



National Technical University of Athens

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF SIGNALS, CONTROL AND ROBOTICS

COMPUTER VISION, SPEECH COMMUNICATION AND SIGNAL PROCESSING
GROUP

An Exploration of Deep Learning Architectures for Handwritten Text Recognition

Diploma Thesis

Vasiliki Tassopoulou

Supervisor: Petros Maragos, Prof NTUA

06-11-2019



National Technical University of Athens

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF SIGNALS, CONTROL AND ROBOTICS

COMPUTER VISION, SPEECH COMMUNICATION AND SIGNAL PROCESSING
GROUP

An Exploration of Deep Learning Architectures for Handwritten Text Recognition

Diploma Thesis
Vasiliki Tassopoulou

Supervisor: Petros Maragos, Prof NTUA

Approved by the examining committee:

.....
Μαραγκός Π.
Καθηγητής Ε.Μ.Π.

.....
Τζαρέστας Κ.
Αν. Καθηγητής Ε.Μ.Π.

.....
Ψυλλάκης Χ.
Λέκτορας Ε.Μ.Π.

.....

Vasiliki Tassopoulou
Electrical and Computer Engineer
©2019 - All rights reserved.

This work is copyright and may not be reproduced, stored nor distributed in whole or in part for commercial purposes. Permission is hereby granted to reproduce, store and distribute this work for non-profit, educational and research purposes, provided that the source is acknowledged and the present copyright message is retained. Enquiries regarding use for profit should be directed to the author. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the National Technical University of Athens.

Περίληψη

Το αντικείμενο αυτής της διπλωματικής είναι η Αναγνώριση Χειρόγραφων Κειμένων με χρήση μεθόδων βαθιάς μάθησης. Στα πλαίσια αυτής πειραματιζόμαστε με μια πληθώρα ιδεών που αφορούν όλα τα επιμέρους τμήματα των εργασιών που συνθέτουν το τελικό μας σύστημα. Αρχικά, ασχολούμαστε με την υλοποίηση της βασικής αρχιτεκτονικής ενώ παράλληλα υλοποιούμε δύο επιπλέον μεθόδους δυναμικής επαύξησης δεδομένων, τον τοπικό αφινικό μετασχηματισμό και τον τοπικό μορφολογικό μετασχηματισμό, με βασικό κίνητρό μας να την υλοποίηση μετασχηματισμών που δεν θα δράσουν σε όλη την εικόνα αλλά θα μετασχηματίσουν και θα επαυξήσουν τοπικά τα γράμματα της κάθε εικόνας. Εξάγουμε το γενικό συμπέρασμα ότι η δυναμική επαύξηση των δεδομένων ενισχύει την ικανότητα γενίκευσης του μοντέλου μας και οδηγεί σε καλύτερα ποσοστά αναγνώρισης. Στην συνέχεια πειραματιζόμαστε με την ακολουθία χαρακτήρων που μαθαίνει το μοντέλο μας με χρήση του CTC αλγορίθμου. Επαυξάνουμε την ακολουθία με διγράμματα επιπέδου χαρακτήρων. Υλοποιούμε αυτό το σύνθετο σχήμα με απώτερο σκοπό την απόκτηση ενός οπτικού στατιστικού μοντέλου που έχει αποκτηθεί παράλληλα με την εκπαίδευση του δικτύου, και τη χρησιμοποίηση αυτού σε δύο νέες εκδοχές αλγορίθμων CTC αποκωδικοποίησης για την βελτίωση του ποσοστού αναγνώρισης. Έπειτα, υλοποιούμε αρχιτεκτονική πολλαπλών εργασιών με το CTC χρησιμοποιώντας ως κλάσεις και μονούς χαρακτήρες και διγράμματα αλλά σε διαφορετικά CTC επίπεδα όπου και λαμβάνουμε σημαντική βελτίωση στο ποσοστό αναγνώρισης. Με αυτό τον τρόπο, εκμεταλλευόμαστε την γλωσσική πληροφορία δύο φορές καθώς την ενσωματώνουμε και στην διαδικασία μάθησης του μοντέλου, μέσω των διγραμμάτων, αλλά και στην διαδικασία αποκωδικοποίησης αυτού με χρήση των στατιστικών γλωσσικών μοντέλων. Επίσης, συγκρίνοντας τα μοντέλα πολλαπλών εργασιών τόσο με το δικό μας μοντέλο μιας εργασίας όσο και με της βιβλιογραφίας συμπεράναμε ότι αυτή η προσέγγιση είναι καλύτερη και στις δύο περιπτώσεις. Έπειτα υλοποιούμε μια αρχιτεκτονική πλήρως συνελικτική τόσο στο οπτικό μας μοντέλο όσο και στο ακολουθιακό. Κίνητρο μας είναι να αποφύγουμε τα αναδρομικά δίκτυα μακροπρόθεσμης μνήμης (LSTM) που φέρουν μεγάλο υπολογιστικό κόστος και εμφανίζουν δυσκολία στη σύγκλιση και στην εκπαίδευση. Το μοντέλο που αναπτύξαμε συγκλίνει γρηγορότερα ενώ έχει και σημαντικά λιγότερες παραμέτρους από τα προαναφερθέντα.

Λέξεις Κλειδιά: Αναγνώριση Χειρόγραφων Κειμένων, Όραση Υπολογιστών, Βαθιά Μάθηση, Συνελικτικά Νευρωνικά Δίκτυα, Μοντελοποίηση Ακολουθιών, Connectionist Temporal Classification, Αλγόριθμοι Αποκωδικοποίησης, Δυναμική Επαύξηση Δεδομένων, Στατιστικά Γλωσσικά Μοντέλα, Μάθηση Πολλαπλών Εργασιών

Abstract

The objective of this thesis is the study of the Handwritten Text Recognition problem with the use of deep learning models. In this thesis, we experiment with a variety of tasks that apply to the whole pipeline that synthesizes our final model. At first, we implement the baseline architecture and then we experiment with dynamic data augmentation. We implement two new augmentation techniques, the local affine transform, and the local morphological transform. Our incentive behind this is the implementation of transformations that will augment the letters and not the whole text line. Generally, we deduced that dynamic data augmentation makes the model more able to generalize and improves recognition rates. Then, we experiment with the CTC alignments that our model learns. We augment the target sequence with bigrams, except for unigrams. We train such complex alignments so as to obtain a bigram level visual language model and we utilize it in two new CTC beam search decoding algorithms, extended in such way so as to support the integration of obtained bigram information, in order to improve the recognition rates. Thereinafter, we experiment with multitask architectures with CTC, both hierarchical and block. Our experiments culminate in significant improvement in the recognition rate. With the multitask approach we exploit the language information (domain knowledge) in two ways. We integrate it both in the learning procedure via the ngrams, that are selected as target units, and the decoding process via the statistical language models. Finally, we implement a fully convolutional architecture where both the optical and sequential models were composed of convolutions. We show that the CTC layer can be successfully employed on top of a CNN network. Also, we found out that one-dimensional convolution can model sufficiently the temporal relationships among the features. Finally, our fully convolutional model converges fast, has significantly lower training and inference time and has also respectfully fewer parameters than the aforementioned architectures.

Keywords: Handwritten Text Recognition, Computer Vision, Deep Learning, Sequence Modeling, Convolutional Neural Networks, Sequence Transduction, Connectionist Temporal Classification, Decoding Algorithms, Dynamic Data Augmentation, Statistical Language Models, Multitask Learning

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Όρασης Υπολογιστών, Επικοινωνίας Λόγου και Επεξεργασίας Σήματος της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών του Εθνικού Μετσοβίου Πολυτεχνείου.

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα Καθηγητή μου κ. Πέτρο Μαραγκό για την ευκαιρία που μου έδωσε να πραγματοποιήσω την διπλωματική μου εργασία σε αυτό το εργαστήριο αλλά και τις σημαντικές συμβουλές του και παρατηρήσεις του κατά τη διάρκεια αυτής της διπλωματικής. Πέρα από τη διπλωματική, είμαι ευγνώμων για τις γνώσεις που απέκτησα στον τομέα της Όρασης Υπολογιστών και της Αναγνώρισης Προτύπων, αντικείμενα που επιθυμώ εμβαθύνω στην συνέχεια της ακαδημαϊκής μου πορείας. Σημαντικό ρόλο στη διεκπαιρέωση αυτής της εργασίας καταλαμβάνει ο Υποψήφιος Διδάκτωρ Γιώργος Ρετσινάς. Η παραπάνω από συχνή επικοινωνία μας, η καθοδήγηση και η στήριξη του τόσο σε θεωρητικά όσο και σε πρακτικά ζητήματα αποτέλεσαν καταλυτικό παράγοντα που συνέβαλε στην ολοκλήρωση αυτού του έργου. Θα ήθελα να τον ευχαριστήσω δημοσίως και νιώθω παραπάνω απο ευγνώμων για αυτή μας τη συνεργασία.

Η πορεία μου στο Πολυτεχνείο κλείνει με πολλή συγκίνηση αλλά και κυρίως αστείρευτη ευγνωμοσύνη για τους ανθρώπους με τους οποίους συμπορεύτηκα και σημάδεψαν την φοιτητική μου πορεία. Αυτοί είναι οι φίλοι που το Πολυτεχνείο μου έδωσε. Σας ευχαριστώ όλους για όλες τις στιγμές που έχουμε ζήσει και για όλες τις επόμενες που θα έρθουν παρά'όλη την απόσταση που μπορεί να μας χωρίζει.

Δίχως την ανιδιοτελή στήριξη της οικογένειας μου δεν θα μπορούσα να έχω φτάσει ως εδώ. Τους ευχαριστώ για όλα και τους χρωστάω τα πάντα.

Βασιλική Τασσοπούλου,
Οκτώβριος 2019

Contents

1 Εκτεταμένη Περίληψη	13
1.1 Εισαγωγή	14
1.2 Δυναμική Επαύξηση Δεδομένων και Βασική Αρχιτεκτονική	15
1.2.1 Μέθοδοι Επαύξησης Δεδομένων	15
1.2.2 Βασική Αρχιτεκτονική	17
1.2.3 Πειράματα	20
1.3 NGram Διάσπαση Ακολουθίας	23
1.3.1 Μοντέλο Unigram-Bigram	23
1.3.2 Προτεινόμενοι Αλγόριθμοι Αποκωδικοποίησης	24
1.3.3 Ανάλυση-Πειράματα-Συμπεράσματα	26
1.4 Μάθηση Πολλαπλών Στόχων	29
1.4.1 Πειράματα - Τελική Αρχιτεκτονική	30
1.4.2 Σύγκριση Μοντέλων Μιας Εργασίας με Μοντέλα Πολλαπλών	32
1.5 Πλήρως Συνελικτική Αρχιτεκτονική με CTC για Αναγνώριση Χειρόγραφων	34
1.5.1 Πλήρως Συνελικτική Αρχιτεκτονική	34
1.5.2 Πειράματα	35
1.5.3 Συνδυασμός Συνελικτικών μοντέλων	37
1.5.4 Σύγκριση CNN+CTC με το βασικό μοντέλο	38
1.5.5 Συμπεράσματα	39
1.6 Συνεισφορές, Συμπεράσματα και Μελλοντική Δουλειά	41
2 Introduction	43
2.1 Problem Definition	44
2.2 Related Problems in the Document Analysis field	45
2.2.1 Keyword Spotting	45
2.2.2 Verification-Identification	45
2.2.3 Layout Analysis	48
2.3 IAM Database	48
2.4 Thesis Outline	50
3 Theoretical Background	51
3.1 Theoretical Background	52
3.2 Machine Learning Preliminaries	52
3.2.1 Supervised Learning	52
3.2.2 Unsupervised Learning	56
3.3 Neural Networks	56
3.3.1 Learning Algorithms	59
3.4 Deep Neural Networks	61

3.4.1	Convolutional Neural Networks	61
3.4.2	Sequence Modeling : Recurrent Neural Networks	64
3.5	Connectionist Temporal Classification	68
3.5.1	CTC Decoding Algorithms	71
3.5.2	N-Gram Language Models in CTC Decoding	72
4	Previous Work	74
4.1	Previous Work	75
4.2	Preprocessing Techniques for HTR	75
4.3	Probabilistic Models for HTR	77
4.3.1	Hidden Markov Models	77
4.3.2	Hidden Markov Models for Handwritten Text	77
4.4	Neural Models for HTR	78
4.4.1	Multidimensional RNN for HTR	78
4.4.2	One-dimensional RNN for HTR	79
5	Data Augmentation for HTR and Baseline Model	82
5.1	Data Augmentation and Baseline Model for HTR	83
5.2	Data Augmentation	83
5.2.1	Global Affine Transform	83
5.2.2	Global Morphological Transform	84
5.2.3	Local Affine Transform	85
5.2.4	Local Morphological Transform	85
5.2.5	Other Transforms	86
5.3	Baseline Architecture	88
5.3.1	Experiments	90
5.4	Decoding the Network Outputs	92
5.4.1	External Language Model	92
5.4.2	Decoding Experiments	92
6	Ngram Models	94
6.1	NGram Models	95
6.2	Unigram-Bigram Model	95
6.3	Shared CTC Layer Architecture for Unigram-Bigram Model	95
6.3.1	Proposed Algorithms for Decoding Unigram-Bigram Model	99
6.3.2	Output Analysis for Revision of Proposed Algorithms	104
6.3.3	Decoding Experiments	112
6.4	Conclusions	112
7	Hierarchical and Multitask Learning in HTR	113
7.1	Hierarchical and Multitask Approaches in HTR	114
7.1.1	Possible Architectures for Multiscale-Target Models	114
7.1.2	Hard Parameter Sharing for Unigram-Bigram Model	115
7.2	Hierarchical Multitask Learning with CTC - An Application to HTR	119
7.2.1	Multitask Learning	120
7.2.2	Experimenting with Multitask Architectures	128
7.3	Single Task Vs Multi Task Architectures for HTR	130
7.4	Conclusions	132

8	Deep Convolutional CTC Network for Handwriting Recognition	133
8.1	Introduction	134
8.2	Motivation	134
8.3	Fully-Convolutional Architecture	135
8.3.1	Experiments	136
8.4	An Ensemble of CNN Models for HTR	138
8.5	Comparing CNN+CTC with Baseline	138
8.6	Conclusions	140
9	Contributions, Conclusions and Future Work	142
9.1	Contributions, Conclusions and Future Work	143
A	Appendix	145
A.1	Examples of Greedy Decoded Alignments of Unigrams and Bigrams	145

List of Figures

1.1	Αρχική Εικόνα	15
1.2	Affine Μετασχηματισμοί	15
1.3	Αρχική Εικόνα	16
1.4	Τοπικοί Αφινικοί Μετασχηματισμοί	16
1.5	Τοπικοί Μορφολογικοί Μετασχηματισμοί	17
1.6	Συνολική βασική αρχιτεκτονική για την αναγνώριση χειρόγραφων κειμένων	19
1.7	Μετατροπή χαρτών ενεργοποίησης σε ακολουθία διανυσμάτων	19
1.8	Alignments για το κοινό Unigram-Bigram μοντέλο υπό το κοινό CTC framework [27]	23
1.9	Παράδειγμα 1	26
1.10	Παράδειγμα 2	27
1.11	Τελική αρχιτεκτονική Block-πολλαπλών στόχων χωρίς BiLSTM επίπεδο σε κάθε στόχο	31
1.12	Αναπαράσταση της CNN-CTC αρχιτεκτονικής. Ο CNN Encoder παραμένει ως έχει και η μόνη αλλαγή που πραγματοποιείται στο δίκτυο είναι η αντικατάσταση των BiLSTM επιπέδων με χρήση 1-D συνελίξεων	35
1.13	Πιθανότητες που το κάθε μοντέλο αναθέτει σε κάθε λέξη της πρότασης.	38
1.14	Απόκλιση Kullback-Leibler λέξεων όμοια αναγνωρισμένων από τα μοντέλα CNN, LSTM με σκοπό τη σύγκριση αυτών	39
2.1	Image Samples that showcase the unconstrained and cursive nature of human handwriting.	44
2.2	Objective of Layout Analysis [3]	48
2.3	Train Samples of IAM Database	49
2.4	Test Samples of IAM Database	49
3.1	Graphical illustration of bias and variance [5]	54
3.2	The Model Complexity as a function of Bias and Variance [5]	54
3.3	For a different type of intuition, consider the following figure, in which x 's represent positive training examples, o 's denote negative training examples, a decision boundary (this is the line given by the equation $\theta^T \mathbf{x} = 0$, and is also called the separating hyperplane). A, B and C are sample points.	55
3.4	The perceptron of Rosenblatt	57
3.5	The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links inputs coming from additional input and hidden variables x_0 and z_0 . Arrows denote the direction of information flow through the network during forward propagation. [54]	58
3.6	Neural Network Activation Functions [24]	58
3.7	Convolution Operation in Input Image [22]	61
3.8	Convolution Operation in Input Image [22]	62
3.9	Unrolling the Recurrent Neural Networks [7]	64
3.10	Bidirectional Recurrent Neural Network [7]	65

3.11	Internal Structure of LSTM Layer [7]	65
3.12	The cell state of LSTM [7]	66
3.13	The Forget-Gate of LSTM. In this step, the LSTM decides what information will be thrown away from the cell state. This is implemented by a sigmoid layer which looks at the h_{t-1} and the x_t and assigns a number of 0 or 1 on each of the units of C_{t-1} . [7]	66
3.14	The Input-Gate of LSTM. In this step, the LSTM decides what new information will be stored. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a tanh layer creates a vector of new candidate values, \hat{C}_t , that could be added to the state. In the next step, we’ll combine these two to create an update to the state. [7]	67
3.15	The Output-Gate of LSTM. In this step, the LSTM updates the C_{t-1} value with the new one C_t and its synthesized by how much information will be abandoned and what new information will be stored in the cell. [7]	67
3.16	Example of a possible alignment for word "cat" [8]	68
3.17	Graphical explanation of the recurrent equation of forward variable	70
4.1	Bleed-Through removal example [34]	75
4.2	Image Binarization Example in Historical Handwritten Documents [49]	76
4.3	Example of the alignment produced by a character HMM modeling the letter “a”. The HMM is composed of four states in a left-to-right topology. The probabilities in the arcs represent the transition probabilities [52]	77
4.4	Forward Pass in MDRNN [26]	78
4.5	Backward Pass in MDRNN [26]	79
4.6	A two dimensional sequence can be processed in various directions. The internal arrows inside the rectangle indicate the direction of the propagation in forward pass. [26]	79
4.7	The basic network architecture used in this paper. The input image on the left is processed pixel-by-pixel using a cascade of convolutional, max-pooling and MDLSTM layers, and finally transcribed by a CTC layer on the right [74]	79
4.8	Randomly selected features extracted after a 2D-LSTM and after a convolutional layers[53]	80
4.9	One-Dimensional LSTM Architecture for HTR [53]	80
4.10	Reported Results on One Dimensional LSTMs [53]	81
5.1	Linear Transformation - Affine Transform Case	83
5.2	Non Linear Image Transformations based on Morphological Filtering	85
5.3	Local Affine Transform Samples	86
5.4	Local Morphological Transforms	86
5.5	Gaussian Blurring	87
5.6	Horizontal Cutout	87
5.7	Vertical Cutout	87
5.8	Noise Induction	87
5.9	Unigram Alignments	88
5.10	End-to-End Baseline Architecture for HTR	90
5.11	Map-to-Sequence Operation	90
6.1	Alignments for Shared Unigram-Bigram Model under the standard CTC Framework [27]	95
6.2	Computation of forward variable a_s^t for Unigram-Bigram Alignments under the standard CTC framework	96
6.3	Example 1	104
6.4	Example 2	106
6.5	Example 3	107

6.6	Example 4	108
6.7	Example 5	109
6.8	Example 6	110
6.9	Both Directions Mismatch	111
6.10	Backward Mismatch	111
6.11	Forward Mismatch	111
7.1	A taxonomy of implemented architectures for HTR	114
7.2	Alignments for Bigrams	115
7.3	Train loss in bigram model training	115
7.4	Comparison WER/CER in proposed bigram decoding schemes	116
7.5	Unigram Greedy Alignment	117
7.6	Bigram Greedy Alignment	117
7.7	Unigram and Bigram Alignments	118
7.8	Hierarchical multitask learning with CTC for ASR [66]	121
7.9	Block multitask learning with CTC for ASR [66]	121
7.10	Block multitask architecture for HTR	122
7.11	Hierarchical multitask architecture for HTR	123
7.12	Bigram level tokens	124
7.13	Trigram level tokens	125
7.14	Fourgram level tokens	126
7.15	Fivegram level tokens	127
7.16	The final proposed architecture for multitask learning. Block multitask without task-specific BiLSTM layer	130
8.1	Illustration of the CNN-CTC Architecture. The CNN Encoder remains as it is and the only change that is applied in the whole network is the substitution of BiLSTM Layers with the 1-D Convolutions	135
8.2	Posterior Probabilities that each mode assigns to each word of the sentence	139
8.3	KL of words for comparing CNN LSTM models for HTR	140

List of Tables

1.1	Συνολική αρχιτεκτονική	18
1.2	Διαμόρφωση πειραμάτων στα επίπεδα BiLSTM	20
1.3	Πειράματα στον αριθμό των επιπέδων του BiLSTM	20
1.4	Διαμόρφωση πειραμάτων δυναμικής επαύξησης δεδομένων	21
1.5	Η επίδραση της επαύξησης δεδομένων στο απόδοση του μοντέλου	21
1.6	Αποτελέσματα CTC αποκωδικοποίησης με χρήση στατιστικού γλωσσικού μοντέλου	21
1.7	Διαμόρφωση Πειραμάτων Unigram-Bigram	24
1.8	Ποσοστά αναγνώρισης στο Unigram-Bigram μοντέλο μόνο με χρήση πιθανοτήτων unigrams	24
1.9	Πειράματα στους προτεινόμενους αλγόριθμους αποκωδικοποίησης για το κοινό μοντέλο Unigram-Bigram	28
1.10	Ιεραρχική αρχιτεκτονική πολλαπλών στόχων - Greedy αποκωδικοποίηση	30
1.11	Block αρχιτεκτονική πολλαπλών στόχων με 1 BiLSTM επίπεδο σε κάθε στόχο - Greedy Αποκωδικοποίηση	30
1.12	Block Multitask αρχιτεκτονική δίχως task-specific LSTM επίπεδο - Greedy Αποκωδικοποίηση	31
1.13	Διαρρύθμιση πειραμάτων για τη σύγκριση αρχιτεκτονικής πολλαπλών εργασιών με αρχιτεκτονική μιας εργασίας	32
1.14	STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων σε επίπεδο unigram - CTC 4-Gram Char LM	32
1.15	STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων σε επίπεδο unigram - CTC 4-Gram Char LM	32
1.16	STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων σε επίπεδο unigram - CTC Word LM 4-Gram	33
1.17	STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων με greedy αποκωδικοποίηση επιπέδου χαρακτήρων	33
1.18	CNN-CTC αρχιτεκτονική	35
1.19	Διαμόρφωση Εκπαίδευσης	36
1.20	Πειράματα δυναμικής επαύξησης δεδομένων σε πλήρως συνελικτικές αρχιτεκτονικές	36
1.21	Αποτελέσματα αποκωδικοποίησης με χρήση εξωτερικού γλωσσικού μοντέλου - επίπεδο γραμμής	36
1.22	Επίδοση των απλών συνελικτικών δικτύων και του συνδυασμού αυτών	37
1.23	Σύγκριση επίδοσης απλών μοντέλων με Ensemble μοντέλα - Char/Word LM αποκωδικοποίηση	38
1.24	Σύγκριση των αρχιτεκτονικών CNN-LSTM-CTC και CNN-CTC	38
1.25	Σύγκριση μεταξύ CNN-LSTM-CTC και CNN-CTC	39
3.1	Learnable Parameters per Layer in a ConvNet	62
5.1	End-to-End Architecture	89
5.2	Configuration of Data Augmentation Experiments	91
5.3	Experiments on BiLSTM Layers	91
5.4	Configuration of Data Augmentation Experiments	91

5.5	The effect of Data Augmentation	92
5.6	Results of decoding with external language information in line level	92
6.1	Configuration of Unigram-Bigram Experiments	98
6.2	Greedy decoding performance utilizing only unigrams and ignoring bigrams	98
6.3	Bigram scoring algorithm effect	105
6.4	Experiments on proposed algorithms for shared unigram-bigram model	112
7.1	Configuration of data augmentation experiments on bigram model	116
7.2	Data augmentation in bigram Model - greedy decoding (Greedy-Old)	117
7.3	Hierarchical multitask architecture - greedy decoding	129
7.4	Block multiscale learning with 1 task-specific BiLSTM layer - greedy decoding	129
7.5	Block multitask learning without task specific LSTM - greedy decoding	129
7.6	Configuration of Multitask Experiments	131
7.7	STL-vs-MTL architectures for HTR in unigram Level - CTC 4-Gram Char LM	131
7.8	STL-vs-MTL architectures for HTR in unigram Level - CTC 4-Gram Char LM	132
7.9	STL-vs-MTL architectures for HTR in unigram level - CTC Word LM 4-Gram	132
7.10	STL-vs-MTL architectures for HTR in unigram level - greedy decoding	132
8.1	CNN-CTC End-to-End Architecture	135
8.2	Train Configuration	136
8.3	Data Augmentation Experiments on Fully Convolutional Neural Network	137
8.4	Results of Decoding with external Language Information - Line Level	137
8.5	Performance of Multiple Single CNN Models and the Ensembling	138
8.6	Performance of Ensemble Models - Char/Word LM Decoding	138
8.7	Comparison of CNN-LSTM-CTC and CNN-CTC	139
8.8	Comparison of CNN-LSTM-CTC and CNN-CTC	140

Κεφάλαιο 1

Εκτεταμένη Περίληψη

1.1 Εισαγωγή

Ορισμός Προβλήματος

Η **Αναγνώριση Χειρόγραφων Κειμένων** ορίζεται ως η διαδικασία μετατροπής μιας ψηφιακής αναπαράστασης του γραπτού χειρόγραφου λόγου σε ψηφιακό κείμενο με στόχο τη διεκπεραίωση εργασιών όπως ευρετηρίαση, ταξινόμηση ή μετάφραση.

Η απόκτηση χειρόγραφου κειμένου μπορεί να γίνει με δύο τρόπους. Είτε μέσω ψηφιακών πλατφορμών που διαθέτουν οθόνη αφής και μέσω εισαγωγής γραπτού κειμένου, είτε με σκανάρισμα ήδη υπαρχόντων εγγράφων. Η πρώτη περίπτωση ουσιαστικά καλύπτει την αναγνώριση κειμένου από ψηφιακά μέσα, ενώ η δεύτερη συνιστά την αναγνώριση χειρόγραφων κειμένων από ψηφιακές εικόνες. Στην πρώτη περίπτωση, λαμβάνεται υπόψιν η τροχιά που αναπτύσσεται από τον γραφέα και πέραν αυτής μετρώνται και άλλα χαρακτηριστικά που βοηθούν στην αναγνώριση όπως η πίεση στην πένα του χρήστη, την κλίση αυτής και τη θέση της. Στην δεύτερη κατηγορία, διαθέσιμη προς ψηφιοποίηση είναι μόνο η σκαναρισμένη εικόνα. Συνεπώς, η αναγνώριση από ψηφιακές εικόνες ενέχει μεγαλύτερης δυσκολίας από την αναγνώριση γραπτού λόγου από ψηφιακά μέσα.

Θεματική αυτής της διπλωματικής εργασίας είναι η *εκτεταμένη-ανεξάρτητη γραφέα, αναγνώριση χειρόγραφων κειμένων από ψηφιακές εικόνες*.

1.2 Δυναμική Επαύξηση Δεδομένων και Βασική Αρχιτεκτονική

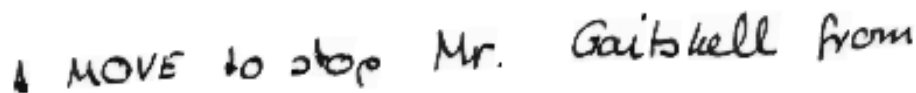
Σε αυτό το κεφάλαιο κατασκευάζουμε την βασική αρχιτεκτονική μας. Επιπλέον, υλοποιούμε δύο νέους μετασχηματισμούς επαύξησης για την περίπτωση των χειρόγραφων εικόνων, τον **Τοπικό Αφινικό Μετασχηματισμό** (Local Affine Transform) και τον **Τοπικό Μορφολογικό Μετασχηματισμό** (Local Morphological Transform). Πραγματοποιούμε πειράματα εφαρμόζοντάς τους για να δούμε την επίδραση τους στην απόδοση του μοντέλου. Τέλος, διεκπεραιώνουμε κάποια πειράματα έτσι ώστε να καταλήξουμε στην βασική αρχιτεκτονική και στις υπερπαραμέτρους που θα συνθέσουν τα υπόλοιπα πειράματά μας σε όλη την έκταση αυτής της διπλωματικής.

1.2.1 Μέθοδοι Επαύξησης Δεδομένων

Αφινικός Μετασχηματισμός



Σχήμα 1.1: Αρχική Εικόνα



Σχήμα 1.2: Affine Μετασχηματισμοί

Μορφολογικός Μετασχηματισμός

Η ιδέα πίσω από την χρήση των μορφολογικών μετασχηματισμών και ειδικά του erosion και dilation αφορά την δημιουργία λεπτότερων ή παχύτερων γραμμάτων που όντως συναντάμε σε πραγματικές περιπτώσεις γραπτού λόγου.

Χρησιμοποιούμε σύνολα για να αναπαραστήσουμε δυαδικές εικόνες και πράξεις συνόλων ώστε να αναπαραστήσουμε τους μετασχηματισμούς της δυαδικής εικόνας. Συγκεκριμένα, δεδομένης μιας δυαδικής εικόνας, υποθέτουμε ότι η πόζα το αντικείμενου συμβολίζεται με X και το υπόβαθρο του με το συμπληρωματικό σύνολο X^c . Το δομικό στοιχείο συμβολίζεται ως B .

$X_{+y} \triangleq \{x + y : x \in X\}$ είναι η μετατόπιση του X κατά μήκος του διανύσματος y και $B^s \triangleq \{x : -x \in B\}$ είναι συμμετρικό του B με αναφορά την αρχή. [43]

- Dilation : $X \oplus B \triangleq \{z : (B^s)_{z+} \cap X \neq \emptyset\} = \bigcup_{y \in B} X_{+y}$
- Erosion : $X \ominus B \triangleq \{z : (B^s)_{z+} \subseteq X\} = \bigcap_{y \in B} X_{-y}$
- Opening : $X \circ B = (X \ominus B) \oplus B$
- Closing : $X \bullet B = (X \oplus B) \ominus B$

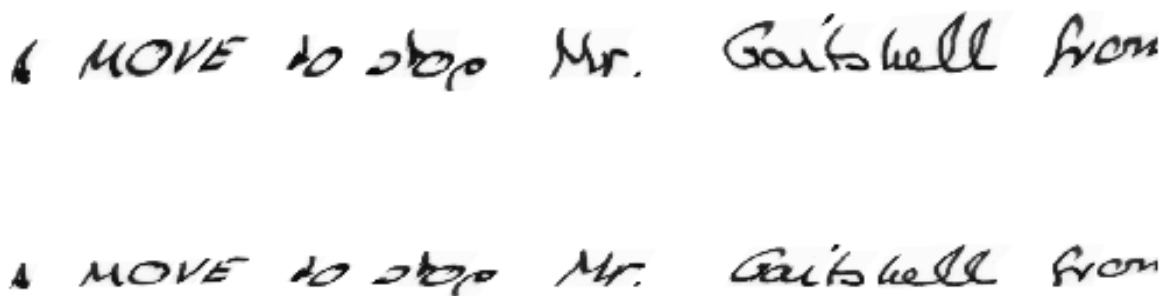
Σε εφαρμογές, το B ονομάζεται συνήθως δομικό στοιχείο και έχει ένα απλό γεωμετρικό σχήμα και μέγεθος μικρότερο από την εικόνα X . Αν το B έχει σχήμα κανονικό, για παράδειγμα μικρού δίσκου τότε και το opening και το closing δρουν ως μη γραμμικά φίλτρα που απαλύνουν τα περιγράμματα της αρχικής εικόνας. Δηλαδή, αν η εικόνα X μπορεί να παρομοιασθεί ως ένα επίπεδο νησί τότε το opening θα έχει ως συνέπεια την μείωση των αιχμηρών ακμών και τους στενούς ισθμούς, ενώ το closing γεμίζει τις μικρές τρύπες και τους στενούς κόλπους.

Τοπικός Αφινικός Μετασχηματισμός

Η διαίσθηση πίσω από την επιλογή αυτού του μετασχηματισμού είναι να προκληθούν παραμορφώσεις όχι στην ολόκληρη γραμμή του κειμένου αλλά τοπικά στα γράμματα. Για το λόγο αυτό, χωρίζουμε την εικόνα σε πλέγμα αποτελούμενο από κατακόρυφες γραμμές δημιουργώντας έτσι υποτμήματα στην εικόνα. Σε κάθε ένα από αυτά επιλέγουμε, ομοίμορφα, σημεία ελέγχου πάνω στα οποία επιδρά ένας αφινικός μετασχηματισμός. Αφού υπολογίσουμε όλα τα σημεία ελέγχου και τα μετασχηματίσουμε με βάση κάποιον αφινικό μετασχηματισμό διαφορετικό σε κάθε τμήμα, χρησιμοποιούμε την τεχνική Thin Plate Spline [69] για extrapolation ώστε να αποκτήσουμε την τελική εικόνα.



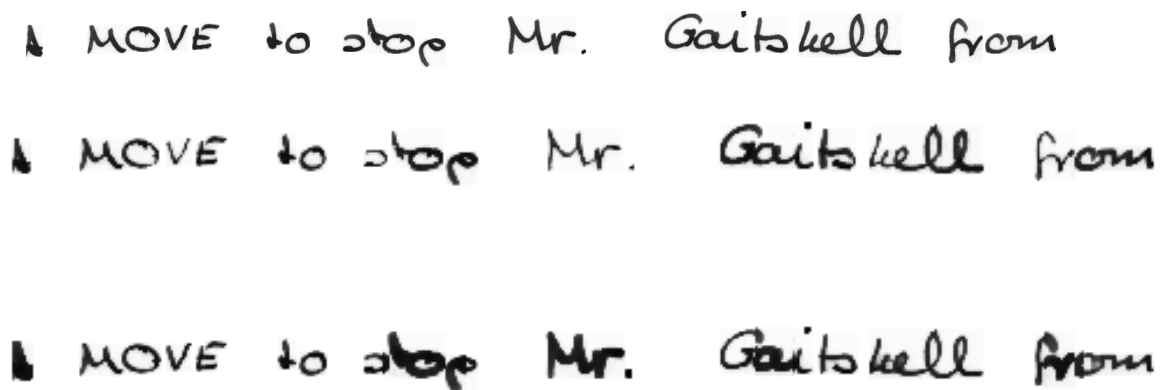
Σχήμα 1.3: Αρχική Εικόνα



Σχήμα 1.4: Τοπικοί Αφινικοί Μετασχηματισμοί

Τοπικός Μορφολογικός Μετασχηματισμός

Ο τοπικός μορφολογικός μετασχηματισμός υλοποιείται με παρόμοια λογική όπως ο τοπικός αφινικός. Η εικόνα χωρίζεται σε κάθετα πλέγματα και σε κάθε ένα από αυτά πραγματοποιείται ένα από τα dilation, erosion, opening και closing. Προκειμένου να μην υπάρχουν απότομες μεταβάσεις πάχους μεταξύ των γραμμάτων επιλέγουμε στα σύνορα μεταξύ των πλεγμάτων να πάρουμε ένα μέσο όρο από τα επικαλυπτόμενα παράθυρα.



Σχήμα 1.5: Τοπικοί Μορφολογικοί Μετασχηματισμοί

1.2.2 Βασική Αρχιτεκτονική

Η Αναγνώριση Χειρόγραφων Κειμένων είναι, τεχνικά, μια διαδικασία μονοτονικής μετατροπής μιας ακολουθίας οπτικών χαρακτηριστικών σε ακολουθία χαρακτήρων, υπολέξεων ή λέξεων. Αποτελεί δηλαδή ένα πρόβλημα μετατροπής ακολουθίας-σε-ακολουθία. Για τέτοια προβλήματα υπάρχουν διάφορες προσεγγίσεις όπως:

- Χρήση του αλγορίθμου Connectionist Temporal Classification
- Αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή (Encoder-Decoder) με χρήση αναδρομικών νευρωνικών δικτύων.

Από τις προαναφερθείσες αρχιτεκτονικές επιλέγουμε αυτή που κάνει χρήση του CTC για την απλότητα της μεθόδου αλλά και επειδή είναι από τις εξέχουσες σε προβλήματα μετατροπής ακολουθιών, ευρέως διαδεδομένη στη κοινότητα της αναγνώρισης φωνής.

Ο CTC αλγόριθμος βασίζεται στον δυναμικό προγραμματισμό για τον υπολογισμό όλων των δυνατών συνδυασμών γραμμάτων που μπορούν να αποδώσουν μια ζητούμενη ακολουθία. Για τον υπολογισμό όλων των δυνατών συνδυασμών απαιτείται, πέρα από τις κλάσεις των γραμμάτων που θέλουμε να αναγνωρίσουμε, ένας επιπρόσθετος χαρακτήρας, ο κενός (blank). Ο μοναδικός λόγος χρήσης αυτού του χαρακτήρα στον CTC αλγόριθμο είναι για να επιτρέπεται η έκφραση συνεχόμενων όμοιων χαρακτήρων όπως για παράδειγμα οι χαρακτήρες ll στην λέξη Hello. Αν δεν υπήρχε ο κενός χαρακτήρας τότε το διπλό γράμμα l δεν θα εκφραζόταν αφού θα είχε γίνει ένα με το γειτονικό του κατά τη διάρκεια της απόκτησης της τελικής λέξης από το alignment αυτής.

Εφόσον έχουμε καθορίσει την συνάρτηση με την οποία θα βελτιστοποιήσουμε το μοντέλο μας, χρειάζεται να καθορίσουμε το μοντέλο που θα υπολογίσει την δεσμευμένη πιθανότητα $p(c|x, t)$ που αποτελεί κατανομή των χαρακτήρων-στόχων c δεδομένης της εικόνας εισόδου x την χρονική στιγμή t . Η φύση του προβλήματός μας, όπου υπάρχουν χρονικές εξαρτήσεις μεταξύ των διανυσμάτων οπτικών χαρακτηριστικών, απαιτεί την χρησιμοποίηση των δικτύων μακροπρόθεσμης μνήμης (LSTM) και μάλιστα αυτών διπλής κατεύθυνσης (BiLSTM)[67].

Η αρχιτεκτονική μας περιγράφεται παρακάτω :

1. **Συνελικτικό δίκτυο για εξαγωγή οπτικών χαρακτηριστικών (οπτικό μοντέλο)**

- χρησιμοποιούμε συνελικτικό δίκτυο ώστε να παράγουμε έναν αριθμό χαρτών ενεργοποίησης (activation maps) της εικόνας εισόδου.

2. **Μετατροπή χαρτών ενεργοποίησης σε ακολουθία διανυσμάτων**

- Μετατρέπει τους διδιάστατους πίνακες σε ακολουθίες λαμβάνοντας την μέγιστη ενεργοποίηση στην διάσταση του ύψους κάθε πίνακα ενεργοποίησης.

3. **Διπλής κατεύθυνσης δίκτυο LSTM (ακολουθιακό μοντέλο)**

4. **CTC Επίπεδο**

- Γραμμικό επίπεδο που προβάλλει την έξοδο του BiLSTM στην διάσταση των κλάσεων εξόδου.

Για να επιτύχουμε γενίκευση και βελτίωση επίδοσης του μοντέλου μας περιλαμβάνουμε τα παρακάτω:

- Batch Normalization [38] στα επίπεδα του συνελικτικού αποκωδικοποιητή
- δυναμική Επάυξηση Δεδομένων
- Dropout [70] στο CTC επίπεδο

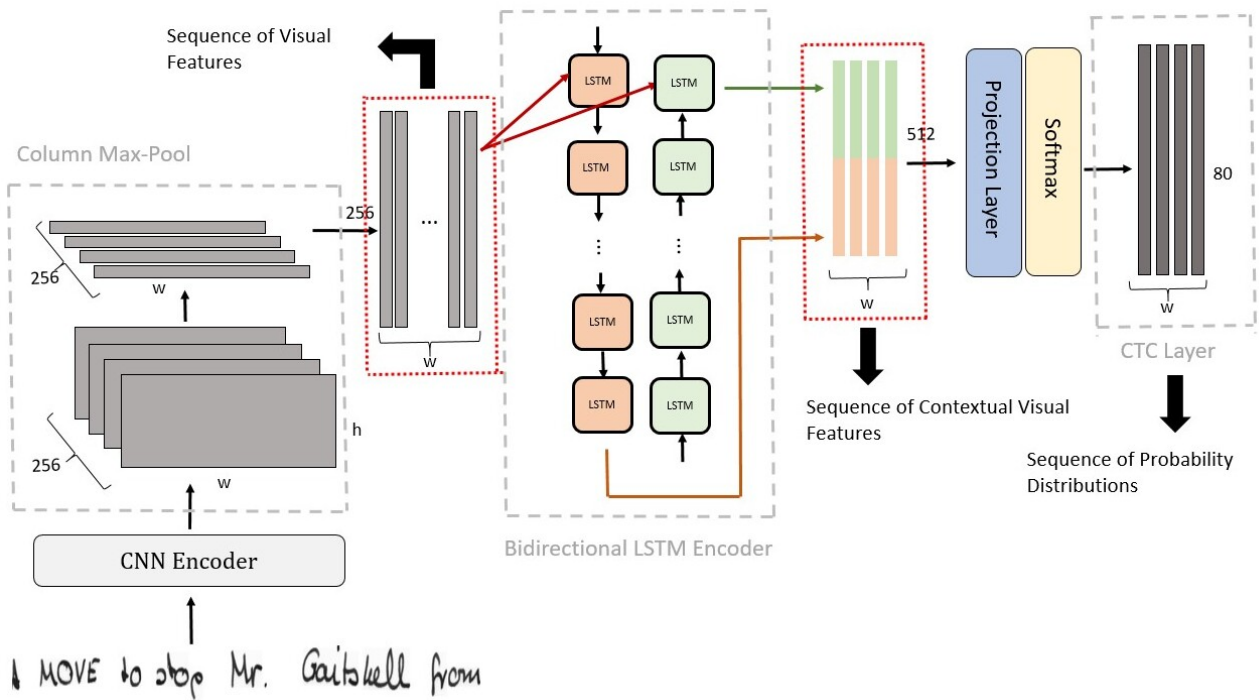
Η χρήση του Batch Normalization χρησιμοποιείται ευρέως στην ανάπτυξη μοντέλων βαθιάς μάθησης διότι προλαμβάνει το δίκτυο από το να είναι επιρρεπές στη διακύμανση των δεδομένων εισόδου. Αυτό έχει ως αποτέλεσμα να κάνει το μοντέλο πιο εύρωστο, ενώ παράλληλα επιταχύνει την διαδικασία εκπαίδευσης. Πρόκειται για το normalization των εξόδων ενεργοποίησης με βάση τη μέση τιμή και την απόκλιση που προκύπτει από όλο το batch.

Το Dropout συνίσταται στην τυχαία αφαίρεση κάποιων νευρώνων, με τον μηδενισμό αυτών, κατά τη διάρκεια της εκπαίδευσης. Με αυτόν τον τρόπο το δίκτυο δεν λαμβάνει όλη την πληροφορία, αλλά καλείται να εξάγει ένα αποτέλεσμα με μια μειωμένη εκδοχή αυτής. Ως αποτέλεσμα, το μοντέλο γενικεύει καλύτερα ενώ απόφευγεται παράλληλα το overfitting στα δεδομένα εκπαίδευσης.

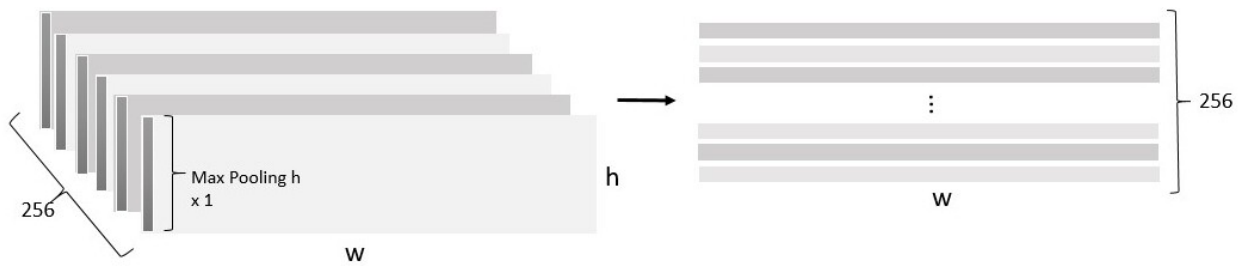
Η λεπτομερής αρχιτεκτονική του δικτύου είναι η ακόλουθη:

2 Conv Layers - 32 Kernels 2x2 - ReLU
Max Pooling
4 Conv Layers - 64 Kernels 2x2 - ReLU
Max Pooling
6 Conv Layers - 128 Kernels 2x2 - ReLU
Max Pooling
2 Conv Layers - 256 Kernels 2x2 - ReLU
2 Conv Layers - 256 Kernels 2x2 - ReLU
Max Pooling - Kernel Size 256x1
3 Layers BiLSTM - Hidden Size 512 - Input Size 256

Πίνακας 1.1: Συνολική αρχιτεκτονική



Σχήμα 1.6: Συνολική βασική αρχιτεκτονική για την αναγνώριση χειρόγραφων κειμένων



Σχήμα 1.7: Μετατροπή χαρτών ενεργοποίησης σε ακολουθία διανυσμάτων

1.2.3 Πειράματα

Η αξιολόγηση των μοντέλων έγινε με χρήση των μετρικών Word Error Rate και Character Error Rate. Αν συμβολίσουμε με Y_{dec} την αποκωδικοποιημένη ακολουθία που δίνει το μοντέλο και με Y_{gt} την πραγματική ακολουθία κειμένου και με w_d και w_t το πλήθος των λέξεων σε κάθε πρόταση τότε έχουμε:

$$CER = \frac{editdistance(Y_{dec}, Y_{gt})}{length(Y_{gt})} \quad (1.1)$$

$$WER = \sum_{w_d, w_t} \frac{editdistance(w_d, w_t)}{length(w_t)}, \forall w_d \in Y_{dec}, \forall w_t \in Y_{gt} \quad (1.2)$$

Οι μετρικές αυτές βασίζονται στην απόσταση Levenshtein [46]. Η απόσταση Levenshtein μετρά την διαφορά δύο ακολουθιών. Για παράδειγμα αν δυο ακολουθίες διαφέρουν κατά ένα γράμμα, τότε η απόσταση είναι 1. Συνεπώς η μετρική CER είναι ουσιαστικά ο λόγος του πλήθους των χαρακτήρων που διαφέρουν, μεταξύ των δύο ακολουθιών, προς το συνολικό πλήθος χαρακτήρων της πραγματικής ακολουθίας. Η μετρική WER ερμηνεύεται ομοίως, με τη μόνη διαφορά ότι εφαρμόζεται σε όλες τις λέξεις της πρότασης ξεχωριστά και έπειτα αθροίζεται.

Επίπεδα BiLSTM

Epochs	70
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	None
batch size	8

Πίνακας 1.2: Διαμόρφωση πειραμάτων στα επίπεδα BiLSTM

BiLSTM Layers	WER	CER
1	0.2454	0.0708
2	0.2155	0.0631
3	0.2047	0.0599
4	0.2058	0.0612
5	0.2062	0.0615

Πίνακας 1.3: Πειράματα στον αριθμό των επιπέδων του BiLSTM

Από τα παραπάνω πειράματα, καταλήγουμε ότι 3 BiLSTM επίπεδα αποτελούν ένα καλό συμβιβασμο μεταξύ της απόδοσης του μοντέλου και της υπολογιστικής πολυπλοκότητας αυτού.

Δυναμική Επάυξηση Δεδομένων

Πραγματοποιούμε κάποια πειράματα στην βασική αρχιτεκτονική με τις παρακάτω συνθήκες εκπαίδευσης:

Δοκιμάσαμε πολλαπλούς συνδυασμούς δυναμικής επάυξης δεδομένων και λάβαμε τα καλύτερα αποτελέσματα στην περίπτωση των αφινικών και μορφολογικών μετασχηματισμών εφαρμοσμένων σε όλο το πλάτος της εικόνας. Στην συνέχεια των πειραμάτων μας θα πορευθούμε με την χρήση των αφινικών μετασχηματισμών.

Layers	3
Epochs	70
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	None
batch size	8

Πίνακας 1.4: Διαμόρφωση πειραμάτων δυναμικής επαύξησης δεδομένων

Data Augmentation	WER	CER
None	0.22185	0.0655
Global Affine	0.1951	0.0577
Global Morphological	0.2107	0.0624
Global Affine + Global Morphological	0.1913	0.0568
Local Affine	0.2034	0.0623
Local Morphological	0.2143	0.0690
Local Affine + Local Morphological	0.2109	0.0621
Nested Global Affine + Morphological	0.1925	0.0568
Nested Global Affine + Morphological Randomly applied to each frame in the batch	0.2006	0.0581

Πίνακας 1.5: Η επίδραση της επαύξησης δεδομένων στο απόδοση του μοντέλου

CTC Αποκωδικοποίηση με χρήση στατιστικών Γλωσσικών Μοντέλων

Η χρήση CTC σε επίπεδο χαρακτήρων έχει φυσικά λιγότερες κλάσεις στο τελικό επίπεδο εξόδου και αντιμετωπίζει το πρόβλημα των λέξεων εκτός λεξιού μιας και οποιαδήποτε λέξη μπορεί να σχηματιστεί. Το γεγονός όμως ότι οποιαδήποτε λέξη μπορεί να σχηματιστεί είναι δυνητικά αρνητικό αφού μπορεί να εκφράζει λανθασμένες λέξεις. Συνεπώς, η χρήση CTC χωρίς τη χρήση εξωτερικής γλωσσικής πληροφορίας οδηγεί σε υψηλό ποσοστό λαθών λέξης και χαρακτήρα (word error rate (wer), character error rate (cer)).

Στα πειράματα μας από δω και στο εξής χρησιμοποιούμε στατιστικά γλωσσικά μοντέλα σε επίπεδα χαρακτήρων και λέξεων. Τα μοντέλα αυτά παράγονται από τα Brown και LOB κειμένων με τη χρήση του εργαλείου KenLM [36].

Ο αλγόριθμος CTC αποκωδικοποίησης που χρησιμοποιούμε είναι ο [28] με υλοποίηση [1].

Decoding Algorithm	WER	CER
Greedy	0.2068	0.0608
CTC Beam Search 2 Char-LM	0.2050	0.0608
CTC Beam Search 3 Char-LM	0.1972	0.0590
CTC Beam Search 4 Char-LM	0.1814	0.0564
CTC Beam Search 4-Word LM	0.1481	0.0460

Πίνακας 1.6: Αποτελέσματα CTC αποκωδικοποίησης με χρήση στατιστικού γλωσσικού μοντέλου

Γενικά παρατηρούμε ότι όσο αυξάνεται η τάξη του στατιστικού γλωσσικού μοντέλου τόσο βελτιώνεται και το ποσοστό αναγνώρισης. Αυτό οφείλεται στο γεγονός ότι καθώς αυξάνεται η τάξη γλωσσικού τόσο περισσότερο αυξάνεται η ιστορία που λαμβάνουμε υπόψη για τον υπολογισμό των συχνοτήτων εμφάνισης. Για παράδειγμα, είναι διαφορετική η συχνότητα εμφάνισης του α δεδομένου του β και διαφορετική η συχνότητα εμφάνισης του α δεδομένου του $\beta\gamma\delta$. Στην δεύτερη περίπτωση, η πληροφορία είναι εμφανώς μεγαλύτερη και για το λόγο αυτό δικαιολογούνται τα πιο ποιοτικά αποτελέσματα που λαμβάνουμε με τη χρήση γλωσσικών μοντέλων υψηλότερης τάξης.

Η μεγαλύτερη βελτίωση επέρχεται με την χρήση των γλωσσικών μοντέλων επιπέδου λέξεων. Αυτό είναι δικαιολογημένο καθώς ευνοούνται λέξεις που υπάρχουν στο λεξικό και άρα έχουν μεγάλη πιθανότητα να κατασκευαστούν κατά τη διάρκεια της αποκωδικοποίησης.

1.3 N-Gram Διάσπαση Ακολουθίας

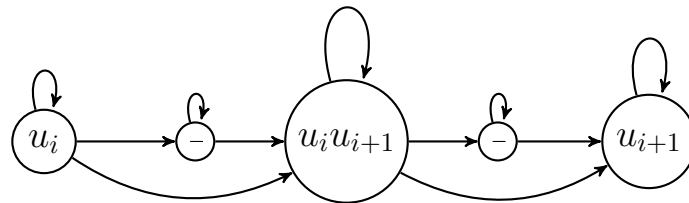
Σε αυτό το κεφάλαιο θέλουμε να διερευνήσουμε τα παρακάτω:

- Αν η διάσπαση της ακολουθίας στόχου σε n-gram μπορεί να έχει κάποια θετική επίδραση στην CTC αποκωδικοποίηση και στα ποσοστά αναγνώρισης.
- Κατά πόσο η ύπαρξη labels στο επίπεδο εξόδου σε πολλαπλές κλίμακες (unigrams, bigrams, trigrams) μπορεί να βελτιώσει την διαδικασία μάθησης.

1.3.1 Μοντέλο Unigram-Bigram

Αρχικά υλοποιούμε τη βασική αρχιτεκτονική με διαστάσεις 256 στα κρυφές μονάδες του BiLSTM. Θεωρούμε ότι επαρκούν για να μοντελοποιηθούν τόσο τα unigrams όσο και τα bigrams μιας και μοιράζονται κοινή πληροφορία.

Ο CTC μαθαίνει ακολουθίες χαρακτήρων της παρακάτω μορφής:



Σχήμα 1.8: Alignments για το κοινό Unigram-Bigram μοντέλο υπό το κοινό CTC framework [27]

Πως επιλέγουμε τα bigrams ;

Η επιλογή των bigrams που θα συμμετέχουν στο κοινό CTC επίπεδο βασίζεται στα παρακάτω :

- Όπως και στα στατιστικά γλωσσικά μοντέλα σε επίπεδο χαρακτήρων ενδιαφερόμαστε μόνο για τα πεζούς χαρακτήρες.
- Τα bigrams από όλους τους 79 χαρακτήρες θα αύξανε το υπολογιστικό κόστος τόσο στο δίκτυο όσο και στον αλγόριθμο αποκωδικοποίησης. Επίσης, πολλοί από αυτούς τους συνδυασμούς εμφανίζονται σπάνια και δεν έχουν κάποιο νόημα για την γλώσσα.
- Παρατηρώντας τις εικόνες χειρόγραφων, οι χαρακτήρες που είναι δύσκολο να αναγνωριστούν σωστά είναι αυτοί που βρίσκονται μέσα σε μια λέξη όπου δεν διακρίνονται εύκολα από τα γειτονικά τους λόγω της συνεχόμενης γραφής.

Συνεπώς, για bigrams επιλέγουμε αυτά που σχηματίζονται συνδυάζοντας τα πεζά unigrams. Από αυτά πειραματιζόμαστε με διάφορα σχήματα όπως για παράδειγμα τα 50 συχνότερα που εμφανίζονται σε ένα κείμενο ή τα 50 συχνότερα που το μοντέλο μας μπερδεύει.

Για να ελέγξουμε πώς η ύπαξη των διγραμμάτων στα σύνθετες ακολουθίες χαρακτήρων έχουν βελτιώσει την επίδοση του μοντέλου θα πραγματοποιήσουμε αποκωδικοποίηση χρησιμοποιώντας μόνο τις πιθανότητες των 80 unigrams.

Layers	5
Epochs	60
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	GA3layers
batch size	8

Πίνακας 1.7: Διαμόρφωση Πειραμάτων Unigram-Bigram

CTC Layer	WER	CER
Unigram	0.2068	0.0608
Unigram + 50 Bigrams	0.1903	0.0570
Unigram + 100 Bigrams	0.1988	0.0593
Unigram + 150 Bigrams	0.1995	0.0593
Unigram + 200 Bigrams	0.1974	0.0583
Unigram + 676 Bigrams	0.2002	0.0601

Πίνακας 1.8: Ποσοστά αναγνώρισης στο Unigram-Bigram μοντέλο μόνο με χρήση πιθανοτήτων unigrams

Από τα παραπάνω πειράματα συμπεραίνουμε ότι το μοντέλο οφελείται από την προσθήκη των bigrams, αλλά αυτό συμβαίνει στον καλύτερο βαθμό για μια μικρή ποσότητα αυτών, τα 50 bigrams. Συμπεραίνουμε λοιπόν ότι τα συνθέτα alignments που το μοντέλο μαθαίνει, βελτιώνουν κατά κάποιο τρόπο την ικανότητα γενίκευσης του μοντέλου, αφού αυτό μαθαίνει να τμηματοποιεί την εικόνα όχι μόνο σε μονούς χαρακτήρες unigrams αλλά και σε bigrams. Παρακάτω θα εξετάσουμε πως η bigram πληροφορία μπορεί να χρησιμοποιηθεί στην διαδικασία της CTC αποκωδικοποίησης.

1.3.2 Προτεινόμενοι Αλγόριθμοι Αποκωδικοποίησης

Ακολουθούμε δύο προσεγγίσεις για τους αλγόριθμους αποκωδικοποίησης. Η μια αφορά τη χρήση της bigram πιθανότητας ως εσωτερικό γλωσσικό μοντέλο bigram. Στην δεύτερη προσέγγιση χρησιμοποιούμε την bigram πιθανότητα με όμοιο τρόπο όπως την unigram, δηλαδή για επέκταση της ήδη υπάρχουσας ακολουθίας με ένα unigram ή bigram.

Ο πρώτος αλγόριθμος χρησιμοποιεί τις πιθανότητες unigram για επέκταση της ακολουθίας και τις πιθανότητες bigram για scoring της εκάστοτε επέκτασης με διάφορες πολιτικές όπως θα δούμε παρακάτω.

Συμβολίζουμε με u_i τον τελευταίο χαρακτήρα που της αναπτυσσόμενης ακολουθίας, s , ο οποίος επιλέχθηκε την χρονική στιγμή t_i και ως u_{i+1} τον χαρακτήρα επέκτασης την χρονική στιγμή t_{i+1} . Ως BS συμβολίζεται *Βιγραμ Σκορε* για την επέκταση της ακολουθίας s με τον χαρακτήρα u_{i+1} . Αυτό ορίζεται ως εξής:

$$BS = \max(Pr(u_i u_{i+1}, t_i : t_{i+1} | x)) \quad (1.3)$$

Το Bigram Score μπορεί να έχει και τους ακόλουθους ορισμούς:

Ορίζουμε ως u_i τον χαρακτήρα επέκτασης την χρονική στιγμή t και με w ένα χρονικό παράθυρο. Συμβολίζουμε με B το σύνολο των bigrams που ξεκινούν από u_i και ως C το σύνολο των bigrams που τελειώνουν σε u_i . Το BS *Bigram Score* ορίζεται ως εξής: δεφινιενδ ασ φολλωας:

1η Περίπτωση : Forward

$$\mathcal{BS} = \max_t^{t+w} \max_{b \in \mathcal{B}} Pr(b, t|X) \quad (1.4)$$

2η Περίπτωση : Backward

$$\mathcal{BS} = \max_{t-w}^t \max_{c \in \mathcal{C}} Pr(c, t|X) \quad (1.5)$$

3η Περίπτωση : Both Directions

$$\mathcal{BS} = \max \left(\max_{t-w}^t (\max_{c \in \mathcal{C}} Pr(c, t|X)), \max_t^{t+w} (\max_{b \in \mathcal{B}} Pr(b, t|X)) \right) \quad (1.6)$$

Η παρουσίαση του αλγορίθμου γίνεται αναλυτικά στο Κεφάλαιο 6.

Στον δεύτερο αλγόριθμο δημιουργούμε την αποκωδικοποιημένη ακολουθία εκμεταλλευόμενοι και τα unigrams και τα bigrams.

Θέτουμε τους παρακάτω κανόνες :

- Όταν ο επόμενος χαρακτήρας είναι bigram, τότε επεκτείνουμε με bigram και προσαρμόζουμε ανάλογα της ήδη σχηματισμένη ακολουθία.
- Όταν ο τελευταίος χαρακτήρας είναι unigram τότε αυτό θα έχει προέλθει από επέκταση unigram μόνο.
- Όταν ο τελευταίος χαρακτήρας είναι bigram τότε αυτός θα έχει προέλθει από επέκταση bigram.

Η περιγραφή του αλγορίθμου γίνεται αναλυτικά στο Κεφάλαιο 6.

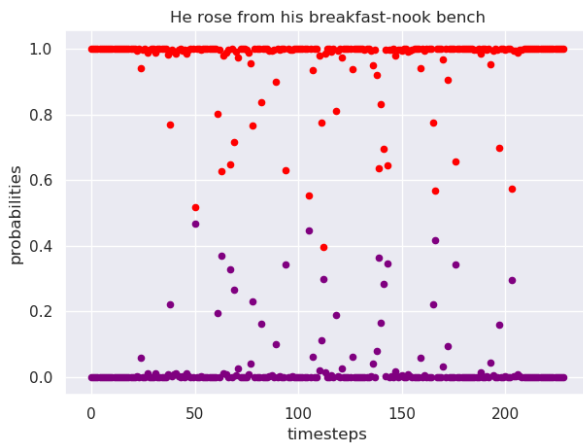
1.3.3 Ανάλυση-Πειράματα-Συμπεράσματα

Παραδείγματα από το Unigram + 50 Bigrams Μοντέλο

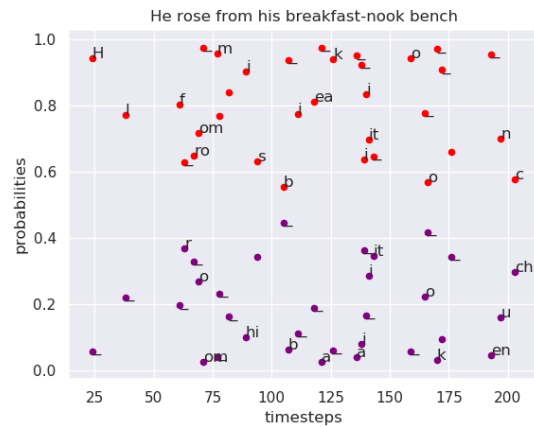
Test Set Example 1

Original : He rose from his breakfast-nook bench

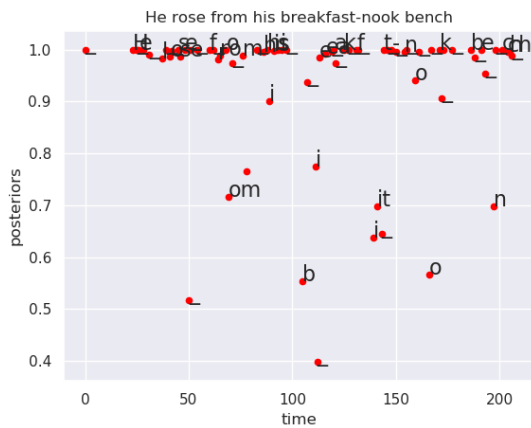
Greedy Decoded : He lose from his beakfit-nook bench



(α') Top 2 Probabilities



(β') Πιθανότητες μεταξύ 0.55-0.98 και η αποκωδικοποίησή



(γ') Greedy Alignments

He lose from his breakfast-nook bench

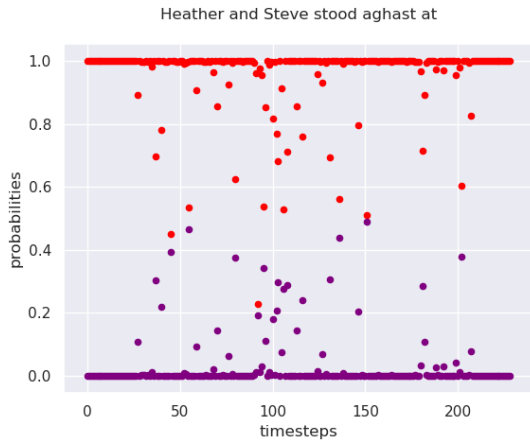
(δ') Εικόνα 1

Σχήμα 1.9: Παράδειγμα 1

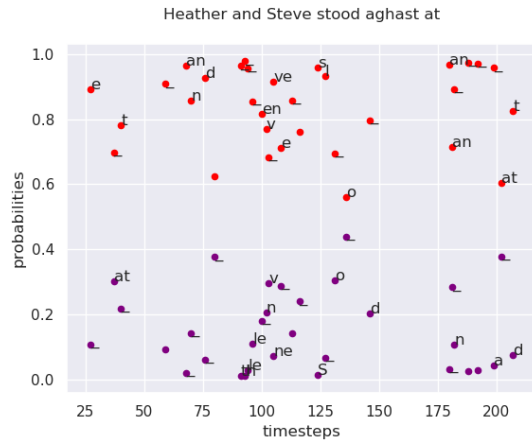
Test Set Example 2

Original : Heather and Steve stood aghast at

Greedy Decoded : Heatler and Meve ood aglan at



(α') Top 2 Probabilities



(β') Πιθανότητες μεταξύ 0.55-0.98 και η αποκωδικοποίησή



(γ') Greedy Alignments

Heatler and Meve ood aglan at

(δ') Εικόνα 2

Σχήμα 1.10: Παράδειγμα 2

Model	Algorithm	WER - CER
Unigrams + 50 Bigrams	Bigram-Scoring	0.2100-0.0626
	Unigram-Bigram Extension	0.2090-0.0591

Πίνακας 1.9: Πειράματα στους προτεινόμενους αλγορίθμους αποκωδικοποίησης για το κοινό μοντέλο Unigram-Bigram

Για την χρήση του CTC αλγορίθμου απαιτείται, πέρα από τις κλάσεις των γραμμάτων που θέλουμε να αναγνωρίσουμε, ένας επιπρόσθετος χαρακτήρας, ο κενός (blank). Ο μοναδικός λόγος χρήσης αυτού του χαρακτήρα στον CTC αλγόριθμο είναι για να επιτρέπεται η έκφραση συνεχόμενων όμοιων χαρακτήρων όπως για παράδειγμα οι χαρακτήρες ll στην λέξη Hello. Αν δεν υπήρχε ο κενός χαρακτήρας τότε το διπλό γράμμα l δεν θα εκφραζόταν αφού θα είχε γίνει ένα με το γειτονικό του κατά τη διάρκεια της απόκτησης της τελικής λέξης από το alignment αυτής. Στην περίπτωση επαύξησης κατανοούμε ότι η ύπαρξη των bigrams δρα ως ένας κενός χαρακτήρας γιατί, τα όποια συνεχόμενα γράμματα υπάρχουν διαχωρίζονται με το αντίστοιχο bigram τους. Για το προαναφερθέν παράδειγμα, το bigram θα είναι το ll.

Γίνεται λοιπόν κατανοητό ότι η χρήση του κανονικού CTC για την αναγνώριση τόσο unigrams όσο και bigrams χαρακτηρίζεται περιττή μιας και τα bigrams δρουν ως κενός χαρακτήρας. Για το λόγο αυτό, σε συνδυασμό με την μη διαφοροποίηση των bigrams από τα γειτονικά τους unigrams σε περίπτωση που κάποιο από αυτά έχει αναγνωριστεί λανθασμένα, μας οδηγούν στο συμπέρασμα ότι η χρήση των bigrams ή όποιων άλλων ngrams, υπό τον κλασικό CTC αλγόριθμο, δεν έχει επίδραση στο ποσοστό αναγνώρισης.

Εκτός όλων των παραπάνω συμπερασμάτων, παρατηρούμε ότι το CTC συγκεντρώνει τη μεγαλύτερα μάζα πιθανότητας μόνο σε ένα συγκεκριμένο μονοπάτι γεγονός που εμποδίζει το exploration που θέλαμε να υλοποιήσουμε στην αρχή.

Συμπεραίνουμε λοιπόν ότι αυτή η προσέγγιση μας δεν αποφέρει βελτίωση υπό αυτή τη μοντελοποίηση. Μια επέκταση αυτής που ενδεχομένως να βοηθούσε είναι η προσέγγιση του CTC όπως και εδώ [42] όπου παρέχεται το Gram-CTC που επιτρέπει κάθε φορά να εκφραστεί μόνο ένα από τα unigrams, bigrams ή γενικά κάποιο ngram.

1.4 Μάθηση Πολλαπλών Στόχων

Στο παρών κεφάλαιο, λαμβάνοντας υπόψιν τα αποτελέσματα του προηγούμενου, αποφασίζουμε να χειριστούμε τα target units πολλαπλής κλίμακας, για παράδειγμα bigrams, trigrams, ngrams, διαφορετικά. Συγκεκριμένα, τα επιλέγουμε ως target units αλλά σε διαφορετικά CTC επίπεδα από τα unigrams. Συνεπώς λοιπόν υλοποιούμε ιεραρχική αρχιτεκτονική πολλαπλών εργασιών (μάθηση unigrams, bigrams κ.ο.κ).

Υπάρχουν δύο βασικές αρχιτεκτονικές με τις οποίες πειραματιστήκαμε. Αυτές είναι η ιεραρχική και η block αρχιτεκτονική πολλαπλών στόχων. Αρχικά χρησιμοποιούμε ως κλάσεις και unigrams και διγράμματα. Τα διγράμματα προκύπτουν ως συνδυασμός των 26 unigrams καθώς είναι αυτά που το μοντέλο μας θέλουμε να μάθει καλύτερα.

Η **Block** αρχιτεκτονική πολλαπλών στόχων είναι η εξής :

Αν συμβολίσουμε με E τα χαρακτηριστικά που εξάγονται από τον CNN Encoder μαζί με το Max Pooling που περιγράφει η βασική αρχιτεκτονική και με X την εικόνα εισόδου, έχουμε :

$$\begin{aligned} E &= \text{SharedCNNEncoder}(X) \\ E_{unigrams} &= \text{BiLSTM}_{unigrams}(E) \\ E_{bigrams} &= \text{BiLSTM}_{bigrams}(E) \\ Y_{unigrams} &= \text{Softmax}(\text{Dense}_{unigrams}(E_{unigrams})) \\ Y_{bigram} &= \text{Softmax}(\text{Dense}_{bigrams}(E_{bigrams})) \end{aligned}$$

Η ιεραρχική αρχιτεκτονική πολλαπλών στόχων είναι η εξής:

$$\begin{aligned} E &= \text{SharedCNNEncoder}(X) \\ E_{unigrams} &= \text{BiLSTM}_{unigrams}(E) \\ E_{intermediate1} &= \text{BiLSTM}_{intermediate1}(E) \\ E_{bigrams} &= \text{BiLSTM}_{bigrams}(E_{intermediate1}) \\ Y_{unigrams} &= \text{Softmax}(\text{Dense}_{unigrams}(E_{unigrams})) \\ Y_{bigram} &= \text{Softmax}(\text{Dense}_{bigrams}(E_{bigrams})) \end{aligned}$$

1.4.1 Πειράματα - Τελική Αρχιτεκτονική

Σε αυτό το σημείο πειραματιζόμαστε με κάποιες υπερπαραμέτρους (layers, units) έτσι ώστε να οριστικοποιήσουμε τη βασική αρχιτεκτονική. Ο CNN αποκωδικοποιητής παραμένει ως έχει όπως και στο βασικό μοντέλο (μάθηση μιας κατηγορίας στόχου). Όσον αφορά το ακολουθιακό μοντέλο από LSTM επίπεδα, αυτά παραμένουν 3, όπου το καθένα αποτελείται από 256 κρυφές μονάδες. Αυτή μας η επιλογή βασίζεται στο γεγονός ότι εφόσον θέλουμε να κάνουμε καθαρή σύγκριση μεταξύ της αρχιτεκτονικής ενός στόχου με τις αρχιτεκτονικές πολλαπλών στόχων, το κοινό τους κομμάτι και των εν συγκρίσει μοντέλων πρέπει να έχει τα ίδια χαρακτηριστικά. Πέρα από αυτό, που αποτελεί κεντρικό μας άξονα, δοκιμάσαμε την αύξηση των κρυφών LSTM μονάδων, για παράδειγμα από 256 σε 300. Ωστόσο δεν παρατηρήσαμε βελτίωση και εν τέλει κρατήσαμε το πλήθος των κρυφών μονάδων στα 256.

Όσον αφορά τις μονάδες στόχου, τις επιλέγουμε με ένα βασικό κριτήριο, τη συχνότητα τους στα δεδομένα εκπαίδευσης. Γενικά όσο αυξάνουμε την κλίμακα και επεκτεινόμαστε σε trigrams, fourgrams και εν γένη ngrams, τόσο μειώνεται η συχνότητα τους στα δεδομένα εκπαίδευσης. Συνεπώς, επιλέγουμε να πειραματιστούμε μέχρι την κλίμακα των fourgrams γιατί από αυτή και πάνω μειώνεται αρκετά το πλήθος του εκάστοτε ngram στα δεδομένα εκπαίδευσης. Όσον αφορά το πλήθος των trigrams και fourgrams επιλέγουμε τα 1000 συχνότερα από ένα κείμενο γιατί ο συνολικός αριθμός όλων είναι απαγορευτικά μεγάλος.

Όσον αφορά τα σχήματα εκπαίδευσης, υπάρχουν δύο τινά. Η πρώτη επιλογή είναι κάθε φορά να επιλέγουμε το δίκτυο να δει μόνο ένα από τα unigrams, bigrams κ.ο.κ. Η δεύτερη επιλογή είναι να βλέπει όλα τα ngrams που έχουμε επιλέξει μαζί μέσω της άθροισης όλων των αντίστοιχων CTC συναρτήσεων κόστους. Επιλέγουμε να πορευθούμε με την δεύτερη επιλογή εκπαίδευσης μιας και οι στόχοι που πρέπει το δίκτυο να μάθει είναι συγγενή και δεν θα προκαλέσουν σύγχυση στο ρύθμισμα των παραμέτρων όπως θα γινόταν στην περίπτωση των αντικρουόμενων στόχων.

Scale	WER Unigram	CER Unigram
Unigrams + Bigrams	0.1772	0.05210
Unigrams + Bigrams + Trigrams	0.1770	0.05372
Unigrams + Bigrams + Trigrams + Fourgrams	0.1758	0.05293

Πίνακας 1.10: Ιεραρχική αρχιτεκτονική πολλαπλών στόχων - Greedy αποκωδικοποίηση

Scale	WER Unigram	CER Unigram
Unigrams + Bigrams	0.1796	0.05280
Unigrams + Bigrams + Trigrams	0.1790	0.05302
Unigrams + Bigrams + Trigrams + Fourgrams	0.1768	0.05183

Πίνακας 1.11: Block αρχιτεκτονική πολλαπλών στόχων με 1 BiLSTM επίπεδο σε κάθε στόχο - Greedy Αποκωδικοποίηση

Επίσης, στο κομμάτι της αρχιτεκτονικής, ύστερα από μια σειρά πειραμάτων που υλοποιήσαμε, καταλήξαμε ότι η ιεραρχική αρχιτεκτονική πολλαπλών εργασιών έχει το σημαντικό μειονέκτημα ότι δεν μπορεί να επεκταθεί, καθώς αυξάνεται πολύ σε βάθος με αποτέλεσμα να δυσχεραίνει την συνολική εκπαίδευση και φυσικά την σύγκλιση. Επίσης, από τα παραπάνω πειράματα είναι ξεκάθαρο ότι δεν υπάρχει κάποια σημαντική διαφορά μεταξύ των δύο αρχιτεκτονικών, με την block να καταλαμβάνει πολύ λιγότερες παραμέτρους. Συνεπώς, καταλήγουμε στην block αρχιτεκτονική πολλαπλών εργασιών.

Επόμενο βήμα για την περαιτέρω βελτιστοποίηση της αρχιτεκτονικής μας είναι η αφαίρεση των επιπέδων που αφορούν τις επιμέρους εργασίες. Επικεντρωνόμαστε στην περίπτωση των unigrams και bigrams όπου η κοινή πληροφορία μεταξύ τους είναι αρκετά μεγάλη μιας και ένας από τους δύο χαρακτήρες είναι κοινός.

Η τελική μας αρχιτεκτονική είναι η block πολλαπλών εργασιών χωρίς συγκεκριμένα ανά εργασία επίπεδα.

$$E = \text{SharedCNNBiLSTMEncoder}(X)$$

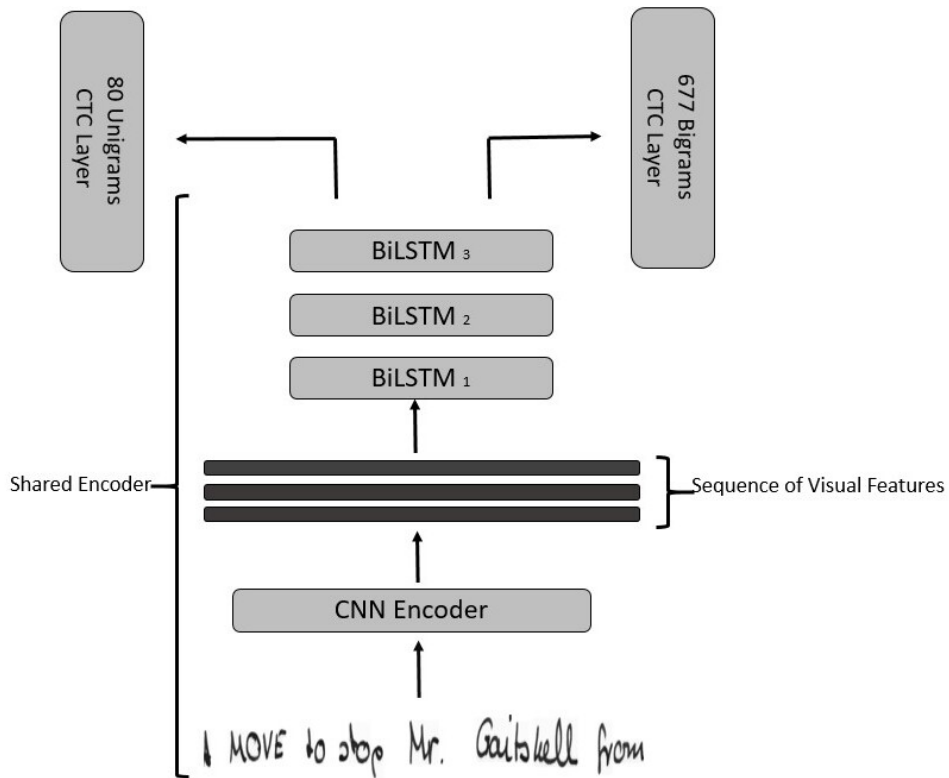
$$Y_{\text{unigrams}} = \text{Dense}_{\text{unigrams}}(E)$$

$$Y_{\text{bigrams}} = \text{Dense}_{\text{bigrams}}(E)$$

Λαμβάνουμε τα εξής αποτελέσματα :

Scale	WER Unigram	CER Unigram
Unigrams + Bigrams	0.1774	0.05218

Πίνακας 1.12: Block Multitask αρχιτεκτονική δίχως task-specific LSTM επίπεδο - Greedy Αποκωδικοποίηση



Σχήμα 1.11: Τελική αρχιτεκτονική Block-πολλαπλών στόχων χωρίς BiLSTM επίπεδο σε κάθε στόχο

1.4.2 Σύγκριση Μοντέλων Μιας Εργασίας με Μοντέλα Πολλαπλών

Σε αυτό το σημείο θα συγκρίνουμε τη βασική μας αρχιτεκτονική που αφορά την αναγνώριση ως προς ένα task, τα unigrams, με τις αρχιτεκτονικές με περισσότερα από ένα task, όπως για παράδειγμα τα unigram, bigrams. Συγκεκριμένα, συγκρίνουμε τις block αρχιτεκτονικές με τη βασική αρχιτεκτονική που παρουσιάζεται στο Κεφάλαιο 5.

Layers	3
Epochs	70
Early Stopping	25
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	3 BiLSTM layers
batch size	8

Πίνακας 1.13: Διαρρύθμιση πειραμάτων για τη σύγκριση αρχιτεκτονικής πολλαπλών εργασιών με αρχιτεκτονική μιας εργασίας

Για την υλοποίηση του early stopping, απαιτούμε ότι αν το μοντέλο δεν βρει καλύτερο word error rate και character error rate από το ήδη υπάρχον, η εκπαίδευση να σταματήσει. Αυτός ο έλεγχος γίνεται ανά 5 εποχές.

Λαμβάνουμε λοιπόν τα παρακάτω αποτελέσματα :

Architecture	WER	CER	parameters M
STL	0.1814	0.0564	5.3
BMTL (UB)	0.1669	0.0515	7.85
BMTL (UBT)	0.1669	0.0532	10.45
BMTL (UBTF)	0.1668	0.0519	13.05

Πίνακας 1.14: STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων σε επίπεδο unigram - CTC 4-Gram Char LM

Architecture	WER	CER	parameters M
STL	0.1814	0.0564	5.3
Linear BMTL (UB)	0.1672	0.0528	5.68

Πίνακας 1.15: STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων σε επίπεδο unigram - CTC 4-Gram Char LM

Έπειτα συγκρίνουμε την αρχιτεκτονική μιας εργασίας στη [53] με την δική μας πολλαπλών εργασιών.

Όλες οι πιο πρόσφατες state-of-the-art προσεγγίσεις είναι αρχιτεκτονικές μιας εργασίας/στόχου χρησιμοποιώντας μόνο unigrams ως κλάσεις στο CTC επίπεδο. Παρ' όλα αυτά, είναι βέβαιο, μέσα από τα πειράματά μας, ότι επιπλέον κλάσεις υψηλότερης κλίμακας (bigrams, trigrams κ.ο.κ) μπορούν να συνεισφέρουν στην διαδικασία μάθησης εμπλουτίζοντας τον encoder με περισσότερη πληροφορία και τελικά να οδηγηθεί σε καλύτερα οπτικά χαρακτηριστικά.

Architecture	WER	CER	parameters M
STL	0.1481	0.046	5.3
Linear BMTL (UB)	0.1392	0.0460	5.68

Πίνακας 1.16: STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων σε επίπεδο unigram - CTC Word LM 4-Gram

Architecture	% WER	% CER	parameters (M)
STL [53]	20.20	6.20	9.30
BMTL Linear CTC Layer (UB)	17.74	5.21	5.68

Πίνακας 1.17: STL-vs-MTL αρχιτεκτονικές για αναγνώριση χειρόγραφων με greedy αποκωδικοποίηση επιπέδου χαρακτήρων

Από τη μελέτη και τα πειράματα της δικής μας προσέγγισης συνοψίζουμε τα παρακάτω :

- Χρησιμοποιούμε μεγαλύτερο συνελικτικό δίκτυο από το [53]. Προτιμούμε να δώσουμε περισσότερο βάρος στο συνελικτικό (CNN) απότι στο ακολουθιακό LSTM που είναι πιο υπολογιστικά βαρύ.
- Υλοποιούμε το Map-to-Sequence με χρήση Max-Pooling στη διάσταση του ύψους για κάθε χάρτη ενεργοποίησης. Στην βιβλιογραφία, αυτή η λειτουργία υλοποιείται σειριακή σύνδεση όλων των στηλών όλων των χαρτών ενεργοποίησης με αποτέλεσμα να επιβαρύνεται πολύ περισσότερο το 1ο LSTM επίπεδο. Με την δική μας τεχνική καταλήγουμε να έχουμε ένα διάγραμμα χαρακτηριστικών για κάθε στήλη του χάρτη ενεργοποίησης.
- Ενσωματώνουμε επιπλέον εξωτερική γνώση στο δίκτυο μέσω των διγραμμάτων τα οποία οφελούν το δίκτυο όχι μόνο στην διαδικασία μάθησης αλλά και στη διαδικασία αποκωδικοποίησης.
- Αποφεύγουμε το πρόβλημα της συχνότητας των υψηλότερων κλάσεων (trigrams, fourgrams) επιλέγοντας τα 1000 συχνότερα από αυτά.

1.5 Πλήρως Συνελικτική Αρχιτεκτονική με CTC για Αναγνώριση Χειρόγραφων

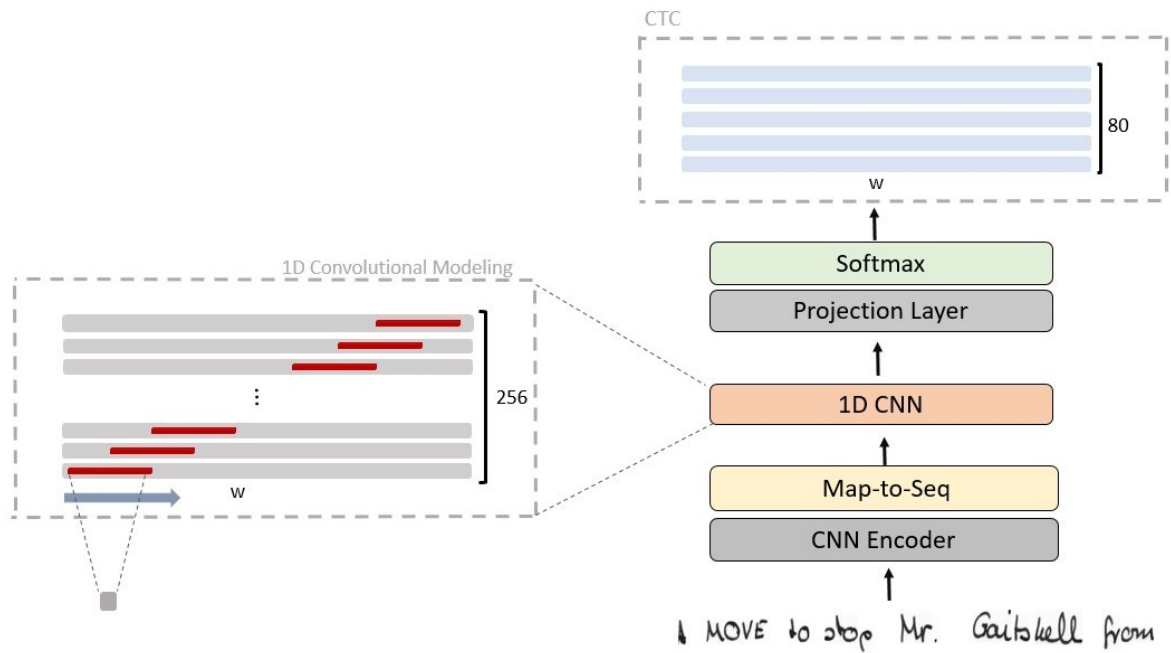
Ο CTC χρησιμοποιείται ευρέως για εργασίες μονοτονικής μετατροπής ακολουθίας όπως η αναγνώριση χειρόγραφων κειμένων και η αυτόματη αναγνώριση φωνής. Είναι ένας αλγόριθμος που επιτρέπει την αποκωδικοποίηση ακολουθιών χωρίς να γνωρίζει την ακριβή τοποθεσία έκφρασης του κάθε χαρακτήρα, είτε πρόκειται για αναγνώριση φωνής είτε για αναγνώριση κειμένων. Ο CTC χρησιμοποιείται συνήθως πάνω από ένα αναδρομικό RNN ή LSTM δίκτυο. Στο παρόν κεφάλαιο εξερευνούμε την συνεργία μεταξύ CNN και CTC για την δημιουργία ενός μοντέλου όπου τόσο το Οπτικό κομμάτι όσο και το Ακολουθιακό θα είναι κατασκευασμένο από συνελίξεις. Παρακάτω, συγκρίνουμε την απόδοση του CNN σε σύγκριση με τα LSTM με βάση τα κριτήρια του ποσοστού αναγνώρισης, του χρόνου εκπαίδευσης και του χρόνου αποκωδικοποίησης. Καταλήγουμε ότι παρόλο που το CTC δεν φτάνει το ποσοστό αναγνώρισης του των τυπικών LSTM μοντέλων, είναι πολύ πιο εύκολα να εκπαιδευτούν και να αποκωδικοποιηθούν γρηγορότερα.

Υπάρχουν πολλές καταγεγραμμένες προσπάθειες όπου η γίνεται αντικατάσταση των LSTMs με απόλυτα συνελικτικές αρχιτεκτονικές κυρίως στον τομέα της αναγνώρισης φωνής [65] [79] [75] [41]. Στον τομέα της αναγνώρισης χειρόγραφων κειμένων δεν έχει γίνει κάποια αντίστοιχη εξερεύνηση σε πλήρεις συνελικτικές αρχιτεκτονικές. Για το λόγο αυτό, εκμαιεύουμε την ευκαιρία και πειραματιζόμαστε με πλήρως συνελικτικές αρχιτεκτονικές.

1.5.1 Πλήρως Συνελικτική Αρχιτεκτονική

Πριν εξηγήσουμε την προτεινόμενη αρχιτεκτονική μας, χρειάζεται πρώτα να εξηγήσουμε γιατί μπορούμε να χρησιμοποιήσουμε Connectionist Temporal Classification με τη χρήση συνελικτικών δικτύων. Για να το απαντήσουμε αυτό πρέπει πρώτα να συλλογιστούμε τι απαιτεί ως είσοδο ο CTC αλγόριθμος. Αυτό είναι μια ακολουθία κατανομών πιθανότητας σε ένα σύνολο χαρακτήρων εξόδου. Στα LSTM αυτή η κατανομή αποκτάται με προβολή στις κλάσεις εξόδου και softmax, ώστε να αποκτήσουμε πιθανότητες, σε κάθε ένα από τα διανύσματα που προκύπτουν από προβολή. Με την ίδια ακριβώς λογική θα καταστρώσουμε την συνεργία μεταξύ του CTC και του συνελικτικού δικτύου.

Αν θέλουμε να αποφύγουμε τη χρήση LSTM για την μοντελοποίηση των χρονικών εξαρτήσεων μεταξύ των διανυσμάτων χαρακτηριστικών θα χρειαστεί να βρούμε έναν άλλο τρόπο να μοντελοποιήσουμε τις χρονικές εξαρτήσεις μεταξύ αυτών. Τα διανύσματα χαρακτηριστικών είναι ουσιαστικά εικόνες διάστασης $(1, w)$ που είναι οργανωμένες σε 256 κανάλια. Κάθε ένα από αυτά τα κανάλια έχει κωδικοποιημένα χαρακτηριστικά από το πλάτος της εικόνας που πρέπει κατά ένα τρόπο να συσχετιστούν. Είναι γνωστό ότι η πράξη του της συνελίξης όπως ορίζεται στα πλαίσια του συνελικτικού δικτύου είναι μια πράξη αυτοσυσχέτισης. Για αυτό το λόγο χρησιμοποιούμε συνελίξεις μιας διάστασης, δηλαδή πυρήνα σχήματος $(1, k)$ που υπολογίζει αυτοσυσχετίσεις σε όλο το μήκος κάθε διανύσματος και για τα 256 διανύσματα.



Σχήμα 1.12: Αναπαράσταση της CNN-CTC αρχιτεκτονικής. Ο CNN Encoder παραμένει ως έχει και η μόνη αλλαγή που πραγματοποιείται στο δίκτυο είναι η αντικατάσταση των BiLSTM επιπέδων με χρήση 1-D συνελίξεων

2 Conv Layers - 32 Kernels 2x2 - ReLU
Max Pooling
4 Conv Layers - 64 Kernels 2x2 - ReLU
Max Pooling
6 Conv Layers - 128 Kernels 2x2 - ReLU
Max Pooling
2 Conv Layers - 256 Kernels 2x2 - ReLU
Max Pooling - Kernel Size 256x1
1-D Convolutional Temporal Modeling

Πίνακας 1.18: CNN-CTC αρχιτεκτονική

1.5.2 Πειράματα

Εκπαιδεύουμε το CNN-CTC δίκτυο με την παρακάτω διαμόρφωση :

Χρησιμοποιούμε early stopping βασισμένο στο WER/CER του validation set. Ανά 5 εποχές απαιτούμε το ποσοστό λαθών να είναι αυστηρά μικρότερο από το καλύτερο που έχει σημειωθεί. Δίνουμε στο δίκτυο 5 ευκαιρίες να βρει το μικρότερο ποσοστό λαθών. Αν δεν το βρει μέσα σε 5 ελέγχους τότε η διαδικασία εκπαίδευσης σταματά.

Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	None
batch size	8

Πίνακας 1.19: Διαμόρφωση Εκπαίδευσης

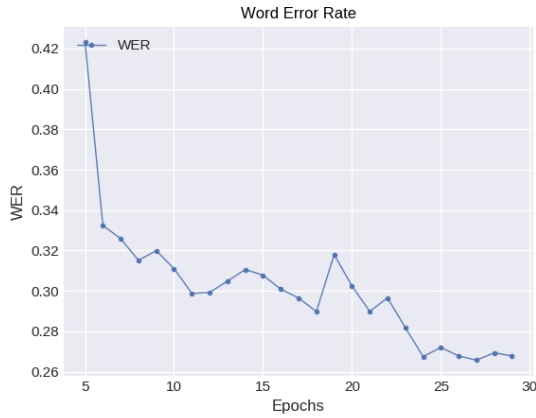
Data Augmentation	WER	CER
None	0.3003	0.0891
Global Affine	0.2670	0.0778
Global Morphological	0.3182	0.0976
Global Affine + Global Morphological	0.3032	0.0911

Πίνακας 1.20: Πειράματα δυναμικής επαύξησης δεδομένων σε πλήρως συνελκτικές αρχιτεκτονικές

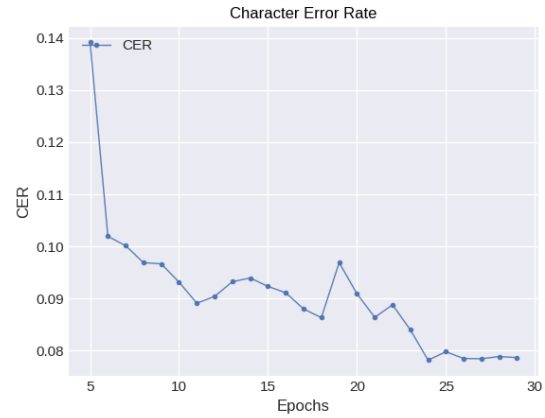
Στα πειράματα αποκωδικοποίησης χρησιμοποιούμε στατιστικά γλωσσικά μοντέλα σε επίπεδα χαρακτήρων και λέξεων. Τα μοντέλα αυτά παράγονται από τις Brown και LOB βάσεις κειμένων με τη χρήση του εργαλείου KenLM [36].

Decoding Algorithm	WER	CER
Greedy	0.2693	0.0794
CTC Beam Search 2 Char-LM	0.2680	0.0785
CTC Beam Search 3 Char-LM	0.2577	0.0753
CTC Beam Search 4 Char-LM	0.2452	0.0732
CTC Beam Search 4-Word LM	0.1620	0.0536

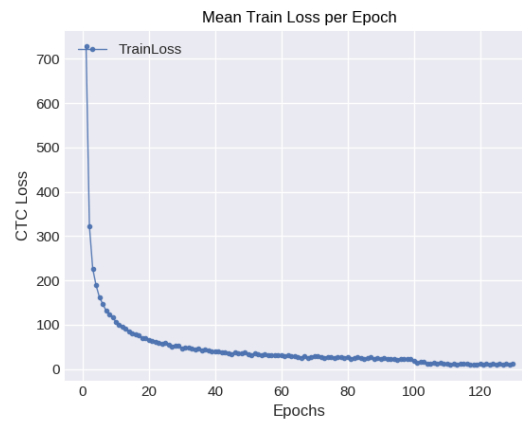
Πίνακας 1.21: Αποτελέσματα αποκωδικοποίησης με χρήση εξωτερικού γλωσσικού μοντέλου - επίπεδο γραμμής



(α) Word Error Rate in Validation



(β) Character Error Rate in Validation



(γ) Mean Train Loss per Epoch

1.5.3 Συνδυασμός Συνελικτικών μοντέλων

Υλοποιούμε ένα απλό late fusion σχήμα όπου συνδυάζουμε εξόδους συνελικτικών δικτύων όπου έχουν εκπαιδευτεί ξεχωριστά. Επειδή κάθε ένα από τα μοντέλα μαθαίνει κάθε φορά κάποια μονοπάτια καλύτερα από τα άλλα, συνδυάζοντας τις εξόδους όλων μαζί λαμβάνουμε υπόψη όλες επιλογές που μπορεί να μην έχουν γίνει σωστά από το ένα δίκτυο αλλά να έχουν γίνει σωστά από τό άλλο. Ο συνδυασμός γίνεται ουσιαστικά με το να πάρουμε την μέση τιμή όλων των τελικών πινάκων ενεργοποίησης και έπειτα να εφαρμόσουμε σε αυτό επίπεδο softmax ώστε να λάβουμε κατανομές πιθανοτήτων σε κάθε χρονική στιγμή.

Model	WER	CER
cnn-model 1	0.2689	0.0788
cnn-model 2	0.2691	0.0799
cnn-model 3	0.2628	0.0771
cnn-model 4	0.2890	0.0865
cnn-ensemble	0.2443	0.0704

Πίνακας 1.22: Επίδοση των απλών συνελικτικών δικτύων και του συνδυασμού αυτών

Παρόλο που ο συνδυασμός των συνελικτικών μοντέλων δεν ξεπερνά σε ποσοστό αναγνώρισης το βασικό μοντέλο, παρατηρούμε ότι προσδίδει σημαντική βελτίωση σε σχέση με τα απλά συνελικτικά μοντέλα.

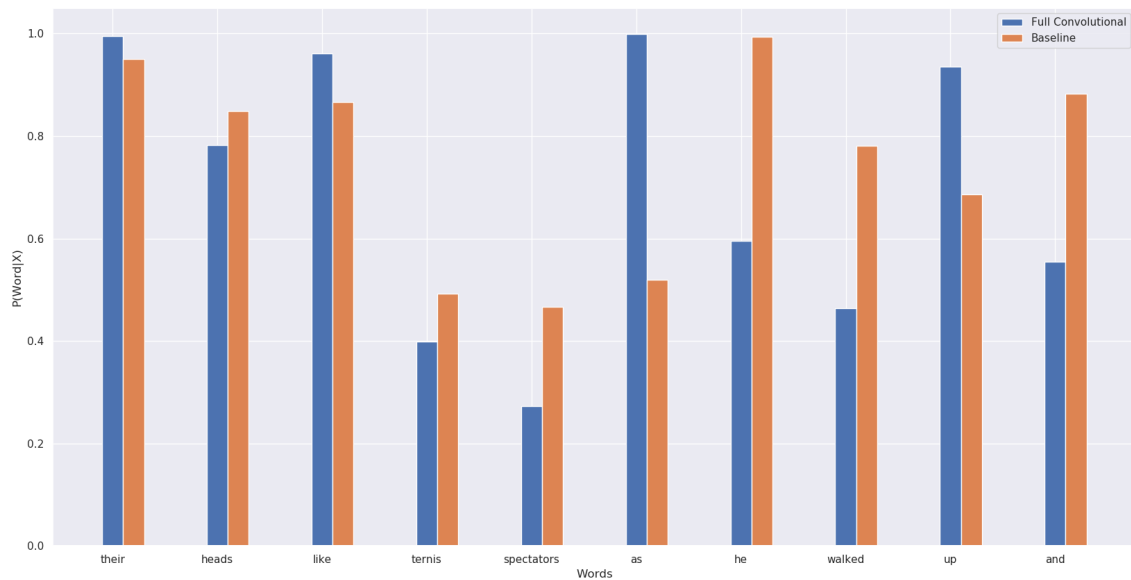
Ensemble Technique	WER	CER
averaging	0.1514	0.0520
single CNN-CTC	0.1620	0.0536
single CNN-LSTM-CTC	0.1490	0.0460

Πίνακας 1.23: Σύγκριση επίδοσης απλών μοντέλων με Ensemble μοντέλα - Char/Word LM αποκωδικοποίηση

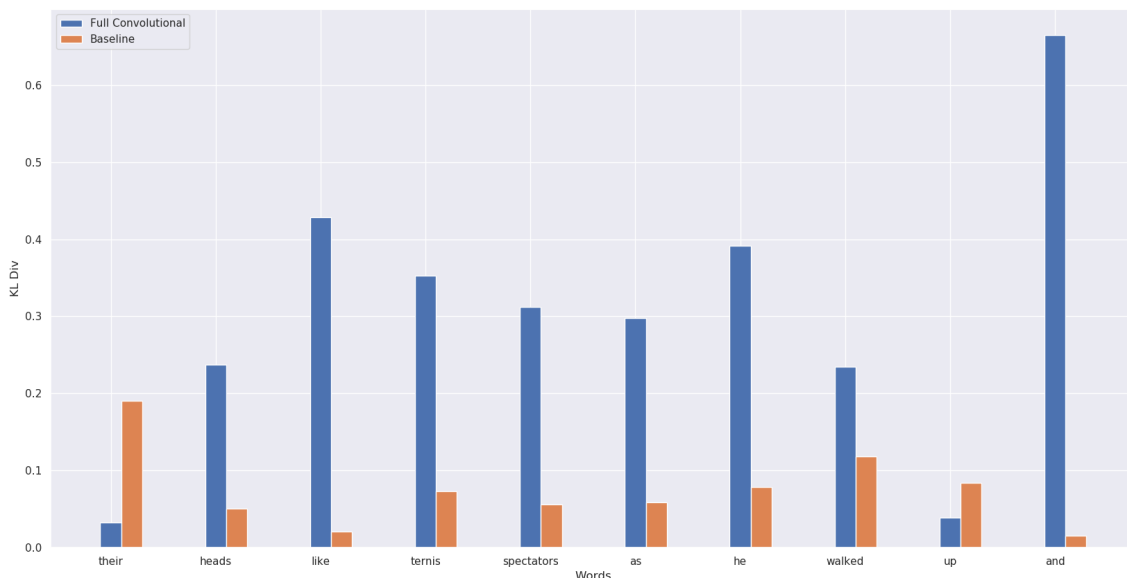
1.5.4 Σύγκριση CNN+CTC με το βασικό μοντέλο

Model	train time(ς)	inference time	Parameters (M)	WER	CER
CNN-LSTM-CTC	370.1	102.1	5.73	0.1481	0.046
CNN-CTC	149.3	22.2	2.34	0.1620	0.0536

Πίνακας 1.24: Σύγκριση των αρχιτεκτονικών CNN-LSTM-CTC και CNN-CTC



Σχήμα 1.13: Πιθανότητες που το κάθε μοντέλο αναθέτει σε κάθε λέξη της πρότασης.



Σχήμα 1.14: Απόκλιση Kullback-Leibler λέξεων όμοια αναγνωρισμένων από τα μοντέλα CNN, LSTM με σκοπό τη σύγκριση αυτών

Το παραπάνω παράδειγμα αποδεικνύει ότι ένα καλό ποσοστό αναγνώρισης δεν σημαίνει απαραίτητα ότι το μοντέλο έχει μοντελοποιήσει μια καλή αναπαράσταση της κατανομής πιθανότητας. Τα δύο μοντέλα, παρόλο που βγάζουν το ίδιο σωστό αποτέλεσμα, το CNN-LSTM αποδεικνύεται ότι έχει μάθει μια καλύτερη αναπαράσταση της κατανομής αναφοράς επειδή έχει χαμηλότερη KL απόκλιση.

Model	% WER/CER Greedy	% WER/CER WordLM	WER/CER % Improvement
CNN-LSTM-CTC	20.68/6.8	14.81/4.60	28.3 /32.4
MTL	17.74/5.22	13.92/4.63	21.53 /11.30
CNN-CTC	26.7/7.77	16.20/5.30	39.33/32.08

Πίνακας 1.25: Σύγκριση μεταξύ CNN-LSTM-CTC και CNN-CTC

Από τον παραπάνω πίνακα παρατηρούμε ότι η μεγαλύτερη βελτίωση λόγω της CTC απόκωδικοποίησης με χρήση εξωτερικού γλωσσικού μοντέλου σημειώνεται στο CNN-CTC μοντέλο. Αυτή η παρατήρηση σε συνδυασμό με τη γραφική παράσταση της KL απόκλισης μας οδηγεί στο συμπέρασμα ότι το γλωσσικό μοντέλο επιπέδου λέξης έχει μεγαλύτερη επίδραση στο CNN-CTC δίκτυο. Αυτό σημαίνει ότι με το CNN-CTC μοντέλο εναλλακτικά μονοπάτια μπορούν εύκολα να ανακαλυφθούν διότι το μοντέλο δεν συγκεντρώνει τελείως την πιθανότητα σε ένα συγκεκριμένο μονοπάτι. Το βάρος του γλωσσικού μοντέλου έχει πιο πολύ επίδραση σε αυτή την περίπτωση.

1.5.5 Συμπεράσματα

Σε αυτό το κεφάλαιο κινηθήκαμε ένα βήμα πριν την κατεύθυνση να φτιάξουμε πλήρως συνελικτική αρχιτεκτονική για το πρόβλημα της Αναγνώρισης Χειρόγραφων Κειμένων. Το CNN-CTC μοντέλο μας είναι ελαφρώς πιο πίσω στα ποσοστά αναγνώρισης από τα αντίστοιχα των CNN-LSTM-CTT. Το μοντέλο μας ωστόσο παρουσιάζει **60 %** λιγότερο χρόνο εκπαίδευσης ανά эпоχή, **78 %** μείωση στον χρόνο inference και έχει **60 %** λιγότερες παραμέτρους από τον CNN-LSTM-CTC μοντέλο. Επίσης, η πλήρως συνελικτική αρχιτεκτονική ανοίγει τον δρόμο για περαιτέρω εξερεύνηση τεχνικών για βελτίωση του ποσοστού αναγνώρισης. Το γεγονός ότι τα

συνελικτικά δίκτυα είναι οικονομικά από θέμα χρόνου εκπαίδευσης και χρόνου inference μας δίνει τη δυνατότητα να κάνουμε μια επιπλέον διερεύνηση σε μεθόδους ensembling. Εμείς πειραματιστήκαμε με το bagging (μέσος όρος των εξόδων των επιμέρους μοντέλων. Σε αυτή την περίπτωση είδαμε ποσοστό μείωσης **3.5 %** στο Word Error Rate και **3 %** στο **Character Error Rate**

1.6 Συνεισφορές, Συμπεράσματα και Μελλοντική Δουλειά

Η παρούσα διπλωματική εργασία μελέτησε διεξοδικά την σύνθεση της δομής του CTC επιπέδου με σκοπό να καθορίσει τη δυναμική της κάθε επιλογής να βελτιώσει το ποσοστό αναγνώρισης. Η πρώτη μας προσέγγιση αφορούσε την μάθηση, από το δίκτυο, σύνθετων στόχων αποτελούμενα από unigrams και διγράμματα. Αρχικός σκοπός αυτής της μελέτης είναι κατά πόσο τα διγράμματα μπορούν να συνεισφέρουν στην βελτίωση του ποσοστού αναγνώρισης μέσα από τη διόρθωση κάποιων λανθασμένων unigrams. Η αρχική μας υπόθεση που στήριξε το κίνητρό μας για αυτή τη προσέγγιση είναι ότι κάποιοι χαρακτήρες μπορεί να είναι ευκολότερα αναγνωρίσιμοι όταν είναι σε ζεύγος απότι μόνοι τους. Σε αυτό το πλαίσιο, αναπτύξαμε δύο βασικούς αλγόριθμους αποκωδικοποίησης που ενσωματώνουν την πληροφορία των διγραμμάτων με διαφορετικό τρόπο σε κάθε περίπτωση. Η πρώτη προσέγγιση χρησιμοποιεί τις πιθανότητες των διγραμμάτων ως σκορ επέκτασης της ακολουθίας κατά ένα γράμμα. Ουσιαστικά αυτή η προσέγγιση αντιμετωπίζει αυτές ως ένα στατιστικό γλωσσικό μοντέλο. Η δεύτερη προσέγγιση χειρίζεται τα bigrams όπως ακριβώς τα unigrams στον κλασικό αλγόριθμο CTC Beam Search Decoding. Εμείς υλοποιούμε ουσιαστικά μια επέκταση αυτού του αλγόριθμου ώστε να υποστηρίζει την επέκταση της αναπτυσσόμενης ακολουθίας όχι μόνο από unigrams αλλά και διγράμματα. Τα πειράματά μας ωστόσο έδειξαν ότι τα διγράμματα δεν προσφέρουν βελτίωση στο ποσοστό αναγνώρισης. Αφενός αυτό οφείλεται στο ότι οι περιπτώσεις που το ενδιάμεσο δίγραμμα δεν είναι διαφορετικό από τα δύο γειτονικά του unigram. Αυτό σημαίνει ότι ακόμα και αν τα γειτονικά unigram ήταν λάθος τότε και το ενδιάμεσο δίγραμμα θα είναι και αυτό λανθασμένο. Αφετέρου παρατηρήσαμε ότι το CTC συγκεντρώνει την μεγαλύτερη μάζα πιθανότητας σε ένα μόνο μονοπάτι, μειώνοντας έτσι τη δυνατότητα περαιτέρω περιήγησης σε μονοπάτια χαμηλότερης πιθανότητας που είναι ενδεχομένως σωστά και θα μπορούσαν κάπως να εκφραστούν.

Τα συμπεράσματα μας από την προηγούμενη ιδέα μας οδήγησαν να διαχειριστούμε με διαφορετικό τρόπο τα διγράμματα. Θεωρήσαμε λοιπόν ότι τα unigrams και τα διγράμματα θα έπρεπε να ήταν σε ξεχωριστά επίπεδα, οπότε και οδηγηθήκαμε στη μάθηση πολλαπλών στόχων. Δεδομένου ότι μια τέτοια μελέτη δεν είχε γίνει στον τομέα των χειρόγραφων κειμένων, παρά μόνο στον τομέα της αυτόματης αναγνώρισης φωνής, αδράξαμε την ευκαιρία να πειραματιστούμε με τέτοιες αρχιτεκτονικές. Ύστερα από εκτενή μελέτη και πειράματα, καταλήξαμε σε αρχιτεκτονική που συνδυάζει τις δύο κατηγορίες κλάσεων σε ξεχωριστά CTC επίπεδα. Συγκρίνοντας την αρχιτεκτονική μας με την αντίστοιχη αρχιτεκτονική μιας εργασίας που είχαμε υλοποιήσει σε προηγούμενο κεφάλαιο και είδαμε ότι με ελάχιστα παραπάνω παραμέτρους το μοντέλο μας λαμβάνει σημαντική βελτίωση στο ποσοστό αναγνώρισης. Συγκρίνοντας την πολλαπλών στόχων αρχιτεκτονική μας με ήδη υπάρχουσα δημοσιευμένη έρευνα [53] καταλήξαμε ότι η δική μας, με λιγότερες παραμέτρους συνολικά, εκμαιεύοντας πληροφορία τόσο από unigrams όσο και από bigrams αποφέρει καλύτερα αποτελέσματα με ποσοστό βελτίωσης 15% του ποσοστού αναγνώρισης της ενός στόχου αρχιτεκτονικής. Συμπεραίνουμε, λοιπόν, ότι μέσω αυτής της αρχιτεκτονικής μπορούμε να εκμεταλευτούμε τα διγράμματα δύο φορές. Αφενός η μια αφορά τη χρήση τους από το εξωτερικό στατιστικό γλωσσικό μοντέλο κατά τη διάρκεια της αποκωδικοποίησης. Αφετέρου με το να τα εισάγουμε στο CTC επίπεδο ως ένα επιπλέον στόχο πετυχαίνουμε την ενσωμάτωση εξωτερικής γνώσης στο μοντέλο.

Στο τελευταίο κεφάλαιο της παρούσας δουλειάς υλοποιούμε μια πλήρως συνελικτική αρχιτεκτονική. Έχοντας κατανοήσει την δυσκολία που παρουσιάζουν τα Αναδρομικά Νευρωνικά Δίκτυα στο να εκπαιδευτούν και να συγκλίνουν, θελήσαμε να απαλλάξουμε την συνολική μας αρχιτεκτονική από αυτά. Οι πλήρως συνελικτικές αρχιτεκτονικές έχουν μελετηθεί εκτενώς στον τομέα της Αναγνώρισης Φωνής. Ωστόσο μια τέτοια αρχιτεκτονική δεν έχει δοκιμαστεί στον τομέα των Χειρόγραφων κειμένων και για το λόγο αυτό αδράξαμε την ευκαιρία να την υλοποιήσουμε. Αρχικά, οδηγηθήκαμε στο συμπέρασμα ότι ο CTC αλγόριθμος μπορεί να χρησιμοποιηθεί πάνω από ένα πλήρως συνελικτικό δίκτυο. Αυτό γιατί μοντελοποιούμε πλέον τους χρονικές εξαρτήσεις μεταξύ των διανυσμάτων χαρακτηριστικών με χρήση συνελίξεων μιας διάστασης. Η βελτίωση που αποφέρει σε μνήμη και σε ταχύτητα εκπαίδευσης και σύγκλισης μας επιτρέπει να πειραματιστούμε με τεχνικές ensembling και συγκεκριμένα λαμβάνοντας και παίρνοντας μέση τιμή από τις εξόδους πολλαπλών συνελικτικών μοντέλων. Συμπερασματικά, το

πλήρες συνελικτικό μας μοντέλο, σε συνδυασμό, με ένα στατιστικό γλωσσικό μοντέλο στο στάδιο της CTC αποκωδικοποίησης μας δίνει αποτελέσματα πολύ κοντινά με αυτά του BiLSTM αλλά παράλληλα είναι πολύ πιο γρήγορο τόσο στο χρόνο εκπαίδευσης όσο και στο χρόνο αποκωδικοποίησης και φυσικά πολύ μικρότερο σε παραμέτρους.

Η ανάπτυξη του πλήρως συνελικτικού μοντέλου μας ανοίγει το δρόμο για μελλοντική δουλειά πάνω σε μελέτη των μηχανισμών Self-Attention που χρησιμοποιούνται στα συνελικτικά δίκτυα [77]. Μια τέτοια προσέγγιση μας ανοίγει τον δρόμο για την εφαρμογή πιο συγχρονων μοντέλων μετατροπής ακολουθιών σε ακολουθίες όπως είναι οι Transformers [72] αντικείμενο που έχει εξαιρετικό ερευνητικό ενδιαφέρον διότι ένα τέτοιο μοντέλο δεν έχει εφαρμοστεί σε κάποιο πρόβλημα της Όρασης Υπολογιστών. Το πρόβλημα της αναγνώρισης κειμένων ίσως είναι μια καλή αρχή για να υλοποιηθεί κάτι τέτοιο.

Επίσης, υπάρχουν πολλές κατευθύνσεις έρευνας εν εξελίξει που θέλουν βελτίωση. Μια από αυτές αφορά το κομμάτι του post-processing του αποκωδικοποιημένου κειμένου. Για παράδειγμα η χρήση δικτύου κωδικοποιητή-αποκωδικοποιητή βασισμένο είτε σε συνελικτικά (με τη λογική του denoising autoencoder) είτε σε LSTM με ή χωρίς attention για την εργασία της αναγνώρισης λέξεων με σκοπό την διόρθωση λαθών αναγνώρισης. Τέτοιες δουλειές έχουν γίνει στα πλαίσια της αναγνώρισης τυπομένου κειμένου. Πιστεύουμε ότι η μεγαλύτερη πρόκληση που αντιμετωπίσαμε σε αυτά τα μοντέλα αφορά το κομμάτι της παραγωγής δεδομένων για την εκπαίδευση μοντέλων που θα υλοποιήσουν το text denoising. Παρόλο που υπάρχουν κάποια βασικά λάθη που παρατηρούνται στα χειρόγραφα κείμενα, τα υπόλοιπα λάθη κατανέμονται τυχαία με αποτέλεσμα η αποτύχια των μοντέλων που έχουμε δοκιμάσει έως τώρα σε αυτό το task να οφείλεται στα από εμάς κατασκευασμένα δεδομένα εκπαίδευσης. Μια πιο προσεκτική μελέτη στη διαδικασία παραγωγής των δεδομένων μας, τόσο από θέμα ποιότητας όσο και ποσότητας, είναι σίγουρα μια από τις μελλοντικές κατευθύνσεις μας.

Μια επιπλέον κατεύθυνση είναι η περαιτέρω βελτιστοποίηση των τοπικών μετασχηματισμών για την δυναμική επαύξηση δεδομένων. Πιστεύουμε ότι ο λόγος που δεν δίνουν αρκετή βελτίωση είναι γιατί δεν είναι κατά κάποιο τρόπο ελεγχόμενοι οι παράμετροι μετασχηματισμών που λαμβάνουν χώρα στις εικόνες, παρόλο τη μελετημένη έκταση των παραμέτρων αυτών. Αυτό έχει ως αποτέλεσμα η εικόνα να αλλοιώνεται σε βαθμό που δυσχεραίνει το δίκτυο στην εκπαίδευση. Μια λύση σε αυτό θα ήταν προσέγγιση δυναμικής επαύξησης δεδομένων με χρήση νευρωνικού δικτύου που θα συμπεραίνει κάθε φορά τις παραμέτρους των μετασχηματισμών με κριτήριο την βελτιστοποίηση του ποσοστού αναγνώρισης.

Τέλος, μια εναλλακτική κατεύθυνση στην οποία έχουμε ήδη εργαστεί και εξάγει κάποια συμπεράσματα είναι αυτή της χρήσης του μοντέλου BERT [20] για την διόρθωση σημασιολογικών λαθών που γίνονται κατά την αποκωδικοποίηση. Πιστεύουμε ότι η ενσωμάτωση σημασιολογικής πληροφορίας μπορεί να επιφέρει βελτίωση στα ποσοστά αναγνώρισης του κειμένου. Μάλιστα, έχουμε συλλέξει παραδείγματα που μας το επιβεβαιώνουν αυτό. Για την επιτυχή εφαρμογή του BERT καλούμαστε να ξεπεράσουμε κάποιες προκλήσεις όπως για παράδειγμα να αντιμετωπίσουμε το γεγονός ότι ένα προεκπαιδευμένο μοντέλο BERT έχει εκπαιδευτεί σε ένα κλειστό λεξικό το οποίο δεν συνάδει με το δικό μας. Κάτι τέτοιο θα μπορούσε να αποφευχθεί με την επανεκπαίδευση του μοντέλου BERT σε κείμενα που έχουμε ήδη χρησιμοποιήσει για την εξαγωγή των στατιστικών μοντέλων. Μια εναλλακτική προσέγγιση είναι να χρησιμοποιήσουμε την αρχιτεκτονική του BERT ώστε να κατασκευάσουμε μια ενιαία αρχιτεκτονική αναγνώρισης και σημασιολογικής διόρθωσης.

Chapter 2

Introduction

2.1 Problem Definition

Handwriting Recognition is the process of transforming a digital representation of the physical result of handwriting into a digital text, generally for further treatments, such as indexing, classification, or translation. One may acquire handwritten text in different manners. For example, with the advent of tablets, touchscreens or digital pens, it is now possible to have access to many physical parameters of the writing process. Therefore, we can know the pen position at every time, and possibly the pen pressure, inclination, and so on. On the other hand, without such tools, we may only have the result of handwriting in the form of a scanned document. The handwritten text must be extracted from the image, using image processing techniques or relevant feature extraction.

The first case is called online recognition, and the second one offline recognition. Historically, the two have been separated, and a clear distinction is made in some surveys [51], while others are only focused on one branch [73]. While their nature makes them suited to different applications, e.g. touchscreen input for the former, and cheque processing for the latter, the techniques employed nowadays to perform the recognition tend to be similar. Namely, they attempt to turn a sequence of feature vectors into a sequence of characters or words, modeling an input signal at lower levels and the language at higher levels. In this respect, these methods are also close to those applied in speech recognition. Offline handwriting recognition is also related to the recognition of printed text from document images, a problem known as Optical Character Recognition (OCR). While both recognize text from images, printed text tends to be much more regular than handwriting, hence generally easier to process. In this thesis, we are concerned with the offline recognition of handwritten text.

Why is this task difficult?

The difficulty in the HTR task lies behind the nature of the human handwriting. There is a great variation in the size, the shape and the intensity in the letters of each individual. Also, the handwriting is unconstrained. That means that letters are not written following a strict positioning, in contrast with the typewriting. Not only the text line but also the single characters may have an inclination. And if one line is inclined, may the other be completely straight. And this burdens, even more, the processing of full-page handwritten documents, such as text segmentation and layout analysis.

What is more, there is a large variability of identical symbols. Factors that synthesize such variability is the writing style of each individual. Another reason is the stroke width and the quality of it. For example, the space among the characters may vary among not only different characters but also among writing instances of the same writer. Below are some examples that delineate such variance.

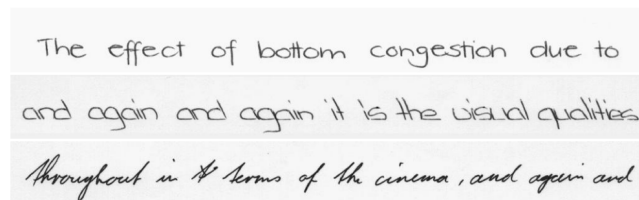


Figure 2.1: Image Samples that showcase the unconstrained and cursive nature of human handwriting.

What are the its applications?

- Automation Purposes in Document Processing : Archiving and Retrieval, Analysis of Forms

- Communication : Human-Machine Interaction (Online Handwritten Recognition) via Smartphones and tablets
- Digitalization and Preserverance of Historical Documents

2.2 Related Problems in the Document Analysis field

2.2.1 Keyword Spotting

Keyword Spotting is the task of searching, locating and retrieving specific words of interest in a set of documents. It stems from the analogous task of Spoken Term Detection in a speech segment.

Numerous are the applications of KWS, but not limited to:

- assisting human transcribers in identifying words in degraded documents, especially those appearing for the first time.
- word spotting in graphical documents such as maps
- keyword retrieval in prehospital care reports
- automatic sorting of handwritten mail containing significant words
- searching online in cultural heritage collections stored in libraries all over the world

There are two dominant archetypes when it comes to searching a keyword in a set of documents.

- Query-by-Example (QbE) paradigm assumes that an exemplar image, containing the query keyword of interest, is given to the system, and it has to find the instances of the same keyword within the collection of document images. Samples of work [13] [56] [63]
- Query-by-String (QbS) paradigm assumes that the query keyword is presented to the system as an individual symbol part of a vocabulary lexicon or, alternatively, as a sequence of characters of a given alphabet.

The first approach of KWS through QbE poses certain limitation in practical applications as the uses hat to identify a query word image from the document image collections. Thus extended research has been conducted in the domain of KWS through QbS [13] [12] [64]

In KWS task the predominant features that have been used to represent a word image are SIFT descriptors, Geometric Features and HOG-based descriptors.

Recent works incorporate Deep Neural Networks for obtaining the representations of word images. [71] [58]

2.2.2 Verification-Identification

Biometrics technology is used in a wide variety of security applications. The aim of such systems is to recognize a person based on physiological or behavioral traits. In the first case, the recognition is based on measurements of biological traits, such as the fingerprint, face, iris, etc. The later case is concerned with behavioral traits such as voice and the handwritten signature. Biometric systems are mainly employed in two scenarios: **verification** and **identification**. In the first case, a user of the system claims an identity, and provides the biometric sample. The role of the verification system is to check if the user is indeed who he or she claims to be. In the identification case, a user provides a biometric sample, and the objective is to identify it among all users enrolled in the system.

Signature Verification

The handwritten signature is a particularly important type of biometric trait, mainly due to its ubiquitous use to verify a person's identity in legal, financial and administrative areas. One of the reasons for its widespread use is that the process to collect handwritten signatures is non-invasive, and people are familiar with the use of signatures in their daily life. Signature verification systems aim to automatically discriminate if the biometric sample is indeed of a claimed individual. In other words, they are used to classify query signatures as genuine or forgeries. Forgeries are commonly classified in three types: random, simple and skilled (or simulated) forgeries. In the case of random forgeries, the forger has no information about the user or his signature and uses his own signature instead. In this case, the forgery contains a different semantic meaning than the genuine signatures from the user, presenting a very different overall shape. In the case of simple forgeries, the forger has knowledge of the user's name, but not about the user's signature. In this case, the forgery may present more similarities to the genuine signature, in particular for users that sign with their full name, or part of it. In skilled forgeries, the forger has access for both the user's name and signature, and often practices imitating the user's signature. This results in forgeries that have higher resemblance to the genuine signature, and therefore are harder to detect. [33]

The problem of automatic handwritten signature verification is commonly modeled as a verification task: given a learning set L , that contains genuine signatures from a set of users, a model is trained. This model is then used for verification: a user claims an identity and provides a query signature X_{new} . The model is used to classify the signature as genuine (belonging to the claimed individual) or forgery (created by someone else). To evaluate the performance of the system, we consider a test set T , consisting of genuine signatures and forgeries. The signatures are acquired in an enrollment phase, while the second phase is referred to operations (or classification) phase.

If a single model is used to classify images from any user, we refer to it as a writer-independent (WI) system. If one model is trained for each user, it is referred to as a writer-dependent (WD) system. For WI systems, the common practice is to train and test the system with a different subset of users. In this case, we consider a development set D (which is used to train the WI model), and an exploitation set E , which represent the users enrolled to the system (and is further divided in L and T , as indicated above).

Related literature in Signature Verification [59] [21] [32] [55] [78]

Writer Identification

Writer Identification is the process of finding the genuine writer from a list of other registered candidates based on the similarity between their handwriting. This task has many challenges due to the intra-variance of the visual features extracted from one's handwriting. This variance appears since our handwriting depends on our mood, the conditions that we are writing, the place etc. For example, when we are writing fast the letters tend to be more skewed and edgy. On the contrary, if we are writing slowly, the letters are different comparing to the first case.

The data acquisition for such task resembles the one of Handwritten Text Recognition. It can be either online or offline. In the first case they are recorded more than the sequence of two-dimensional trajectories of the writer, such as the writing speed, the position of the pen, the angles, the pressure etc. Such dynamic features are used for the identification. On the other hand, offline writer identification is based only on scanned text of each writer and thus is considered a more challenging task due to the aforementioned intra-variance of human writing.

The general pipeline of the Writer Identification task consists of preprocessing of the text image such as binarization, normalization, noise removal and segmentation in the level that the writing is going to be

performed, such as line, paragraph or word level. The next step consists of feature extraction from the sample test and finally the feature vector from this step is utilized so as to perform classification. Since we have multiple writers in a database, we talk about multiclass classification.

In the terms of feature extraction the image is converted in a feature vector with the use some of statistical methods. global features and local features. The global features describe the global traits of entire text image. It represents texture features, contour representations, and shape descriptors in the entire image. Some example of global features are Invariant Moments like Hu, Zernike, Shape Matrices like perimeter, area, compactness etc., texture matrices like local binary patterns, Histogram Oriented Gradients (HOG). The local structure and topology of characters or writing such as Edges, loops, dots and diacritics, vertical and horizontal lines, start and end point, direction of writing, thickness or thinness of strokes and corners. [57]

The emerging sector of Machine Learning in Computer Vision brought the automated feature extraction from text images. Convolutional Neural Networks and Recurrent Neural Networks could be utilized. The automated features in comparison with the hand-engineered features show higher performance and provide better recognition rate.

Concerning the classification, the algorithms utilized can be divided into three categories. The first one is distance based classification applying distances such as Eukleidian, Manhattan, hamming etc. The second one is machine learning which includes Naive bayes, Hidden Markov Models, KNNs, SVMs etc. The final category is the classification using deep learning such as CNN with on top a Linear Layer, LSTM etc.

2.2.3 Layout Analysis

Document Layout Analysis is the process of identifying and categorizing the regions of interest in the scanned image of a text document. The content of a document page is categorized in textual and non-textual content such as images, plots, math symbols and tables. Also, text zones hold a crucial semantic role in the structure of the document. Thus, semantic labeling of all the parts of the document, such as the title, the captions and the footnotes are part of the logical layout analysis. [2]

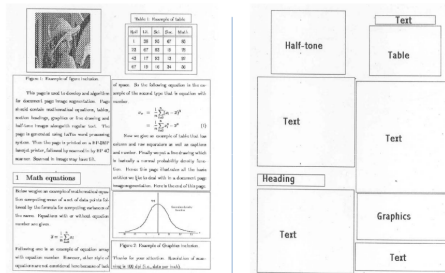


Figure 2.2: Objective of Layout Analysis [3]

2.3 IAM Database

The IAM database [44] consists of images of handwritten pages. They correspond to English texts extracted from the LOB corpus [39], copied by different writers. The database consists of text images in different scales such as line, paragraph and document level. In our case we utilize the line level format. This format is also been used in other publications such as [74] [53] [17].

The IAM Database is structured as follows.

- 657 writers contributed samples of their handwriting
- 1'539 pages of scanned text
- 5'685 isolated and labeled sentences
- 13'353 isolated and labeled text lines
- 115'320 isolated and labeled words

For the Task of Large Writer-Independent Text Line Recognition there are :

Set Name	Number of Text Lines	Number of Writers
Train	6161	283
Validation 1	900	46
Validation 2	940	43
Test	1861	128
Total	9862	500

Parties of Northern Rhodesia have agreed to
as Mr. Robert Weaver of New York. One
he confined his reply to the observation
would provide an ambulance service for the absolutely
food scheme. It was maintained during

Figure 2.3: Train Samples of IAM Database

With five and seven frame devices Algeria the right to self-
on Mr. Harold Macmillan to join the
level than in the past' might be

Figure 2.4: Test Samples of IAM Database

2.4 Thesis Outline

In this Thesis we explore the whole end-to-end Handwritten Text Recognition (HTR) pipeline.

At the **Chapter 3 (3)** we make a detailed presentation of the rudimentary theoretical background needed to clarify the basic blocks of this thesis.

At the **Chapter 4 (4)** we present the previous work that is conducted in the domain of Document Analysis and Information Retrieval so as to showcase the importance and the applications of the so far work.

At the **Chapter 5 (5)** we develop the backbone Optical Model, two Data Augmentation Techniques for the HTR task and experiments are conducted.

At the **Chapter 6 (6)** we explore the potential of decomposing the target sequence into n-gram in the learning and decoding process.

At the **Chapter 7 (7)** we propose a novel Multitask Architectures for HTR. and we deduce crucial conclusions about it.

At the **Chapter 8 (8)** we implement a novel deep architecture based only on Convolutional Neural Networks and conduct several experiments on data augmentation and decoding under an ensemble framework.

At the **Chapter 9 (9)** we discuss the overall contributions of this Thesis, the ongoing steps and the future directions that are going to extend our current work.

Chapter 3

Theoretical Background

3.1 Theoretical Background

This chapter provides the background material needed for clearly explaining the parts of our work that follows in the next chapters. Section 3.2 briefly reviews the basic ideas of Machine Learning with extended details in Supervised Learning. Section 3.4 covers details about the main blocks that were used all along this Thesis. Finally, section 3.5 constitutes a thorough presentation of Connectionist Temporal Classification for Sequence Transduction which in the inherent nature of our task.

3.2 Machine Learning Preliminaries

Machine Learning (ML) is the form of applied statistics with an inclination of creating models that estimate a function rather than calculating confidence intervals around this function. Solving a ML problem requires a set of components. That is, an optimization algorithm, a cost function, a model and a dataset.

According to Tom Mitchel [4]:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .

Some examples of the task T is problems like Classification, Regression, Sequence Transduction like Machine Translation, Speech/Handwritten Recognition etc. In order to calculate the performance of the ML algorithm, we need to design metrics that numerically interpret its ability to perform the specific task that was trained to. The experience E refers to the way the data are exposed to ML Algorithm. If the data are 'consumed' by the model in the form of features in some N -dimensional space, then we talk about **Unsupervised Learning**. If the data are accompanied by labels or targets then we refer to **Supervised Learning**. Some machine learning algorithms do not just experience a fixed dataset. For example, **Reinforcement Learning** algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences.

3.2.1 Supervised Learning

The subject of this thesis, namely Handwritten Text Recognition, falls in the general category of Supervised Classification Task. Consequently, we are going to provide the substantial context for Supervised Learning in the following paragraphs.

Definition: Given a dataset of N_g training samples $D = \{f(\mathbf{x}_n, \mathbf{y}_n), n = 1, \dots, N_g\}$, the task is to learn a function mapping the input X to the output Y . How well the function fits the training data, i.e. how accurately it maps X to Y , is quantified by a loss function $L : Y \times Y \rightarrow \mathbb{R}^d$. For instance, given a training example $(\mathbf{x}_i, \mathbf{y}_i)$ the loss of predicting the value $\mathbf{y}_i = f(\mathbf{x}_i)$ is computed by $L(\hat{\mathbf{y}}_i; \mathbf{y}_i)$.

The accuracy of a learning algorithm is measured by its ability to make an accurate prediction $\hat{\mathbf{y}}_i$, when it is estimated with a novel input $\hat{\mathbf{x}} \notin D$. This is referred to as the *generalization* ability of the algorithm.

In order to estimate the generalization ability of our model we split the entirety of our data into the *training set* and the *test set*. The one is used for training the model and the later is reserved for evaluating the one. The training error is calculated on the training data set and it is minimized by an optimization algorithm. The generalization error is formally the expectation of the model's error if we apply it to an infinite set of unseen inputs. In practice, it is approximated by the model's error on the test set.

No Free Lunch Theorem and the Inductive Bias

The “no free lunch” (NFL) theorem [76] states that all optimization problem algorithms perform equally well when averaged over all possible problems. This implies that none algorithm works best for learning all possible target functions. Note that the NFL theorem only applies to problems drawn uniformly from the space of all problems, which is actually not the case for real-world problems. This highlights the importance of inductive bias, i.e. making assumptions about the nature of the target function when selecting an algorithm for a particular problem. Some other factors to consider when choosing a learning algorithm are accuracy, model complexity, training time, number of parameters and number of features.

Inductive Bias : The *inductive bias* (also known as learning bias) of a learning algorithm is the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered.

In machine learning, one aims to construct algorithms that are able to learn to predict a certain target output. To achieve this, the learning algorithm is presented some training examples that demonstrate the intended relation of input and output values. Then the learner is supposed to approximate the correct output, even for examples that have not been shown during training. Without any additional assumptions, this problem cannot be solved exactly since unseen situations might have an arbitrary output value. The kind of necessary assumptions about the nature of the target function is subsumed in the phrase inductive bias. [47] [23]

Bias and Variance Trade-off

The bias-variance tradeoff [35] is a fundamental tradeoff between reducing the two sources of errors due to which learning algorithms fail to estimate the target function (or generalize on unseen data). The expected generalization error of a learning algorithm can be written as the sum of these two sources of errors, namely the *bias* and the *variance*

Consider an independent variable denoted by X and a dependent variable denoted by Y which are related to each other by a relation like $Y = f(X) + \varepsilon$. Using a learning algorithm, we estimate a model $f(\hat{x})$ of $f(x)$, whose expected error can be written as :

$$\begin{aligned} \text{Bias}(f(\hat{x})) &= E[f(\hat{x})] - f(x) \\ \text{Variance}(f(\hat{x})) &= E[f(\hat{x})^2] - E[f(\hat{x})]^2 \\ \text{Err}((f(x) - f(\hat{x}))^2) &= \text{Bias}(f(\hat{x}))^2 + \text{Variance}(f(\hat{x})) + \sigma^2 \end{aligned} \tag{3.1}$$

The first term is called *bias*, which pertains to erroneous simplifying-assumptions in the learning algorithm. The second term is the *variance*, which corresponds to sensitivity to inputs (and hence noise) in the training data. The third term is irreducible error, which corresponds to noise in the true function itself. An algorithm with high bias might not have enough flexibility to model the target function $f(x)$ (underfitting). On the other hand, a model with high variance tends to model the random noise in data along with the target function (overfitting).

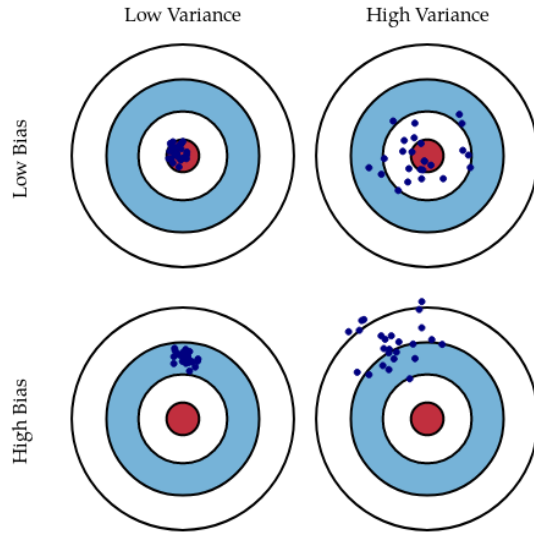


Figure 3.1: Graphical illustration of bias and variance [5]

The above presents a bulls-eye diagram to visualize the bias and variance in a 2-dimensional space. As we move away from the bulls-eye, our predictions get worse and worse. Imagine we can repeat our entire model building process to get a number of separate hits on the target. Each hit represents an individual realization of our model, given the chance variability in the training data we gather. Sometimes we will get a good distribution of training data so we predict very well and we are close to the bulls-eye, while sometimes our training data might be full of outliers or non-standard values resulting in poorer predictions.

Overfitting vs Underfitting

At its root, dealing with bias and variance is really about dealing with over- and under-fitting.

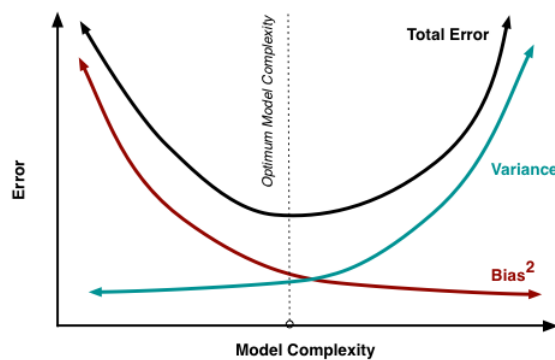


Figure 3.2: The Model Complexity as a function of Bias and Variance [5]

Bias is reduced and variance is increased in relation to model complexity. As more and more parameters are added to a model, the complexity of the model rises and variance becomes our primary concern while bias steadily falls. For example, as more polynomial terms are added to a linear regression, the greater the resulting model's complexity will be. In other words, bias has a negative first-order derivative in response to model complexity while variance has a positive slope.

Support Vector Machines

Support Vector Machines is considered the best off-the-shelf supervised learning algorithm. In order to explain briefly the algorithm we need first introduce the idea behind the margins in the binary (and in general multiclass) classification problem.

Consider a classification task, where the posterior probability $p(y = 1|\mathbf{x}; \theta)$ is modeled by the function $h_\theta(\mathbf{x}) = g(\theta^T \mathbf{x})$. We would then predict “1” on an input x if and only if $h_\theta(\mathbf{x}) \geq 0.5$, or equivalently, if and only if $\theta^T \mathbf{x} \geq 0$. Consider a positive training example ($y = 1$). The larger $\theta^T \mathbf{x}$ is, the larger also is $\theta^T \mathbf{x} = p(y = 1|x; w, b)$, and thus also the higher our degree of “confidence” that the label is 1. Thus, informally we can think of our prediction as being a very confident one that $y = 1$ if $\theta^T \mathbf{x} \geq 0$. The same holds for the case of $y = 0$ and $\theta^T \mathbf{x} \leq 0$

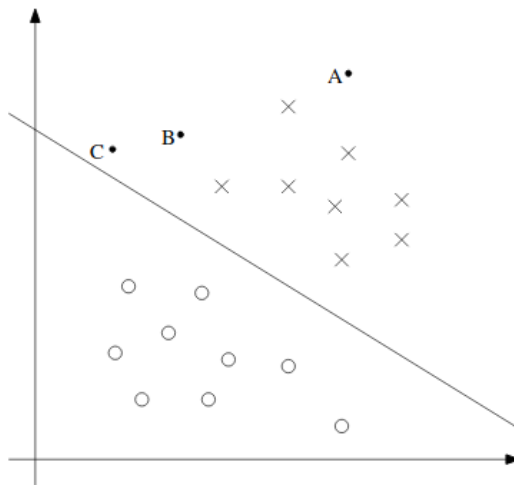


Figure 3.3: For a different type of intuition, consider the following figure, in which x’s represent positive training examples, o’s denote negative training examples, a decision boundary (this is the line given by the equation $\theta^T \mathbf{x} = 0$, and is also called the separating hyperplane). A, B and C are sample points.

From the above figure, we can observe that A is far from the decision boundary. Thus, if we make a prediction about the A our model will be confident enough for this prediction. On the contrary, the model’s confidence drops for sample C which lies very close to the separating hyperplane. Sample B lies in between and has the ideal distance from the hyperplane. So, informally we think it would be nice if, given a training set, we manage to find a decision boundary that allows us to make all correct and confident (meaning far from the decision boundary) predictions on the training examples. This is formalized with the **geometric margins** and **functional margins**.

One key innovation associated with support vector machines is the kernel trick. The kernel trick consists of observing that many machine learning algorithms can be written exclusively in terms of dot products between examples. For example, it can be shown that the linear function used by the support vector machine can be rewritten as :

$$\theta^T \cdot \mathbf{x} + b = b + \sum_{i=1}^m a_i \cdot \mathbf{x}^T \cdot \mathbf{x}^i \quad (3.2)$$

where $\mathbf{x}^{(i)}$ is a training example and a is a vector of coefficients. Rewriting the learning algorithm this way allows us to replace \mathbf{x} by the output of a given feature function $\phi(\mathbf{x})$ and the dot product with a function

$K(\mathbf{x}, \mathbf{x}^{(i)}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}^{(i)})$ called a **kernel**. This function is nonlinear with respect to \mathbf{x} , but the relationship between $\phi(\mathbf{x})$ and $f(\mathbf{x})$ is linear. Also, the relationship between a and $f(\mathbf{x})$ is linear. The kernel-based function is exactly equivalent to preprocessing the data by applying $\phi(\mathbf{x})$ to all inputs, then learning a linear model in the new transformed space. [16]

3.2.2 Unsupervised Learning

For completeness, we will sketch out a fundamental outline for Unsupervised Learning.

The learning process involves observing several examples of a random vector \mathbf{x} , and attempting to implicitly or explicitly learn the probability distribution $p(\mathbf{x})$, or some interesting properties of that distribution.

In unsupervised learning, algorithms learn to infer patterns within a dataset without being presented with target values for each learning example. This leaves the algorithm to discover the underlying structure or distribution of the data. Two of the most common unsupervised learning problems are clustering and representation learning.

- Clustering is the task of finding groupings in the data based on a predesignated similarity measure such that objects that belong to the same group are more similar to each other than to those in other groups. Clustering is often used for exploratory data analysis, which aims at providing insight into a dataset by identifying patterns, trends and outliers.
- Representation learning comprises a set of techniques for discovering representations of raw data that are conducive to classification or prediction tasks. This replaces manual feature engineering, which can be a difficult and expensive process since it requires domain knowledge. Moreover, unsupervised methods for representation learning often perform dimensionality reduction, i.e. finding representations of the input that lie in a low-dimensional space.

3.3 Neural Networks

The term ‘neural network’ has its origins in attempts to find mathematical representations of information processing in biological systems [45] [60] [62]. Indeed, it has been used very broadly to cover a wide range of different models, many of which have been the subject of exaggerated claims regarding their biological plausibility. From the perspective of practical applications of pattern recognition, however, biological realism would impose entirely unnecessary constraints. Our focus in this chapter is therefore on neural networks as efficient models for statistical pattern recognition. In particular, we shall restrict our attention to the specific class of neural networks that have proven to be of greatest practical value, namely the multilayer perceptron [16]

The Perceptron

The simplest possible type of ANNs is the perceptron [60]. The perceptron is a network with a single unit that can be used to for binary classification problems assuming that the two classes are linearly separable. The algorithm was developed in the 1950s by Frank Rosenblatt. As it is illustrated in Figure 3.4, the perceptron first sums up the weighted inputs and a bias, and then applies to the weighted sum an activation function such as the sign or the sigmoid function.

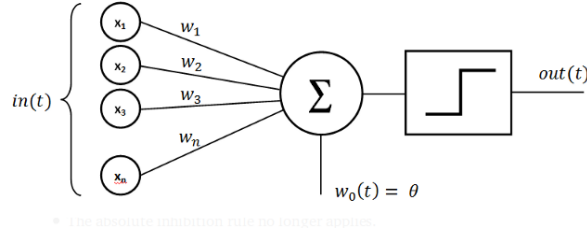


Figure 3.4: The perceptron of Rosenblatt

The basic Neural Network (NN) model can be defined by the following equation.

$$a_j = \sum_{i=1}^N (W_{ji})^{(1)} \cdot x^i + W_{j0} \quad (3.3)$$

where j and the superscript (1) indicates the corresponding parameters of the 1st layer of the network. The parameters W_{ij} are the *weights* and the parameters W_{j0} are the *biases*. The quantities a_j are the *activations*. Each of them are transformed with the use of a non linear, differentiable function, the *activation function* $h(\cdot)$ and are transformed into the so called *hidden units* :

$$z_j = h(a_j) \quad (3.4)$$

The hidden units are combined linearly so as to give the *output units activations* :

$$a_k = \sum_{j=1}^M (W_{kj})^{(2)} \cdot z^j + W_{k0}^{(2)} \quad (3.5)$$

where $k = 1 \dots K$ is the total number of outputs

The choice of the activation function is determined by the nature of the data and the assumed distribution of target variables and the problem that we wish to solve. Thus for standard regression problems, the activation function is the identity so that $y_k = a_k$. Similarly, for multiple binary classification problems, each output unit activation is transformed using a logistic sigmoid function so that :

$$y_k = \sigma(a_k) \quad (3.6)$$

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

Now, it is demandable to denote that except the sigmoid activation function, pivotal role in the Multiclass Classification Problems, which is our case, is the *Softmax Activation Function*. It is a function that takes as input a vector of K real numbers and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. That is, prior to applying softmax, some vector components could be negative, or greater than one and might not even sum to one. However, after applying softmax, each component will be in the interval $(0, 1)$. [6]

The equation of the softmax activation function $\sigma : \mathbb{R}^K \rightarrow \mathbb{R}^K$ is the bellow :

$$\sigma(a) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (3.7)$$

By combining all the above equations we get the final one that describes the so called *Multilayer Perceptron*

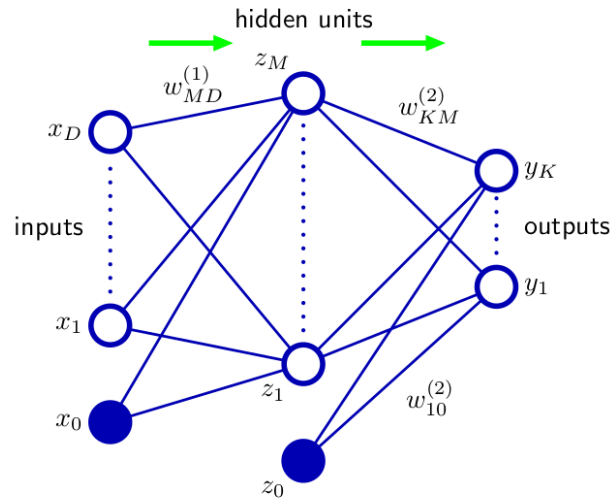


Figure 3.5: The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links inputs coming from additional input and hidden variables x_0 and z_0 . Arrows denote the direction of information flow through the network during forward propagation. [54]

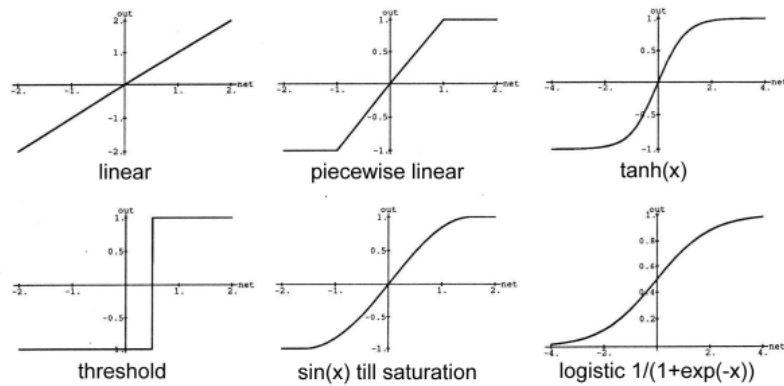


Figure 3.6: Neural Network Activations Functions [24]

3.3.1 Learning Algorithms

Batch Gradient Descent

Vanilla Gradient descent computes the gradient score of the objective function to the parameters θ of the entire training dataset.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (3.8)$$

The batch gradient descent algorithm needs to calculate the gradients of the whole train set. This is very slow and can evoke technical impediments such as memory insufficiency. Also, Batch gradient descent does not allow to train our model online.

Stochastic Gradient Descent

On the contrary, Stochastic Gradient Descent (SGD) performs a parameter update for each training example x^i and y^i :

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^i; y^i) \quad (3.9)$$

Comparing both batch gradient descent and stochastic gradient descent algorithms, we conclude that the first implement abundant computations of the gradients. In large datasets it is possible that some examples will be similar. On the contrary, Stochastic Gradient eschews the redundancy by updating the parameters per one training sample. Also, SGD can also be utilized in online learning.

While batch gradient descent converges to the minimum of the basin the parameters are placed in, SGD's fluctuation, on the one hand, enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting. However, it has been shown that when we slowly decrease the learning rate, SGD shows the same convergence behaviour as batch gradient descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively

Mini-Batch Gradient Descent

The Mini-Batch Gradient Descent Algorithm combines both BGD and SGD ideas. For one update it takes into consideration only n samples from the training set.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{i,i+n}; y^{i,i+n}) \quad (3.10)$$

However Mini-Batch Gradient descent does not guarantee good convergence and some emerging challenges must be addressed.

- The selection and initialization of the suitable learning rate is a difficult task and may demand lots of experiments so as to deduce the appropriate one.
- A defined learning rate applies to all the data no matter what is the nature of them. If our data is sparse and our features have very different frequencies, we might not want to update all of them to the same extent, but perform a larger update for rarely occurring features

Adaptive-Moment-Estimation

Adaptive Moment Estimation [40] is a method that computes adaptive learning rate for each parameter. Adam keeps an exponentially decaying average of past gradients m_t , 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 [15]. We compute the decaying averages of past and past squared gradients m_t and u_t respectively as follows:

$$\begin{aligned}m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\u_t &= \beta_2 \cdot u_{t-1} + (1 - \beta_2) \cdot g_t^2\end{aligned}\tag{3.11}$$

m_t and u_t are estimates of the first moment (the mean) and the second moment (the variance) of the gradients respectively. For shunning the bias of m_t and u_t towards zero, authors [40] introduced the unbiased first and second moments.

$$\begin{aligned}\widehat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \widehat{u}_t &= \frac{u_t}{1 - \beta_2^t}\end{aligned}\tag{3.12}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{u}_t} + \varepsilon} \cdot \widehat{m}_t\tag{3.13}$$

3.4 Deep Neural Networks

3.4.1 Convolutional Neural Networks

Motivation and Properties of CNN

Fully Connected Neural Networks cause three problems in practice. First and foremost, traditional NNs use matrix multiplication to describe the interactions of neurons between two layers. Every output unit interacts with every input unit. In fact, this imposes a significant computational burden when it comes for image data where the dimension of a single image is of that shape, that one vector will be too big for the NN to efficiently process. What is more, a NN demands fixed-size image in spatial dimension and only this sets significant limitations. Secondly, in NNs each element of the weight matrix is used exactly once when computing the output of a layer. It is multiplied by one element of the input and then never revisited. As a synonym for parameter sharing, one can say that a network has tied weights, because the value of the weight applied to one input is tied to the value of a weight applied elsewhere. Last but not least, some kinds of data cannot be processed by neural networks defined by matrix multiplication with a fixed-shape matrix. An example of such are the image data.[22]

All the above limitations of the NNs are surpassed by the Convolutional Neural Networks (CNN). Convolution leverages three important ideas that can help improve a machine learning system: sparse interactions, parameter sharing and equivariant representations. Moreover, CNNs provide a means for working with inputs of variable size. We now proceed with describing each of these ideas.

CNNs have **sparse interactions** (also referred to as sparse connectivity or sparse weights). This is accomplished by making the kernel smaller than the input. For example, when processing an image, the input image might have thousands or millions of pixels, but we can detect small, meaningful features such as edges with kernels that occupy only tens or hundreds of pixels. This means that we need to store fewer parameters, which both reduces the memory requirements of the model and improves its statistical efficiency.

Parameter sharing refers to using the same parameter for more than one function in a model. In a traditional neural net, each element of the weight matrix is used exactly once when computing the output of a layer.[19]

The parameter sharing causes the layer to have a property called **equivariance to translation**. For example, let I be a function giving image brightness at integer coordinates. Let g be a function mapping one image function to another image function, such that $I_{inv} = g(I)$ is the image function with \cdot . This shifts every pixel of I one unit to the right. If we apply this transformation to I , then apply convolution, the result will be the same as if we applied convolution to I_{inv} , then applied the transformation g to the output.

Convolutional Operation

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n).$$

Figure 3.7: Convolution Operation in Input Image [22]

CNN Modules

Before digging into what a CNN learns, we need to clarify the basic modules that are used in a basic CNN.

- Convolutional Layer

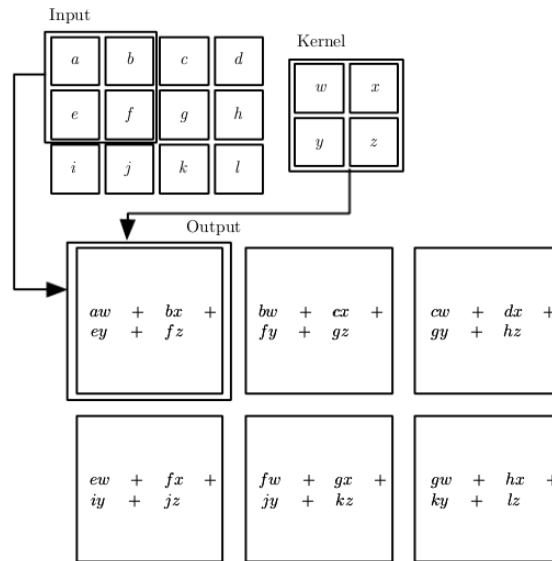


Figure 3.8: Convolution Operation in Input Image [22]

- Non Linear Activation Function

In Computer Vision Tasks, it is widely used the Rectifier Linear Unit (ReLU) which performs a thresholding to zero.

- Pooling Layer

Performs a downsampling in the spatial dimensions of height and width.

- Fully Connected Layer

An ordinary Neural Network, which usually performs the classification task.

Stacking all the above components consecutive, we obtain a Convolutional Network.

Layer	Learnable Parameters
Convolutional Layer	Yes
Non Linear Activation Function	No
Pooling Layer	No
Fully Connected Layer	Yes

Table 3.1: Learnable Parameters per Layer in a ConvNet

The **Convolutional Layer's** learnable parameters are spatially small kernels which convolve through the entire width and height of the input image. As we slide the filter over spatial dimensions of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual features. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

Local Connectivity :

Input Images are high-dimensional inputs and so it is impractical to connect all neurons from the previous level with the neurons of the next level. Instead we are going to connect each neuron with a subset grid of the input image. This size of the grid actually defines the kernel size, also the so called **receptive field**. The kernel size is a hyperparameter. The Result of a Convolutional Operation between an Input Volume and a Kernel is called Feature Map.

The term convolution does not correspond strictly to the convolution that is defined in the signal processing domain. In ConvNets the Convolutional Operation does not contain any kernel 180 degrees flipping. Instead is a simple dot product between the image grid and the kernel. The sliding of the kernel reminds the signal processing convolution and that is why is called so.

The Avoidance of Feature Map Shrinking:

This need actually demands the need of another hyperparameter of the convolutional layer, which is the zero-padding. If we think that Feature Map always shrinks after a convolutional layer, without zero padding we would not be able to create deeper architectures.

The size of the kernel defines actually the receptive field and is considered one of the hyperparameters of the ConvLayer. The zero-padding and the stride are considered the other two hyperparameters. The output of a ConvOp is often called Feature Map. Zero-padding is used so as to avoid the elimination of the spatial dimensions of the Feature Maps. Stride defines the step of the kernel in the spatial dimensions when it comes to apply the Convolution.

Pooling Layer :

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

3.4.2 Sequence Modeling : Recurrent Neural Networks

Recurrent neural networks or RNNs [61] are a family of neural networks for processing sequential data. Much as a convolutional network is a neural network that is specialized for processing a grid of values X such as an image, a recurrent neural network is a neural network that is specialized for processing a sequence of values x_1, \dots, x_t . Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length. [22]

RNN Structure

Recurrent layers (RNNs) are a type of neural network layer with an internal state for each unit. These were originally designed to process sequences of vectors instead of a single vectors, as fully connected neural networks. At each time-step, the output of of layer depends on the current input and the previous state. In its simplest form, the state of each neuron in a recurrent layer is just its output. Thus, for a sequence of T elements of m -dimensional (row) vectors, x_1, \dots, x_t , the output of a simple RNN, parameterized by the matrices $W \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{n \times n}$ is a sequence of n -dimensional vectors y_1, y_2, \dots, y_n , given by the equation :

$$y_t = \sigma(x_t W + y_{t-1} R)$$

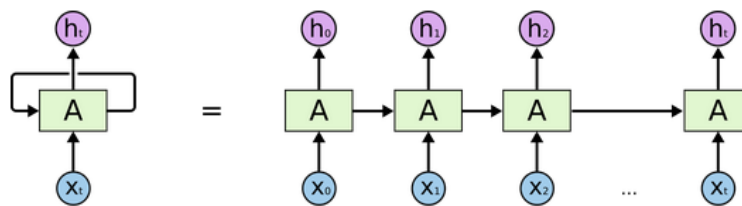


Figure 3.9: Unrolling the Recurrent Neural Networks [7]

Bidirectional RNNs

All of the recurrent networks we have considered up to now have a “causal” structure, meaning that the state at time t only captures information from the past, x_1, \dots, x_{t-1} , and the present input x_t . Some of the models we have discussed also allow information from past y values to affect the current state when the y values are available.

However, in many applications we want to output a prediction of y_t which may depend on the whole input sequence. For example, in speech recognition, the correct interpretation of the current sound as a phoneme may depend on the next few phonemes because of co-articulation and potentially may even depend on the next few words because of the linguistic dependencies between nearby words: if there are two interpretations of the current word that are both acoustically plausible, we may have to look far into the future (and the past) to disambiguate them. This is also true of handwriting recognition and many other sequence-to-sequence learning tasks, described in the next section.

Bidirectional Recurrent Neural Networks (BiRNNs) were invented from [67] and hold numerous applications in fields like Handwriting Recognition and Automatic Speech Recognition[25] [29] [28] [30]

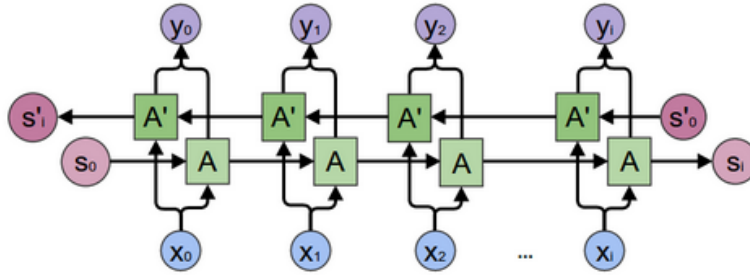


Figure 3.10: Bidirectional Recurrent Neural Network [7]

Long Term Dependencies

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in “the clouds are in the sky,” we don’t need any further context – it’s pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it’s needed is small, RNNs can learn to use the past information.

But there are numerous cases, such as in the Language Modeling domain, where in order to deduce safe conclusions about the next word in a sentence we need to go many words back since the earliest context does not give a discernible result. For example, suppose the example "I grew up in France and I lived there for twenty years. I speak fluent French. If we wanted our model to predict the last word of the presented sentence, It is understandable that in order to do so correctly should wide the context window and search for older history in the sentence and not just the last words in the row. Thus, it is understandable that as the information explodes the context window and the history to be models grows. In such cases, RNNs become unable to learn and to connect the information.

Long Short Term Memory Networks [37], called LSTMs in abbreviation, are a special kind of RNN, capable of learning long-term dependencies.

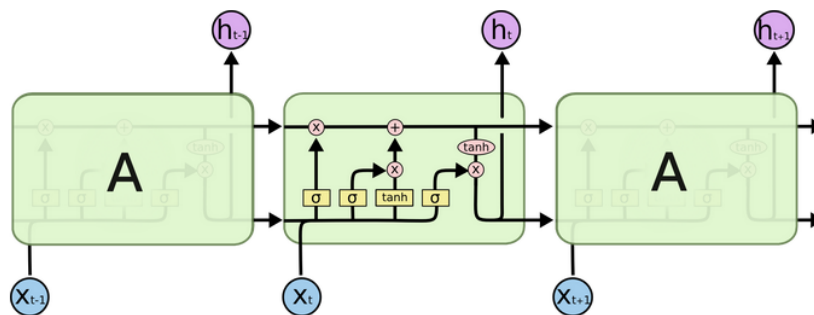


Figure 3.11: Internal Structure of LSTM Layer [7]

$$\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\hat{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t \cdot \tanh(C_t)
\end{aligned}
\tag{3.14}$$

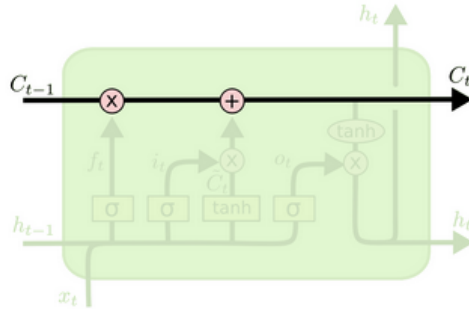


Figure 3.12: The cell state of LSTM [7]

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. The information is controlled through three gates.

- Input Gate
- Forget Gate
- Output Gate

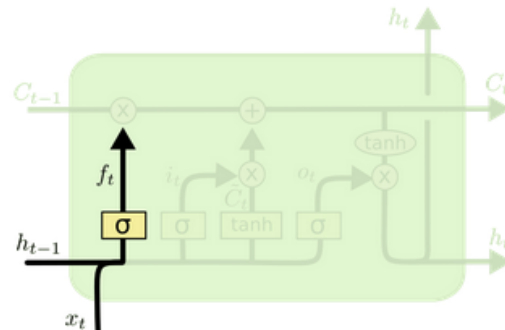


Figure 3.13: The Forget-Gate of LSTM. In this step, the LSTM decides what information will be thrown away from the cell state. This is implemented by a sigmoid layer which looks at the h_{t-1} and the x_t and assigns a number of 0 or 1 on each of the units of C_{t-1} . [7]

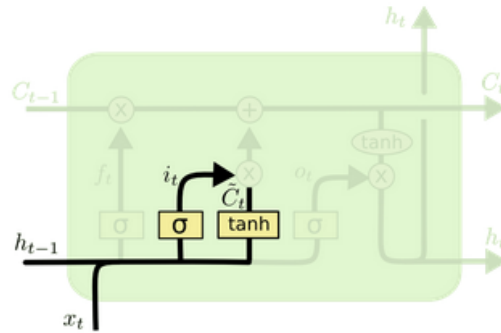


Figure 3.14: The Input-Gate of LSTM. In this step, the LSTM decides what new information will be stored. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a tanh layer creates a vector of new candidate values, \hat{C}_t , that could be added to the state. In the next step, we’ll combine these two to create an update to the state. [7]

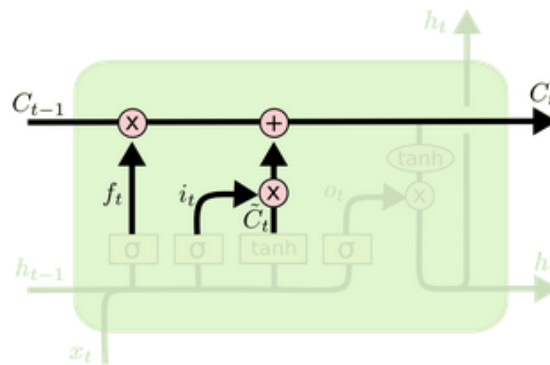


Figure 3.15: The Output-Gate of LSTM. In this step, the LSTM updates the C_{t-1} value with the new one C_t and its synthesized by how much information will be abandoned and what new information will be stored in the cell. [7]

3.5 Connectionist Temporal Classification

In this section we present the *Connectionist Temporal Classification* (CTC) introduced first by Alex Graves [31] [28]. CTC is a novel output layer for temporal classification with RNNs.

CTC is a segmentation-free and alignment-free framework for sequence-to-sequence transduction. The input image does not need to be segmented to tokens or characters so as to be recognized. With this framework, we are surpassing Sayre’s Paradox. The latest supports that cursively written word cannot be recognized without being segmented and cannot be segmented without being recognized.[27]

For a sequence labelling task where the labels are drawn from an alphabet L , CTC consists of a softmax output layer [18] with one more unit than there are labels in L . The activations of the first $|L|$ units are used to estimate the probabilities of observing the corresponding labels at particular times, conditioned on the training set and the current input sequence. The activation of the extra unit estimates the probability of observing a ‘blank’, or no label. Together, the outputs give the joint conditional probability of all labels at all timesteps. The conditional probability of any one label sequence can then be found by summing over the corresponding joint probabilities.

A counter example that proves the necessity of the blank character is the below.

Suppose that the input has length 6, let it be $X = [x_0 : x_5]$ and the ground truth is $Y = [c,a,t]$. One possible way to align the input sequence with the output sequence is the below :

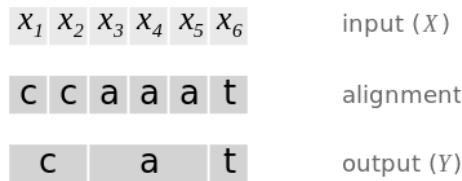


Figure 3.16: Example of a possible alignment for word "cat" [8]

From the above example, it is clear that the transformation between the output transcript and the alignment is the concatenation between the identical characters. The problem with this approach is that we cannot produce words that have double letters, such as the word “Hello”. So, we are inserting the blank character in order to permit the same consecutive characters.

More formally, let \mathbf{X} be an input sequence of length T and \mathbf{Y} be the output sequence that contains the probability distribution for all labels for all time points till T . We denote as y_k^t as the probability of observing label k at time t . These probabilities come from the network output, after being softmaxed. Let a be an alignment for some input X . With L_i we symbolize the initial set of labels, our vocabulary, and with $L_a = L_i \cup \text{blank}$ the augmented one. The conditional probability of a is the defined below :

$$p(a|\mathbf{X}) = \prod_{t=1}^T y_{a_t}^t, \forall a \in L_a^T \tag{3.15}$$

Implicit from the above equation is the assumption that the network outputs at different times are conditionally independent, given the internal state of the network. This is ensured by requiring that no feedback connections exist from the output layer to itself or the network.

Finally, we need to calculate the conditional probability of a specific labeling Y . But first, we need to convert any alignment to labeling.

We define a many-to-one map, let it be \mathbf{B} . This operation maps all the possible alignments in a specific sequence. So, it concatenates all the same consecutive characters between two blank characters and the removes all the blank ones.

For example $\mathbf{B}(-aa-abb) = \mathbf{B}(-a-a-b) = aab$

We are utilizing \mathbf{B} so as to calculate the conditional probability of a given labeling L by summing up the probabilities off all the possible alignments for this labeling to occur.

$$p(Y|\mathbf{X}) = \sum_{a \in B^{-1}(Y)} p(a|\mathbf{X}) \quad (3.16)$$

The CTC forward-backward Algorithm

Now that we have determined how the output layer must be formulated so as to be trained with CTC, we need to define the objective function for training CTC Networks with gradient descent.

We need an efficient way for calculating the conditional probabilities of all the alignments that correspond to a specific labeling. Thus, we make use of dynamic programming and specifically an algorithm that is similar with the HMM forward-backward algorithm [54]. The key idea is that the sum of paths over a labeling can be broken into an iterative sum over paths corresponding to prefixes of that labeling. The iterations can be computed efficiently with the **forward** and **backward** variables.

Let Q be a sequence of length r , N be the set of the target units and T the length of the input sequence. All the possible alignments that can be formulated are N^T . Let $Q_{1:p}$ be a subsequence containing the first p symbols of the Q . The remaining sequence is $Q_{p:r}$.

For a labeling l , the *forward* variable $a_s(t)$ is the probability of the labeling $l_{1:s}$ at time t .

$$a_t^s = \sum_{\substack{a \in N^T: \\ B(a)=l_{1:s}}} \prod_{t=1}^t y_{a_t}^t \quad (3.17)$$

The forward variable a_t^s is calculated recurrently.

For calculating the forward and backward variables we increment the initial labeling with the blank character placed at the start, at the end and between the target units.

For example the word 'CAT' becomes '-C-A-T-'. Where $l_0 = -, l_1 = C, l_2 = -, l_3 = A$ etc.

The string can either begin with the blank character or with the l_1 . That means :

$$a_0^{l_0} = p_b^0 \text{ or } a_0^{l_1} = p_b^0$$

The recursion is defined as follows :

$$a_t^s = \begin{cases} (a_{t-1}^s + a_{t-1}^{s-1}) \cdot y_{l_s}^t & \text{if } l_s = \text{blank or } l_{s-2} = l_s \\ (a_{t-1}^s + a_{t-1}^{s-1} + a_{t-1}^{s-2}) \cdot y_{l_s}^t & \text{otherwise} \end{cases}$$

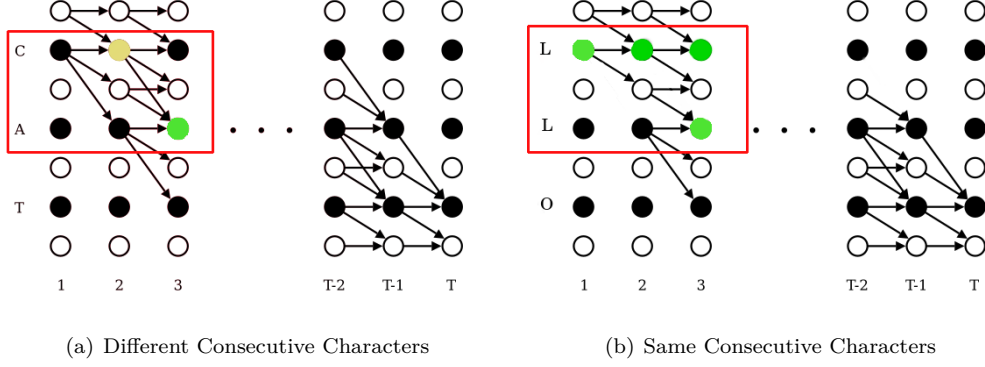


Figure 3.17: Graphical explanation of the recurrent equation of forward variable

The *backward* variable is calculated as if we read the lattice of probabilities from the right to the left. So, thinking with the same way as above, we have:

The string can either end with the blank character or with the $l_{|l|}$. That means :

$$b_T^{l_{|l|}} = p_b^0 \text{ or } b_0^{l_1} = p_b^0$$

The recursion is defined as follows :

$$b_t^s = \begin{cases} (b_{t+1}^s + b_{t+1}^{s+1}) \cdot y_{i_s}^t & \text{if } l_{s+1} = \text{blank or } l_{s+2} = l_s \\ (b_{t+1}^s + b_{t+1}^{s+1} + b_{t+1}^{s+2}) \cdot y_{i_s}^t & \text{otherwise} \end{cases}$$

Due to underflow, a rescale is performed both to forward and backward variables.

Rescaling factor :

$$\begin{aligned} C_t &= \sum_s a_t^s \in \mathbb{R}^T \\ D_t &= \sum_s b_t^s \in \mathbb{R}^T \end{aligned} \tag{3.18}$$

The C_t represents actually the greatest probability of each timestep t . It works like an 'or' gate that lets the highest probability to be expressed. Same interpretation for the D_t . Thus it is clear that the vector C_t encapsulates the probability of the most possible alignment at time t .

To evaluate the maximum likelihood error, we need the natural logs of the target labelling probabilities.

$$\ln(p(l|x)) = \sum_{t=1}^T \ln(C_t) \tag{3.19}$$

Note 1 : From the above equation, we conclude that the CTC Loss is independent of the number of target units. This is significant for the following work.

3.5.1 CTC Decoding Algorithms

Decoding a CTC network—that is, finding the most probable output transcription for a given input sequence—can be done to a first approximation by picking the single most probable output at every timestep and returning the corresponding transcription. [28]

More accurate decoding can be performed with a Beam Search algorithm, which also makes it possible to integrate language model. The algorithm is similar to decoding methods used for HMM-based systems, but differs slightly due to the changed interpretation of the network outputs. In a hybrid system the network outputs are interpreted posterior probabilities of state occupancy, which are then combined with transition probabilities provided by a language model and an HMM.

With CTC the network outputs themselves represent transition probabilities (in HMM terms, the label activations are the probability of making transitions into different states, and the blank activation is the probability of remaining in the current state). The situation is further complicated by the removal of repeated label missions on successive time-steps, which makes it necessary to distinguish alignments ending with blanks from those ending with labels.

- Blank Probability : $Blank(y, t)$ is the probability of substring y to end up with blank character at time t (I)
- Non Blank Probability : $NonBlank(y, t)$ is the probability of substring y end up with a non blank character at time t (II)
- Total Probability of substring y at time t : $Total(y, t) = Blank(y, t) + NonBlank(y, t)$ (III)

Algorithm 1 Vanilla CTC Beam Search

```

1: procedure CTC BEAM SEARCH( $P, W$ )  $\triangleright$   $P$  is the Posterior Probabilities' Matrix,  $W$  is the Beam Width
2:    $B \leftarrow \{\emptyset\}, Blank(\emptyset, 0)$ 
3:   for  $t=1 \dots T$  do
4:      $D \leftarrow BestBeams(B, W)$ 
5:      $Beams \leftarrow \{\}$ 
6:     for  $b$  in  $D$  do
7:       if  $b \neq \emptyset$  then
8:          $NonBlank(b, t) \leftarrow NonBlank(b, t - 1) \cdot P[b^e, t, |X]$ 
9:       end if
10:       $Blank(b, t) \leftarrow Total(b, t - 1) \cdot P[blank, t|X]$ 
11:       $AddBeam(newbeam, b)$ 
12:      for character in Characters do
13:         $NewBeam \leftarrow b + character$ 
14:         $Blank(newbeam, t) \leftarrow 0$ 
15:        if  $k = b^e$  then
16:           $NonBlank(newbeam, t) = P[k, t|x] \cdot Blank(b, t - 1)$ 
17:        else
18:           $NoneBlank(newbeam, t) = P[k, t|x] \cdot Total(b, t - 1)$ 
19:        end if
20:         $AddBeam(newbeam, b)$ 
21:      end for
22:    end for
23:  end for
24:  return  $max_{x \in Beams} Total(y, t)$ 
25: end procedure

```

3.5.2 N-Gram Language Models in CTC Decoding

The effects of the independence assumption made by the CTC algorithm can be mitigated by combining the output distribution of the NN with an additional language model. A language model is just prior distribution over all (meaningful) sequences of words (or characters) of a given language, or collection of documents.

Statistical N-Gram Language Models

Models that assign probabilities to sequences of words are called **Language Models** (LMs). The simplest model that assigns probabilities to sentences and sequences of words is the **n-gram**. An n-gram is a sequence of N n-gram tokens. For example the bigram is a sequence of 2 words such as "turn on". The n-gram tokens can be either word level or character level. The previous example was a word level language model. An example of a character fourgram model is "lish" .

Statistical Language Models are used to model the prior distribution of the language. This idea was first adopted by the Shannon [68]. The assumption that the probability of a word depends only on the previous words is called **Markov Assumption**. Markov models are the class of probabilistic models that assume we can predict the probability of some future unit without looking too far into the past. We can generalize the bigram (which looks one word into the past) to the trigram (which looks two words into the past) and thus to the n-gram, which looks $n - 1$ words in the past. Thus, the general equation for this n-gram approximation to the conditional probability of the next word in a sequence :

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1}) \quad (3.20)$$

This N-gram probabilities are estimated with **Maximum likelihood estimation** or MLE. We get maximum likelihood estimation for the parameters of an n-gram model by getting counts from a corpus, and normalizing the counts so that they lie between 0 and 1. So in case for a n-gram LM we have the bellow:

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})} \quad (3.21)$$

CTC Beam Search Decoding Integrating Character Level and Word Level Language Model

CTC Beam Search can support the integration of a Language Model, either **Character Level** or **Word Level Language Model**. For this reason we introduce the LM Probability :

$$LM(y, k) = NGramProb(y, k) \quad (3.22)$$

In this case we want to optimize the selection of each Beam at every step by taking into consideration not only the *Total Score* (III), but also the *LM Score*. That means, at every timestep we select sequence of characters that have the best CTC Score (what the model says) and the best score from a Language Model (how language supports it - domain knowledge)

Algorithm 2 CTC Beam Search + LM Integration

```
1: procedure CTC BEAM SEARCH( $P, W$ )  $\triangleright$   $P$  is the Posterior Probabilities' Matrix,  $W$  is the Beam Width
2:    $B \leftarrow \{\emptyset\}$ ,  $Blank(\emptyset, 0)$ 
3:   for  $\tau=1 \dots T$  do
4:      $D \leftarrow BestBeams(B, W)$ 
5:      $Beams \leftarrow \{\}$ 
6:     for  $b$  in  $D$  do
7:       if  $b \neq \emptyset$  then
8:          $NonBlank(b, \tau) \leftarrow NonBlank(b, \tau - 1) \cdot P[b^e, \tau, |X]$ 
9:       end if
10:       $Blank(b, \tau) \leftarrow Total(b, \tau - 1) \cdot P[blank, \tau | X]$ 
11:       $AddBeam(newbeam, b)$ 
12:      for character in Characters do
13:         $NewBeam \leftarrow b + character$ 
14:         $Blank(NewBeam, \tau) \leftarrow 0$ 
15:        if  $k = b^e$  then
16:           $NonBlank(NewBeam, \tau) = P[k, \tau | x] \cdot Blank(b, \tau - 1)$ 
17:        else
18:           $NoneBlank(NewBeam, \tau) = P[k, \tau | x] \cdot Total(b, \tau - 1)$ 
19:        end if
20:         $LM(NewBeam, \tau) \leftarrow NGramLM(NewBeam, b)$ 
21:         $NoneBlank(NewBeam, \tau) = NoneBlank(NewBeam, \tau) * LM(NewBeam, \tau)$ 
22:         $AddBeam(NewBeam, b)$ 
23:      end for
24:    end for
25:  end for
26:  return  $max_{x_{Beams}} Total(y, T) \cdot LM(y, T)$ 
27: end procedure
```

Chapter 4

Previous Work

4.1 Previous Work

In this chapter, two families of models used to represent the text (and its alignment) written in images are reviewed: Hidden Markov Models and Recurrent Neural Networks. Both models have been widely and successfully used, for Handwritten Text Recognition, Keyword Spotting and other document related applications.

4.2 Preprocessing Techniques for HTR

As we have already seen in the Chapter 2 (2), most of the tasks in the Document Analysis field were tackled with machine learning algorithms or probabilistic models and the features extracted were based on statistics. Such feature extraction demanded some processing techniques at first in order to reduce the variance of the text regions. Below we will refer to the most significant preprocessing techniques such as **Binarization**, and **Line Normalization**.

Other preprocessing steps that are carried at page level are the removal of bleed-through and other ink noise, the correction of rotated pages or contrast normalization. Some of these steps play a less significant role under controlled environments, but they are very important if the page images were not obtained using good scanning equipment

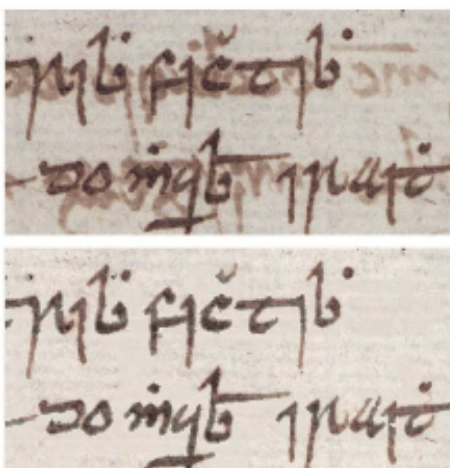


Figure 4.1: Bleed-Through removal example [34]

Binarization

Document image binarization refers to the conversion of a color or grayscale image into a binary image. The main goal is not only to enhance the readability of the image but also to separate the useful textual content from the background by categorizing all the pixels as text or non-text without missing any useful information. Document image binarization techniques are usually classified into two main categories, namely global and local thresholding. Global thresholding methods use a single threshold value for the entire image, while local thresholding methods detect a local (adaptive) threshold value for each pixel. Global techniques are capable of extracting the document text efficiently in the case that there is a good separation between the foreground and the background. Several historical binarization methods have incorporated background subtraction in order to cope with several degradations.

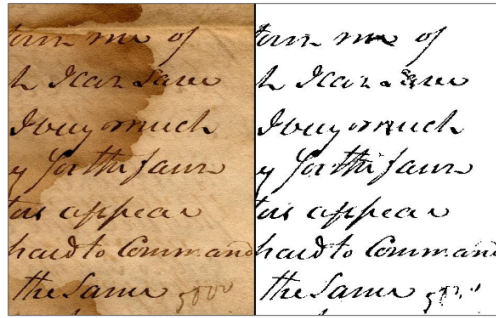


Figure 4.2: Image Binarization Example in Historical Handwritten Documents [49]

Text Line Normalization

After text is segmented into text lines, it is crucial to apply several normalization techniques that are going to decrease the variance among the different writing styles. Often, text lines do not follow a strict vertical orientation and this is a usual phenomenon in the handwritten text since it is unconstrained. This one is called *skew*. The second distortion that needs to be restored is the slant where the letters have an inclination either left or right, namely an italic-like effect.

Feature Extraction

Statistical Models such as Hidden Markov Model are not able to model images, even after the processing that have gone through such as binarization, deslanting etc. Images, is demanded to be converted into feature vectors that then will be used as input in Hidden Markov Models.

In the early days of Handwritten Text Recognition research, researchers were utilizing simple statistics from every image including :

- number of white and black pixels
- number of white and black pixels on each column
- the average value of pixel in the rectangular area

Later, high level features enrich the simpler ones. These were:

- the derivative of pixel intensity in each dimension in a rectangular cell in the image
- Speeded Up Robust Features (SURF)
- Scale-Invariant Feature Transform (SIFT)

4.3 Probabilistic Models for HTR

4.3.1 Hidden Markov Models

A Hidden Markov Model (HMM) describes a stochastic process involving two random variables: the random variable representing the sequence of observed values, denoted by X , and the random variable representing the sequence of states that produced such values, denoted by S . In particular, Hidden Markov Models are probabilistic graphical models of the joint likelihood of the two variables $P(X, S)$.

Hidden Markov Models are typically defined using the following elements:

Definition :

- A set of emitting states $S = s_1, \dots, s_n$ and a special non-emitting state final state s_F .
- A probability distribution over initial states: $P(S_1 = s), \forall s \in S^*$
- A probability distribution describing the transition model between states: $P(S_{t+1} = s' | S_t = s), \forall s \in S, \forall s' \in S \cup s_F$
- And a probability mass or density function describing the likelihood of an observed value according to each non final state : $P(X_t = x | S_t = s), \forall s \in S'$

We will refer to a particular sequence of observed values with the notation $x_{1:T} = x_1, \dots, x_T$. Notice that a sequence of observed values of length T is emitted by a sequence of states of length $T + 1$, since all valid sequences of states must end at the non-emitting final state, s_F . Thus, a particular sequence of states of length $T + 1$ is represented by $s_{1:T+1} = s_1, \dots, s_T, s_F$.

4.3.2 Hidden Markov Models for Handwritten Text

As depicted in the picture bellow, most works that need to model handwritten text choose to use an individual HMM to represent each character in the alphabet. There are some works that represent full words by a single HMM, however this becomes problematic when dealing with vocabularies of a large number of words, since the number of parameters required to estimate grows significantly.

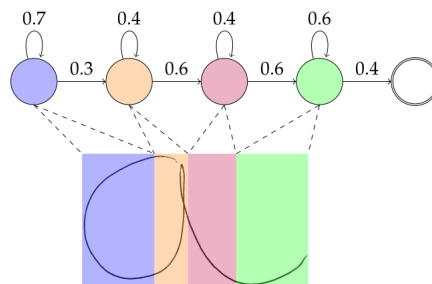


Figure 4.3: Example of the alignment produced by a character HMM modeling the letter “a”. The HMM is composed of four states in a left-to-right topology. The probabilities in the arcs represent the transition probabilities [52]

The number of states in each HMM can be fixed (i.e. all characters have the same number of states), or can be variable, since the length of each character is expected to be different from one class to the other. For instance characters like “i” or “l” are typically much shorter (horizontally) than characters like “m” or “n”.

Finally, regarding the emitting states, we will use Gaussian Mixture Models (GMM) with diagonal covariance matrices as the probability density functions used to model $P(X_t = x_t | S_t = s_t)$. Although this is the common choice in the handwritten text and speech community, the reader should be aware that it is not the only option, and models for sequences of discrete observed values have also been used in the past.

4.4 Neural Models for HTR

During many years recurrent artificial neural networks based on Multidimensional Long-Short Term Memories dominated the state-of-the-art solutions to model handwritten text [30] [74] [48]

4.4.1 Multidimensional RNN for HTR

Recurrent neural networks (RNNs) have proved effective at one dimensional sequence learning tasks, such as speech and online handwriting recognition. Some of the properties that make RNNs suitable for such tasks, for example robustness to input warping, and the ability to access contextual information, are also desirable in multidimensional domains. However, there has so far been no direct way of applying RNNs to data with more than one spatio-temporal dimension. This paper introduces multidimensional recurrent neural networks (MDRNNs), there by extending the potential applicability of RNNs to vision, video processing, medical imaging and many other areas, while avoiding the scaling problems that have plagued other multidimensional models. Experimental results are provided for two image segmentation tasks. [26]

The basic idea of MDRNNs is to replace the single recurrent connection found in standard RNNs with as many recurrent connections as there are dimensions in the data. During the forward pass, at each point in the data sequence, the hidden layer of the network receives both an external input and its own activations from one step back along all dimensions. illustrates the two dimensional case. Note that, although the word sequence usually connotes one dimensional data, we will use it to refer to data exemplars of any dimensionality. For example, an image is a two dimensional sequence, a video is a three dimensional sequence, and a series of MRI brain scans is a four dimensional sequence

Multidimensional recurrent layers are a derivative of recurrent units designed to process signals of an arbitrary size and an arbitrary number of dimensions. For the simple, one-directional RNNs/LSTMs the output at a given time step depend on the hidden state of the previous timestep. However, It is not absolutely clear what means "previous timestep" in the two dimensional case. As the following images 4.4 and 4.5 show, the pixel (i, j) depends on the previous pixels on each dimensions, namely pixels $(i, j - 1)$ and $(i - 1, j)$.

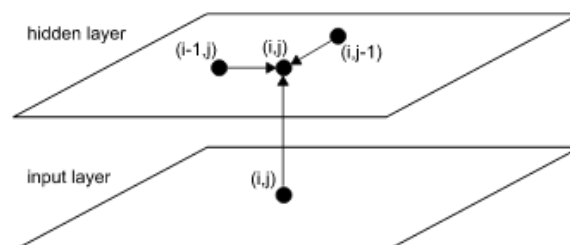


Figure 4.4: Forward Pass in MDRNN [26]

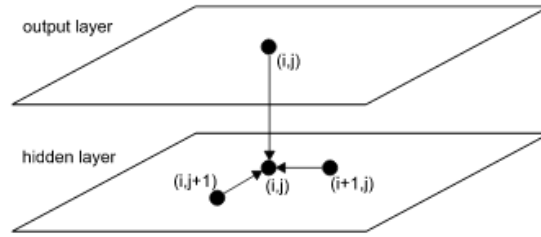


Figure 4.5: Backward Pass in MDRNN [26]

It is crucial to mention that the sequence of multiple dimensions can be processed with various ways. The aforementioned way described sequence processing in a vertical and horizontal scheme. However, a 2-dimensional sequence, or any with greater number of dimensions, can be processed with many various ways such as a diagonal as the below image depicts.

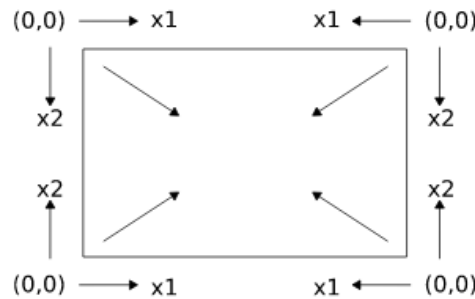


Figure 4.6: A two dimensional sequence can be processed in various directions. The internal arrows inside the rectangle indicate the direction of the propagation in forward pass. [26]

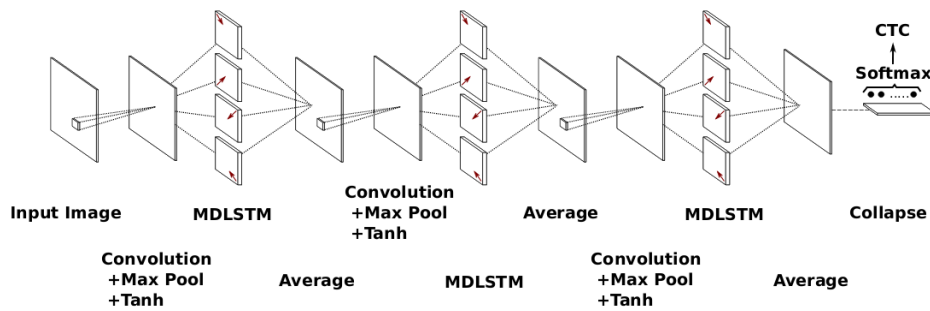


Figure 4.7: The basic network architecture used in this paper. The input image on the left is processed pixel-by-pixel using a cascade of convolutional, max-pooling and MDLSTM layers, and finally transcribed by a CTC layer on the right [74]

Various research works have adopted the usage of MDLSTM networks such as [50] [74] and have reported state-of-the-art results.

4.4.2 One-dimensional RNN for HTR

The Multidimensional Recurrent Neural Networks may denote state-of-the-art results but they hold some critical drawbacks related with the computational cost. The calculation cost is due to the dependency among

the current point and the history that needs to be calculated for its computation. This burden is multiplied when the current point depends on history from two directions.

The recent work [53] of J. Puigcerver puts into question the necessity of the MDLSTM networks for the Handwritten Text Recognition and showcases that the synergy of a CNN with one-dimensional LSTM network can culminate to satisfactory recognition rates.

The author initially, argues that visual features that are constructed through a MDLSTM are of equal quality with the ones obtained from a CNN. And after that, supports that MDLSTM can be substituted by CNNs and then one dimensional LSTMs are enough to model the dependencies among characters.

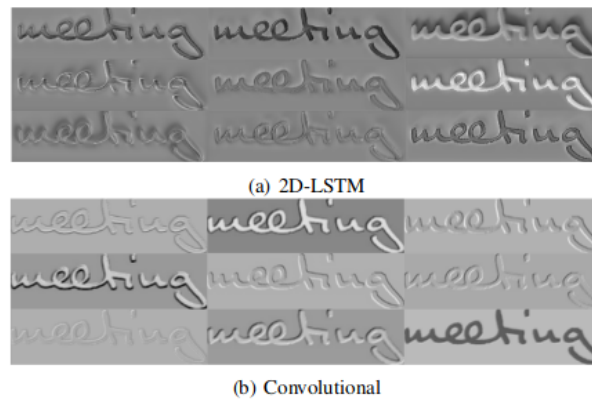


Figure 4.8: Randomly selected features extracted after a 2D-LSTM and after a convolutional layers[53]

The recommended architecture from [53] is the bellow :

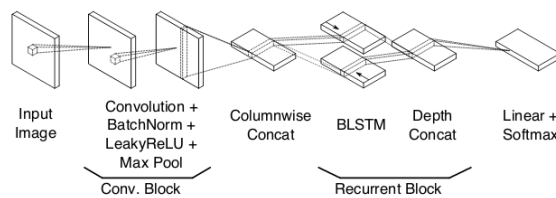


Figure 4.9: One-Dimensional LSTM Architecture for HTR [53]

System	CER (%)		WER (%)	
	Validation	Test	Validation	Test
Baseline	5.1 [4.6–5.7]	8.2 [7.6–8.9]	17.9 [16.3–19.7]	25.4 [23.9–27.0]
+ BN	5.2 [4.7–5.8]	8.3 [7.9–8.6]	18.5 [16.9–20.2]	24.9 [23.4–26.4]
+ Distortions	4.4 [3.9–4.9]	6.4 [5.8–6.9]	15.5 [14.0–17.0]	20.8 [19.6–22.1]
+ BN + Distortions	4.1 [3.6–4.5]	6.2 [5.7–6.8]	14.6 [13.1–16.1]	20.2 [19.0–21.4]

Figure 4.10: Reported Results on One Dimensional LSTMs [53]

Chapter 5

Data Augmentation for HTR and Baseline Model

5.1 Data Augmentation and Baseline Model for HTR

In this chapter, we build up the baseline architecture for our Optical Model. Moreover, we implement two new augmentation techniques for the HTR case, Local Affine, and Local Morphological, and test them on how they can improve the Learning Outcome. Finally, we make various experiments so as to find the ideal set up of our problem and perform decoding by utilizing external language resources.

5.2 Data Augmentation

5.2.1 Global Affine Transform

Affine Transform

An affine transformation is an important class of linear 2-D geometric transformations which maps variables located at position (x_1, y_1) in an input image into new variables (x_2, y_2) in the output image by applying a linear combination of translation, rotation, scaling and/or shearing (i.e. non-uniform scaling in some directions) operations.

If X and Y are affine spaces, then every affine transformation $f : X \rightarrow Y$ is of the form $x \rightarrow M \cdot x + b$, where M is a linear transformation on the space X , x is a vector in X , and b is a vector in Y . Unlike a purely linear transformation, an affine map need not preserve the zero point in a linear space. Thus, every linear transformation is affine, but not every affine transformation is linear.

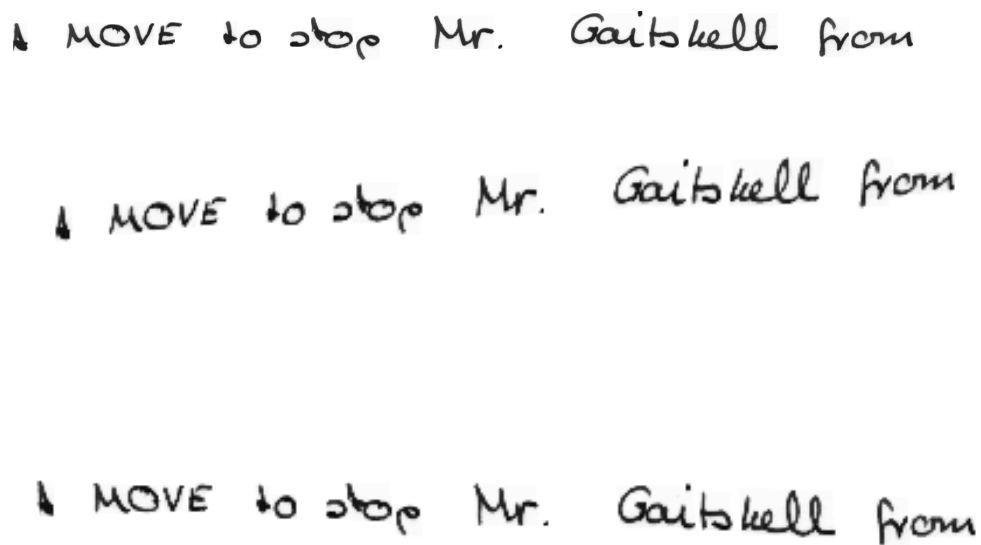


Figure 5.1: Linear Transformation - Affine Transform Case

5.2.2 Global Morphological Transform

The intuition behind the utilization of Morphological Transforms and especially Dilation and Erosion is to create thinner and denser letters, that we also meet in real-time cases.

We use sets to represent binary images and set operations to represent binary image transformations. Specifically, given a binary image, let the object be represented as X and its background by the set complement X^c . The *structural element* is symbolized with B .

$X_{+y} \triangleq \{x + y : x \in X\}$ is the *translation* of X along the vector y and $B^s \triangleq \{x : -x \in B\}$ is a *symmetric* of B with respect to the origin. [43]

Dilation

$$X \oplus B \triangleq \{z : (B^s)_{z+} \cap X \neq \emptyset\} = \bigcup_{y \in B} X_{+y} \quad (5.1)$$

Erosion

$$X \ominus B \triangleq \{z : (B^s)_{z+} \subseteq X\} = \bigcap_{y \in B} X_{-y} \quad (5.2)$$

Opening

$$X \circ B = (X \ominus B) \oplus B \quad (5.3)$$

Closing

$$X \bullet B = (X \oplus B) \ominus B \quad (5.4)$$

In applications, B is usually called a structuring element and has a simple geometrical shape and a size smaller than the image X . If B has a regular shape, e.g., a small disk, then both opening and closing act as nonlinear filters that smooth the contours of the input image. Namely, if X is viewed as a flat island, the opening suppresses the sharp capes and cuts the narrow isthmuses of X , whereas the closing fills in the thingulfs and small holes.

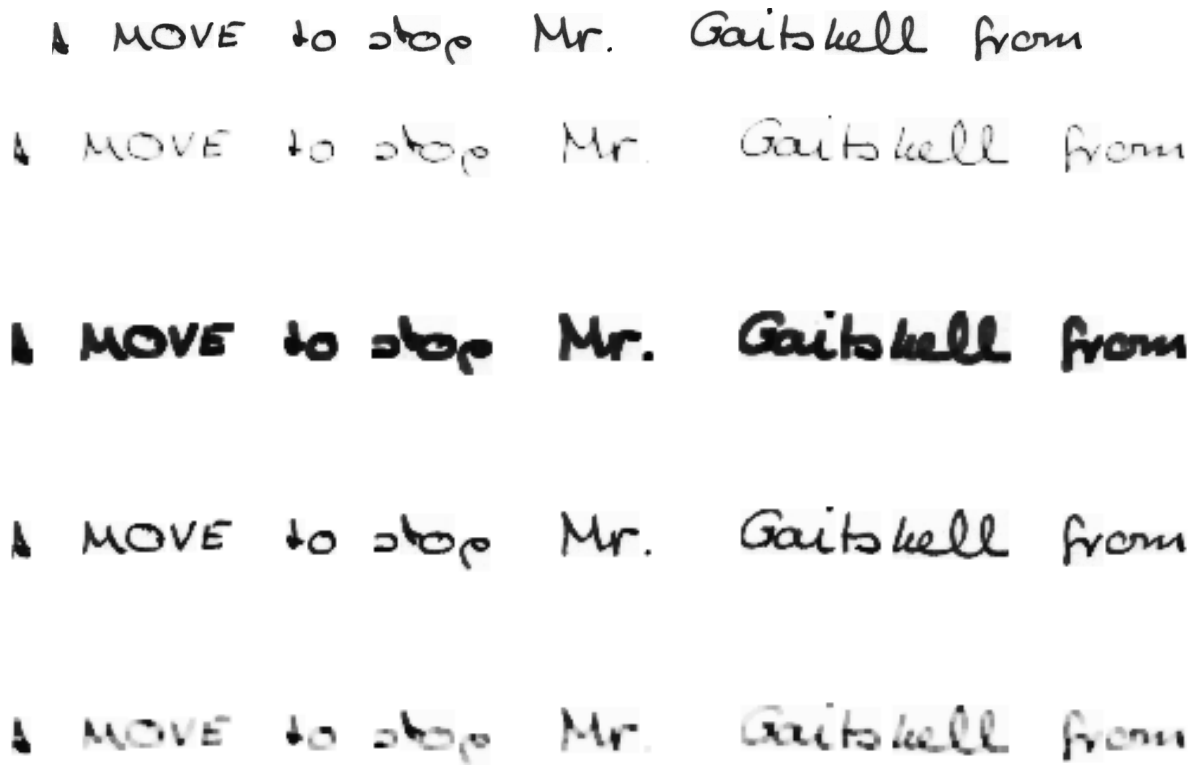


Figure 5.2: Non Linear Image Transformations based on Morphological Filtering

5.2.3 Local Affine Transform

The intuition behind the selection of this algorithm is to create local distortions on the letters and not in the line alone. For this reason, we split the image into a random number of vertical grids. On each grid a control point was chosen uniformly since we wanted all the points on it to have an equal probability to be selected as control points. After that, we apply an affine transform in each of these points and by utilizing the Thin Plate Spline Interpolation Technique [69] we obtain the final image.

5.2.4 Local Morphological Transform

The **Local Morphological** transform is simply made by splitting the image in a grid and applying randomly morphological filtering with random values in the kernel each time. So the resulted image has variations in the letter's width.

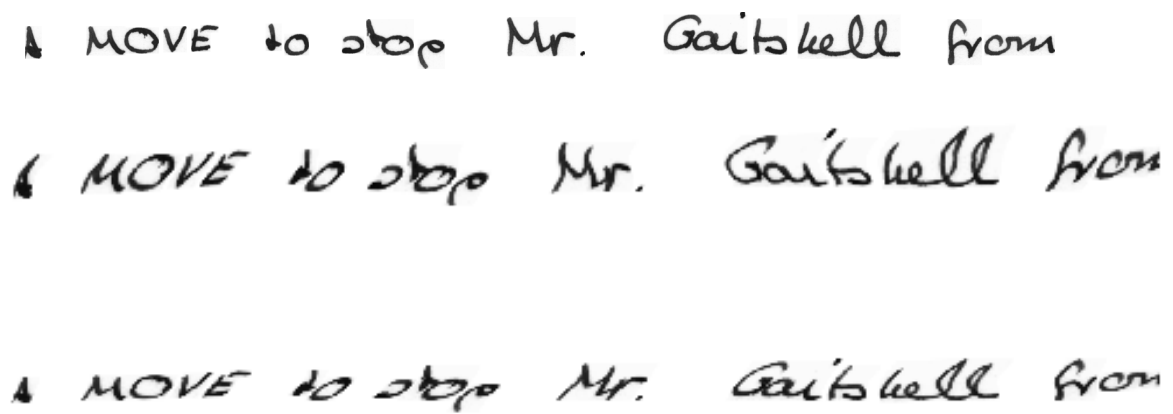


Figure 5.3: Local Affine Transform Samples

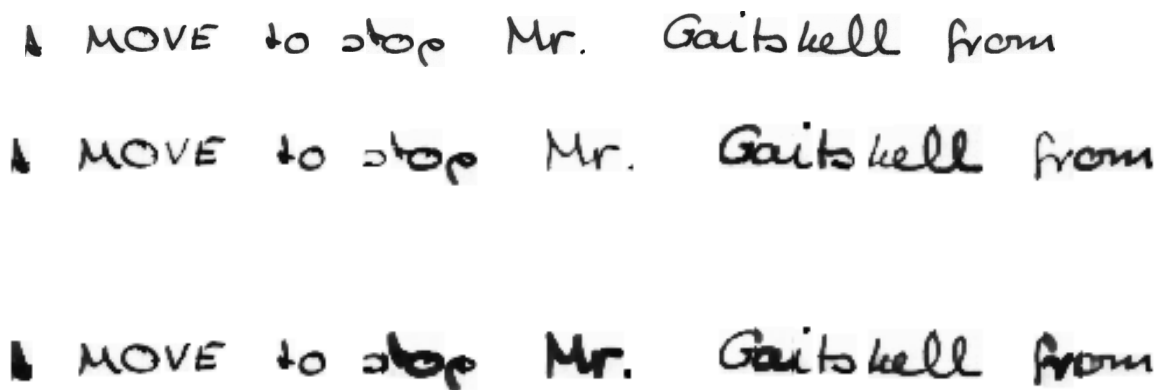


Figure 5.4: Local Morphological Transforms

5.2.5 Other Transforms

Gaussian Blurring

The **Gaussian Blur** is a type of image-blurring filter that uses a Gaussian function (normal distribution in statistics) for calculating the transformation to apply to each pixel in the image.

$$G(x, y) = \frac{1}{2 \cdot \pi \sigma^2} \cdot e^{-\frac{x^2 + y^2}{2 \cdot \sigma^2}} \quad (5.5)$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. When applied in two dimensions, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. Values from this distribution are used to build a convolution matrix that is applied to the original image. This convolution process is illustrated visually in the figure on the right. Each pixel's new value is set to a weighted average of that pixel's neighborhood. [9]

Noise Induction

Cut Out

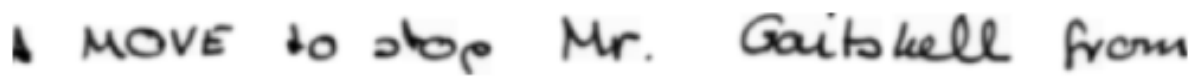
A handwritten sentence "MOVE to stop Mr. Gaitskell from" is shown with a Gaussian blur effect applied, making the characters soft and out of focus.

Figure 5.5: Gaussian Blurring

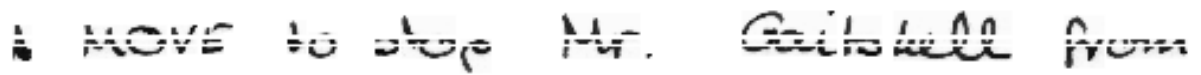
The same handwritten sentence "MOVE to stop Mr. Gaitskell from" is shown with a horizontal cutout effect, where the text is fragmented into several pieces with gaps between them.

Figure 5.6: Horizontal Cutout

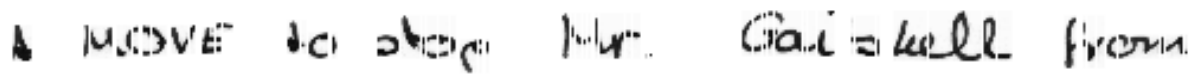
The same handwritten sentence "MOVE to stop Mr. Gaitskell from" is shown with a vertical cutout effect, where the text is fragmented into several pieces with gaps between them.

Figure 5.7: Vertical Cutout

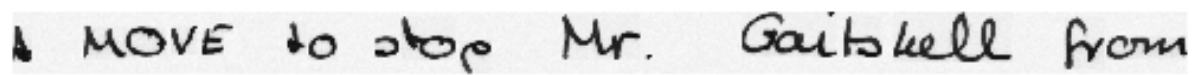
The same handwritten sentence "MOVE to stop Mr. Gaitskell from" is shown with a noise induction effect, where the text is overlaid with a semi-transparent grey rectangular area.

Figure 5.8: Noise Induction

5.3 Baseline Architecture

Handwritten Text Recognition is by its nature a monotonical sequential problem where we have to convert a sequence of visual features to a sequence of target units, either character level or ngram level etc. There are many sequence-to-sequence transduction approaches such as:

- Connectionist Temporal Classification
- Recurrent Neural Network Encoder-Decoder
- RNN Transducer

Among the three aforementioned end-to-end methods, CTC enjoys its training simplicity and is one of the most popular methods used in the contemporary HTR models and in speech community.

Our first goal is to implement a baseline architecture for HTR and perform experiments on it, so as to find a good configuration for the model. A single training example is an image of a text line and its corresponding transcript (ground truth). We make use of the CTC objective function to train our model. ¹

IAM corpus consists of 79 different characters. As we have mention in a previous chapter, CTC demands an additional character, the blank, so as to distinguish identical repetitive characters. So, overall we have 80 classes.

For example, the word 'Hello' is written as 'H-e-l-l-o'.

So the alignments have this appearance:

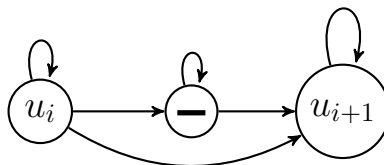


Figure 5.9: Unigram Alignments

Since we have determined the objective function, we need to compute $p(c|x, t)$, the predicted distribution over output characters c given the image features X at time t . While there are many function approximators, the recurrent nature of the task and the need to model long term dependencies leads to the use of Long Sort Term Memory Networks (LSTM). Specifically, we make use of Bidirectional LSTM Network since we care for the context from both sides when processing a feature [67].

¹We utilized the warp-ctc [10] [14]

Our architecture is described below :

1. **Convolutional Neural Networks for feature extraction (Visual Model)**

- We are utilizing CNNs so as to generate a number of Activation Maps of the current image.

2. **Map-to-Sequence**

- Converts the maps into feature vectors by taking the maximum activation in the dimension of height in every Feature Map.

3. **Bidirectional LSTM (Sequence Model)**

4. **CTC Layer**

- A Dense Layer that projects the output of LSTM layer into the dimension of our labels' space

For improving the performance we :

- add Batch Normalization Layer [38]
- perform Dynamic Data Augmentation (Affine,Morphological etc)
- Apply Dropout in the CTC Layer [70]

The use of Batch Normalization is widely used in the development of deep learning models because it prevents the network from being prone to fluctuating input data. This has the effect of making the model more robust while accelerating the training process. This is the normalization of activation costs based on the mean and the deviation resulting from the whole batch.

Dropout consists of randomly removing some neurons by resetting them during training. This way the network does not receive all the information but is required to output a result with a reduced version of it. As a result, the model generalizes better while avoiding overfitting in training data.

The detailed configuration of the total architecture is :

2 Conv Layers - 32 Kernels 2x2 - ReLU Max Pooling
4 Conv Layers - 64 Kernels 2x2 - ReLU Max Pooling
6 Conv Layers - 128 Kernels 2x2 - ReLU Max Pooling
2 Conv Layers - 256 Kernels 2x2 - ReLU 2 Conv Layers - 256 Kernels 2x2 - ReLU Max Pooling - Kernel Size hx1
3 Layers BiLSTM - Hidden Size 512 - Input Size 256

Table 5.1: End-to-End Architecture

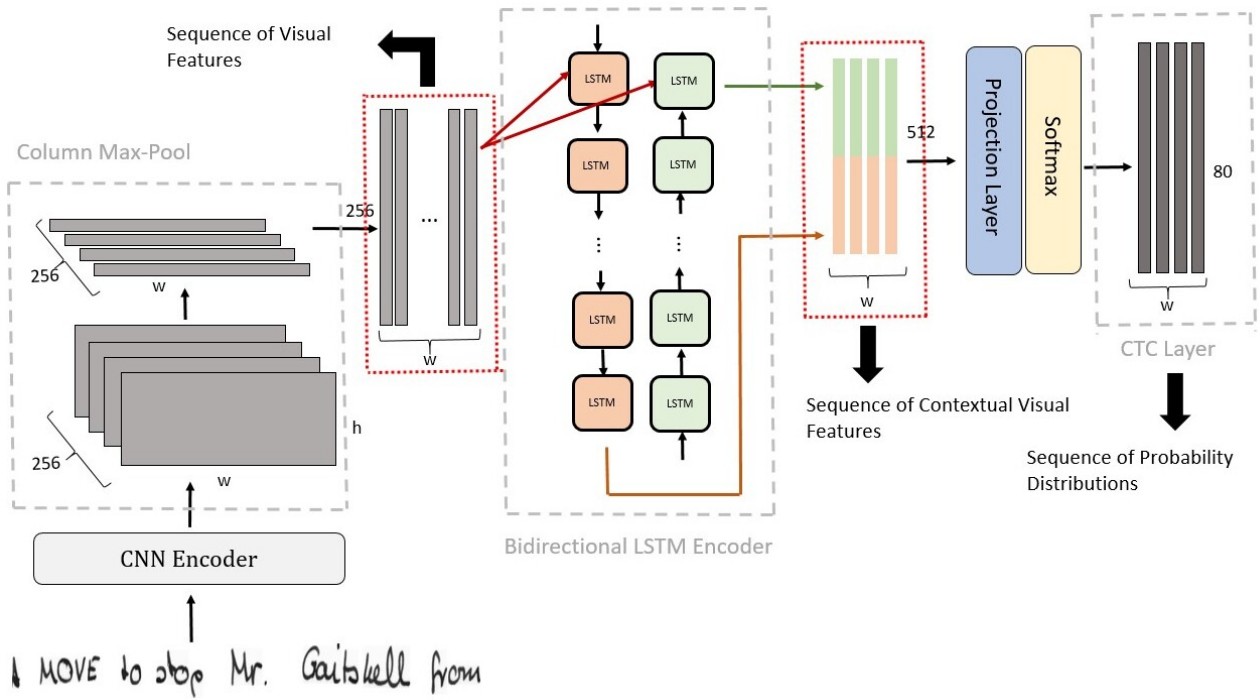


Figure 5.10: End-to-End Baseline Architecture for HTR

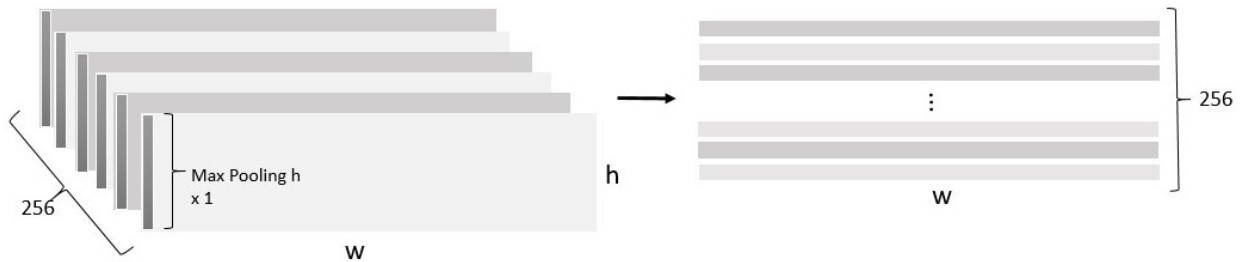


Figure 5.11: Map-to-Sequence Operation

For the decoding process we utilize the CTC Beam Search Decoding implementation. [1] The decoding algorithm is described here [28]

5.3.1 Experiments

The models are evaluated using the Word Error Rate (WER) and the Character Error Rate (CER) metrics. Let the decoded transcription be Y_{dec} and the ground truth text Y_{gt} of a text line and w_d and w_t the correspondent words on each text line.

The metrics are defined below:

$$CER = \frac{editdistance(Y_{dec}, Y_{gt})}{length(Y_{gt})} \quad (5.6)$$

$$WER = \sum_{w_d, w_t} \frac{editdistance(w_d, w_t)}{length(w_t)}, \forall w_d \in Y_{dec}, \forall w_t \in Y_{gt} \quad (5.7)$$

These metrics are based on Levenshtein distance [46]. The Levenshtein distance measures the difference between two sequences. For example, if two sequences differ by one letter, then the distance is 1. So the metric CER is essentially the ratio of the number of characters that differ, between the two sequences, to the total number of characters in the actual sequence. The metric WER is interpreted similarly, except that it is applied to all the words of the sentence separately and then summed.

BiLSTM Layers

Epochs	70
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	None
batch size	8

Table 5.2: Configuration of Data Augmentation Experiments

BiLSTM layers	WER	CER
1	0.2454	0.0708
2	0.2155	0.0631
3	0.2047	0.0599
4	0.2058	0.0612
5	0.2062	0.0615

Table 5.3: Experiments on BiLSTM Layers

From the above experiments, we consider that **3 bidirectional LSTM** layers ensure a good trade-off between the performance of the model, regarding the recognition rate, and the complexity.

Data Augmentation

Then, we applied all data augmentation techniques that we have implemented, in the overall system. The train and architecture configuration in the same for all the experiments bellow.

Layers	3
Epochs	70
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	2LSTM no augm
batch size	8

Table 5.4: Configuration of Data Augmentation Experiments

Data Augmentation	WER	CER
None	0.22185	0.0655
Global Affine	0.1951	0.0577
Global Morphological	0.2107	0.0624
Global Affine + Global Morphological	0.1913	0.0568
Local Affine	0.2034	0.0623
Local Morphological	0.2143	0.0690
Local Affine + Local Morphological	0.2109	0.0621
Nested Global Affine + Morphological	0.1925	0.0568
Nested Global Affine + Morphological Randomly applied to each frame in the batch	0.2006	0.0581

Table 5.5: The effect of Data Augmentation

5.4 Decoding the Network Outputs

5.4.1 External Language Model

A character-based CTC, with significantly fewer targets comparing to higher in scale target units, can naturally solve the OOV issue as the word output sequence is generated by collapsing the character output sequence. Because there is no constraint when generating the character output sequence, a character-based CTC, in theory, can generate any word. However, this is also a drawback of the character-based CTC because it can generate any ill-formed word. As a result, a character-based CTC without any LM and complex decoding usually results in very high word error rates (WER)

Moreover, CTC Loss holds property of conditional independence. The model assumes that every output is conditionally independent of the other outputs given the input. This is a bad assumption in sequence-to-sequence problems such as HTR/ASR because the outputs are dependent on one another since they are connected with either visual or language information.

So, we utilize a statistical character-level n-gram language models generated from the LOB and the Brown corpuses. We note that the Test Set that is contained in the LOB corpus was removed for the Word LM and Character Level LM generation. The probabilities of this model are inserted during the CTC Beam Search Decoding [28][1]

Except for the character-level language models, we also created statistical word-level language models with the KenLM Tool [36]. KenLM tool provides a library that implements two data structures for efficient language model queries, reducing both time and memory costs.

5.4.2 Decoding Experiments

Below we perform decoding of the same model’s posteriors with different algorithms, utilizing external language information, character and word level language model.

Decoding Algorithm	WER	CER
Greedy	0.2068	0.0608
CTC Beam Search 2 Char-LM	0.2050	0.0608
CTC Beam Search 3 Char-LM	0.1972	0.0590
CTC Beam Search 4 Char-LM	0.1814	0.0564
CTC Beam Search 4 Word LM	0.1481	0.0460

Table 5.6: Results of decoding with external language information in line level

In general, we observe that the higher the level of the statistical linguistic model, the better the recognition rate. This is due to the fact that as the language model's class grows, so does the history we take into account in calculating frequency occurrences. For example, the incidence of a given b is different and the incidence of a given bcd is different. In the latter case, the information is obviously larger and therefore justifies the more qualitative results we obtain using higher-order language models.

The biggest improvement is with the use of word-level language models. This is justified as words in the dictionary are favored and are therefore very likely to be constructed during decoding.

Chapter 6

Ngram Models

6.1 N-Gram Models

In this chapter we want to explore the below:

- Whether n-gram decomposition of the target sequence can be useful as a constraint in the decoding process
- How the existence of multiscale target units in a common CTC Layer can affect the CTC Learning process.

6.2 Unigram-Bigram Model

In our study, we focus on the case of unigram and bigrams. We are incrementing the target units with character level bigram classes. The number of bigrams selected and the way selected will be a hyperparameter for the problem and we are going to examine the effect of it in the model's performance.

Our motivation for studying this case is to examine whether or not the model can learn better visual features of letters when it is forced to learn more complex alignments. Secondly, we want to examine whether the bigram posteriors, used properly in the decoding, can correct some OCR errors.

6.3 Shared CTC Layer Architecture for Unigram-Bigram Model

At first we implement a simple architecture, as the baseline one, since we consider that the sequential model with 256 dimension hidden units is sufficient enough so as to learn bigram probabilities since bigram and unigrams hold common information.

The target alignments are formulated as below :

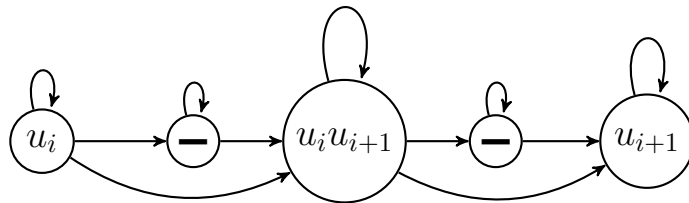


Figure 6.1: Alignments for Shared Unigram-Bigram Model under the standard CTC Framework [27]

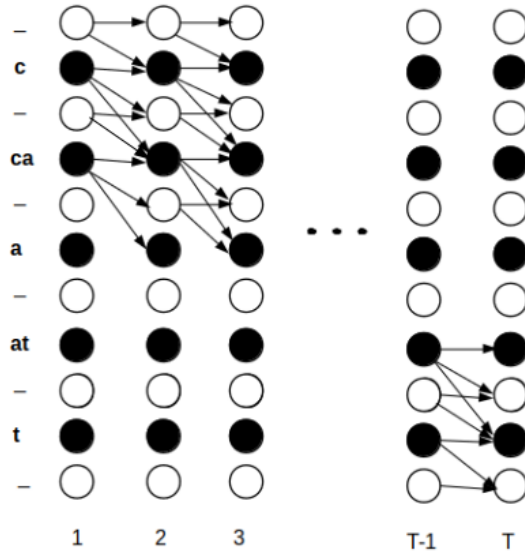


Figure 6.2: Computation of forward variable a_s^t for Unigram-Bigram Alignments under the standard CTC framework

What does it mean for the CTC Loss to calculate the probabilities over this specific alignment ?

We know that the CTC Loss marginalizes over the possible alignments that can be created from the target units, given a specific target text. When the number of target units increases, that means that more possible alignments will be formulated and the array a_s^t will have stored probabilities for more possible alignments s . However, the final probability $p(l|x)$ is the sum of all possible alignments, as this equation 3.5

What does the model learn ?

The aim of maximum likelihood training is to simultaneously maximise the log probabilities of all the correct classifications in the training set. That means minimising the following objective function:

$$- \sum_{(x,y) \in S} \ln(p(z|x)) \tag{6.1}$$

So, for evaluating the train loss, we need to keep in mind that the smaller the better. And when the loss gets smaller means that higher probability is assigned in the alignments that correctly fall into the groundtruth text

Why do we select these types of alignments ?

We want to constrain the choice of the next unigram, based on the bigram that appeared between them.

How do we select bigrams ?

The Intuition behind the bigram selection policy is the bellow.

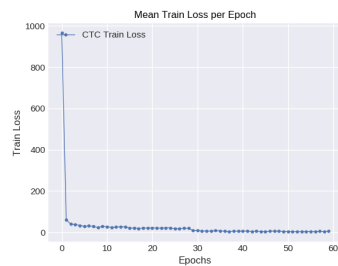
- Just like a character level LM, we care only about the lower case letters, so we will focus on these bigrams.
- Bigrams from all the 79 unigram classes will be a computational overhead both for the network and the decoding algorithm.
- Observing handwritten datasets, the cursive nature of them is mainly spotted in the words, neither in numbers nor in symbols.

Thus, our targets for constraining the unigrams, are the bigrams formulated from lowercase letters. These bigrams are totally $26 * 26 = 676$. From them, we can explore different schemes, such as the n most frequent etc. Bellow are some examples :

So, bellow we can see all the alignments of word 'Hello' for different number of bigrams.

- **50 Bigrams** : H-e-el-l-ll-l-o
- **100 Bigrams** : H-e-el-l-ll-l-lo-o
- **150 Bigrams** : H-e-el-l-ll-l-lo-o
- **200 Bigrams** : H-He-e-el-l-ll-l-lo-o
- **All Bigrams** : H-He-e-el-l-ll-l-lo-o

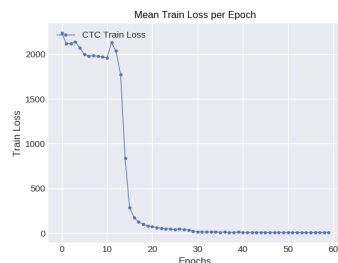
Training Plots of Shared CTC Layer



(a) Mean CTC Train Loss for 100 Bi-grams + Unigrams



(b) Mean CTC Train Loss for 100 Bi-grams + Unigrams in last epochs



(c) Mean CTC Train Loss for 150 Bi-grams + Unigrams



(d) Mean CTC Train Loss for 150 Bi-grams + Unigrams in last epochs

From the above plots, we conclude that CTC Loss is slightly affected from the number of target units in Unigram-Bigram Case.

Then, we have to test whether the model’s performance has changed with the addition of bigrams as target units. One possible way to prove this is, is to perform simple greedy search in the decoding, by expressing only the unigram and ignoring the bigrams.

Layers	5
Epochs	60
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	GA3layers
batch size	8

Table 6.1: Configuration of Unigram-Bigram Experiments

CTC Layer	WER	CER
Unigram	0.2068	0.0608
Unigram + 50 Bigrams	0.1903	0.0570
Unigram + 100 Bigrams	0.1988	0.0593
Unigram + 150 Bigrams	0.1995	0.0593
Unigram + 200 Bigrams	0.1974	0.0583
Unigram + 676 Bigrams	0.2002	0.0601

Table 6.2: Greedy decoding performance utilizing only unigrams and ignoring bigrams

From the above experiments, we can conclude that the model benefits from the addition of bigrams, but this happens only for a small amount of them. So, from the one side, it can learn some bigrams at the same time with unigrams but not all bigrams since it cannot encode them in the same CTC Layer. This can be seen from both the train plots and decoding. In the train plots we see that the CTC Loss finds it difficult to converge to a smaller value and from the decoding, the WER/CER recognition is getting better only with a small number of bigrams.

6.3.1 Proposed Algorithms for Decoding Unigram-Bigram Model

Now that we have used additional classes in the output, and these share common information with the unigram classes we need to implement algorithms that are adapted to this change and also utilize the bigram information in several ways. There are two ways of utilizing bigrams in the decoding. The first one is to utilize bigram probability for extending the bigrams and bigram probability for scoring each unigram extension when applied.

The first step that we need to perform before start writing down the decoding algorithm, is to think about what the network has trained to learn. In our case, we feed in the CTC Loss, alignments that between unigrams, place a bigram if it exists. For example, the word 'Hello' is converted into 'H-He-e-el-l-l-lo-o'. In our case though, where we do not consider all bigrams in the target units, a missing bigram is represented by the blank character.

So, we conclude that after a unigram we should observe either blank characters or a bigram. In this idea we will support the first algorithm which utilizes bigram posteriors as a score for enhancing the transition between two unigrams through the probability of the contained bigram.

Note: We are developing algorithms with the assumption that the networks does not ignore in the output more than one target units. That means that we take into account all the cases of alignments that one unigram or one bigram is ignored, although existed in the targets classes.

Example :

- **a-ab-b-cd-d** : In this case, unigram **c** is not expressed
- **a-b-bc-cd-d** : In this case, bigram **ab** is not expressed

The **first algorithm** utilizes the unigram posteriors for extending at each time step every unigram and the bigram posteriors for scoring the extension of each unigram. The bigram scoring can be defined with several different ways.

Let us denote as u_i the last character of the already formed string, s , that was selected at time t_i and as u_{i+1} the character to extend at time t_{i+1} . Let \mathcal{BS} be the *Bigram Score* for the extension of string s with the character u_{i+1} . This is defined as follows:

$$\mathcal{BS} = \max(Pr(u_i u_{i+1}, t_i : t_{i+1} | x)) \quad (6.2)$$

The bigram score can also have alternative definition as follows:

Let us denote as u_i the character to extend at time t and w a window size in the time dimension. We symbolize B the set of available bigrams starting from u_i and as C the set of bigrams ending with u_i . The \mathcal{BS} *Bigram Score is defined* as follows: defined as follows:

1st case : Looking Forward

$$\mathcal{BS} = \max_t^{t+w} \max_{b \in B} Pr(b, t | X) \quad (6.3)$$

2nd case : Looking Backward

$$BS = \max_{t-w}^t \max_{c \in \mathcal{C}} Pr(c, t|X) \quad (6.4)$$

3rd case : Looking in Both directions

$$BS = \max \left(\max_{t-w}^t \max_{c \in \mathcal{C}} Pr(c, t|X), \max_t^{t+w} \max_{b \in \mathcal{B}} Pr(b, t|X) \right) \quad (6.5)$$

Algorithm 3 CTC Beam Search - BigramScoring

```

1: procedure CTC BEAM SEARCH( $P, W$ ) ▷  $P$  is the Posterior Probabilities' Matrix,  $W$  is the Beam Width
2:    $B \leftarrow \{\emptyset\}, Blank(\emptyset, 0)$ 
3:   for  $\tau=1 \dots T$  do
4:      $D \leftarrow BestBeams(B, W)$ 
5:      $Beams \leftarrow \{\}$ 
6:     for  $b$  in  $D$  do
7:       if  $b \neq \emptyset$  then
8:          $NonBlank(b, \tau) \leftarrow NonBlank(b, \tau - 1) \cdot P[b^e, \tau, |X]$ 
9:       end if
10:       $Blank(b, \tau) \leftarrow Total(b, \tau - 1) \cdot P[blank, \tau|X]$ 
11:       $AddBeam(newbeam, b)$ 
12:      for  $character$  in  $Characters$  do
13:         $NewBeam \leftarrow b + character$ 
14:         $Blank(newbeam, \tau) \leftarrow 0$ 
15:        if  $k = b^e$  then
16:           $NonBlank(newbeam, \tau) = P[k, \tau|x] \cdot Blank(b, \tau - 1)$ 
17:        else
18:           $NoneBlank(newbeam, \tau) = P[k, \tau|x] \cdot Total(b, \tau - 1)$ 
19:        end if
20:         $AddBeam(newbeam, b)$ 
21:         $BigramScore(newbeam, b, \tau)$ 
22:      end for
23:    end for
24:  end for
25:  return  $max_{x \in Beams} Total(y, T) \cdot BS(y, T)$ 
26: end procedure

```

Algorithm 4 BigramScoring Procedure

```

1: procedure BIGRAMSCORE( $NewBeam, b, t$ )
2:    $bigram = b[-1] + NewBeam[-1]$ 
3:   if  $bigram \in Bigrams$  then
4:      $TransitionProb = BackwardBS(bigram)$ 
5:   end if
6:   for  $bigram$  in  $Forward Bigrams$  do
7:      $ForwardProb = ForwardBS(bigram)$ 
8:   end for
9:   return  $max(TransitionProb, ForwardProb)$ 
10: end procedure

```

In the second algorithm we create the substring exploiting all the possible cases that a transcript can be created from, either unigrams or bigrams.

The second algorithm that exploits both unigrams and bigrams for extending each time is more complex and we need to define some rules first in order to make it work properly.

- When the next character is a bigram, we only extend by bigram and properly adapt the already formed string. This is due to the fact that when we need to copy the bigram in the next step we need to find it as a last character (Instead of extending unigram and annotating that this unigram has come from a bigram)
- When the last character is unigram, we will now that it will be the outcome of a unigram extension only. Last Unigram is always result of a unigram Extension.
- Last Bigram is only result of a bigram extension.

A common extension in the algorithm is to first extend by unigram and then substitute this unigram with the correspondent bigram. This is legit, since the alignments are formulated like this a-an-n, and so the algorithm meets first the highest probability in the 'a' character and then in the 'an'. So, we conclude that a unigram can be substituted by bigram and a bigram can be substituted by unigram since the last unigram if the bigram can be wrong. (Case : c-a-an-m-e for came, 'an' bigram is wrongly recognized here)

During the development of the algorithm we observed that probabilities were downgrading after a bigram extension. Because bigram plays the role of blank character we end up that for keeping high the probability of the bigram as a non blank character we need to include in the Non Blank Probability a portion of the Blank. And that is due to the dual role of the bigram. So, a conclusion is that the existence of both bigrams and blanks is redundant. This will lead to the final proposal of a new ngram etc that will learn alignments without blank character.

Algorithm 5 CTC Beam Search - Unigrams and Bigrams Extension

```
procedure CTC BEAM SEARCH( $P, W$ )  ▷  $P$  is the Posterior Probabilities' Matrix,  $W$  is the Beam Width
   $B \leftarrow \{\emptyset\}, Blank(\emptyset, 0)$ 
  for  $t=1 \dots T$  do
     $D \leftarrow BestBeams(B, W)$ 
     $Beams \leftarrow \{\}$ 
    for  $b$  in  $D$  do
      if  $b \neq \emptyset$  then
        if lastchar is Bigram then
           $NonBlank(b, t) \leftarrow NonBlank(b, t-1) \cdot (P[b, t|X] + P[b^e, t|X]) + a \cdot Total(b, t-1) \cdot$ 
 $P[blank, t|X]$ 
        else
           $NonBlank(b, t) \leftarrow NonBlank(b, t-1) \cdot P[b^e, t|X]$ 
        end if
      end if
       $Blank(b, t) \leftarrow Total(b, t-1) \cdot P[blank, t|X]$ 
       $AddBeam(NewBeam, b)$ 
      for  $k$  in TargetUnits do
        if  $k$  is Unigram then
          if  $b^e$  is Unigram then
             $ExtendUnigramByUnigram(b, k)$ 
          else
             $ExtendBigramByUnigram(b, k)$ 
          end if
        else
          if  $b^e$  is Unigram then
             $ExtendUnigramByBigram(b, k)$ 
          else
             $ExtendBigramBigram(b, k)$ 
          end if
        end if
      end for
    end for
  return  $max_{x \in Beams} Total(y, t)$ 
end procedure
```

Algorithm 6 Unigram-Bigram Procedure 1

```
1: procedure EXTENDUNIGRAMBYUNIGRAM( $b, k$ )
2:    $NewBeam \leftarrow b + k$ 
3:    $Blank(NewBeam, t) \leftarrow 0$ 
4:   if  $k = b^e$  then
5:      $NonBlank(NewBeam, t) = P[k, t|x] \cdot Blank(b, t-1)$ 
6:   else
7:      $NonBlank(NewBeam, t) = P[k, t|x] \cdot Total(b, t-1)$ 
8:      $AddBeam(NewBeam, b)$ 
9:   end if
10: end procedure
```

Algorithm 7 Unigram-Bigram Procedure 2

```
1: procedure EXTENDBIGRAMBYUNIGRAM( $b, k$ )
2:   if  $b^e = k$  then
3:     continue ▷ th h is th
4:   else
5:      $NewBeam \leftarrow b + k$  ▷ ab c is abc
6:      $NonBlank(NewBeam, t) \leftarrow P[k, t|x] \cdot Total(b, t - 1)$ 
7:      $AddBeam(NewBeam, b)$ 
8:   end if
9: end procedure
```

Algorithm 8 Unigram-Bigram Procedure 3

```
1: procedure EXTENDUNIGRAMBYBIGRAM( $b, k$ )
2:    $a_1 = k[0]$ 
3:    $a_2 = k[1]$ 
4:   if  $b^e = a_1$  then ▷ a ab is ab
5:      $NewBeam = b + a_2$ 
6:      $NonBlank(NewBeam, t) \leftarrow P[k, t|x] \cdot Total(b, t - 1)$ 
7:      $AddBeam(NewBeam, b)$ 
8:   end if
9:    $NewBeam = b + k$  ▷ a bc is abc
10:   $NonBlank(NewBeam, t) \leftarrow P[k, t|x] \cdot Total(b, t - 1)$ 
11:   $AddBeam(NewBeam, b)$ 
12:   $NewBeam = b[: -1] + k$  ▷ a bc is bc
13:   $NonBlank(NewBeam, t) \leftarrow P[k, t|x] \cdot Total(b, t - 1)$ 
14:   $AddBeam(NewBeam, b)$ 
15: end procedure
```

Algorithm 9 Unigram-Bigram Procedure 4

```
1: procedure EXTENDBIGRAMBYBIGRAM( $b, k$ )
2:    $a_1 = k[0]$ 
3:    $a_2 = k[1]$ 
4:   if  $b^e = a_1$  then
5:      $NewBeam \leftarrow b + a_2$  ▷ ab bc is abc
6:      $Blank(NewBeam, t) \leftarrow 0$ 
7:      $AddBeam(NewBeam, b)$ 
8:   else
9:      $NewBeam \leftarrow b + k$  ▷ ab cd is abcd
10:     $NonBlank(NewBeam, t) \leftarrow P[k, t|x] \cdot Total(b, t - 1)$ 
11:     $AddBeam(NewBeam, b)$ 
12:  end if
13: end procedure
```

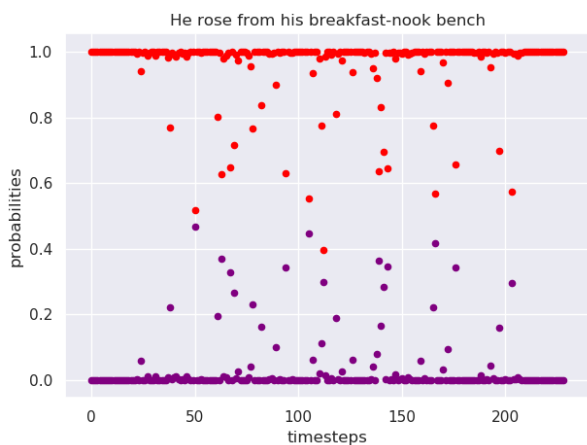
6.3.2 Output Analysis for Revision of Proposed Algorithms

Probabilities comes from the **Unigram + 50 Bigrams Model**.

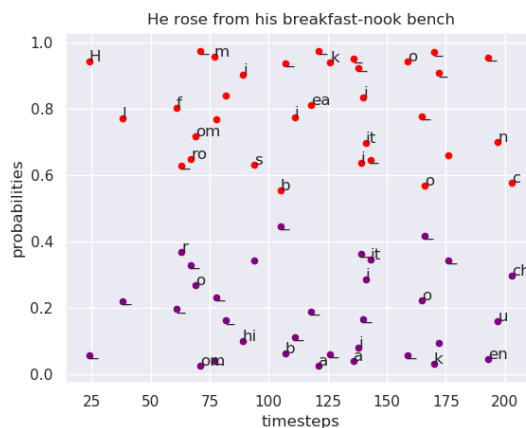
Test Set Example 1

Original : He rose from his breakfast-nook bench

Greedy Decoded : He lose from his beakfit-nook bench



(e) Top 2 Probabilities



(f) Probabilities between 0.55-0.98 and their transcripts



(g) Greedy Alignments

He rose from his breakfast-nook bench

(h) Test Image 1

Figure 6.3: Example 1

Problem Detected 1 : In timestep approx **30** *l* holds a big probability as a unigram. However, looking forward in a window size of **20** we do not see any bigram probability starting from *l*, although there are bigrams starting from *l* in the target units. This case outlines misrecognition, and this should be surpassed by our proposed algorithms.

Timestep	Text	Next Char	CTC Score	Bigram Score	Total Score
35	He	l	$6.86 \cdot 10^{-8}$	$3 \cdot 10^{-4}$	$2.058 \cdot 10^{-3}$
	He	r	$3.43 \cdot 10^{-8}$	0.0329	0.1128
37	He	l	1.0898	$3 \cdot 10^{-4}$	$3.2694 \cdot 10^{-4}$
	He	r	0.0104	0.0329	$3.4216 \cdot 10^{-4}$

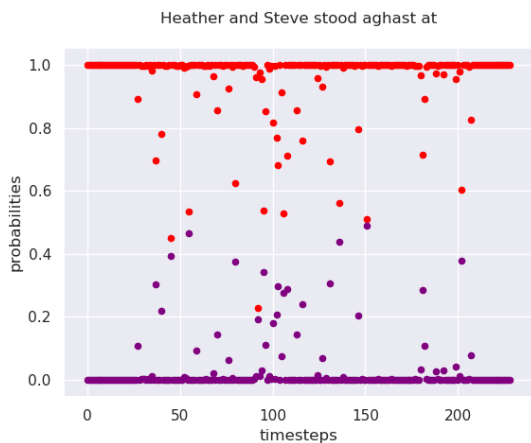
Table 6.3: Bigram scoring algorithm effect

However, because at the next timestep the *l* probability was high - the model was 100% sure about this expression - the *l* finally was expressed in the output target.

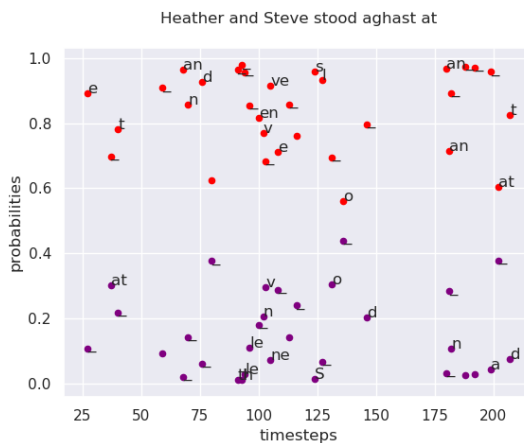
Test Set Example 2

Original : Heather and Steve stood aghast at

Greedy Decoded : Heatler and Meve ood aglan at



(a) Top 2 Probabilities



(b) Probabilities between 0.55-0.98 and their transcripts



(c) Greedy Alignments

Heather and Meve stood aglan at

(d) Test Image 3

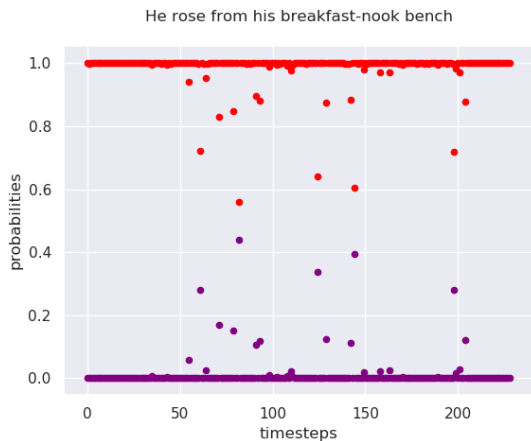
Figure 6.4: Example 2

Probabilities comes from the **Unigram + 150 Bigrams** Model.

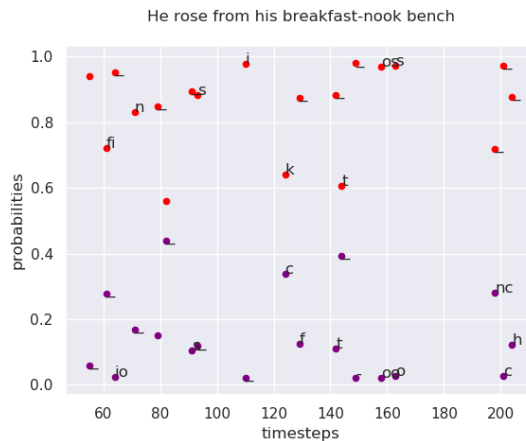
Test Set Example 1

Original : He rose from his breakfast-nook bench

Greedy Decoded : H e l loos s see f fii iooon n hhi iis s b iieeee aac k f faaai iit t - n nooos s k b een n nc c chh



(a) Top 2 Probabilities



(b) Probabilities between 0.55-0.98 and their transcripts



(c) Greedy Alignments

He rose from his breakfast-nook bench

(d) Test Image 1

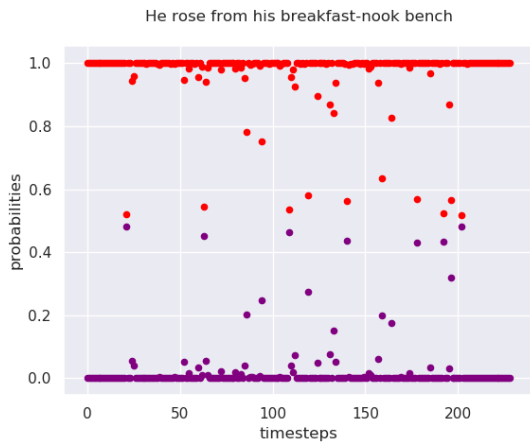
Figure 6.6: Example 4

Probabilities comes from the **Unigram + 200 Bigrams** Model.

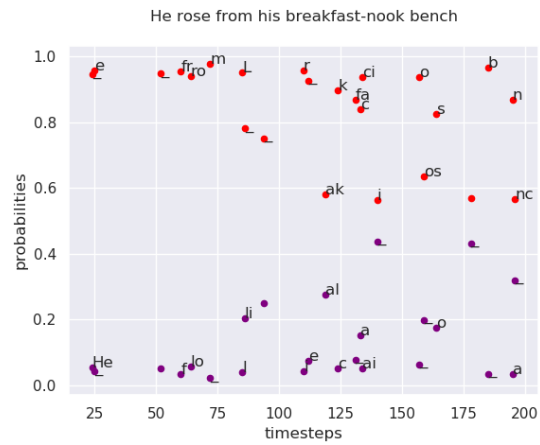
Test Set Example 1

Original : He rose from his breakfast-nook bench

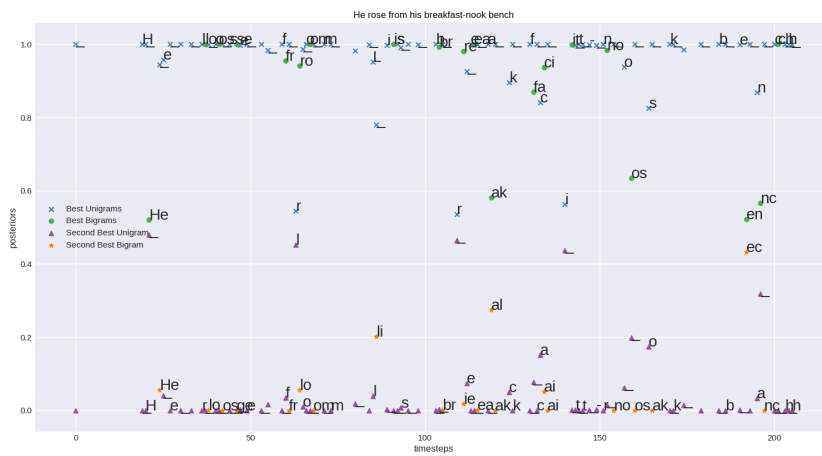
Greedy Decoded : H He e llo oos ssee ffr rro oom m L iiss bbr rre eea ak k ffa cci iit t - nno oos s k b een nnc ech h



(a) Top 2 Probabilities



(b) Probabilities between 0.55-0.98 and their transcripts



(c) Greedy Alignments

He rose from his breakfast-nook bench

(d) Test Image 1

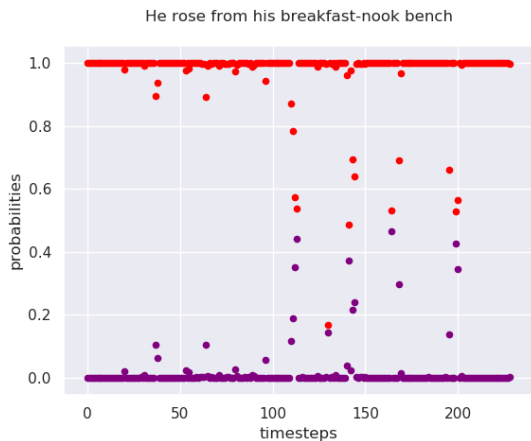
Figure 6.7: Example 5

Probabilities comes from the **Unigram + 676 Bigrams** Model.

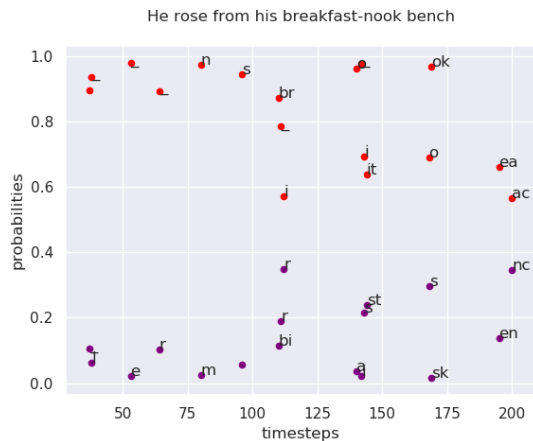
Test Set Example 1

Original : He rose from his breakfast-nook bench

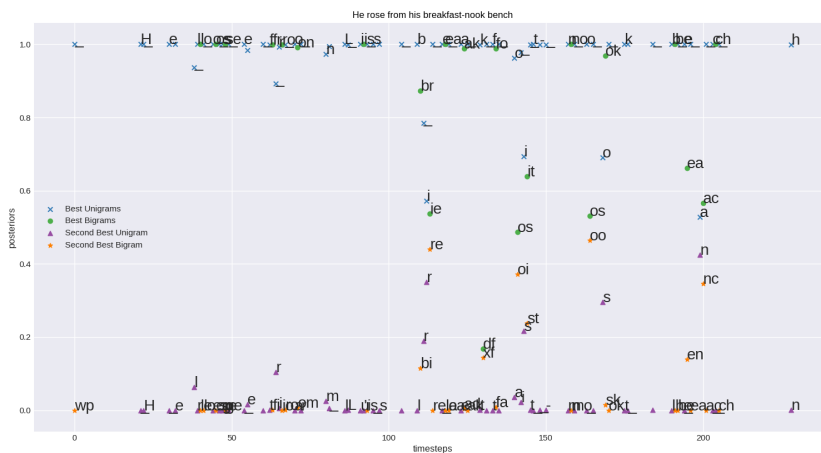
Greedy Decoded : H e llo oos sse e ffr rro on n L iis s bbr iie eea aak kdf ffo oos iit t- nno oos ook k bbe eea aac cch h



(a) Top 2 Probabilities



(b) Probabilities between 0.55-0.98 and their transcripts



(c) Greedy Alignments

He rose from his breakfast-nook bench

(d) Test Image 1

Figure 6.8: Example 6

Further Analysis of Alignments

What are our target errors that we want to correct?

Bellow we showcase a problematic case that we target to solve by utilizing our proposed algorithms. This is when a unigram has a mismatch with bigram, either looking forward or looking backward or in both directions.

That means that we have 3 kinds of mismatches.

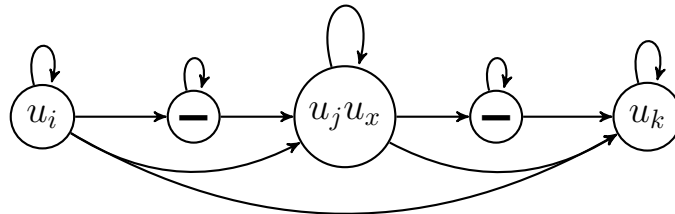


Figure 6.9: Both Directions Mismatch

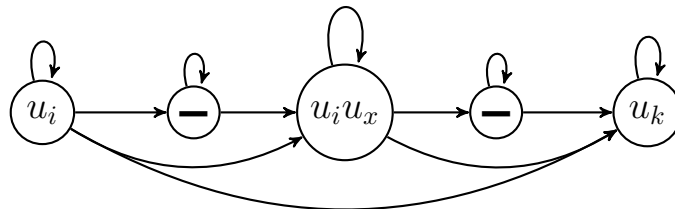


Figure 6.10: Backward Mismatch

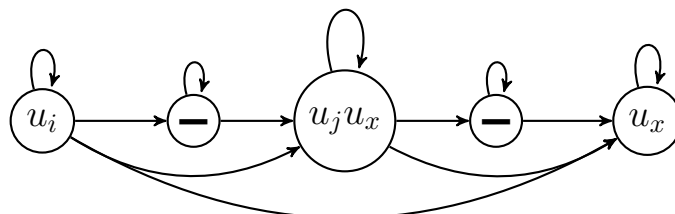


Figure 6.11: Forward Mismatch

So, in order to see some improvement in the decoding we need to find cases that bigrams can correct a wrongly recognized unigram. So we need to count all the mismatch cases.

After calculating the mismatch instances, we found out that only a very small percentage of mismatches occur in the test set. That explains why our proposed algorithms lead to same results as decoding only unigrams.

6.3.3 Decoding Experiments

Model	Algorithm	WER - CER
50 Bigrams	Bigram-Scoring	0.2100-0.0626
	Unigram-Bigram Extension	0.2090-0.0591

Table 6.4: Experiments on proposed algorithms for shared unigram-bigram model

6.4 Conclusions

We can conclude that bigrams, the way modeled, were not helpful in the decoding process since in most amount of cases the bigrams were always following the unigrams although both or one of them were wrongly recognized. So, there was no additional information to leverage from bigrams that could lead to an improvement.

This approach has the drawback that the alignment between unigrams and bigrams demand a blank character since CTC Loss is this way implemented. This is unnecessary if we keep in mind the reason that the blank character was invented. Blank Character was utilized so as to allow consecutive identical characters to appear in the decoding. However, when we are utilizing unigrams and bigrams this is not valid since the presence of bigrams allows the existence of consecutive identical unigrams.

For example :

- H-e-el-l-l-lo-o. 'll' bigram absent from bigrams' set. So blank character is needed in the alignment.
- H He e el ll l lo o. Bigram Set contains all the possible bigrams created from the unigrams. Here the blank character is not needed.

The last case generates the need of the creation of a modified CTC Loss where the presence of blank character into the alignment will be depended on the set of bigrams.

All in all, the future directions of this part of the work may include the development of a new CTC Loss similar to the Gram-CTC [42] approach for learning a hybrid segmentation of a word into unigrams, bigrams, trigrams etc.

Chapter 7

Hierarchical and Multitask Learning in HTR

7.1 Hierarchical and Multitask Approaches in HTR

As we saw in the previous chapter, forcing the network to learn more complex alignments that are composed of higher level scale target units do not lead to a significant improvement neither in the decoding process nor in the learning one. In this chapter we focus on decoupling the higher level target units from the unigrams and study how the additional information can affect the learning process by learning better representations that will be more able to discriminate optically identical cursive characters.

A different approach for combining multiscale target units in an end-to-end neural architecture are the Multitask Architectures and the Transfer Learning Technique. In the first one we utilize pretrained models that are taught to recognize smaller, in scale, target units and transfer this knowledge for learning higher level at scale ones. In the second approach, we train simultaneously all the target in all the scales.

7.1.1 Possible Architectures for Multiscale-Target Models

- **Multitask Architectures**, includes models that are trained jointly to different scales of target units.
- **Transfer Learning**. That means, training a network to learn unigram alignments and then utilize Transfer Learning from the unigram level so as to be trained the with bigrams etc.

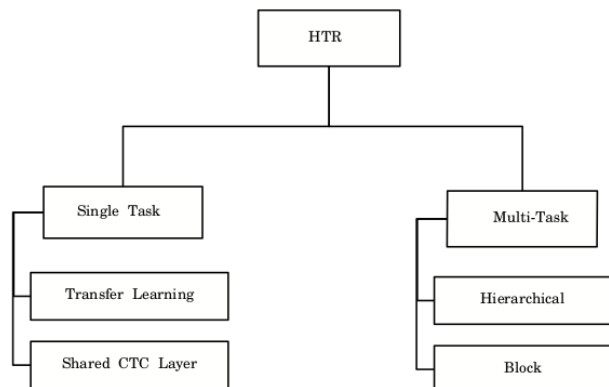


Figure 7.1: A taxonomy of implemented architectures for HTR

7.1.2 Hard Parameter Sharing for Unigram-Bigram Model

The second architecture for the Unigram-Bigram model applies **Transfer Learning** for training a Bigram HTR model. The pretrained model is a unigram-level 3 Layer BiLSTM. The model is forced to learn "sliding-window" alignments.

For the bigram layer, our classes are all the possible bigrams, that can be created from all the IAM unigrams plus the blank character. That means, that the output CTC layer has $79 \times 79 + 1$ classes, so 6242 classes. The alignments are formulated as bellow :

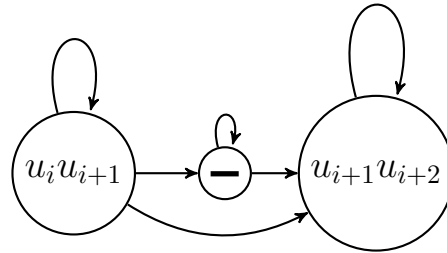


Figure 7.2: Alignments for Bigrams

So, the alignment for word 'Hello' is : **He-el-l-lo**

Training and Decoding the Network Outputs

For Training Bigram Models we utilize Transfer Learning from a Unigram Model and apply Data Augmentation Techniques. We train the models for 20 epochs, since we believe that there needs a small number of epochs for the Bigram Model to infer the bigram posteriors from the Unigram ones.

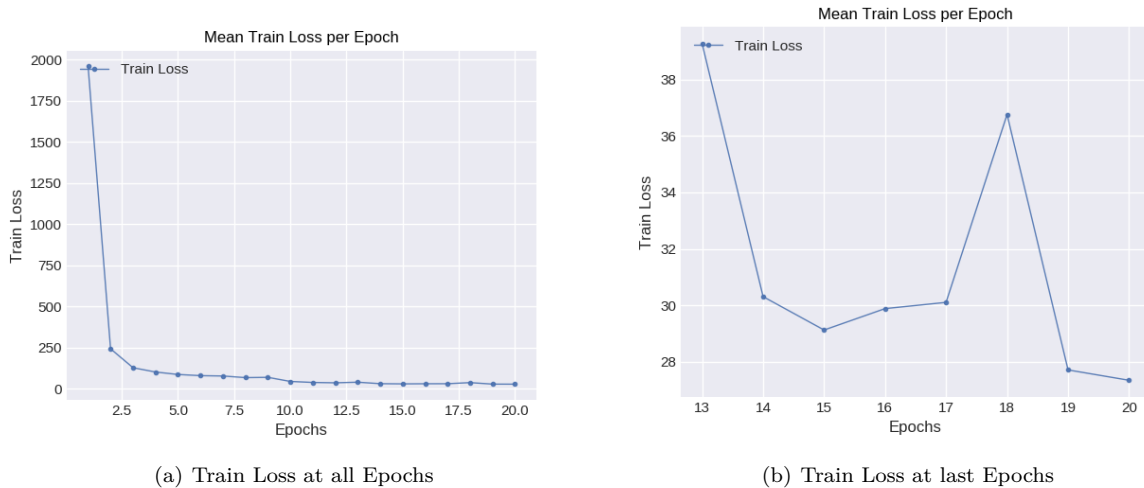


Figure 7.3: Train loss in bigram model training

In the terms of Decoding, in the learned alignment there are 3 different decoding schemes that can be applied. If the consecutive selected units share a common unigram (He - el), we merge. However if they do not, there are the choices bellow :

- Accept them both, for example $Ha - el \leftarrow Hael$

- Accept the already existed character and extend by the last unigram of the new, for example $Ha - el \leftarrow Hal$ (Greedy-2)
- Accept the new unigram, substitute the old one with it and extend by the last unigram of the new target, for example $Ha - el \leftarrow Hel$ (Greedy-3)
- Decide what to accept based on the Character Level LM probability. (Greedy-4)

We perform decoding experiments on the above trained bigram model based on these schemes.

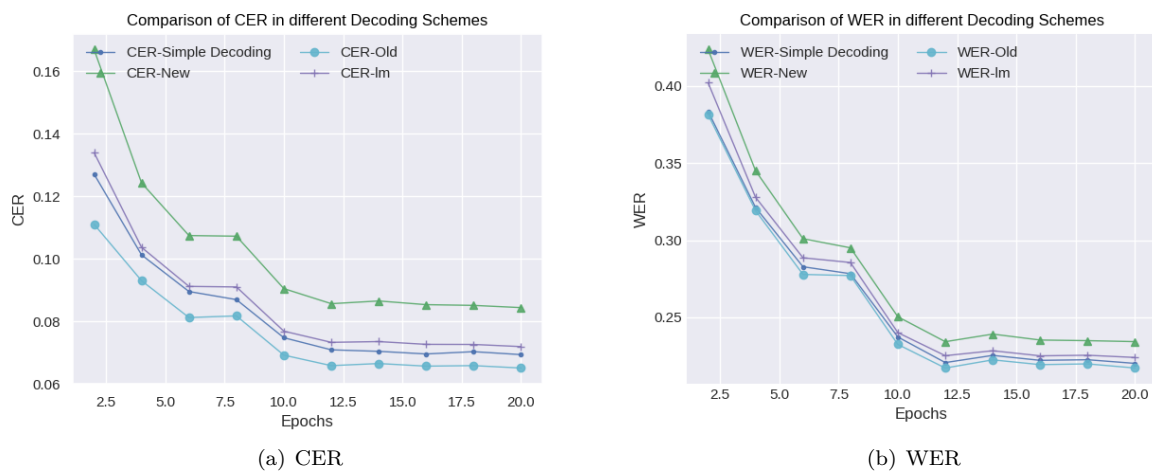


Figure 7.4: Comparison WER/CER in proposed bigram decoding schemes

Layers	5
Epochs	20
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	3 Layers Unigram Model
batch size	8

Table 7.1: Configuration of data augmentation experiments on bigram model

Data Augmentation Technique	WER	CER
Global Affine	0.2459	0.0720
Global Affine + Global Morph	0.2512	0.083
Global Augmentation per image frame on batch	0.2171	0.0651

Table 7.2: Data augmentation in bigram Model - greedy decoding (Greedy-Old)

Visualization of Bigram Alignments Comparing Unigrams

In this section we want to compare the relative position between bigram and unigram alignments, in order to see whether a bigram is expressed between unigrams when the network has not learned it this explicitly through the training.

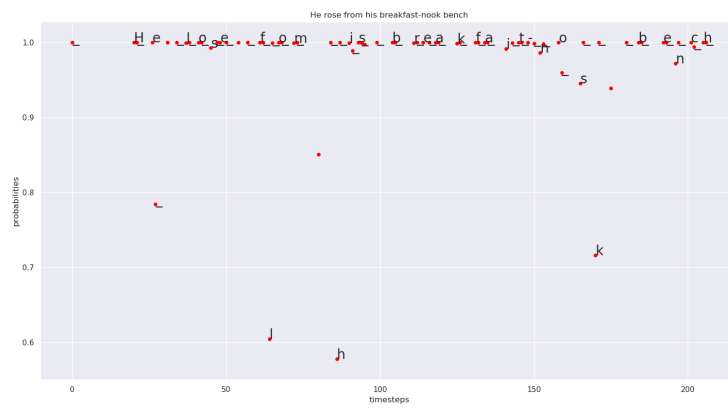


Figure 7.5: Unigram Greedy Alignment

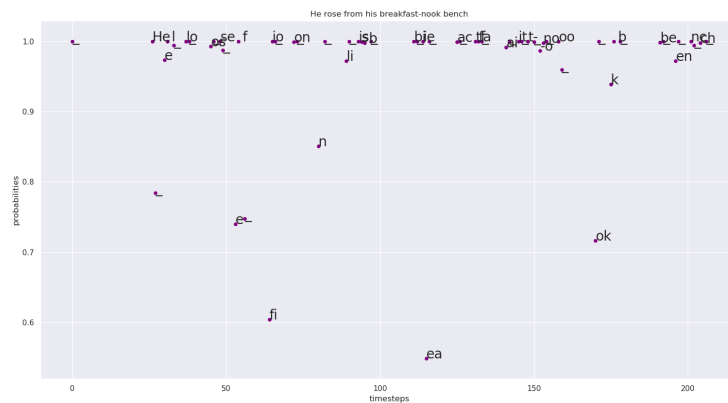


Figure 7.6: Bigram Greedy Alignment



Figure 7.7: Unigram and Bigram Alignments

Probabilities come from the bigram model with the best scores from the table.

From the above plots, we can see that although bigrams are not trained to appear between unigrams, we see that CTC localizes them in the same place as when they were trained in the shared CTC Layer.

This approach has several drawbacks and due to them we will not keep on experimenting.

- LSTM and in general RNNs need task-specific fine tuning when pretrained models are used. This can lead to the so called *catastrophic forgetting*. That means that it is probable to lose the unigram information so as to perform a simple decoding. If we want to perform decoding with higher level target units such as bigrams we need to include them all as target units and this is kind of exploding in the amount of target units. Moreover, this approach is difficult to scale in trigrams, fourgrams etc for the same reason.
- Greedy bigram decoding do not perform as good as unigram decoding.

7.2 Hierarchical Multitask Learning with CTC - An Application to HTR

Introduction

In Machine Learning (ML), we typically care about optimizing for a particular metric, whether this is a score on a certain benchmark or a performance indicator. In order to achieve this, we generally train a single model or an ensemble of models that combined all together with one way we succeed a desirable outcome. While we can generally achieve acceptable performance this way, by being focused on our single task, we ignore information that might help us do even better on the metric we care about. Specifically, this information comes from the training signals of related tasks. By sharing representations between related tasks, we can enable our model to generalize better on our original task. This approach is called Multi-Task Learning (MTL).

Advantages of Multitask Learning

MTL effectively increases the sample size that we are using for training our model. As all tasks are at least somewhat noisy, when training a model on some task A, our aim is to learn a good representation for task A that ideally ignores the data-dependent noise and generalizes well. As different tasks have different noise patterns, a model that learns two tasks simultaneously is able to learn a more general representation. Learning just task A bears the risk of overfitting to task A, while learning A and B jointly enables the model to obtain a better representation F through averaging the noise patterns.

- **Attentions Focusing** : If a task is very noisy or data is limited and high-dimensional, it can be difficult for a model to differentiate between relevant and irrelevant features. MTL can help the model focus its attention on those features that actually matter as other tasks will provide additional evidence for the relevance or irrelevance of those features.
- **Eavesdropping** : Some features G are easy to learn for some task B, while being difficult to learn for another task A. This might either be because A interacts with the features in a more complex way or because other features are impeding the model's ability to learn G. Through MTL, we can allow the model to eavesdrop, i.e. learn G through task B. The easiest way to do this is through hints [11], i.e. directly training the model to predict the most important features.
- **Representation bias**: MTL biases the model to prefer representations that other tasks also prefer. This will also help the model to generalize to new tasks in the future as a hypothesis space that performs well for a sufficiently large number of training tasks will also perform well for learning novel tasks as long as they are from the same environment [15].
- **Regularization** : Finally, MTL acts as a regularizer by introducing an inductive bias. As such, it reduces the risk of overfitting as well as the Rademacher complexity of the model, i.e. its ability to fit random noise.

The Hierarchical Structure of Human Language and Learning

Human Language, either spoken or written, is hierarchical. It is generated from smaller units, can call them language quanta. In spoken language, the correspondent quanta are the phonemes. In written language, the fundamental quanta are the simple characters, called unigrams. Combining the quanta on each case, we take higher level tokens in both forms. If we keep on building finally we will see formed sounds and words and if we going on, whole spoken and written sentences will be formed. And sure, when whole sentences are formed, further information is generated and this refers to the syntactic and the semantic information of language. If we keep on unzooming, we go beyond recognition. And beyond recognition is language understanding.

This hierarchical structure is also observed in the learning process a human learns a language. We begin from the quantum units that form each language. This is the alphabet. We learn both how to write them and how each letter is pronounced. Then we learn how to combine them so as to form written and spoken words. Going even higher in the scale, we learn how to give meaning to words by reading or hearing many words in context. However, most of the times in our early years mainly we have someone to give meaning to our words and label them in order to learn their meaning or their position in sentence.

So we can motivate multi-task learning from the Human Learning process by imitating the way we learn to read. Our task is focused on recognizing the handwritten lines and not understanding the content of them. Moreover, the IAM database was built this way. To go beyond recognition we need have vast amount of Handwritten Documents that will have semantically rich content. Maybe understanding or semantically rich content would help in recognition.

7.2.1 Multitask Learning

In this section, we implement Multitask Models that they are going to be trained jointly with the correspondent task-specific train objective. By sharing representations between related tasks, we can enable our model to generalize better on our original task. This approach is called Multi-Task Learning (MTL)

This approach is inspired by the ASR domain and the paper "Hierarchical Multitask Learning with CTC" from [66]. The author implements a hierarchical architecture as depicted below. The number of target units is considered a hyperparameter and these are extracted with the Byte Pair Encoding Algorithm (BPE), an unsupervised subword segmentation algorithm. For Decoding, he utilizes simple greedy search or shallow fusion with unit-specific LM composed of two unidirectional LSTM layers.

Each BPE step/operation performs a unit merging. For that reason, the number of BPE operations roughly defines the number of units of the target set. For instance, if two targets in our vocabulary are 'HELL' and 'O' and they appear together frequently, by applying a BPE operation we can merge them in a new target unit: 'HELLO'. This operation can be repeated an arbitrary number of times until converging to a target set that contains all words of the vocabulary.

Then he makes a comparison between Hierarchical Multitask Architecture and Single Task Architecture. He concludes that at lower level the model predicts few and general targets, such as characters, and at the higher layers the model predicts highly specific and abstract targets such as words. The success of the model, is due to the fact that the model has learned better feature representations and thus it can recognize better each target.

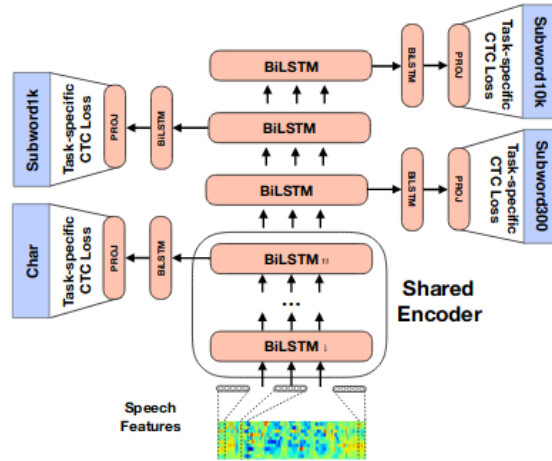


Figure 7.8: Hierarchical multitask learning with CTC for ASR [66]

Except from the HMTL, author also refers to the Block Multitask Architecture.

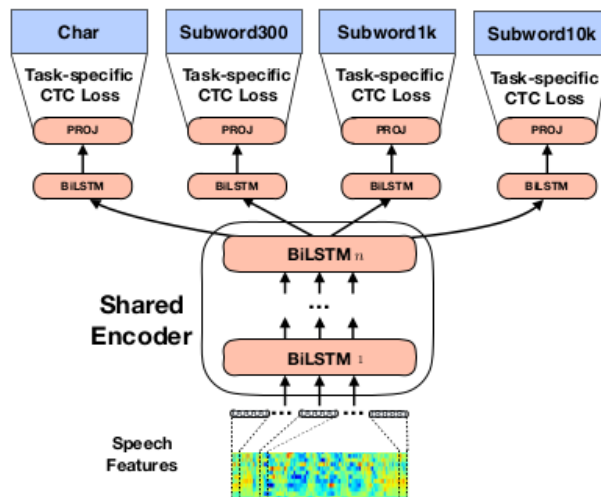


Figure 7.9: Block multitask learning with CTC for ASR [66]

We are going to adapt the Multitask Approach for Handwritten Recognition since to the best of our knowledge we do not know any published work that investigates the Multitask framework in the HTR domain. We are optimistic that this approach will bring gain to the recognition rate. It encapsulates the idea of curriculum learning and also mimics the way that we, humans, learn to read. In our case, it is critical to study till which specific extend this hierarchicity can be applied since data play a crucial role when it comes to the extending further the scale.

We experiment with both HMTL and BMTL Architectures. At first we use as target units in the first level the unigrams and in the second level all the bigrams formed by the 26 unigrams.

Our **Block Multitask Architecture** is described above :

Let E be a sequence of features extracted from the CNN Encoder and the Max Pooling Operation as described in Baseline Model in section ?? and X the input volume.

$$\begin{aligned}
 E &= \text{SharedCNNEncoder}(X) \\
 E_{unigrams} &= \text{BiLSTM}_{unigrams}(E) \\
 E_{bigrams} &= \text{BiLSTM}_{bigrams}(E) \\
 Y_{unigrams} &= \text{Softmax}(\text{Dense}_{unigrams}(E_{unigrams})) \\
 Y_{bigram} &= \text{Softmax}(\text{Dense}_{bigrams}(E_{bigrams}))
 \end{aligned}$$

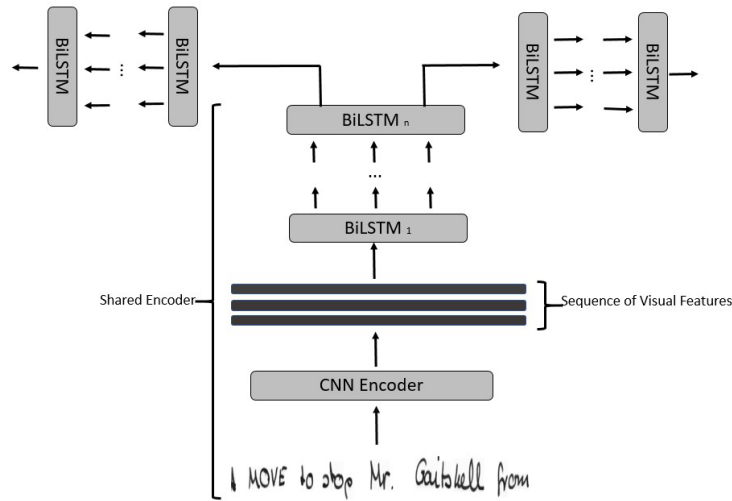


Figure 7.10: Block multitask architecture for HTR

Note: The BiLSTM_{bigram} is actually an extension of $\text{BiLSTM}_{unigram}$. So there is also a shared recurrent encoder between unigrams and bigrams.

Our **Hierarchical multitask** architecture:

$$\begin{aligned}
 E &= \text{SharedCNNEncoder}(X) \\
 E_{unigrams} &= \text{BiLSTM}_{unigrams}(E) \\
 E_{intermediate1} &= \text{BiLSTM}_{intermediate1}(E) \\
 E_{bigrams} &= \text{BiLSTM}_{bigrams}(E_{intermediate1}) \\
 Y_{unigrams} &= \text{Softmax}(\text{Dense}_{unigrams}(E_{unigrams})) \\
 Y_{bigram} &= \text{Softmax}(\text{Dense}_{bigrams}(E_{bigrams}))
 \end{aligned}$$

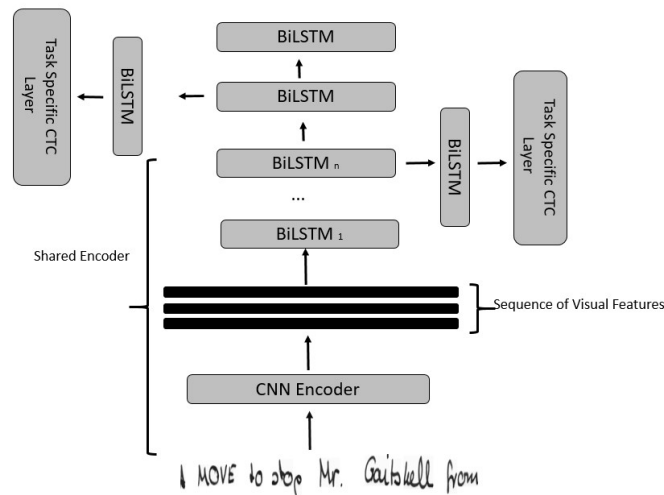
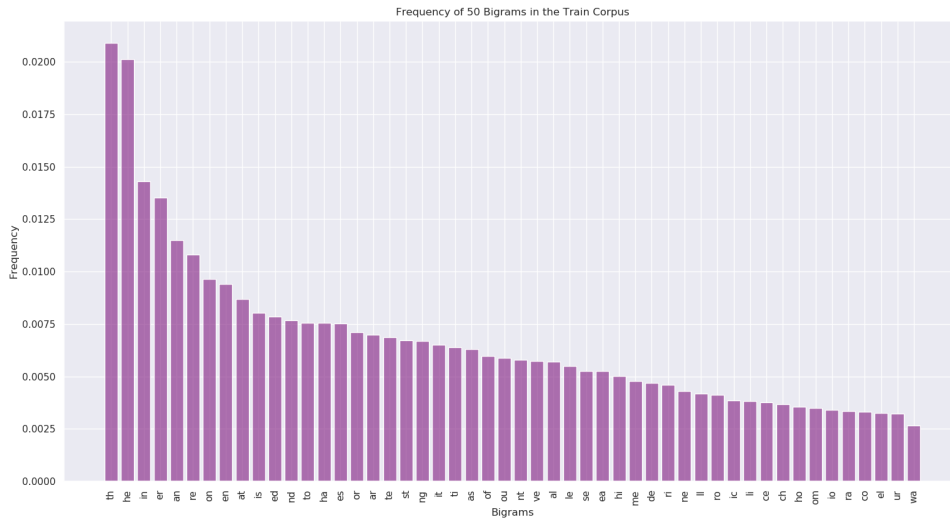


Figure 7.11: Hierarchical multitask architecture for HTR

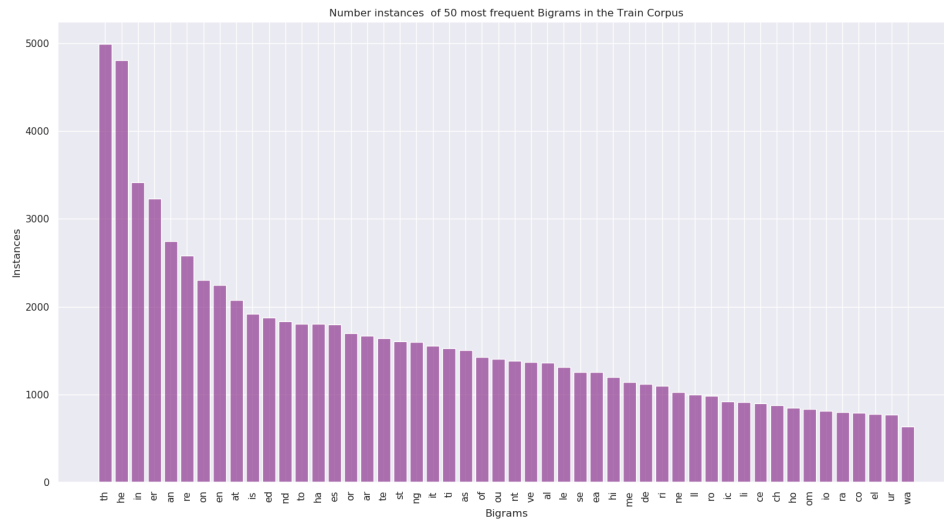
The above Hierarchical and Block Multitask Architectures are defined for Unigrams and Bigrams only. These architectures can also accept higher level target units such as trigrams, fourgrams etc in the ngram level or even subwords and words as target units. Before proceeding in scaling up the models, we need to carefully select the high level target units.

In the HTR problem the data are limited, comparing to the ASR/Language where models are trained for days and the data are hundreds of hours of speaking. In HTR, this is absolutely not the case. The IAM database is created this way, so as to provide the model visual variety of characters. This is decent, since the database was implemented for developing robust visual models. The IAM database contains text that is not semantically rich and the word level tokens are really sparse in the total dataset.

Current Deep Learning Models are trained with Gradient-Descent-like Learning Algorithms and Back Propagation. A limitation in this approach is that the developing model need to 'see' every sample many times in order to learn it sufficiently. From this observation, stems also the need for balanced datasets when it comes to regression and classification tasks. In our tasks, high level tokens do not appear frequently. That means, that we cannot scale up in word level target units when some words appear only 2 or 3 times in the total corpus. For this reason we focus on the subword units and specific trigrams and fourgrams.

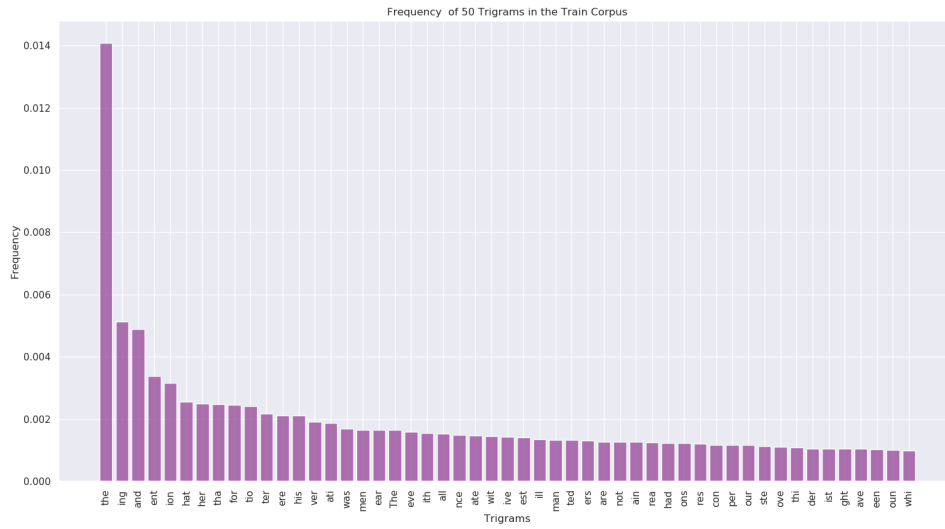


(a) Frequency

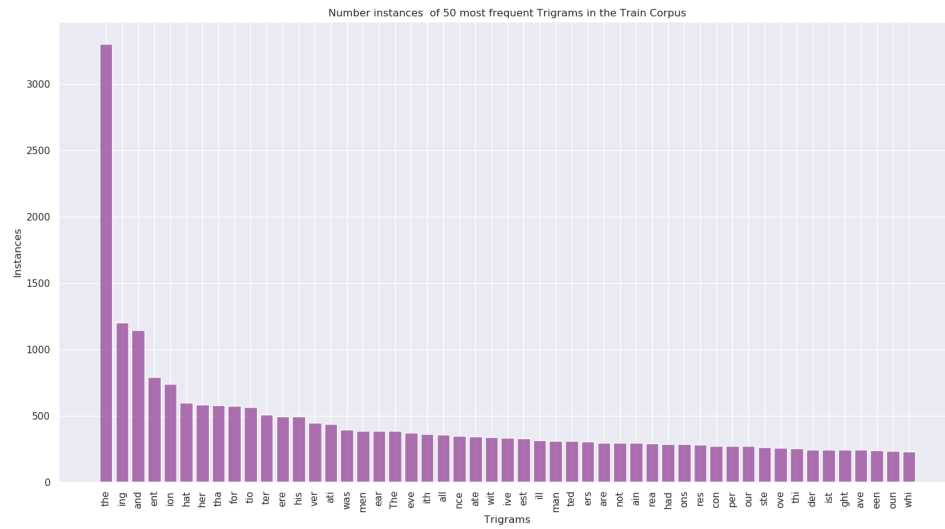


(b) Number of Instances

Figure 7.12: Bigram level tokens

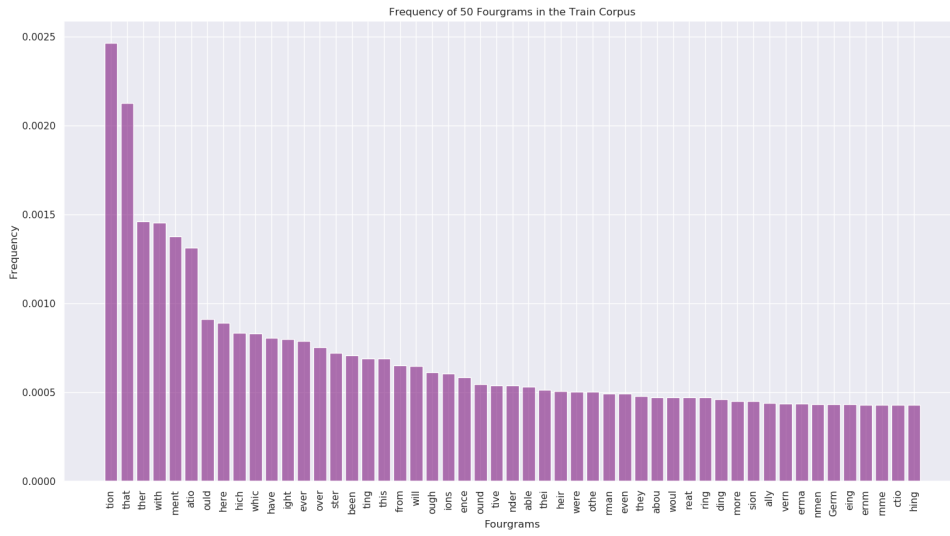


(a) Frequency

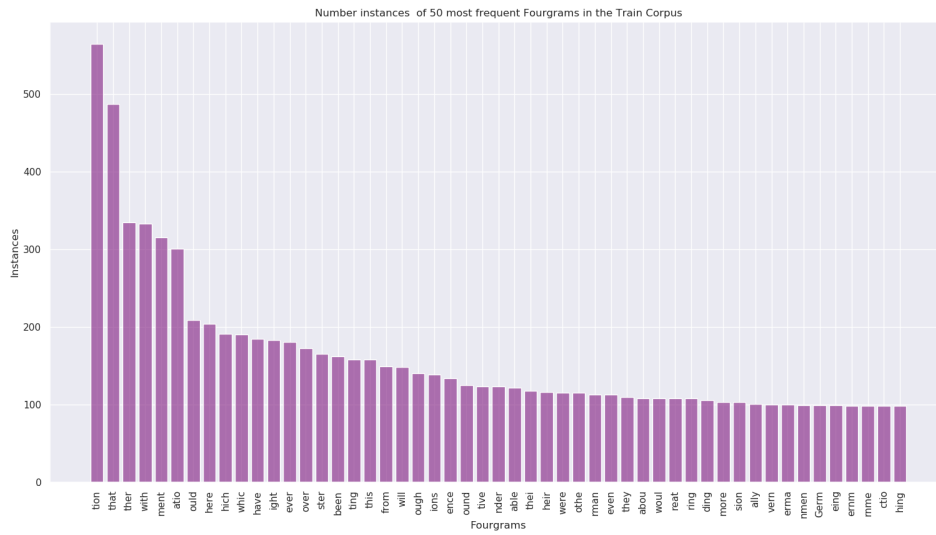


(b) Number of Instances

Figure 7.13: Trigram level tokens

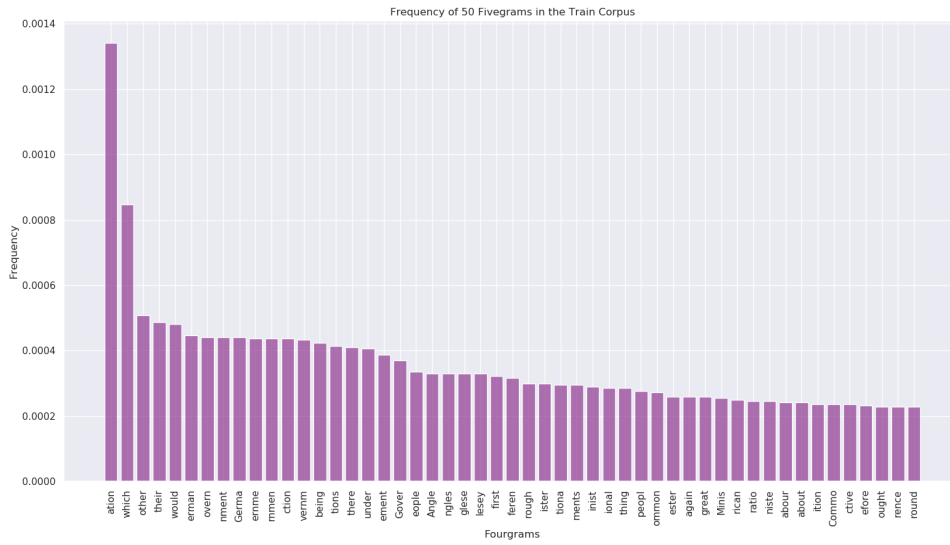


(a) Frequency

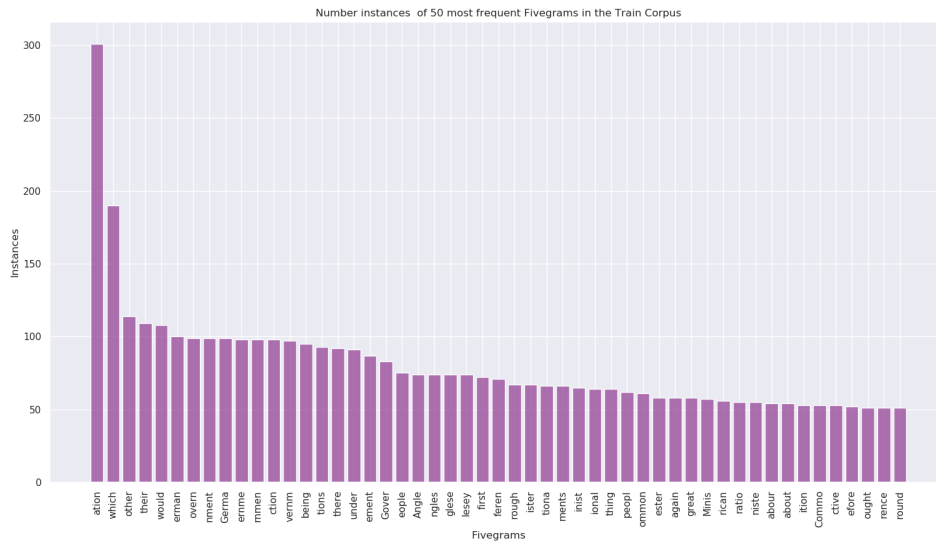


(b) Number of Instances

Figure 7.14: Fourgram level tokens



(a) Frequency



(b) Number of Instances

Figure 7.15: Fivegram level tokens

7.2.2 Experimenting with Multitask Architectures

In this section we performed extended experimentation so as to select the final architecture, selecting network hyperparameters (layers, units etc), training schemes and target units.

Tranning Schemes:

- Sum of CTC Losses backpropagates in every iteration.
- One CTC Loss, either from Unigram Level or from Bigram Level backpropagates every time. The decision is made from a uniform distribution in (0,1) so as both CTC Losses have equal opportunities to backpropagate.

After experimentation we saw that is better for the model's optimization to take into consideration all the tasks at each back propagation and not only one of them. What is more, tasks are not contradictive so as to Also, both tasks that we want to optimize are not contrandictive, which means that we can train them simulatneously and not only one by one.

Target Units :

- Bigrams
- Trigrams
- Fourgrams
- Fivegrams

As it is known as frequency decreases, sparcity increases. Thus when we scale to fivegrams, network does not converges easily since both the network has grown too much and the target units are sparce. Frequency really matters because when scalling up the models does not seem to scale in fivegrams.

One drawback of Hierarchical Multitask Architecture is that it is difficult to scale. As a result, it grows in depth by adding more LSTM Layers for Task Specific Modules and Intermediate Layers that it is difficult to train, it takes a lot more epochs.

After Experimentation we come up to the bellow conclusions:

- Scaling the target units above fourgrams, does not offers an improvement to the network and that is due to the frequency of Fivegram Units. They appear less in the train set.
- We focus only in the 1000 most frequent ngram-tokens in the train corpus. We want the model to learn well the most frequent ngram tokens.
- Intermediate LSTM layers, do not offer improvements in the Model's performance. On the contary, they make slower the convergence during training.

Are task specific LSTM layers really needed?

No, they do not really need since high level target units integrate long term dependences that do not need to be modeled sequentially. At least in the case of Unigram and Bigram Target Units.

Scale	WER Unigram	CER Unigram
Unigrams + Bigrams	0.1772	0.05210
Unigrams + Bigrams + Trigrams	0.1770	0.05372
Unigrams + Bigrams + Trigrams + Fourgrams	0.1758	0.05293

Table 7.3: Hierarchical multitask architecture - greedy decoding

Scale	WER Unigram	CER Unigram
Unigrams + Bigrams	0.1796	0.05280
Unigrams + Bigrams + Trigrams	0.1790	0.05302
Unigrams + Bigrams + Trigrams + Fourgrams	0.1768	0.05183

Table 7.4: Block multiscale learning with 1 task-specific BiLSTM layer - greedy decoding

From the above plot we can deduce that as we increase the scale of target units we get an improvement in the recognition rate. However, the obtained improvement does not provide a satisfactory trade off between the number of parameters and the recognition rate. Thus, instead of scaling up the target units, that may need task specific BiLSTM layers, we focus in the case of Unigrams and Bigrams only by keeping up at the same time the network architecture simple.

Our Architecture is now comprised of the shared CNN + BiLSTM layers with and Linear Task Specific Layer for the Bigrams and Unigrams that projects the BiLSTM outputs in the corresponding classes of unigrams and bigrams. Linear layers are selected so as not to further put computational and memory burden in the network.

$$E = SharedCNNBiLSTMEncoder(X)$$

$$Y_{unigrams} = Dense_{unigrams}(E)$$

$$Y_{bigrams} = Dense_{bigrams}(E)$$

After training the network with the identical configuration of training epochs, learning rates and optimizers we obtained the bellow results:

Scale	WER Unigram	CER Unigram
Unigrams + Bigrams	0.1774	0.05218

Table 7.5: Block multitask learning without task specific LSTM - greedy decoding

After intense experimentation about the set up of our Multitask Architecture, our efforts culminated to an architecture, that although is not significantly greater than the single-task one, succeeds to achieve better performance in the recognition rates by incorporating additional external domain knowledge.

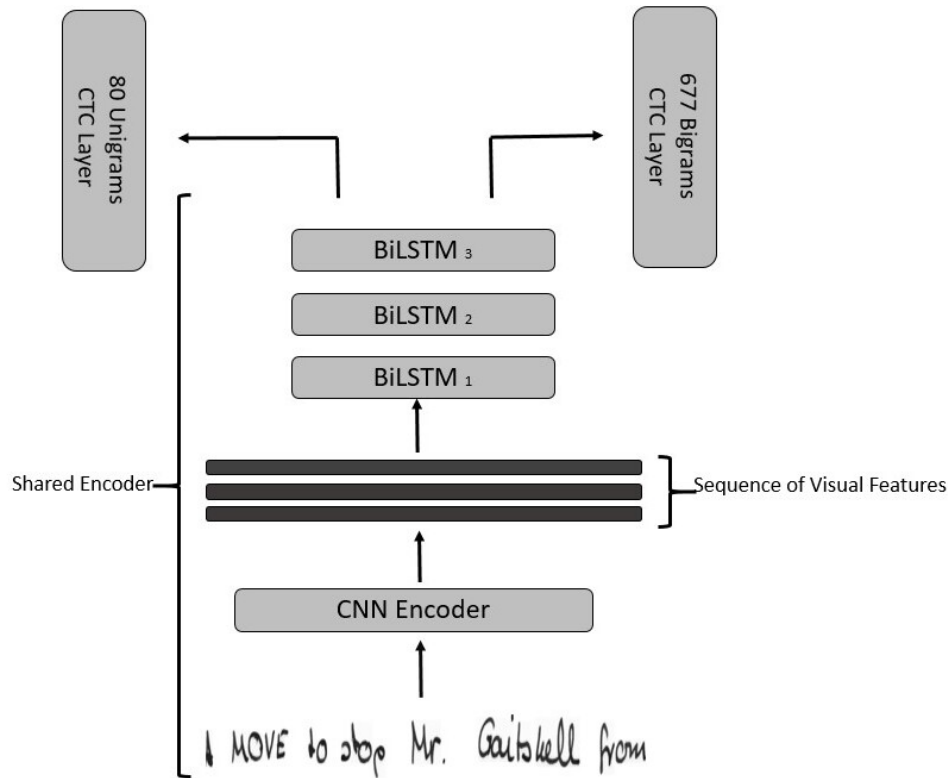


Figure 7.16: The final proposed architecture for multitask learning. Block multitask without task-specific BiLSTM layer

7.3 Single Task Vs Multi Task Architectures for HTR

In this section we compare the Single Task Approach with the Multitask one. Models in both single task and multitask scheme have been trained with the same train configurations. The encoder network shares also the same architecture in both cases. For Decoding, we utilize simple CTC Beam Search Decoding in Unigram Level augmented with a 4-Gram character level Language Model.

Parameter Calculation in Convolutional Neural Networks

We need to think about what a Convolutional Model learns. It learns through backpropagation the filters that perform the convolutions to the input images. So the number of the learnable parameters on each layer is the filter shape on this layer.

- Input Layer provides the input image and has zero learnable parameters
- Convolutional Layer Parameters are : $kernelHeight \cdot kernelWidth + 1) \cdot inputChannels \cdot outputChannels$
- Max Pooling Layers and Activation Function have no parameters

So, our Convolutional Encoder that is both common for STL and MTL Architectures has :

$$((3 \cdot 3 + 1) * 32 * 1) * 2 + ((3 \cdot 3 + 1) * 64 * 32) * 4 + ((3 \cdot 3 + 1) * 128 * 64) * 6 + ((3 \cdot 3 + 1) * 256 * 128) * 2 = 640 + 81920 + 491520 + 655360 = 1.3M \text{ parameters}$$

Parameter Calculation in LSTM Layers

Since we utilize LSTM units, we have 4 gates. So $g = 4$. The hidden size is $h = 256$ and the input size is $i = 256$. The formula that describes the number of parameters in one LSTM Layer is :

$$params = g \cdot (h \cdot (h + i) + h)$$

If the LSTM is Bidirectional the number of parameters are twice as the above. So in our shared LSTM encoder we have 2 BiLSTM Layers. The first one has $2 \cdot 4 \cdot (256 \cdot (256 + 256) + 256) = 1.050M$ parameters. The second layer has input size $256 \cdot 2 = 512$ that means that parameters in the second layer are $2 \cdot 4 \cdot (256 \cdot (256 + 512) + 256) = 1.5M$ parametes.

So the three BiLSTM Layers that compose the shared recurrent decoder have overall $4M$ parameters. The total convolutional and recurrent encoder has parameters $1.3M + 4M = 5.3M$ parameters

A linear layer that projects an input of size I to an output of size O has parameters $I \cdot (O + 1)$. The 1 is for the bias parameter.

So the parameters of task specific linear layer are $512 \cdot (targetUnits + 1)$

Each Task Specific Layer that is composed from 1 BiLSTM layer and one Linear Layer has parameters:

- Unigram Targets : $2.5M + 512 \cdot 81 = 2.5M$
- Bigram Target : $2.5M + 512 \cdot 676 = 2.55M$
- Trigram Target $2.5M + 512 \cdot 1000 = 2.6M$
- Fourgram Targets : $2.5M + 512 \cdot 1000 = 2.6M$

Layers	3
Epochs	70
Early Stopping	25
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	3 BiLSTM layers
batch size	8

Table 7.6: Configuration of Multitask Experiments

For implementing Early Stopping, we demanded that at every test that happens every 5 Epochs, if the model does not gives WER and CER both less that the already minimum WER and CER, the persistence of the model will increase. If it becomes equal to early stopping the model will stop the training process. Because this early stopping criterion is strict, we give the Model a chance of 25 epochs to find the next best WER and CER.

Under these configurations, both STL and MTL Approaches were trained.

Architecture	WER	CER	parameters M
STL	0.1814	0.0564	5.30
BMTL (UB)	0.1669	0.0515	7.85
BMTL (UBT)	0.1669	0.0532	10.45
BMTL (UBTF)	0.1668	0.0519	13.05

Table 7.7: STL-vs-MTL architectures for HTR in unigram Level - CTC 4-Gram Char LM

Architecture	WER	CER	parameters M
STL	0.1814	0.0564	5.30
Linear BMTL (UB)	0.1672	0.0528	5.68

Table 7.8: STL-vs-MTL architectures for HTR in unigram Level - CTC 4-Gram Char LM

Architecture	WER	CER	parameters M
STL (Ours)	0.1481	0.0460	5.30
Linear BMTL (UB)	0.1392	0.0460	5.68

Table 7.9: STL-vs-MTL architectures for HTR in unigram level - CTC Word LM 4-Gram

7.4 Conclusions

Most current state-of-the art approaches perform Single Task Learning for Handwritten Text Recognition by utilizing only unigram character level target units in the CTC Layer. However, it is undeniable that higher scale target units can also contribute to the learning process, by enriching the neural encoder with higher level information and eventually with better visual features.

Architecture	% WER	% CER	parameters (M)
STL (Ours)	19.10	5.6	5.30
STL [53]	20.20	6.2	9.30
Linear BMTL (UB)	17.74	5.2	5.68

Table 7.10: STL-vs-MTL architectures for HTR in unigram level - greedy decoding

From the above experiments we conclude that our approach holds some advantages:

- Bigger CNN network that is forced to encode the input image. CNNs are easier and faster to train than the LSTMs that have slower convergence and sure are more computationally heavy.
- We implement Map-to-Sequence with Max Pooling over the dimension of columns, so as to take the max out of them. In the Literature, they concat all the columns of the feature maps into one and they burden the first LSTM layer with no reason.
- We integrate domain knowledge in the learning process and we show that the model can learn better when higher level target units. For example, utilizing only unigrams and bigrams we can overcome the state of the art (of 1D LSTM approach) WER/CER with less parameters.
- High level target units incorporate long term dependencies and so in some cases no LSTM is needed, like in the Unigrams and Bigrams case.
- We surpass the problem of the frequency of high level tokens by utilizing in all scales the 1000 most frequent.

Chapter 8

Deep Convolutional CTC Network for Handwriting Recognition

8.1 Introduction

Connectionist Temporal Classification (CTC) is commonly used in monotonical sequence transduction tasks such as Handwritten Text and Automatic Speech Recognition. CTC is a segmentation-free and alignment-free algorithm that allows model training without indicating the position of each target unit either on an image or a speech segment. CTC is utilized on top of a Recurrent Neural Network and most commonly on top of an LSTM. Working in the Handwriting Recognition field, in this Chapter, we make an investigation of CNN encoders for modeling the temporal relationships among feature vectors that represent finally our initial image. In contrast with LSTM, CNNs do not hold memory and cannot model, directly, long-term relationships that sequential tasks demand. Below, we make an investigation in a fully Convolutional Network for Handwriting Recognition with CTC on top. We compare the performance of our CNN based models against typical LSTM models in the terms of training time, decoding time and recognition performance. We find that our CNN-based HTR models, with the use of a word-level language model, can obtain a performance close to the LSTM one. While not, reaching the exact performance, the all-convolutional models are faster to train and to infer.

There have been several recorded attempts that are substituting the LSTMs with fully-Convolutional Architectures mainly for Automatic Speech Recognition [65] [79] [75] [41]. To the best of our knowledge there has been no exploration in the fully-convolutional architectures for HTR. Thus, we seize the opportunity to implement and experiment with such architecture for our case.

8.2 Motivation

In order to present our motivation for building a fully-Convolutional Neural Network as our Optical Model, we underline below some of the drawbacks of LSTM neural nets that burden the training process in general.

The power of Long Short Term Memory Networks is the ability to successfully model long-term dependencies in tasks such as Speech/Handwriting Recognition, Activity Recognition, Image Generation, Language Modeling, Image Captioning, and Sentiment Analysis.

However, when they also hold some crucial disadvantages that cannot be ignored. At first, LSTM suffer from the **vanishing gradient** problem. This problem refers to the difficulty found in training artificial neural networks with gradient-based learning methods and backpropagation. Generally, adding more hidden layers tends to make the network able to learn more complex arbitrary functions, and thus do a better job in predicting future outcomes. Although, when it comes to training such networks early layers, that are significant for a model, tend to be trained slower comparing to a shallower network. This is because, when we perform backpropagation, i.e moving backward in the Network and calculating the gradient of the Loss function with respect to the weights, the gradient tends to get smaller and smaller as we keep on moving backward in the later layers in the hierarchy. Additionally, there is also the **exploding gradient** problem, where the gradient that accumulates can lead up to very large gradients. As a result, the network weights are updated by large gradients and this can lead to an unstable network.

One more disadvantage of Recurrent Neural Network is the fact that hidden units have **temporal dependencies** among them. The calculation of the next hidden unit demands the calculation of the previously hidden unit. Thus, LSTM calculation cannot be parallelized in GPUs. So, training time is much more time-costly. On the contrary, Convolutional Neural Networks can be easily parallelized since the convolutional operations that are performed on different channels are independent and thus can be easily parallelized in CUDA environments.

8.3 Fully-Convolutional Architecture

Before diving into our proposed architecture, we need first to explain why we can still utilize Connectionist Temporal Classification with Convolutional Neural Networks. For answering this question we need to think about what CTC demands as input so as to calculate the probabilities of each path. That is actually a grid of probability distributions over a set of target units per each timestep. LSTM neural network emits these probabilities by softmax the network outputs. An LSTM network, the output is emitted at every time step and depends both in the current input and the previous hidden state.

If we want to avoid the LSTM Modeling we need to find other way to model the temporal relationships between the feature vectors. Feature vectors are actually images of shape $(1, w)$ that are organized in 256 channels. Each of these channels have encoded features from all the width of the image that in some way need to be correlated. As it is known, Convolutional Operation is in fact a cross-correlation since the kernel is not flipped because it is learnable. For this reason we utilize 1-Dimensional Convolution, namely a Convolution where the kernel is of shape $(1, k)$ and thus operates each distinct feature map from the 256.

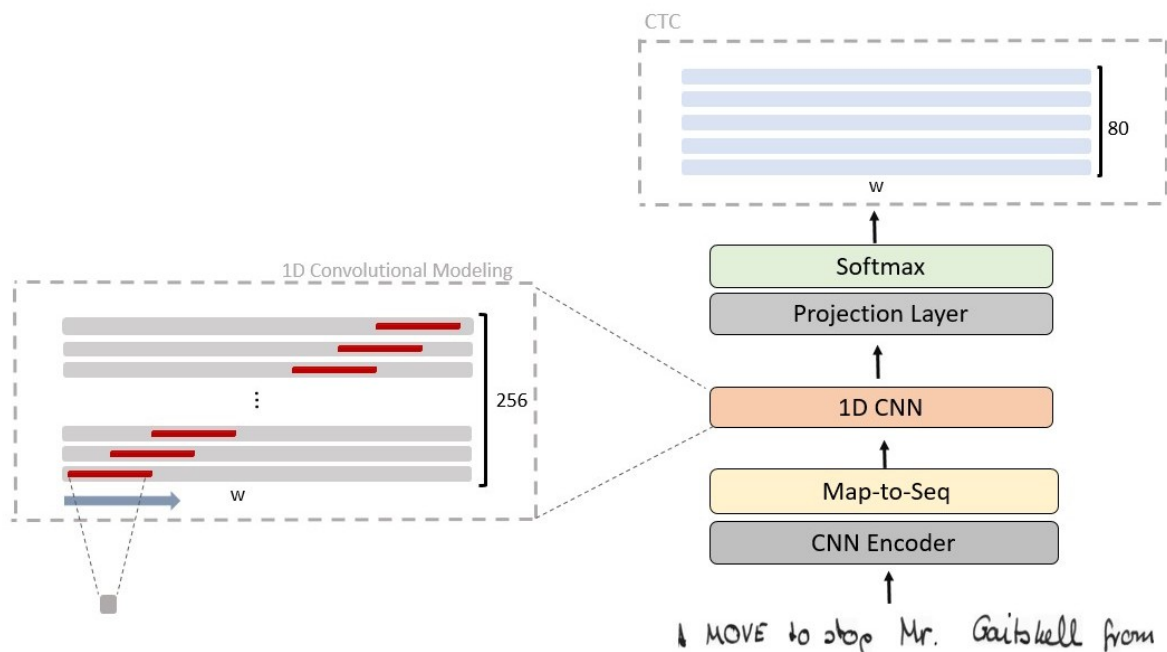


Figure 8.1: Illustration of the CNN-CTC Architecture. The CNN Encoder remains as it is and the only change that is applied in the whole network is the substitution of BiLSTM Layers with the 1-D Convolutions

2 Conv Layers - 32 Kernels 2x2 - ReLU
Max Pooling
4 Conv Layers - 64 Kernels 2x2 - ReLU
Max Pooling
6 Conv Layers - 128 Kernels 2x2 - ReLU
Max Pooling
2 Conv Layers - 256 Kernels 2x2 - ReLU
Max Pooling - Kernel Size 256x1
1-D Convolutional Temporal Modeling

Table 8.1: CNN-CTC End-to-End Architecture

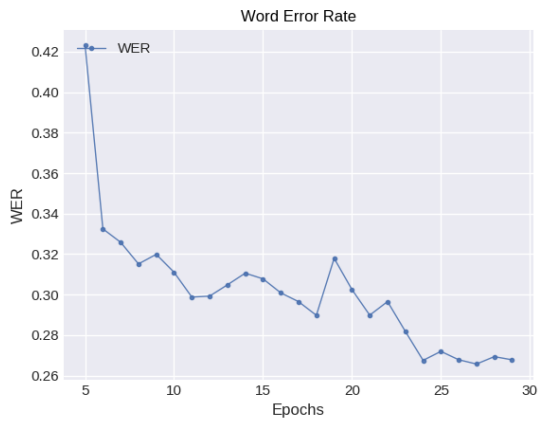
8.3.1 Experiments

We train the CNN-CTC Network with the following configuration.

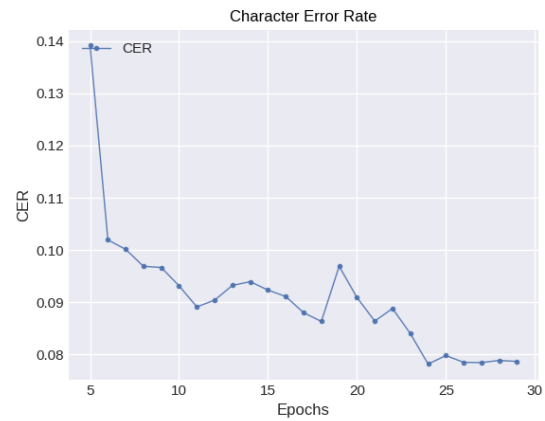
Learning Rate	10^{-3}
Optimizer	RMSProp
Pretrained Model	None
batch size	8

Table 8.2: Train Configuration

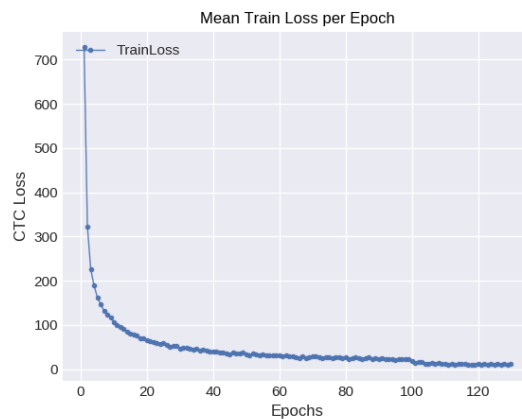
We utilized Early Stopping based on WER/CER on validation set. Per validation interval, every 5 epochs, we demanded both the WER and CER be strictly less than the least noted WER/CER. The persistancy was 5. That means, that if the validation WER/CER performance did not improved for 5 consecutive times the training would stop.



(a) Word Error Rate in Validation



(b) Character Error Rate in Validation



(c) Mean Train Loss per Epoch

Dynamic Data Augmentation Experiments

We perform experiments utilizing dynamic data augmentation techniques as on the . The term dynamic means that every time we extract a batch of images we randomly select whether augmentation is going to be applied or what augmentation is going to be used.

Data Augmentation	WER	CER
None	0.3003	0.0891
Global Affine	0.2670	0.0778
Global Morphological	0.3182	0.0976
Global Affine + Global Morphological	0.3032	0.0911

Table 8.3: Data Augmentation Experiments on Fully Convolutional Neural Network

From the above experiments we conclude that global affine transform gives the most boost in recognition rate comparing with all the others. Thus, we will utilize it for the all of our experiments bellow.

Decoding Network Outputs with External Language Knowledge

The experiments below are done with the utilization of Statistical Character Level and Word-Level Language Models. The Word-level Language Model is the same as the one from the so as to be fair in our comparisons. It has been built with the use of the LOB [39], after the removal of the test lines, and Brown Corpuses. The Word-LM is generated with the KenLM tool [36].

Decoding Algorithm	WER	CER
Greedy	0.2693	0.0794
CTC Beam Search 2 Char-LM	0.2680	0.0785
CTC Beam Search 3 Char-LM	0.2577	0.0753
CTC Beam Search 4 Char-LM	0.2452	0.0732
CTC Beam Search 4-Word LM	0.1620	0.0536

Table 8.4: Results of Decoding with external Language Information - Line Level

8.4 An Ensemble of CNN Models for HTR

We implemented a simple late fusion technique that sums all the feature maps before proceeding them to the time convolutional module. With this way we allow that multiple models may have optimized on different paths, thus combining them all together we take all the possible paths.

Model	WER	CER
cnn-model 1	0.2689	0.0788
cnn-model 2	0.2691	0.0799
cnn-model 3	0.2628	0.0771
cnn-model 4	0.2890	0.0865
cnn-late fusion	0.2443	0.0704

Table 8.5: Performance of Multiple Single CNN Models and the Ensembling

Model	WER	CER
CNN Late Fusion	0.1514	0.0520
CNN-CTC	0.1620	0.0536
CNN-LSTM-CTC	0.1490	0.0460

Table 8.6: Performance of Ensemble Models - Char/Word LM Decoding

Although the ensemble model still does not outperform the baseline (CNN-LSTM-CTC) we observe that the difference in the recognition rate is even more abridged.

8.5 Comparing CNN+CTC with Baseline

Below we make a comparison on Training Time, Inference Time and Number of Parameters. The calculation of training and inference time occurred under the same computational resources and conditions for both models.

Both CNN-CTC and CNN-LSTM-CTC share the same CNN-Encoder that accepts the input image and encodes it into a set of Feature Maps. Thus, both they will have the same number of parameters till then.

The number of parameters of the Convolutional Encoder is:

$$[(3 \cdot 3 + 1) \cdot 1 \cdot 32] \cdot 2 + [(3 \cdot 3 + 1) \cdot 32 \cdot 64] \cdot 4 + [(3 \cdot 3 + 1) \cdot 64 \cdot 128] \cdot 6 + [(3 \cdot 3 + 1) \cdot 128 \cdot 256] \cdot 2 = 1.55M$$

The parameters of parameters of Temporal Convolutional are :

$$[(1 \cdot 5 + 1) \cdot 256 \cdot 256] + (1 \cdot 5 + 1) \cdot 256 \cdot 256 + (1 \cdot 5 + 1) \cdot 256 \cdot 256 = 0.79M$$

Thus, the overall parameters for CNN-CTC are $0.79 + 1.55 = 2.34M$

The formula that describes the number of parameters in one LSTM Layer is :

$$params = g \cdot (h \cdot (h + i) + h)$$

If the LSTM is Bidirectional the number of parameters are twice as the above.

Now in the LSTM layers applies:

- 1st BiLSTM Layer : $2 \cdot 4(256 \cdot (256 + 256) + 256)$
- 2nd BiLSTM Layer : $2 \cdot 4(256 \cdot (256 + 512) + 256)$
- 3rd BiLSTM Layer : $2 \cdot 4(256 \cdot (256 + 512) + 256)$

Thus the overall parameters in the BiSTM Layers are $1.05M + 1.57M + 1.57M = 4.19M$

In the CNN-LSTM-CTC on top of the BiLSTM Layers there is a Linear layer that projects the BiLSTM outputs to the target units. The parameters for this layer are $512 \cdot 80 = 0.04M$. Overall parameters for CNN-LSTM-CTC model are $1.50 + 4.19 + 0.04M = 5.73M$

Model	train time (s)	inference time (s)	Parameters (M)	WER	CER
CNN-LSTM-CTC	370.1	102.1	5.73	0.1481	0.046
CNN-CTC	149.3	22.2	2.34	0.1620	0.0536

Table 8.7: Comparison of CNN-LSTM-CTC and CNN-CTC

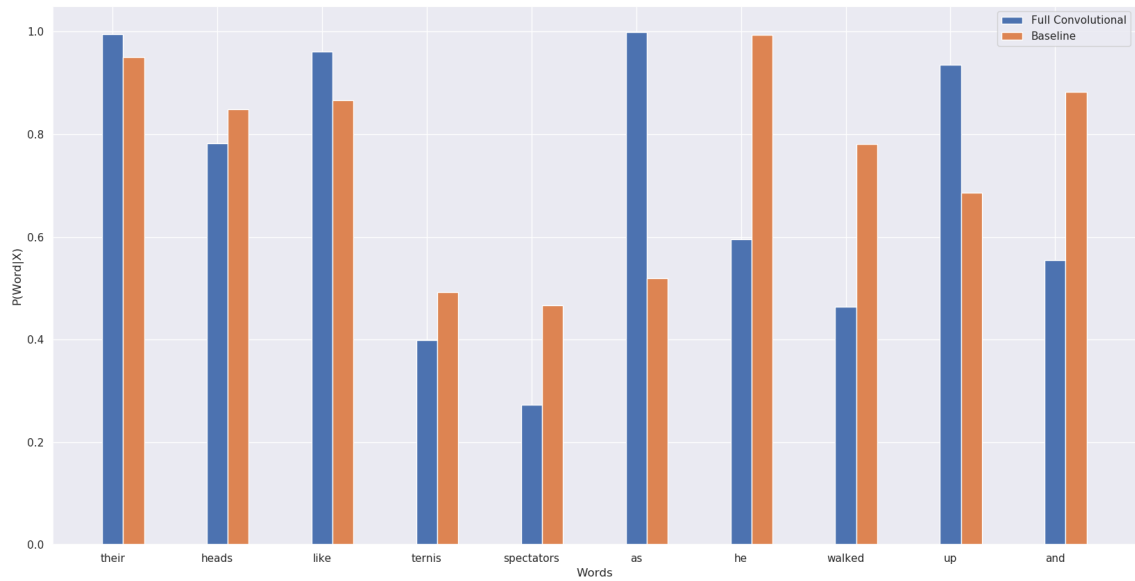


Figure 8.2: Posterior Probabilities that each mode assigns to each word of the sentence

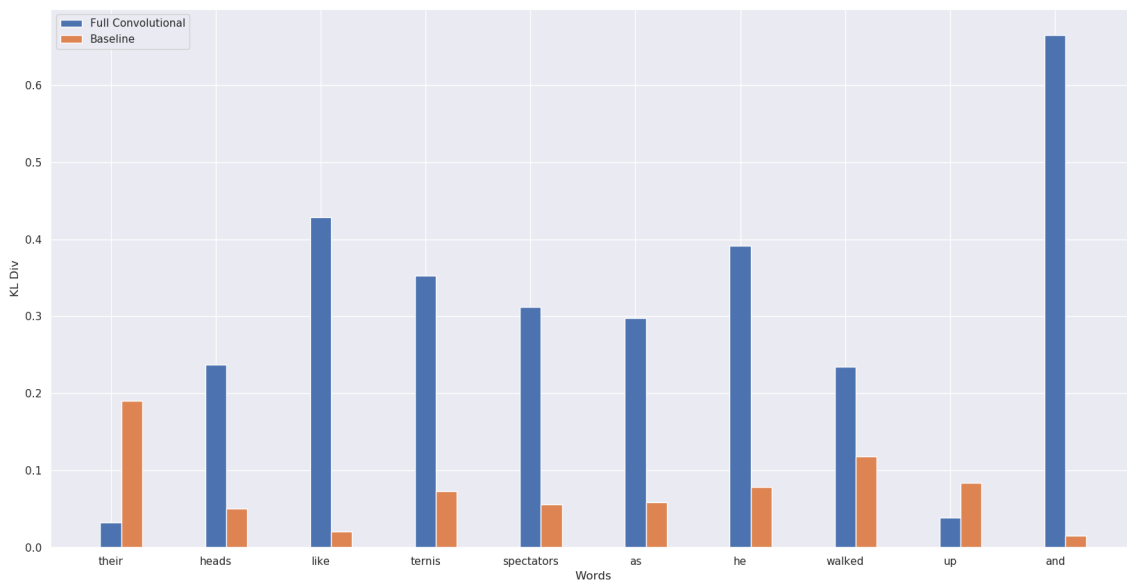


Figure 8.3: KL of words for comparing CNN LSTM models for HTR

An example illustrating that achieving a good recognition accuracy does not necessarily imply having a good representation of the target probability distribution. The two models will achieve a perfect recognition accuracy, but the first model is a better approximation to the reference distribution (has a lower Kullback–Leibler divergence).

Additionally to the above plot, we observed something pretty interesting that is underscored in the table below.

Model	% WER/CER Greedy	% WER/CER WordLM	% WER/CER Improvement
CNN-LSTM-CTC	20.68/6.8	14.81/4.60	28.30 /32.40
MTL	17.74/5.22	13.92/4.63	21.53/11.30
CNN-CTC	26.7/7.77	16.20/5.30	39.33/32.08

Table 8.8: Comparison of CNN-LSTM-CTC and CNN-CTC

As we can observe above, the highest difference on improvement, due to the CTC Beam Search with Word LM, appears in the CNN-CTC model. This observation shows that word LM has more impact on CNN-CTC networks and combined with the Kullback-Leibler plot we may identify why this is the case. As we see in the KL-plot, CNN-CTC has greater KL-divergence in recognizing a correct word. That means that alternative paths are easily to be explored since the models does not concentrates probability in a specific path. The weight in the Language Model is more impactful in that case and this is why we see the such an improvement in the CNN-CTC Model.

8.6 Conclusions

In this Chapter, we made a step towards making the architecture of Handwritten Text Recognition all-convolutional. In particular, we explored that 1-D Convolutions can greatly cooperate with Connectionist Temporal Classification. Our final model is slightly behind the LSTMs performance but it offers **60 %** percent

decrease on the **training time** per epoch, **78 %** percent decrease on the **inference time** and has **60 %** less parameters than the CNN-LSTM-CTC Model. Furthermore, the all-convolutional model opens a way for further exploration of the improvement of the recognition rate. The fact that the purely-Convolutional model is fast and cheap to train and to infer makes it possible to experiment with ensemble techniques. In our case, we experimented with a simple addition of multiple grids of probability distributions and we saw a percentage improvement of **3.5 %** in the **Word Error Rate** and **3 %** in **Character Error Rate**.

The fully-convolutional architecture provides opportunities for incorporating other ideas in the CNN networks such as the self-attention mechanism [77] or even Transformer-based models [72] for the sequence modeling part on top of the CNN Encoder. Thus, one of the future steps towards employing fully Convolutional Architectures for HTR is to experiment and compare the different self-attentions schemes and transformer models for sequence transduction.

Chapter 9

Contributions, Conclusions and Future Work

9.1 Contributions, Conclusions and Future Work

This Thesis has studied extensively the CTC output layer so as to identify whether the augmentation of the output layer with bigrams, except for unigrams, could have the potential to improve the recognition rates. Our initial goal was to examine whether the bigrams could improve the recognition rates by correcting some wrongly classified unigrams. In this context, we implemented two algorithms that were utilizing the bigram information in different ways. The first one utilizes bigram information as an extension probability that scored at each time step the extension of unigrams in the CTC Beam Search Decoding. The second algorithm utilized bigrams as unigrams, namely bigrams were also used for the string extension. After experimentation with both decoding algorithms, we concluded that we had no improvement at all. We found out that the main reason for this was the fact the approximately every bigram was in accordance with its neighboring unigrams. Thus, there was no different information that could correct any wrongly classified unigrams. The second reason is that CTC assigns the whole probability mass into one path and in this way it obstructs the investigation of several alternative paths.

The conclusions from our previous work lead us to operate the bigrams in an utterly different manner. We considered that bigrams and unigrams should exist in different layers and so we lead up to Multitask Learning. Given that such an approach is not been conducted in the sector of Handwritten Text Recognition, we seized the opportunity to experiment on this. After long experiments and further investigation, we concluded to the optimal architecture that combined both unigrams and bigrams. Comparing this multitask architecture with our single task, we deduced that the multitask with a slightly greater number of parameters yield a significant decrease in the Word and Character Error Rate. Comparing our multitask architecture with published research [53] we lead up to the conclusion that with significantly fewer parameters we achieved crucial improvement in the recognition rate. Finally, we conclude that we can leverage the benefits of bigrams in two ways. The first one is from the usage of the external character-level language model in the CTC Decoding process. The second one is from the integration of domain knowledge (language) in the model by forcing it to recognize bigrams in the image.

In the last chapter of the present work, we implement a fully convolutional architecture. Having comprehended the computational burden and the difficulty in the convergence of the LSTMs, we wanted to remove such structures from our model. The fully convolutional architectures are studied in the sector of Automatic Speech Recognition, but to the best of our knowledge, in the sector of Handwritten Text Recognition, such experimentation is not been implemented. At first, we explain why we can still utilize CTC on top of a fully convolutional layer. Temporal dependencies can be modeled with one-dimensional convolutions. The improvement we obtain in memory usage in training and inference speed allows us to experiment with ensembling techniques and specifically bagging. In conclusion, the full convolutional model, in combination with a word-level statistical language model yield recognition rates close to the LSTM ones and is faster both in inference, train and convergence time and of course with significantly fewer parameters.

The development of the fully convolutional opens the way for extended future work in since a fast and small model is an exceptional basis for building up any module in the HTR domain. Except from such general benefit, a specific direction is the study and the integration of the Self-Attention Mechanism in the computer vision domain [77]. Such study opens also the way for the exploration of the newly defined models for sequence transduction, the Transformerms [72] in the Computer Vision Field. The problem of Handwritten Text Recognition is a good start for such implementation.

An additional direction is to further optimize local transformations for dynamic data enhancement. We believe that the reason they do not improve greatly is because the parameters of the transformations in the images are not controlled in some way, despite the studied extent of these parameters. This results in the

image being distorted to the extent that the network makes it difficult to educate. One solution to this would be a dynamic neural network data enhancement approach that will always incorporate transform parameters to optimize the recognition rate.

Finally, an alternative direction in which we have already worked and drawn some conclusions is that of using the BERT [20] model to correct semantic errors made in decoding. We believe that incorporating semantic information can improve the recognition rates of the text. In fact, we have collected examples that confirm this. For the successful implementation of BERT we have to overcome some challenges, such as facing the fact that a pre-trained model BERT has been trained in a closed dictionary that is not compatible with ours. This could be avoided by retraining the BERT model in texts that we have already used to extract statistical models. An alternative approach is to use the architecture of BERT to construct a unified recognition and semantic correction architecture.

Appendix A

Appendix

A.1 Examples of Greedy Decoded Alignments of Unigrams and Bigrams

50 Bigrams

He rose from his breakfast-nook bench

H_e_r_r_o_o_s_s_e_e_f_r_r_o_o_m_m_h_h_i_i_s_s_b_r_r_e_e_e_a_a_k_k_f_f_a_a_s_s_t_t-n_n_o_o_k_k_b_b_e_e_n_n_c_c_h_h
_H_e_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_b_r_r_e_e_e_e_a_a_k_k_f_f_a_a_i_i_t_t-n_n_o_o_s_s_k_k_b_b_e_e_n_n_c_c_h_h

and came into the livingroom , where

a_a_n_n_d_d_c_c_a_a_m_m_e_e_i_i_n_n_t_t_t_o_o_t_t_h_h_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_m_m_w_w_h_h_h_h_e_e_r_r_e_e
_a_a_n_n_d_d_c_c_a_a_m_m_e_e_i_i_n_n_t_t_t_o_o_t_t_h_h_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_i_i_o_o_n_n_o_u_u_n_n_w_w_l_l_e_e_r_r_e_e

Then , abruptly , he drew himself up

T_h_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_h_h_e_e_d_d_r_r_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p
_T_t_l_l_e_e_r_r_e_e_n_n_a_a_b_b_u_u_p_p_t_t_l_l_y_y_l_l_e_e_d_d_e_e_w_w_l_l_i_i_n_n_s_s_e_e_e_l_l_f_f_u_u_p_p

and walked on the very tips of

a_a_n_n_d_d_w_w_a_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f
_a_a_n_n_d_d_w_w_a_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f

his toes . He stretched his arms

h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s
_h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s

over his head and yawned agape ,

o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_h_h_e_e_e_e_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_a_a_p_p_e_e
_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_h_h_e_e_e_e_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_a_a_p_p_e_e

(a) 50 Bigrams

100 Bigrams

He rose from his breakfast-nook bench

H_e_r_r_o_o_s_s_e_e_f_r_r_o_o_m_m_h_h_i_i_s_s_b_r_r_e_e_e_a_a_k_k_f_f_a_a_s_s_t_t-n_n_o_o_k_k_b_b_e_e_n_n_c_c_h_h
_H_e_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_b_r_r_e_e_e_e_a_a_k_k_f_f_a_a_s_s_t_t-n_n_o_o_k_k_b_b_e_e_n_n_c_c_h_h

and came into the livingroom , where

a_a_n_n_d_d_c_c_a_a_m_m_e_e_i_i_n_n_t_t_t_o_o_t_t_h_h_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_m_m_w_w_h_h_h_h_e_e_r_r_e_e
_a_a_n_n_d_d_c_c_a_a_m_m_e_e_i_i_n_n_t_t_t_o_o_t_t_h_h_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_i_i_o_o_n_n_o_u_u_n_n_w_w_l_l_e_e_e_e_s_s_s_e_e

Then , abruptly , he drew himself up

T_T_h_h_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_h_h_e_e_d_d_r_r_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p
_T_t_l_l_e_e_r_r_e_e_n_n_a_a_b_b_u_u_p_p_t_t_l_l_y_y_l_l_e_e_d_d_e_e_w_w_l_l_i_i_n_n_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p

and walked on the very tips of

a_a_n_n_d_d_w_w_a_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f
_a_a_n_n_d_d_w_w_a_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f

his toes . He stretched his arms

h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s
_h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s

over his head and yawned agape ,

o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_h_h_e_e_e_e_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_a_a_p_p_e_e
_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_h_h_e_e_e_e_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_a_a_p_p_e_e

(b) 100 Bigrams

150 Bigrams

He rose from his breakfast-nook bench

H_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_r_r_e_e_e_a_a_k_k_f_f_a_a_s_s_t_t-n_n_o_o_o_o_k_k_b_e_e_n_n_c_c_h_h
H_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_b_r_r_e_e_e_e_a_a_k_k_f_f_a_a_s_s_t_t-n_n_o_o_o_o_s_s_k_k_b_b_e_e_n_n_c_c_h_h

and came into the livingroom , where

a_a_n_n_d_d_c_c_a_a_m_m_m_e_e_i_i_n_n_t_t_o_o_t_t_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_o_m_m_w_w_h_h_e_e_e_r_r_e_e
a_a_n_n_d_d_c_c_a_a_m_m_m_e_e_i_i_n_n_t_t_o_o_t_t_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_o_m_m_w_w_h_h_e_e_e_r_r_e_e

Then , abruptly , he drew himself up

T_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_e_e_d_d_r_r_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p
T_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_e_e_d_d_r_r_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p

and walked on the very tips of

a_a_n_n_d_d_w_w_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f
a_a_n_n_d_d_w_w_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f

his toes . He stretched his arms

h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s
h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s

over his head and yawned agape ,

o_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_e_e_e_e_a_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_a_a_p_p_p_e_e
o_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_e_e_e_e_a_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_a_a_p_p_p_e_e

(c) 150 Bigrams

200 Bigrams

He rose from his breakfast-nook bench

H_H_e_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_b_r_r_e_e_e_e_a_a_k_k_f_f_a_a_s_s_t_t-n_n_o_o_o_o_k_k_b_b_e_e_n_n_c_c_h_h
H_H_e_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_b_r_r_e_e_e_e_a_a_k_k_f_f_a_a_s_s_t_t-n_n_o_o_o_o_s_s_k_k_b_b_e_e_n_n_c_c_h_h

and came into the livingroom , where

a_a_n_n_d_d_c_c_a_a_m_m_m_e_e_i_i_n_n_t_t_o_o_t_t_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_o_m_m_w_w_h_h_e_e_e_r_r_e_e
a_a_n_n_d_d_c_c_a_a_m_m_m_e_e_i_i_n_n_t_t_o_o_t_t_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_o_m_m_w_w_h_h_e_e_e_r_r_e_e

Then , abruptly , he drew himself up

T_T_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_e_e_d_d_r_r_e_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p
T_T_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_e_e_d_d_r_r_e_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p

and walked on the very tips of

a_a_n_n_d_d_w_w_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f
a_a_n_n_d_d_w_w_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f

his toes . He stretched his arms

h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s
h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s

over his head and yawned agape ,

o_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_e_e_e_e_a_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_g_a_a_a_p_p_p_e_e
o_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_e_e_e_e_a_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_g_a_a_a_p_p_p_e_e

(d) 200 Bigrams

676 Bigrams

He rose from his breakfast-nook bench

H_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_b_r_r_e_e_e_e_a_a_k_k_k_k_f_f_a_a_s_s_t_t-n_n_o_o_o_o_k_k_b_b_e_e_n_n_c_c_h_h
H_e_r_r_o_o_s_s_e_e_f_f_r_r_o_o_m_m_h_h_i_i_s_s_b_b_r_r_e_e_e_e_a_a_k_k_k_k_f_f_a_a_s_s_t_t-n_n_o_o_o_o_s_s_k_k_b_b_e_e_n_n_c_c_h_h

and came into the livingroom , where

a_a_n_n_d_d_c_c_a_a_m_m_m_e_e_i_i_n_n_t_t_o_o_t_t_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_o_m_m_w_w_h_h_e_e_e_r_r_e_e
a_a_n_n_d_d_c_c_a_a_m_m_m_e_e_i_i_n_n_t_t_o_o_t_t_h_h_e_e_l_l_i_i_v_v_i_i_n_n_g_g_r_r_o_o_o_o_m_m_w_w_h_h_e_e_e_r_r_e_e

Then , abruptly , he drew himself up

T_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_e_e_d_d_r_r_e_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p
T_h_h_e_e_e_n_n_a_a_b_b_r_r_u_u_p_p_t_t_l_l_y_y_h_h_e_e_d_d_r_r_e_e_e_w_w_h_h_i_i_m_m_s_s_e_e_e_l_l_f_f_u_u_p_p

and walked on the very tips of

a_a_n_n_d_d_w_w_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f
a_a_n_n_d_d_w_w_a_a_l_l_k_k_e_e_d_d_o_o_n_n_t_t_h_h_e_e_v_v_e_e_e_r_r_y_y_t_t_i_i_p_p_s_s_o_o_f_f

his toes . He stretched his arms

h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s
h_h_i_i_s_s_t_t_o_o_e_e_s_s_H_h_e_e_s_s_t_t_r_r_e_e_e_e_t_t_c_c_h_h_h_h_e_e_d_d_h_h_i_i_s_s_a_a_r_r_m_m_s_s

over his head and yawned agape ,

o_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_e_e_e_e_a_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_g_a_a_a_p_p_p_e_e
o_o_v_v_e_e_e_r_r_h_h_i_i_s_s_h_h_e_e_e_e_a_a_a_d_d_a_a_n_n_d_d_y_y_a_a_w_w_n_n_e_e_e_d_d_a_a_g_g_g_a_a_a_p_p_p_e_e

(e) 676 Bigrams

Bibliography

- [1] URL: <https://github.com/parlance/ctcdecode>.
- [2] URL: https://en.wikipedia.org/wiki/Document_layout_analysis.
- [3] URL: http://cdn.iiit.ac.in/cdn/cvit.iiit.ac.in/SSDA/slides/SSDA_Jaipur_BhabatoshChanda.pdf.
- [4] URL: <http://www.cs.cmu.edu/~tom/>.
- [5] URL: <http://scott.fortmann-roe.com/docs/BiasVariance.html>.
- [6] URL: https://en.wikipedia.org/wiki/Softmax_function.
- [7] <http://colah.github.io/posts/2015-09-NN-Types-FP/>.
- [8] URL: <https://distill.pub/2017/ctc/>.
- [9] URL: https://en.wikipedia.org/wiki/Gaussian_blur.
- [10] <https://github.com/baidu-research/warp-ctc>.
- [11] Yaser S. Abu-Mostafa. “Learning from hints in neural networks”. In: *J. Complexity* 6 (1990), pp. 192–198.
- [12] David Aldavert, Marçal Rusiñol, Ricardo Toledo, and Josep Lladós. “Integrating Visual and Textual Cues for Query-by-String Word Spotting”. In: *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition. ICDAR '13*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 511–515. ISBN: 978-0-7695-4999-6. DOI: [10.1109/ICDAR.2013.108](https://doi.org/10.1109/ICDAR.2013.108). URL: <https://doi.org/10.1109/ICDAR.2013.108>.
- [13] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. “Word Spotting and Recognition with Embedded Attributes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), pp. 2552–2566.
- [14] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. “Deep Speech 2: End-to-End Speech Recognition in English and Mandarin”. In: *CoRR* abs/1512.02595 (2015). arXiv: [1512.02595](https://arxiv.org/abs/1512.02595). URL: <http://arxiv.org/abs/1512.02595>.
- [15] Jonathan Baxter. “A Model of Inductive Bias Learning”. In: *J. Artif. Int. Res.* 12.1 (Mar. 2000), pp. 149–198. ISSN: 1076-9757. URL: <http://dl.acm.org/citation.cfm?id=1622248.1622254>.
- [16] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.
- [17] Theodore Bluche and Ronaldo Messina. “Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 646–651. DOI: [10.1109/ICDAR.2017.111](https://doi.org/10.1109/ICDAR.2017.111).

- [18] John S. Bridle. “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition”. In: *Neurocomputing*. Ed. by Françoise Fogelman Soulié and Jeanny Héroult. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 227–236. ISBN: 978-3-642-76153-9.
- [19] *Convolutional Neural Networks for Visual Recognition, Stanford CS231n*. <http://cs231n.github.io/>. Accessed: 2019-05-3.
- [20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: [1810.04805](https://arxiv.org/abs/1810.04805). URL: <http://arxiv.org/abs/1810.04805>.
- [21] George Ekladious, Robert Sabourin, and Eric Granger. “Hybrid writer-independent–writer-dependent offline signature verification system”. In: *Biometrics, IET* 2 (Dec. 2013), pp. 169–181. DOI: [10.1049/iet-bmt.2013.0024](https://doi.org/10.1049/iet-bmt.2013.0024).
- [22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [23] Diana F. Gordon and Marie Desjardins. “Evaluation and Selection of Biases in Machine Learning”. In: *Mach. Learn.* 20.1-2 (July 1995), pp. 5–22. ISSN: 0885-6125. DOI: [10.1007/BF00993472](https://doi.org/10.1007/BF00993472). URL: <https://doi.org/10.1007/BF00993472>.
- [24] Alex Graves. “Supervised sequence labelling with recurrent neural networks”. In: *Studies in Computational Intelligence*. 2008.
- [25] Alex Graves. “Supervised sequence labelling with recurrent neural networks”. PhD thesis. Technical University Munich, 2008. URL: <http://d-nb.info/99115827X>.
- [26] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. “Multi-Dimensional Recurrent Neural Networks”. In: *CoRR* abs/0705.2011 (2007). arXiv: [0705.2011](https://arxiv.org/abs/0705.2011). URL: <http://arxiv.org/abs/0705.2011>.
- [27] Alex Graves, Santiago Fernández, and Faustino Gomez. “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks”. In: *In Proceedings of the International Conference on Machine Learning, ICML 2006*. 2006, pp. 369–376.
- [28] Alex Graves and Navdeep Jaitly. “Towards End-To-End Speech Recognition with Recurrent Neural Networks”. In: *ICML*. 2014.
- [29] Alex Graves and Jürgen Schmidhuber. “Framework phoneme classification with bidirectional LSTM networks”. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*. Vol. 4. 2005, 2047–2052 vol. 4. DOI: [10.1109/IJCNN.2005.1556215](https://doi.org/10.1109/IJCNN.2005.1556215).
- [30] Alex Graves and Jürgen Schmidhuber. “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks”. In: *Proceedings of the 21st International Conference on Neural Information Processing Systems. NIPS’08*. Vancouver, British Columbia, Canada: Curran Associates Inc., 2008, pp. 545–552. ISBN: 978-1-6056-0-949-2. URL: <http://dl.acm.org/citation.cfm?id=2981780.2981848>.
- [31] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”. In: *Proceedings of the 23rd International Conference on Machine Learning. ICML ’06*. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 369–376. ISBN: 1-59593-383-2. DOI: [10.1145/1143844.1143891](https://doi.org/10.1145/1143844.1143891). URL: <http://doi.acm.org/10.1145/1143844.1143891>.
- [32] Luiz G. Hafemann, Robert Sabourin, and Luiz S. Oliveira. “Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks”. In: *CoRR* abs/1705.05787 (2017). arXiv: [1705.05787](https://arxiv.org/abs/1705.05787). URL: <http://arxiv.org/abs/1705.05787>.

- [33] Luiz G. Hafemann, Robert Sabourin, and Luiz S. Oliveira. “Offline Handwritten Signature Verification - Literature Review”. In: *CoRR* abs/1507.07909 (2015). arXiv: [1507.07909](https://arxiv.org/abs/1507.07909). URL: <http://arxiv.org/abs/1507.07909>.
- [34] Muhammad Hanif, Anna Tonazzini, Pasquale Savino, Emanuele Salerno, and Gregory Tsagakatakis. “Document Bleed-Through Removal Using Sparse Image Inpainting”. In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. 2018, pp. 281–286. DOI: [10.1109/DAS.2018.21](https://doi.org/10.1109/DAS.2018.21).
- [35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.
- [36] Kenneth Heafield. “KenLM: Faster and Smaller Language Model Queries”. In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. WMT ’11. Edinburgh, Scotland: Association for Computational Linguistics, 2011, pp. 187–197. ISBN: 978-1-937284-12-1. URL: <http://dl.acm.org/citation.cfm?id=2132960.2132986>.
- [37] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735). URL: <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.
- [38] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). arXiv: [1502.03167](https://arxiv.org/abs/1502.03167). URL: <http://arxiv.org/abs/1502.03167>.
- [39] Stig Johansson. “The LOB corpus of British English texts: Presentation and comments”. In: 1980.
- [40] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014).
- [41] Kalpesh Krishna, Liang Lu, Kevin Gimpel, and Karen Livescu. “A Study of All-Convolutional Encoders for Connectionist Temporal Classification”. In: *CoRR* abs/1710.10398 (2017). arXiv: [1710.10398](https://arxiv.org/abs/1710.10398). URL: <http://arxiv.org/abs/1710.10398>.
- [42] Hairong Liu, Zhenyao Zhu, Xiangang Li, and Sanjeev Satheesh. “Gram-CTC: Automatic Unit Selection and Target Decomposition for Sequence Labelling”. In: *CoRR* abs/1703.00096 (2017). arXiv: [1703.00096](https://arxiv.org/abs/1703.00096). URL: <http://arxiv.org/abs/1703.00096>.
- [43] Petros Maragos and Ronald W. Schafer. “Morphological filters-Part I: Their set-theoretic analysis and relations to linear shift-invariant filters”. In: *IEEE Trans. Acoustics, Speech, and Signal Processing* 35 (1987), pp. 1153–1169.
- [44] Urs-Viktor Marti and Horst Bunke. “The IAM-database: an English sentence database for offline handwriting recognition”. In: *International Journal on Document Analysis and Recognition* 5 (2002), pp. 39–46.
- [45] Warren S. McCulloch and Walter Pitts. “Neurocomputing: Foundations of Research”. In: ed. by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press, 1988. Chap. A Logical Calculus of the Ideas Immanent in Nervous Activity, pp. 15–27. ISBN: 0-262-01097-6. URL: <http://dl.acm.org/citation.cfm?id=65669.104377>.
- [46] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau-Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press, 2009. ISBN: 6130216904, 9786130216900.
- [47] Thomas M. Mitchell. *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077, 9780070428072.

- [48] Bastien Moysset, Theodore Bluche, Maxime Knibbe, Mohamed Faouzi Benzeghiba, Ronaldo Messina, Jerome Louradour, and Chistrophe Kermorvant. “The A2iA Multi-lingual Text Recognition System at the Second Maurdor Evaluation”. In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. 2014, pp. 297–302. DOI: [10.1109/ICFHR.2014.57](https://doi.org/10.1109/ICFHR.2014.57).
- [49] Jon Parker, Ophir Frieder, and Gideon Frieder. “Robust binarization of degraded document images using heuristics”. In: vol. 9021. Dec. 2013, 90210U. DOI: [10.1117/12.2042581](https://doi.org/10.1117/12.2042581).
- [50] Vu Pham, Christopher Kermorvant, and Jérôme Louradour. “Dropout improves Recurrent Neural Networks for Handwriting Recognition”. In: *CoRR* abs/1312.4569 (2013). arXiv: [1312.4569](https://arxiv.org/abs/1312.4569). URL: <http://arxiv.org/abs/1312.4569>.
- [51] Rejean Plamondon and Sargur Srihari. “Online and off-line handwriting recognition: a comprehensive survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.1 (2000), pp. 63–84. DOI: [10.1109/34.824821](https://doi.org/10.1109/34.824821).
- [52] Joan Puigcerver. “A Probabilistic Formulation of Keyword Spotting”. PhD thesis. jpuigcerver@google.com: Polytechnic University of Valencia, Nov. 2018.
- [53] Joan Puigcerver. “Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?” In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 67–72. DOI: [10.1109/ICDAR.2017.20](https://doi.org/10.1109/ICDAR.2017.20).
- [54] Lawrence R. Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286. ISSN: 0018-9219. DOI: [10.1109/5.18626](https://doi.org/10.1109/5.18626).
- [55] Hannes Rantzsch, Haojin Yang, and Christoph Meinel. “Signature Embedding: Writer Independent Offline Signature Verification with Deep Metric Learning”. In: *ISVC*. 2016.
- [56] Tony M. Rath and R. Manmatha. “Word Spotting for Historical Documents”. In: *Int. J. Doc. Anal. Recognit.* 9.2-4 (Apr. 2007), pp. 139–152. ISSN: 1433-2833. DOI: [10.1007/s10032-006-0027-8](https://doi.org/10.1007/s10032-006-0027-8). URL: <http://dx.doi.org/10.1007/s10032-006-0027-8>.
- [57] Arshia Rehman, Saeeda Naz, and Muhammad Muhammad Razzak. “Writer Identification Using Machine Learning Approaches: A Comprehensive Review”. In: *Multimedia Tools Appl.* 78.8 (Apr. 2019), pp. 10889–10931. ISSN: 1380-7501. DOI: [10.1007/s11042-018-6577-1](https://doi.org/10.1007/s11042-018-6577-1). URL: <https://doi.org/10.1007/s11042-018-6577-1>.
- [58] George Retsinas, Giorgos Sfikas, and Basilis Gatos. “Transferable Deep Features for Keyword Spotting”. In: *Proceedings 2* (Jan. 2018), p. 89. DOI: [10.3390/proceedings2020089](https://doi.org/10.3390/proceedings2020089).
- [59] Dominique Rivard, Eric Grange, and Robert Sabourin. “Multi-feature Extraction and Selection in Writer-independent Off-line Signature Verification”. In: *Int. J. Doc. Anal. Recognit.* 16.1 (Mar. 2013), pp. 83–103. ISSN: 1433-2833. DOI: [10.1007/s10032-011-0180-6](https://doi.org/10.1007/s10032-011-0180-6). URL: <http://dx.doi.org/10.1007/s10032-011-0180-6>.
- [60] Frank Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain”. In: *Psychological Review* (1958), pp. 65–386.
- [61] David Rumelhart, Geoffrey Hinton, and Ronald J. Williams. “Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1”. In: ed. by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal Representations by Error Propagation, pp. 318–362. ISBN: 0-262-68053-X. URL: <http://dl.acm.org/citation.cfm?id=104279.104293>.
- [62] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-propagating Errors”. In: *Nature* 323.6088 (1986), pp. 533–536. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0). URL: <http://www.nature.com/articles/323533a0>.

- [63] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. “Efficient Segmentation-free Keyword Spotting in Historical Document Collections”. In: *Pattern Recogn.* 48.2 (Feb. 2015), pp. 545–555. ISSN: 0031-3203. DOI: [10.1016/j.patcog.2014.08.021](https://doi.org/10.1016/j.patcog.2014.08.021). URL: <http://dx.doi.org/10.1016/j.patcog.2014.08.021>.
- [64] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. “Towards query-by-speech handwritten keyword spotting”. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)* (2015), pp. 501–505.
- [65] Tara N. Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E. Dahl, George Saon, Hagen Soltau, Tomás Beran, Aleksandr Y. Aravkin, and Bhuvana Ramabhadran. “Improvements to deep convolutional neural networks for LVCSR”. In: *CoRR* abs/1309.1501 (2013). arXiv: [1309.1501](https://arxiv.org/abs/1309.1501). URL: <http://arxiv.org/abs/1309.1501>.
- [66] Ramon Sanabria and Florian Metz. “Hierarchical Multi Task Learning With CTC”. In: *CoRR* abs/1807.07104 (2018). arXiv: [1807.07104](https://arxiv.org/abs/1807.07104). URL: <http://arxiv.org/abs/1807.07104>.
- [67] Mike Schuster and Kuldeep K. Paliwal. “Bidirectional Recurrent Neural Networks”. In: *Trans. Sig. Proc.* 45.11 (Nov. 1997), pp. 2673–2681. ISSN: 1053-587X. DOI: [10.1109/78.650093](https://doi.org/10.1109/78.650093). URL: <http://dx.doi.org/10.1109/78.650093>.
- [68] Claude Shannon. “Prediction and entropy of printed English”. In: *The Bell System Technical Journal* 30.1 (1951), pp. 50–64. DOI: [10.1002/j.1538-7305.1951.tb01366.x](https://doi.org/10.1002/j.1538-7305.1951.tb01366.x).
- [69] Robin Sibson and G. Stone. “Computation of Thin-plate Splines”. In: *SIAM J. Sci. Stat. Comput.* 12.6 (Sept. 1991), pp. 1304–1313. ISSN: 0196-5204. DOI: [10.1137/0912070](https://doi.org/10.1137/0912070). URL: <http://dx.doi.org/10.1137/0912070>.
- [70] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=2627435.2670313>.
- [71] Sebastian Sudholt and Gernot A. Fink. “PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents”. In: *CoRR* abs/1604.00187 (2016). arXiv: [1604.00187](https://arxiv.org/abs/1604.00187). URL: <http://arxiv.org/abs/1604.00187>.
- [72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: [1706.03762](https://arxiv.org/abs/1706.03762). URL: <http://arxiv.org/abs/1706.03762>.
- [73] Alessandro Vinciarelli. “A survey on off-line Cursive Word Recognition”. In: *Pattern Recognition* 35 (2002), pp. 1433–1446.
- [74] Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. “Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks”. In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, pp. 228–233. DOI: [10.1109/ICFHR.2016.0052](https://doi.org/10.1109/ICFHR.2016.0052).
- [75] Yisen Wang, Xuejiao Deng, Songbai Pu, and Zhiheng Huang. “Residual Convolutional CTC Networks for Automatic Speech Recognition”. In: *CoRR* abs/1702.07793 (2017). arXiv: [1702.07793](https://arxiv.org/abs/1702.07793). URL: <http://arxiv.org/abs/1702.07793>.
- [76] David H. Wolpert. “The Lack of a Priori Distinctions Between Learning Algorithms”. In: *Neural Comput.* 8.7 (Oct. 1996), pp. 1341–1390. ISSN: 0899-7667. DOI: [10.1162/neco.1996.8.7.1341](https://doi.org/10.1162/neco.1996.8.7.1341). URL: <http://dx.doi.org/10.1162/neco.1996.8.7.1341>.
- [77] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. “CBAM: Convolutional Block Attention Module”. In: *CoRR* abs/1807.06521 (2018). arXiv: [1807.06521](https://arxiv.org/abs/1807.06521). URL: <http://arxiv.org/abs/1807.06521>.

- [78] Mustafa Berkay Ylmaz and Berrin Yankolu. “Score Level Fusion of Classifiers in Off-line Signature Verification”. In: *Inf. Fusion* 32:PB (Nov. 2016), pp. 109–119. ISSN: 1566-2535. DOI: [10.1016/j.inffus.2016.02.003](https://doi.org/10.1016/j.inffus.2016.02.003). URL: <https://doi.org/10.1016/j.inffus.2016.02.003>.
- [79] Ying Zhang, Mohammad Pezeshki, Philemon Brakel, Saizheng Zhang, César Laurent, Yoshua Bengio, and Aaron C. Courville. “Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks”. In: *CoRR* abs/1701.02720 (2017). arXiv: [1701.02720](https://arxiv.org/abs/1701.02720). URL: <http://arxiv.org/abs/1701.02720>.