ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ, ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές μηχανικής μάθησης για την αναγνώριση θέσης χειρών σε εικονοσειρές από φορητές και ενσωματωμένες συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Α. Μαρούσης

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ, ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές μηχανικής μάθησης για την αναγνώριση θέσης χειρών σε εικονοσειρές από φορητές και ενσωματωμένες συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Α. Μαρούσης

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή επιτροπή την 8η Νοεμβρίου 2019.

.....	.....	.....
Π. Τσανάκας	Δ. Κουτσούρης	Γ. Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2019

.....
Νικόλαος Α. Μαρούσης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η εργασία πραγματεύεται τον σχεδιασμό και την αξιολόγηση ενός συστήματος αναγνώρισης της τοποθεσίας χεριών σε εικονοσειρές που προέρχονται από φορητές και ενσωματωμένες συσκευές σε πραγματικό χρόνο. Για το σκοπό αυτό γίνεται χρήση εξειδικευμένων μοντέλων όρασης για συσκευές χαμηλής ισχύος, ώστε το σύστημα να έχει μια βιώσιμη εφαρμογή χρήσης σε φορητές και ενσωματωμένες συσκευές. Η μελέτη μας βασίζεται στην χρήση μοντέλων αρχιτεκτονικής MobileNet v1 και v2, τα οποία εκπαιδεύονται σε ένα σύνολο δεδομένων που απεικονίζει την αλληλεπίδραση δύο ατόμων από εγωκεντρική σκοπιά. Η αξιολόγηση του συστήματος γίνεται σε δύο άξονες, με το πρώτο σκέλος να αφορά τις επιδόσεις των μοντέλων κατά την εκπαίδευση και το δεύτερο σκέλος να αφορά τις επιδόσεις χρήσης τους σε φορητές και ενσωματωμένες συσκευές.

Λέξεις Κλειδιά

Μηχανική Μάθηση, Συνελικτικά Νευρωνικά Δίκτυα, Όραση Υπολογιστών, Φορητές συσκευές, Ενσωματωμένες συσκευές, TensorFlow

Abstract

This thesis is concerned with the design and assessment of a hand location detection system on video frames from mobile and embedded devices in real time. To this end, we utilize vision models specialized for low-power devices, in order to create a system with viable use cases in mobile and embedded devices. The models under study utilize the MobileNet v1 and v2 architectures and they are trained on a data set depicting two interacting individuals from an egocentric view. The system assessment is concerned with two main aspects, one being the training performance of the models and the other being their usage benchmarks on mobile and embedded devices.

Key Words

Machine Learning, Convolutional Neural Networks, Computer Vision, Mobile devices, Embedded devices, TensorFlow

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Τσανάκα που μου εμπιστεύτηκε τη εκπόνηση της συγκεκριμένης διπλωματικής και μου παρείχε το απαραίτητο υλικό για την ολοκλήρωσή της.

Θα ήθελα επίσης να ευχαριστήσω τους γονείς μου, Τάσο και Μαρία, για υπεραριθμήσιμο πλήθος πραγμάτων, καθώς και τους φίλους μου, με τους οποίους έχουμε συμπορευθεί σε καλές και δύσκολες στιγμές και έχουν συμβάλει, ο καθένας με τον τρόπο του, στην εξέλιξη της ζωής μου.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός της εργασίας

Το κινητό τηλέφωνο είναι μια από τις σημαντικότερες και ευρέως χρησιμοποιημένες εφευρέσεις της σύγχρονης τεχνολογικής ιστορίας. Μια συσκευή που αρχικά προοριζόταν για εξειδικευμένη χρήση από μικρό τμήμα του πληθυσμού μετατράπηκε σε ευρύ καταναλωτικό αγαθό μετασχηματίζοντας ριζικά την έννοια της επικοινωνίας και συνδεσιμότητας στη σύγχρονη εποχή.

Οι πρόσφατες εξελίξεις στο σχεδιασμό και τη κατασκευή ολοκληρωμένων κυκλωμάτων, αλλά και η σημαντική πρόοδος στο σχεδιασμό φορητών λειτουργικών συστημάτων οδήγησε στην επέκταση των βασικών λειτουργιών του τηλεφώνου, ώστε πέρα από μια απλή συσκευή συνομιλίας και ανταλλαγής γραπτών μηνυμάτων να μετατραπεί σε μια συσκευή γενικής χρήσης, η οποία σταδιακά έγινε υποκατάστατο ενός φορητού υπολογιστή. Χαρακτηριστικότερο παράδειγμα αυτής της εξέλιξης αποτελεί η συμμετοχή των φορητών συσκευών στην κυκλοφορία του παγκόσμιου ιστού, η οποία, διατηρώντας μια σταθερά ανοδική τάση, ξεπερνά πλέον το 50% του συνόλου των συσκευών[Bro19]. Η σταθερή άνοδος της επεξεργαστικής ισχύος των φορητών συσκευών τα τελευταία χρόνια [Ign+18] επιτρέπει το σχεδιασμό φορητών εφαρμογών με τεχνολογίες υψηλού υπολογιστικού κόστους, όπως η όραση υπολογιστών, ανοίγοντας μια ευρεία κατεύθυνση ανάπτυξης καινοτόμων υπηρεσιών και συστημάτων.

Αντίστοιχα, η ανάδειξη του βιομηχανικού κλάδου του Ίντερνετ των Πραγμάτων (Internet of Things) υποδεικνύει σχεδιαστικές τεχνικές συστημάτων μεγάλης κλίμακας, στα οποία διαδραματίζουν σημαντικό ρόλο περιφερειακές ενσωματωμένες συσκευές χαμηλής ισχύος που είναι σε θέση να επικοινωνούν και να μεταφέρουν δεδομένα μεταξύ τους, χωρίς την ανάγκη ανθρώπινης παρέμβασης. Παράλληλα, η κυκλοφορία εξειδικευμένων επεξεργαστών και επιταχυντών εκτέλεσης μηχανικής μάθησης επιτρέπουν σε συσκευές άκρων (edge devices) να ξεπεράσουν το ρόλο της απλής συλλογής και μετάδοσης δεδομέ-

νων, αφού το εξειδικευμένο υλικό επιτρέπει την εκτέλεση υπολογισμών υψηλού φόρτου στη συσκευή, με σκοπό την παραγωγή και μετάδοση πληροφορίας υψηλότερης τάξης.

Συνοψίζοντας, οι παραπάνω τεχνολογικές εξελίξεις μας επιτρέπουν να σχεδιάζουμε εφαρμογές όρασης υπολογιστών που μπορούν να εκτελούνται αυτόνομα σε φορητές και ενσωματωμένες συσκευές, δίχως να υπεισέρχονται οι περιορισμοί που θα απαιτούσε η σύνδεση με ένα κεντρικό σύστημα ανάλυσης, όπως οι απαιτήσεις εύρους ζώνης της σύνδεσης αλλά και η αναπόφευκτη υστέρηση της μετάδοσης της πληροφορίας.

Στο πλαίσιο της εργασίας, θα παρουσιάσουμε μια ολοκληρωμένη μέθοδο σχεδιασμού και αξιολόγησης ενός συστήματος αναγνώρισης θέσης χειρών σε εικονοσειρές από φορητές και ενσωματωμένες συσκευές. Αφού περιγράψουμε τις βασικές σχεδιαστικές επιλογές του συστήματος, θα αξιολογήσουμε την επίδραση των σχεδιαστικών παραμέτρων στην επίδοσή του, με βάση τόσο την ακρίβεια των αποτελεσμάτων που παράγει, όσο και τη δυνατότητα χρήσης του σε φορητές και ενσωματωμένες συσκευές, ως προς τον χώρο που καταλαμβάνει στη συσκευή αλλά και τη χρονική υστέρηση που απαιτεί η λειτουργία του.

Ένα τέτοιο σύστημα θα μπορούσε να χρησιμοποιηθεί αυτόνομα ή ως μέρος ενός ευρύτερου συστήματος. Ενδεικτικά, για την πρώτη περίπτωση, η θέση των χειρών στην οθόνη θα μπορούσε να χρησιμοποιηθεί ως ένα νέο μέσο αλληλεπίδρασης του χρήστη με τη συσκευή, επεκτείνοντας τις υπάρχουσες απτικές και φωνητικές μεθόδους. Στη δεύτερη περίπτωση, το σύστημα θα μπορούσε να χρησιμοποιηθεί για την απομόνωση της θέσης των χειρών σε μια εικονοσειρά, με σκοπό να τροφοδοτηθεί σε ένα δευτερεύον σύστημα να επιτελέσει διερμηνεία νοηματικής γλώσσας.

1.2 Δομή

Η διπλωματική εργασία έχει συνταχθεί με την ακόλουθη δομή:

Κεφάλαιο 2 Παρουσίαση βασικών στοιχείων μηχανικής μάθησης, ώστε να είναι εφικτή η κατανόηση των τεχνικών τμημάτων της εργασίας χωρίς συνεχή αναφορά σε εξωτερικές πηγές.

Κεφάλαιο 3 Περιγραφή της λειτουργίας των συνελικτικών δικτύων και ανάλυση ειδικών αρχιτεκτονικών που θα χρησιμοποιηθούν στη συνέχεια.

Κεφάλαιο 4 Παράθεση μεθόδων μηχανικής μάθησης στο πεδίο της όρασης υπολογιστών, με αναφορά στη τεχνική τους εξέλιξη και την ανάδειξή τους σε μεθόδους αιχμής. Ανάλυση της οικογένειας μοντέλων όρασης

για φορητές συσκευές MobileNet και παρουσίαση τεχνικών βελτιστοποίησης των δικτύων για χρήση σε φορητές και ενσωματωμένες συσκευές.

Κεφάλαιο 5 Παρουσίαση του συνόλου δεδομένων εκπαίδευσης και αξιολόγησης του συστήματος. Καταγραφή των μοντέλων όρασης υπό εξέταση, καθώς και των παραμέτρων του σχεδιασμού και της εκπαίδευσής τους.

Κεφάλαιο 6 Καταγραφή του συνόλου των συσκευών υπό δοκιμή, παρουσίαση και ανάλυση των πειραματικών αποτελεσμάτων.

Κεφάλαιο 2

Στοιχεία Μηχανικής Μάθησης

2.1 Βασικές αρχιτεκτονικές δικτύων

2.1.1 Το perceptron του Rosenblatt

Το απλό perceptron αποτελεί ένα από τα πλέον θεμελιώδη στοιχεία της μηχανικής μάθησης, καθώς αποτέλεσε την πρώτη περιγραφή ενός τεχνητού νευρωνικού δικτύου με τον επιστημονικό φορμαλισμό μιας αναλυτικής μεθόδου[Ros58, σελ. 388], παρέχοντας τις ισχυρές τεχνικές βάσεις για την περαιτέρω μελέτη των νευρωνικών δικτύων[Hay10, σελ. 47].

Πηγή έμπνευσης για το σχεδιασμό του perceptron αποτέλεσε το φυσικό πρότυπο ενός βιολογικού νευρώνα, όπως περιγράφτηκε από τους McCulloch και Pitts[Hay10, σελ. 47]. Συγκεκριμένα, ο σχεδιασμός του perceptron βασίστηκε στις παραδοχές ότι:

1. Η αρχική κατάσταση των νευρικών συστημάτων είναι επί το πλείστον τυχαία.
2. Το νευρικό σύστημα έχει πλαστικότητα, δηλαδή επιδέχεται αλλαγή απόκρισης στις διεγέρσεις.
3. Σε ένα μεγάλο πλήθος διεγέρσεων, παρόμοιες διεγέρσεις ενεργοποιούν τα ίδια σύνολα νευρώνων, τα οποία διαφέρουν για διαφορετικές διεγέρσεις.
4. Η εφαρμογή θετικής ή αρνητικής ενίσχυσης επιδρά στη δημιουργία συναπτικών συνδέσεων.
5. Η ομοιότητα, δηλαδή ο τρόπος που ένα νευρικό σύστημα ταξινομεί όσα αντιλαμβάνεται, αντιπροσωπεύεται από την ενεργοποίηση παρόμοιων νευρώνων από όμοιες διεγέρσεις.

Με βάση τα παραπάνω στοιχεία, ο Rosenblatt κατέληξε σε ένα μοντέλο νευρώνα όπως παρουσιάζεται στο γράφημα 2.1 και έχει τα ακόλουθα χαρακτηριστικά[Hay10, σελ. 10]:

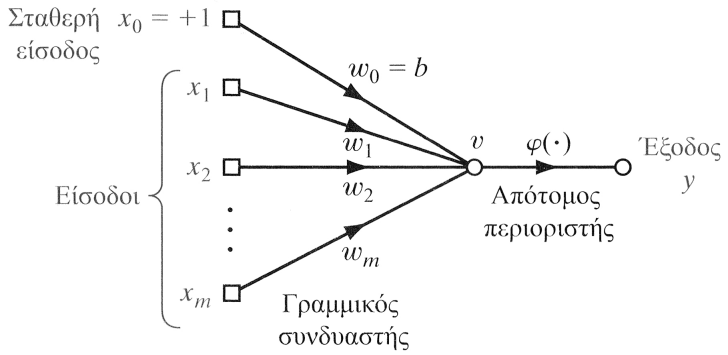
Συνάψεις Κάθε σύναψη j αντιστοιχεί σε ένα σήμα εισόδου x_j και συμμετέχει σε διαφορετικό βαθμό (βάρος w_{kj}) στη διέγερση του νευρώνα k.

Πόλωση Η πόλωση του νευρώνα k είναι μια ξεχωριστή διέγερση b_k που επιδρά ενισχυτικά ή αποτρεπτικά στη συνολική διέγερση του νευρώνα. Για ευκολότερη αναπαράσταση του μοντέλου, η πόλωση λογίζεται ως μια επιπλέον σύναψη διέγερσης $x_0 = +1$ και συναπτικού βάρους $w_{k0} = b_k$.

Αθροιστής Ο αθροιστής φροντίζει για το γραμμικό συνδυασμό των σημάτων εισόδου, σύμφωνα με τα αντίστοιχα συναπτικά βάρη, παράγοντας, σε συνδυασμό με την εφαρμοσμένη πόλωση, την δικτυακή διέγερση του νευρώνα k. Για το νευρώνα k με m το πλήθος συνάψεις η δικτυακή διέγερση έχει τη μορφή:

$$u_k = \sum_{j=0}^{j=m} w_{kj}\, x_j$$

Συνάρτηση ενεργοποίησης Η συνάρτηση ενεργοποίησης (ή συνάρτηση περιορισμού) κανονικοποιεί το εύρος της τιμής εξόδου του νευρώνα σε τιμές μοναδιαίου εύρους. Συγκεκριμένα είδη συναρτήσεων ενεργοποίησης θα εξεταστούν στην ενότητα 2.3.



Σχήμα 2.1: Αρχιτεκτονικό διάγραμμα perceptron. Πηγή: [Hay10, σελ. 50]

Το perceptron είναι σε θέση να ταξινομήσει επιτυχώς διεγέρσεις (δεδομένα εισόδου) που ανήκουν σε ακριβώς δύο γραμμικά διαχωρίσιμες κλάσεις, όπως απεικονίζονται στο γράφημα 2.2, ακολουθώντας μια διαδικασία εκπαίδευσης πεπερασμένου αριθμού βημάτων[Hay10, σελ. 50]. Αν θεωρήσουμε, λαμβάνοντας υπόψιν την πόλωση, κάθε διέγερση ως ένα διάνυσμα εισόδου

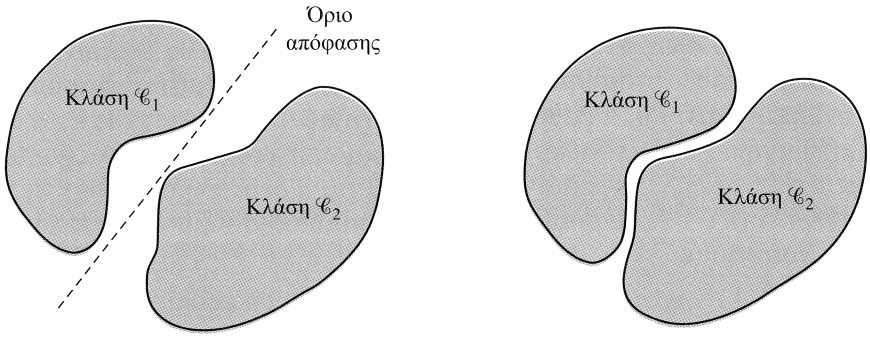
$$\mathbf{x} = \begin{bmatrix} +1 & x_1 & x_2 & \cdots & x_m \end{bmatrix}^\top \tag{2.1.1}$$

και τα συναπτικά βάρη του δικτύου ως ένα διάνυσμα βαρών

$$\mathbf{w} = \begin{bmatrix} b & w_1 & w_2 & \cdots & w_m \end{bmatrix}^\top \tag{2.1.2}$$

τότε το όριο απόφασης για την ταξινόμηση των δύο κλάσεων ορίζεται από το υπερεπίπεδο

$$\sum_{i=0}^{i=m} w_i \, x_i = 0 \quad \text{ή ισοδύναμα} \quad \mathbf{w}^\top \mathbf{x} = 0 \tag{2.1.3}$$



(a) Παράδειγμα γραμμικά διαχωρίσιμων κλάσεων

(b) Παράδειγμα γραμμικά μη-διαχωρίσιμων κλάσεων

Σχήμα 2.2: Αναπαράσταση πιθανών διατάξεων δεδομένων εισόδου δύο κλάσεων. Πηγή: [Hay10, σελ. 51]

Η διαδικασία εκπαίδευσης του μοντέλου γίνεται online, δηλαδή εκτελείται σε διακριτά βήματα, σε καθένα από τα οποία το perceptron τροφοδοτείται με ένα στοιχείο των δεδομένων. Πλέον τα συναπτικά βάρη και το πρότυπο εισόδου εμφανίζουν εξάρτηση από το βήμα της εκπαίδευσης, ενώ η ενημέρωσή τους στο πέρας του βήματος n γίνεται με τον ακόλουθο τρόπο:

$$w(n+1) = \begin{cases} w(n), & \text{αν ταξινομήθηκε επιτυχώς} \\ & \text{το πρότυπο εισόδου } x(n) \\ w(n) - \eta(n)x(n), & \text{σε αντίθετη περίπτωση} \end{cases}$$

όπου η παράμετρος $\eta(n)$ είναι ο ρυθμός μάθησης του perceptron.

2.1.2 Αλγόριθμος Ελαχίστων Μέσων Τετραγώνων

Επόμενη σημαντική εξέλιξη ήταν ο σχεδιασμός του αλγορίθμου εκπαίδευσης Ελαχίστων Μέσων Τετραγώνων, ο οποίος διατηρεί τη διαδικασία του γραμμικού συνδυασμού διεγέρσεων του perceptron, καθώς και την παραγωγή μιας

μοναδικής εξόδου $y(i)$. Επιπλέον, ο αλγόριθμος παρέχει σημαντικά πλεονεκτήματα ως προς το υπολογιστικό κόστος, την ευκολία αποτύπωσης σε κώδικα αλλά και την αντοχή σε εξωτερικές διαταραχές[Hay10, σελ. 92].

Παρόμοια με πριν, τα δεδομένα εκπαίδευσης περιλαμβάνουν ένα σύνολο προτύπων εισόδου

$$\mathbf{x}(i) = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}^\top, \quad i = 1, 2, \cdots, n \tag{2.1.4}$$

καθώς και τις αντίστοιχες επιθυμητές τιμές εξόδου για κάθε πρότυπο

$$d(i), \quad i = 1, 2, \cdots, n \tag{2.1.5}$$

Ξεκινώντας από μια τυχαία αρχικοποίηση των συναπτικών βαρών, σε καθε βήμα i ο αλγόριθμος μάθησης τροφοδοτεί το δίκτυο με το πρότυπο $\mathbf{x}(i)$, υπολογίζει την απόκριση του συστήματος

$$y(i) = \mathbf{x}^\top(i)\mathbf{w}(i) \tag{2.1.6}$$

και αξιολογεί την επίδοσή του βάσει του σήματος σφάλματος

$$e(i) = d(i) - y(i) = d(i) - \mathbf{x}^\top(i)\mathbf{w}(i) \tag{2.1.7}$$

Καθώς ο αλγόριθμος αποσκοπεί στην επίτευξη βέλτιστου σχεδιασμού του δικτύου, άρα κατ' επέκταση βέλτιστης επιλογής συναπτικών βαρών, είναι αναγκαίος ο καθορισμός μιας συνάρτησης των συναπτικών βαρών, που θα τεθεί προς ελαχιστοποίηση κατά τη διάρκεια της εκπαίδευσης. Στην περίπτωση του αλγόριθμου Ελαχίστων Μέσων Τετραγώνων επιδιώκουμε την ελαχιστοποίηση της στιγμιαίας τιμής της ακόλουθης συνάρτησης κόστους:

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}e^2(i) \tag{2.1.8}$$

Με διαφόριση της συνάρτησης ως προς το διάνυσμα συναπτικών βαρών $\mathbf{w}$ προκύπτει η σχέση:

$$\frac{\partial \mathcal{E}(\mathbf{w})}{\partial \mathbf{w}} = e(i)\frac{\partial e(i)}{\partial \mathbf{w}} \overset{(2.1.7)}{=\!=\!=} -\mathbf{x}(i)e(i) \tag{2.1.9}$$

Πλέον η ενημέρωση των συναπτικών βαρών στο πέρας του βήματος i πραγματοποιείται βάσει της σχέσης:

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \eta\mathbf{x}(i)e(i) \tag{2.1.10}$$

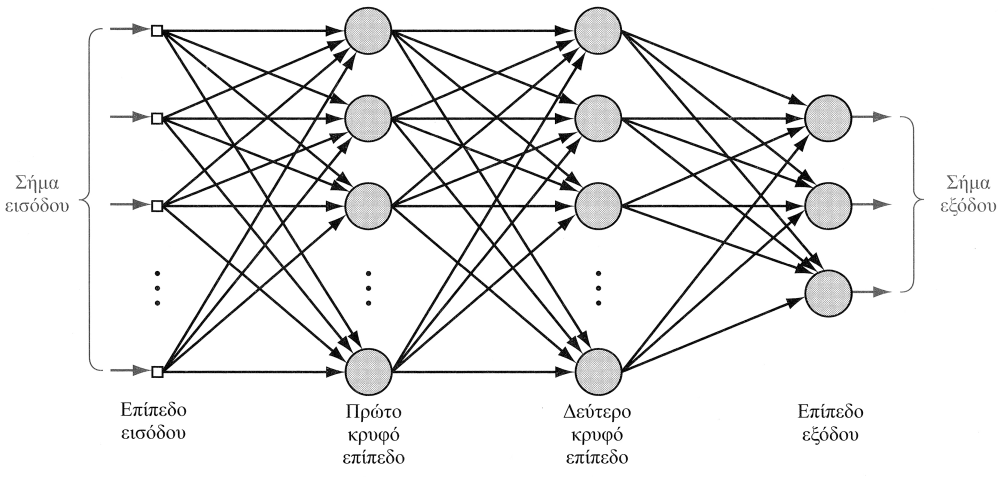
όπου η παράμετρος η είναι ο ρυθμός μάθησης του αλγόριθμου Ελαχίστων Μέσων Τετραγώνων.

2.1.3 Πολυεπίπεδα perceptron

Καθώς οι αρχιτεκτονικές του perceptron και του αλγόριθμου Ελαχίστων Μέσων Τετραγώνων επιλύουν ένα σχετικά μικρό υποσύνολο των προβλημάτων της μηχανικής μάθησης, η επίλυση σύνθετων προβλημάτων απαιτεί τη συνδυαστική χρήση πολλαπλών νευρώνων, κατασκευάζοντας ένα πολυεπίπεδο νευρωνικό δίκτυο.

Όπως αναφέρει ο Haykin[Hay10, σελ. 123], τα βασικά χαρακτηριστικά ενός πολυεπίπεδου perceptron, εικονιζόμενου στο γράφημα 2.3, είναι:

1. Κάθε επιμέρους νευρώνας έχει μια διαφορίσιμη και μη-γραμμική συνάρτηση ενεργοποίησης.
2. Τουλάχιστον ένα επίπεδο νευρώνων παραμένει κρυφό, δηλαδή παρεμβάλλεται μεταξύ των επιπέδων εισόδου και εξόδου.
3. Υπάρχει υψηλή διασυνδεσιμότητα στο δίκτυο, δηλαδή οι νευρώνες μεταξύ δύο διαδοχικών επιπέδων είναι πλήρως συνδεδεμένοι, στο βαθμό που υποδεικνύουν τα αντίστοιχα συναπτικά βάρη.



Σχήμα 2.3: Αρχιτεκτονικό διάγραμμα ενός πολυεπίπεδου perceptron. Πηγή: [Hay10, σελ.]

Γίνεται σαφές ότι η πολυπλοκότητα των δικτύων μιας τέτοιας αρχιτεκτονικής είναι σημαντικά υψηλότερη σε σχέση με τα απλά δίκτυα ενός νευρώνα που μελετήθηκαν στις προηγούμενες ενότητες. Η επίδραση του μεγέθους των παραγόμενων δικτύων είναι ιδιαίτερα εμφανής στη διαδικασία της εκπαίδευσης, καθώς κατά τη διάρκειά της θα πρέπει να ανατεθεί σε συγκεκριμένα τμήματα του δικτύου η αναπαράσταση γνώσης των εισαγόμενων προτύπων εκπαίδευσης. Επιπλέον κατά την αξιολόγηση της επίδοσης του δικτύου θα πρέπει να

γίνει ένας διεξοδικός υπολογισμός της συμμετοχής κάθε νευρώνα στο παραγόμενο αποτέλεσμα, έτσι ώστε να υπάρξει και η ανάλογη αναπροσαρμογή των συναπτικών του βαρών.

2.2 Βελτιστοποίηση βαθμιαίας καθόδου

Όπως είδαμε στον αλγόριθμο Ελαχίστων Τετραγώνων, η διαδικασία μάθησης απαιτεί τον υπολογισμό μιας συνάρτησης κόστους $\mathcal{E}$ προς βελτιστοποίηση, η οποία στη συνέχεια χρησιμοποιείται για την αναπροσαρμογή των βαρών. Στην τυπική εκτέλεση του αλγορίθμου Βαθμιαίας Καθόδου, η αναπροσαρμογή κάθε βάρους υπολογίζεται βάσει του τύπου:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \nabla_{\mathbf{w}} \mathcal{E}(n) \tag{2.2.1}$$

2.2.1 Είδη μάθησης

Κατά την εκπαίδευση ενός πολυεπίπεδου νευρωνικού δικτύου με δείγμα εκπαίδευσης μεγέθους :

$$\mathcal{T} = \left\{\mathbf{x}(n), \mathbf{d}(n)\right\}_{n=1}^{N} \tag{2.2.2}$$

διακρίνονται οι εξής περιπτώσεις για τη διαδικασία μάθησης, ανάλογα με τον τρόπο επεξεργασίας του δείγματος εκπαίδευσης.[Rud17]

Μαζική μάθηση Κατά την εκτέλεση μαζικής μάθησης η συνάρτηση κόστους υπολογίζεται στο πέρας της επεξεργασίας όλων των παραδειγμάτων του δείγματος εκπαίδευσης, ως ο μέσος όρος των σφαλμάτων των επιμέρους παραδειγμάτων. Κατά συνέπεια, η αναπροσαρμογή των βαρών γίνεται μια φορά, στο τελευταίο στάδιο κάθε εποχής εκπαίδευσης και, μετά το πέρας της, γίνεται τυχαία ανακατανομή των παραδειγμάτων του δείγματος εκπαίδευσης. Τα βασικά πλεονεκτήματα της μαζικής μάθησης συναντώνται στην επακριβή εκτίμηση του διανύσματος κλίσης, που συνεισφέρει στην επιτυχή σύγκλιση της μεθόδου εκπαίδευσης, αλλά και τη δυνατότητα παράλληλης εκτέλεσης της διαδικασίας μάθησης, εφόσον το επιτρέπουν οι περιορισμοί μνήμης του συστήματος. Η συγκεκριμένη μέθοδος αποτελεί την τυπική Βαθμιαία Κάθοδο (Gradient Descent).

On-line μάθηση Κατά την εκτέλεση on-line μάθησης η συνάρτηση κόστους και η αναπροσαρμογή των βαρών υπολογίζονται ξεχωριστά για κάθε

επιμέρους πρότυπο του δείγματος εκπαίδευσης. Μετά το πέρας κάθε εποχής, γίνεται τυχαία ανακατανομή των παραδειγμάτων του δείγματος εκπαίδευσης. Καθώς τα παραδείγματα του δείγματος εκπαίδευσης παρουσιάζονται μεμονωμένα και με τυχαία σειρά, ο αλγόριθμος εκτελεί στοχαστική αναζήτηση στον χώρο των βαρών, άρα έχει μικρότερες πιθανότητες να εγκλωβιστεί σε τοπικά ελάχιστα. Για αυτό το λόγο, η μέθοδος καλείται Στοχαστική Βαθμιαία Κάθοδος (Stochastic Gradient Descent). Επιπλέον, παρουσιάζει συγκριτικά μικρότερη δυσκολία υλοποίησης, σε σχέση με τη μαζική μάθηση.

Μάθηση σε παρτίδες Η μάθηση σε παρτίδες αποτελεί μια χρυσή τομή μεταξύ των δύο παραπάνω διαδικασιών, υπολογίζοντας τη συνάρτηση κόστους και εκτελώντας αναπροσαρμογή των βαρών μετά την επεξεργασία ενός υποσυνόλου $k << N$ παραδειγμάτων του δείγματος εκπαίδευσης. Με αυτό τον τρόπο, επιτρέπει την εκτίμηση του διανύσματος κλίσης με αρκετή ακρίβεια, αλλά φυσικά μικρότερη από την περίπτωση της μαζικής μάθησης, αποκτώντας καλές ιδιότητες σύγκλισης, ενώ το μικρότερο μέγεθος του συνόλου παραδειγμάτων προς επεξεργασία επιτρέπει την αποδοτικότερη χρήση του διαθέσιμου υλικού. Σημειώνουμε επίσης ότι η χρήση τυχαίων υποσυνόλων του δείγματος εκπαίδευσης δίνει ένα χαρακτήρα στοχαστικής αναζήτησης στον αλγόριθμο, μειώνοντας την πιθανότητα εγκλωβισμού σε τοπικά ελάχιστα.

2.2.2 Αλγόριθμοι βελτιστοποίησης

Βελτιστοποίηση με ορμή Ο αλγόριθμος Στοχαστικής Βαθμιαίας Καθόδου συναντά δυσκολίες στο χειρισμό τοπολογιών της συνάρτησης κόστους που παραπέμπουν σε φαράγγια, δηλαδή παρουσιάζουν μεγάλη διακύμανση ως προς την κλίση, ανάλογα με τη φορά κατεύθυνσης. Για αυτό το σκοπό, η αναπροσαρμογή των βαρών τροποποιείται κατάλληλα, με τη χρήση ενός όρου ορμής $\mathbf{u}(n)$, που περιορίζει τις πιθανές ταλαντώσεις ανάμεσα στα τοιχώματα του φαραγγιού. Η επίδραση της ορμής καθορίζεται βάσει της παραμέτρου γ και ορίζεται ως:

$$\mathbf{u}(n) = \gamma\,\mathbf{u}(n-1) + \eta\nabla_{\mathbf{w}}\mathcal{E}(n) \tag{2.2.3}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mathbf{u}(n) \tag{2.2.4}$$

Adagrad Ο αλγόριθμος Adagrad προσαρμόζει το ρυθμό μάθησης ανάλογα με τη συχνότητα εμφάνισης ενός συναπτικού βάρους, προτιμώντας τη συντηρητική αναπροσαρμογή των συχνότερα εμφανιζόμενων βαρών και την ενισχυμένη αναπροσαρμογή των αραιότερα εμφανιζόμενων βαρών. Πλέον λογίζουμε μεμονωμένα την κλίση κάθε βάρους $w_i(n)$:

$$\delta_i(n) = \frac{\partial \mathcal{E}(n)}{\partial w_i(n)} \tag{2.2.5}$$

Τότε, η ενημέρωση του βάρους $w_i(n)$ γίνεται σύμφωνα με τη σχέση:

$$w_i(n+1) = w_i(n) - \frac{\eta}{\sqrt{G_{ii}(n) + \epsilon}} \cdot \delta_i(n) \tag{2.2.6}$$

όπου $G(n)$ ένας τετραγωνικός πίνακας, όπου κάθε στοιχείο ii της διαγωνίου του αντιστοιχεί στο άθροισμα των τετραγώνων των κλίσεων του βάρους w_i μέχρι το βήμα n, και ϵ ένας διορθωτικός παράγοντας, που αποτρέπει τη διαίρεση με το μηδέν. Το βασικό πλεονέκτημα του αλγορίθμου είναι η αυτοματοποίηση της διαδικασίας του καθορισμού του ρυθμού μάθησης, καθιστώντας ευκολότερη την διαδικασία της εκπαίδευσης.

RMSprop Ο αλγόριθμος RMSprop αποτελεί μια επέκταση του αλγορίθμου Adagrad, ώστε να αποφύγουμε τη μονότονη μείωση του ρυθμού μάθησης ανά βάρος. Για αυτό το λόγο, δεν λαμβάνουμε υπόψιν το σύνολο των κλίσεων του βάρους για όλα τα βήματα που έχουν εκτελεστεί μέχρι στιγμής, αλλά εργαζόμαστε σε ένα κυλιόμενο παράθυρο των τελευταίων w βημάτων. Για τον αποδοτικότερο υπολογισμό της συνεισφοράς καθεμιάς από τις τελευταίες w κλίσεις, υπολογίζεται ο εκθετικά φθίνων μέσος όρος τους, έναντι του αθροίσματος τετραγώνων του Adagrad, στις διαγωνίους ενός τετραγωνικού πίνακα $E(n)$. Έτσι, για κάθε βάρος w_i στο βήμα n έχουμε

$$E_{ii}(n) = 0.9 \cdot E_{ii}(n-1) + 0.1 \cdot \delta_i^2(n) \tag{2.2.7}$$

ενώ η ενημέρωση του βάρους $w_i(n)$ γίνεται σύμφωνα με τη σχέση:

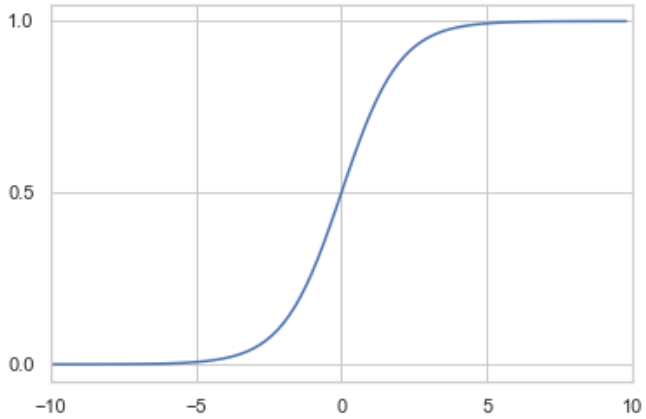
$$w_i(n+1) = w_i(n) - \frac{\eta}{\sqrt{E_{ii}(n) + \epsilon}} \cdot \delta_i(n) \tag{2.2.8}$$

2.3 Συναρτήσεις ενεργοποίησης

2.3.1 Σιγμοειδής συνάρτηση

Η σιγμοειδής συνάρτηση είναι μια από τις συνηθέστερες μορφές συνάρτησης ενεργοποίησης, απεικονίζεται στο σχήμα 2.4 και ορίζεται από τη σχέση:

$$\Phi(z) = \frac{1}{1 + e^{-z}} \tag{2.3.1}$$



Σχήμα 2.4: Σιγμοειδής συνάρτηση ενεργοποίησης.

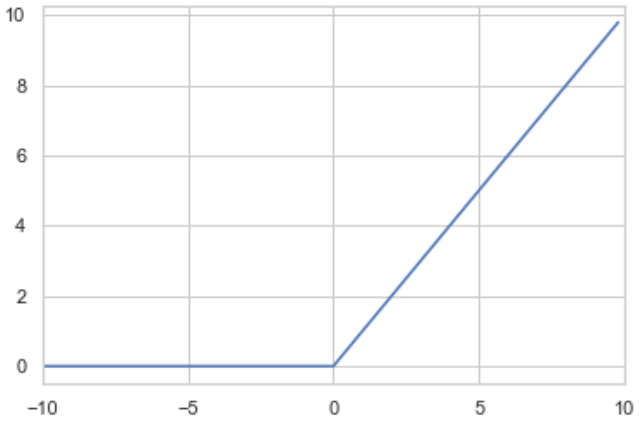
Βασικά πλεονεκτήματα της σιγμοειδούς συνάρτησης είναι η ομαλότητα που εμφανίζει, αποτρέποντας απότομες διακυμάνσεις στις τιμές εξόδου, αλλά και η εγγενής κανονικοποίηση της εξόδου στο διάστημα $(0, 1)$, γεγονός που την καθιστά χρήσιμη στην παραγωγή πιθανοτικών εκτιμήσεων. Σημαντικότερο μειονέκτημα είναι η πολύ μικρή τιμής της κλίσης της για διεγέρσεις που απέχουν πολύ από το μηδέν, γεγονός που εμποδίζει την αποτελεσματική εκπαίδευση σε τέτοιες περιπτώσεις.

2.3.2 Ανορθωμένη γραμμική μονάδα

Η συνάρτηση ενεργοποίσης ανορθωμένης γραμμικής μονάδας (Rectified Linear Unit) απεικονίζεται στο σχήμα 2.5 και ορίζεται από τη σχέση:

$$g(z) = max(0, z) \tag{2.3.2}$$

Η συνάρτηση ενεργοποίσης ανορθωμένης γραμμικής μονάδας έχει αναδειχθεί σε βασική σχεδιαστική επιλογή για τα σύγχρονα νευρωνικά δίκτυα, κα-



Σχήμα 2.5: Συνάρτηση ενεργοποίσης ανορθωμένης γραμμικής μονάδας.

θώς συνδυάζει την ευκολία υλοποίησης με την υψηλή απόδοση. Συγκεκριμένα, ο υπολογισμός της τιμής της γίνεται με ελάχιστο κόστος για τα σύγχρονα υπολογιστικά συστήματα, ενώ ο υπολογισμός της κλίσης της για την εκτέλεση του αλγορίθμου back-propagation γίνεται εξίσου απλά, αφού υπολογίζεται ως:

$$g'(z) = \begin{cases} 0, & z < 0 \\ 1, & z > 0 \\ \text{δεν ορίζεται}, & z = 0 \end{cases} \tag{2.3.3}$$

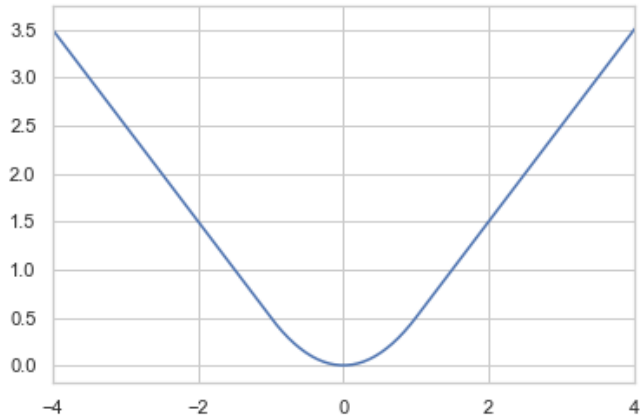
Βασικό μειονέκτημα της χρήσης της είναι η μηδενική κλίση για αρνητικές τιμές εισόδου, που αποτρέπει εντελώς την εκπαίδευση του νευρώνα.

2.4 Συναρτήσεις κόστους

2.4.1 Smooth L1

Η συνάρτηση κόστους $\text{smooth}_{\text{L1}}$ αποτελεί μια συνάρτηση που συνδυάζει σταθερή κλίση για μεγάλες τιμές σφάλματος, διευκολύνοντας την αρχική εκπαίδευση, και διαφορισιμότητα και μικρές κλίσεις για μικρές τιμές σφάλματος, εμποδίζοντας την έντονη ταλάντωση. Η συνάρτηση απεικονίζεται στο σχήμα 2.6 και ορίζεται ως:

$$\text{smooth}_{\text{L1}}(x) = \begin{cases} \frac{1}{2}x^2, & |x| < 1 \\ |x| - \frac{1}{2}, & \text{διαφορετικά} \end{cases} \tag{2.4.1}$$



Σχήμα 2.6: Συνάρτηση κόστους $\text{smooth}_{\text{L1}}$.

2.4.2 Weighted Sigmoid

Η συνάρτηση κόστους Weighted Sigmoid, όπως απεικονίζεται στο σχήμα 2.7, επεκτείνει τη συνάρτηση Σταυροειδούς Εντροπίας (cross-entropy), που χρησιμοποιείται για την αξιολόγηση του σφάλματος σε περιπτώσεις ταξινόμησης σε πολλές κλάσεις, όπου δεν υπάρχει αλληλεξάρτηση μεταξύ των διάφορων κλάσεων. Η συνάρτηση Δυαδικής Σταυροειδούς Εντροπίας υπολογίζεται ξεχωριστά ανά εξεταζόμενη κλάση $\mathcal{C}$, για την οποία το σύστημα παράγει πρόβλεψη ταξινόμησης $s_{\mathcal{C}}$, και ορίζεται ως:

$$CE(p_{\mathcal{C}}) = -\log(p_{\mathcal{C}}) \tag{2.4.2}$$

όπου:

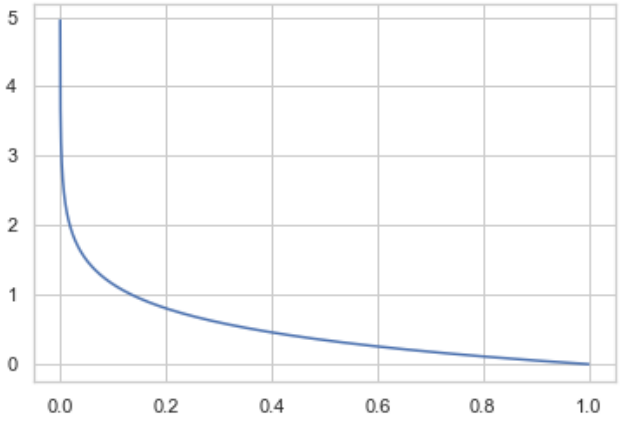
$$p_{\mathcal{C}} = \begin{cases} \frac{1}{1+\exp(-s_{\mathcal{C}})}, & \text{το αντικείμενο ανήκει στην κλάση } \mathcal{C} \\ 1 - \frac{1}{1+\exp(-s_{\mathcal{C}})}, & \text{διαφορετικά} \end{cases} \tag{2.4.3}$$

Με σκοπό τη βελτίωση της εκπαίδευσης συστημάτων εντοπισμού αντικειμένων, όταν υπάρχει μεγάλη ανισορροπία μεταξύ επιτυχημένων και αποτυχημένων εντοπισμών κατά την εκπαίδευση, ορίζουμε τη συνάρτηση Weighted Sigmoid, με την εισαγωγή της παραμέτρου α, ως:

$$L(p_{\mathcal{C}}) = -\alpha_{\mathcal{C}} \log(p_{\mathcal{C}}) \tag{2.4.4}$$

όπου

$$\alpha_{\mathcal{C}} = \begin{cases} \alpha, & \text{το αντικείμενο ανήκει στην κλάση } \mathcal{C} \\ 1-\alpha, & \text{διαφορετικά} \end{cases} \tag{2.4.5}$$

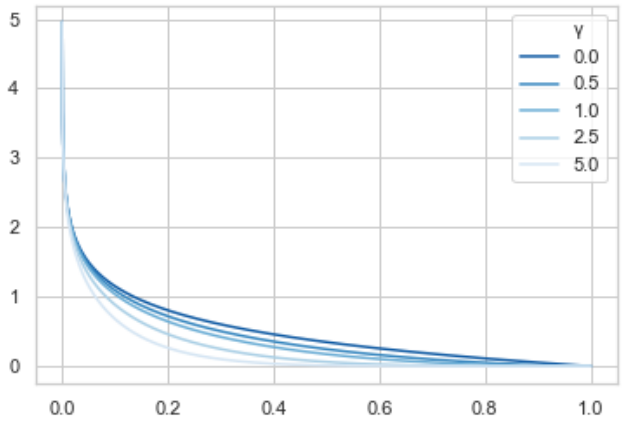


Σχήμα 2.7: Συνάρτηση κόστους Weighted Sigmoid. Παράδειγμα με $\alpha = 0.5$.

2.4.3 Weighted Sigmoid Focal

Η συνάρτηση κόστους Weighted Sigmoid Focal, όπως απεικονίζεται στο σχήμα 2.8, αποτελεί επέκταση της Weighted Sigmoi, με σκοπό να δοθεί μεγαλύτερη έμφαση στα πρότυπα εκπαίδευσης με δυσκολότερες περιπτώσεις αποτυχημένου εντοπισμού[Lin+17]. Η συνάρτηση εισάγει ένα παράγοντα γ που ρυθμίζει την ποσότητα $(1 - p_{\mathcal{C}})^{\gamma}$, ώστε να ενισχύεται περισσότερο η συνεισφορά των δυσκολότερων περιπτώσεων και τελικά ορίζεται ως:

$$L(p_{\mathcal{C}}) = -\alpha_{\mathcal{C}}(1 - p_{\mathcal{C}})^{\gamma} \log(p_{\mathcal{C}}) \tag{2.4.6}$$



Σχήμα 2.8: Συνάρτηση κόστους Weighted Sigmoid Focal. Παράδειγμα με $\alpha = 0.5$.

2.5 Αλγόριθμος back-propagation

Ο καθιερωμένος αλγόριθμος για τον επιμερισμό των παραγόμενων σφαλμάτων στους νευρώνες ενός πολυεπίπεδου δικτύου και την αναπροσαρμογή των βαρών τους δεν είναι άλλος από των αλγόριθμο οπίσθιας διάδοσης σφαλμάτων (back-propagation), που διατυπώθηκε από τους Rumelhart και McClelland. Σύμφωνα με τον Haykin[Hay10, σελ. 125], οι ανάγκες λειτουργίας του αλγορίθμου απαιτούν δύο είδη σημάτων που διασχίζουν ένα πολυεπίπεδο perceptron:

Λειτουργικά σήματα Τα λειτουργικά σήματα συνιστούν διεγέρσεις με εμπρόσθια φορά διάδοσης στο δίκτυο. Συγκεκριμένα, ξεκινούν από το επίπεδο εισόδου του δικτύου και, μέσω της διαδοχικής διέγερσης, βάσει της εκάστοτε συνάρτησης ενεργοποίησης, των νευρώνων των κρυφών επιπέδων, καταλήγουν στη διέγερση του επιπέδου εξόδου.

Σήματα σφάλματος Τα σήματα σφάλματος συνιστούν διεγέρσεις με οπίσθια φορά διάδοσης στο δίκτυο. Συγκεκριμένα, ξεκινούν από το επίπεδο εξόδου του δικτύου και, μέσω της διαδοχικής διέγερσης των νευρώνων των κρυφών επιπέδων, βάσει μιας συναρτήσεως σφάλματος, καταλήγουν στη διέγερση του επιπέδου εισόδου.

Στο ευρύτερο πλαίσιο λειτουργίας του αλγόριθμου back-propagation, ο τρόπος αξιολόγησης του σφάλματος και αναπροσαρμογής των συναπτικών βαρών προσδιορίζεται από τις σχεδιαστικές επιλογές της διαδικασίας της εκπαίδευσης και, ειδικότερα, από την επιλογή συναρτήσεων ενεργοποίησης, συνάρτησης κόστους και αλγόριθμου βελτιστοποίησης.

Για να περιγράψουμε με σαφήνεια τον τρόπο λειτουργίας του αλγορίθμου, θα παραθέσουμε το ακόλουθο παράδειγμα που υλοποιεί μια σχετικά απλή διαδικασία εκπαίδευσης ενός πολυεπίπεδου perceptron, με χρήση του μέσου τετραγωνικού σφάλματος και της βελτιστοποίησης βαθμιαίας καθόδου με online μάθηση.

Κατά τη διάρκεια του n-στου βήματος της εκπαίδευσης, υποθέτουμε ότι κάθε νευρώνας j διεγείρεται από το σύνολο των λειτουργικών σημάτων y_i που παράγουν οι m το πλήθος νευρώνες του προηγούμενου επιπέδου και, συμπεριλαμβάνοντας το χαρακτηριστικό της πόλωσης με τον τρόπο που είδαμε νωρίτερα, αποκτά δικτυακή διέγερση:

$$u_j(n) = \sum_{i=0}^{i=m} w_{ji}(n) y_i(n) \tag{2.5.1}$$

Το παραγόμενο λειτουργικό σήμα του νευρώνα j, δεδομένης μιας μη-γραμμικής, διαφορίσιμης συνάρτησης ενεργοποίησης $\phi_j(\cdot)$ θα είναι:

$$y_j(n) = \phi_j(u_j(n)) \tag{2.5.2}$$

Καθώς αίρεται ο περιορισμός της μοναδικής τιμής εξόδου του απλού perceptron, το δείγμα εκπαίδευσης ενός πολυεπίπεδου perceptron με k το πλήθος νευρώνες εξόδου θα πρέπει να περιέχει ένα διάνυσμα επιθυμητών αποκρίσεων

$$\mathbf{d}(n) = \begin{bmatrix} d_1(n) & d_2(n) & \cdots & d_k(n) \end{bmatrix} \tag{2.5.3}$$

όπου κάθε τιμή $d_j(n)$ αντιστοιχεί στην επιθυμητή έξοδο του νευρώνα εξόδου j για το πρότυπο εισόδου n.

Κατ' επέκταση, για τον υπολογισμό της συνάρτησης κόστους θα πρέπει να ληφθεί υπ' όψιν η συνεισφορά του συνόλου των νευρώνων εξόδου, συνεπώς ορίζουμε τη συνάρτηση συνολικού κόστους ως:

$$\mathcal{E}(n) = \sum_{j=1}^{j=k} \mathcal{E}_j(n) = \frac{1}{2} \sum_{j=1}^{j=k} e_j^2(n) \tag{2.5.4}$$

όπου το σφάλμα που συνεισφέρει κάθε νευρώνας εξόδου είναι όπως και πριν

$$e_j(n) = d_j(n) - y_j(n) \tag{2.5.5}$$

Για τον υπολογισμό της αναπροσαρμογής των συναπτικών βαρών του νευρώνα j, θα πρέπει να υπολογίσουμε πρώτα τη επίδραση κάθε συναπτικού βάρους w_{ji} στη συνάρτηση κόστους. Όπως περιγράφει ο Haykin[Hay10, σελ.

130], εφαρμόζουμε την ίδια διαδικασία με τον αλγόριθμο Ελαχίστων Μέσων Τετραγώνων, κάνοντας παράλληλη χρήση του κανόνα της αλυσίδας:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} \frac{\partial u_j(n)}{\partial w_{ji}(n)} \tag{2.5.6}$$

Εκμεταλλευόμενοι το γεγονός ότι:

$$\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \overset{(2.5.4)}{=\!=} e_j(n) \tag{2.5.7}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} \overset{(2.5.5)}{=\!=} -1 \tag{2.5.8}$$

$$\frac{\partial y_j(n)}{\partial u_j(n)} \overset{(2.5.2)}{=\!=} \phi'(u_j(n)) \tag{2.5.9}$$

$$\frac{\partial u_j(n)}{\partial w_{ji}(n)} \overset{(2.5.1)}{=\!=} y_i(n) \tag{2.5.10}$$

καταλήγουμε σε μια τελική σχέση της μορφής:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\phi'_j(u_i(n))y_i(n) \tag{2.5.11}$$

Θα πρέπει να σημειώσουμε ότι στη σχέση (2.5.8), το αποτέλεσμα επεκτείνεται και στους κρυφούς νευρώνες του δικτύου, καθώς εμφανίζουν σφάλμα σε κάθε εκτέλεση του αλγορίθμου, αλλά βάσει σχεδιασμού μας είναι άγνωστη η επιθυμητή τους έξοδος. Μετέπειτα θα δούμε με ποιο τρόπο είναι εφικτό να ξεπεράσουμε αυτή τη τεχνική δυσκολία, ώστε να γίνει εφικτή η εκπαίδευση του συνόλου του δικτύου.

Στη συνέχεια, καθώς εκτελούμε βελτιστοποίηση μέσω βαθμιαίας καθόδου, η διόρθωση $\Delta w_{ji}(n)$ που επιδέχεται το βάρος w_{ji} θα είναι

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} \overset{(2.5.11)}{=\!=} \eta\, \delta_j(n) y_i(n) \tag{2.5.12}$$

όπου ορίζουμε την τοπική κλίση $\delta_j(n)$ ως:

$$\delta_j(n) = \frac{\partial \mathcal{E}(n)}{\partial u_j(n)} = e_j(n)\phi'_j(u_i(n)) \tag{2.5.13}$$

Για την τελική έκφραση της διόρθωσης των βαρών διακρίνουμε δύο διαδικασίες ενημέρωσης, ανάλογα με τον τύπο του νευρώνα προς ενημέρωση:

Νευρώνας εξόδου Σε περίπτωση που ενημερώνουμε τα βάρη ενός νευρώνα εξόδου, γνωρίζουμε ήδη την επιθυμητή του έξοδο $d_j(n)$, άρα, χρησιμοποιώντας τη σχέση (2.5.5), μπορούμε να εκφράσουμε την κλίση ως:

$$\delta_j(n) = (d_j(n) - y_j(n))\phi_j'(u_i(n)) \tag{2.5.14}$$

Κρυφός νευρώνας Σε περίπτωση που ενημερώνουμε τα βάρη ενός κρυφού νευρώνα, μας είναι εξ ορισμού άγνωστη η επιθυμητή του έξοδος, οπότε θα πρέπει να εργαστούμε διαφορετικά για τον υπολογισμό της τοπικής κλίσης. Χωρίς βλάβη της γενικότητας, θα υποθέσουμε ότι ο νευρώνας j ανήκει στο τελευταίο κρυφό επίπεδο και ότι ακολουθούν k το πλήθος νευρώνες εξόδου. Με χρήση του κανόνα της αλυσίδας στη σχέση (2.5.14), θα έχουμε:

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \phi_j'(u_i(n)) \tag{2.5.15}$$

όπου:

$$\frac{\partial e_l(n)}{\partial u_l(n)} \overset{(2.5.5),(2.5.2)}{=\!=} -\phi_l'(u_l(n)) \tag{2.5.16}$$

$$\frac{\partial u_l(n)}{\partial y_j(n)} \overset{(2.5.1)}{=\!=} w_{lj}(n) \tag{2.5.17}$$

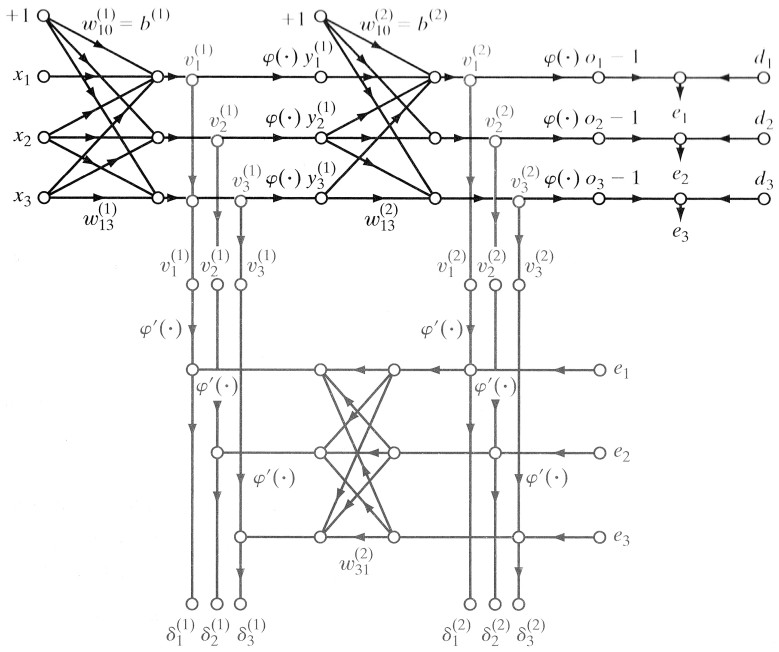
$$\begin{aligned}
\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} &\overset{(2.5.4)}{=\!=} \sum_{l=1}^{l=k} e_l(n) \frac{\partial e_l(n)}{\partial y_j(n)} = \sum_{l=1}^{l=k} e_l(n) \frac{\partial e_l(n)}{\partial u_l(n)} \frac{\partial u_l(n)}{\partial y_j(n)} \\
&\overset{(2.5.16),(2.5.17)}{=\!=\!=} -\sum_{l=1}^{l=k} e_l(n)\, \phi_l'(u_l(n))\, w_{lj}(n) \\
&\overset{(2.5.13)}{=\!=} -\sum_{l=1}^{l=k} \delta_l(n)\, w_{lj}(n)
\end{aligned} \tag{2.5.18}$$

Τελικά, υπολογίζουμε την τοπική κλίση για ένα νευρώνα j που ανήκει στο τελευταίο κρυφό επίπεδο ως:

$$\delta_j(n) \overset{(2.5.15),(2.5.18)}{=\!=\!=} \phi_j'(u_i(n)) \sum_{l=1}^{l=k} \delta_l(n)\, w_{lj}(n) \tag{2.5.19}$$

Για λόγους απλότητας νωρίτερα υποθέσαμε ότι ο νευρώνας j ανήκει στο τελευταίο κρυφό επίπεδο και ότι ακολουθούν k το πλήθος νευρώνες εξόδου. Στη συνέχεια καταλήξαμε σε ένα τύπο υπολογισμού των τοπικών

κλίσεων $\delta_j(n)$ με βάση τις τοπικές κλίσεις των νευρώνων του επόμενου επιπέδου $\delta_l(n)$. Συνεπώς, καθώς γνωρίζουμε πλέον το σύνολο των τοπικών κλίσεων $\delta_j(n)$, μπορούμε να εργαστούμε αναδρομικά, υπολογίζοντας τις τοπικές κλίσεις του κάθε φορά προηγούμενου επιπέδου, μέχρι να φτάσουμε στο νευρώνα εξόδου. Η διαδικασία αυτή για ένα απλό πολυεπίπεδο perceptron ενός κρυφού επιπέδου απεικονίζεται στο σχήμα 2.9.



Σχήμα 2.9: Διάγραμμα ροής σημάτων αλγορίθμου back-propagation. Πηγή: [Hay10, σελ. 140]

Κεφάλαιο 3

Συνελικτικά νευρωνικά δίκτυα

3.1 Βασικά χαρακτηριστικά

Τα συνελικτικά δίκτυα είναι μια ειδική κατηγορία πολυεπίπεδων νευρωνικών δικτύων, που εξειδικεύονται στην αναγνώριση δισδιάστατων χαρακτηριστικών, διατηρώντας το αναλλοίωτο σε μεταβολές, δηλαδή είναι σε θέση να πετύχουν μια αναγνώριση με μικρή ευαισθησία σε μετατόπιση, κλιμάκωση ή οποιαδήποτε άλλη παραμόρφωση.[Hay10]

Όπως σημειώνουν οι Goodfellow, Bengio και Courville, τα συνελικτικά δίκτυα είναι απλά νευρωνικά δίκτυα που αντικαθιστούν τον πολλαπλασιασμό πινάκων με την πράξη της συνέλιξης, σε τουλάχιστον ένα από τα επίπεδά τους.[GBC16]

3.2 Δομικά στοιχεία

3.2.1 Επίπεδο συνέλιξης

Στο πεδίο της ανάλυσης, η πράξη της συνέλιξης δύο πραγματικών συναρτήσεων $f, g = \mathbb{R}^d \to \mathbb{R}$ που ορίζεται ως:

$$f * g = \int_{\mathbb{R}^d} f(z)g(x-z)dz \tag{3.2.1}$$

Στα πλαίσια της όρασης υπολογιστών , η συνέλιξη είναι η εφαρμογή ενός πυρήνα **K** σε μια είσοδο **I**, παράγοντας ένα χάρτη χαρακτηριστικών **S**. Στην περίπτωση δισδιάστατης εισόδου και εξόδου, η συνέλιξη υπολογίζεται με τον ακόλουθο τρόπο, όπου στη σχέση 3.2.2 εκμεταλλευτήκαμε την αντιμεταθετική ιδιότητα της συνέλιξης:

$$\mathbf{S}_{i,j} = (\mathbf{I} * \mathbf{K})_{i,j} = (\mathbf{K} * \mathbf{I})_{i,j} \tag{3.2.2}$$

$$= \sum_{m,n} \mathbf{K}_{m,n} \mathbf{I}_{i-m,j-n} \tag{3.2.3}$$

Στην περίπτωση τρισδιάστατης εισόδου και εξόδου (όχι απαραίτητα ίδιου βάθους) η πράξη της συνέλιξης μετατρέπεται κατάλληλα σε:

$$\mathbf{S}_{i,j,k} = \sum_{m,n,c} \mathbf{K}_{m,n,c,k} \mathbf{I}_{i-m,j-n,c} \tag{3.2.4}$$

Η χρήση ενός συνελικτικού επιπέδου με πυρήνα **K** υπερέχει της χρήσης ενός πλήρως συνδεδεμένου επιπέδου ως προς τα ακόλουθα χαρακτηριστικά:

Αραιή αλληλεπίδραση Ο πυρήνας **K** έχει συνήθως αρκετά μικρότερο μέγεθος από την είσοδο, αφού καλείται να εξάγει χαρακτηριστικά μικρού μεγέθους. Αυτό συνεπάγεται μικρότερες απαιτήσεις του δικτύου σε υπολογιστική ισχύ, αφού σε ένα πλήρως συνδεδεμένο επίπεδο m εισόδων και n εξόδων οι υπολογισμοί απαιτούν χρόνο $\mathcal{O}(m \times n)$, ενώ σε ένα συνελικτικό επίπεδο με πυρήνα k παραμέτρων οι υπολογισμοί απαιτούν χρόνο $\mathcal{O}(k \times n)$.

Κοινές παράμετροι Καθώς οι παράμετροι που καθορίζουν στον πυρήνα της συνέλιξης είναι κοινοί για όλο το εύρος της εφαρμογής του, ένα συνελικτικό επίπεδο απαιτεί λιγότερες ελεύθερες παραμέτρους σε σχέση με ένα πλήρως συνδεδεμένο επίπεδο, διευκολύνοντας τη διαδικασία της εκπαίδευσης και μειώνοντας τον κίνδυνο υπερπροσαρμογής στα δεδομένα.

Συμμετρία μεταβολών Καθώς η εφαρμογή του πυρήνα **K** γίνεται σε εκάστοτε τμήματα ολόκληρης της έκτασης της εισόδου, η μεταβολή ενός χαρακτηριστικού στην είσοδο θα επιφέρει ανάλογη μεταβολή στον παραγόμενο χάρτη χαρακτηριστικών της εξόδου.

Κατά τον υπολογισμό της συνέλιξης, χρειάζεται ειδικός χειρισμός στις περιπτώσεις των στοιχείων που βρίσκονται στα όρια της εισόδου, αφού τμήμα του πυρήνα θα πρέπει να εφαρμοστεί σε στοιχεία εκτός της εισόδου. Σε αυτή την περίπτωση, έχουμε δύο επιλογές:

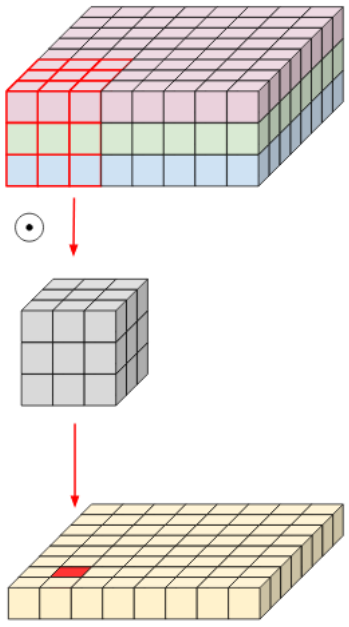
Παράλειψη του σημείου Δεν γίνεται υπολογισμός της συνέλιξης για όσα σημεία προκαλούν την εφαρμογή της εκτός ορίου εισόδου. Σε αυτή την περίπτωση έχουμε αναγκαστικά μείωση των διαστάσεων της εξόδου, σε σχέση με την είσοδο.

Επέκταση της εισόδου Μπορούμε να επεκτείνουμε την είσοδο με επιπλέον στοιχεία, συνήθως μηδενικής τιμής, στα οποία δύναται να εφαρμοστεί στη συνέχεια ο πυρήνας της συνέλιξης. Ανάλογα με το πλήθος των στοιχείων που προστέθηκαν ανά διάσταση, διαμορφώνονται ανάλογα και οι διαστάσεις της εξόδου. Μια αρκετά συνηθισμένη επιλογή είναι η επέκταση της εισόδου ώστε η παραγόμενη έξοδος να έχει τις ίδιες διαστάσεις.

Επιπλέον, μπορούμε να τροποποιήσουμε κατάλληλα τον τρόπο υπολογισμού της συνέλιξης, ώστε να μην εφαρμόζεται απαραίτητα σε συνεχόμενα στοιχεία της εισόδου, αλλά να χρησιμοποιεί ένα βήμα s ανάμεσα σε διαδοχικά στοιχεία. Τοτε, ο υπολογισμός της εξόδου γίνεται:

$$\mathbf{S}_{i,j,k} = \sum_{m,n,c} \mathbf{K}_{m,n,c,k} \mathbf{I}_{s \cdot i - m, s \cdot j - n, c} \tag{3.2.5}$$

Στο σχήμα 3.1 απεικονίζεται η εκτέλεση μιας συνέλιξης με μοναδιαίο βήμα και επέκταση της εισόδου, ώστε η έξοδος να έχει τις ίδιες διαστάσεις.

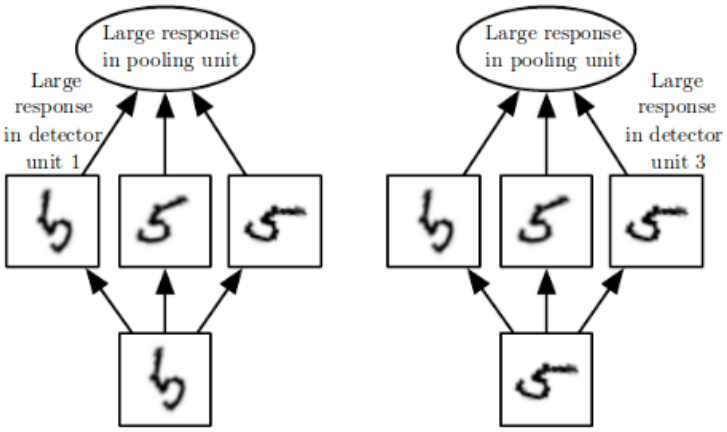


Σχήμα 3.1: Παράδειγμα εφαρμογής της πράξης της συνέλιξης. Εφαρμογή πυρήνα διαστάσεων $3 \times 3 \times 3 \times 1$ σε είσοδο διαστάσεων $8 \times 8 \times 3$ για την παραγωγή χάρτη χαρακτηριστικών διαστάσεων $8 \times 8 \times 1$. Πηγή: [Ben18]

3.2.2 Επίπεδο συγκέντρωσης

Η λειτουργία που επιτελεί ένα επίπεδο συγκέντρωσης είναι η αντικατάσταση κάθε στοιχείου της εισόδου με την τιμή μιας συνάρτησης ομαδοποίησης (aggregation function) που εφαρμόζεται σε μια κοντινή γειτονιά του στοιχείου. Ουσιαστικά, εκτελείται υποδειγματοληψία των χαρακτηριστικών της εισόδου, αφού η ανάλυση του παραγόμενου χάρτη χαρακτηριστικών μειώνεται. Ενδεικτικό παράδειγμα συνάρτησης ομαδοποίησης αποτελεί η συνάρτηση **max pooling**, που επιστρέφει το μέγιστο στοιχείο μιας ορθογωνικής γειτονιάς.

Η χρήση του επιπέδου συγκέντρωσης ενισχύει το αναλλοίωτο του δικτύου σε μεταβολές της εισόδου. Αυτή η ιδιότητα αποδεικνύεται ιδιαίτερα χρήσιμη σε περιπτώσεις που μας ενδιαφέρει περισσότερο η ανίχνευση της ύπαρξης ενός χαρακτηριστικού, χωρίς να μας απασχολεί ο προσδιορισμός της θέσης του. Ένα παράδειγμα της παραπάνω συμπεριφοράς περιγράφεται στο σχήμα 3.2.



Σχήμα 3.2: Παράδειγμα αναλλοίωτου σε μεταβολές. Πηγή: [GBC16]

3.2.3 Επίπεδο ενεργοποίησης

Το επίπεδο ενεργοποίησης εφαρμόζει την επιλεγμένη μη-γραμμική συνάρτηση ενεργοποίησης στα στοιχεία της εισόδου, παράγοντας μια έξοδο με τα αντίστοιχα αποτελέσματα.

3.2.4 Επίπεδο κανονικοποίησης

Κατά την εκπαίδευση ενός πολυεπίπεδου συνελικτικού δικτύου, η αναπροσαρμογή των βαρών ενός επιπέδου έχει άμεσες επιπτώσεις στο παραγόμενο χάρτη χαρακτηριστικών που λαμβάνει το επόμενο επίπεδο ως είσοδο. Συγκεκριμένα, αν δεν έχει ληφθεί υπ' όψιν η επίδραση της κλίμακας των στοιχείων

του χάρτη εισόδου, τότε μια αλλαγή σε αυτή θα επιφέρει αναπόφευκτα αλλαγή στα συναπτικά βάρη του επόμενου επιπέδου. Έτσι, εμφανίζονται περιπτώσεις που το επόμενο επίπεδο θα πρέπει να αναπροσαρμόσει τα βάρη του, όχι λόγω ποιοτικής διαφοροποίησης των δεδομένων εισόδου, αλλά απλά λόγω αλλαγής της κλίμακάς τους.

Αν υποθέσουμε ότι τα δεδομένα εισόδου έχουν τη μορφή

$$\mathbf{x}_i = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\top \tag{3.2.6}$$

και τροφοδοτούνται στο υπό εκπαίδευση δίκτυο σε παρτίδες της μορφής

$$\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\} \tag{3.2.7}$$

τότε μπορούμε να κανονικοποιήσουμε τις τιμές κάθε καναλιού εισόδου k βάσει της σχέσης:

$$\hat{x}_k = \frac{x_k - E[x_k]}{\sqrt{Var[x_k]}} \tag{3.2.8}$$

όπου ο μέσος όρος $E[x_k]$ και διακύμανση $Var[x_k]$ υπολογίζονται βάσει του συνόλου των δεδομένων εκπαίδευσης.

Όμως, οπως σημειώνουν οι Ioffe και Szegedy[IS15] η εφαρμογή μιας τυπικής κανονικοποίησης της εισόδου δεν μπορεί να είναι αποδοτική, καθώς είναι σε θέση να αλλοιώσει την αναπαράσταση πληροφορίας που παράγει το επίπεδο. Για παράδειγμα, η απλή κανονικοποίηση της εισόδου ενός επιπέδου σιγμοειδούς συνάρτησης ενεργοποίησης οδηγεί στη χρήση μόνο του γραμμικού της τμήματος, ελαττώνοντας τη δυνατότητα μάθησης του επιπέδου.

Αντ' αυτού, προτείνεται η χρήση δύο νέων παραμέτρων της εκπαίδευσης γ_k και β_k, που συνδυαστικά παράγουν την κανονικοποιημένη τιμή μέσω του εξής γραμμικού μετασχηματισμού:

$$y_k = \gamma_k \cdot \hat{x}_k + \beta_k \tag{3.2.9}$$

Για τη χρήση της μεθόδου για εκπαίδευση σε παρτίδες, παρατηρούμε ότι ο μέσος όρος και η διακύμανση που υπολογίζονται στο εύρος μιας παρτίδας αποτελούν εκτιμήτριες των αντίστοιχων συναρτήσεων για το σύνολο των δεδομένων. Με αυτό τον τρόπο, οι καθολικές συναρτήσεις μέσου όρου και διακύμανσης μπορούν να αντικατασταθούν με τις εκτιμήτριες τους, που είναι συμβατές με την επεξεργασία σε παρτίδες και με τον αλγόριθμο back-propagation.

Βασικά πλεονεκτήματα της κανονικοποίησης σε παρτίδες είναι η δυνατότητα χρήσης υψηλών ρυθμών μάθησης, χωρίς να κινδυνεύει η δυνατότητα σύγκλισης, και η συνολικά ταχύτερη διαδικασία μάθησης.

3.3 Διαχωρίσιμες ανά κανάλι Συνελίξεις

Οι Διαχωρίσιμες ανά κανάλι Συνελίξεις (Depthwise Separable Convolutions) αποτελούν μια υπολογιστικά αποδοτικότερη εξέλιξη των συμβατικών συνελίξεων, όταν αυτές αφορούν δεδομένα εισόδου τριών διαστάσεων. Αν υποθέσουμε ένα επίπεδο συνέλιξης με είσοδο ένα χάρτη χαρακτηριστικών με βάθος M και έξοδο ένα χάρτη χαρακτηριστικών με βάθος N, ο βασικός τρόπος λειτουργίας των διαχωρίσιμων ανά κανάλι συνελίξεων, όπως απεικονίζεται στο παράδειγμα του σχήματος 3.3, αναλύεται σε δύο στάδια:

Συνέλιξη ανά κανάλι (Depthwise Convolution) Στο πρώτο στάδιο επιμερίζουμε το σύνολο της συνέλιξης σε M το πλήθος δισδιάστατες συνελίξεις. Στην m-στη τέτοια συνέλιξη, εκτελείται ο συνηθισμένος υπολογισμός της συνέλιξης χρησιμοποιώντας αποκλειστικά το m-στο κανάλι του χάρτη εισόδου και το m-στο κανάλι του πυρήνα. Με το πέρας των υπολογισμών, οι M το πλήθος δισδιάστατοι χάρτες που παράγονται συνδιάζονται για τη δημιουργία ενός ενδιάμεσου χάρτη εξόδου, βάθους M.

Συνέλιξη ανά σημείο (Pointwise Convolution) Στο δεύτερο στάδιο, για να καταλήξουμε σε ένα χάρτη εξόδου με διαστάσεις εκείνου μιας τυπικής συνέλιξης, θα εκτελέσουμε μια $1 \times 1 \times M \times N$ συνέλιξη για κάθε επιμέρους στοιχείο του ενδιάμεσου χάρτη που παράχθηκε προηγουμένως. Το τελικό αποτέλεσμα θα είναι ένας χάρτης εξόδου βάθους N, όπως και στην εφαρμογή μιας απλής συνέλιξης.

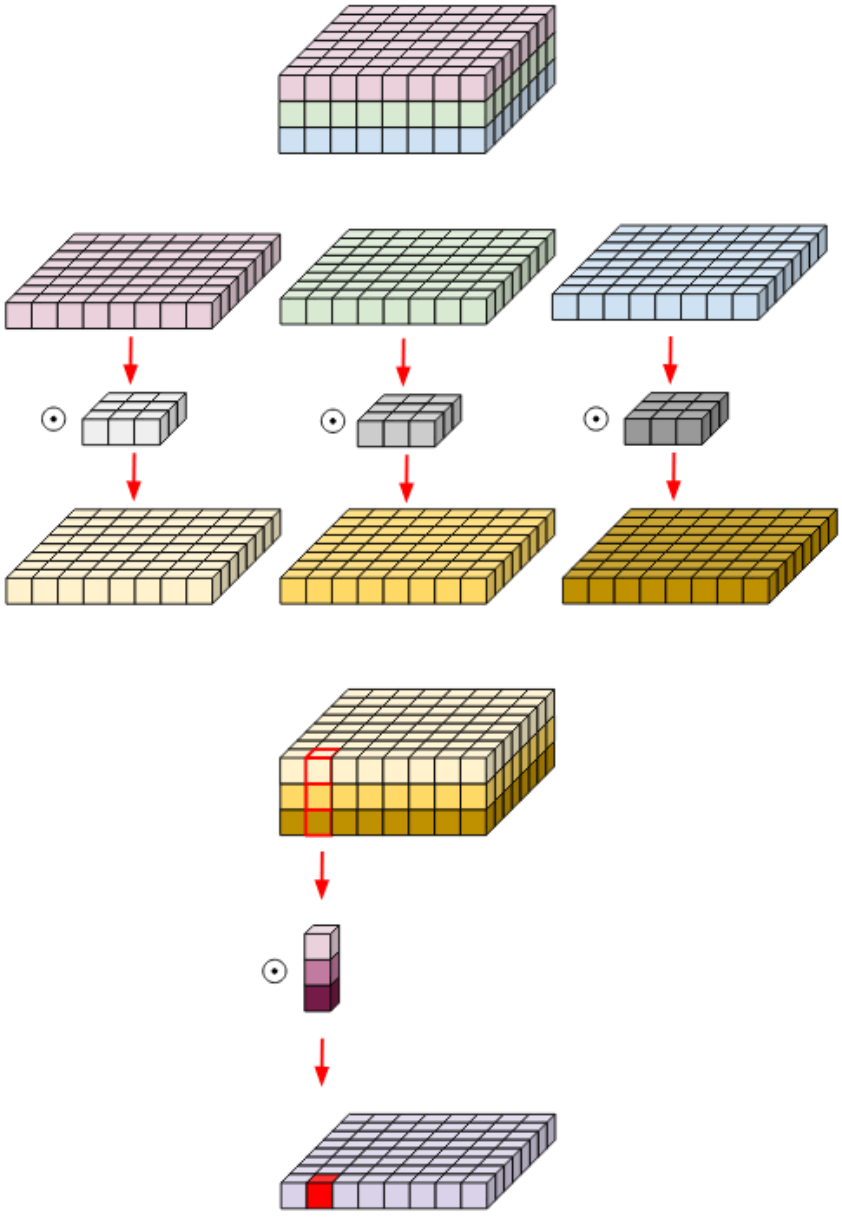
Όπως περιγράφεται στην αρχιτεκτονική ανάλυση των MobileNets[How+17], η ειδοποιός διαφορά με την απλή συνέλιξη γίνεται εμφανής αν αναλογιστούμε το υπολογιστικό κόστος της κάθε περίπτωσης. Γι' αυτό το σκοπό, ας θεωρήσουμε ένα επίπεδο συνέλιξης με είσοδο ένα χάρτη χαρακτηριστικών **F** διαστάσεων $H_F \times W_F \times M$, όπου τα H_F, W_F αντιστοιχούν στο ύψος και το πλάτος των δεδομένων εισόδου και το M αντιστοιχεί στο πλήθος των καναλιών εισόδου (βάθος). Θεωρούμε επίσης ότι ο παραγόμενος χάρτης χαρακτηριστικών **G** έχει διαστάσεις $H_G \times W_G \times N$ και ο πυρήνας της συνέλιξης **K** έχει διαστάσεις $H_K \times W_K \times M \times N$

Σε μια εκτέλεση τυπικής συνέλιξης, ο χάρτης εξόδου υπολογίζεται ως:

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \, \mathbf{F}_{k+i-1,l+j-1,m} \tag{3.3.1}$$

με υπολογιστικό κόστος:

$$H_K \cdot W_K \cdot M \cdot N \cdot H_F \cdot W_F \tag{3.3.2}$$



Σχήμα 3.3: Απεικόνιση διαχωρίσιμης ανά κανάλι συνέλιξης. Στο παραπάνω παράδειγμα έχουμε χάρτη εισόδου διαστάσεων $8 \times 8 \times 3$, πυρήνα διαστάσεων $3 \times 3 \times 3$ και χάρτη εξόδου διαστάσεων $8 \times 8 \times 1$. Πηγή: Eli Bendersky [Ben18]

Αντίθετα, σε μια εκτέλεση διαχωρίσιμης ανά κανάλι συνέλιξης, στο πρώτο στάδιο η παραπάνω συνέλιξη επιμερίζεται σε M το πλήθος πυρήνες δύο διαστάσεων, με τον ενδιάμεσο χάρτη εξόδου $\mathbf{G}'$ για το m-στό κανάλι να δίνεται από τη σχέση:

$$\mathbf{G}'_{k,l,m} = \sum_{i,j} \mathbf{K}_{i,j,m} \, \mathbf{F}_{k_i-1,l+j-1,m} \tag{3.3.3}$$

και συνολικό υπολογιστικό κόστος να είναι:

$$H_K \cdot W_K \cdot M \cdot H_F \cdot W_F \tag{3.3.4}$$

Στο δεύτερο βήμα, εκτελείται μια $1 \times 1 \times M \times N$ συνέλιξη σε κάθε ένα από τα $H_F \cdot W_F$ σημεία του ενδιάμεσου χάρτη εξόδου, με συνολικό υπολογιστικό κόστος:

$$M \cdot N \cdot H_F \cdot W_F \tag{3.3.5}$$

Για το σύνολο της εκτέλεσης της διαχωρίσιμης ανά κανάλι συνέλιξης, έχουμε υπολογιστικό κόστος:

$$H_K \cdot W_K \cdot M \cdot H_F \cdot W_F + M \cdot N \cdot H_F \cdot W_F \tag{3.3.6}$$

Ο συντελεστής μείωσης του υπολογιστικού κόστους υπολογίζεται ως

$$\frac{H_K \cdot W_K \cdot M \cdot H_F \cdot W_F + M \cdot N \cdot H_F \cdot W_F}{H_K \cdot W_K \cdot M \cdot N \cdot H_F \cdot W_F} = \frac{1}{N} + \frac{1}{H_K \cdot W_K} \tag{3.3.7}$$

και μπορεί να γίνει κατανοητό ότι για σχετικά μικρό βάθος εξόδου και μικρό μέγεθος φίλτρου, όπως $N = 20$ και $H_K = 4, W_K = 5$, αντίστοιχα, είναι εφικτή μια μείωση του υπολογιστικού κόστους κατά μια τάξη μεγέθους.

3.4 Επίπεδο συμφόρησης

Όπως είδαμε παραπάνω, η εφαρμογή ενός πυρήνα συνέλιξης σε μια είσοδο οδηγεί στην παραγωγή ενός χάρτη χαρακτηριστικών . Η βασική υπόθεση σχεδιασμού συνελικτικών δικτύων είναι ότι το σύνολο της χρήσιμης πληροφορίας του χώρου της εισόδου μπορεί να ενσωματωθεί σε υποχώρους μικρότερου βάθους. Συνεπώς, θα ήταν σχεδιαστικά καλή επιλογή να επιδιώξουμε τη μείωση του βάθους σε βαθμό που να ταυτίζεται με τη διάσταση του χώρου της χρήσιμης πληροφορίας.

Το επίπεδο συμφόρησης αποτελεί μια ειδική περίπτωση Συνέλιξης ανά σημειό, όπου το βάθος του χάρτη εξόδου είναι μικρότερο από το βάθος της εισόδου. Με αυτό τον τρόπο, πετυχαίνουμε μείωση του βάθους της εξόδου και, κατ' επέκταση, μειώνουμε τον αριθμό παραμέτρων του συστήματος, αφού η έξοδος μειωμένου βάθους χρησιμοποιείται ως είσοδος στο επόμενο επίπεδο.

3.5 Επίπεδο γραμμικής συμφόρησης

Το επίπεδο γραμμικής συμφόρησης αποτελεί ειδική περίπτωση του επιπέδου συμφόρησης, με την απαίτηση ότι δεν έχει εφαρμοστεί κάποιος μη-γραμμικός μετασχηματισμός στην είσοδό της, αφού η εφαρμογή μη-γραμμικών μετασχηματισμών αλλοιώνει το περιεχόμενο της πληροφορίας.

Υπό περιπτώσεις, είναι εφικτή η διατήρηση της πληροφορίας μετά την εφαρμογή της Ανορθωμένης Γραμμικής Μονάδας στην είσοδο, εφόσον υπάρχει επαρκές πλήθος μη-μηδενικών ενεργοποιήσεων και το μέγεθος της χρήσιμης πληροφορίας είναι αρκετά μικρό σε σχέση με το μέγεθος της εισόδου[San+18].

3.6 Μπλοκ με κατάλοιπο

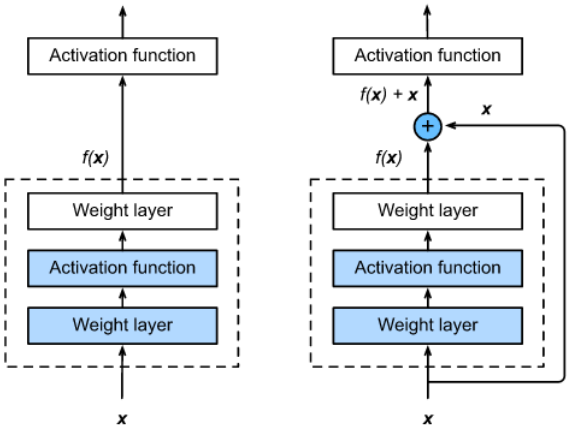
Μια βασική σχεδιαστική επιλογή ενός συνελικτικού δικτύου είναι το πλήθος των επιπέδων που το απαρτίζουν. Καθώς καλούμαστε να επιλύσουμε ολοένα και πιο σύνθετα προβλήματα, αξίζει να εξετάσουμε τι επίδραση έχει η προσθήκη επιπέδων στις επιδόσεις του δικτύου και πως μπορεί να γίνει αποδοτικά.

Αν υποθέσουμε ότι η αρχιτεκτονική ενός δικτύου, με σύνολο παραμέτρων $\mathcal{W}$, είναι σε θέση να προσεγγίσει συναρτήσεις του συνόλου $\mathcal{F}$ (εκφραστικότητα), η εκπαίδευση του δικτύου για την προσέγγιση μιας συνάρτησης f μεταφράζεται στην διάσχιση του χώρου των παραμέτρων βάσει μιας μεθόδου βελτιστοποίησης, ώστε να καταλήξουμε στην πλησιέστερή της προσέγγιση $f_{\mathcal{F}}$.

Αν επιλέξουμε το σχεδιασμό ενός πιο σύνθετου δικτύου, με σύνολο παραμέτρων $\mathcal{W}'$, όπου $|\mathcal{W}'| > |\mathcal{W}|$, θα αναμέναμε η προσέγγιση $f_{\mathcal{F}'}$ της συνάρτησης f να είναι τουλάχιστον εφάμιλλη της προσέγγισης του απλούστερου δικτύου, αφού το νέο δίκτυο θα έχει ευρύτερη εκφραστικότητα $\mathcal{F}'$. Όμως, μια τέτοια συμπεριφορά δεν είναι δεδομένη, καθώς είναι πιθανό να ισχύει ότι $\mathcal{F} \not\subseteq \mathcal{F}'$. Συνεπώς, αν η βέλτιστη προσέγγιση ανήκει στο σύνολο $\mathcal{F} \setminus \mathcal{F}'$, τότε η επίδοση του σύνθετου δικτύου θα είναι χειρότερη.

Για να αποφύγουμε την πιθανότητα εμφάνισης του παραπάνω φαινομένου, θα πρέπει να εξασφαλίσουμε ότι κάθε επέκταση του δικτύου είναι ικανή να αναπαραστήσει την εκφραστικότητα του αρχικού δικτύου, δηλαδή ότι $\mathcal{F} \subseteq \mathcal{F}'$[Zha+19]. Όπως περιγράφει ο He[He+15], η παραπάνω συμπεριφορά μπορεί

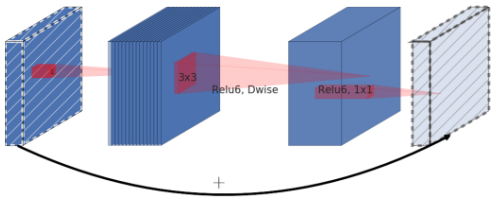
να επιτευχθεί με την προσθήκη μιας σύνδεσης καταλοίπου (residual connection) μεταξύ δύο επιπέδων του δικτύου, με το περιεχόμενο της αρχής της σύνδεσης να προστίθεται στην έξοδο του τέλους της. Απαραίτητος περιορισμός για την εφαρμογή του παραπάνω σχεδιασμού είναι η ταύτιση διαστάσεων των επιπέδων του δικτύου που βρίσκονται στην αρχή και το τέλος της σύνδεσης, ώστε να είναι εφικτή η πράξη της πρόσθεσης. Η συγκεκριμένη σχεδιαστική τεχνική καλείται μπλοκ με κατάλοιπο (residual block) και η δομή ενός ενδεικτικού μπλοκ απεικονίζεται στο σχήμα 3.4.



Σχήμα 3.4: Παράδειγμα μπλοκ με κατάλοιπο. Πηγή: [Zha+19]

3.7 Ανεστραμμένο μπλοκ με κατάλοιπο

Το ανεστραμμένο επίπεδο με κατάλοιπο (inverted residual block) αποτελεί μια τροποποίηση του μπλοκ με κατάλοιπο, λαμβάνοντας υπόψιν την ιδιότητα του επιπέδου γραμμικής συμφόρησης να περιέχει μια αναπαράσταση της χρήσιμης πληροφορίας που διασχίζει το δίκτυο. Έτσι, αντί να τοποθετούμε συνδέσεις καταλοίπου μεταξύ των συνελικτικών πράξεων του δικτύου, επιλέγουμε να τις τοποθετήσουμε μεταξύ των επιπέδων συμφόρησης που παρεμβάλλονται στις συνελικτικές πράξεις.



Σχήμα 3.5: Απεικόνιση Ανεστραμμένου επιπέδου με κατάλοιπο. Πηγή: [San+18]

Έτσι, δημιουργούμε μια συνολική αρχιτεκτονική δικτύου που στα επίπεδα συμφόρησης περιέχει τη χρήσιμη πληροφορία που διασχίζει το δίκτυο και μεταξύ δύο τέτοιων σταδιών παρεμβάλλεται η επεξεργασία τους σε υψηλότερες διαστάσεις, για την παραγωγή χαρακτηριστικών υψηλότερης τάξης. Η δομή ενός τέτοιου επιπέδου απεικονίζεται στο σχήμα 3.5.

Κεφάλαιο 4

Μηχανική Μάθηση και Όραση Υπολογιστών

4.1 Τι είναι η Όραση Υπολογιστών;

Ο κλάδος της όρασης υπολογιστών διακρίνεται για τον διεπιστημονικό του χαρακτήρα, καθώς αποτελεί ένα σύνολο τεχνικών μεθόδων που επιτρέπουν σε υπολογιστές να αναλύσουν και να ερμηνεύσουν το περιεχόμενο μιας ψηφιακής εικόνας ή ενός βίντεο. Οι βασικοί άξονες εξέλιξης του πεδίου είναι: (α) η κατανόηση και προσομοίωση της λειτουργίας φυσικών προτύπων όρασης και κυρίως αυτού της ανθρώπινης όρασης, (β) ο σχεδιασμός αυτόνομων υπολογιστικών συστημάτων που είναι σε θέση να φτάσουν ή και να ξεπεράσουν την επίδοση ενός ανθρώπου σε εργασίες σχετικές με την όραση[Hua96, σελ. 1].

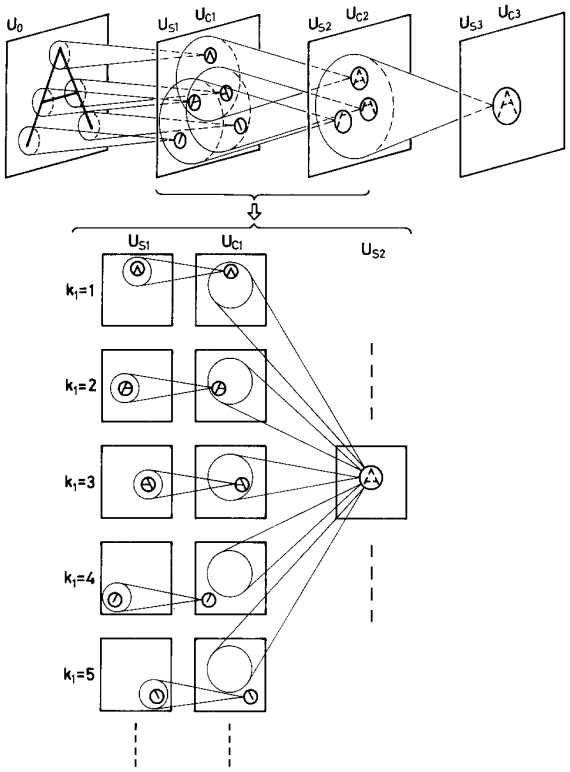
4.2 Εξέλιξη των συνελικτικών νευρωνικών δικτύων

4.2.1 Το φυσικό πρότυπο

Το πρώτο σημείο αναφοράς στο σχεδιασμό και την ανάπτυξη συνελικτικών νευρωνικών δικτύων αποτελεί η εργασία των νευροψυγολόγων David H. Hubel και Torsten Wiesel πάνω στα βιολογικά χαρακτηριστικά του μοντέλου όρασης της γάτας[Hay10]. Τα συμπεράσματα της εργασίας ανέδειξαν ότι ο οπτικός φλοιός της γάτας διακρίνεται σε απλά κύτταρα εστιασμένης ευαισθησίας, που είναι ικανά να ανιχνεύσουν απλά χαρακτηριστικά, όπως κατευθυνόμενες ακμές, αλλά και σύνθετα κύτταρα ευρύτερης ευαισθησίας, που παραμένουν ανενεργά κατά τη διέγερση με απλά χαρακτηριστικά, αλλά είναι σε θέση να ανιχνεύσουν σύνθετα χαρακτηριστικά, με μικρότερη εξάρτηση από την τοποθεσία και τη κατεύθυνσή τους[HW59].

4.2.2 Το neocognitron

Το επόμενο μεγάλο βήμα στον κλάδο ήρθε το 1980 από τον Kunihiko Fukushima, με το σχεδιασμό του μοντέλου όρασης neocognitron[Fuk80], όπως απεικονίζεται στο σχήμα 4.1.



Σχήμα 4.1: Αρχιτεκτονικό διάγραμμα neocognitron. Παράδειγμα χρήσης neocognitron 3 επιπέδων S-Cells και C-Cells, για την αναγνώριση γραμμάτων. Πηγή: [Fuk80]

Το neocogitron είναι ένα πολυεπίπεδο νευρωνικό δίκτυο, που απαρτίζεται από την εναλλαγή δύο ειδών επιπέδων:

S-Cell Τα συγκεκριμένα επίπεδα επιτελούν το ρόλο της εξαγωγής χαρακτηριστικών, όμοια με τα απλά κύτταρα που περιγράφουν οι Hubel και Wiesel. Τα συναπτικά βάρη που συνδέουν κάθε επίπεδο S-Cell με το προηγούμενο επίπεδο εισόδου προσδιορίζονται δυναμικά κατά τη διάρκεια της εκπαίδευσης του δικτύου. Σε ένα εκπαιδευμένο δίκτυο, η αναπαράσταση γνώσης στα χαμηλότερα επίπεδα[1] S-Cell ξεκινά από απλά τοπικά χαρακτηριστικά, όπως οι προσανατολισμένες ακμές, και στα υψηλότερα επίπεδα

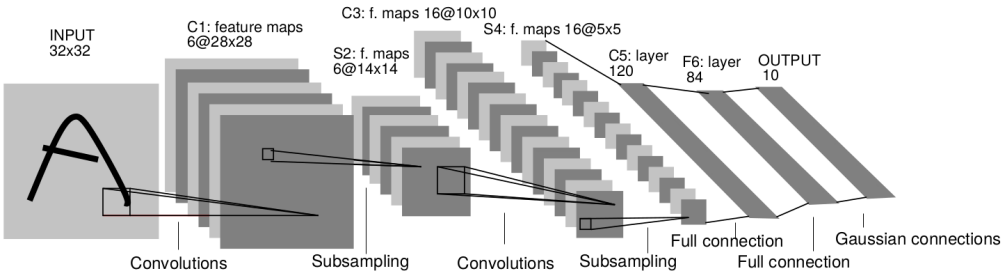
[1]Πλησιέστερα στο επίπεδο της αρχικής διέγερσης.

εξελίσσεται σε σύνθετα χαρακτηριστικά, όπως τμήματα των προτύπων εκπαίδευσης[Fuk88].

C-Cell Τα επίπεδα C-Cell διαδραματίζουν το ρόλο των σύνθετων κυττάρων του μοντέλου όρασης των Hubel και Wiesel, με σκοπό να κάνουν το μοντέλο ανθεκτικό σε χωρικές διακυμάνσεις των χαρακτηριστικών της διέγερσης. Για αυτό το σκοπό, παρεμβάλλονται μεταξύ των επιπέδων S-Cell, με κάθε C-Cell να συνδέεται μέσω σταθερών συναπτικών βαρών με μια συστάδα S-Cells του προηγούμενου επιπέδου. Κάθε C-Cell διεγείρεται αν υπάρξει διέγερση **τουλάχιστον ενός** από τα συνδεδεμένα S-Cells, συνεπώς η παραγόμενη διέγερση έχει μικρότερη ευαισθησία σε μικρές χωρικές διακυμάνσεις[Fuk88].

4.2.3 Δίκτυο LeNet-5

Το επόμενο μεγάλο βήμα ήρθε το 1989, όταν ο LeCun παρουσίασε την αρχιτεκτονική δικτύου LeNet-5[LeC98], όπως απεικονίζεται στο σχήμα 4.2, που χρησιμοποιήθηκε για την αναγνώριση στοιχείων σε χειρόγραφα έγγραφα. Οι επιτυχείς πειραματικές δοκιμές αξιολόγησης του δικτύου LeCun στην αναγνώριση εικόνων μεμονωμένων χαρακτήρων ήταν η πρώτη ισχυρή ένδειξη ότι ήταν εφικτή η εκπαίδευση ενός συνελικτικού νευρωνικού δικτύου, χρησιμοποιώντας τον αλγόριθμο του back-propagation[Zha+19].

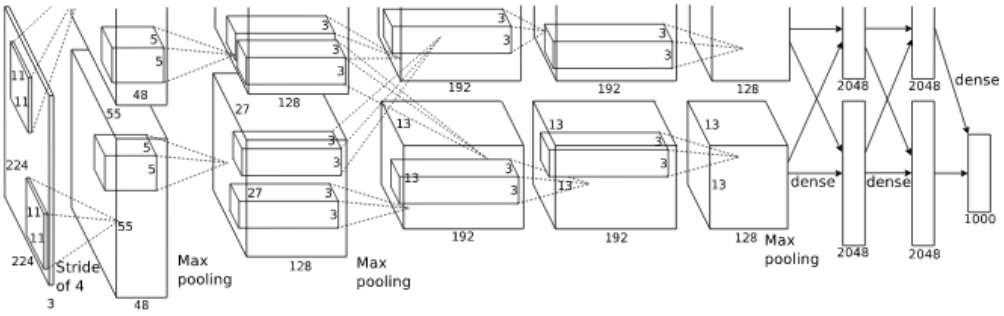


Σχήμα 4.2: Αρχιτεκτονικό διάγραμμα δικτύου LeNet-5. Παράδειγμα αναγνώρισης χαρακτήρα με 10 πιθανές κλάσεις εξόδου. Πηγή: [LeC98]

Το δίκτυο απαρτίζεται από δύο ζεύγη επιπέδων συνέλιξης και συγκέντρωσης, που λειτουργούν ως εξαγωγέας χαρακτηριστικών της εικόνας, και τριών πλήρως συνδεδεμένων επιπέδων, που μετατρέπουν τα χαρακτηριστικά σε προβλέψεις ταξινόμησης για το ψηφίο της εισόδου. Η αξιολόγηση σε δείγματα ASCII χαρακτήρων και στο δείγμα χειρόγραφων ψηφίων MNIST το ανέδειξαν σε κορυφαία μέθοδο για την εποχή του και δημιούργησαν έντονο ερευνητικό ενδιαφέρον γύρω από τις σχεδιαστικές του επιλογές.

4.2.4 Το δίκτυο AlexNet

Το δίκτυο AlexNet παρουσιάστηκε από τον Alex Krizhevsky το 2012 στα πλαίσια του διαγωνισμού όρασης υπολογιστών ImageNet, όπου και αναδείχθηκε νικητής με αξιοσημείωτη διαφορά επίδοσης από τους υπόλοιπους συμμετέχοντες.



Σχήμα 4.3: Αρχιτεκτονικό διάγραμμα AlexNet Πηγή: [KSH12]

Η αρχιτεκτονική του, όπως απεικονίζεται στο σχήμα 4.3,χρησιμοποιεί 8 επίπεδα, 5 από τα οποία είναι συνελικτικά και τα υπόλοιπα είναι επίπεδα συγκέντρωσης, για την εξαγωγή χαρακτηριστικών της εισόδου και 3 πλήρως συνδεδεμένα επίπεδα για την ταξινόμηση των παραγόμενων χαρακτηριστικών της εισόδου στις 1000 πιθανές τιμές εξόδου.

Λόγω των περιορισμών διαθέσιμης μνήμης για της κάρτες γραφικών του 2012, η εκπαίδευση του δικτύου απαιτούσε τη χρήση δύο καρτών. Κατ' επέκταση, το δίκτυο είναι χωρισμένο σε δύο μέρη, ώστε κάθε μέρος να χωράει εξολοκλήρου στη μνήμη κάθε κάρτας γραφικών, επιταχύνοντας τη διαδικασία της εκπαίδευσης.

Οι σχεδιαστικές επιλογές του δικτύου AlexNet, όπως η χρήση της συνάρτησης ενεργοποίησης Ανορθωμένης Γραμμικής Μονάδας και η χρήση πολλών επιπέδων και, αντίστοιχα, πλήθους εκπαιδεύσιμων παραμέτρων, ήταν ένα καθοριστικό βήμα στη μετάβαση από ρηχά νευρωνικά δίκτυα, όπως το LeNet-5, στη σύγχρονη αρχιτεκτονική των βαθιών νευρωνικών δικτύων[Zha+19].

4.3 Όραση σε φορητές και ενσωματωμένες συσκευές

Οι σχεδιαστικές εξελίξεις των συνελικτικών δικτύων αλλά και η επωφελής χρήση επεξεργαστών γραφικών, που επέφεραν ραγδαία αύξηση της διαθέσιμης υπολογιστικής ισχύος για την εκπαίδευση και χρήση των δικτύων [OJ04], οδηγούν στην κατασκευή δικτύων ολοένα αυξανόμενης κλίμακας και πολυπλοκότητας, με σκοπό την επίτευξη βέλτιστης ακρίβειας όρασης.

Καθώς ο σχεδιασμός τέτοιων συστημάτων στοχεύει κυρίως στη βελτιστοποίηση της ακρίβειας, παραλείπεται συχνά η επίδραση του σχεδιασμού στο μέγεθος και την ταχύτητα του δικτύου. Μια τέτοια σχεδιαστική προσέγγιση δεν είναι σε θέση να καλύψει τις απαιτήσεις συστημάτων όρασης σε συσκευές χαμηλής ισχύος, οι οποίες λειτουργούν υπό αυστηρούς περιορισμούς προσωρινής και μόνιμης μνήμης, αλλά και επεξεργαστικής ισχύος.

Συνεπώς, ο κλάδος της όρασης σε φορητές και ενσωματωμένες συσκευές ακολουθεί διαφορετική τροχιά, επικεντρώνοντας στο σχεδιασμό υπολογιστικά αποδοτικών μοντέλων όρασης μικρής κλίμακας. Οι βασικές προσεγγίσεις σχεδιασμού συνοψίζονται στη συμπίεση εκπαιδευμένων μοντέλων μεγάλης κλίμακας και την απευθείας εκπαίδευση μοντέλων μικρής κλίμακας[How+17, σελ. 1].

Στο πλαίσιο της εργασίας προτιμήθηκε η χρήση δικτύων αρχιτεκτονικής MobileNet, καθώς ο σχεδιασμός και η αξιολόγησή τους έχει γίνει με βασικό γνώμονα το **μέγεθος** και την **ταχύτητα εκτέλεσης** των μοντέλων[How+17, σελ. 1], χαρακτηριστικά που είναι απαραίτητα για την κατασκευή ενός συστήματος που θέλουμε να δουλεύει όσο δυνατόν καλύτερα σε πραγματικό χρόνο. Παράλληλα, η αξιολόγηση των επιδόσεών τους σε ένα ευρύ φάσμα εργασιών της όρασης υπολογιστών έχει αναδείξει τόσο την ικανότητά τους να παράγουν αποδοτικά μοντέλα , όσο και την υπεροχή τους σε σχέση με αντίστοιχα μοντέλα όρασης για συσκευές χαμηλής ισχύος.[How+17, σελ. 8].

Τέλος, αξίζει να σημειώσουμε ότι η επιλογή ειδικών μοντέλων MobileNet επιτρέπει, μέσω της χρήσης του πακέτου λογισμικού μηχανικής μάθησης TensorFlow, την κατασκευή μιας **ενιαίας διαδικασίας εκπαίδευσης**, με τα παραγόμενα μοντέλα να είναι σε θέση να μεταφραστούν για χρήση στην κατάλληλη συσκευή, δίχως να απαιτείται επανασχεδιασμός ή επανεκπαίδευσή τους.

4.4 Δίκτυα MobileNet

Τα δίκτυα MobileNet αποτελούν μια αρχιτεκτονική συνελικτικών δικτύων μικρού υπολογιστικού αποτυπώματος, που στοχεύουν στην κατασκευή αποδοτικών μοντέλων όρασης για φορητές και ενσωματωμένες συσκευές.

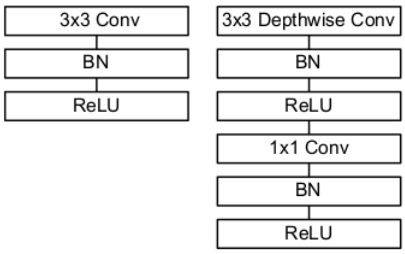
4.4.1 Περιγραφή MobileNet v1

Όπως αναφέρεται στην τεχνική περιγραφή των δικτύων MobileNets v1[How+17, σελ. 3], η υλοποίησή τους βασίζεται σε δύο δομικά συστατικά, που απεικονίζονται στο σχήμα 4.4 και περιγράφονται ως εξής:

Επίπεδο συνέλιξης Απαρτίζεται από εφαρμογή πυρήνα διαστάσεων 3×3 και

βάθους αντίστοιχου του βάθους εισόδου, που ακολουθείται από ένα επίπεδο κανονικοποίησης και στη συνέχεια τροφοδοτεί ένα επίπεδο ενεργοποίησης Ανορθωμένης Γραμμικής Μονάδας.

Επίπεδο Διαχωρίσιμης Συνέλιξης ανά κανάλι Υλοποιείται από την προσθήκη ενός επιπέδου κανονικοποίησης και ενός επιπέδου ενεργοποίησης Ανορθωμένης Γραμμικής Μονάδας, μετά από κάθε στάδιο μιας Διαχωρίσιμης ανά κανάλι Συνέλιξης με πυρήνα διαστάσεων 3×3.



Σχήμα 4.4: Δομικά στοιχεία αρχιτεκτονικής MobileNet v1. Πηγή: [How+17]

Στα πλαίσια της εργασίας, θα χρησιμοποιήσουμε δίκτυα αρχιτεκτονικής MobileNet v1 ως εξαγωγείς χαρακτηριστικών της εικόνας εισόδου, τα οποία στη συνέχεια θα τροφοδοτηθούν σε επόμενο δίκτυο για την παραγωγή εκτιμήσεων θέσης. Για αυτό το σκοπό, στον πίνακα 4.1 παραθέτουμε την περιγραφή της αρχιτεκτονικής ανά επίπεδο, για τη λειτουργία της εξαγωγής χαρακτηριστικών.

Η βασική υπερπαράμετρος των δικτύων MobileNet v1 είναι ο πολλαπλασιαστής βάθους $\alpha \in (0, 1]$, που ρυθμίζει το μέγεθος και την ταχύτητα εκτέλεσης του δικτύου, μεταβάλλοντας πολλαπλασιαστικά το βάθος των επιμέρους συνελίξεών του. Για παράδειγμα, με χρήση $\alpha = 0.5$, ο πυρήνας του πρώτου επιπέδου του δικτύου μειώνεται σε διαστάσεις $3 \times 3 \times 3 \times 16$, απαιτώντας λιγότερο χώρο για την περιγραφή του και μικρότερο υπολογιστικό κόστος για την εκτέλεσή του.

Η συνολική επίδραση του πολλαπλασιαστή βάθους α στο υπολογιστικό κόστος της διαχωρίσιμης ανά κανάλι συνέλιξης, υπολογίζεται, τροποποιώντας κατάλληλα τη σχέση 3.3.6, ως εξής:

$$H_K \cdot W_K \cdot \alpha M \cdot H_F \cdot W_F + \alpha M \cdot \alpha N \cdot H_F \cdot W_F \tag{4.4.1}$$

Επίπεδο	Διαστάσεις πυρήνα
Συνέλιξη με βήμα 2	$3 \times 3 \times 3 \times 32$
Συνέλιξη ανά κανάλι με βήμα 1	$3 \times 3 \times 32$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 32 \times 64$
Συνέλιξη ανά κανάλι με βήμα 2	$3 \times 3 \times 64$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 64 \times 128$
Συνέλιξη ανά κανάλι με βήμα 1	$3 \times 3 \times 128$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 128 \times 128$
Συνέλιξη ανά κανάλι με βήμα 2	$3 \times 3 \times 128$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 128 \times 256$
Συνέλιξη ανά κανάλι με βήμα 1	$3 \times 3 \times 256$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 256 \times 256$
Συνέλιξη ανά κανάλι με βήμα 2	$3 \times 3 \times 256$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 256 \times 512$
$5 \times$ Συνέλιξη ανά κανάλι με βήμα 1 Συνέλιξη με βήμα 1	$3 \times 3 \times 512$ ανά κανάλι $1 \times 1 \times 512 \times 512$
Συνέλιξη ανά κανάλι με βήμα 2	$3 \times 3 \times 512$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 512 \times 1024$
Συνέλιξη ανά κανάλι με βήμα 2	$3 \times 3 \times 1024$ ανά κανάλι
Συνέλιξη με βήμα 1	$1 \times 1 \times 1024 \times 1024$

Πίνακας 4.1: Αρχιτεκτονική MobileNet v1. Πηγή: [How+17]

4.4.2 Περιγραφή MobileNet v2

Όπως αναφέρεται στην τεχνική περιγραφή των δικτύων MobileNet v2[San+18], η υλοποίησή τους βασίζεται δομικό σχήμα του μπλοκ συμφόρησης με κατάλοιπο (bottleneck residual block), που περιλαμβάνει ένα επίπεδο επέκτασης, που χρησιμοποιεί συνέλιξη ανά σημείο για να επεκτείνει τα χαρακτηριστικά εισόδου σε μεγαλύτερο βάθος ανάλυσης, μια διαχωρίσιμη ανά κανάλι συνέλιξη και ένα επίπεδο γραμμικής συμφόρησης. Κάθε επιμέρους επίπεδο ακολουθείται από ένα επίπεδο κανονικοποίησης και, πλην του επιπέδου γραμμικής συμφόρησης, από ένα επίπεδο ενεργοποίησης με Ανορθωμένη Γραμμική Μονάδα.

Ένα μπλοκ συμφόρησης με κατάλοιπο, όπως απεικονίζεται στον πίνακα 4.2 και το σχήμα 4.5, ρυθμίζεται από τις ακόλουθες παραμέτρους παραμέτρους:

Επαναλήψεις n Το πλήθος επαναλήψεων του σχήματος, συνδεδεμένων σε σειρά. Για κάθε επανάληψη του σχήματος, πλήν της πρώτης, προστίθεται η δυνατότητα σύνδεσης με κατάλοιπο μεταξύ του επιπέδου επέκτασης και του επιπέδου γραμμικής συμφόρησης.

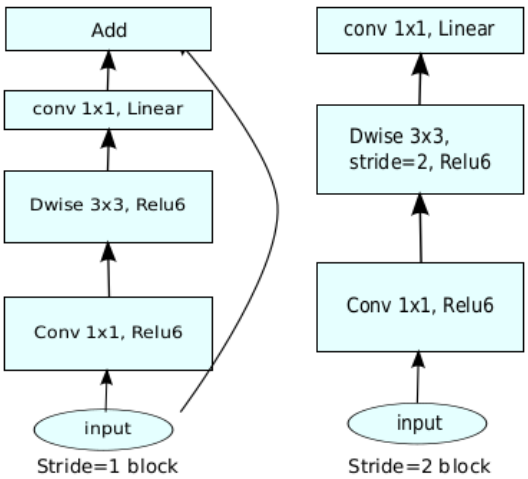
Βάθος εισόδου c Το βάθος της εξόδου της δομικής μονάδας.

Παράγοντας επέκτασης k Ο παράγοντας επέκτασης της εισόδου κατά το πρώτο στάδιο του σχήματος συμφόρησης, ώστε η διαχωρίσιμη ανά κανάλι συνέλιξη να λάβει είσοδο βάθους $k \cdot c$.

Βήμα s Το βήμα εφαρμογής της διαχωρίσιμης ανά κανάλι συνέλιξης. Σε περίπτωση που το σχήμα επαναλαμβάνεται, το βήμα έχει τιμή s για το πρώτο σχήμα και τιμή $s = 1$ για τα επόμενα. Αυτό είναι απαραίτητο για τη διατήρηση ίδιων διαστάσεων στην είσοδο και την έξοδο των μετέπειτα σχημάτων, ώστε να είναι εφικτή η εφαρμογή της σύνδεσης καταλοίπου μεταξύ διαδοχικών επιπέδων.

Είσοδος	Επίπεδο	Έξοδος
$h \times w \times c$	Συνέλιξη ανά σημείο	$h \times w \times (kc)$
$h \times w \times (kc)$	Συνέλιξη ανά κανάλι με βήμα s	$\frac{h}{s} \times \frac{w}{s} \times (kc)$
$\frac{h}{s} \times \frac{w}{s} \times (kc)$	Συνέλιξη ανά σημείο	$\frac{h}{s} \times \frac{w}{s} \times k'$

Πίνακας 4.2: Περιγραφή μπλοκ συμφόρησης με κατάλοιπο. Πηγή: [San+18]



Σχήμα 4.5: Απεικόνιση μπλοκ συμφόρησης με κατάλοιπο. Παράδειγμα block με $n > 1$ και $s = 2$. Στα δεξιά απεικονίζεται η δομή του πρώτου block, και στα δεξιά η δομή όλων των μετέπειτα blocks. Πηγή: [San+18]

Στα πλαίσια της εργασίας, θα χρησιμοποιήσουμε δίκτυα αρχιτεκτονικής MobileNet v2 ως εξαγωγείς χαρακτηριστικών της εικόνας εισόδου, τα οποία στη συνέχεια θα τροφοδοτηθούν σε επόμενο δίκτυο για την παραγωγή εκτιμήσεων θέσης. Για αυτό το σκοπό, στον πίνακα 4.3 παραθέτουμε την περιγραφή

της αρχιτεκτονικής ανά επίπεδο, για τη λειτουργία της εξαγωγής χαρακτηριστικών.

Παρόμοια με τα δίκτυα MobileNet v1, βασική υπερπαράμετρος είναι ο πολλαπλασιαστής βάθους $\alpha \in (0, 1]$, που ρυθμίζει το μέγεθος και την ταχύτητα εκτέλεσης του δικτύου, μεταβάλλοντας πολλαπλασιαστικά το βάθος των επιμέρους συνελίξεών του.

Επίπεδο	t	c	n	s
Συνέλιξη	-	32	1	2
Bottleneck	1	16	1	1
Bottleneck	6	24	2	2
Bottleneck	6	32	3	2
Bottleneck	6	64	4	2
Bottleneck	6	96	3	1
Bottleneck	6	160	3	2
Bottleneck	6	320	1	1
Συνέλιξη 1×1	-	1280	1	1

Πίνακας 4.3: Αρχιτεκτονική MobileNet v2. Πηγή: [San+18]

4.5 Δίκτυα Εντοπισμού Ενός Περάσματος

4.5.1 Εισαγωγή

Για την ολοκλήρωση της λειτουργίας ενός συνελικτικού δικτύου εντοπισμού αντικειμένων είναι αναγκαίο να μεταφράσουμε τους χάρτες χαρακτηριστικών, όπως παράγονται από τους εξαγωγείς χαρακτηριστικών που περιγράψαμε παραπάνω, σε προβλέψεις θέσης των αντικειμένων στην εικόνα, με τη χρήση ενός πλαισίου, σε μορφή ορθογώνιου παραλληλογράμμου, που περικλείει το κάθε αντικείμενο.

Η επιλογή του κατάλληλου δικτύου εντοπισμού είναι ιδιαίτερα σημαντική για την επίδοση του δικτύου. Σε σχετική έρευνα, ο Huang εξετάζει την διακύμανση των επιδόσεων ανάμεσα σε πλήθος συνδυασμών εξαγωγέων χαρακτηριστικών και δικτύων εντοπισμού, ώστε να αναδείξει το συμβιβασμό μεταξύ ακρίβειας και χρόνου εκτέλεσης για κάθε περίπτωση.[Hua+16]

Σε περιπτώσεις όπου το ζητούμενο είναι η ελαχιστοποίηση του υπολογιστικού αποτυπώματος του δικτύου, προκρίνεται ως βέλτιστη λύση ο συνδυασμός ενός εξαγωγέα χαρακτηριστικών αρχιτεκτονικής MobileNet και ενός Δικτύου Εντοπισμού Ενός Περάσματος (Single Shot Detector).[Hua+16, Ενότητα 4.1.2]

4.5.2 Περιγραφή λειτουργίας

Όπως περιγράφει ο Liu[Liu+16], βασική αρχιτεκτονική αρχή ενός δικτύου SSD είναι η πρόβλεψη της τοποθεσίας με βάση προκαθορισμένες πιθανές διατάξεις πλαισίων αναφοράς (default bounding boxes) σε κάθε σημείο των επιμέρους χαρτών χαρακτηριστικών, οι οποίες είναι ανεξάρτητες της δομής των δεδομένων εισόδου. Οι κύριες λειτουργίες ενός δικτύου SSD είναι οι εξής:

Εντοπισμός σε πολλές κλίμακες Οι προβλέψεις θέσης αντικειμένων προκύπτουν από την επεξεργασία χαρτών χαρακτηριστικών σε ένα εύρος από κλίμακες, ώστε το δίκτυο να είναι σε θέση να αναγνωρίζει χαρακτηριστικά διαφορετικής ανάλυσης και αντικείμενα διαφόρων μεγεθών. Σε περίπτωση που ο εξαγωγέας δεν παρέχει τους απαραίτητους χάρτες, τότε επεκτείνεται κατάλληλα με χάρτες χαρακτηριστικών φθίνοντος μεγέθους, όπως απεικονίζεται στο σχήμα 4.6. Σε αντίθετη περίπτωση, χρησιμοποιούνται οι χάρτες χαρακτηριστικών που παράγονται στα τελευταία επίπεδα του εξαγωγέα χαρακτηριστικών.

Πλαίσια αναφοράς Για κάθε στοιχείο των παραπάνω χαρτών χαρακτηριστικών, εξετάζονται ένα σύνολο πλαισίων αναφοράς, όπως απεικονίζονται στο σχήμα 4.7. Η παραγωγή των προβλέψεων θέσης και κατηγορίας ενός αντικειμένου γίνονται ξεχωριστά για κάθε πλαίσιο αναφοράς κάθε σημείου κάθε χάρτη χαρακτηριστικών. Η βασική κλίμακα των πλαισίων αναφοράς στον k χάρτη χαρακτηριστικών για ένα δίκτυο με m χάρτες είναι:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1), \quad k \in \{1, \cdots, m\} \tag{4.5.1}$$

όπου s_{min}, s_{max} είναι αντίστοιχα η ελάχιστη και μέγιστη κλίμακα, ως προς το μέγεθος του εκάστοτε χάρτη χαρακτηριστικών, για τα παραγόμενα πλαίσια αναφοράς.

Το σχήμα των πλαισίων αναφοράς ρυθμίζεται από το λόγο διαστάσεων a_r, που συνήθως έχει τιμές $a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$. Τότε, οι διαστάσεις κάθε πλαισίου στον k χάρτη χαρακτηριστικών υπολογίζονται ως:

$$(w_k^a, h_k^a) = \left(s_k\sqrt{a_r}, \frac{s_k}{\sqrt{a_r}}\right) \tag{4.5.2}$$

Επιπλέον, προστίθεται ένα ακόμα πλαίσιο με λόγο διαστάσεων $a_r = 1$ και βασική κλίμακα $s'_k = \sqrt{s_k^2 + 1}$.

Τα παραγόμενα πλαίσια τοποθετούνται με κατάλληλο τρόπο, ώστε το κέντρο τους να ταυτίζεται με το κέντρο του εκάστοτε σημείου του χάρτη χαρακτηριστικών.

Παραγωγή προβλέψεων με συνέλιξη Η παραγωγή προβλέψεων, για την θέση και την κατηγορία του αντικειμένου, γίνεται με την εφαρμογή συνελικτικών πυρήνων στον υπό εξέταση χάρτη χαρακτηριστικών. Ας υποθέσουμε ότι ο χάρτης έχει διαστάσεις $H_F \times W_F \times D_F$ και ότι το μοντέλο χρησιμοποιεί N_{boxes} πλαίσια αναφοράς ανά σημείο του χάρτη χαρακτηριστικών και αναγνωρίζει $N_{classes}$ διαφορετικά αντικείμενα.

Για την πρόβλεψη της κατηγορίας του αντικειμένου εφαρμόζεται σε αυτόν πυρήνας διαστάσεων:

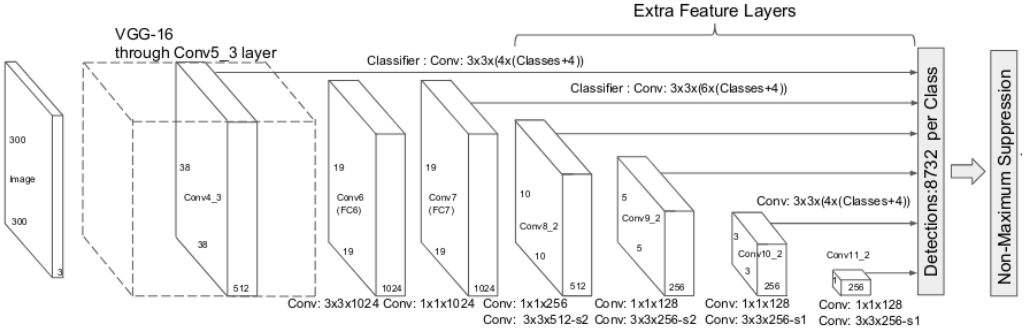
$$3 \times 3 \times D_F \times D_{class}$$
$$\text{με} \quad D_{class} = N_{boxes} \cdot (N_{classes} + 1)$$

όπου ο αριθμός των κλάσεων προσαυξάνεται κατά ένα, για να συμπεριλάβει την κλάση background.

Αντίστοιχα, για την πρόβλεψη της τοποθεσίας του αντικειμένου εφαρμόζεται σε αυτόν πυρήνας διαστάσεων:

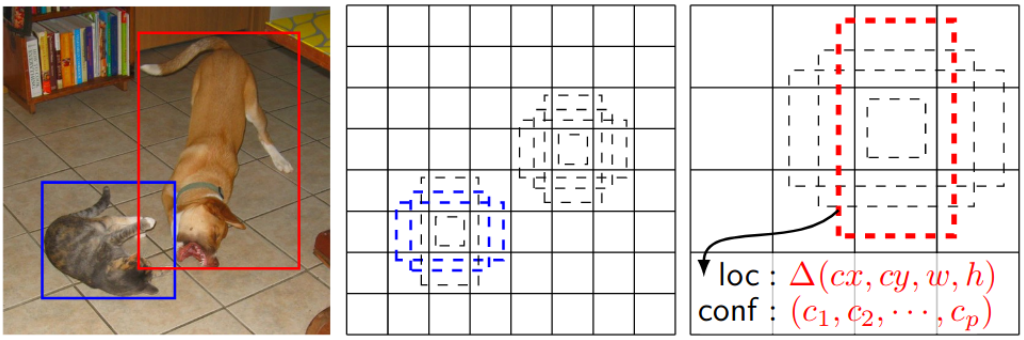
$$3 \times 3 \times D_F \times D_{loc}$$
$$\text{με} \quad D_{loc} = N_{boxes} \cdot 4$$

αφού κάθε πλαίσιο καθορίζεται από την κωδικοποιημένη κεντροειδή αναπαράσταση, όπως περιγράφεται στις σχέσεις (4.5.3), της θέσης του ως προς τη θέση του αντίστοιχου πλαισίου αναφοράς.



Σχήμα 4.6: Παράδειγμα επέκτασης χαρτών χαρακτηριστικών. Επέκταση του χάρτη χαρακτηριστικών του δικτύου VGG-16, για χρήση με δίκτυο εντοπισμού SSD. Πηγή: [Liu+16]

Τελικά, οι παραγόμενες προβλέψεις των χαρτών χαρακτηριστικών διαφόρων κλιμάκων συνενώνονται σε ένα ενιαίο επίπεδο, αποκωδικοποιούνται και αποκλιμακώνονται, ώστε να αντιστοιχούν σε πραγματικές θέσεις της αρχικής εικόνας. Καθώς το παραγόμενο πλήθος προβλέψεων είναι ιδιαίτερα μεγάλο και συχνά εμφανίζονται επικαλυπτόμενες προβλέψεις, γίνεται χρήση της Καταστολής Μη-Μεγίστων (Non-Max Suppression), ώστε ένα σύνολο επικαλυπτόμενων προβλέψεων να παράγει μόνο μια τελική πρόβλεψη.



Σχήμα 4.7: Παράδειγμα πλαισίων αναφοράς δικτύου SSD. Πηγή: [Liu+16]

4.5.3 Διαδικασία εκπαίδευσης

Κατά τη διάρκεια της εκπαίδευσης, καλούμαστε να αποφασίσουμε μια αντιστοίχιση των πλαισίων αναφοράς με τα αντικείμενα του προτύπου εισόδου. Για το σκοπό αυτό, χρησιμοποιούμε το δείκτη Jaccard, όπως περιγράφεται στην σχέση 6.2.1 ανάμεσα στο πλαίσιο και τη θέση του αντικειμένου, επιλέγοντας να τα ταιριάξουμε αν ισχύει ότι $IoU \geq 0.5$. Το πλεονέκτημα της παραπάνω διαδικασίας είναι η σχετικά απλή υλοποίηση και ικανότητα εντοπισμού υψηλής ακρίβειας για επικαλυπτόμενα πλαίσια αναφοράς.[Liu+16]

Καθώς η πλειοψηφία των παραγόμενων πλαισίων αναφοράς δεν ταιριάζουν με κάποιο πρότυπο εισόδου, δημιουργείται μια ανισορροπία μεταξύ περιπτώσεων επιτυχημένου και αποτυχημένου ταιριάσματος. Σε αυτές τις περιπτώσεις, είναι χρήσιμο να επιλεχθεί ένα υποσύνολό των πλαισίων αναφοράς, με τη μεγαλύτερη απόσταση από το πρότυπο εισόδου, ώστε ο λόγος μεταξύ αποτυχημένων και επιτυχημένων ταιριασμάτων να είναι το πολύ 3 προς 1.[Liu+16]

Ας υποθέσουμε ότι το δίκτυο έχει ταιριάξει N το πλήθος πλαίσια αναφοράς με αντικείμενα του προτύπου εισόδου, και, ειδικότερα, ότι $\mathbf{x}_{ij}^{p} \in 0, 1$ είναι ο δείκτης ταιριάσματος ανάμεσα στο πλαίσιο αναφοράς i και το αντικείμενο j της κλάσης p. Ας θεωρήσουμε επίσης τους συμβολισμούς:

- **x** οι δείκτες ταιριάσματος κάθε πλαισίου με τα αντικείμενα των προτύπων εισόδου.
- **c** οι προβλέψεις ταξινόμησης κάθε πλαισίου.
- **l** οι προβλέψεις τοποθεσίας για κάθε ταιριασμένο πλαίσιο.
- **g** οι τοποθεσίες των αντικειμένων του προτύπου εισόδου.

Η συνάρτηση κόστους που χρησιμοποιείται για την εκπαίδευση του συστήματος αποτελεί ένα γραμμικό συνδυασμό της συνάρτησης κόστους ταξινόμησης L_{class} και τη συνάρτησης κόστους εντοπισμού L_{loc}. Η συνάρτηση κόστους ταξινόμησης αποτελεί σχεδιαστική επιλογή του εκάστοτε δικτύου.

Η συνάρτηση κόστους εντοπισμού υπολογίζεται ως η Smooth L1 συνάρτηση κόστους, όπως περιγράφεται στην ενότητα 2.4.1, ανάμεσα στην προβλέ-

ψεις τοποθεσίας $\mathbf{l}$ και τις πραγματικές θέσεις $\mathbf{g}$. Οι παραπάνω περιγραφές τοποθεσίας κωδικοποιούνται κατάλληλα, με βάση το κέντρο (cx, cy) και τις διαστάσεις (w, h) του υπό μελέτη πλαισίου αναφοράς d_i. Ο μετασχηματισμός καλείται Κωδικοποίηση Κεντροειδούς Αναπαράστασης και εκτελείται ως εξής:

$$\begin{aligned} \hat{\mathbf{g}}_j^{cx} &= s_x \cdot \frac{g_j^{cx} - d_i^{cx}}{d_i^w} & \hat{\mathbf{g}}_j^{cy} &= s_y \cdot \frac{g_j^{cy} - d_i^{cy}}{d_i^h} \\ \hat{\mathbf{g}}_j^{w} &= s_w \cdot \log\left(\frac{g_j^w}{d_j^w}\right) & \hat{\mathbf{g}}_j^{h} &= s_h \cdot \log\left(\frac{g_j^h}{d_j^h}\right) \end{aligned} \tag{4.5.3}$$

όπου (s_x, s_y, s_w, s_h) οι συντελεστές κλιμάκωσης για κάθε αντίστοιχη διάσταση, με συνηθέστερες τιμές τις $(10.0, 10.0, 5.0, 5.0)$. Με αντίστοιχο τρόπο εφαρμόζεται ο μετασχηματισμός και στις προβλέψεις $\mathbf{l}$.

Ο παραπάνω μετασχηματισμός βρίσκει εφαρμογή μόνο στις περιπτώσεις που έχουμε επιτυχή εντοπισμό, δηλαδή αφορά τα πλαίσια αναφοράς του $N_{pos} = \{i \quad | \quad \exists \mathbf{x}_{ij}^p = 1\}$. Οδηγούμαστε τελικά στον υπολογισμό της συνάρτησης κόστους εντοπισμού ως εξής:

$$L_{loc}(\mathbf{x}, \mathbf{l}, \mathbf{g}) = \sum_{i \in N_{pos}} \sum_{j,p} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^p \text{smooth}_{\text{L1}}(l_i^m - \hat{\mathbf{g}}_i^m) \tag{4.5.4}$$

Τελικά, η συνάρτηση κόστους υπολογίζεται ως:

$$L(\mathbf{x}, \mathbf{c}, \mathbf{l}, \mathbf{g}) = \begin{cases} \frac{1}{N}\left(L_{class}(\mathbf{x}, \mathbf{c}) + \alpha L_{loc}(\mathbf{x}, \mathbf{l}, \mathbf{g})\right), & N \neq 0 \\ 0, & \text{διαφορετικά} \end{cases} \tag{4.5.5}$$

όπου η παράμετρος α είναι το βάρος συμμετοχής της συνάρτησης κόστους εντοπισμού στη συνολική συνάρτηση κόστους.

4.6 Κβαντισμός δικτύων

4.6.1 Εισαγωγή

Ως κβαντισμός ενός δικτύου ορίζεται η διαδικασία δημιουργίας ή μετατροπής των παραμέτρων του σε μορφή ακεραίων, από τη μορφή πραγματικών αριθμών που συναντάται συνήθως. Συνηθέστερη ακρίβεια κβαντισμού είναι τα 8 ακέραια ψηφία (κβαντισμός 8-bit) και, όπως αναφέρει ο Krishnamoorthi[Kri18], η διαδικασία μας παρέχει μια σειρά από οφέλη:

Εύρος εφαρμογής Ο κβαντισμός ενός δικτύου είναι ανεξάρτητος από το πεδίο εφαρμογής του δικτύου. Με τη χρήση κατάλληλων τεχνικών, μπορούμε να επιτύχουμε τη μετατροπή με μικρό κόστος σε ακρίβεια.

Μικρότερο μέγεθος Με κβαντισμό 8-bit, μπορούμε να επιτύχουμε μείωση του μεγέθους του μοντέλου έως και 75%, σε σχέση με το μοντέλο με παραμέτρους σε μορφή πραγματικών αριθμών 32-bit.

Μικρότερη χρήση μνήμης Το μικρότερο μέγεθος του μοντέλου επιτρέπει την καλύτερη χρήση μνήμης κατά την εκτέλεσή του.

Ταχύτητα υπολογισμών Πολλοί επεξεργαστές παρέχουν ειδικά σύνολα εντολών που επιταχύνουν την επεξεργασία δεδομένων 8-bit.

Χαμηλή κατανάλωση Το μικρότερο μέγεθος απαιτεί λιγότερες προσβάσεις στη μνήμη, γεγονός που μειώνει την κατανάλωση ισχύος.

4.6.2 Είδη κβαντιστών

Μπορούμε να ξεχωρίσουμε διάφορα είδη κβαντιστών, ανάλογα με το μετασχηματισμό που εφαρμόζεται στην μεταβλητή εισόδου. Παρακάτω παραθέτουμε τους δύο συνηθέστερους τέτοιους μετασχηματισμούς, για μια μεταβλητή $x \in \mathbf{R}$, με εύρος (x_{min}, x_{max}), που θα πρέπει να κβαντιστεί σε ένα εύρος $(0, 255)$:

Ομοιόμορφος Αφινικός Κβαντιστής Για την εφαρμογή του μετασχηματισμού θα πρέπει να καθορίσουμε δύο παραμέτρους, την κλίμακα Δ και το μηδενικό σημείο z. Η κλίμακα καθορίζει το μέγεθος του βήματος του κβαντιστή και το μηδενικό σημείο είναι ένας ακέραιος, για να αποφύγουμε σφάλματα κβάντισης. Για να μπορέσουμε να ανακτήσουμε επιτυχώς τον πραγματικό αριθμό που αντιστοιχεί σε μια κβαντισμένη μεταβλητή, θα πρέπει το εύρος της να περιέχει το 0. που αντιστοιχεί στο πραγματικό μηδενικό σημείο. Σε περίπτωση που $x_{min} > 0$, το εύρος της μεταβλητής επεκτείνεται σε $[0, x_{max})$ και, αντίστοιχα, αν $x_{max} < 0$ επεκτείνεται σε $(x_{min}, 0]$. Στη συνέχεια, η κβαντισμένη τιμή της μεταβλητής x_Q υπολογίζεται ως:

$$x_{int} = round(\frac{x}{\Delta}) + z \tag{4.6.1}$$

$$x_Q = \begin{cases} 0, & x_{int} \leq 0 \\ x_{int}, & x_{int} \in (0, 255) \\ 255, & x_{int} \geq 255 \end{cases} \tag{4.6.2}$$

Μπορούμε να ανακτήσουμε, με απώλεια, την αρχική τιμή σύμφωνα με τη σχέση:

$$x = (x_Q - z) \cdot \Delta \tag{4.6.3}$$

Ομοιόμορφος Συμμετρικός Κβαντιστής Αποτελεί μια απλοποιημένη έκδοση του Ομοιόμορφου Αφινικού Κβαντιστή, στο οποίο έχουμε πάντα $z = 0$. Η διαδικασία μετασχηματισμού μετατρέπεται σε:

$$x_{int} = round(\frac{x}{\Delta}) \tag{4.6.4}$$

$$x_{int} = round(\frac{x}{\Delta}) \tag{4.6.5}$$

και στη συνέχεια λαμβάνουμε περιπτώσεις, ανάλογα με την επιθυμία μας να αποκτήσουμε κβαντισμένη τιμή με πρόσημο ή χωρίς. Για μη-προσημασμένες τιμές υπολογίζουμε την τιμή όπως στη σχέση 4.6.2, ενώ για προσημασμένες τιμές εργαζόμαστε ως εξής:

$$x_Q = \begin{cases} 0, & x_{int} \leq -128 \\ x_{int}, & x_{int} \in (-128, 127) \\ 255, & x_{int} \geq 127 \end{cases} \tag{4.6.6}$$

Μπορούμε να ανακτήσουμε, με απώλεια, την αρχική τιμή σύμφωνα με τη σχέση:

$$x = x_Q \cdot \Delta \tag{4.6.7}$$

4.6.3 Μέθοδοι κβαντισμού

Ανάλογα με το στάδιο της εκπαίδευσης στο οποίο εφαρμόζεται ο κβαντισμός του δικτύου, διακρίνουμε τις ακόλουθες μεθόδους εφαρμογής:

Κβαντισμός μετά την εκπαίδευση Ξεκινώντας από ένα εκπαιδευμένο δίκτυο με παραμέτρους σε μορφή πραγματικών αριθμών 32-bit, εφαρμόζουμε ένα μετασχηματισμό κβαντισμού στα βάρη ή/και τις παραμέτρους ενεργοποίησης. Στην περίπτωση του κβαντισμού των παραμέτρων ενεργοποίησης, θα πρέπει να εκτελεστεί εκ νέου ένα βήμα επαλήθευσης του μοντέλου, ώστε να διαπιστωθεί το εύρος κάθε ενεργοποίησης και να υπολογιστεί κατάλληλα ο κβαντισμός της. Σε κάθε περίπτωση, η μετατροπή του δικτύου προκαλεί μια μετρήσιμη μείωση των επιδόσεων του μοντέλου.

Εκπαίδευση με υποστήριξη κβαντισμού Η εκπαίδευση του δικτύου τροποποιείται κατάλληλα με "ψεύτικους" κόμβους κβαντισμού, που βοηθούν στην αξιολόγηση της επίδοσης του κβαντισμένου μοντέλου που θα προκύψει. Ειδικότερα, η εκπαίδευση εκτελείται ως συνήθως με βάρη και παραμέτρους ενεργοποίησης σε μορφή πραγματικών αριθμών 32-bit, καθώς η αυξημένη ακρίβεια έχει ευεργετική επίδραση στην διαδικασία της εκπαίδευσης. Στη συνέχεια, έπειτα από ένα αριθμό βημάτων, τροποποιούμε

το δίκτυο, ώστε να περιγράφει τη λειτουργία ενός κβαντισμένου μοντέλου και βελτιστοποιούμε τις παραμέτρους του. Η συγκεκριμένη μέθοδος είναι και η προτεινόμενη, καθώς το παραγόμενο κβαντισμένο δίκτυο έχει αμελητέα μείωση στην επίδοσή του.

4.7 Μετατροπή δικτύων σε μορφή TFLite

Η μορφή αποθήκευσης των μοντέλων που προκύπτουν αρχικά από το στάδιο της εκπαίδευσης είναι συχνά ασύμφορη για απευθείας χρήση σε φορητές και ενσωματωμένες συσκευές, καθώς συχνά ενσωματώνουν πλεονάζουσα πληροφορία, που ήταν επωφελής κατά τη διάρκεια της εκπαίδευσης και αξιολόγησης του μοντέλου, περιέχουν τελεστές που έχουν σχεδιαστεί με γνώμονα το υλικό του περιβάλλοντος εκπαίδευσης και έχουν αρκετά μεγάλο μέγεθος αρχείων.

Καθώς οι προς μελέτη φορητές και ενσωματωμένες συσκευές δεν είναι σε θέση να υποστηρίξουν αποδοτικώς την επιτόπου εκπαίδευση των μοντέλων, εκμεταλλευόμαστε τη λειτουργία που μας παρέχει το λογισμικό του TensorFlow για τη μετατροπή των μοντέλων στην ειδική μορφή TFLite. Σε αυτή τη μορφή, γίνεται μια αφαίρεση των στοιχείων του μοντέλου που δεν σχετίζονται με την παραγωγή προβλέψεων, καθώς και μια σειρά από βελτιστοποιήσεις του υπολογιστικού κόστους της εκτέλεσής του. Συγκεκριμένα, υποστηρίζεται ένα υποσύνολο των τελεστών του TensorfFlow, το οποίο δύναται να μεταφραστεί σε ειδικούς ισοδύναμους τελεστές, που έχουν σχεδιαστεί ειδικά για χρήση σε φορητές και ενσωματωμένες συσκευές. Επιπλέον, το μοντέλο αποθηκεύεται σε δυαδική μορφή δεδομένων, ώστε να μειωθούν οι απαιτήσεις μνήμης κατά τη χρήση του.

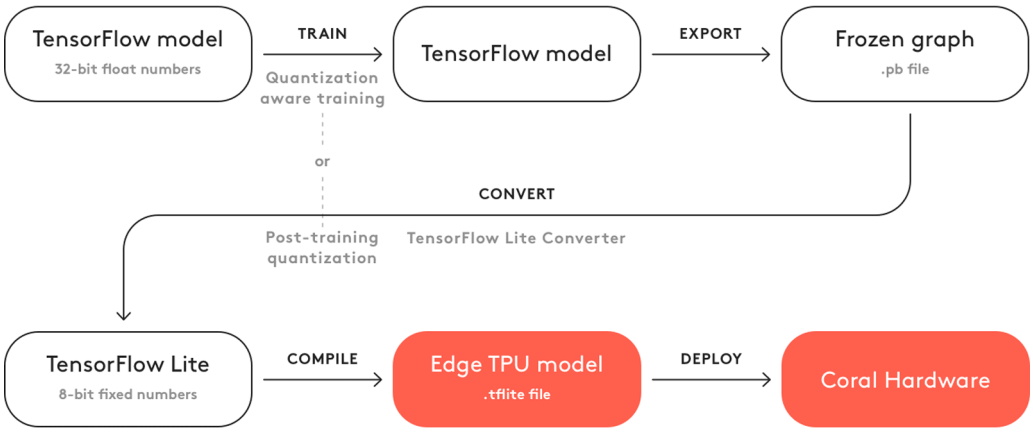
4.8 Μεταγλώττιση δικτύων σε κώδικα Coral TPU

Η αποδοτική χρήση των παραγόμενων μοντέλων σε ενσωματωμένες συσκευές χαμηλής υπολογιστικής ισχύος μας κάνει να στραφούμε στη χρήση περιφερειακών συσκευών, που εξειδικεύονται στην εκτέλεση μοντέλων με χαμηλό ενεργειακό κόστος. Η περιφερειακή συσκευή που χρησιμοποιήθηκε στην εργασία είναι το Coral TPU USB Accelerator και η διαδικασία εκπαίδευσης και μετατροπής των μοντέλων, για να χρησιμοποιηθούν σε αυτό, περιγράφεται στο σχήμα 4.8.

Ο σχεδιασμός του USB Accelerator καθορίζει σημαντικούς περιορισμούς ως προς τα μοντέλα που μπορεί να υποστηρίξει. Συνοπτικά, οι περιορισμοί είναι:

- Οι παράμετροι του μοντέλου είναι κβαντισμένες με ακρίβεια 8 bit.

- Οι διαστάσεις της εισόδου κάθε επιπέδου του μοντέλου είναι σταθερές κατά τη μεταγλώττιση.
- Οι παράμετροι του μοντέλου είναι σταθερές κατά τη μεταγλώττιση.
- Η είσοδος κάθε επιπέδου του μοντέλου είναι το πολύ τρισδιάστατη ή, σε αντίθετη περίπτωση, μόνο οι τρεις εσωτερικές διαστάσεις μπορούν να περιέχουν παραπάνω από ένα στοιχείο.
- Το μοντέλο χρησιμοποιεί το υποσύνολο των τελεστών που είναι συμβατό με το USB Accelerator[Cor].



Σχήμα 4.8: Διαδικασία εκπαίδευσης και μετατροπής μοντέλων για χρήση στο Coral TPU. Πηγή: [Cor]

Κατά τη διάρκεια της μεταγλώττισης, ο παρεχόμενος μεταγλωττιστής μετατρέπει το σύνολο των συμβατών τελεστών σε κώδικα Coral TPU, ενώ διατηρεί ως έχουν τους τελεστές που είναι ασύμβατοι. Έτσι, όταν ένα μοντέλο εκτελείται, οι συμβατοί τελεστές εκτελούνται στο Coral TPU και οι ασύμβατοι εκτελούνται στο υλικό της ενσωματωμένης συσκευής. Όπως γίνεται κατανοητό, η εκτέλεση τελεστών στο υλικό συσκευής απαιτεί σημαντικά μεγαλύτερο χρόνο εκτέλεσης συγκριτικά με το Coral TPU, επηρεάζοντας το συνολικό χρόνο εκτέλεσης του μοντέλου, οπότε είναι επωφελές να επιδιώξουμε την όσο το δυνατόν μικρότερη χρήση ασύμβατων τελεστών.

Κεφάλαιο 5

Σχεδιασμός συστήματος αναγνώριση θέσης χειρών

5.1 Δεδομένα εκπαίδευσης

5.1.1 Περιγραφή

Για την εκπαίδευση και αξιολόγηση των μοντέλων του συστήματος στραφήκαμε στο σύνολο δεδομένων EgoHands[Bam+15, σελ.3], που δημοσιεύεται και συντηρείται από το Indiana University. Η συλλογή των δεδομένων έγινε στο πλαίσιο μιας μελέτης του πανεπιστημίου για την αναγνώριση θέσης χειρών και την αναγνώριση δραστηριότητας, σε εικονοσειρές που απεικονίζουν την αλληλεπίδραση δύο ατόμων, υπό την εγωκεντρική σκοπιά του ενός.

Καθώς το σύστημα μας σχεδιάστηκε με σενάρια χρήσης που αφορούν την αλληλεπίδραση του χρήστη με την κάμερα μιας φορητής ή ενσωματωμένης συσκευής, περιμένουμε ότι τα δεδομένα εισόδου θα εκφράζουν σημαντική ομοιότητα με αυτά του συνόλου δεδομένων.

Για να εξασφαλιστεί ότι τα δεδομένα εκπαίδευσης έχουν ένα επαρκώς γενικό χαρακτήρα, το σύνολο των δεδομένων έχει παραχθεί από την καταγραφή τεσσάρων ηθοποιών σε ζεύγη, ενώ αλληλεπιδρούν εκτελώντας ένα από τέσσερα είδη δραστηριοτήτων (χαρτοπαικτική, σκάκι, επίλυση παζλ, παιχνίδι Jenga), ευρισκόμενοι σε έναν από τρεις πιθανούς χώρους (τραπέζι αίθουσας συνεδριάσεων, τραπέζι εξωτερικού χώρου, οικιακό τραπέζι). Οι παραπάνω διαφοροποιήσεις επαρκούν για τη δημιουργία 48 μοναδικών συνδυασμών για τα παραγόμενα βίντεο[Bam+15, σελ.3].

Τα δεδομένα εκπαίδευσης είναι βίντεο διάρκειας 90 δευτερολέπτων, ανάλυσης 720X1280 και συχνότητας 30 Hz. Καθώς οι ανάγκες της εκπαίδευσης απαιτούν τη χρήση μεμονωμένων εικόνων, επιλέχθηκαν τυχαία 100 στιγμιό-

Σχήμα 5.1: Παραδείγματα δεδομένων εισόδου με μάσκες διαχωρισμού για κάθε χέρι. Πηγή: [Bam+15]

τυπα από κάθε βίντεο και έγινε καταγραφή των επιμέρους εικονοστοιχείων που ανήκουν στο δεξί ή αριστερό χέρι του χρήστη που καταγράφει το βίντεο ή του χρήστη που αλληλεπιδρά. Τα εικονοστοιχεία που καταγράφηκαν αφορούν κάθε τμήμα του χεριού μέχρι τον καρπό, που ήταν ορατό στην εικόνα, ακόμα και αν καλυπτόταν μερικώς από κάποιο αντικείμενο ή ήταν αποκομμένο από το όριο της εικόνας[Bam+15, σελ.3]. Όπως μπορεί να γίνει εμφανές από τις εικόνες στο σχήμα 5.1, η παραπάνω καταγραφή οδηγεί σε ένα λεπτομερή καθορισμό της θέσης των χεριών σε κάθε εικόνα, μέσω των κατάλληλων μασκών διαχωρισμού (segmentation masks).

Συνολικά, τα δεδομένα εκπαίδευσης αποτελούνται από 4800 καταγεγραμμένες εικόνες, που περιέχουν 15053 καταγραφές χειρών. Για τον χρήστη με την εγωκεντρική σκοπία, το αριστερό του χέρι εμφανίζεται στο 53,3% των περιπτώσεων και το δεξί στο 71%. Αυτή η διαφορά μπορεί να αποδοθεί στη προτίμηση χρήσης του επιδέξιου χεριού του χρήστη, καθώς και οι τέσσερις ηθοποιοί ήταν δεξιόχειρες. Για τον χρήστη που αλληλεπιδρά, το αριστερό χέρι εμφανίζεται το αριστερό και δεξί χέρι εμφανίζονται στο 95,2% και 94,0% των περιπτώσεων, αντίστοιχα.[Bam+15, σελ.3]

5.1.2 Επεξεργασία

Για τη διαδικασία εκπαίδευσης του συστήματος της εργασίας μας, απαιτείται διαφορετική μεταχείριση των δεδομένων εισόδου, ώστε να αποκτήσουν την απλούστερη μορφή του περιβάλλοντος πλαισίου για κάθε χέρι, δηλαδή ενός ορθογωνίου παραλληλογράμμου που περικλείει το σύνολο του χεριού. Κατ' επέκταση, ήταν αναγκαία η μετάφραση της αναπαράστασης των χεριών στην κατάλληλη μορφή. Παράλληλα, καθώς το μοντέλο μας δεν διακρίνει ανάμεσα σε δεξί ή αριστερό χέρι, αλλά ούτε ως προς την ταυτότητα του χρήστη, φρο-

ντίζουμε ώστε τα δεδομένα να έχουν μια και μοναδική κλάση, αυτή του απλού χεριού. Καθώς το σύστημά μας έχει γενικό χαρακτήρα και δεν θα πρέπει να επηρεάζεται από το επιδέξιο χέρι του εκάστοτε χρήστη, χρειάζεται να λάβουμε μέτρα για να αποφύγουμε την ενδεχόμενη επιρροή της ασυμμετρίας της εμφάνισης μεταξύ του δεξιού και του αριστερού χεριού του χρήστη. Συγκεκριμένα, φροντίζουμε να χρησιμοποιήσουμε κατάλληλες μεθόδους επαύξησης των δεδομένων κατά την εκπαίδευση και ειδικότερα τη μέθοδο του κατοπτρισμού της εικόνας. Στα πλαίσια της εργασίας, η λήψη και προεπεξεργασία του συνόλου δεδομένων υλοποιείται από τον κώδικα της ενότητας Α.5.1.

Επιπλέον, για την αποδοτικότερη εκτέλεση της διαδικασίας της εκπαίδευσης, απαιτείται η μετατροπή των επιμέρους φωτογραφιών και των επισημάνσεών τους σε ένα ενιαίο σειριοποιημένο αρχείο μορφότυπου TFRecord, ενός τύπος αποθήκευσης δεδομένων σε δυαδική μορφή. Η παροχή των δεδομένων στη συγκεκριμένη μορφή έχει ευεργετικές ιδιότητες στη χρήση προσωρινής και μόνιμης μνήμης αλλά και στη ταχύτητα ανάγνωσης των δεδομένων, μειώνοντας σημαντικά τον χρόνο εκτέλεσης που απαιτούν οι σχετικές διαδικασίες [Gam18]. Στα πλαίσια της εργασίας, η μετατροπή των δεδομένων σε TFRecord υλοποιείται από τον κώδικα της ενότητας Α.5.2.

5.2 Περιγραφή μοντέλων

Για την εκπόνηση της εργασίας, επιλέχθηκαν ως εξαγωγείς χαρακτηριστικών τα μοντέλα MobileNet v1 και v2, με συναρτήσεις ενεργοποίησης των συνελίξεων μια τροποποιημένη μορφή της Ανορθωμένης Γραμμικής Μονάδας, που έχει την τιμή 6 ως άνω φράγμα των τιμών εξόδου.

Η αρχιτεκτονική τους επαυξάνεται με ένα δίκτυο Αναγνώρισης Μιας Λήψης, το οποίο φροντίζει να μετατρέψει τα παραγόμενα χαρακτηριστικά σε πιθανές θέσεις των χειρών, στη μορφή ορθογωνίων παραλληλόγραμμων. Στην περίπτωση και των δύο δικτύων, χρησιμοποιήθηκαν δύο χάρτες χαρακτηριστικών τους και στη συνέχεια το κάθε δίκτυο επεκτάθηκε με 4 επιπλέον χάρτες φθίνουσας κλίμακας, όπως περιγράφεται στον πίνακα 5.1.

Για το MobileNet v1, οι 2 χάρτες χαρακτηριστικών του δικτύου που χρησιμοποιήθηκαν είναι ο χάρτης στην έξοδο της ομάδας 5 διαδοχικών συνελίξεων και ο χάρτης που παράγεται στο τελευταίο επίπεδο του δικτύου, όπως αυτό περιγράφεται στον πίνακα 4.1.

Για το MobileNet v2, οι 2 χάρτες χαρακτηριστικών του δικτύου που χρησιμοποιήθηκαν είναι ο χάρτης εξόδου της συνέλιξης ανά σημείο στην πρώτη επανάληψη του Bottleneck επιπέδου με $c = 160$ και ο χάρτης που παράγεται στο τελευταίο επίπεδο του δικτύου, όπως αυτό περιγράφεται στον πίνακα 4.3.

Για την βέλτιστη αρχικοποίηση των παραμέτρων κάθε μοντέλου χρησιμο-

MobileNet v1	MobileNet v2	Βήμα	Έξοδος	SSD
$1 \times 1 \times 1024 \times 256$	$1 \times 1 \times 1280 \times 256$	1	$10 \times 10 \times 256$	
$3 \times 3 \times 256 \times 512$		2	$5 \times 5 \times 512$	✓
$1 \times 1 \times 512 \times 128$		1	$5 \times 5 \times 128$	
$3 \times 3 \times 128 \times 256$		2	$3 \times 3 \times 256$	✓
$1 \times 1 \times 256 \times 128$		1	$3 \times 3 \times 128$	
$3 \times 3 \times 128 \times 256$		2	$2 \times 2 \times 256$	✓
$1 \times 1 \times 256 \times 64$		1	$2 \times 2 \times 64$	
$3 \times 3 \times 64 \times 128$		2	$1 \times 1 \times 128$	✓

Πίνακας 5.1: Περιγραφή επέκτασης χαρτών χαρακτηριστικών Mobilenet v1 και v2. Παραθέτουμε τις διαστάσεις του πυρήνα και το βήμα κάθε συνέλιξης που εφαρμόζεται. Έπειτα, παραθέτουμε τις διαστάσεις της εξόδου κάθε επιπέδου και επισημαίνουμε τους χάρτες χαρακτηριστικών που τροφοδοτούν το δίκτυο SSD.

ποιήθηκαν τα αντίστοιχα προεκπαιδευμένα checkpoints των μοντέλων, όπως αυτά παρέχονται στο αποθετήριο μοντέλων TensorFlow, στη σελίδα https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md. Με αυτό τον τρόπο, θέλουμε να εκμεταλλευτούμε την ικανότητα που έχουν αποκτήσει τα προεκπαιδευμένα μοντέλα στην αναγνώριση χαρακτηριστικών στο σύνολο δεδομένων COCO, που έχει έναν αρκετά γενικό χαρακτήρα. Μοναδική εξαίρεση αποτελεί το μοντέλο MobileNet v2 με $\alpha = 0.75$, καθώς δεν παρέχεται το κατάλληλο προεκπαιδευμένο checkpoint.

Στο στάδιο της εκπαίδευσης επιλέξαμε τη χρήση μάθησης σε παρτίδες, με παρτίδες μεγέθους $|\mathcal{B}| = 10$, ώστε να εκμεταλλευτούμε πλήρως την διαθέσιμη υπολογιστική ισχύ του περιβάλλοντος εκπαίδευσης, χωρίς να δημιουργηθούν ζητήματα έλλειψης μνήμης. Η συνολική διάρκεια της εκπαίδευσης καθορίστηκε στα 50.000 βήματα. Για τα μοντέλα MobileNet v1 επιλέξαμε τη χρήση βελτιστοποίησης με ορμή, ενώ για τα μοντέλα MobileNet v2 επιλέξαμε τη χρήση του αλγόριθμου βελτιστοποίησης RMSprop.

Στο στάδιο της επαλήθευσης, επιλέξαμε τη χρήση 500 προτύπων επαλήθευσης. Στα τελευταία 2.000 βήματα της εκπαίδευσης γίνεται η αξιολόγηση και εκπαίδευση της λειτουργίας του δικτύου του δικτύου σε κβαντισμένη μορφή, ακρίβειας 8-bit.

Η αναλυτική περιγραφή του συνόλου των παραμέτρων των επιλεγμένων μοντέλων παρατίθεται σε μορφή παραμετρικού αρχείου στο παράρτημα Β, ενώ ο κώδικας που υλοποιεί την εκπαίδευση των μοντέλων περιγράφεται στο παράρτημα Α.3.1. Μετά το πέρας της εκπαίδευσης κάθε μοντέλου, παράγονται τα σχετικά μοντέλα για χρήση σε φορητές και ενσωματωμένες συσκευές, όπως περιγράφεται στο παράρτημα Α.3.2.

Κεφάλαιο 6

Αξιολόγηση επιδόσεων

6.1 Υλικό αξιολόγησης

Στους πίνακες 6.1 και 6.2 παρουσιάζουμε το σύνολο των φορητών και ενσωματωμένων συσκευών, αντίστοιχα, που χρησιμοποιήθηκαν για την αξιολόγηση των επιδόσεων χρήσης του συστήματος. Για κάθε συσκευή παραθέτουμε τα δομικά αρχιτεκτονικά στοιχεία, την έκδοση του λειτουργικού συστήματος, αλλά και ένα σύντομο αναγνωριστικό (Tag), για την ευκολότερη αναπαράστασή της στα πειραματικά αποτελέσματα.

Μοντέλο	SoC	CPU	RAM	Android	Tag
Samsung Galaxy A50	Exynos 9610	4-core 2.3 GHz Cortex-A73 & 4-core 1.7 GHz Cortex-A53	4 GB	9.0	SA50
Samsung Galaxy A20	Exynos 7884	2-core 1.6 GHz Cortex-A73 & 6-core 1.35 GHz Cortex-A53	3 GB	9.0	SA20
Xiaomi Mi A2 Lite	Snapdragon 625	2.0 GHz 8-core Cortex-A53	3 GB	9.0	XA2L
Samsung Galaxy Tab A 10.1 (2016)	Exynos 7870 Octa	4-core 1.6 GHz Cortex-A53 & 4-core 1.0 GHz Cortex-A53	2 GB	8.1.0	SGTA

Πίνακας 6.1: Φορητές συσκευές προς αξιολόγηση

Μοντέλο	SoC	CPU	RAM	OS	TPU	Tag
Raspberry Pi 3 Model B+	Broadcom BCM2837B0	Cortex-A53	1 GB	Rasbian	✗	RP3
Raspberry Pi 3 Model B+	Broadcom BCM2837B0	Cortex-A53	1 GB	Rasbian	✓	RP3+C

Πίνακας 6.2: Ενσωματωμένες συσκευές προς αξιολόγηση

6.2 Μετρικές αξιολόγησης

6.2.1 Αξιολόγηση εκπαίδευσης

Η αξιολόγηση του σταδίου της εκπαίδευσης αφορά τόσο την ακρίβεια του εκάστοτε μοντέλου στην παραγωγή προβλέψεων θέσης (precision), όσο και στην ικανότητά ανάκλησης επιτυχών προβλέψεων (recall).

Βασικό συστατικό των μετρικών που θα χρησιμοποιηθούν αποτελεί ο δείκτης Jaccard ή δείκτης Τομής ως προς Ένωση (Intersection over Union). Ο δείκτης υπολογίζεται βάσει της σχέσης:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{6.2.1}$$

Στο πεδίο της όρασης υπολογιστών κατά την αξιολόγηση των προβλέψεων, η παραπάνω ποσότητα μεταφράζεται ως ο λόγος μεταξύ των κοινών εικονοστοιχείων μεταξύ πρόβλεψης και δεδομένων και της ένωσης του συνόλου των εικονοστοιχείων τους.

Για τις ανάγκες τις εργασίας χρησιμοποιήθηκαν οι μετρικές του COCO API που συνοψίζονται ως εξής[Con15b]:

AP Μέση ακρίβεια πρόβλεψης, που προκύπτει από το μέσο όρο της ακρίβειας πρόβλεψης σε $IoU \in \left[.50, .55, \cdots, .95\right]$.

AP@IoU=.50 Μέση ακρίβεια πρόβλεψης σε $IoU = .50$.

AP@IoU=.75 Μέση ακρίβεια πρόβλεψης σε $IoU = .75$.

AP (small) Μέση ακρίβεια πρόβλεψης (AP) για μικρά αντικείμενα, επιφάνειας το πολύ 32^2 εικονοστοιχείων.

AP (medium) Μέση ακρίβεια πρόβλεψης (AP) για μεσαία αντικείμενα, επιφάνειας το πολύ 96^2 εικονοστοιχείων.

AP (large) Μέση ακρίβεια πρόβλεψης (AP) για μεγάλα αντικείμενα, επιφάνειας τουλάχιστον 96^2 εικονοστοιχείων.

AR@1 Μέση ανάκληση που προκύπτει από το μέσο όρο της ανάκλησης σε $IoU \in \left[.50, .55, \cdots, .95\right]$, όταν η εικόνα περιέχει το πολύ 1 αντικείμενο.

AR@10 Μέση ανάκληση που προκύπτει από το μέσο όρο της ανάκλησης σε $IoU \in \left[.50, .55, \cdots, .95\right]$, όταν η εικόνα περιέχει το πολύ 10 αντικείμενα.

AR@100 Μέση ανάκληση που προκύπτει από το μέσο όρο της ανάκλησης σε $IoU \in \left[.50, .55, \cdots, .95\right]$, όταν η εικόνα περιέχει το πολύ 100 αντικείμενα.

AR (small) Μέση ανάκληση (AR) για μικρά αντικείμενα, επιφάνειας το πολύ 32^2 εικονοστοιχείων, όταν η εικόνα περιέχει το πολύ 100 αντικείμενα.

AR (medium) Μέση ανάκληση (AR) για μικρά αντικείμενα, επιφάνειας το πολύ 96^2 εικονοστοιχείων, όταν η εικόνα περιέχει το πολύ 100 αντικείμενα.

AR (large) Μέση ανάκληση (AR) για μικρά αντικείμενα, επιφάνειας τουλάχιστον 96^2 εικονοστοιχείων, όταν η εικόνα περιέχει το πολύ 100 αντικείμενα.

Θα πρέπει να σημειώσουμε ότι η λήψη μετρήσεων μέσου όρου ακρίβειας και ανάκλησης σε πολλαπλές κλίμακες IoU και ο συνδυασμός τους σε ενιαίο μέσο όρο επιβραβεύει περισσότερο μοντέλα που πετυχαίνουν συνολικά καλύτερες επιδόσεις πρόβλεψης τοποθεσίας.

Επιπλέον, θα χρησιμοποιήσουμε την καμπύλη Ακρίβειας-Ανάκλησης για μια σειρά από τιμές του IoU, ώστε να αναπαραστήσουμε τη σχέση μεταξύ ακρίβειας και ευαισθησίας του δικτύου, για διαφορετικά όρια επικάλυψης της πρόβλεψης με την πραγματική θέση.

6.2.2 Αξιολόγηση χρήσης

Η αξιολόγηση της χρήσης των μοντέλων στις επιμέρους συσκευές αφορά το μέσο χρόνο παραγωγής προβλέψεων σε πραγματικές συνθήκες λειτουργίας, με την ανάλυση εικονοσειρών προερχόμενων από την κάμερα της εκάστοτε συσκευής και την απεικόνιση των αποτελεσμάτων στην οθόνη.

Για να λάβουμε μια αντιπροσωπευτική εικόνα της επίδοσης του μοντέλου για κάθε συσκευή, θα υπολογίσουμε το μέσο όρο παραγωγής πρόβλεψης για ένα δείγμα 100 συνεχόμενων παραγόμενων προβλέψεων. Για να αποφύγουμε την ενδεχόμενη επιβάρυνση του χρόνου εκτέλεσης κατά την εκκίνηση της αξιολόγησης, φροντίσαμε να παραλείψουμε ένα εύλογο χρονικό διάστημα στην αρχή κάθε πειράματος.

Για την αξιολόγηση στις φορητές συσκευές χρησιμοποιήθηκε η διαδικασία που περιγράφεται στο παράρτημα Α.4.1. Για την αξιολόγηση στις ενσωματωμένες συσκευές, χρησιμοποιήθηκε ο κώδικας του παραρτήματος Α.4.2.

6.3 Επιδόσεις εκπαίδευσης

Στη σελίδα 57 παρουσιάζονται οι καμπύλες Ακρίβειας-Ανάκλησης για το σύνολο των δικτύων, ενώ στις σελίδες 58-61 παρουσιάζονται συνολικά οι μετρικές κάθε μοντέλου κατά τη διάρκεια της εκπαίδευσης. Στον πίνακα 6.3 παραθέτουμε τις τελικές επιδόσεις ανά μοντέλο.

Μοντέλο	α	AP	AP@.75IoU	AR@100
MobileNet v1	1.0	0.6934	0.8386	0.7505
MobileNet v1	0.75	0.6883	0.8362	0.7451
Διαφορά		−0.0051	−0.0024	−0.0054
MobileNet v2	1.0	0.6896	0.8373	0.7492
MobileNet v2	0.75	0.6432	0.7836	0.7088
Διαφορά		−0.0464	−0.0538	−0.0403

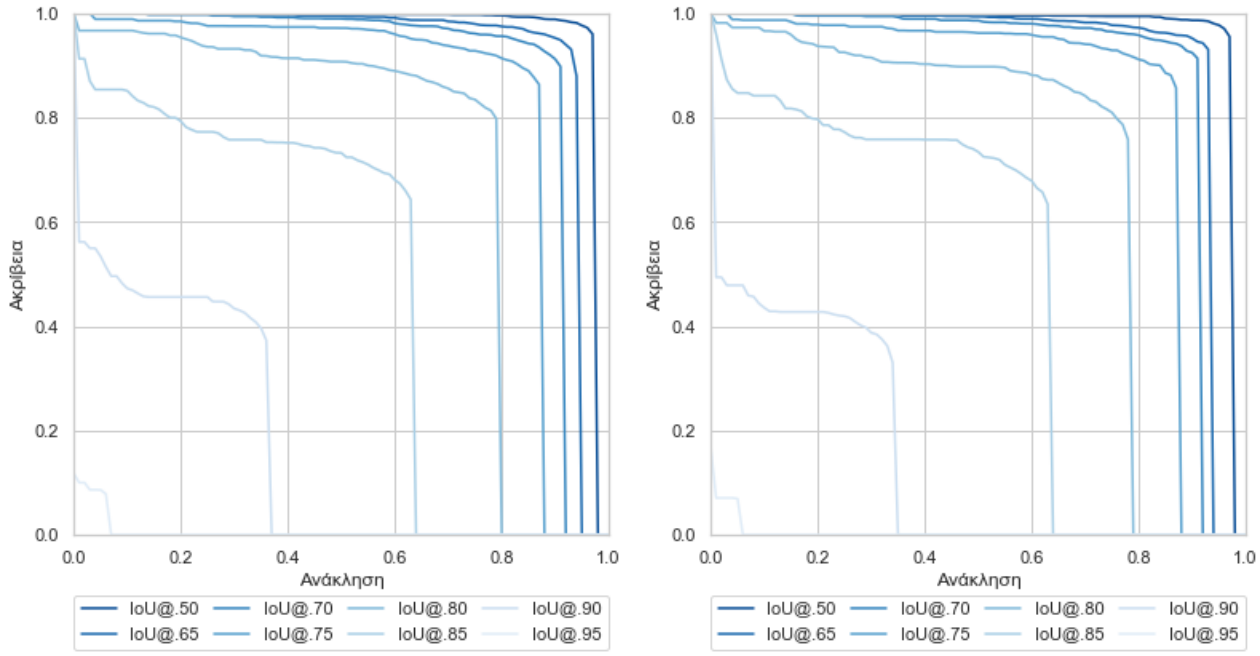
Πίνακας 6.3: Πίνακας τελικών επιδόσεων εκπαίδευσης ανά μοντέλο.

Παρατηρούμε ότι οι επιδόσεις των μοντέλων είναι εφάμιλλες αντίστοιχων μελετών πάνω στο ίδιο σύνολο δεδομένων, εμφανίζοντας ικανοποιητική ακρίβεια και ανάκληση [Dib17][Bam+15][Jun18]. Το σύνολο των μοντέλων εμφανίζει καλή ικανότητα στην παραγωγή προβλέψεων που επικαλύπτονται κατά τουλάχιστον 75% με την πραγματική θέση των χεριών, γεγονός που συνεπάγεται την συνολικά πετυχημένη εκπαίδευση των μοντέλων.[Hua+16]

Οι επιδόσεις είναι ιδιαίτερα καλές στην αναγνώριση χεριών μεγάλου και μεσαίου μεγέθους (ως προς τα εικονοστοιχεία που τα απαρτίζουν), εμφανίζουν όμως χαμηλές τιμές στην αναγνώριση χεριών μικρού μεγέθους, κάτι που μπορεί να αποδοθεί στην εγγενή αδυναμία δικτύων με δίκτυα εντοπισμού αρχιτεκτονικής SSD να αναγνωρίσουν μικρά αντικείμενα.[Hua+16]

Οι επιδόσεις του μοντέλου MobileNet v2 με $\alpha = 0.75$ χρήζουν ειδικής αναφοράς, καθώς αποτέλεσε το μοναδικό μοντέλο για το οποίο δεν έγινε αρχικοποίηση των παραμέτρων του με βάση ένα προεκπαιδευμένο μοντέλο. Όπως αναδεικνύουν τα σχήματα 6.1, 6.8 και 6.9, το μοντέλο εμφανίζει μια υστέρηση 8000 βημάτων εκπαίδευσης στη σύγκλισή του και οι επιδόσεις του είναι συγκριτικά χαμηλότερες σε σχέση με τα υπόλοιπα μοντέλα υπό εξέταση.

Αν αναλογιστούμε την επίδραση του πολλαπλασιαστή βάθους α στις επιδόσεις του εκάστοτε μοντέλου, όπως παρουσιάζεται στον πίνακα 6.3, μπορούμε να δούμε ότι στην περίπτωση του δικτύου MobileNet v1 η επίδραση είναι αμελητέα, της τάξεως του 0.5%, άρα σε εφαρμογές με αυστηρούς περιορισμούς μνήμης και υπολογιστικού κόστους η χρήση του μοντέλου με $\alpha = 0.75$ είναι προτιμότερη. Η περίπτωση του MobileNet v2 είναι διαφορετική, καθώς όπως είδαμε το μοντέλο μειωμένου μεγέθους έχει διαφορετική αφετηρία εκπαίδευσης, παρουσιάζοντας μια απώλεια της τάξεως του 5% στις τελικές επιδόσεις.



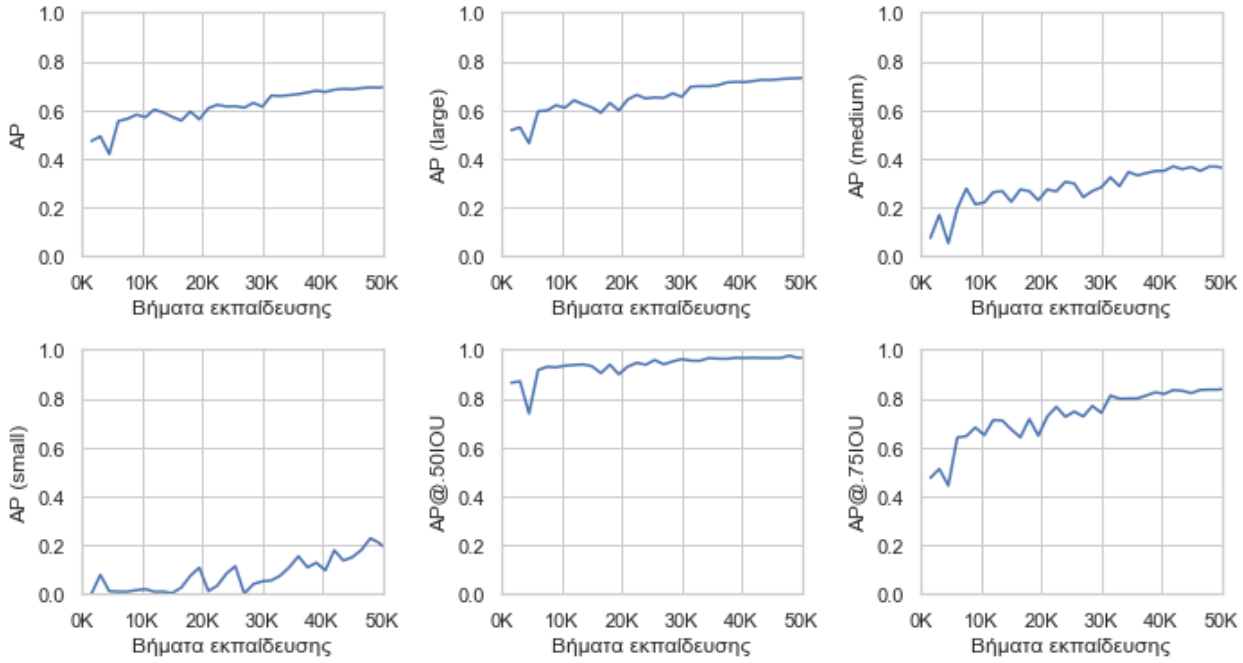
(a) MobileNet v1 με $\alpha = 1.0$.

(b) MobileNet v1 με $\alpha = 0.75$.

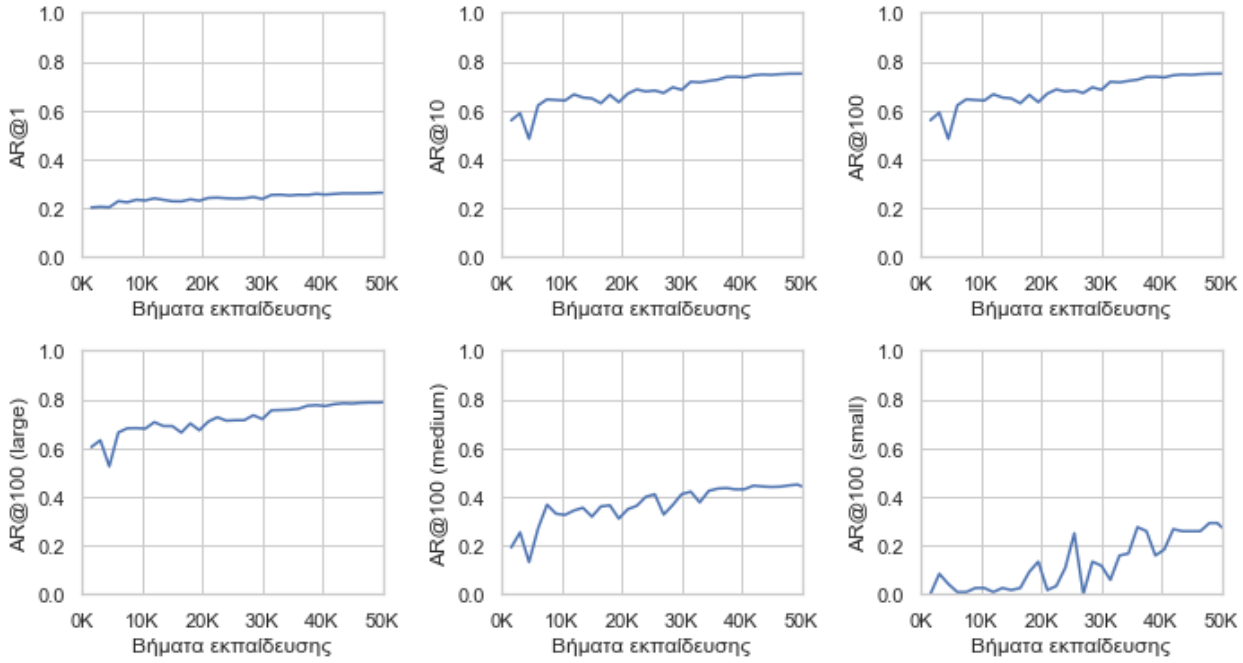
(c) MobileNet v με $\alpha = 1.0$.

(d) MobileNet v με $\alpha = 0.75$.

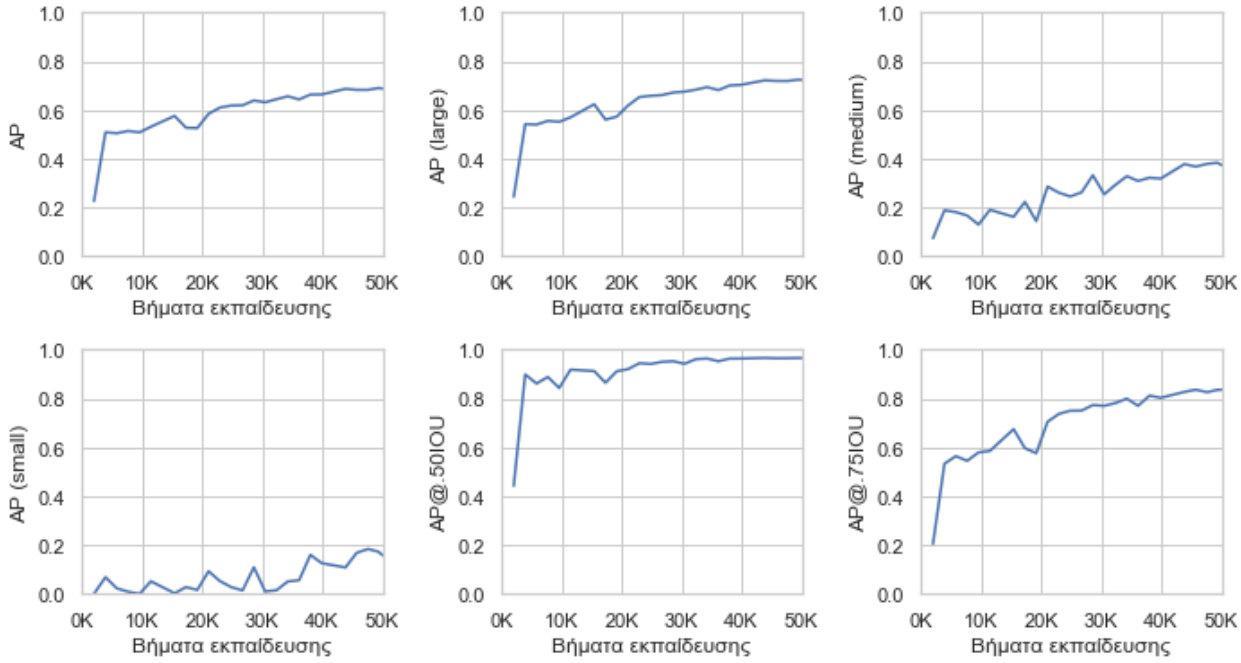
Σχήμα 6.1: Καμπύλες Ακρίβειας-Ανάκλησης ως προς τον ελάχιστο βαθμό επικάλυψης.



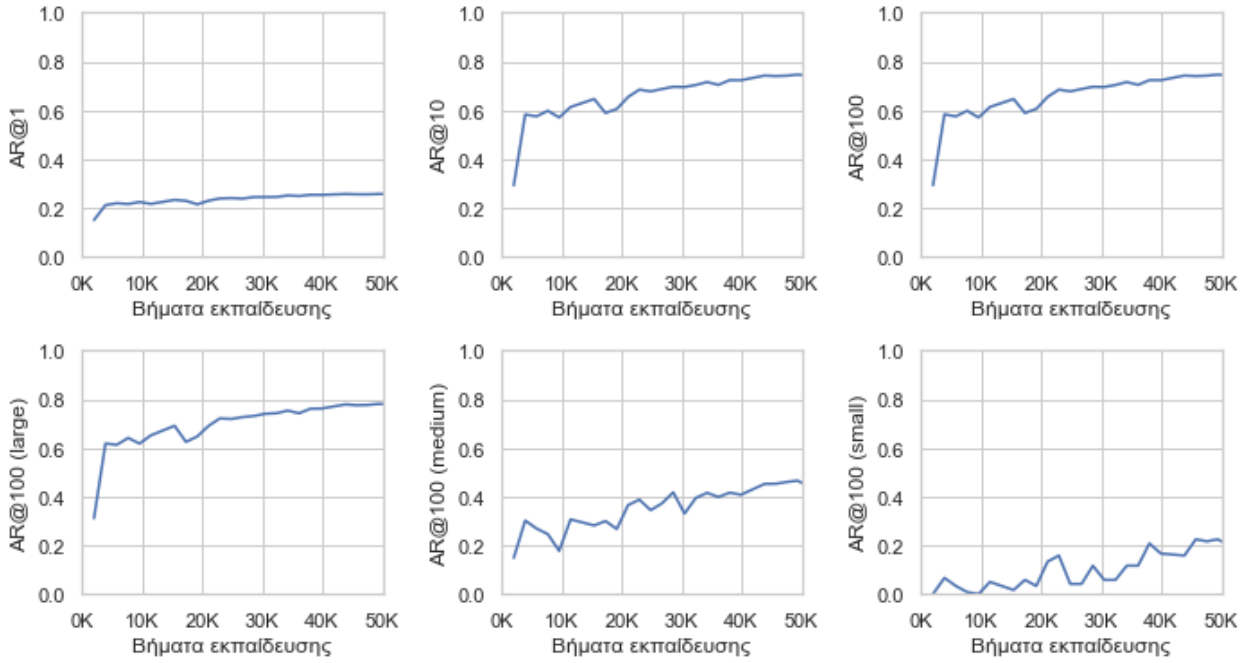
Σχήμα 6.2: Μετρικές ακρίβειας για μοντέλο MobileNet v1 με $\alpha = 1.0$



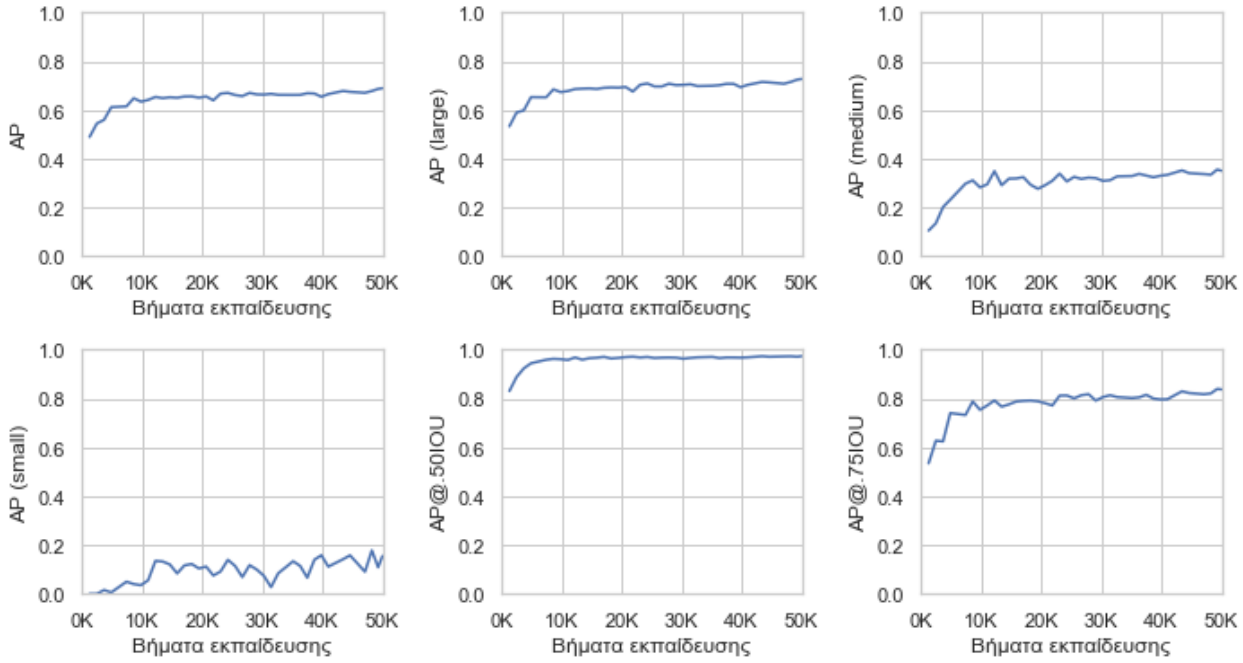
Σχήμα 6.3: Μετρικές ανάκλησης για μοντέλο MobileNet v1 με $\alpha = 1.0$



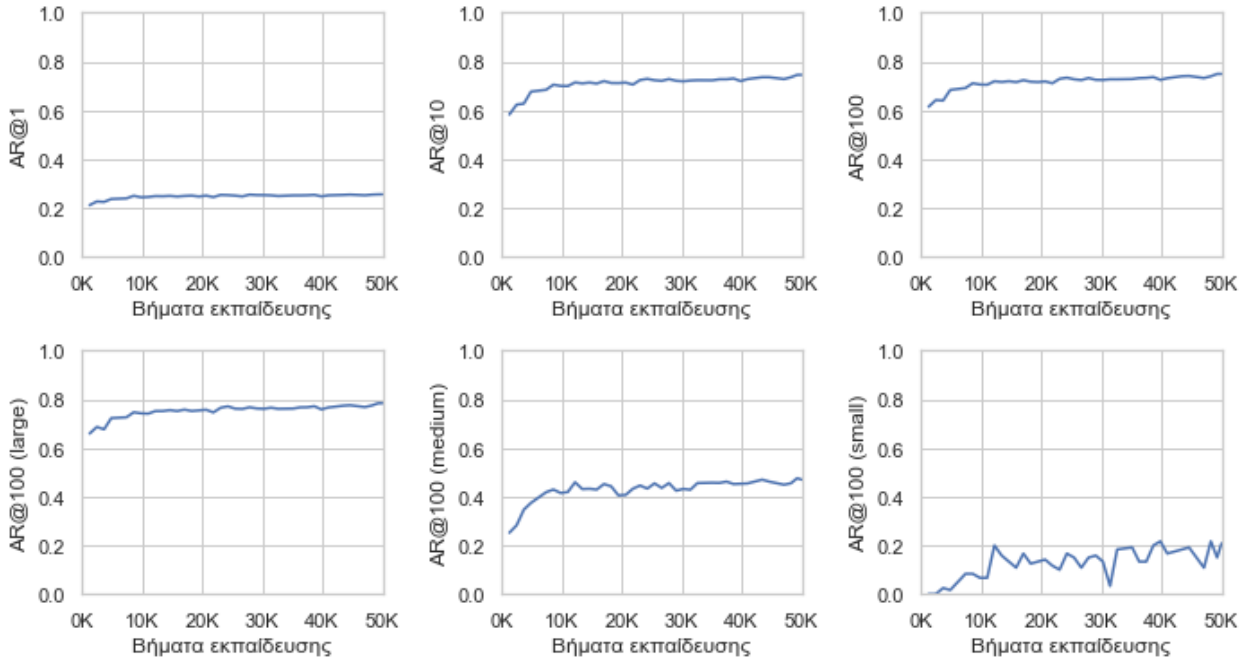
Σχήμα 6.4: Μετρικές ακρίβειας για μοντέλο MobileNet v1 με $\alpha = 0.75$



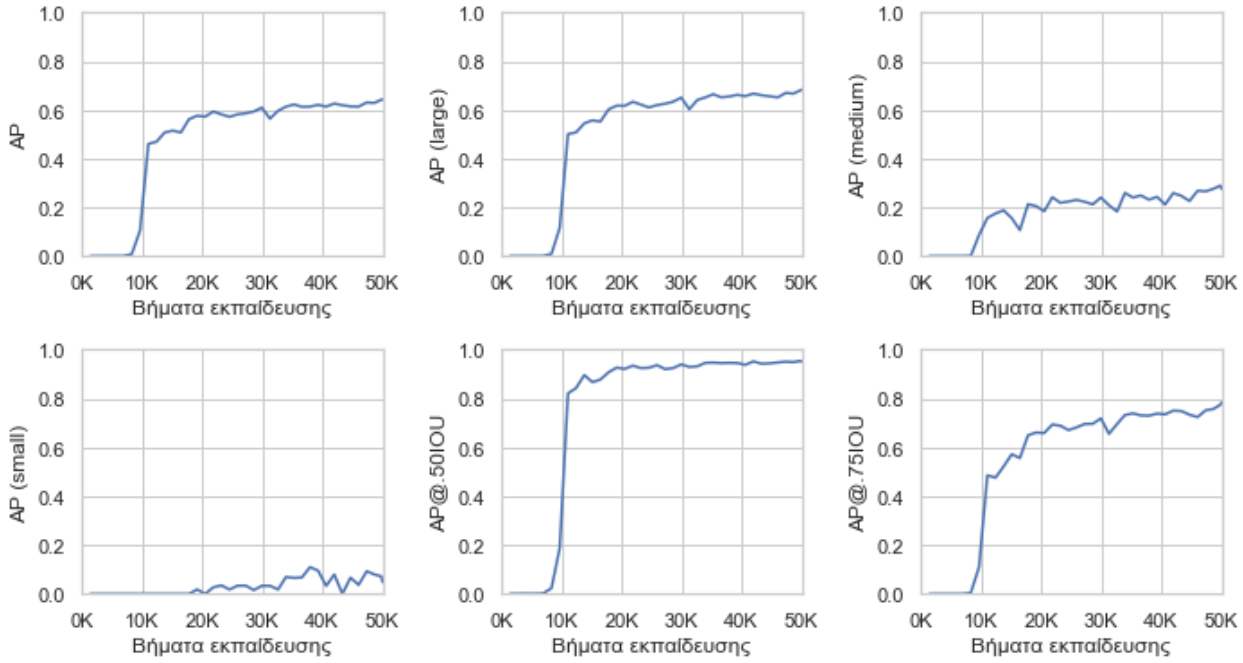
Σχήμα 6.5: Μετρικές ανάκλησης για μοντέλο MobileNet v1 με $\alpha = 0.75$



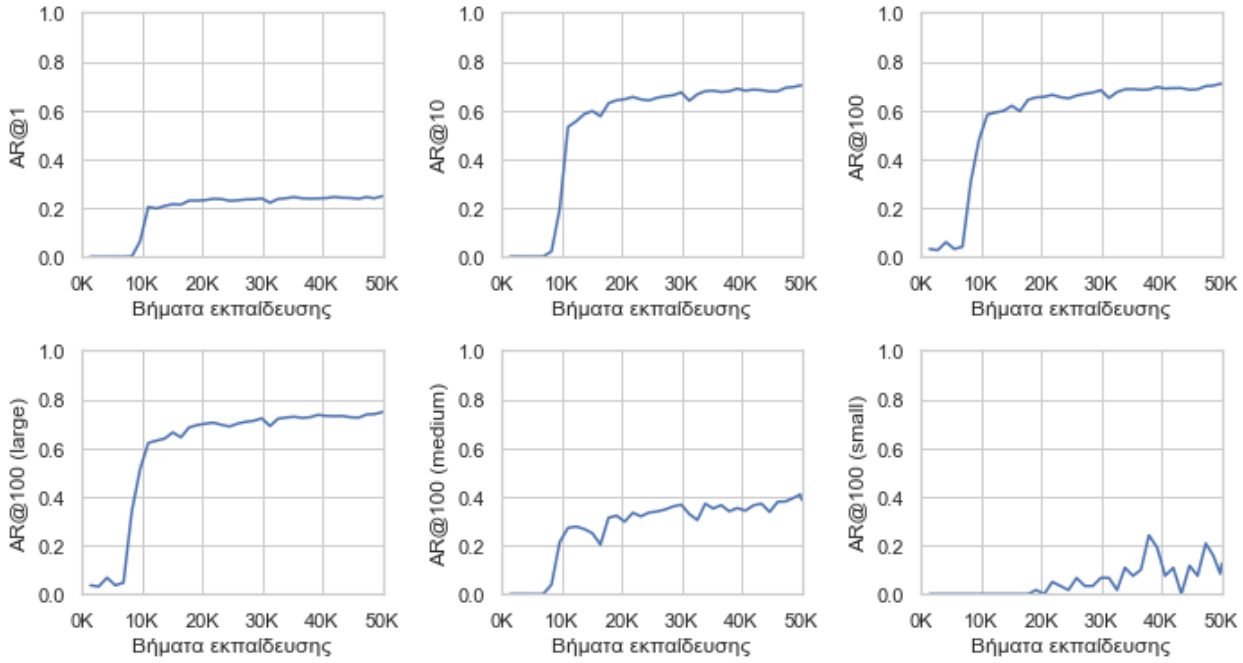
Σχήμα 6.6: Μετρικές ανάκλησης για μοντέλο MobileNet v2 με $\alpha = 1.0$



Σχήμα 6.7: Μετρικές ανάκλησης για μοντέλο MobileNet v2 με $\alpha = 1.0$



Σχήμα 6.8: Μετρικές ακρίβειας για μοντέλο MobileNet v2 με $\alpha = 0.75$

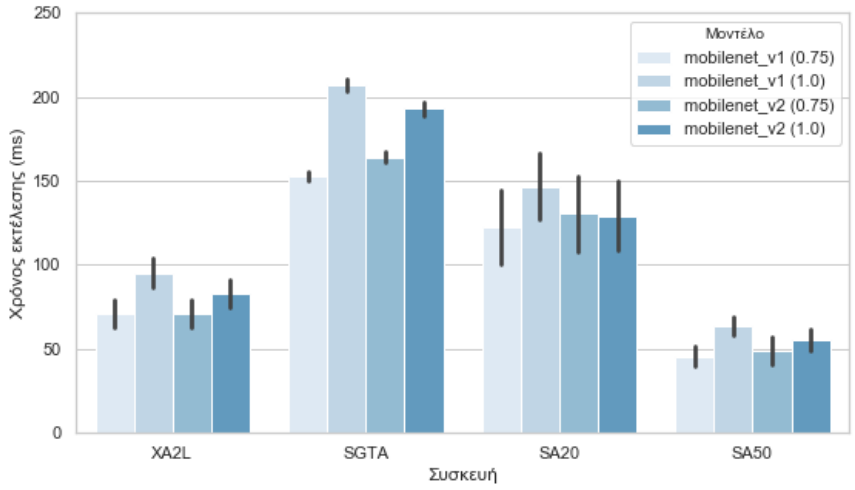


Σχήμα 6.9: Μετρικές ανάκλησης για μοντέλο MobileNet v2 με $\alpha = 0.75$

6.4 Επιδόσεις χρήσης

6.4.1 Φορητές συσκευές

Στο σχήμα 6.10 απεικονίζεται η επίδοση των υπό εξέταση φορητών συσκευών για κάθε μοντέλο. Ο πίνακας απεικονίζει το μέσο όρο και τη διασπορά του χρόνου εκτέλεσης για 100 εκτελέσεις του κάθε μοντέλου.



Σχήμα 6.10: Συγκριτική επίδοση φορητών συσκευών ανά μοντέλο.

Παρατηρούμε ότι οι συσκευές με τουλάχιστον 4 ισχυρούς πυρήνες επεξεργαστή (XA2L, SGTA, SA50) παρουσιάζουν μικρή διακύμανση στο χρόνο εκτέλεσης του μοντέλου, σε αντίθεση με τη συσκευή SA20, που μόλις 2 από τους 8 διαθέσιμους πυρήνες είναι ισχυροί. Εικάζουμε ότι η συγκεκριμένη συμπεριφορά οφείλεται στην εναλλαγή χρήσης μεταξύ ισχυρών και αδύναμων επεξεργαστικών πυρήνων.

Στους πίνακες 6.4 και 6.5 παρουσιάζουμε την επίδραση του πολλαπλασιαστή βάθους α στο χρόνο εκτέλεσης των μοντέλων MobileNet v1 και v2, αντίστοιχα. Με εξαίρεση τη συσκευή SA20, που αναλύθηκε παραπάνω, παρατηρούμε ότι τα μοντέλα με $a = 0.75$ ήταν ταχύτερα κατά 25% σε αρχιτεκτονικές MobileNet v1 και κατά 14% σε αρχιτεκτονικές MobileNet v2. Αυτή η απόκλιση οφείλεται στην χρήση επιπλέον δομικών μπλοκ στα δίκτυα MobileNet v2, τα οποία δεν επηρεάζονται αντίστοιχα από τη μείωση του βάθους.

Συνδυαστικά με τα αποτελέσματα αξιολόγησης της εκπαίδευσης, στην ενότητα 6.3, σημειώνουμε ότι το μοντέλο MobileNet v1 με $\alpha = 0.75$ παρουσιάζει εξαιρετικές επιδόσεις κατά την εκπαίδευση και έχει το μικρότερο υπολογιστικό κόστος στο εύρος όλων των συσκευών, άρα προτείνεται ως ιδανική αρχιτεκτονική για την υλοποίηση του συστήματος σε φορητές συσκευές.

Συσκευή	$\alpha = 1.0$ (ms)	$\alpha = 0.75$ (ms)	Μείωση (%)
SA20	146.66	122.36	16.57
SA50	63.20	45.32	28.29
SGTA	207.31	152.62	26.38
XA2L	95.10	71.11	25.23

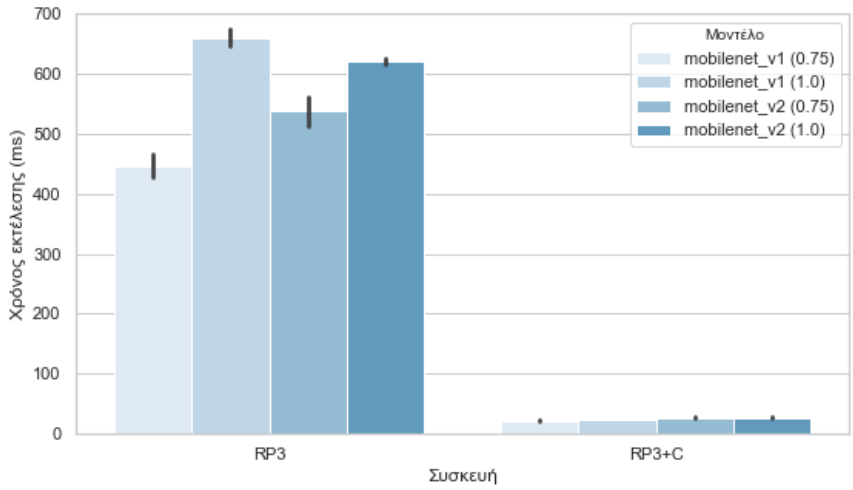
Πίνακας 6.4: MobileNet v1: Μείωση χρόνου εκτέλεσης ως προς τον πολλαπλασιαστή βάθους α.

Συσκευή	$\alpha = 1.0$ (ms)	$\alpha = 0.75$ (ms)	Μείωση (%)
SA20	129.04	130.16	−0.87
SA50	55.27	48.51	12.23
SGTA	192.91	164.02	14.98
XA2L	82.71	70.77	14.44

Πίνακας 6.5: MobileNet v2: Μείωση χρόνου εκτέλεσης ως προς τον πολλαπλασιαστή βάθους α.

6.4.2 Ενσωματωμένες συσκευές

Στο σχήμα 6.11 απεικονίζεται η επίδοση του Raspberry Pi για κάθε μοντέλο, ανάλογα με τη χρήση Coral TPU Accelerator. Ο πίνακας απεικονίζει το μέσο όρο και τη διασπορά του χρόνου εκτέλεσης για 100 εκτελέσεις του κάθε μοντέλου. Είναι εμφανής η διαφορά της κλίμακας του χρόνου εκτέλεσης για τις δύο περιπτώσεις, αφού η χρήση του Accelerator μειώνει περίπου 20 φορές τον απαιτούμενο χρόνο.



Σχήμα 6.11: Συγκριτική επίδοση ενσωματωμένων συσκευών ανά μοντέλο.

Η παραγωγή προβλέψεων ανά περίπου 500 ms σε ένα απλό Raspberry Pi καθιστά ασύμφορη τη χρήση του για εργασίες σε πραγματικό χρόνο, αφού οποιαδήποτε εργασία επεξεργασίας του αποτελέσματος επιδρά προσθετικά στο συνολικό χρόνο εκτέλεσης, οδηγώντας σε μεγάλη υστέρηση της απόκρισης του συστήματος.

Αντίθετα, ο χαμηλός χρόνος εκτέλεσης, της τάξεως των 25 ms, σε ένα Raspberry Pi που αναθέτει τους υπολογισμούς του μοντέλου σε ένα Accelerator είναι απόλυτα βιώσιμος για εφαρμογές σε πραγματικό χρόνο, αφού η υστέρηση που συνεισφέρει είναι αμελητέα. Παρατηρούμε επίσης ότι ο χρόνος εκτέλεσης είναι πρακτικά ίδιος για το σύνολο των μοντέλων που εκτελέστηκαν στο Accelerator, οπότε το μοντέλο MobileNet v1 με $\alpha = 1.0$ αποτελεί την ιδανική αρχιτεκτονική για την υλοποίηση του συστήματος σε ενσωματωμένες συσκευές.

6.5 Συμπεράσματα

Ο σχεδιασμός του συστήματος οδήγησε στην παραγωγή μοντέλων με αρκετά ικανοποιητικές επιδόσεις τόσο στις απαιτήσεις της εκπαίδευσης όσο και στη βιωσιμότητα της χρήσης τους σε φορητές και ενσωματωμένες συσκευές, ως προς το μέγεθος και τον χρόνο εκτέλεσης.

Για το σύνολο των συσκευών, θα πρέπει να σημειώσουμε ότι κατά τη δοκιμή τους σε πραγματικές συνθήκες παρατηρήθηκε περιστασιακά μια αξιοσημείωτη απόσταση από το επιθυμητό αποτέλεσμα του εντοπισμού της θέσης χειρών. Υπήρξαν περιπτώσεις που είχαμε παντελή έλλειψη εντοπισμών ή λανθασμένους εντοπισμούς άλλων αντικειμένων, ενώ η εικόνα εισόδου περιείχε με ξεκάθαρο τρόπο εμφανίσεις χειρών.

Καθώς οι επιδόσεις των δικτύων κατά την εκπαίδευση ήταν ικανοποιητικές, μπορούμε να αποδώσουμε τις αστοχίες χρήσης στο γενικό πλαίσιο σε δύο κατευθύνσεις:

Μη-αντιπροσωπευτικά δεδομένα Καθώς τα δεδομένα περιλαμβάνουν πρότυπα εκπαίδευσης προερχόμενα μόνο από τέσσερα άτομα και τρεις χώρους, είναι πιθανό να αντιπροσωπεύουν μόνο ένα μικρό δείγμα των εμφανίσεων χειρών σε τυχαία περιβάλλοντα. Συνεπώς, τα υπό εκπαίδευση μοντέλα δεν είχαν στη διάθεσή τους την απαραίτητη ποικιλία προτύπων εισόδου, ώστε να αποκτήσουν ένα γενικό χαρακτήρα.

Ποιοτικά χαρακτηριστικά εικόνας εισόδου Τα χαρακτηριστικά της εικόνας εισόδου εξαρτώνται σε μεγάλο βαθμό από την κάμερα που παρέχει η κάθε συσκευή, αλλά και το λογισμικό της. Έτσι, τα δεδομένα εισόδου που παράγουν οι διάφορες συσκευές είναι εν γένει διαφορετικά, αφού καθεμία έχει διαφορετική μέθοδο ρύθμισης για την εστίαση, το φωτισμό, την αντίθεση και άλλα ποιοτικά χαρακτηριστικά της εισόδου.

6.6 Μελλοντικό έργο

Με την εκπόνηση της παρούσας εργασίας θελήσαμε να παρουσιάσουμε την διαδικασία κατασκευής και αξιολόγησης ενός συστήματος αναγνώρισης θέσης χειρών σε φορητές και ενσωματωμένες συσκευές, δίνοντας παράλληλα μια λεπτομερή ανάλυση των επιμέρους αρχιτεκτονικών του συστατικών.

Σκοπός μας είναι να παρέχουμε μια αρχιτεκτονική αφετηρία για την εκπόνηση αντίστοιχων συστημάτων όρασης, παράλληλα με μια εκτίμηση των αναμενόμενων επιδόσεων, τόσο στο στάδιο της εκπαίδευσης, όσο και στο στάδιο της εφαρμογής σε κοινές συσκευές.

Στο πλαίσιο της μελλοντικής εργασίας πάνω στο πεδίο, διακρίνουμε τις ακόλουθες κατευθύνσεις ανάπτυξης, που μπορούν να συνεισφέρουν, μεμονωμένα ή συνδυαστικά, στην βελτίωση των παραγόμενων συστημάτων και την επέκταση των εφαρμογών τους:

Χρήση σε ευρύτερο σύστημα Τα παραγόμενα μοντέλα όρασης δύνανται να χρησιμοποιηθούν ως μέλη ενός ευρύτερου συστήματος, το οποίο μπορεί είτε να τα χρησιμοποιήσει ως μια επιπλέον διεπαφή του συστήματος με το χρήστη ή να αποτελέσει μέρος ενός συστήματος επεξεργασίας εικόνας.

Χρήση διαφορετικών αρχιτεκτονικών Καθώς η Μηχανική Μάθηση και η Όραση Υπολογιστών είναι δύο από τους πλέον ραγδαία εξελισσόμενους επιστημονικούς τομείς, υπάρχει και αναπτύσσεται πλήθος αρχιτεκτονικών συνδυασμών για την υλοποίηση ενός συστήματος. Ενδεικτικά, στα τελευταία στάδια της εκπόνησης της παρούσας εργασίας υπήρξε πλήθος νέων διαθέσιμων μοντέλων από την ομάδα του Coral TPU, ενώ παράλληλα διατέθηκε και η $3^{η}$ έκδοση της αρχιτεκτονικής MobileNet. Συνεπώς, θα είχε αξία η μελέτη και αξιολόγηση περαιτέρω αρχιτεκτονικών συνδυασμών για την αναγνώριση της θέσης χειρών, ώστε να εντοπιστεί η κορυφαία διαθέσιμη μέθοδος.

Ανάλυση φορητών συσκευών Με τη χρήση σημαντικά μεγαλύτερου πλήθους φορητών συσκευών για την αξιολόγηση του συστήματος, θα ήταν εφικτό να αναλυθεί η επίδραση των επιμέρους αρχιτεκτονικών στοιχείων και του λογισμικού τους στον τελικό χρόνο εκτέλεσης των μοντέλων. Μια τέτοια μελέτη θα διευκόλυνε σημαντικά την αξιολόγηση σεναρίων χρήσης των μοντέλων σε φορητές συσκευές.

Εφαρμογή μεθόδου σε άλλα δεδομένα Καθώς η κατασκευή των επιμέρους μοντέλων δεν κάνει ιδιαίτερες παραδοχές για το είδος του προβλήματος όρασης που καλείται να λύση, είναι εφικτή η χρήση τμημάτων της

εργασίας για την ανάπτυξη συστημάτων αναγνώρισης άλλων δεδομένων εισόδου σε παρόμοια σενάρια χρήσης (φορητές ή/και ενσωματωμένες συσκευές). Σε τέτοια περίπτωση, η εργασία παραθέτει τον ενδεικτικό τρόπο επεξεργασίας των δεδομένων εισόδου, όπως περιγράφεται στην ενότητα Α.5.

Συλλογή δεδομένων Όπως είδαμε παραπάνω, είναι ιδιαίτερα σημαντικό να έχουμε διαθέσιμα δεδομένα που είναι όσο το δυνατόν αντιπροσωπευτικά του δείγματος των χειρών σε τυχαίο περιβάλλον. Συνεπώς, θα ήταν επωφελής η συλλογή ενός συνόλου δεδομένων που απεικονίζει τις θέσεις χειρών για ένα μεγάλο πληθυσμό ατόμων, που εκτελούν ποικιλόμορφες δραστηριότητες σε εικόνες με μεγάλο εύρος διακυμάνσεων φωτισμού και ποιότητας εικόνας. Ένα τέτοιο σύνολο δεδομένων θα πρέπει να περιέχει χέρια σε διάφορες κλίμακες και με μεγάλο εύρος πλήθους απεικονίσεων ανά εικόνα.

Βιβλιογραφικές αναφορές

[Bam+15] Sven Bambach κ.ά. «Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions». Στο: *The IEEE International Conference on Computer Vision (ICCV)*. Δεκ. 2015.

[Ben18] Eli Bendersky. *Depthwise separable convolutions for machine learning*. 2018. URL: https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/ (επίσκεψη 23/10/2019).

[Bro19] BroadbandSearch.net. *Mobile Vs. Desktop Usage (Latest 2019 Data)*. 2019. URL: https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics (επίσκεψη 19/10/2019).

[Con15a] COCO Consortium. *COCO API*. 2015. URL: https://github.com/cocodataset/cocoapi (επίσκεψη 10/07/2019).

[Con15b] COCO Consortium. *Detection Evaluation*. 2015. URL: http://cocodataset.org/#detection-eval (επίσκεψη 23/10/2019).

[Cor] Coral. *TensorFlow models on the Edge TPU*. URL: https://coral.withgoogle.com/docs/edgetpu/models-intro/ (επίσκεψη 10/10/2019).

[Dib17] Victor Dibia. *Real-time Hand Tracking Using SSD on Tensorflow*. https://github.com/victordibia/handtracking. 2017.

[Fou04] Apache Software Foundation. *Apache License, Version 2.0*. Ιαν. 2004. URL: https://www.apache.org/licenses/LICENSE-2.0.

[Fuk80] Kunihiko Fukushima. «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position». Στο: *Biological Cybernetics* 36 (1980), σσ. 193–202.

[Fuk88] Kunihiko Fukushima. «Neocognitron: A hierarchical neural network capable of visual pattern recognition». Στο: *Neural Networks* 1 (1988), σσ. 119–130.

[Gam18] Thomas Gamauf. *Tensorflow Records? What they are and how to use them*. 2018. URL: https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564.

[GBC16] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Hay10] Simon Haykin. *Νευρωνικά Δίκτυα και Μηχανική Μάθηση*. 3η έκδοση. Εκδόσεις Παπασωτηρίου, 2010.

[He+15] Kaiming He κ.ά. «Deep Residual Learning for Image Recognition». Στο: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), σσ. 770–778.

[How+17] Andrew G. Howard κ.ά. «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications». Στο: *ArXiv* abs/1704.04861 (2017).

[Hua+16] Jonathan Huang κ.ά. «Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors». Στο: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), σσ. 3296–3297.

[Hua96] Thomas S. Huang. «Computer Vision : Evolution And Promise». Στο: 1996.

[HW59] David H. Hubel και Torsten N. Wiesel. «Receptive fields of single neurones in the cat's striate cortex.» Στο: *The Journal of physiology* 148 (1959), σσ. 574–91.

[Ign+18] Andrey Ignatov κ.ά. «AI Benchmark: Running Deep Neural Networks on Android Smartphones». Στο: *ECCV Workshops*. 2018.

[IS15] Sergey Ioffe και Christian Szegedy. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift». Στο: *ArXiv* abs/1502.03167 (2015).

[Jun18] JK Jung. *Hand Detection Tutorial*. https://github.com/jkjung-avt/hand-detection-tutorial. 2018.

[Kri18] Raghuraman Krishnamoorthi. «Quantizing deep convolutional networks for efficient inference: A whitepaper». Στο: *ArXiv* abs/1806.08342 (2018).

[KSH12] Alex Krizhevsky, Ilya Sutskever και Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». Στο: *Commun. ACM* 60 (2012), σσ. 84–90.

[LeC98] Yann LeCun. «Gradient-based learning applied to document recognition». Στο: 1998.

[Lin+17] Tsung-Yi Lin κ.ά. «Focal Loss for Dense Object Detection». Στο: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), σσ. 2999–3007.

[Liu+16] Wei Liu κ.ά. «SSD: Single Shot MultiBox Detector». Στο: *ECCV*. 2016.

[OJ04] Kyoung-Su Oh και Keechul Jung. «GPU implementation of neural networks». Στο: *Pattern Recognition* 37 (2004), σσ. 1311–1314.

[Ros58] Frank F. Rosenblatt. «The perceptron: a probabilistic model for information storage and organization in the brain.» Στο: *Psychological review* 65 6 (1958), σσ. 386–408.

[Rud17] Sebastian Ruder. «An overview of gradient descent optimizationalgorithms». Στο: Insight Centre for Data Analytics, NUI Galway. 2017.

[San+18] Mark Sandler κ.ά. «MobileNetV2: Inverted Residuals and Linear Bottlenecks». Στο: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), σσ. 4510–4520.

[Zha+19] Aston Zhang κ.ά. *Dive into Deep Learning*. http://www.d2l.ai. 2019.

Παράρτημα Α

Πηγιαίος κώδικας

Α.1 Περιβάλλον ανάπτυξης

Α.1.1 Χαρακτηριστικά υλικού

Στον πίνακα Α.1 παραθέτουμε το υλικό του υπολογιστή που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος.

Λειτουργική Μονάδα	Προϊόν
Κεντρικός Επεξεργαστής	AMD Ryzen 5 1600
Μνήμες RAM	2 × 8 GB DDR4-2400
Κάρτα Γραφικών	NVIDIA GTX 1060 6 GB
SSD	KINGSTON SA400S3
Μητρική	GIGABYTE AB350M-Gaming v3

Πίνακας Α.1: Υλικό του περιβάλλοντος ανάπτυξης

Α.1.2 Απαιτήσεις λογισμικού

Η εκπόνηση του υπολογιστικού μέρους της εργασίας έγινε στο περιβάλλον του λειτουργικού συστήματος `Ubuntu 18.04.3 LTS`. Στον πίνακα Α.2 παραθέτουμε τα βασικά πακέτα λογισμικού που χρησιμοποιήθηκαν, καθώς και τις αντίστοιχες εκδόσεις τους.

Λογισμικό	Έκδοση
NVIDIA GPU Driver	430.26
NVIDIA CUDA	10.0
NVIDIA cuDNN SDK	7.6.2
Edge TPU Compiler	2.0.267685300
Android Studio	3.5.1

Πίνακας Α.2: Πακέτα λογισμικού του περιβάλλοντος ανάπτυξης

Ειδικότερα, για την εκτέλεση της διαδικασίας της εκπαίδευσης και αξιολόγησης του συστήματος, εργαστήκαμε σε περιβάλλον `Python 3.7.3` και χρησιμοποιήσαμε τα ακόλουθα πακέτα λογισμικού, όπως περιγράφονται στο αρχείο `requirements.txt`:

```
contextlib2==0.5.5
Cython==0.29.11
h5py==2.9.0
jupyter==1.0.0
lxml==4.3.4
matplotlib==3.1.1
numpy==1.16.4
pandas==0.24.2
Pillow==6.1.0
pyperclip==1.7.0
requests==2.22.0
scipy==1.3.0
seaborn==0.9.0
tensorflow-gpu==1.14.0
tqdm==4.32.2
```

Επιπλέον, επικουρώντας το περιβάλλον της `Python`, χρησιμοποιήθηκαν οι εξής βοηθητικές βιβλιοθήκες, που προήλθαν αυτούσιες από τα αντίστοιχα δημόσια αποθετήρια κώδικα.

Object Detection API Το αποθετήριο κώδικα της ομάδας του TensorFlow για μοντέλα αναγνώρισης αντικειμένων, που βρίσκεται στη σελίδα https://github.com/tensorflow/models/tree/master/research/object_detection. Το αποθετήριο περιέχει βοηθητικό κώδικα για εκπαίδευση και αξιολόγηση μοντέλων, καθώς και προεκπαιδευμένα μοντέλα, για χρήση μάθησης με μεταφορά.

COCO API [Con15a] Το αποθετήριο κώδικα του COCO Consortium, στο οποίο παρέχονται οι υλοποιήσεις των μετρικών αξιολόγησης του COCO API, όπως περιγράφονται στην ενότητα 6.2.1.

Με κατάλληλη τροποποίηση του παρεχόμενου κώδικα, σύμφωνα με τον τρόπο που περιγράφεται στη σελίδα https://github.com/facebookresearch/maskrcnn-benchmark/issues/353#issuecomment-455490678, μπορούμε να αποκτήσουμε πρόσβαση σε κατάλληλα στοιχεία για την απεικόνιση της καμπύλης Ακρίβειας-Ανάκλησης.

A.1.3 Διάρθρωση αρχείων

Καθώς τα τμήματα κώδικα εξαρτώνται σε μεγάλο βαθμό από την τοποθεσία των κατάλληλων αρχείων, παρακάτω θα παραθέσουμε την διάρθρωση των αρχείων στον αρχικό φάκελο της εργασίας.

```
$HOME/Documents/Thesis/
├── Data
│   ├── egohands_label_map.pbtxt
│   ├── egohands_train.tfrecord
│   └── egohands_val.tfrecord
├── Libraries
│   ├── Tensorflow ..... Object Detection API
│   └── COCO_API
├── Models
│   ├── mobilenet_v1
│   ├── mobilenet_v1_0.75
│   ├── mobilenet_v2
│   └── mobilenet_v2_0.75
└── Scripts
    ├── tflite_converter.sh
    ├── train_script.sh
    └── make_figures.py
```

A.2 Περιβάλλον Raspberry Pi 3

A.2.1 Χαρακτηριστικά υλικού

Στον πίνακα A.3 παραθέτουμε το υλικό του ενσωματωμένου συστήματος που χρησιμοποιήθηκε για αξιολόγηση των μοντέλων.

Λειτουργική Μονάδα	Προϊόν
Ενσωματωμένο σύστημα	Raspberry Pi 3 Model B Plus Rev 1.3
Κάμερα	Raspberry Pi, Camera Board V2, 8MP
Επιταχυντής USB	Coral Edge TPU Accelerator version 1.0

Πίνακας A.3: Υλικό του ενσωματωμένου συστήματος.

Α.2.2 Απαιτήσεις λογισμικού

Το ενσωματωμένο σύστημα χρησιμοποιεί το λειτουργικό σύστημα `Raspbian GNU/Linux 10.0 (buster)`. Για την εκτέλεση της αξιολόγησης εργαστήκαμε στο περιβάλλον της `Python 3.7.3` που παρέχεται με το λειτουργικό και χρησιμοποιήσαμε τα ακόλουθα πακέτα λογισμικού, όπως περιγράφονται στο αρχείο `requirements.txt`:

```
numpy==1.16.2
tflite_runtime==1.14.0
picamera==1.13
Pillow==5.4.1
```

Για τη χρήση του Coral TPU, απαιτείται η εγκατάσταση του κατάλληλου οδηγού της συσκευής, όπως περιγράφεται στις οδηγίες εγκατάστασής του στη σελίδα https://coral.withgoogle.com/docs/accelerator/get-started/.

Η εγκατάσταση του tflite-runtime γίνεται χειροκίνητα, όπως περιγράφουν οι οδηγίες στη σελίδα του Tensorflow https://www.tensorflow.org/lite/guide/python.

Α.2.3 Διάρθρωση αρχείων

Καθώς τα τμήματα κώδικα εξαρτώνται από την τοποθεσία των κατάλληλων αρχείων, παρακάτω θα παραθέσουμε την διάρθρωση των αρχείων στον αρχικό φάκελο της εφαρμογής αξιολόγησης.

```
/home/pi/Documents/Thesis/
├── Models
│   ├── mobilenet_v1
│   ├── mobilenet_v1_0.75
│   ├── mobilenet_v2
│   └── mobilenet_v2_0.75
└── Scripts
    ├── analyse_display.py
    └── detect.py
```

A.3 Παραγωγή μοντέλων

A.3.1 Εκπαίδευση μοντέλων

Για την εκπαίδευση των μοντέλων χρησιμοποιήθηκε το `train_script.sh` , ένα script της γραμμής εντολών που εκτελείται με τον ακόλουθο τρόπο:

```
./train_script.sh <model_directory>
```

```
#!/bin/bash

# Read some arguments.
MODEL_DIR=$1
NUM_TRAIN_STEPS=50000

# Define some paths and constants.
ABSOLUTE_PATH=$HOME/Documents/Thesis/Models
PIPELINE_CONFIG_PATH=${ABSOLUTE_PATH}/${MODEL_DIR}/pipeline.config
TENSORFLOW_PATH=$HOME/Documents/Thesis/Libraries/Tensorflow/research
SAMPLE_1_OF_N_EVAL_EXAMPLES=1

# Start the training session.
python3 ${TENSORFLOW_PATH}/object_detection/model_main.py \
   --pipeline_config_path=${PIPELINE_CONFIG_PATH} \
   --model_dir=${ABSOLUTE_PATH}/${MODEL_DIR} \
   --num_train_steps=${NUM_TRAIN_STEPS} \
   --sample_1_of_n_eval_samples=${SAMPLE_1_OF_N_EVAL_EXAMPLES} \
   --alsologtostderr

# Convert directly to the TFLite format.
./tflite_converter.sh ${MODEL_DIR} ${NUM_TRAIN_STEPS}
```

A.3.2 Μετατροπή και μεταγλώττιση μοντέλων

Για την μετατροπή των μοντέλων για χρήση σε φορητές συσκευές, καθώς και για τη μεταγλώττιση των παραγόμενων μοντέλων για χρήση σε επιταχυντή Coral TPU χρησιμοποιήθηκε το `tflite_converter.sh`, ένα script της γραμμής εντολών που εκτελείται με τον ακόλουθο τρόπο:

```
./tflite_converter.sh <model_directory> <training_steps>
```

```
#!/bin/bash

# Read some arguments.
MODEL=$1
NUM_TRAIN_STEPS=$2

# Define some paths.
ABSOLUTE_PATH=$HOME/Documents/Thesis/Models
OBJECT_DETECTION_PATH=$HOME/Documents/Thesis/Libraries/Tensorflow/research/
    ↪ object_detection
OUTPUT_PATH=${ABSOLUTE_PATH}/${MODEL}/${NUM_TRAIN_STEPS}/TFLite

# Create the full output path.
mkdir -p ${OUTPUT_PATH}
echo "Storing output in ${OUTPUT_PATH}"

# Convert the initial model to a TFLite compatible graph.
python ${OBJECT_DETECTION_PATH}/export_tflite_ssd_graph.py \
   --pipeline_config_path=${ABSOLUTE_PATH}/${MODEL}/pipeline_${
       ↪ NUM_TRAIN_STEPS}.config \
   --output_directory=${OUTPUT_PATH} \
   --trained_checkpoint_prefix=${ABSOLUTE_PATH}/${MODEL}/model.ckpt-${
       ↪ NUM_TRAIN_STEPS} \
   --add_postprocessing_op=true
echo "Successfully created TFLITE .pb graph."

# Convert the above graph to the TFLite format.
tflite_convert \
   --graph_def_file=${OUTPUT_PATH}/tflite_graph.pb \
   --output_file=${OUTPUT_PATH}/detect.tflite \
   --input_shapes=1,300,300,3 \
   --input_arrays=normalized_input_image_tensor \
   --output_arrays='TFLite_Detection_PostProcess','
       ↪ TFLite_Detection_PostProcess:1','TFLite_Detection_PostProcess:2','
       ↪ TFLite_Detection_PostProcess:3' \
   --inference_type=QUANTIZED_UINT8 \
   --mean_values=128 \
   --std_dev_values=127 \
   --change_concat_input_ranges=FALSE \
   --allow_custom_ops

echo "Successfully created .tflite graph."

# Compile the TFLite graph for Coral TPU use.
edgetpu_compiler \
   --out_dir=${OUTPUT_PATH} \
   --show_operations \
   ${OUTPUT_PATH}/detect.tflite
```

A.4 Αξιολόγηση συστήματος

A.4.1 Αξιολόγηση μοντέλων σε φορητές συσκευές

Για την υλοποίηση της αξιολόγησης των μοντέλων σε φορητές συσκευές βασιστήκαμε στο παράδειγμα χρήσης TensorFlow Lite για φορητές συσκευές, που εμπεριέχεται στο αποθετήριο παραδειγμάτων TensorFlow, όπως διατίθεται στην τοποθεσία https://github.com/tensorflow/examples. Η χρήση του παραπάνω κώδικα καλύπτεται από την άδεια Apache License 2.0[Fou04].

Καθώς οι επεμβάσεις στον κώδικα είναι μικρές, συγκριτικά με τον κώδικα της εφαρμογής, κρίνεται ασύμφορο να παραθέσουμε το σύνολο του κώδικα. Αντ' αυτού, παρακάτω παραθέτουμε τις τροποποιήσεις ανά αρχείο, όπως αυτές παράγονται από την εντολή `git diff <file>`, ως προς τον αρχικό κώδικα του commit `3ea5ead84aaf2b1f49851c266fb7e0197854b2c7`. Για οικονομία χώρου, η θέση κάθε αρχείου βασίζεται στον φάκελο της θέσης:

```
lite/examples/object\_detection/android/app/src/main/java/org/tensorflow/
    ↪ lite/examples/detection
```

<detection>/LegacyCameraConnectionFragment.java

```
@@ -192,7 +192,7 @@ public class LegacyCameraConnectionFragment extends
    ↪ Fragment {
    CameraInfo ci = new CameraInfo();
    for (int i = 0; i < Camera.getNumberOfCameras(); i++) {
      Camera.getCameraInfo(i, ci);
-     if (ci.facing == CameraInfo.CAMERA_FACING_BACK) return i;
+     if (ci.facing == CameraInfo.CAMERA_FACING_FRONT) return i;
    }
    return -1; // No camera found
  }
```

<detection>/tflite/TFLiteObjectDetectionAPIModel.java

```
@@ -200,9 +200,9 @@ public class TFLiteObjectDetectionAPIModel implements
    ↪ Classifier {
      final RectF detection =
          new RectF(
              outputLocations[0][i][1] * inputSize,
-             outputLocations[0][i][0] * inputSize,
+             (1 - outputLocations[0][i][0]) * inputSize,
              outputLocations[0][i][3] * inputSize,
-             outputLocations[0][i][2] * inputSize);
+             (1 - outputLocations[0][i][2] ) * inputSize);
      // SSD Mobilenet V1 Model assumes class 0 is background class
      // in label file and class labels start from 1 to number_of_classes+1,
      // while outputClasses correspond to class index from 0 to
          ↪ number_of_classes
@@ -210,7 +210,7 @@ public class TFLiteObjectDetectionAPIModel implements
    ↪ Classifier {
      recognitions.add(
          new Recognition(
              "" + i,
-             labels.get((int) outputClasses[0][i] + labelOffset),
```

```
+           "hand",
            outputScores[0][i],
            detection));
    }
```

<detection>/DetectorActivity.java

```
@@ -27,10 +27,12 @@ import android.graphics.RectF;
 import android.graphics.Typeface;
 import android.media.ImageReader.OnImageAvailableListener;
 import android.os.SystemClock;
+import android.util.Log;
 import android.util.Size;
 import android.util.TypedValue;
 import android.widget.Toast;
 import java.io.IOException;
+import java.util.ArrayList;
 import java.util.LinkedList;
 import java.util.List;
 import org.tensorflow.lite.examples.detection.customview.OverlayView;
@@ -56,7 +58,7 @@ public class DetectorActivity extends CameraActivity
    ↪ implements OnImageAvailable
   private static final String TF_OD_API_LABELS_FILE = "file:///
       ↪ android_asset/labelmap.txt";
   private static final DetectorMode MODE = DetectorMode.TF_OD_API;
   // Minimum detection confidence to track a detection.
-  private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.5f;
+  private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.3f;
   private static final boolean MAINTAIN_ASPECT = false;
   private static final Size DESIRED_PREVIEW_SIZE = new Size(640, 480);
   private static final boolean SAVE_PREVIEW_BITMAP = false;
@@ -82,6 +84,8 @@ public class DetectorActivity extends CameraActivity
    ↪ implements OnImageAvailable

   private BorderedText borderedText;

+  private ArrayList<Long> inferenceTimes = new ArrayList<>();
+
   @Override
   public void onPreviewSizeChosen(final Size size, final int rotation) {
     final float textSizePx =
@@ -181,6 +185,11 @@ public class DetectorActivity extends CameraActivity
    ↪ implements OnImageAvailable
           final List<Classifier.Recognition> results = detector.
               ↪ recognizeImage(croppedBitmap);
           lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

+          inferenceTimes.add(lastProcessingTimeMs);
+          if(inferenceTimes.size() == 110) {
+            Log.d("INFERENCE_TIMES",TF_OD_API_MODEL_FILE + ": " +
    ↪ inferenceTimes.subList(10,110).toString());
+          };
+
           cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
           final Canvas canvas = new Canvas(cropCopyBitmap);
           final Paint paint = new Paint();
```

Η εφαρμογή που χρησιμοποιήσαμε βρίσκεται στη θέση:

`./lite/examples/object_detection/android/app`

Καθώς πρέπει να αξιολογήσουμε κάθε μοντέλο ξεχωριστά, φροντίζουμε να

μεταφέρουμε το αντίστοιχο μοντέλο `detect.tflite` στην τοποθεσία:

```
./lite/examples/object_detection/android/app/src/main/assets
```

Τα παραγόμενα αποτελέσματα εμφανίζονται στη ροή μηνυμάτων αποσφαλμάτωσης του Adroid Studio, χρησιμοποιώντας το αναγνωριστικό INFERENCE_TIMES.

A.4.2 Αξιολόγηση μοντέλων σε ενσωματωμένες συσκευές

Η εκτέλεση του κώδικα αξιολόγησης παραμετροποιείται με τον τρόπο που περιγράφεται στις παρακάτω οδηγίες χρήσης του:

```
usage: analysis_display.py [-h] --model MODEL [--no-tpu]
                           [--threshold THRESHOLD]

optional arguments:
  -h, --help            show this help message and exit
  --model MODEL         Model name. (default: None)
  --no-tpu              Don't use the Coral-TPU. (default: False)
  --threshold THRESHOLD
                        Score threshold for detected objects. (default: 0.4)
```

Επιπλέον, τόσο το αρχείο detect.py, όσο και η μέθοδος draw_object προέρχονται αυτούσιες από το παράδειγμα χρήσης Coral-TPU, στην τοποθεσία https://github.com/google-coral/tflite/tree/master/python/examples/detection, από το commit `61e54947977db7917ea35d7a8c77803651da2750`, και καλύπτεται από την άδεια χρήσης Apache License 2.0[Fou04].

```
import numpy as np
import tflite_runtime.interpreter as tflite
import time
from PIL import Image, ImageDraw
from os.path import join
from picamera import PiCamera, array
from io import BytesIO
import pickle
import argparse

import detect

# Initialize some global variables.
RESOLUTION=(640,480)
MODEL_DIR="/home/pi/Documents/Thesis/Models"
LABELS = { 0: "hand" }

# Initialize config dictionaries.
POSITIONING_OPTS = {
   "window":(500,10,RESOLUTION[0],RESOLUTION[1]),
   "fullscreen":False
}
RENDERER_OPTS = {
   "layer":5,
   "alpha":60,
}

# A hacky way of managin picamera changing preview overlays.
class OverlayManager:
   def __init__(self, camera):
      self.camera = camera
      overlay_image = Image.new('RGB', RESOLUTION)
      self.overlay = self.camera.add_overlay(
         overlay_image.tobytes(),**RENDERER_OPTS, **POSITIONING_OPTS)
      self.last_update = get_millis()
      self.empty = True

   def maybe_flush(self):
      if get_millis() - self.last_update > 500 and not self.empty:
         self.update_overlay(Image.new('RGB', RESOLUTION))
```

```
            self.empty = True


      def update_overlay(self, image):
         self.overlay.layer = 4
         temp_overlay = self.camera.add_overlay(
            image.tobytes(), **RENDERER_OPTS, **POSITIONING_OPTS)
         self.remove_overlay()
         self.overlay = temp_overlay
         self.last_update = get_millis()
         self.empty = False

      def remove_overlay(self):
         self.camera.remove_overlay(self.overlay)

   # Get time in milliseconds.
   def get_millis():
      return time.monotonic_ns() // 1000000

   # A helper for printing a running status report.
   def multiprint(text, width, end_char='\n'):
      params = [str(t).center(width) for t in text]
      print("|{}|{}|{}|{}|".format(*params), end=end_char)

   # An object to hold inference times.
   class record_statistics():
      def __init__(self):
         self.inference_times = []
         self.elements = 0

      def add_time(self, inf_time):
         if self.elements < 110:
            self.inference_times.append(inf_time)
         self.elements += 1

      def is_not_done(self):
         return self.elements < 120

      def get_stats(self):
         return self.inference_times[10:]

   # FROM GOOGLE - Start #
   def draw_objects(draw, objs, labels):
     """Draws the bounding box and label for each object."""
     for obj in objs:
       bbox = obj.bbox
       draw.rectangle([(bbox.xmin, bbox.ymin), (bbox.xmax, bbox.ymax)],
                      outline='red', width=2)
       draw.text((bbox.xmin + 10, bbox.ymin + 10),
                 '%s\n%.2f' % (labels.get(obj.id, obj.id), obj.score),
                 fill='red')
   # FROM GOOGLE - End #

   # Running the benchmark.
   def run_benchmark(args):
      MODEL=args.model
      DETECTION_THRESHOLD = args.threshold
      USE_TPU = not args.no_tpu

      TAG = "RP3+C" if USE_TPU else "RP3"
      MODEL_PATH=join(
        MODEL_DIR,
        MODEL,
        'detect_edgetpu.tflite' if USE_TPU else 'detect.tflite')
      EDGETPU_SHARED_LIB = 'libedgetpu.so.1'
```

```
print("Model:\t\t{}\nUse TPU:\t{}".format(MODEL,USE_TPU))

# Initialize the intepreter object.
interpreter = tflite.Interpreter(
   model_path=MODEL_PATH,
   experimental_delegates=[
   tflite.load_delegate(EDGETPU_SHARED_LIB, {})
])
interpreter.allocate_tensors()

stat_keeper = record_statistics()

# Access the Raspberry Pi camera.
with PiCamera() as camera:
   camera.resolution = RESOLUTION
   camera.hflip = True
   camera.start_preview(resolution=RESOLUTION, **POSITIONING_OPTS)
   overlay_manager = OverlayManager(camera)

   print("Started camera preview.")
   time.sleep(2)
   print("Started detection.\n")

   multiprint([
      "Capturing",
      "Processing",
      "Inference",
      "Detections"
   ], 12)
   print(53*"-")

   # Read input data in raw byte form for better performance.
   with array.PiRGBArray(camera) as stream:
      while stat_keeper.is_not_done():

         # Reading a frame from the camera. It is important to use the
         # video port, since proper camera captures are too costly.
         time_start = get_millis()
         camera.capture(stream, format='rgb', use_video_port=True)
         capturing_time = get_millis() - time_start

         # Process the input image.
         time_start = get_millis()
         image = Image.fromarray(stream.array)
         stream.truncate(0)
         scale = detect.set_input(
            interpreter,
            image.size,
            lambda size: image.resize(size, Image.ANTIALIAS))
         processing_time = get_millis() - time_start

         # Run inference on the image.
         time_start = get_millis()
         interpreter.invoke()
         inference_time = get_millis() - time_start
         stat_keeper.add_time(inference_time)

         # Receive detection results.
         objs = detect.get_output(
            interpreter,
            DETECTION_THRESHOLD,
            scale)

         # Print a status report.
         multiprint([
            capturing_time,
```

```
                processing_time,
                inference_time,
                len(objs)
            ], 12, '\r')

            # Draw all detections on the overlay.
            if len(objs) > 0:
                overlay_image = Image.new('RGB', (
                    RESOLUTION[0],
                    RESOLUTION[1]
                ))
                draw_objects(ImageDraw.Draw(overlay_image), objs, LABELS)
                overlay_manager.update_overlay(overlay_image)
            else:
                overlay_manager.maybe_flush()

        # Benchmark finished.
        multiprint([
              capturing_time,
              processing_time,
              inference_time,
              len(objs)
        ], 12)
        overlay_manager.remove_overlay()
        camera.stop_preview()
        with open(join('Data',"{}-{}.pkl".format(TAG,MODEL)),"wb") as file
            ↪ :
            pickle.dump(stat_keeper.get_stats(),file)

def main():
    parser = argparse.ArgumentParser(
        formatter_class=argparse.ArgumentDefaultsHelpFormatter
    )
    parser.add_argument('--model', required=True,
                        help='Model name.')
    parser.add_argument('--no-tpu', dest='no_tpu', action='store_true',
                        help='Don\'t use the Coral-TPU.')
    parser.add_argument('--threshold', type=float, default=0.4,
                        help='Score threshold for detected objects.')
    parser.set_defaults(no_tpu=False)
    args = parser.parse_args()

    run_benchmark(args)

if __name__ == "__main__":
    main()
```

A.5 Επεξεργασία Δεδομένων

Για το σύνολο της επεξεργασίας των δεδομένων χρησιμοποιήθηκαν τμήματα κώδικα από το αποθετήριο κώδικα του JK Jung[Jun18], τα οποία καλύπτονται από την άδεια χρήσης MIT License.

A.5.1 Λήψη και προπεξεργασία συνόλου δεδομένων

prepare_egohands.py

```
"""prepare_egohands.py

This script downloads the 'egohands' dataset and convert its annotations
into bounding boxes in KITTI format.

Output of this script:

  ./egohands_data.zip
  ./egohands├──
    (egohands dataset unzipped)└──
    ......
  ./egohands_kitti_formatted├──
    images├──
      CARDS_COURTYARD_B_T_frame_0011.jpg├──
      ......└──
      PUZZLE_OFFICE_T_S_frame_2697.jpg└──
    labels├──
       CARDS_COURTYARD_B_T_frame_0011.txt├──
       ......└──
       PUZZLE_OFFICE_T_S_frame_2697.txt
"""


import os
import sys
import math
import logging
import argparse
from zipfile import ZipFile
from shutil import rmtree, copyfile

import numpy as np
from scipy.io import loadmat
import cv2


EGOHANDS_DATASET_URL = \
   'http://vision.soic.indiana.edu/egohands_files/egohands_data.zip'
EGOHANDS_DIR = './egohands'
EGOHANDS_DATA_DIR = './egohands/_LABELLED_SAMPLES'
CONVERTED_DIR = './egohands_kitti_formatted'
CONVERTED_IMG_DIR = './egohands_kitti_formatted/images'
CONVERTED_LBL_DIR = './egohands_kitti_formatted/labels'

VISUALIZE = False # visualize each image (for debugging)


def parse_args():
   """Parse input arguments."""
   desc = ('This script downloads the egohands dataset and convert'
```

```
        'the annotations into bounding boxes in KITTI format.')
    parser = argparse.ArgumentParser(description=desc)
    parser.add_argument('--verify', dest='do_verify',
                        help='show and verify each images',
                        action='store_true')
    args = parser.parse_args()
    return args


def download_file(url, dest=None):
    """Download file from an URL."""
    from tqdm import tqdm
    import requests

    if not dest:
        dest = url.split('/')[-1]

    # Streaming, so we can iterate over the response.
    r = requests.get(url, stream=True)

    # Total size in bytes.
    total_size = int(r.headers.get('content-length', 0))
    assert total_size != 0
    block_size = 1024
    wrote = 0
    with open(dest, 'wb') as f:
        for data in tqdm(r.iter_content(block_size),
                         total=math.ceil(total_size//block_size),
                         unit='KB', unit_scale=True):
            wrote = wrote + len(data)
            f.write(data)
    assert wrote == total_size


def polygon_to_box(polygon):
    """Convert 1 polygon into a bounding box.

    # Arguments
      polygon: a numpy array of shape (N, 2) representing N vertices
               of the hand segmentation label (polygon); each vertex
               is a point: (x, y)
    """
    if len(polygon) < 3: # a polygon has at least 3 vertices
        return None

    x_min = np.min(polygon[:, 0])
    y_min = np.min(polygon[:, 1])
    x_max = np.max(polygon[:, 0])
    y_max = np.max(polygon[:, 1])

    x_min = int(math.floor(x_min))
    y_min = int(math.floor(y_min))
    x_max = int(math.ceil(x_max))
    y_max = int(math.ceil(y_max))

    return [x_min, y_min, x_max, y_max]


def box_to_line(box):
    """Convert 1 bounding box into 1 line in the KITTI txt file.

    # Arguments
      box: [x_min, y_min, x_max, y_max].

    KITTI format:
    Values Name   Description
```

```
    ----------------------------------------------------------------------
        ↪
      1   type        Describes the type of object: 'Car', 'Van',
                      'Truck', 'Pedestrian', 'Person_sitting',
                      'Cyclist', 'Tram', 'Misc' or 'DontCare'
      1   truncated Float from 0 (non-truncated) to 1 (truncated),
                      where truncated refers to the object leaving
                      image boundaries
      1   occluded Integer (0,1,2,3) indicating occlusion state:
                      0 = fully visible, 1 = partly occluded
                      2 = largely occluded, 3 = unknown
      1   alpha       Observation angle of object, ranging [-pi..pi]
      4   bbox        2D bounding box of object in the image
                      (0-based index): contains left, top, right,
                      bottom pixel coordinates
      3   dimensions 3D object dimensions: height, width, length
      3   location 3D object location x,y,z in camera coordinates
      1   rotation_y Rotation ry around Y-axis in camera coordinates
                      [-pi..pi]
      1   score       Only for results: Float, indicating confidence
                      in detection, needed for p/r curves, higher is
                      better.
    """
    return ' '.join(['hand',
                     '0',
                     '0',
                     '0',
                     '{} {} {} {}'.format(*box),
                     '0 0 0',
                     '0 0 0',
                     '0',
                     '0'])


def convert_one_folder(folder):
    """Convert egohands to KITTI for 1 data folder (100 images).

    Refer to README.txt in the egohands folder for the format of the
    MATLAB annotation files and how jpg image files are organized.
    The code in this function loads the 'video' struct from the
    MATLAB file, converts polygons into bounding boxes and write
    annotation into KITTI format.
    """
    folder_path = os.path.join(EGOHANDS_DATA_DIR, folder)
    logging.debug('Converting %s' % folder_path)
    frames = [os.path.splitext(f)[0]
              for f in os.listdir(folder_path) if f.endswith('jpg')]
    frames.sort()
    assert len(frames) == 100
    video = loadmat(os.path.join(folder_path, 'polygons.mat'))
    polygons = video['polygons'][0] # there are 100*4 entries in polygons
    for i, frame in enumerate(frames):
        # copy and rename jpg file to the 'converted' folder
        src_jpg = frame + '.jpg'
        dst_jpg = folder + '_' + src_jpg
        copyfile(os.path.join(folder_path, src_jpg),
                 os.path.join(CONVERTED_IMG_DIR, dst_jpg))
        # generate txt (the KITTI annotation corresponding to the jpg)
        dst_txt = folder + '_' + frame + '.txt'
        boxes = []
        with open(os.path.join(CONVERTED_LBL_DIR, dst_txt), 'w') as f:
            for polygon in polygons[i]:
                box = polygon_to_box(polygon)
                if box:
                    boxes.append(box)
                    f.write(box_to_line(box) + '\n')
```

```
      if VISUALIZE:
         img = cv2.imread(os.path.join(CONVERTED_IMG_DIR, dst_jpg))
         for box in boxes:
            cv2.rectangle(img, (box[0], box[1]), (box[2], box[3]),
                          (0, 224, 0), 2)
         cv2.imshow('Visualization', img)
         if cv2.waitKey(0) == 27:
            sys.exit()


def egohands_to_kitti():
   """Convert egohands data and annotations to KITTI format.

   Steps:
     1. walk through each sub-directory in egohands' data folder.
     2. copy each jpg file to the 'converted' image folder and give
        each file a unique name.
     3. convert the original annotations ('polygon.mat') into
        bounding boxes and write a KITTI txt file for each image.
   """
   rmtree(CONVERTED_DIR, ignore_errors=True)
   os.makedirs(CONVERTED_IMG_DIR)
   os.makedirs(CONVERTED_LBL_DIR)
   for folder in os.listdir(EGOHANDS_DATA_DIR):
      convert_one_folder(folder)


def main():
   """main"""
   logging.basicConfig(level=logging.DEBUG)

   egohands_zip_path = EGOHANDS_DATASET_URL.split('/')[-1]
   if not os.path.isfile(egohands_zip_path):
      logging.info('Downloading %s...' % egohands_zip_path)
      download_file(EGOHANDS_DATASET_URL, egohands_zip_path)

   if not os.path.exists(EGOHANDS_DIR):
      with ZipFile(egohands_zip_path, 'r') as zf:
         logging.info('Extracting egohands dataset files...')
         zf.extractall(EGOHANDS_DIR)

   logging.info('Copying jpg files and converting annotations...')
   egohands_to_kitti()

   logging.info('All done.')


if __name__ == '__main__':
   main()
   sys.exit()
```

A.5.2 Μετατροπή του συνόλου δεδομένων σε TFRecords

Data/egohands_label_map.pbtxt

```
item {
  id: 1
  name: 'hand'
}
```

create_tfrecords.sh

```
#!/bin/bash

PYTHONPATH=`pwd`/models/research:`pwd`/models/research/slim \
   python3 create_kitti_tf_record.py \
         --data_dir=egohands_kitti_formatted \
-        --output_path=data/egohands \
+        --output_path=Data/egohands \
         --classes_to_use=hand \
-        --label_map_path=data/egohands_label_map.pbtxt \
+        --label_map_path=Data/egohands_label_map.pbtxt \
         --validation_set_size=500
```

create_kitti_tf_record.py

```
# https://github.com/tensorflow/models/blob/master/research/
# object_detection/dataset_tools/create_kitti_tf_record.py

# ==============================================================================
# Copyright 2017 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================


"""Convert KITTI detection dataset to TFRecord for object detection.

Example usage:

   python3 create_kitti_tf_record.py \
         --data_dir=egohands_kitti_formatted \
         --output_path=data/egohands \
         --classes_to_use=hand \
         --label_map_path=data/egohands_label_map.pbtxt \
         --validation_set_size=500
"""


from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
```

```
import hashlib
import io
import os
import random

import numpy as np
import PIL.Image as pil
import tensorflow as tf

import sys
sys.path.append('/home/nikos/Documents/Thesis/Libraries/Tensorflow/research
    ↪ /')

from object_detection.utils import dataset_util
from object_detection.utils import label_map_util
from object_detection.utils.np_box_ops import iou


tf.app.flags.DEFINE_string('data_dir', '', 'Location for the data. '
                    'All jpg and txt files are assumed to be '
                    'present at this location.')
tf.app.flags.DEFINE_string('output_path', '', 'Path to which TFRecord '
                    'files will be written. The TFRecord with '
                    'the training set will be located at: '
                    '<output_path>_train.tfrecord. And the '
                    'TFRecord with the validation set will be '
                    'located at: <output_path>_val.tfrecord.')
tf.app.flags.DEFINE_string('classes_to_use', 'hand,dontcare',
                    'Comma separated list of class names that '
                    'will be used. Adding the dontcare class '
                    'will remove all bboxs in the dontcare '
                    'regions.')
tf.app.flags.DEFINE_string('label_map_path',
                    'data/egohands_label_map.pbtxt',
                    'Path to label map proto.')
tf.app.flags.DEFINE_integer('validation_set_size', '500',
                     'Number of images to be used as a '
                     'validation set.')
FLAGS = tf.app.flags.FLAGS


def convert_kitti_to_tfrecords(data_dir, output_path, classes_to_use,
                       label_map_path, validation_set_size):
  """Convert the KITTI detection dataset to TFRecords.

  Args:
    data_dir: The full path containing KITTI formatted data, with all
      jpg files located in <data_dir>/images/ and all txt files in
      <data_dir>/labels/.
    output_path: The path to which TFRecord files will be written.
      The TFRecord with the training set will be located at:
      <output_path>_train.tfrecord
      And the TFRecord with the validation set will be located at:
      <output_path>_val.tfrecord
    classes_to_use: List of strings naming the classes for which
      data should be converted. Use the same names as presented in
      the KIITI README file. Adding dontcare class will remove all
      other bounding boxes that overlap with areas marked as dontcare
       regions.
    label_map_path: Path to label map proto
    validation_set_size: How many images should be left as the
      validation set. (First `validation_set_size` examples are
      selected to be in the validation set).
  """
  label_map_dict = label_map_util.get_label_map_dict(label_map_path)
  train_count = 0
```

```
  val_count = 0

  annotation_dir = os.path.join(data_dir, 'labels')
  image_dir = os.path.join(data_dir, 'images')

  train_writer = tf.python_io.TFRecordWriter('%s_train.tfrecord'%
                                  output_path)
  val_writer = tf.python_io.TFRecordWriter('%s_val.tfrecord'%
                                output_path)

  images = sorted(tf.gfile.ListDirectory(image_dir))
  images = [f for f in images if f.endswith('jpg')] # only keep jpg files
  assert len(images) > 0
  random.shuffle(images)
  for idx, img_name in enumerate(images):
    img_num = img_name.split('.')[0]
    is_validation_img = idx < validation_set_size
    img_anno = read_annotation_file(os.path.join(annotation_dir,
                                    img_num+'.txt'))

    image_path = os.path.join(image_dir, img_name)

    # Filter all bounding boxes of this frame that are of a legal class,
    # and don't overlap with a dontcare region.
    # TODO(talremez) filter out targets that are truncated or heavily
    # occluded.
    annotation_for_image = filter_annotations(img_anno, classes_to_use)

    example = prepare_example(image_path, annotation_for_image,
        ↪ label_map_dict)
    if is_validation_img:
      val_writer.write(example.SerializeToString())
      val_count += 1
    else:
      train_writer.write(example.SerializeToString())
      train_count += 1

  train_writer.close()
  val_writer.close()


def prepare_example(image_path, annotations, label_map_dict):
  """Converts a dictionary with annotations for an image to tf.Example proto
      ↪ .

  Args:
    image_path: The complete path to image.
    annotations: A dictionary representing the annotation of a single object
      that appears in the image.
    label_map_dict: A map from string label names to integer ids.

  Returns:
    example: The converted tf.Example.
  """
  with tf.gfile.GFile(image_path, 'rb') as fid:
    encoded_png = fid.read()
  encoded_png_io = io.BytesIO(encoded_png)
  image = pil.open(encoded_png_io)
  image = np.asarray(image)

  key = hashlib.sha256(encoded_png).hexdigest()

  width = int(image.shape[1])
  height = int(image.shape[0])

  xmin_norm = annotations['2d_bbox_left'] / float(width)
```

```
  ymin_norm = annotations['2d_bbox_top'] / float(height)
  xmax_norm = annotations['2d_bbox_right'] / float(width)
  ymax_norm = annotations['2d_bbox_bottom'] / float(height)

  difficult_obj = [0]*len(xmin_norm)

  example = tf.train.Example(features=tf.train.Features(feature={
     'image/height': dataset_util.int64_feature(height),
     'image/width': dataset_util.int64_feature(width),
     'image/filename': dataset_util.bytes_feature(image_path.encode('utf8'))
         ↪ ,
     'image/source_id': dataset_util.bytes_feature(image_path.encode('utf8')
         ↪ ),
     'image/key/sha256': dataset_util.bytes_feature(key.encode('utf8')),
     'image/encoded': dataset_util.bytes_feature(encoded_png),
     'image/format': dataset_util.bytes_feature('png'.encode('utf8')),
     'image/object/bbox/xmin': dataset_util.float_list_feature(xmin_norm),
     'image/object/bbox/xmax': dataset_util.float_list_feature(xmax_norm),
     'image/object/bbox/ymin': dataset_util.float_list_feature(ymin_norm),
     'image/object/bbox/ymax': dataset_util.float_list_feature(ymax_norm),
     'image/object/class/text': dataset_util.bytes_list_feature(
       [x.encode('utf8') for x in annotations['type']]),
     'image/object/class/label': dataset_util.int64_list_feature(
       [label_map_dict[x] for x in annotations['type']]),
     'image/object/difficult': dataset_util.int64_list_feature(difficult_obj
         ↪ ),
     'image/object/truncated': dataset_util.float_list_feature(
       annotations['truncated']),
     'image/object/alpha': dataset_util.float_list_feature(
       annotations['alpha']),
     'image/object/3d_bbox/height': dataset_util.float_list_feature(
       annotations['3d_bbox_height']),
     'image/object/3d_bbox/width': dataset_util.float_list_feature(
       annotations['3d_bbox_width']),
     'image/object/3d_bbox/length': dataset_util.float_list_feature(
       annotations['3d_bbox_length']),
     'image/object/3d_bbox/x': dataset_util.float_list_feature(
       annotations['3d_bbox_x']),
     'image/object/3d_bbox/y': dataset_util.float_list_feature(
       annotations['3d_bbox_y']),
     'image/object/3d_bbox/z': dataset_util.float_list_feature(
       annotations['3d_bbox_z']),
     'image/object/3d_bbox/rot_y': dataset_util.float_list_feature(
       annotations['3d_bbox_rot_y']),
  }))

  return example


def filter_annotations(img_all_annotations, used_classes):
  """Filters out annotations from the unused classes and dontcare regions.

  Filters out the annotations that belong to classes we do now wish to use
      ↪ and
  (optionally) also removes all boxes that overlap with dontcare regions.

  Args:
    img_all_annotations: A list of annotation dictionaries. See
        ↪ documentation of
      read_annotation_file for more details about the format of the
          ↪ annotations.
    used_classes: A list of strings listing the classes we want to keep, if
        ↪ the
    list contains "dontcare", all bounding boxes with overlapping with dont
    care regions will also be filtered out.
```

```
  Returns:
    img_filtered_annotations: A list of annotation dictionaries that have
        ↪ passed
     the filtering.
  """

  img_filtered_annotations = {}

  # Filter the type of the objects.
  relevant_annotation_indices = [
     i for i, x in enumerate(img_all_annotations['type']) if x in
         ↪ used_classes
  ]

  for key in img_all_annotations.keys():
    img_filtered_annotations[key] = (
       img_all_annotations[key][relevant_annotation_indices])

  if 'dontcare' in used_classes:
    dont_care_indices = [i for i,
                         x in enumerate(img_filtered_annotations['type'])
                         if x == 'dontcare']

    # bounding box format [y_min, x_min, y_max, x_max]
    all_boxes = np.stack([img_filtered_annotations['2d_bbox_top'],
                          img_filtered_annotations['2d_bbox_left'],
                          img_filtered_annotations['2d_bbox_bottom'],
                          img_filtered_annotations['2d_bbox_right']],
                         axis=1)

    ious = iou(boxes1=all_boxes,
               boxes2=all_boxes[dont_care_indices])

    # Remove all bounding boxes that overlap with a dontcare region.
    if ious.size > 0:
      boxes_to_remove = np.amax(ious, axis=1) > 0.0
      for key in img_all_annotations.keys():
        img_filtered_annotations[key] = (
           img_filtered_annotations[key][np.logical_not(boxes_to_remove)])

  return img_filtered_annotations

def read_annotation_file(filename):
  """Reads a KITTI annotation file.

  Converts a KITTI annotation file into a dictionary containing all the
  relevant information.

  Args:
    filename: the path to the annotataion text file.

  Returns:
    anno: A dictionary with the converted annotation information. See
        ↪ annotation
    README file for details on the different fields.
  """
  with open(filename) as f:
    content = f.readlines()
  content = [x.strip().split(' ') for x in content]

  anno = {}
  anno['type'] = np.array([x[0].lower() for x in content])
  anno['truncated'] = np.array([float(x[1]) for x in content])
  anno['occluded'] = np.array([int(x[2]) for x in content])
  anno['alpha'] = np.array([float(x[3]) for x in content])
```

```
  anno['2d_bbox_left'] = np.array([float(x[4]) for x in content])
  anno['2d_bbox_top'] = np.array([float(x[5]) for x in content])
  anno['2d_bbox_right'] = np.array([float(x[6]) for x in content])
  anno['2d_bbox_bottom'] = np.array([float(x[7]) for x in content])

  anno['3d_bbox_height'] = np.array([float(x[8]) for x in content])
  anno['3d_bbox_width'] = np.array([float(x[9]) for x in content])
  anno['3d_bbox_length'] = np.array([float(x[10]) for x in content])
  anno['3d_bbox_x'] = np.array([float(x[11]) for x in content])
  anno['3d_bbox_y'] = np.array([float(x[12]) for x in content])
  anno['3d_bbox_z'] = np.array([float(x[13]) for x in content])
  anno['3d_bbox_rot_y'] = np.array([float(x[14]) for x in content])

  return anno

def main(_):
  convert_kitti_to_tfrecords(
     data_dir=FLAGS.data_dir,
     output_path=FLAGS.output_path,
     classes_to_use=FLAGS.classes_to_use.split(','),
     label_map_path=FLAGS.label_map_path,
     validation_set_size=FLAGS.validation_set_size)

if __name__ == '__main__':
  tf.app.run()
```

Παράρτημα Β

Παραμετρικά αρχεία εκπαίδευσης μοντέλων

Β.1 MobileNet v1

Ο παρακάτω πίνακας παρουσιάζει τον συνδυασμό παραμέτρων εκπαίδευσης που χρησιμοποιήθηκαν για κάθε υπό εξέταση μοντέλο αρχιτεκτονικής MobileNet v1. Κάθε γραμμή αντιστοιχεί σε ένα τέτοιο μοντέλο. Για οικονομία χώρου, παραθέτουμε ένα παραμετρικό αρχείο εκπαίδευσης με κατάλληλες επισημάνσεις, αντίστοιχες των ονομάτων της κάθε στήλης, στις οποίες τοποθετείται κάθε φορά η κατάλληλη τιμή.

depth_multiplier	model_location
1.0	mobilenet_v1
0.75	mobilenet_v1_0.75

Πίνακας Β.1: Συνδυασμοί παραμέτρων για την εκπαίδευση δικτύων MobileNet v1

```
ssd {
  num_classes: 1
  image_resizer {
    fixed_shape_resizer {
      height: 300
      width: 300
    }
  }
  feature_extractor {
    type: "ssd_mobilenet_v1"
    depth_multiplier: <depth_multiplier>
    min_depth: 16
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 3.99999989895e-05
        }
```

```
    }
    initializer {
     random_normal_initializer {
      mean: 0.0
      stddev: 0.00999999977648
     }
    }
    activation: RELU_6
    batch_norm {
     decay: 0.97000002861
     center: true
     scale: true
     epsilon: 0.0010000000475
    }
   }
   override_base_feature_extractor_hyperparams: true
}
box_coder {
  faster_rcnn_box_coder {
    y_scale: 10.0
    x_scale: 10.0
    height_scale: 5.0
    width_scale: 5.0
  }
}
matcher {
  argmax_matcher {
    matched_threshold: 0.5
    unmatched_threshold: 0.5
    ignore_thresholds: false
    negatives_lower_than_unmatched: true
    force_match_for_each_row: true
    use_matmul_gather: true
  }
}
similarity_calculator {
  iou_similarity {
  }
}
box_predictor {
  convolutional_box_predictor {
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 3.99999989895e-05
        }
      }
      initializer {
        random_normal_initializer {
          mean: 0.0
          stddev: 0.00999999977648
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.97000002861
        center: true
        scale: true
        epsilon: 0.0010000000475
      }
    }
    min_depth: 0
    max_depth: 0
    num_layers_before_predictor: 0
    use_dropout: false
    dropout_keep_probability: 0.800000011921
```

```
      kernel_size: 1
      box_code_size: 4
      apply_sigmoid_to_scores: false
      class_prediction_bias_init: -4.59999990463
    }
  }
  anchor_generator {
    ssd_anchor_generator {
      num_layers: 6
      min_scale: 0.20000000298
      max_scale: 0.949999988079
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      aspect_ratios: 3.0
      aspect_ratios: 0.333299994469
    }
  }
  post_processing {
    batch_non_max_suppression {
      score_threshold: 0.300000011921
      iou_threshold: 0.600000023842
      max_detections_per_class: 100
      max_total_detections: 100
    }
    score_converter: SIGMOID
  }
  normalize_loss_by_num_matches: true
  loss {
    localization_loss {
      weighted_smooth_l1 {
      }
    }
    classification_loss {
      weighted_sigmoid_focal {
        gamma: 2.0
        alpha: 0.75
      }
    }
    classification_weight: 1.0
    localization_weight: 1.0
  }
  encode_background_as_zeros: true
  normalize_loc_loss_by_codesize: true
  inplace_batchnorm_update: true
  freeze_batchnorm: false
 }
}
train_config {
 batch_size: 10
 data_augmentation_options {
  random_horizontal_flip {
  }
 }
 data_augmentation_options {
  ssd_random_crop {
  }
 }
 sync_replicas: true
 optimizer {
  momentum_optimizer {
   learning_rate {
    cosine_decay_learning_rate {
     learning_rate_base: 0.20000000298
     total_steps: 50000
     warmup_learning_rate: 0.0599999986589
```

```
        warmup_steps: 2000
       }
      }
      momentum_optimizer_value: 0.899999976158
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "../Models/<model_location>/model.ckpt"
  from_detection_checkpoint: true
  load_all_detection_checkpoint_vars: true
  num_steps: 50000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
}
train_input_reader {
  label_map_path: "../Data/egohands_label_map.pbtxt"
  tf_record_input_reader {
    input_path: "../Data/egohands_train.tfrecord"
  }
}
eval_config {
  num_examples: 500
  eval_interval_secs: 100
  num_visualizations: 20
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "../Data/egohands_label_map.pbtxt"
  shuffle: false
  num_readers: 1
  tf_record_input_reader {
    input_path: "../Data/egohands_val.tfrecord"
  }
}
graph_rewriter {
  quantization {
    delay: 48000
    weight_bits: 8
    activation_bits: 8
  }
}
```

B.2 MobileNet v2

Ο παρακάτω πίνακας παρουσιάζει τον συνδυασμό παραμέτρων εκπαίδευσης που χρησιμοποιήθηκαν για κάθε υπό εξέταση μοντέλο αρχιτεκτονικής MobileNet v2. Κάθε γραμμή αντιστοιχεί σε ένα τέτοιο μοντέλο. Για οικονομία χώρου, παραθέτουμε ένα παραμετρικό αρχείο εκπαίδευσης με κατάλληλες επισημάνσεις, αντίστοιχες των ονομάτων της κάθε στήλης, στις οποίες τοποθετείται κάθε φορά η κατάλληλη τιμή.

depth_multiplier	model_location
1.0	mobilenet_v2
0.75	mobilenet_v2_0.75

Πίνακας B.2: Συνδυασμοί παραμέτρων για την εκπαίδευση δικτύων MobileNet v2

```
model {
  ssd {
    num_classes: 1
    image_resizer {
      fixed_shape_resizer {
        height: 300
        width: 300
      }
    }
    feature_extractor {
      type: "ssd_mobilenet_v2"
      depth_multiplier: <depth_multiplier>
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.99999989895e-05
          }
        }
        initializer {
          truncated_normal_initializer {
            mean: 0.0
            stddev: 0.0299999993294
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.999700009823
          center: true
          scale: true
          epsilon: 0.0010000000475
          train: true
        }
      }
    }
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
```

```
}
matcher {
  argmax_matcher {
    matched_threshold: 0.5
    unmatched_threshold: 0.5
    ignore_thresholds: false
    negatives_lower_than_unmatched: true
    force_match_for_each_row: true
  }
}
similarity_calculator {
  iou_similarity {
  }
}
box_predictor {
  convolutional_box_predictor {
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 3.99999989895e-05
        }
      }
      initializer {
        truncated_normal_initializer {
          mean: 0.0
          stddev: 0.0299999993294
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.999700009823
        center: true
        scale: true
        epsilon: 0.0010000000475
        train: true
      }
    }
    min_depth: 0
    max_depth: 0
    num_layers_before_predictor: 0
    use_dropout: false
    dropout_keep_probability: 0.800000011921
    kernel_size: 1
    box_code_size: 4
    apply_sigmoid_to_scores: false
  }
}
anchor_generator {
  ssd_anchor_generator {
    num_layers: 6
    min_scale: 0.20000000298
    max_scale: 0.949999988079
    aspect_ratios: 1.0
    aspect_ratios: 2.0
    aspect_ratios: 0.5
    aspect_ratios: 3.0
    aspect_ratios: 0.333299994469
  }
}
post_processing {
  batch_non_max_suppression {
    score_threshold: 9.99999993923e-09
    iou_threshold: 0.600000023842
    max_detections_per_class: 100
    max_total_detections: 100
  }
```

```
    score_converter: SIGMOID
   }
   normalize_loss_by_num_matches: true
   loss {
    localization_loss {
     weighted_smooth_l1 {
     }
    }
    classification_loss {
     weighted_sigmoid {
     }
    }
    hard_example_miner {
     num_hard_examples: 250
     iou_threshold: 0.990000009537
     loss_type: CLASSIFICATION
     max_negatives_per_positive: 3
     min_negatives_per_image: 3
    }
    classification_weight: 1.0
    localization_weight: 1.0
   }
  }
}
train_config {
  batch_size: 10
  data_augmentation_options {
   random_horizontal_flip {
   }
  }
  data_augmentation_options {
   ssd_random_crop {
   }
  }
  optimizer {
   rms_prop_optimizer {
    learning_rate {
     exponential_decay_learning_rate {
      initial_learning_rate: 0.00400000018999
      decay_steps: 2000
      decay_factor: 0.949999988079
     }
    }
    momentum_optimizer_value: 0.899999976158
    decay: 0.899999976158
    epsilon: 1.0
   }
  }
  fine_tune_checkpoint: "../Models/<model_location>/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 50000
}
train_input_reader {
  label_map_path: "../Data/egohands_label_map.pbtxt"
  tf_record_input_reader {
   input_path: "../Data/egohands_train.tfrecord"
  }
}
eval_config {
  num_examples: 500
  eval_interval_secs: 100
  num_visualizations: 20
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
```

```
  label_map_path: "../Data/egohands_label_map.pbtxt"
  shuffle: false
  num_readers: 1
  tf_record_input_reader {
    input_path: "../Data/egohands_val.tfrecord"
  }
}
graph_rewriter {
  quantization {
    delay: 48000
    weight_bits: 8
    activation_bits: 8
  }
}
```