and eliminate ambiguity.

- *Gated Recurrent Unit (GRU) Network:* A gated recurrent unit has an update gate, which decides on how much of information from the hidden state should be let through, and a reset gate, which decides on how much of information from the hidden state should be discarded [33]. GRU has been proposed as a cost-efficient alternative to Long Short Term Memory (LSTM) Networks and both networks aim to retain information from distant time steps, without vanishing it through time. We will only explore the GRU case, since one major goal of this work is the construction of compact architectures.

The aforementioned RNN variants will be evaluated at the experimental section 7.2.4. For the rest of the paper we will use the vanilla RNN formulation of Eq. (7.3) unless is stated otherwise.

7.2.2 Overall Architecture

Using the proposed CRC layer as the backbone, we design a family of neural network architectures, called RecNets, by using successive CRC layers connected by 1×1 convolutions. Specifically, each CRC layer is followed by a *Transition Block* layer, denoted TB(C_{in}, C_{out}), of C_{in} input channels and C_{out} output channels which consists of a 1×1 convolutional layer followed by a BN layer and a ReLU non-linearity. In essence, the role of a TB layer is to perplex the channels that are generated by the different segments in a CRC layer so that each input segment of the following CRC layer carries information about the entire output of the previous CRC layer. Using CRCs alone, whilst they perform complex computations of high receptive field, they have an inherently imbalanced input-output segment correlation (e.g. $\mathbf{h}_0$ has seen only $\mathbf{x}_0$, while $\mathbf{h}_{d-1}$ has seen the entire $\mathbf{x}$ - see Eq. (7.6)). Thus, if we create a network consisting only of CRC layers, this structural imbalance would be propagated and the leftmost segment, at any depth, would always be very narrow-sighted (similar concept to DenseNets [87]). We will henceforth refer to the composition of a CRC layer with a TB layer as a *Recurrent* module. Figure 7.3 provides an overview of a Recurrent module with $d = 4$ segments.

In the context of the RecNet architecture, a CRC layer expands the channels' dimension, while the subsequent TB layer shrinks it, preparing the input for the next CRC layer. If we fix the channel expansion at each CRC layer using an expansion parameter e ($S_{out} = e * S_{in}$ for every CRC layer), we can describe the whole network through e along

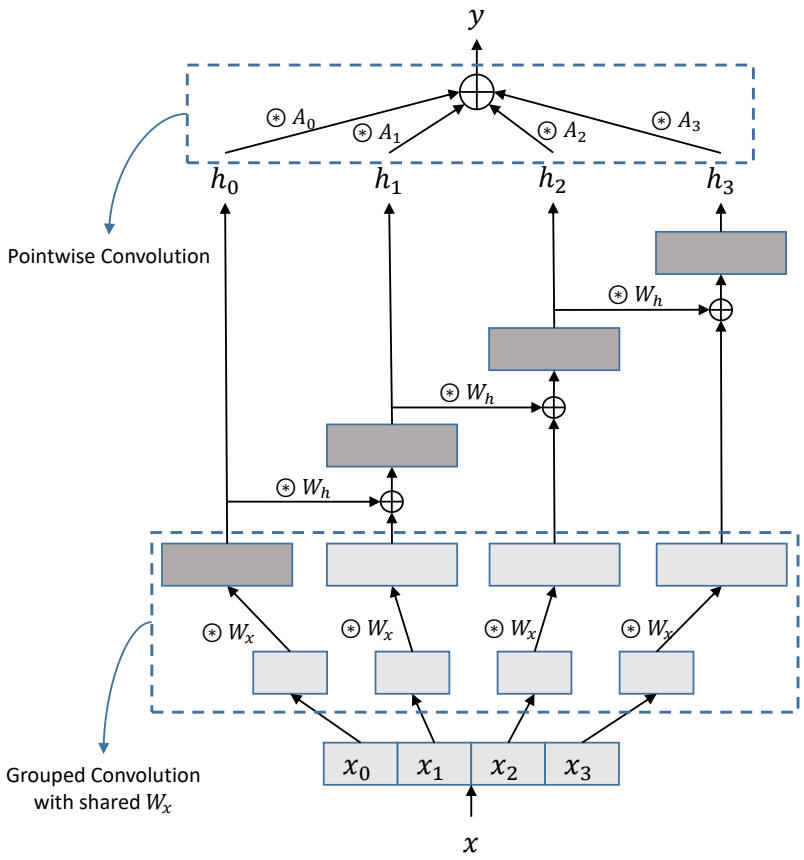


Figure 7.3: Visualization of a Recurrent module with $d = 4$ segments. Feature map $\mathbf{x}$ is segmented into $\{\mathbf{x_i}\}$, forming a sequence of length d. Each segment has S_{in} channels and is transformed into a new feature map of S_{out} channels. This procedure is equivalent to a grouped convolution with shared weights ($\mathbf{W_x}$). The new sequence $\{\mathbf{h_i}\}$ is generated using the recurrent weight $\mathbf{W_h}$ and corresponds to the output of the DRC layer. The Recurrent module output $\mathbf{y}$ consists of C_{out} channels and can be computed by a linear transformation, i.e. a point-wise convolution, over the DRC output $\{\mathbf{h_i}\}$.

with the input segment channels S_{in} and the depth d of each CRC layer. In this work, we experiment with a RecNet network composed of three pairs of Recurrent modules, which we will henceforth denote as *RecNet(e, S_1, S_2, S_3, d_1, d_2, d_3)*. The S_i hyper-parameters correspond to the segments' input channels S_{in} of the i-th pair of Recurrent modules. For short, we also use the acronym *RecNet-w-d*, where w is the maximum width (#channels) of the model ($w = e * \max\{S_i \cdot d_i\}$) and d is the overall depth of the CRC layers ($d = \sum d_i$). The complete RecNet architecture is given in Table 7.1.

7.2.3 Computational Simplification

The expansion parameter e (see Section 7.2.2) allows us to simulate wide layers, which may have great representational power (see Wide ResNets [192]). This, however, comes at the cost of large intermediate results between CRC and TB layers ($e \cdot C_{in} \times H \times W$), which increase memory requirements during inference. In practice, the CRC layer creates wide feature maps, while the TB layer reduces them in order to generate the input for the next CRC layer. Nevertheless, the recurrent formulation enables us to rewrite the successive

Table 7.1: RecNet(e, S_1, S_2, S_3, d_1, d_2, d_3) architecture for CIFAR. The expansion hyper-parameter e controls the output channels of each segment in a DRC layer ($S_{out} = e \cdot S_{in}$). The S_i hyper-parameters define the input channels of the segments, while the depth hyper-parameters d_i define the number of segments.

Layer/Block	Output Channels	Output Size
CONV (3×3) + BN + ReLU	$S_1 \cdot d_1$	32×32
CRC (S_1, $e \cdot S_1$, d_1)	$e \cdot S_1 \cdot d_1$	32×32
TB ($e \cdot S_1 \cdot d_1$, $S_1 \cdot d_1$)	$S_1 \cdot d_1$	32×32
CRC (S_1, $e \cdot S_1$, d_1)	$e \cdot S_1 \cdot d_1$	32×32
TB($e \cdot S_1 \cdot d_1$, $S_2 \cdot d_2$)	$S_2 \cdot d_2$	32×32
Max Pooling (2×2)	$S_2 \cdot d_2$	16×16
CRC (S_2, $e \cdot S_2$, d_2)	$e \cdot S_2 \cdot d_2$	16×16
TB($e \cdot S_2 \cdot d_2$, $S_2 \cdot d_2$)	$S_2 \cdot d_2$	16×16
CRC (S_2, $e \cdot S_2$, d_2)	$e \cdot S_2 \cdot d_2$	16×16
TB ($e \cdot S_2 \cdot d_2$, $S_3 \cdot d_3$)	$S_3 \cdot d_3$	16×16
Max Pooling (2×2)	$S_3 \cdot d_3$	8×8
CRC (S_3, $e \cdot S_3$, d_3)	$e \cdot S_3 \cdot d_3$	8×8
TB ($e \cdot S_3 \cdot d_3$, $i_2 \cdot d_3$)	$S_3 \cdot d_3$	8×8
CRC (S_3, $e \cdot S_3$, d_3)	$e \cdot S_3 \cdot d_3$	8×8
TB ($e \cdot S_3 \cdot d_3$, $S_3 \cdot d_3$)	$S_3 \cdot d_3$	8×8
Average Pooling (8×8)	$S_3 \cdot d_3$	1×1
Linear ($S_3 \cdot d_3$, $n_{classes}$)	$n_{classes}$	1×1

use of CRC and TB layers in a more compact way and merge them into a single layer.

Assume that a Recurrent module is denoted as Rec(S_{in}, S_{out}, C_{out}, d) = {CRC(S_{in}, S_{out}, d), TB($d \cdot S_{out}$, C_{out})}. For simplicity, let the TB layer consist only of a 1×1 convolutional layer, whose weight matrix is $\mathbf{A}$. Since the input channels dimension of the TB layer is $d \cdot S_{out}$, the matrix $\mathbf{A}$ is sized $C_{out} \times d \cdot S_{out}$ and can be divided into d groups as follows: $\mathbf{A} = [\mathbf{A}_0, \ldots, \mathbf{A}_{d-1}]$, $\mathbf{A}_i : C_{out} \times S_{out}$. If $\mathbf{h}^p = [\mathbf{h}^p_0, \ldots, \mathbf{h}^p_{d-1}]^\intercal$ are the output segments of CRC for a specific pixel p, then the output of the subsequent TB y^p for the pixel p can be expressed as:

$$\mathbf{y}^p = \mathbf{A} \cdot \mathbf{h}^p = \begin{bmatrix} \mathbf{A}_0 \ldots \mathbf{A}_{d-1} \end{bmatrix} \begin{bmatrix} \mathbf{h}^p_0 \\ \ldots \\ \mathbf{h}^p_{d-1} \end{bmatrix} = \sum_0^{d-1} \mathbf{A}_i \cdot \mathbf{h}^p_i \tag{7.7}$$

Based on the above formulation, the output of a Recurrent module can be computed as follows:

$$\mathbf{y} = \sum_0^{d-1} \mathbf{h}_i * \mathbf{A_i}\,, \quad \mathbf{h}_i = \begin{cases} \sigma(\mathbf{x}_i * \mathbf{W_x} + \mathbf{b}), i = 0 \\ \sigma(\mathbf{x_i} * \mathbf{W_x} + \mathbf{h}_{i-1} * \mathbf{W_h} + \mathbf{b}),\ \forall i \in [1, d-1], \end{cases} \tag{7.8}$$

Using the above notation, we obtain the same results without the concatenation of the output sequence $\{\mathbf{h_i}\}$ of the CRC layer, which results in large intermediate representations. Instead we use a summation over the separate responses of $\{\mathbf{A_i}\}$ at each step of the RNN formulation using an intermediate feature map of dimension S_{out}.

To attain an insight on the memory efficiency of this proposed variation, Let us assume the following layer: Rec(32,128,640,20). The CRC component has 640 input channels and 2560 output channels, while the TB component has 2560 input channels and 640 output channels. However, even though the intermediate feature map is very large (2560 dimensions), using the proposed variation at each step of the RNN, we get an 128-d intermediate feature map (the same memory space can be used at each step) which is subsequently transformed to a 640-d feature map. Therefore, using Eqs. (7.8), we simulate precisely the combination of a CRC and a TB layer with very large intermediate channel dimensions without explicitly generating these large feature maps.

7.2.4 Experimental Results

In this section we first evaluate different options related to the design of the CRC layer (see Section 7.2.1). We then explore RecNet's hyper-parameters controlling the depth and the width of the network, and, finally, we compare the proposed architecture against state-of-the-art networks.

The experiments are performed on the popular CIFAR datasets [104]. The CIFAR-10 and CIFAR-100 datasets consist of 32×32 color images, corresponding to 10 and 100 classes respectively. Both datasets are split into 50,000 train and 10,000 test images. We employ data augmentation during training following the de facto standard methodology, consisting of horizontal flips and random crops. A simple mean/std normalization (per channel) is used on the input images. Training on CIFAR is performed using the SGD algorithm with Nesterov's momentum with the initial learning rate set to 0.1, weight decay to 0.0005, dampening to 0, momentum to 0.9 and minibatch size to 64. The overall epochs are set to 200, while the learning rate is changing according to a cosine annealing schedule [116]. The learning rate scheduler is restarted at 20, 60 and 120 epochs.

Exploration of CRC layer variants

Non-linearity: As we presented earlier (see Section 7.2.1), one differentiation in the design of the CRC layer from the typical RNN formulation is the choice of the non-linearity function $\sigma()$. In order to evaluate the importance of this choice, we evaluate four different cases: 1) ReLU (typical RNN approach), 2) shared BN layer for all segments along with ReLU, 3) separate BN layers for each segment along with ReLU and 4) no $\sigma()$ function at all, i.e. evaluate a linear recursion formulation. For the evaluation we used the RecNet(4,8,16,32,10,10,10) architecture and the results are summarized at Table 7.2. The shared BN approach performs poorly, since it significantly constrains the feature map value range. The ReLU non-linearity, which corresponds to the usual RNN usage, performs considerably well, but is on par with the computationally simpler linear recursion approach. Note that the non-linear recursion and the linear recursion have very similar accuracy even though we expected the non-linearity to play a crucial role in high-performing recursive layers. Nevertheless, using separate BN layers at each segment along with the ReLU function leads to non-trivial improvement in accuracy at the cost of a minimal increase of the network's parameters. Therefore, the separate BN approach will be used for the rest of the experimental section.

Expansion Hyper-Parameter: The expansion hyper-parameter e controls the number of output channels of CRC layers (see Section 7.2.2) and by increasing its value we approximate wider architectures. We experimented with the RecNet(e,8,16,32,10,10,10) architecture and the results are presented at Table 7.3. One can observe that higher values

Non-linearity σ	CIFAR-10	CIFAR-100
ReLU	94.62	75.48
shared BN + ReLU	87.44	62.70
separate BN + ReLU	95.15	78.25
None (linear)	94.24	75.71

Table 7.2: Exploration of non-linearities for the case of RecNet(4, 8, 16, 32, 10, 10, 10) architecture.

of the expansion hyper-parameter e lead to both improved performance and an increase in the number of parameters. However, the accuracy gain between $e = 4$ and $e = 8$ is trivial compared to the increase in the number of parameters. Therefore, we conclude that $e = 4$ is a good compromise with regards to the size-accuracy trade-off of the model.

e	#params	CIFAR-10	CIFAR-100
1	424K	93.38	71.94
2	824K	94.16	73.10
4 (default)	1,769K	95.15	78.25
8	4,239K	95.47	79.33

Table 7.3: Impact of the expansion hyper-parameter e on the performance accuracy for the case of RecNet(4, 8, 16, 32, 10, 10, 10) architecture.

Kernel Size of $\mathbf{W_x}$ and $\mathbf{W_h}$: The CRC layer can be still effective even one of the weight tensors $\mathbf{W_h}$, $\mathbf{W_x}$ uses 3×3 convolution filters, while the other uses 1×1. By selecting 1×1 convolution for either of the tensors, we further reduce the model parameters. Assuming the RecNet(4,8,16,32,10,10,10) architecture, these implementation choices are evaluated in Table 7.4. We can observe that the use of 1×1 convolutions compresses the parameters by $5-20\%$ at the cost of a small decrease in performance. Nevertheless, we shall continue the experimental section with 3×3 kernels for both matrices, since they correspond to the best performing setting.

$\mathbf{W_x}$	$\mathbf{W_h}$	#params	CIFAR-10	CIFAR-100
3×3	1×1	1,425K	94.71	77.48
1×1	3×3	1,683K	94.72	77.42
3×3	3×3	1,769K	95.15	78.25

Table 7.4: Changing the kernel sizes of $\mathbf{W_x}$ and $\mathbf{W_h}$ for the case of RecNet(4, 8, 16, 32, 10, 10, 10) architecture.

Recurrent vs Grouped Convolution: To highlight the impact of the recurrent formulation, we also implemented a network with exactly the same parameters, but without

recurrence, i.e. each layer consists of a grouped convolution with shared parameters formulated as: $\mathbf{h}_i = \sigma_i(\mathbf{x_i} * (\mathbf{W_x} * \mathbf{W_h}))\, \forall i \in [0, d-1]$. Table 7.5 contains the evaluation of such a network which is considerably worse compared to the proposed one, since the CRC layer assumes a much more complex structure and larger receptive field.

	#params	CIFAR-10	CIFAR-100
recurrent	1,769K	95.15	78.25
grouped	1,769K	94.65	76.04

Table 7.5: Comparing the recurrent formulation vs using grouped convolutions with shared parameters. Both experiments correspond to a RecNet(4, 8, 16, 32, 10, 10, 10) architecture.

RNN Variants The proposed CRC layer adopts a single-layered one-directional RNN formulation. Nevertheless, popular variants of RNNs include multi-layered RNNs, bi-directional RNNs (BRNNs) as well as Gated Recurrent Unit (GRU) networks [33]. Multi-layered RNNs use multiple consecutive vanilla RNNs in order to build deeper architectures. Bi-directional RNNs learn representations from past as well future time steps in order to better understand the context and eliminate ambiguity. GRU networks learn to retain information from distant time steps without vanishing it through time.

The performance of these variants is presented in Table 7.6. The results hint that the more complex recurrent variants of the literature do not provide a significant performance boost, even though they are prominent for many tasks of sequential nature (e.g. NLP). This is not surprising, since these methods aim to better model the existing contextual information of long input sequences, while we have short sequences (defined by d_i). On the contrary, such approaches increase the complexity of the layer and therefore it is preferable to choose the simple RNN formulation as the default one.

recurrent unit	#params	CIFAR-10	CIFAR-100
RNN	1,769K	95.15	78.25
RNN, 2 layers	2,553K	95.21	78.05
BRNN	2,187K	95.32	78.54
GRU	2,748K	94.78	77.81

Table 7.6: RNN variants for the case of RecNet(4, 8, 16, 32, 10, 10, 10).

Depth/Width of RecNet

Having concluded on the most efficient non-linearity, expansion parameter and kernel size for the weight tensors, we proceed with the evaluation of the hyper-parameters which control the simulated width and the depth of the CRC layers, i.e. $S_1, S_2, S_3, d_1, d_2, d_3$. Specifically, we progressively increase the intermediate feature map channels ($e \cdot S_i \cdot d_i$) at each pair of Recurrent modules by increasing S_i ($S_1 < S_2 < S_3$) while having d_i fixed ($d_1 = d_2 = d_3 = d_f$) or vice versa. The results are summarized in Table 7.7. The reported depth corresponds to $d_1 + d_2 + d_3$ for simplicity. However, the depth of the network is $d_1 +$

d_2+d_3+7, if we include the first convolution layer as well as the six Transition Blocks. All the networks included at Table 7.7 perform well and are fairly compact, requiring at most a few million parameters. Compared to the popular MobileNet architecture [85] which has an accuracy of 73.65% on CIFAR-100 at 3.3M parameters, RecNet(4,8,8,8,5,10,15) has an accuracy of 73.68% at 0.3M parameters and RecNet(4,8,16,32,15,15,15) has an accuracy of 79.01% at 3.3M parameters.

Architecture (e, S_1, S_2, S_3, d_1, d_2, d_3)	acronym	depth	#params	CIFAR-10	CIFAR-100
RecNet(4, 4, 8, 16, 10, 10, 10)	RecNet-60-640	60	471K	93.45	74.94
RecNet(4, 4, 8, 16, 15, 15, 15)	RecNet-90-960	90	863K	94.32	76.31
RecNet(4, 4, 8, 16, 20, 20, 20)	RecNet-120-1280	120	1,406K	94.58	77.62
RecNet(4, 8, 16, 32, 10, 10, 10)	RecNet-60-1280	60	1,769K	95.15	78.25
RecNet(4, 8, 16, 32, 15, 15, 15)	RecNet-90-1920	90	3,306K	95.22	79.01
RecNet(4, 8, 16, 32, 20, 20, 20)	RecNet-120-2560	120	5,444K	95.46	80.31
RecNet(4, 8, 8, 8, 5, 10, 15)	RecNet-60-480	60	316K	93.28	73.68
RecNet(4, 8, 8, 8, 10, 15, 20)	RecNet-90-640	90	537K	93.95	75.86
RecNet(4, 8, 8, 8, 10, 20, 30)	RecNet-120-960	120	930K	94.44	77.58
RecNet(4, 16, 16, 16, 5, 10, 15)	RecNet-60-960	60	1,137K	94.54	77.91
RecNet(4, 16, 16, 16, 10, 15, 20)	RecNet-90-1280	90	2,028K	94.77	78.00
RecNet(4, 16, 16, 16, 10, 20, 30)	RecNet-120-1920	120	3,569K	95.47	79.38

Table 7.7: Exploration of hyper-parameters S_1, S_2, S_3, d_1, d_2, d_3, which control the size of the RecNet architecture. The acronym *RecNet-d-w* reports a model with a depth d and a maximum width w.

It should be noted that the reported depth is in line with the parallelization weakness of the proposed CRC layers (see Section 7.2.1), since the simulated wide layers are only six (see Table 7.2.1). In other words, implementation-wise, the recurrent CRC module is unfolded into a sequential set of simple convolution layers, resulting to a notable architecture depth.

Comparison to the state-of-the-art

Table 7.8 shows the performance of several compact state-of-the-art models on CIFAR-100—this dataset is significantly more challenging than CIFAR-10 and, therefore, it is easier to draw meaningful conclusions about the evaluated architectures. We select models consisting of at most a few million parameters. For clarity, we only report two RecNet architectures in Table 7.8. Specifically, we select two extreme cases, one with only 240K parameters and one with almost 9M parameters. Even though RecNets perform well compared to the majority of reported networks, its superiority is effectively shown in Figure 7.4. RecNets demonstrate an improved performance-parameter trade-off compared to the majority of other networks, with the exception of DenseNets, which provide a comparable trade-off. In fact, RecNets and DenseNets share the concept of a history preserving step, but implement it in a different manner (hidden state of recurrent formulation in case of RecNets and densely-connected layers in case of DenseNets). This observation hints towards a property that seems essential to constructing efficient compact networks: *re-using/perplexing information at different receptive fields.*

Model/Method	#Params (Million)	Accuracy (%)
GoogleNet [170]	6.8	77.91
ResNet-110 [77]	1.7	75.73
WRN-40-1 [192]	0.6	69.11
WRN-40-2 [192]	2.2	73.96
WRN-40-4 [192]	8.9	77.11
MobileNet [85]	3.3	73.65
ResNeXt-29,1x64d [186]	4.5	77.85
PyramidNet-48 [69]	1.7	76.78
PyramidNet-84 [69]	3.8	79.12
DenseNet-40-12 [87]	1.0	75.58
DenseNet-100-12 [87]	7.0	79.80
DenseNet-BC-40-18 [87]	0.4	74.72
DenseNet-BC-100-12 [87]	0.8	77.73
RecNet-30-60	0.2	70.85
RecNet-120-2880	8.2	80.56

Table 7.8: Accuracy of state-of-the-art networks (<10M parameters) on CIFAR-100.

7.3 Weight Pruning: Inducing Sparsity Over Weights

Even though the previously proposed RecNet architecture provides notably compact, yet well performing models, it cannot be implemented efficiently due to the sequential nature of the CRC building block (cannot be parallelized and thus it cannot be implemented in a GPU efficient way). In this section, we propose a pruning-based approach for model compression, tackling the problem from an entirely different point of view. Specifically, we focus on unstructured weight pruning, rather than channel/filter pruning, which leads to the creation of sparse weight tensors. Concerning the proposed approach, we introduce a robust compressive learning framework that efficiently prunes network parameters during training with minimal computational overhead. First, we propose a methodology to remove unimportant weights in every training iteration under user-defined sparsity constraints using fast and effective magnitude-based pruning functions on individual layers (subsection 7.3.2). Then, we extend this methodology so that the per-layer sparsities can also be discovered automatically by introducing a novel adaptive sparsity loss (subsection 7.3.2).

For pruning individual layers, we explore two cost-effective magnitude-based threshold functions: the first sets the pruning threshold using a binary search algorithm, while the second uses a confidence interval analysis under a Gaussian assumption for the distribution of weights. Both variations aim to retain a certain predefined sparsity level. Motivated by previous work [68, 27], we also allow pruned connections to be recovered in subsequent iterations. The key enabling factor to fast and efficient recovery of previously pruned connections is a novel application of the Straight Through Estimator (STE) [18], which is used to update weights during the back-propagation step. STE enables weight re-use while preserving a bell-shaped weight distribution, which is crucial for effectively removing

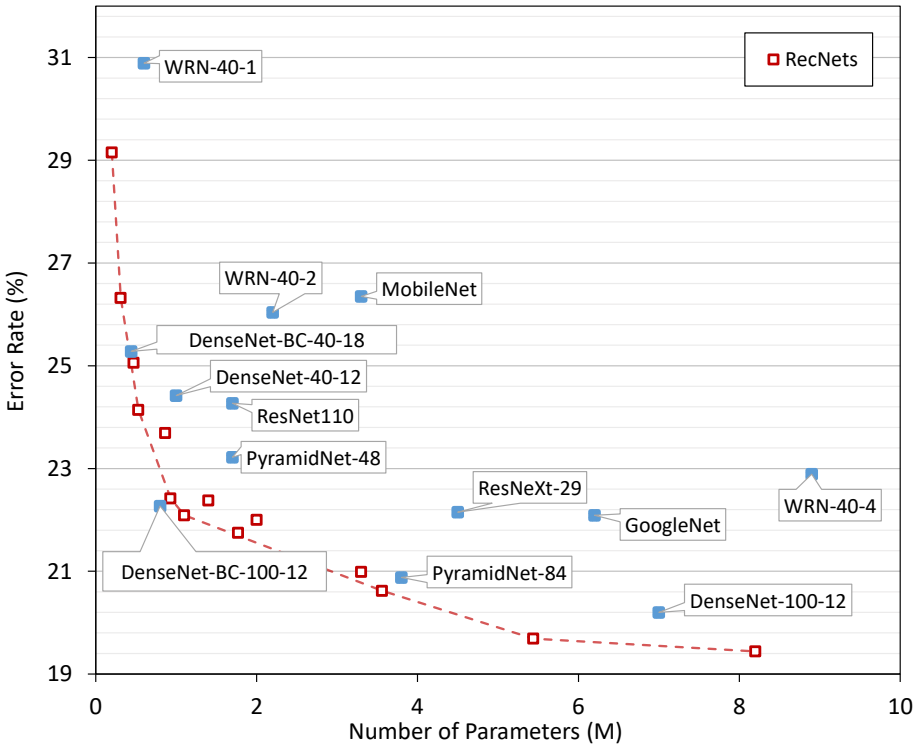


Figure 7.4: Comparison of state-of-the-art networks on CIFAR-100 in terms of size-accuracy trade-off.

connections using the considered pruning functions.

Unlike previous methods that rely on sparsity inducing losses (e.g. L^1 penalty [71]), we propose an intuitive sparsity controlling loss. Specifically, under a Gaussian assumption for the weight distributions, we can formulate the sparsity level of each layer as the error function (erf) depending on the threshold value. This function is differentiable and, thus, back-propagatable. Therefore, we can define the overall sparsity of the network by combining the per-layer sparsities into an extra loss function to be optimized along with the task-related loss. Eventually, each layer's threshold is a trainable parameter controlled by the sparsity function and we can combine sparsities into a budget-aware fashion, enforcing an overall budget on the number of parameters or operations and granting the method freedom to prune layers that are more redundant more aggressively.

Overall, the proposed pruning techniques have several compelling properties:

- *Trainable sparsity*: optimal per-layer sparsities are automatically determined under a user-defined budget constraint.
- *Versatility*: out methods are robust to different layer types (convolutional, fully connected etc.) and architectures. Moreover, the proposed adaptive sparsity loss can be easily modified to match the user's needs.
- *Minimal computational overhead:* efficient pruning operations without any convergence overhead.

7.3.1 Online Weight Pruning

Preliminaries

Given a layer comprising a weight tensor $\mathbf{W}$ and a bias tensor $\mathbf{b}$, we focus on sparsifying $\mathbf{W}$, since the bias term has a minor contribution to its budget. Pruning individual weights directly affects the size of the layer (and consequently the model) and the number of floating-point operations required to infer. Specifically, both layer characteristics (size and required operations) are scaled linearly by the density percentage, i.e. $1 - s$ if s is the layer's sparsity. In the following, we assume a neural network consisting of N such layers and even though the notation describes convolutional layers, it can be straightforardly apply to any type of trainable layer.

Magnitude-based Pruning

Using magnitudes to determine the saliency of connections in neural networks is prominent in both structured and unstructured pruning techniques. The reasoning is simple and intuitive: a small magnitude indicates a small contribution to the output. For the case of weight (unstructured) pruning, the simplest approach is to remove individual weights w according to their absolute value given a bound b as follows:

$$f_{prune}(w; b) = \begin{cases} 0, & \text{if } |w| < b \\ w, & \text{otherwise} \end{cases} \tag{7.9}$$

Despite the simplicity of this approach, it has been shown to be effective, leading to significant compression rates [68]. Key to its effectiveness is the assumption of a unimodal unknown weight distribution, centered at zero. In practice, weight decay [106], which is typically used in training, imposes the minimization of the parameters' values or, in other words, the concentration of the parameters' values around zero, and, thus, favors this assumption. Indeed, Figure 7.5 verifies this assumption for layers of a fully-trained ResNet50 model. For the sake of brevity, we henceforth assume a zero-mean weight distribution per layer. Nonetheless, the following analysis can be easily adapted for a non-zero mean without loss of generality.

Following the pruning condition of Equation 7.9, we formulate the problem of weight pruning as selecting the appropriate bound b per layer to achieve a target sparsity level with minimal accuracy loss. In online pruning, imposing a fixed value for the bound is destined to fail since the weights are continuously updated and, eventually, the vast majority will end up outside the bounded interval, resulting in an almost entirely dense layer. Therefore, dynamic bounds need to be considered. Specifically, the bound b should depend on the weights $\mathbf{W}$, i.e. $b = f_b(\mathbf{W})$. In the following sections, we use such dynamic bounds in order to develop efficient weight pruning approaches under different constraints.

7.3.2 Fixed Sparsity

First, we aim to achieve a predefined level of sparsity per layer. Nonetheless, it is straightforward to define an overall sparsity for the neural network by requesting the same sparsity level for every layer. Following the discussed hypothesis of a unimodal weight distribution, given a predefined sparsity level s, we seek to find an appropriate bound function $f_b(\mathbf{W}; s)$. We follow two approaches for this task: (a) find b using a binary search algorithm and (b) assume a normal distribution over the weights and find b using a confidence interval analysis. The trade-off between the two approaches lies in the sparsity precision versus

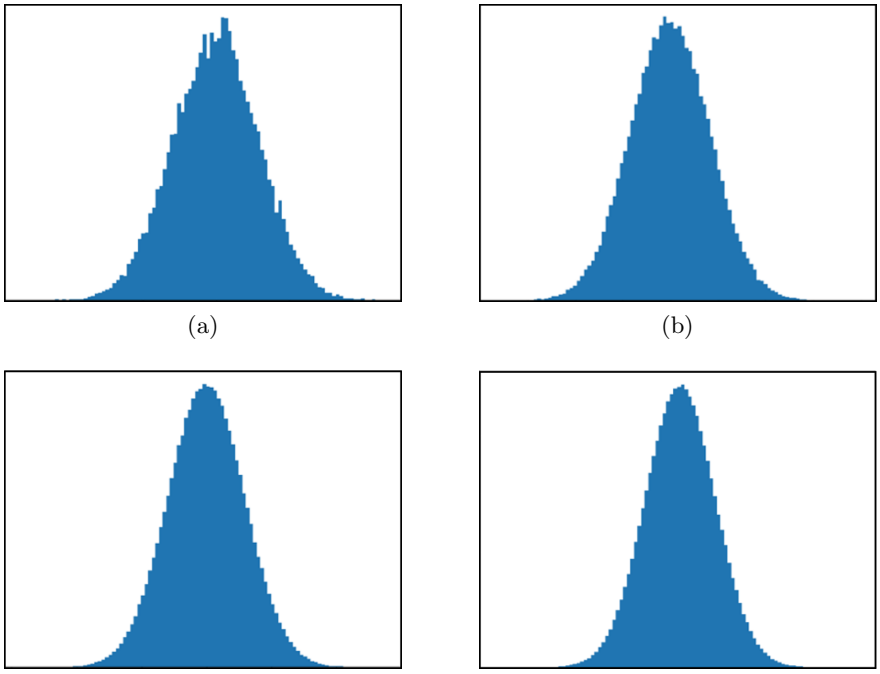


Figure 7.5: Weight distribution of the last convolutional layer in each block of a fully-trained ResNet50 architecture.

the computational overhead. Specifically, the first approach is more precise but more expensive, while the second is faster to compute but is based on a rather weak assumption (normal distribution) that may lead to deviations from the requested sparsity level.

Binary Search Given an approximation margin ϵ and a sparsity level s, the problem of finding an appropriate bound b can be formulated as follows:

$$\left| \frac{\#\{|w| < b \,|\, w \in \mathbf{W}\}}{\#\mathbf{W}} - s \right| < \epsilon \tag{7.10}$$

The above problem formulation can be efficiently solved w.r.t. b via binary search. At each step of the binary search algorithm we need to re-estimate the bound b and, thus, we have to count the newly pruned weights. The total number of steps depends on the approximation parameter e. Contrary to using a sorting algorithm for this task, the binary search approach provides a faster convergence to bound b under the approximation error ϵ, which is crucial when considering layers comprising millions of parameters.

Gaussian Assumption The per-layer weight distribution often resembles a Gaussian bell curve (e.g., see Figure 7.5). This observation leads to the idea of selecting a statistical-related pruning bound according to a confidence interval analysis, by requesting a confidence level equal to the target sparsity level, as shown in Figure 7.6. Similar statistical approximation of b has been previously examined at [187] by introducing a sparsity level controlling factor a and computing the bound according to: $b = a \times \text{std}(\text{vec}(\mathbf{W}))$. Nonetheless, the authors of [187] do not explore the ability to precisely define sparsity via an appropriate a based on confidence intervals.

More formally, assuming a layer's weights follow a zero-centered normal distribution, i.e. $w \sim N(0, \sigma)$, and given a desired level of sparsity $s \in (0, 1)$, the pruning bound b can

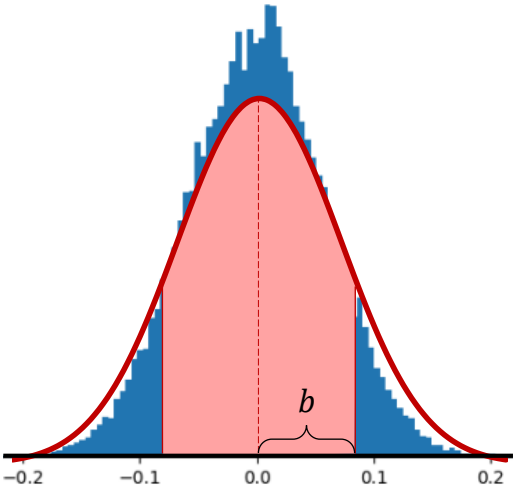


Figure 7.6: Example of a Gaussian bell over the real distribution of weights (layer2_1 of ResNet50).

be computed as:

$$b = f_b(\sigma; s) = \sigma\sqrt{2}\,\mathrm{erf}^{-1}(s) \tag{7.11}$$

$$\text{where} \quad \mathrm{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} dt \tag{7.12}$$

In practice, we only need to compute the standard deviation σ of the weights in order to find an appropriate b.

It is important to note that the Gaussian assumption is fairly weak (see Figure 7.6). Therefore, it is possible to encounter deviations between the requested and attained sparsity levels. If the sparsity level exceeds the target, there may be a negative impact on the attainable accuracy. If it is lower, it may violate the user's requirements.

Training using a Straight-Through Estimator The efficiency of the proposed methodology relies on a simple, yet powerful concept, referred to as Straight-Through Estimator (STE) [18]. STE was initially motivated by the need for a differentiable function for threshold operations in order to properly train a network. To overcome this problem, Hinton et al. [18] proposed to back-propagate through such hard threshold functions as if it had been the identity function (hence the straight-through term). Considering the problem at hand, the main idea is to update all weights, as if they actively contribute to the layer output, via the back-propagation scheme of STE. However, during the forward propagation, we only use the sparse subset of weights in order to generate the layer output. In other words, forward and backward propagation steps (with respect to the weight tensor) are mutually inconsistent. This inconsistency is very helpful; the network is trained according to the sparsity constraints, while simultaneously all weights, even if they are pruned, are updated in order to optimize the layer output. In this way, an already pruned weight may return to the used weight set without any complications since it has already been trained all along.

Let $\mathbf{X}$ and $\mathbf{Y}$ be the input and output of a layer, respectively, $\mathbf{W}$ be the weight tensor of the layer and $\widetilde{\mathbf{W}} = f_{prune}(\mathbf{W})$ be the pruned weight tensor. Given a loss function L at the output of the network, the aforementioned approach for a given convolutional layer is formulated as follows ($*$ and $*^\intercal$ are the convolution and the transposed convolution operations, respectively):

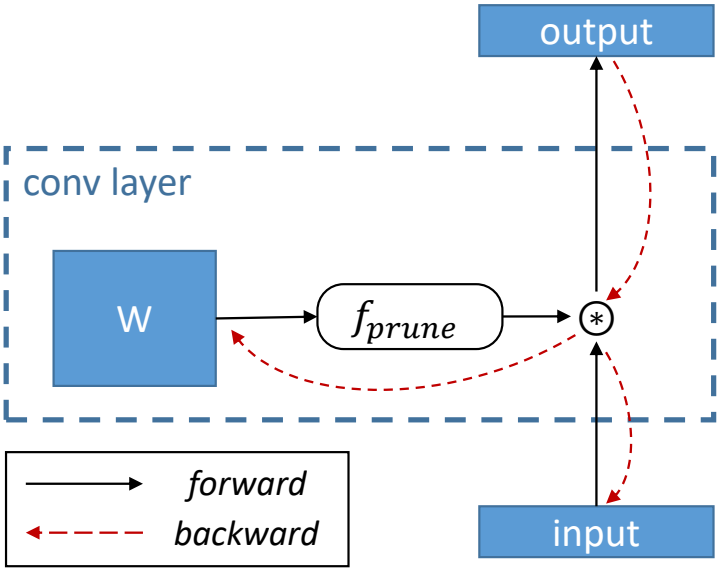


Figure 7.7: Online pruning using Straight-Through Estimator for updating weights.

- *Forward:* $\mathbf{Y} = \mathbf{X} * \widetilde{\mathbf{W}}$
- *Backward:* $\frac{\vartheta L}{\vartheta \mathbf{X}} = \frac{\vartheta L}{\vartheta \mathbf{Y}} *^{\mathsf{T}} \widetilde{\mathbf{W}}\,, \quad \frac{\vartheta L}{\vartheta \mathbf{W}} = \mathbf{X} * \frac{\vartheta L}{\vartheta \mathbf{Y}}$
- *Update:* $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\vartheta L}{\vartheta \mathbf{W}}$

The functionality of STE in the context of our pruning framework is depicted in Figure 7.7.

The main motivation behind using STE in our work is to enable weight re-use and preserve a bell-shaped distribution, as opposed to the initial motivation of meaningful derivatives. Indeed, the hard-shrink function of Eq. 7.9 can be back-propagated without complications. Nevertheless, back-propagating through this function would exclude the pruned weights from the update step, resulting in non-unimodal distributions. An example of this phenomenon is depicted in Figure 7.8a, where the non-pruned weights form two small areas around the main pruned area. On the contrary, using STE leads to a well-formed bell-shaped weight distribution as Figure 7.8b suggests. Therefore, STE is crucial to the efficiency of the second bound selection approach (see Section 7.3.2) which makes a Gaussian assumption of the distribution of weights. Updating the entire weight tensor for dynamic weight re-usage is not a new concept (e.g. [68]), even though it has not been formulated as a use case of STE before and, more importantly, it has not been connected to the distribution preservation key property.

Apart from STE, we also adopt the L^2 regularization term, also referred to as weight decay [106], which has been shown to provide solutions that generalize well. Overall, the use of STE along with weight decay aims to assist the convergence on a subset of weights with good generalization properties and a bell-shaped distribution, while the pruning function is responsible for sparsifying the network. A key advantage of this approach is that it enables low-overhead online pruning, as opposed to the typical three-stage pipeline consisting of first fully-training the reference network, then pruning and fine-tuning to recover accuracy[1] [4, 78, 126, 79].

[1]The last two steps may be performed once, often referred to as *one-shot* pruning, or repeated, often referred to as *iterative* pruning.

(a) w/o STE (b) w/ STE

Figure 7.8: Examples of training a sparse layer for 20 epochs (a) without STE and (b) with STE. We used the statistical (confidence intervals) bound selection.

Adaptive Sparsity

The pruning methods presented in the previous section require the user to define the level of sparsity per layer. While a straightforward solution would be to request the same sparsity per layer, it may be difficult to achieve the desired compression/accuracy trade-off. To overcome this issue, in this section we propose a novel adaptive sparsity loss, capable of automatically pruning a network with variable sparsity per layer given an overall sparsity constraint. In order to define such a loss, we need a differentiable loss function. From the previously proposed approaches for fixed sparsity pruning, the binary search variation can not be described by an analytical differentiable function. On the other hand, we can use the Gaussian variation to back propagate a sparsity loss. Similarly to fixed sparsity pruning, training with a sparsity loss heavily relies on STE for preserving the desired properties of the weights' distribution and retaining a competitive convergence rate.

Sparsity percentage as a loss function: As the analysis of 7.3.2 hints, the sparsity percentage s of a layer can be defined as a function of the adaptive threshold b and the standard deviation of weights σ as follows:

$$s = \text{erf}(\frac{b}{\sigma\sqrt{2}}) \tag{7.13}$$

The key observation here is that the erf() function is differentiable and can be used to define a simple yet intuitive extra loss term with the aim to find the best possible threshold b that maximizes the sparsity (or minimizes the density) of a particular layer. To this end, a straightforward solution is to define the average network density as the following loss:

$$\begin{aligned} L_s(\{b_i\}) &= \frac{1}{N}\sum_{i}^{N}(1 - s_i) \\ &= 1 - \frac{1}{N}\sum_{i}^{N}\text{erf}\Big(\frac{b_i}{\sigma_i\sqrt{2}}\Big) \end{aligned} \tag{7.14}$$

At the same time, we aim to preserve the network accuracy, thus the overall loss is defined as a multitask problem:

$$L_{CE}(\{\mathbf{W}_i\}, \{b_i\}) + \lambda L_s(\{b_i\}) \tag{7.15}$$

where $\mathbf{W}_i$ is the weight tensor and b_i the boundary of the i^{th} layer. The term L_{CE} corresponds to typical classification problems, requiring the cross entropy loss. Nevertheless,

our approach is independent of the main loss and can be used with any architecture and loss function.

As Eq. 7.15 suggests, we minimize the loss with respect to both the set of weight tensors $\{\mathbf{W}_i\}$ and the set of boundaries $\{b_i\}$. This means that for every layer we add a single parameter b_i, resulting in a trivial parameter overhead (N in total). The parameter b_i, apart from contributing to the sparsity loss, affects the hard-shrink operation that is responsible for weight pruning[2]. Therefore, if we optimize freely over the $L_{CE}(\{\mathbf{W}_i\}, \{b_i\})$ term, i.e. without the sparsity constraint, the boundaries $\{b_i\}$ would be minimized in order to leave the majority of weights un-pruned and subsequently achieve high accuracy. In other words, the L_{CE} and L_s terms act in a competitive manner. The user defined hyper-parameter λ controls the contribution of the sparsity loss term. Note that for a set of tasks (e.g. find optimum sparsity with minimal impact on accuracy) it is easier to set the sparsity loss contribution than the level of sparsity.

Even though we defined the loss function with respect to both boundary b and standard deviation σ, we did not take into account that σ is also a function of the layer's weights, i.e. $\sigma = f(\mathbf{W})$, and thus, theoretically, we can also back-propagate the derived error towards the weights. However, we choose not to propagate through σ to avoid adding extra gradient flows that would over-complicate training.

Fixed Sparsity Alternative: A drawback of the fixed sparsity pruning strategy described in Section 7.3.2 is that enforcing a high sparsity constraint may lead to a slightly decreased convergence rate, since we "discard" the majority of parameters at the beginning of the learning process. A better alternative would be to progressively reach the target sparsity. To this end, we can use the adaptive sparsity loss methodology by replacing L_s in Eq. 7.15 with the following loss term:

$$L_s(\{b_i\}) = \frac{1}{N}\sum_{i}^{N}\Big[\text{erf}\Big(\frac{b_i}{\sigma_i\sqrt{2}}\Big) - s_t\Big]^2 \tag{7.16}$$

Budget-Constrained Adaptive Sparsity: So far we have assumed that we want to equally prune (percentage-wise) every layer. However, this is not always the case. We may want, for instance, to minimize the total number of weights, in which case layers with more parameters should be pruned more aggressively. This concept can be easily translated into a loss of weighted summation of the erf() functions, i.e.:

$$L_s(\{b_i\}; \{c_i\}) = 1 - \frac{1}{N}\sum_{i}^{N} c_i \text{erf}\Big(\frac{b_i}{\sigma_i\sqrt{2}}\Big) \tag{7.17}$$

where c_i the per layer contribution weight. For example, in order to minimize the number of parameters in the network, we could define the contribution weights as:

$$c_i = \frac{\#\mathbf{W}_i}{\sum_j \#\mathbf{W}_j} \tag{7.18}$$

Similarly, we can define contribution weights with respect to flops per layer in order to construct a flops-centered minimization loss.

Finally, we also examine the case of budget-constrained optimization by transforming the overall loss as follows:

[2]In practice, at each layer, we formulate the pruning operation as: $\widetilde{\mathbf{W_i}} = b_i \cdot \sigma_i \cdot f_{prune}(\mathbf{W_i}/(b_i \cdot \sigma_i); 1)$.

$$L_{CE}(\{\mathbf{W}_i\},\{b_i\}) + \lambda_p \|L_s(\{b_i\};\{c_{p,i}\}) - B_p\|^2 + \lambda_f \|L_s(\{b_i\};\{c_{f,i}\}) - B_f\|^2 \tag{7.19}$$

where B_p and B_f are the budget percentage for the number of parameters and flops respectively, $c_{p,i}$ and $c_{f,i}$ are the contribution weights per layer (defined as in Eq. 7.18), while λ_p and λ_f are user-defined hyper-parameters.

Possible Extensions: The proposed adaptive sparsity loss is modular and can, in fact, be combined with any task-specific loss and neural network type, e.g. convolutional, recurrent, etc. Apart from using any type of network and task, the user can also formulate the sparsity constraints as an extra loss according to the problem at hand. For example, one may require to either have very sparse or very dense layers, while approximating an overall target sparsity s_t, i.e. $\sum_i^N s_i \simeq s_t$, while $s_i < s_l$ or $s_i > s_u$, where s_l and s_u are the lower and upper thresholds respectively.

7.3.3 Back-propagating through STE with Adaptive Sparsity Loss: Gradient Derivation

Adaptive sparsity loss relies on a trainable parameter b (per-layer), which denotes the bound of a magnitude-based pruning operation. The task-related loss is affected by these bounds even if we do not add the proposed sparsity loss ($\lambda = 0$), i.e. there exists a gradient flow from the task-related loss to the bounds. Intuitively, if no sparsity controlling loss exists, the bounds, and consequently the per-layer sparsity, should be minimized in order to use as much weights as possible, unhindered. In this section, we will show the effect of STE to a bound parameter b by deriving its corresponding gradient.

First, we re-define the pruning threshold function f_{prune} with a fixed unity boundary as:

$$T(x) = \begin{cases} 0, & \text{if } |x| < 1 \\ x, & \text{otherwise} \end{cases} \tag{7.20}$$

We denote the vectorized version of the weights and the pruned weights as $\text{vec}(\mathbf{W}) = [w_0, w_1, \dots w_{n-1}]$ and $\text{vec}(\widetilde{\mathbf{W}}) = [\widetilde{w}_0, \widetilde{w}_1, \dots \widetilde{w}_{n-1}]$, respectively. Then, using the Gaussian Assumption approach (given a zero mean value), a weight w_i is pruned according to:

$$\widetilde{w}_i = \begin{cases} 0, & \text{if } |w_i| < b\sigma \\ w_i, & \text{otherwise} \end{cases} \tag{7.21}$$

Or, equivalently, using $T(\cdot)$:

$$\widetilde{w}_i = b\sigma T\Big(\frac{w_i}{b\sigma}\Big) \tag{7.22}$$

Both $\mathbf{W}$ and b are trainable parameters and therefore we need to define their respective gradients. To compute the task-related loss (L) gradient w.r.t. the pruned weights ($\vartheta L/\vartheta\widetilde{\mathbf{W}}$), we follow the typical formulation of neural networks and their back-propagation process. Nevertheless, in order to fully describe the training process of the proposed work, we should define the derivatives $\vartheta\widetilde{w}_i/\vartheta w_i$ and $\vartheta\widetilde{w}_i/\vartheta b$. Note that STE is applied on the thresholding function $T(x)$, i.e. we assume the identity function during the backpropagation.

The first case of $\vartheta\widetilde{w}_i/\vartheta w_i$ is rather straightforward:

$$\frac{\vartheta\widetilde{w}_i}{\vartheta w_i} = \frac{\vartheta\Big(b\sigma T\Big(\frac{w_i}{b\sigma}\Big)\Big)}{\vartheta w_i} \overset{STE}{=} \frac{b\sigma\vartheta\Big(\frac{w_i}{b\sigma}\Big)}{\vartheta w_i} = 1 \tag{7.23}$$

This is inline with the main functionality of STE, which passes through the gradient to the weights as if no threshold operation was applied. However, the gradient of a pruned weight $\widetilde{w}_i$ with respect to the single extra parameter b, that control sparsity, has not such simple and intuitive interpretation. Specifically, the gradient is computed as:

$$\frac{\vartheta \widetilde{w}_i}{\vartheta b} = \frac{\vartheta\Big(b\sigma T\Big(\frac{w_i}{b\sigma}\Big)\Big)}{\vartheta b} = \sigma T\Big(\frac{w_i}{b\sigma}\Big) + b\sigma\frac{\vartheta T\Big(\frac{w_i}{b\sigma}\Big)}{\vartheta b}$$

$$\overset{STE}{=} \sigma T\Big(\frac{w_i}{b\sigma}\Big) + b\sigma\frac{\vartheta\Big(\frac{w_i}{b\sigma}\Big)}{\vartheta b} = \frac{1}{b}\widetilde{w}_i - \frac{1}{b}w_i \tag{7.24}$$

Overall, the loss gradient with respect to b is computed as:

$$\frac{\vartheta L}{\vartheta b} = \sum_{i=0}^{n-1}\Big(\frac{\widetilde{w}_i - w_i}{b}\Big)\frac{\vartheta L}{\vartheta \widetilde{w}_i} \tag{7.25}$$

The derived gradient has some notable properties:

- The gradient is minimized when either the majority of the weights is unpruned (which corresponds to a low bound b) or a well-performing model is found and no further action is needed ($\vartheta L/\vartheta \widetilde{w}_i \to 0$). Note that only pruned weights ($\widetilde{w}_i = 0$) contribute to the gradient.
- If STE was not applied, the gradient would be zero. This behavior is not desirable for the proposed sparsity controlling problem. In fact, the bound minimization property under the absence of a sparsity inducing loss is very intuitive and in line with the abstract idea of a robust sparsification approach.

Regarding the sparsity loss, its gradient (for a single layer) $L_s = 1 - \text{erf}(b/(\sigma\sqrt{2}))$ with respect to b is:

$$\frac{\vartheta L_s}{\vartheta b} = -\frac{2e^{-b^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}} \tag{7.26}$$

7.3.4 Experimental Evaluation

In this section, we evaluate the proposed pruning methods for the task of image classification on two popular datasets, namely CIFAR-100 [104] and ImageNet [154]. Although CIFAR-10 is also a popular choice, we see no merit in exploring such an extreme case of over-parameterization, as demonstrated in Figure 7.1. The considered architectures are the widely-used ResNets [77] and Wide ResNets [192], as well as AlexNet, which is commonly used in the literature to evaluate weight pruning methods.

CIFAR-100 Exploration

We evaluate the proposed methods on CIFAR-100 using the Wide ResNet architecture and specifically the WRN-16-8 model (∼11 million parameters). We employ a SGD optimizer for 120 epochs and a cosine annealing scheduling with restarts every 40 epochs [116].

First, we define the possible pruning variations, presented in this work using the following acronyms:

$$CL - \{BS|GA\} - \{LW|NW\} - \{F|A\} - \{b|\lambda\}$$

Method	#Params (Million)	Sparsity (%)	Acc. (%)
WRN-16-8 (reference model)	11.012	0	78.52
WRN-16-3 (equivalent dense model)	1.672	85.00	74.89
CL-BS-LW-F-b	1.719	84.90	78.66
CL-GA-LW-F-b	1.588	86.09	78.45
CL-BS-LW-A-b	1.596	86.02	78.70
CL-BS-NW-A-b-avg	2.559	77.21	78.62
CL-BS-NW-A-b-wavg	1.655	85.45	78.55

Table 7.9: Budget-aware pruning approaches under 85% target sparsity constraint.

where *CL* stands for *Compressive Learning*, *BS* and *GA* stand for the *Binary Search* and *Gaussian Assumption* pruning functions respectively, *LW* and *NW* distinguish the *Layer-Wise* or *Network-Wise* focus, while *F* and *A* stand for the *Fixed* and *Adaptive* pruning alternatives. The last term denotes if the user-defined hyper-parameter that controls the overall sparsity is either a specific target *budget* (*b*) or the λ parameter of Equation 7.15. Note that not every combination is valid (e.g. binary search is not compatible with adaptive sparsity loss).

Briefly, *CL-BS-LW-F-b* and *CL-GA-LW-F-b* refer to the binary search and Gaussian assumption approaches of fixed pre-defined per layer sparsity (Section 7.3.2), while *CL-GA-LW-A-b* refers to the relaxed per layer constraint, defined by Eq. 7.16. For the case of adaptive overall sparsity under a budget constraint, *CL-GA-NW-A-b*, we also distinguish the average sparsity approach (*avg*) of Eq. 7.14 and the weighted average alternative (*wavg*) of Eq. 7.17. We evaluate the case of budget-constrained optimization with an overall requested sparsity set to 85% at Table 7.9, reporting performance of both the initial WRN-16-8 and the compressed dense equivalent WRN-16-3.

Notably, even if a considerable amount of parameters are pruned (nearly 10 million), no accuracy loss is observed, contrary to the equivalent dense model of the same size, while all the reported pruning approaches have similar performance to the initial over-redundant model. Moreover, the binary search approach presents the best approximation of the target sparsity, confirming its superior precision compared to the Gaussian assumption alternative. Nevertheless, both Gaussian assumption alternatives display a minor sparsity divergence which is not important when considering practical budget-constrained applications. More importantly, we can conclude that the average sparsity constraint (*CL-GA-NW-A-b-avg*) is not appropriate for budget-constrained tasks, since it does not take into account the difference in number of parameters between different layers and results to an overall sparsity of 77.21%. On the contrary the weighted average alternative is built to address this problem and thus generates a precise overall sparsity (*wavg* would be considered as the default adaptive pruning approach for the rest of the paper).

The effect of these methods on the per layer sparsity distribution is presented at Figure 7.9. As expected, the binary search approach achieves the target sparsity for all layers. Both GA layer-wise variants have the same behavior, similar to binary search yet less precise, while the network-wise adaptive variants display significant fluctuations. Specifically, the average sparsity alternative displays comparatively low sparsity at the last conv layers, which leads to the decrease of the overall sparsity, whilst weighted average alternative displays high compression on the last layers and notably low at shortcut layers. This result is aligned with our initial goal of relaxed, adaptive sparsification per layer.

Regarding the convergence rate of our methods, we report the evolution of the evaluated model's accuracy over training in Figure 7.10. We can observe that, indeed, the

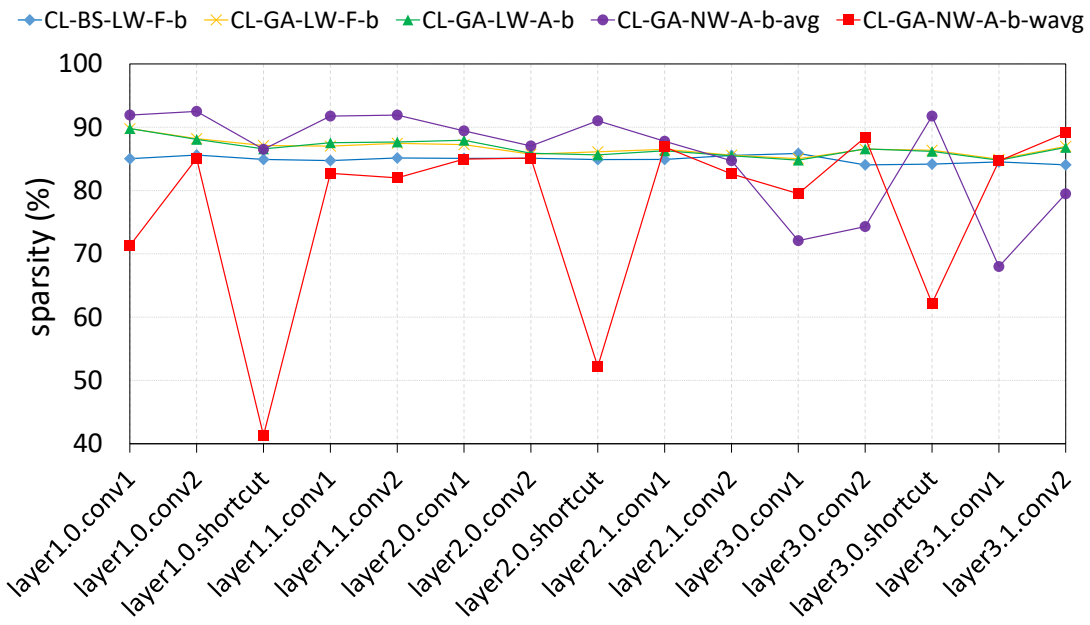


Figure 7.9: Actual per layer sparsity for WRN-16-8 under the constraint of 85% overall sparsity.

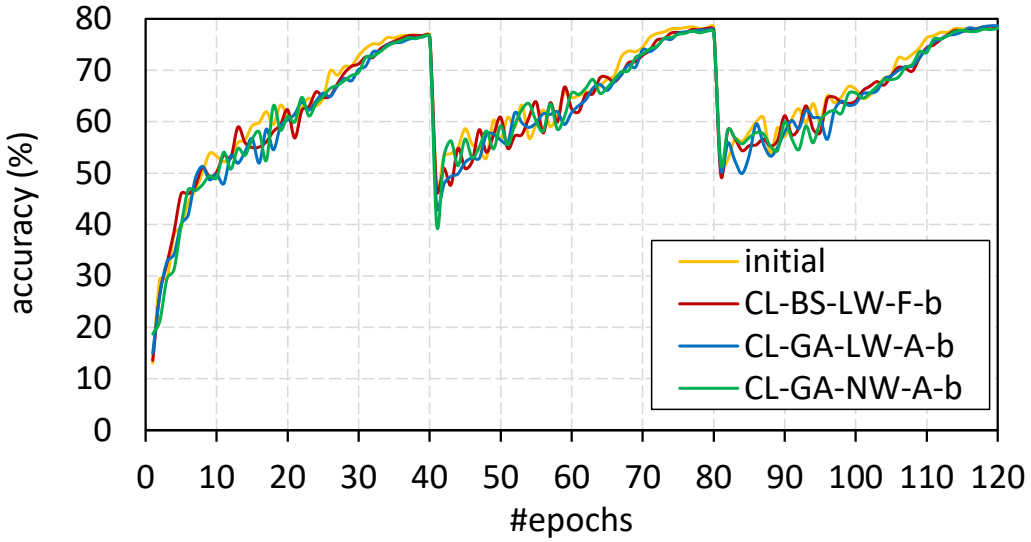


Figure 7.10: Convergence of the different pruning alternatives under the constraint of 85% overall sparsity.

convergence of the proposed methods is akin to that of the initial dense uncompressed network, as we have already advocated in the analysis of Section 7.3.1.

To thoroughly evaluate the proposed methods, we use the analysis of Section 7.1.4 to generate the error vs number of parameters plot for both the *CL-GA-LW-F-b* and *CL-GA-NW-A-b* approaches, while we also consider the non-budget constrained alternative of weighed average approach (Eq. 7.17), referred to as *CL-GA-NW-A-b-*λ (we select a set of λ sampled in $[0.01, 10.0]$). The generated curves are shown at Figure 7.11, where we can notice the effectiveness of the similarly performing proposed methods.

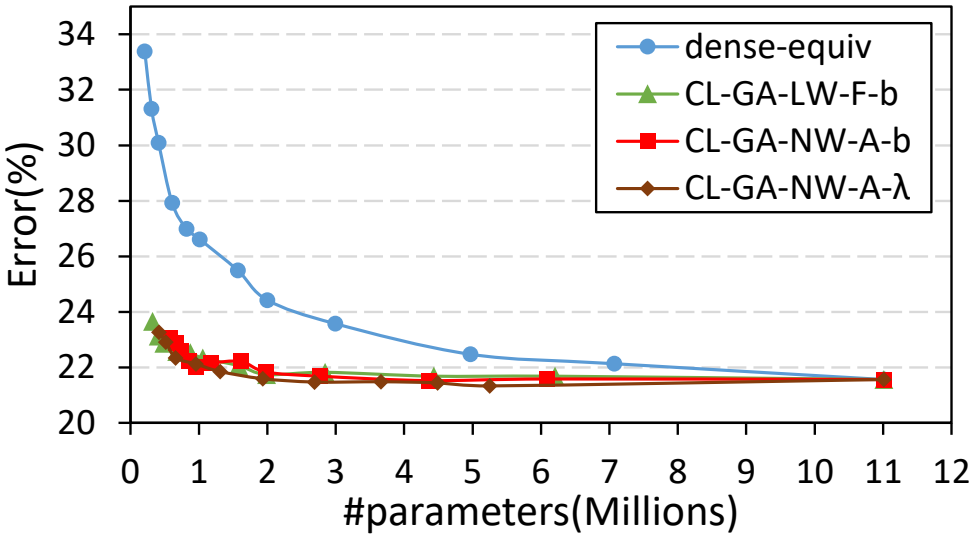


Figure 7.11: Trade-off curves for the proposed pruning approaches and the dense equivalent models (WRN-16-8 / CIFAR-100).

Method	#Params (Million)	Top-1 Acc. (%)	Top-5 Acc. (%)
AlexNet reference	61.10	56.58	79.88
LWC [71]	6.71	57.20	80.30
L-OBS [42]	6.71	56.89	79.99
DNS [68]	3.48	56.91	80.01
Constraint-aware [30]	2.97	54.84	-
ADMM [196]	2.90	-	80.20
FDNP [114]	2.90	56.84	80.02
BA-FDNP [114]	2.70	56.82	79.96
CL-GA-NW-A-b-λ	2.65	56.62	79.64

Table 7.10: Comparison of several unstructured pruning approaches on AlexNet/ImageNet setting.

ImageNet

Due to resource limitations, experiments on ImageNet were performed by sparsifying existing pre-trained networks[3] for only 20 epochs. First, we apply the unconstrained variation (*CL-GA-NW-A-b-*λ) with $\lambda = 10$ on AlexNet with the aim to minimize the overall size of the network. AlexNet consists of 8 layers (5 convolutional and 3 fully-connected) with high variability in size, which is ideal to showcase the value of adaptive pruning approaches. Table 7.10 compares the results of our method with existing unstructured pruning techniques and shows we achieve significant compression with no accuracy loss (on par with the state-of-the-art frequency domain pruning approach BA-FDNP [114]).

Next, we consider the case of ResNet50 for the proposed fixed and adaptive budget-constraint variants, as shown at Figure 7.12. The dense equivalent models for the trade-off curve were trained from scratch for 60 epochs, which is not optimal, but provides a credible accuracy vs compression trend. The behavior of the fixed sparsity approach, i.e. similar performance to the dense equivalent for very small budgets, confirms our initial observation in Section 7.3.2; assuming a very high sparsity percentage from the start may lead to slow convergence. This, however, is not the case for the adaptive alternative, highlighting

[3]taken from PyTorch model zoo

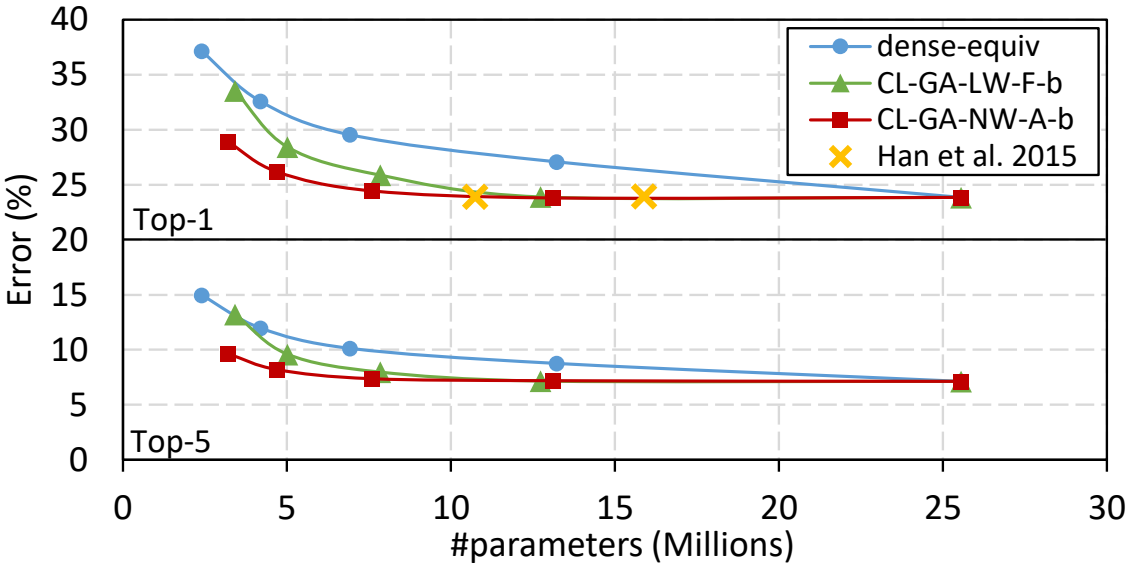


Figure 7.12: Trade-off curves for the proposed pruning approaches and the dense equivalent models (ResNet-50 / ImageNet).

its importance. Note that, based on the formed trade-off trends of Figure 7.12, we can increase the overall sparsity up to 70% (∼7.5M parameters, i.e. smaller than the dense ResNet18 architecture) without any significant accuracy loss.

7.4 KWS using Compressed Models

Having proposed two vastly different approaches to obtain compact neural networks, we revisit the keyword spotting problem. Out of the two aforementioned approaches, we choose the pruning one, since it is significantly more robust to different settings and losses and can converge notably fast. Moreover, the proposed pruning, which in fact is a sparsity inducing technique, has been described as easy-to-apply modular approach, enforcing sparsity by an extra loss term. The case of KWS is an ideal use case in order to validate this task-agnostic sparsity loss.

As KWS reference, we use the WSRNet, proposed at Section 6.4, which can perform both spotting and recognition. The WSRNet case is ideal for evaluating the proposed sparsity loss, since it has already been form as a multi-task optimization (jointly minimizing the loss of the CTC and Seq2Seq components). The addition of the sparsity loss, would be an extra term of the overall loss, resulting to a more complex multi-task problem. The WSRNet architecture consists of several subcomponents, namely CNN backbone and the recurrent networks: CTC, Seq2Seq and Autoencoder components (for details, see 6.4). The majority of the WSRNet's parameters correspond to the CNN backbone, since the recurrent networks of the other components re-use specific sets of parameters in a recurrent fashion. Therefore, we sparsify only the CNN backbone. To further stress the capability of deep learning to generate compact descriptors while using minimal resources, we consider the case of binarized word representation (as output of the encoder module of Seq2Seq component).

The adaptive sparsity loss without a target budget (see Equation 7.17) is selected in order to prune as much weights as possible, without sacrificing performance. The selected loss assumes that each layer's sparsity is weighted proportionally to the layer's number of

parameters, but does not impose a requested overall parameter budget. Moreover, this adaptive loss term is simply added to overall multi-task loss without any scale factor.

The pre-trained models of the the dense counterparts are used at the initialization. The training scheme is exactly the same with the addition of the extra loss term. Only 20 epochs are considered, since we start from a pre-trained model.

The experimental results are summarized at Tables 7.11 and 7.12, where the sparse counterparts of the WSRNet are evaluated against state-of-the-art techniques for both the word recognition and word spotting, respectively. The sparse version uses only $0.7M$ paramaters for the CNN backbone, achieving an overall sparsity of 87.6%. The results indicate that we have minor to none performance loss for both tasks (recognition and spotting), still competing state-of-the-art approaches, even though the reported compression is notable. Furthermore, the per layer sparsity for this network is reported at Table 7.13, where we observe that deeper layers achieve higher sparsity percentages. This observation is in line with the selected loss definition (weighted summation with respect to layers' size).

Method	CER	WER
Sueiras et al. [168]	8.8	23.8
Wigington et al. [184]	6.07	19.07
Krishnan et al. [101]	6.34	16.19
Dutta et al. [44]	4.88	12.61
Proposed Models		
WSRNet+LM+Bin (dense)	5.7	15.4
WSRNet+LM+Bin+beam (dense)	5.1	15.0
WSRNet+LM+Bin (sparse)	5.6	15.4
WSRNet+LM+Bin+beam (sparse)	5.4	15.1

Table 7.11: Comparison of the state of the art for word recognition versus the binarized version of WSRNet. The greedy decoding variation versus beam search-based variation results are reported (+beam). Dense versus sparse versions of the network are compared. All figures are CER and WER percentages computed for the IAM dataset.

Method	QbE	QbS
Attributes+KCSR [9]	55.73	73.72
PHOCNet [166]	72.51	82.97
HWNet [102]	80.61	-
Triplet-CNN [185]	81.58	89.49
PHOCNet-TPP [167]	82.74	93.42
DeepEmbed [100]	84.25	91.58
Zoning Ensemble PHOCNet [148]	87.48	-
End2End Embed [101]	89.07	91.26
DeepEmbed [101]	90.38	94.04
Synth+DeepEmbed [101]	-	95.09
HWNetV2[103]	90.65	-
Proposed Models		
WSRNet+LM+Bin (dense)	91.31	93.69
WSRNet+LM+Bin (sparse)	90.93	93.68

Table 7.12: Comparison of the state of the art for keyword spotting versus the binarized version of WSRNet. Dense versus sparse versions of the network are reported. Figures are MAP percentages computed on the IAM dataset.

Layer	Sparsity
cnv0	32.53%
cnv1.conv1	24.29%
cnv1.conv2	28.2%
cnv1.shortcut.0	29.54%
cnv2.conv1	25.01%
cnv2.conv2	31.0%
cnv3.conv1	58.01%
cnv3.conv2	76.23%
cnv3.shortcut.0	24.94%
cnv4.conv1	80.53%
cnv4.conv2	82.14%
cnv5.conv1	79.27%
cnv5.conv2	81.53%
cnv6.conv1	80.48%
cnv6.conv2	82.7%
cnv7.conv1	90.0%
cnv7.conv2	94.05%
cnv7.shortcut.0	58.62%
cnv8.conv1	91.1%
cnv8.conv2	90.83%
cnv9.conv1	92.0%
cnv9.conv2	92.52%
cnv10.conv1	91.66%
cnv10.conv2	91.86%

Table 7.13: Per layer sparsity percentage for the CNN backbone of WSRNet.

Chapter 8

Conclusions

8.1 Summary of Main Contributions

This thesis can be divided into two main sections with respect to the use of deep learning, while the major theme of this work is word image representation under the keyword spotting paradigm. The first part of our work focuses on traditional feature extraction techniques, while the second address the problem of word representation from the deep learning viewpoint, providing novel ideas to further improve the existing literature. Our engagement with deep learning and its practical application to document analysis led to an extra research direction, namely neural network compression.

The main contributions of this thesis are summarized below:

- **Projections of Oriented Gradients:** We presented a novel descriptor, dubbed POG, which act as a projection-based alternative to the popular Histogram of Oriented Gradients approach. The main idea behind the proposed descriptor is to encode strokes as projections (utilizing Radon Transform). Strokes can characterize efficiently typical text image appearance. Results on character classification and keyword spotting highlight the effectiveness of POG.

- **User-assisted OCR:** We also addressed the problem of unlabeled document recognition, where neither the language nor the number of possible characters is known. In order to retrieve as much information as possible, we developed a user-assisted system, consisted of iterative re-clustering of possible characters according to users suggestions. A major concern during the development of the aforementioned system was the minimization of user interaction.

- **Learning-Free KWS:** Aiming to optimize the learning-free QbE case, we analyzed the most common variations of handwritten text-images and we efficiently address them at preprocessing (remove global rotations and translations), feature-extraction (use POG as a robust descriptor to small deformations) and matching (inner translations between characters). Note that typical feature extraction techniques are very prone to such variations, while interest point approaches are time-consuming (matching of several points introduces significant overhead). The proposed method relies on dynamic programming for fast matching and further improves computational requirements by assuming a re-ranking scheme: perform a coarse retrieval, followed by a finer one only for the top retrieved words. The developed method outperformed significantly all existing learning-free approaches.

- **Manifold Embedded Representations:** Non-linear dimensionality reduction techniques, such as manifold learning, can effectively project high-dimensional data into a low-dimensional space, where euclidean distance is much more intuitive and effective. Such dimensionality reduction can be beneficial to word representations, generated by learning-free KWS methods, which typically are descriptors of high-dimensionality. However, concerning KWS, queries are unknown beforehand and considered as out-of-sample data. Thus their projection to the embedded low-dimensional space is not trivial, especially when time constraints are important. To overcome this problem, we proposed a novel solution to out-of-sample problem for the t-SNE [182] manifold learning technique, generating projections with minor computational overhead. Evaluation over several high-dimensional descriptors (for the KWS task) showed significant retrieval improvement, even when only 5 dimensions were used.

- **Exploration of Deep Features:** Moving on to the deep learning field of research, we explored several deep learning aspects (architecture, training strategies) under the prism of keyword spotting, aiming to generate robust word image representations from Convolutional Networks. Our initial study case is the popular PHOCNet architecture [166], a modified VGG [160] architecture adjusted to the KWS problem. We proposed several modifications for speeding up the training procedure (avoid the costly pyramidal flattening operation) and improving the performance (extract deep features from intermediate layers along with manifold learning and evaluate their generalization capabilities, employ ensemble of nets).

- **Joint HTR and KWS:** Traditionally HTR and KWS tasks are treated separately, even though they are closely related tasks. Conceptually HTR is complementary to KWS, in the sense that both tasks seek meaningful visual representations developed under a different goal/target. However, in practice, either KWS is seen as an HTR system with extra constraints, or their connection is not further investigated. Similarly to a training-based word spotting approach, an HTR system can generate discriminative word representations. This idea was the main motivation behind the deep learning methodology that we developed, called WSRNet, which can simultaneously perform both spotting and recognition and achieves state-of-the-art results in both tasks. WSRNet consists of two main modules: the Encoder and The Decoder. Encoder module encodes the input image, treated as a sequence, into a fixed-sized vector. This vector is then decoded by the Decoder module into the target sequence of characters, i.e. the corresponding word text. The intermediate fixed-sized representation can be used as an effective word representation, enabling word spotting applications as a by-product. Simultaneously, this representation can be fully decoded by the Decoder module.

- **Line-level KWS:** Most existing KWS deep learning techniques assume a word-segmentation step was preceded. On the other hand, line-level spotting is governed by modified recognition techniques which perform a query-constrained decoding to compute a matching score. Contrary to these approaches, we developed a line-level KWS methodology which relies on deep features, resembling a template matching approach. The proposed approach can efficiently perform both QbE and QbS alternatives (even though QbE is not addressed at a line level in the existing literature), achieving state-of-the-art performance. Key components for the success of our method are two additional neural networks: the *character width estimator* which

serves as a scale detector in order to assert scale invariance and the *common space encoder* which projects both visual features and attributes into a common space, enabling both QbE and QbS.

- **Model Compression:** The need for computational efficient networks was often a recurring theme during this thesis. To this end, we developed two novel approaches for creating compact, yet well-performing, networks. Note that model compression is an active research area of great importance, not limited to document-related tasks. Specifically, we proposed a novel compact architecture based on the concept of sharing weights across different layers, employing a recurrent formulation akin to RNNs. Additionally, we developed a weight sparsification method, which essentially belongs to the weight pruning categorization. According to our approach, we introduced an adaptive sparsity loss, enabling budget constrained optimization, while retaining the same convergence with the initial dense network. This approach achieves state-of-the-art results across different settings and datasets, including the compression of a keyword spotting network.

8.2 Future Directions

Having explored various aspects of document recognition, we can highlight the most important research directions, which will help us build a efficient HTR/KWS system, capable of processing vast amount of documents.

8.2.1 KWS on entire documents

So far, this thesis was constrained mostly on segmented word images. In fact this problem is the most crucial one in order to evaluate various word representations. However, in practice, we do not have segmented words and we want to detect a query on the entire document image. Even if an ideal descriptor exists, the problem of segmenting the words is not resolved. Following the success on object detection, one can propose the usage of Regions of Interest (ROIs), which are image regions, typically bounding boxes, that contain the sought object with high probability. The number of ROIs is expected to be small for a successful detection system. State-of-the-art deep learning techniques [142, 75] address simultaneously the problem of object detection and object classification. Due to the nature of the text images, object detection, as performed in the literature, is not appropriate. In fact, object detection is akin to character detection, while a word is formulated by a set of these objects and depends on their spatial correlation. Based on this observation, we could seek alternative ways for detecting a small number of candidate regions taking into account the special characteristics of text images.

8.2.2 NLP-assisted Recognition and Spotting.

As we have already stated, visual cues do not suffice for a perfect recognition. In fact, humans often solve character ambiguities, which appear frequently due to writing style and merging consecutive characters into n-grams, relying on the context of the sentence. Therefore, Language Models (LMs) are extensively used in practice to enhance recognition performance. These LMs, either formed as n-grams of characters or n-grams of entire words, are created by parsing corpora and extracting their existence probability according to corpora's statistics. Nonetheless, NLP community have made significant advancements

towards creating complex contextual language models, relying on deep learning and straying from traditional statistical LMs. Modern NLP approaches, amongst others, include contextual word embeddings and missing word completing methods [138, 41]. Such approaches are in line with the desired assistance for the recognition task, providing an extra contextual information to the visual recognition. For example, we can assign a contextual score between similar words (e.g. "cat" vs "car") by utilizing the missing word completion module of BERT [41]. Words that are more contextually coherent with their sentence, would be selected. Similar arguments apply for the keyword spotting case.

8.2.3 Compression of Neural Networks.

As we have already mentioned, building deeper and wider neural networks has been the main research direction in order to further improve performance in a variety of visual recognition tasks. Therefore model compression is necessary in order to perform real-life applications on mobile embedded devices. We have already proposed two very different compression methods, based on sharing and pruning weights, respectively. Nonetheless, model compression is a vast area of research with very interesting applications. Hence we would like to further explore compression techniques, pushing weight pruning to its limits, adding quantization aspects and enhancing compression capabilities by knowledge distillation [132].

Bibliography

[1] A. Antonacopoulos and C. Casado Castilla, "Flexible text recovery from degraded typewritten historical documents," in *Proc. of International Conference on Pattern Recognition (ICPR)*, 2006.

[2] A. Antonacopoulos and D. Karatzas, "Semantics-based content extraction in typewritten historical documents," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2005.

[3] ABBYY FineReader 11.0: _urlhttp://www.abbyy.com.

[4] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-trim: Convex pruning of deep neural networks with performance guarantee," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[5] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[6] D. Aldavert, M. Rusiñol, R. Toledo, and J. Lladós, "A study of bag-of-visual-words representations for handwritten keyword spotting," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 3, pp. 223–234, 2015.

[7] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Efficient exemplar word spotting." in *Proc. of the British Machine Vision Conference (BMVC)*, 2012.

[8] ——, "Handwritten word spotting with corrected attributes," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

[9] ——, "Word spotting and recognition with embedded attributes," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, 2014.

[10] M. J. Anderson, "Distance-based tests for homogeneity of multivariate dispersions," *Biometrics*, vol. 62, no. 1, pp. 245–253, 2006.

[11] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[12] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Proc. of the European Conference in Computer Vision (ECCV)*, 2014.

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[15] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2002.

[16] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[17] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*. Citeseer, 2017.

[18] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[19] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2004.

[20] G. Beylkin, "Discrete radon transform," *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 35, no. 2, pp. 162–172, 1987.

[21] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] G. Brown, "Ensemble learning," in *Encyclopedia of Machine Learning*. Springer, 2011, pp. 312–320.

[23] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[24] H. Bunke, S. Bengio, and A. Vinciarelli, "Offline recognition of unconstrained handwritten texts using hmms and statistical language models," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 709–720, 2004.

[25] C. Neudecker and A. Tzadok, "User collaboration for improving access to historical texts," *Liber Quarterly*, vol. 2, no. 1, pp. 119–128, 2010.

[26] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] M. A. Carreira-Perpinan and Y. Idelbayev, "Learning-compression algorithms for neural net pruning," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] M. A. Carreira-Perpinán and Z. Lu, "The laplacian eigenmaps latent variable model," in *Proc. of Artificial Intelligence and Statistics*, 2007.

[29] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[30] C. Chen, F. Tung, N. Vedula, and G. Mori, "Constraint-aware deep neural network compression," in *Proc. of the European Conference in Computer Vision (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018.

[31] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[34] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[35] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[36] E. J. Crowley, G. Gray, and A. J. Storkey, "Moonshine: Distilling with cheap convolutions," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2018.

[37] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[38] A. Delmas Lascorz, P. Judd, D. M. Stuart, Z. Poulos, M. Mahmoud, S. Sharify, M. Nikolic, K. Siu, and A. Moshovos, "Bit-tactical: A software/hardware approach to exploiting value and bit sparsity in neural networks," in *Proc. of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19, 2019.

[39] M. Denil, B. Shakibi, L. Dinh, M. A. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2013.

[40] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2014.

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[42] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[43] D. E. Dudgeon, "Multidimensional digital signal processing," *Engewood Cliffs*, 1983.

[44] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Improving CNN-RNN hybrid networks for handwriting recognition," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[45] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid hmm/ann models," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 767–779, 2010.

[46] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[47] E. Fiesler, A. Choudry, and H. J. Caulfield, "Weight discretization paradigm for optical neural networks," in *Proc. of Optical Interconnections and Networks*, 1990.

[48] A. Fischer, V. Frinken, H. Bunke, and C. Suen, "Improving HMM-based keyword spotting with character language models," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2013.

[49] A. Fischer, "Handwriting recognition in historical documents," Ph.D. dissertation, Verlag nicht ermittelbar, 2012.

[50] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character hmms," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 934–942, 2012.

[51] W. N. Francis and H. Kucera, "Brown corpus," *Department of Linguistics, Brown University, Providence, Rhode Island*, vol. 1, 1964.

[52] V. Frinken, A. Fischer, and H. Bunke, "A novel word spotting algorithm using bidirectional long short-term memory neural networks," in *Proc. of IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, 2010.

[53] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 211–224, 2012.

[54] B. Gatos, A. L. Kesidis, and A. Papandreou, "Adaptive zoning features for character and word recognition," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2011.

[55] B. Gatos, N. Stamatopoulos, G. Louloudis, G. Sfikas, G. Retsinas, V. Papavassiliou, F. Sunistira, and V. Katsouros, "Grpoly-db: An old greek polytonic document image database," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[56] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou, "A survey of document image word spotting techniques," *Pattern Recognition*, vol. 68, pp. 310 – 332, 2017.

[57] A. Gisbrecht, A. Schulz, and B. Hammer, "Parametric nonlinear dimensionality reduction using kernel t-sne," *Neurocomputing*, vol. 147, pp. 71–82, 2015.

[58] A. N. Gomez, I. Zhang, K. Swersky, Y. Gal, and G. E. Hinton, "Learning sparse networks using targeted dropout," *arXiv preprint arXiv:1905.13678*, 2019.

[59] L. Gómez, M. Rusiñol, and D. Karatzas, "Lsde: Levenshtein space deep embedding for query-by-string word spotting," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[60] A. Gondimalla, N. Chesnut, M. Thottethodi, and T. N. Vijaykumar, "Sparten: A sparse tensor accelerator for convolutional neural networks," in *Proc. of the Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52, 2019.

[61] G.P. Silva and R.D. Lins, "An automatic method for enhancing character recognition in degraded historical documents," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2011.

[62] A. Graves, "Connectionist temporal classification," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, pp. 61–93.

[63] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. of the International Conference on Machine learning (ICML)*, 2006.

[64] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. of the International Conference on Machine learning (ICML)*, 2014.

[65] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2008.

[66] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 545–552.

[67] T. Gruning, G. Leifert, T. Strauss, and R. Labahn, "A robust and binarization-free approach for text line detection in historical documents," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[68] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2016.

[69] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[70] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[71] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[72] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[73] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Object instance segmentation and fine-grained localization using hypercolumns," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 627–639, 2017.

[74] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 1993, pp. 164–171.

[75] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[77] ——, "Deep residual learning for image recognition," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[78] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[79] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[80] K. Hegde, H. Asghari-Moghaddam, M. Pellauer, N. Crago, A. Jaleel, E. Solomonik, J. Emer, and C. W. Fletcher, "Extensor: An accelerator for sparse tensor algebra," in *Proc. of the Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52, 2019.

[81] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[82] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2003.

[83] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[84] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics-theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

[85] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[86] N. R. Howe, "Part-structured inkball models for one-shot handwritten word spotting," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2013.

[87] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[88] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research (JMLR)*, vol. 18, no. 1, 2017.

[89] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[90] K. Hwang and W. Sung, "Character-level incremental speech recognition with recurrent neural networks," *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[91] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, "Deep roots: Improving cnn efficiency with hierarchical filter groups," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[92] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of the International Conference on Machine learning (ICML)*, 2015.

[93] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *Proc. of the European Conference in Computer Vision (ECCV)*, 2014.

[94] S. Johansson, E. Atwell, R. Garside, and G. N. Leech, *The Tagged LOB corpus: users' manual*. Norwegian Computing Centre for the Humanities, 1986.

[95] J.P. Lewis, "Fast normalized cross-correlation," *Industrial Light & Magic*, 2000.

[96] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *Proc. of the European Conference in Computer Vision (ECCV)*, 2016.

[97] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[98] A. L. Koerich, R. Sabourin, and C. Y. Suen, "Large vocabulary off-line handwriting recognition: A survey," *Pattern Analysis & Applications*, vol. 6, no. 2, pp. 97–121, 2003.

[99] A. Kovalchuk, L. Wolf, and N. Dershowitz, "A simple and fast word spotting method," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[100] P. Krishnan, K. Dutta, and C. Jawahar, "Deep feature embedding for accurate recognition and retrieval of handwritten text," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[101] ——, "Word spotting and recognition using deep embedding," in *International Workshop on Document Analysis Systems (DAS)*, 2018.

[102] P. Krishnan and C. Jawahar, "Matching handwritten document images," in *Proc. of the European Conference in Computer Vision (ECCV)*, 2016.

[103] ——, "HWNet v2: An efficient word image representation for handwritten documents," *arXiv preprint arXiv:1802.06194*, 2018.

[104] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[105] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2012.

[106] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 1992.

[107] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648*, 2016.

[108] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[109] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 1990.

[110] J. Li, J. Sun, and R. Vuduc, "Hicoo: Hierarchical storage of sparse tensors," in *Proc. of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18, 2018.

[111] L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text line segmentation of historical documents: a survey," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 123–138, 2007.

[112] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[113] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[114] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-domain dynamic pruning for convolutional neural networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2018.

[115] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[116] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[117] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis, "Text line detection in handwritten documents," *Pattern Recognition*, vol. 41, no. 12, pp. 3758–3772, 2008.

[118] G. Louloudis, G. Sfikas, N. Stamatopoulos, and B. Gatos, "Word segmentation using the student's-t distribution," in *International Workshop on Document Analysis Systems (DAS)*, 2016.

[119] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[120] L. Lu and Y. Liang, "Spwa: An efficient sparse winograd convolutional neural networks accelerator on fpgas," in *Proc. of the Annual Design Automation Conference (DAC)*, 2018.

[121] M. Cannon, J. Hochberg and P. Kelly, "Quarc: A remarkably effective method for increasing the ocr accuracy of degraded typewritten documents," in *Proc. of Symposium on Document Image Understanding Technology*, 1999.

[122] P. Maragos, "Morphological filtering for image enhancement and feature detection," *The Image and Video Processing Handbook*, pp. 135–156, 2005.

[123] U.-V. Marti and H. Bunke, "The iam-database: an english sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 5, no. 1, pp. 39–46, 2002.

[124] J. Michael, R. Labahn, T. Grüning, and J. Zöllner, "Evaluating sequence-to-sequence models for handwritten text recognition," *arXiv preprint arXiv:1903.07377*, 2019.

[125] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proc. of the International Conference on Machine learning (ICML)*, 2017.

[126] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[127] S. Ortmanns, A. Eiden, H. Ney, and N. Coenen, "Look-ahead techniques for fast beam search," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1997.

[128] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," in *Proc. of the Annual International Symposium on Computer Architecture (ISCA)*, 2017.

[129] J. Pastor-Pellicer, S. Espana-Boquera, M. Castro-Bleda, and F. Zamora-Martinez, "A combined convolutional neural network and dynamic programming approach for text line normalization," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[130] H. Pesch, M. Hamdani, J. Forster, and H. Ney, "Analysis of preprocessing techniques for latin handwriting recognition," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2012.

[131] T. Plötz and G. A. Fink, "Markov models for offline handwriting recognition: a survey," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 4, p. 269, 2009.

[132] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[133] R. Prabhavalkar, T. N. Sainath, B. Li, K. Rao, and N. Jaitly, "An analysis of attention in sequence-to-sequence models," in *Proc. of Interspeech*, 2017.

[134] I. Pratikakis, K. Zagoris, B. Gatos, G. Louloudis, and N. Stamatopoulos, "Icfhr 2014 competition on handwritten keyword spotting (h-kws 2014)," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[135] I. Pratikakis, K. Zagoris, B. Gatos, J. Puigcerver, A. H. Toselli, and E. Vidal, "Icfhr2016 handwritten keyword spotting competition (h-kws 2016)," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[136] J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?" in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[137] J. Puigcerver, A. H. Toselli, and E. Vidal, "Icdar2015 competition on keyword spotting for handwritten documents," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[138] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://openai.com/blog/language-unsupervised/

[139] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. of the European Conference in Computer Vision (ECCV)*, 2016.

[140] T. M. Rath and R. Manmatha, "Word image matching using dynamic time warping," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[141] ——, "Word spotting for historical documents," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 139–152, 2007.

[142] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[143] G. Retsinas, B. Gatos, N. Stamatopoulos, and G. Louloudis, "Isolated character recognition using projections of oriented gradients," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[144] G. Retsinas, G. Louloudis, N. Stamatopoulos, and B. Gatos, "Keyword spotting in handwritten documents using projections of oriented gradients," in *International Workshop on Document Analysis Systems (DAS)*, 2016.

[145] G. Retsinas, G. Louloudis, N. Stamatopoulos, G. Sfikas, and B. Gatos, "An alternative deep feature approach to line level keyword spotting," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[146] G. Retsinas, G. Sfikas, and B. Gatos, "Transferable deep features for keyword spotting," in *Proc. of IWCIM (in conjuction with EUSIPCO)*, 2017.

[147] G. Retsinas, G. Sfikas, G. Louloudis, N. Stamatopoulos, and B. Gatos, "Compact deep descriptors for keyword spotting," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[148] G. Retsinas, G. Sfikas, N. Stamatopoulos, G. Louloudis, and B. Gatos, "Exploring critical aspects of cnn-based keyword spotting. a phocnet study," in *International Workshop on Document Analysis Systems (DAS)*, 2018.

[149] G. Retsinas, N. Stamatopoulos, G. Louloudis, G. Sfikas, and B. Gatos, "Nonlinear manifold embedding on keyword spotting using t-sne," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[150] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[151] J. A. Rodriguez-Serrano, F. Perronnin, and F. Meylan, "Label embedding for text recognition," in *Proc. of the British Machine Vision Conference (BMVC)*, 2013.

[152] L. Rothacker and G. A. Fink, "Segmentation-free query-by-string word spotting with bag-of-features hmms," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[153] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[154] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[155] S. Pletschacher, J. Hu and A. Antonacopoulos, "A new framework for recognition of heavily degraded characters in historical typewritten documents based on semi-supervised clustering," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2009.

[156] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[157] H. Scheidl, S. Fiel, and R. Sablatnig, "Word beam search: A connectionist temporal classification decoding algorithm," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[158] G. Sfikas, G. Retsinas, and B. Gatos, "Zoning aggregated hypercolumns for keyword spotting," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[159] A. Sharma and P. Sankar K., "Adapting off-the-shelf cnns for word spotting & recognition," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[160] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[161] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[162] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[163] G. Strang, *Linear algebra and learning from data.* Wellesley-Cambridge Press, 2019.

[164] H. Strange and R. Zwiggelaar, "A generalised solution to the out-of-sample extension problem in manifold learning," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2011.

[165] S. Sudholt and G. A. Fink, "A modified isomap approach to manifold learning in word spotting," in *Proc. of the German Conference on Pattern Recognition*, 2015.

[166] ——, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[167] ——, "Evaluating word string embeddings and loss functions for cnn-based word spotting," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[168] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez, "Offline continuous handwriting recognition using sequence to sequence neural networks," *Neurocomputing*, vol. 289, pp. 119–128, 2018.

[169] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2014.

[170] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[171] J. B. Tenenbaum, "Mapping a manifold of perceptual observations," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 1998.

[172] C. Tensmeyer and T. Martinez, "Document image binarization with fully convolutional neural networks," *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[173] A. H. Toselli, E. Vidal, V. Romero, and V. Frinken, "HMM word graph based keyword spotting in handwritten document images," *Information Sciences*, vol. 370, pp. 497–518, 2016.

[174] A. H. Toselli, V. Romero, M. Pastor, and E. Vidal, "Multimodal interactive transcription of text images," *Pattern Recognition*, vol. 43, no. 5, pp. 1814–1825, 2010.

[175] A. H. Toselli, A. Juan, J. González, I. Salvador, E. Vidal, F. Casacuberta, D. Keysers, and H. Ney, "Integrated handwriting recognition and interpretation using finite-state models," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 04, pp. 519–539, 2004.

[176] A. H. Toselli, J. Puigcerver, and E. Vidal, "Two methods to improve confidence scores for lexicon-free word spotting in handwritten text," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[177] A. H. Toselli and E. Vidal, "Fast hmm-filler approach for key word spotting in handwritten documents," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2013.

[178] G. Vamvakas, B. Gatos, S. Petridis, and N. Stamatopoulos, "An efficient feature extraction and dimensionality reduction scheme for isolated greek handwritten character recognition," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2007.

[179] G. Vamvakas, B. Gatos, and S. J. Perantonis, "Handwritten character recognition through two-stage foreground sub-sampling," *Pattern Recognition*, vol. 43, no. 8, pp. 2807–2816, 2010.

[180] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *Proc. of Artificial Intelligence and Statistics*, 2009.

[181] ——, "Accelerating t-SNE using tree-based algorithms," *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 3221–3245, 2014.

[182] L. Van Der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research (JMLR)*, vol. 9, pp. 2579–2605, 2008.

[183] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, and A. Fornés, "A novel learning-free word spotting approach based on graph representation," in *International Workshop on Document Analysis Systems (DAS)*, 2014.

[184] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, "Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[185] T. Wilkinson and A. Brun, "Semantic and verbatim word spotting using deep neural networks," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[186] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[187] X. Xu, M. S. Park, and C. Brick, "Hybrid pruning: Thinner sparse networks for fast inference on edge devices," *arXiv preprint arXiv:1811.00482*, 2018.

[188] W. You and C. Wu, "A reconfigurable accelerator for sparse convolutional neural networks," in *Proc. of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19, 2019.

[189] K. Zagoris, K. Ergina, and N. Papamarkos, "Image retrieval systems based on compact shape descriptor and relevance feedback information," *Journal of Visual Communication and Image Representation*, vol. 22, no. 5, pp. 378–390, 2011.

[190] K. Zagoris, I. Pratikakis, and B. Gatos, "Segmentation-based historical handwritten word spotting using document-specific local features," in *Proc. of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[191] ——, "Unsupervised word spotting in historical handwritten document images using document-oriented local features," *Transactions on Image Processing*, vol. 26, no. 8, pp. 4032–4041, 2017.

[192] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016.

[193] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. of the European Conference in Computer Vision (ECCV)*, 2014.

[194] S. Zeng, Y. Lin, S. Liang, J. Kang, D. Xie, Y. Shan, S. Han, Y. Wang, and H. Yang, "A fine-grained sparse accelerator for multi-precision dnn," in *Proc. of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19, 2019.

[195] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *arXiv preprint*, 2018.

[196] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proc. of the European Conference in Computer Vision (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018.

[197] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[198] X. Zhou, Z. Du, Q. Guo, S. Liu, C. Liu, C. Wang, X. Zhou, L. Li, T. Chen, and Y. Chen, "Cambricon-s: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach," in *Proc. of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-51, 2018.

[199] M. Zhu, T. Zhang, Z. Gu, and Y. Xie, "Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern gpus," in *Proc. of the Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52, 2019.

Appendix A

Glossary

AP	Average Precision
BCE	Binary Cross Entropy
BiLSTM	Bidirectional Long Short-Term Memory Network
BN	Batch Normalization
BoVW	Bag of Visual Words
CE	Cross-Entropy
CER	Character Error Rate
CNN	Convolutional Neural Network
CRC	Channel-wise Recurrent Convolution
CTC	Connectionist Temporal Classification
DNN	Deep Neural Network
DTW	Dynamic Time Warping
EM	Expectation Maximization
FC	Fully-Connected
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
HTR	Handwritten Text Recognition
kNN	k-Nearest Neighbor classifier
KWS	Keyword Spotting
LDA	Linear Discriminant Analysis
LLE	Locally-linear Embedding
LM	Language Model
LSTM	Long Short-Term Memory Network

mAP	mean Average Precision
MDS	Multi Dimensional Scaling
MLE	Maximum Likelihood Estimation
MISM	Multi-Instance Sequential Matching
MSE	Mean-Squared Error
NLP	Natural Language Processing
OCR	Optical Character Recognition
OOS	Out-of-Sample
PCA	Principal Component Analysis
PHOC	Pyramidal Histogram of Characters
POG	Projections of Oriented Gradients
QbE	Query-by-Example
QbS	Query-by-String
ROI	Region of Interest
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SIFT	Scale Invariant Feature Transform
SM	Sequential Matching
SNE	Stochastic Neighbor Embedding
SPP	Spatial Pyramid Pooling
STE	Straight-Through Estimator
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
TB	Transition Block
TPP	Temporal Pyramid Pooling
t-SNE	t-distributed Stochastic Neighbor Embedding
WER	Word Error Rate
ZAH	Zoning-Aggregated Hypercolumns

Appendix B

List of Publications

B.1 Publications in Peer-reviewed International Journals

J1 **G. Retsinas**, G. Louloudis, N. Stamatopoulos, B. Gatos, "Efficient Learning-Free Keyword Spotting", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 41, No. 7, pp. 1587-1600, June, 2018.

J2 G. Mühlberger, L. Seaward, M. Terras, S. Ares Oliveira, V. Bosch, M. Bryan, S. Colutto, H. Déjean, M. Diem, S. Fiel, B. Gatos, T. Grüning, A. Greinoecker, G. Hackl, V. Haukkovaara, G. Heyer, L. Hirvonen, T. Hodel, M. Jokinen, P. Kahle, M. Kallio, F. Kaplan, K. Florian, R. Labahn, E.M. Lang, S. Laube, G. Leifert, G. Louloudis, R. McNicholl, J.L. Meunier, J. Michael, E. Mühlbauer, N. Philipp, I. Pratikakis, J. Puigcerver Pérez, H. Putz, **G. Retsinas**, V. Romero, R. Sablatnig, J.A. Sánchez, P. Schofield, G. Sfikas, C. Sieber, N. Stamatopoulos, T. Strauß, T. Terbul, A.H. Toselli, B. Ulreich, M. Villegas, E. Vidal, J. Walcher, M. Weidemann, H. Wurster, K. Zagoris, "Transforming Scholarship in the Archives Through Handwritten Text Recognition: Transkribus as a Case Study", Journal of Documentation, Vol. 75, No. 5, pp. 954-976, September, 2019.

B.2 Publications in Peer-reviewed International Conferences

C1 **G. Retsinas**, B. Gatos, N. Stamatopoulos and G. Louloudis, "Isolated Character Recognition using Projections of Oriented Gradients", In Proc. of the 13th International Conference on Document Analysis and Recognition (ICDAR'15), pp. 336-340, Nancy, France, 2015.

C2 B. Gatos, N. Stamatopoulos, G. Louloudis, G. Sfikas, **G. Retsinas**, V. Papavassiliou, F. Simistira and V. Katsouros, "GRPOLY-DB: An Old Greek Polytonic Document Image Database", In Proc. of the 13th International Conference on Document Analysis and Recognition (ICDAR'15), pp. 646-650, Nancy, France, 2015.

C3 **G. Retsinas**, B. Gatos, A. Antonacopoulos, G. Louloudis and N. Stamatopoulos, "Historical Typewritten Document Recognition Using Minimal User Interaction", In Proc. of the 3rd International Workshop on Historical Document Imaging and Processing (HIP'15), pp. 31-38, Nancy, France, 2015.

C4 **G. Retsinas**, G. Louloudis, N. Stamatopoulos and B. Gatos, "Efficient Document Image Segmentation Representation by Approximating Minimum-Link Polygons", In Proc. of the 12th Workshop on Document Analysis Systems (DAS'16), pp. 293-298, Santorini, Greece, 2016.

C5 **G. Retsinas**, G. Louloudis, N. Stamatopoulos and B. Gatos, "Keyword Spotting in Handwritten Documents using Projections of Oriented Gradients", In Proc. of the 12th Workshop on Document Analysis Systems (DAS'16), pp. 411-416, Santorini, Greece, 2016.

C6 G. Sfikas, **G. Retsinas**, B. Gatos, "Zoning Aggregated Hypercolumns for Keyword Spotting", In Proc. of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR'16), Shenzhen, China, 2016.

C7 **G. Retsinas**, N. Stamatopoulos, G. Louloudis, G. Sfikas, B. Gatos, "Nonlinear Manifold Embedding on Keyword Spot ting using t-SNE", In Proc. of the 14th International Conference on Document Analysis and Recognition (ICDAR'17), pp. 487-492, Kyoto, Japan, 2017.

C8 G. Sfikas, **G. Retsinas**, B. Gatos, "A PHOC Decoder for Lexicon-Free Handwritten Word Recognition", In Proc. of the 14th International Conference on Document Analysis and Recognition (ICDAR'17), pp. 513-518, Kyoto, Japan, 2017.

C9 **G. Retsinas**, G. Sfikas, and B. Gatos, "Transferable deep features for keyword spotting", In Multidisciplinary Digital Publishing Institute Proceedings (IWCIM in conjunction with EUSIPCO), 2017.

C10 **G. Retsinas**, G. Sfikas, N. Stamatopoulos, G. Louloudis, B. Gatos, "Exploring Critical Aspects of CNN-based Keyword Spotting. A PHOCNet Study", In Proc. of the 13th Workshop on Document Analysis Systems (DAS'18), pp. 13-18, Vienna, Austria, 2018.

C11 **G. Retsinas**, G. Sfikas, G. Louloudis, N. Stamatopoulos and B. Gatos, "Compact Deep Descriptors for Keyword Spotting", In Proc. of the 16th International Conference on Frontiers in Handwriting Recognition (ICFHR'18), Niagara Falls, USA, 2018.

C12 **G. Retsinas**, G. Louloudis, N. Stamatopoulos, G. Sfikas and B. Gatos, "An Alternative Deep Feature Approach to Line Level Keyword Spotting", In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, USA, 2019.

C13 **G. Retsinas**, A. Elafrou, G. Goumas and P. Maragos, "RecNets: Depthwise-Recurrent Convolutional Neural Networks", In Proc. of the British Machine Vision Conference (BMVC), Cardiff, UK, 2019.

C14 **G. Retsinas**, P. P. Filntisis, N. Efthymiou, E. Theodosis, A. Zlatintsi and P. Maragos, "Person Identification using Deep Convolutional Neural Networks on Short-Term Signals from Wearable Sensors", In Proc. of the 45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, 2020.

C15 G. Smyrnis, **G. Retsinas**, P. Maragos, "Maxpolynomial Division with Application to Neural Network Simplification", In Proc. of the 45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, 2020.

B.3 Publications Under Review in Peer-reviewed International Conferences

C16 **G. Retsinas**, A. Elafrou, G. Goumas and P. Maragos, "Compressive Learning via Adaptive Sparsity Loss", European Conference on Computer Vision (ECCV), under review, 2020.

C17 **G. Retsinas**, G. Sfikas and P. Maragos, "WSRNet: Joint Spotting and Recognition of Handwritten Words", European Conference on Computer Vision (ECCV), under review, 2020.