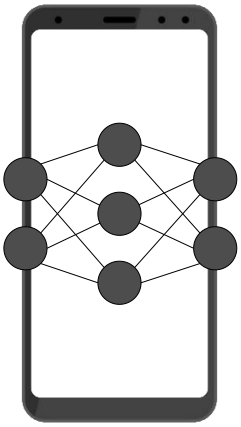
Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Συστηματων Μεταδοσης Πληροφοριας και Τεχνολογιας Υλικων

Αρχιτεκτονική, Σχεδίαση και Ανάπτυξη Μοντέλου Μηχανικής Μάθησης για Εφαρμογή Λήψης και Διαχείρισης Εικόνας σε Κινητές Συσκευές

Διπλωματικη Εργασια

του

ΠΑΝΟΠΟΥΛΟΥ Α. ΙΩΑΝΝΗ



Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Συστηματων Μεταδοσης Πληροφοριας και Τεχνολογιας Υλικων

Αρχιτεκτονική, Σχεδίαση και Ανάπτυξη Μοντέλου Μηχανικής Μάθησης για Εφαρμογή Λήψης και Διαχείρισης Εικόνας σε Κινητές Συσκευές

Διπλωματικη Εργασια

του

ΠΑΝΟΠΟΥΛΟΥ Α. ΙΩΑΝΝΗ

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Οκτωβρίου 2020.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Ιάκωβος Βενιέρης	Δήμητρα-Θεοδώρα Κακλαμάνη	Γεώργιος Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Συστηματων Μεταδοσης Πληροφοριας και Τεχνολογιας Υλικων

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Ιωάννης Α. Πανόπουλος

23 Οκτωβρίου 2020

Περίληψη

Στην παρούσα διπλωματική εργασία σχεδιάζεται και αναπτύσσεται ένα σύστημα λήψης και διαχείρισης εικόνων σε κινητές συσκευές με εφαρμογές Βαθιάς Μάθησης. Πιο συγκεκριμένα, το σύστημα εκτελεί αναγνώριση εικόνας σε πραγματικό χρόνο από την κάμερα μιας κινητής συσκευής χρησιμοποιώντας βαθιά νευρωνικά δίκτυα. Στην ουσία πρόκειται για μια απλή, «έξυπνη» εφαρμογή κάμερας (AI camera app) σε λειτουργικό σύστημα Android.

Η ενσωμάτωση της Βαθιάς Μάθησης στις κινητές συσκευές δεν είναι εύκολη διαδικασία καθώς οι διαθέσιμοι υπολογιστικοί πόροι είναι περιορισμένοι, ενώ οι απαιτήσεις των μοντέλων Βαθιάς Μάθησης υψηλές. Αν αναλογιστεί κανείς ότι μια εφαρμογή κάμερας είναι επίσης απαιτητική μιας και συσχετίζεται άμεσα με το υλικό της συσκευής, τότε η παράλληλη χρησιμοποίηση της κάμερας και του συστήματος αναγνώρισης ίσως μεταβάλλει τη συμπεριφορά ή μειώσει την απόδοση ενός εκ των δύο. Επομένως, απαιτείται η σχεδίαση ενός συστήματος με την κατάλληλη αρχιτεκτονική, η οποία θα δίνει τη δυνατότητα διερεύνησης όλων των ζητημάτων απόδοσης και αξιολόγησης που θα προκύψουν.

Στο πρώτο Κεφάλαιο της εργασίας παρουσιάζεται το θεωρητικό υπόβαθρο, στο δεύτερο περιγράφονται οι τεχνολογίες που αξιοποιήθηκαν, στο τρίτο περιγράφεται η αρχιτεκτονική - διαστρωμάτωση του ολοκληρωμένου συστήματος, στο τέταρτο αναλύονται τα βασικά δομικά στοιχεία και οι λειτουργίες της Android εφαρμογής, στο πέμπτο αναφέρονται βασικά στοιχεία για την ανάπτυξη του συστήματος, όπως κλάσεις και μοντέλα, στο έκτο παρουσιάζονται οι μετρήσεις χρόνου και μνήμης, ενώ στο έβδομο και τελευταίο Κεφάλαιο αναλύονται τα συμπεράσματα, η συνεισφορά της διπλωματικής εργασίας και οι μελλοντικές επεκτάσεις της.

Λέξεις Κλειδιά

Βαθιά Μάθηση, Κινητές Συσκευές, Συνελικτικά Νευρωνικά Δίκτυα, Αναγνώριση Εικόνας, Κατηγοριοποίηση, Android, Εφαρμογή Κάμερας, TensorFlow Lite, Αρχιτεκτονική

Abstract

In this diploma thesis, a system for capturing and managing images on mobile devices with Deep Learning applications is designed and developed. More specifically, the system performs real-time image recognition from the camera of a mobile device using deep neural networks. It is essentially a simple, "smart" camera application developed on the Android system.

Integrating Deep Learning into mobile devices is not an easy process as the available computing resources are limited and the requirements of Deep Learning models are high. If one considers that a camera application is also demanding since it includes device hardware in the foreground, then the parallel use of the camera and the recognition system may change the behavior or reduce the performance of one of the two. Therefore, it is necessary to design a system with the appropriate architecture, which will enable the investigation of all performance and evaluation issues that will arise.

The first Chapter of the work presents the theoretical background, the second describes the technologies used, the third describes the architecture - stratification of the integrated system, the fourth analyzes the basic structural elements and functions of the Android application, the fifth describes basic elements about the development of the system, such as classes and models, in the sixth Chapter the measurements of time and memory are presented, while in the seventh and last Chapter the conclusions, the contribution of the diploma thesis and its future extensions are analyzed.

Keywords

Deep Learning, Mobile Devices, Convolutional Neural Networks, Image Recognition, Categorization, Android, Camera Application, TensorFlow Lite, Architecture

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον καθηγητή κ. Ιάκωβο Βενιέρη για την επίβλεψη αυτής της διπλωματικής εργασίας, την εμπιστοσύνη που μου έδειξε από την πρώτη στιγμή και την ευκαιρία που μου έδωσε να ασχοληθώ με έναν ραγδαία αναπτυσσόμενο και πολλά υποσχόμενο επιστημονικό κλάδο. Ακόμη, ευχαριστώ τους καθηγητές κ. Δήμητρα-Θεοδώρα Κακλαμάνη και κ. Γιώργο Ματσόπουλο για τη συμμετοχή τους στην τριμελή εξεταστική επιτροπή αλλά και ολόκληρο το εργαστήριο Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης για την υποστήριξή του.

Επίσης ευχαριστώ ιδιαίτερα τον Δρ. Στυλιανό Βενιέρη, ερευνητή στο Κέντρο Τεχνητής Νοημοσύνης της Samsung στο Cambridge για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Οι συμβουλές και οι γνώσεις του υπήρξαν καθοριστικές για την ομαλή έκβαση και περάτωση της διπλωματικής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους, τους δικούς μου ανθρώπους και την οικογένειά μου για όλη τη στήριξη, τη βοήθεια και τις στιγμές που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Οκτώβριος 2020

Ιωάννης Α. Παυόπουλος

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Αντί Προλόγου

Η διπλωματική εργασία εκπονήθηκε κατά το ακαδημαϊκό έτος 2019-2020, κατά το οποίο ήμουν τελειόφοιτος φοιτητής στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Μέσα από αυτή γνώρισα έναν πολύ «φρέσκο» και ανερχόμενο επιστημονικό κλάδο, τη Μηχανική Μάθηση για Κινητές Συσκευές, ο οποίος μου τράβηξε από την πρώτη στιγμή το ενδιαφέρον και με τον οποίο θα συνεχίσω να ασχολούμαι κατά τις διδακτορικές μου σπουδές.

Ζούμε σε μια εποχή όπου σχεδόν όλοι οι επιστήμονες επιθυμούν να εντάξουν την Τεχνητή Νοημοσύνη στις ερευνητικές δραστηριότητές τους. Το κίνητρο πίσω από αυτό το τεράστιο ενδιαφέρον δεν είναι άλλο από το γεγονός ότι η Μηχανική και η Βαθιά Μάθηση τα τελευταία χρόνια έχουν επιφέρει εκπληκτικά αποτελέσματα σε πολυάριθμα προβλήματα. Το μυστικό της Μηχανικής Μάθησης είναι η δυνατότητά της να εντοπίζει συσχετίσεις μεταξύ των διαθέσιμων δεδομένων, ενώ η Βαθιά Μάθηση μπορεί να μαθαίνει καλά και να αναπαριστά τον κόσμο σαν μια ιεραρχία από έννοιες, στην οποία κάθε έννοια ορίζεται μέσα από τη σχέση της με απλούστερες έννοιες.

Τα τελευταία τρία με τέσσερα χρόνια η Μηχανική Μάθηση έχει κάνει τα πρώτα της σημαντικά βήματα προς τις κινητές συσκευές σκοπεύοντας να έρθει πιο κοντά στον απλό χρήστη. Τα δεδομένα που παράγει καθημερινά μια κινητή συσκευή αντικατοπτρίζουν τις συνήθειες, τη συμπεριφορά και τις προτιμήσεις των χρηστών, γεγονός που τα καθιστά σημαντικά και κατάλληλα για τις εφαρμογές της Μηχανικής Μάθησης. Πολλοί αναφέρουν ότι το επόμενο μεγάλο βήμα είναι η μεταφορά της Μηχανικής Μάθησης από τους υπολογιστές στις κινητές συσκευές και συμφωνώ. Η προσαρμογή σε αυτό το νέο περιβάλλον, ωστόσο, έχει αρκετές δυσκολίες και προκλήσεις.

Κεφάλαιο 1

Εισαγωγή

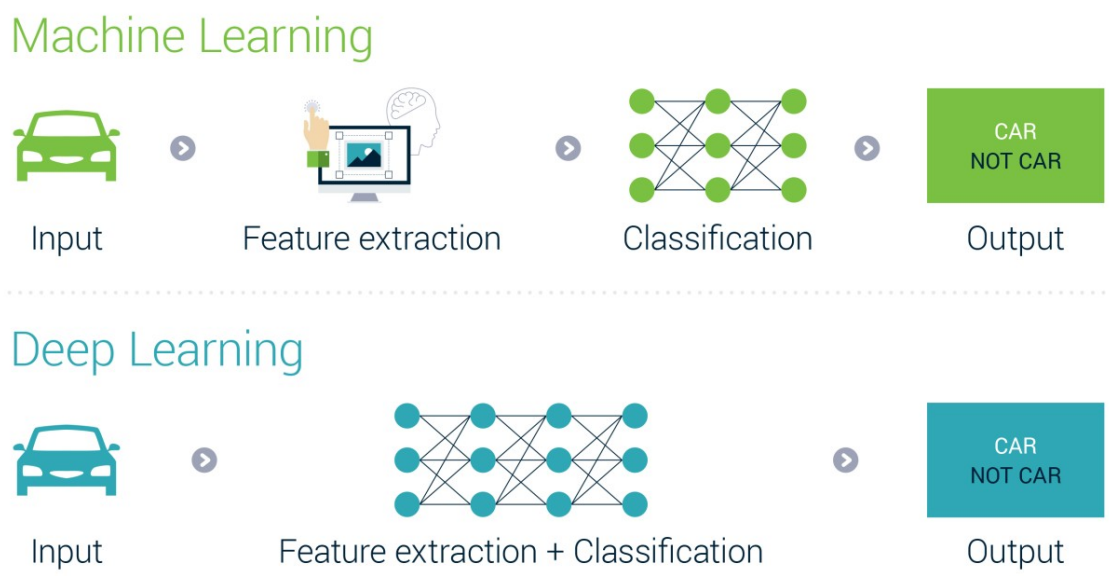
Στο πρώτο Κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο που θα πρέπει να έχει κανείς ώστε να μπορέσει να παρακολουθήσει την πορεία της εργασίας. Στις παρακάτω Ενότητες αναλύονται οι βασικές έννοιες που σχετίζονται με την εργασία, ξεκινώντας από πιο τις πιο γενικές και καταλήγοντας στις πιο ειδικές.

1.1 Βαθιά Μάθηση

Η Βαθιά Μάθηση (Deep Learning) είναι υποκατηγορία της Μηχανικής Μάθησης (Machine Learning) και γεννήθηκε από την επιθυμία των ερευνητών να λύσουν απλά προβλήματα που ο άνθρωπος λύνει διαισθητικά ή αυτόματα, χρησιμοποιώντας Τεχνητή Νοημοσύνη (Artificial Intelligence). Τέτοια προβλήματα περιλαμβάνουν την αναγνώριση ήχων ή την αναγνώριση προσώπων σε μια εικόνα. Οι αλγόριθμοι που χρησιμοποιούνται στην Μηχανική Μάθηση είναι απλοί και γι' αυτό συνήθως δεν μπορούν να διαχειριστούν ακατέργαστα δεδομένα (raw data), οπότε για να λειτουργήσουν αποτελεσματικά απαιτούν από τον ερευνητή να εξάγει τα κατάλληλα χαρακτηριστικά (hand-designed features). Σε πολλά προβλήματα ωστόσο, δεν είναι εύκολο να βρεθεί ποια είναι τα χαρακτηριστικά ή ποια από αυτά είναι κατάλληλα. Με τη Βαθιά Μάθηση, ο ερευνητής δεν χρειάζεται να εξάγει χαρακτηριστικά, αφού πλέον αυτό γίνεται από τα μοντέλα βαθιάς μάθησης, τα οποία δέχονται ως είσοδο απευθείας τα ακατέργαστα δεδομένα [1]. Αυτή η διαφορά φαίνεται καθαρά στο Σχήμα 1.1 [2] της επόμενης σελίδας.

Ο όρος «βαθιά» προέρχεται από το γεγονός ότι οι αρχιτεκτονικές που χρησιμοποιούνται έχουν πολλαπλά επίπεδα, με το καθένα να μετασχηματίζει τα δεδομένα εισόδου του σε μια ελαφρώς πιο αφηρημένη και ταυτόχρονα πιο σύνθετη αναπαράσταση [3]. Αυτό σημαίνει ότι οι μηχανές μαθαίνουν από εμπειρία και βλέπουν τον κόσμο σαν μια ιεραρχία από έννοιες, στην οποία κάθε έννοια ορίζεται μέσα από τη σχέση της με απλούστερες έννοιες [1]. Για παράδειγμα, στην επεξεργασία εικόνας, τα χαμηλότερα επίπεδα μπορεί να αναγνωρίζουν ακμές ή γωνίες, ενώ τα υψηλότερα έννοιες πιο κοντά στον άνθρωπο όπως ψηφία, γράμματα ή πρόσωπα.

Τα περισσότερα μοντέλα βαθιάς μάθησης βασίζονται σε τεχνητά νευρωνικά δίκτυα (artificial neural networks - ANNs). Ο όρος αυτός προέκυψε από το γεγονός ότι τα αρχικά μοντέλα σχεδιάστηκαν με σκοπό να μιμούνται τον τρόπο με τον οποίο γίνεται ή πιθανώς γίνεται η μάθηση σε έναν εγκέφαλο. Σήμερα, η νευροεπιστήμη θεωρείται σημαντική πηγή



Σχήμα 1.1: *Διαφορά μεταξύ Μηχανικής και Βαθιάς Μάθησης*

έμπνευσης για τη Βαθιά Μάθηση, αλλά δεν είναι πια ο κυρίαρχος οδηγός [1].

Δύο πολύ δημοφιλείς αρχιτεκτονικές βαθιών νευρωνικών δικτύων που χρησιμοποιούνται ευρέως είναι (α) τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs), τα οποία είναι κατάλληλα για την επεξεργασία πληροφορίας που μπορεί να αναπαρασταθεί σε μορφή πλέγματος, όπως είναι για παράδειγμα το δισδιάστατο πλέγμα μιας εικόνας (βλ. Ενότητα 1.3) και (β) τα Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNNs), τα οποία μπορούν να επεξεργαστούν αποτελεσματικά ακολουθιακά δεδομένα, όπως είναι ο ήχος ή ένα κείμενο.

Τα βαθιά νευρωνικά δίκτυα υπάρχουν ως θεωρητικά μοντέλα από τη δεκαετία του 1980. Ωστόσο εκείνη την εποχή δεν μπορούσαν να εφαρμοστούν πρακτικά λόγω έλλειψης:

(α΄) υπολογιστικών πόρων, καθώς ο αριθμός των παραμέτρων που πρέπει να υπολογίζονται σε κάθε βήμα εκπαίδευσης είναι πολύ μεγάλος και

(β΄) κατάλληλων (μεγάλων) συνόλων δεδομένων, καθώς ο όγκος των δεδομένων με ετικέτα που απαιτούνται για την εκπαίδευση ενός βαθιού δικτύου ώστε να έχει ικανοποιητικά αποτελέσματα είναι αρκετά μεγάλος [4].

Τα βαθιά νευρωνικά δίκτυα ήρθαν πάλι στο προσκήνιο αρχικά από τον Geoffrey Hinton το 2006, όπου και γεννήθηκε ο όρος «Deep Learning» και στη συνέχεια από το 2012 και μετά, όταν πια έγινε δυνατή η εκπαίδευση βαθιών νευρωνικών δικτύων ακόμα και από απλούς τελικούς χρήστες χάρη (α) στην ανάπτυξη του hardware και συγκεκριμένα των επιταχυντών και (β) στην δημιουργία μεγαλύτερων συνόλων δεδομένων με ετικέτα.

Οι επιταχυντές (accelerators) είναι υπολογιστικές μονάδες που επιτρέπουν την εκτέλεση μαζικά παράλληλου υπολογισμού, σαν αυτόν που απαιτείται για την εκπαίδευση νευρωνικών δικτύων με μεγάλο αριθμό κρυφών επιπέδων και νευρώνων. Οι πιο γνωστοί επιταχυντές είναι οι Μονάδες Επεξεργασίας Γραφικών (Graphics Processing Units - GPUs). Οι μονάδες αυτές, όπως φαίνεται και από το όνομά τους, αρχικά σχεδιάστηκαν για την απεικόνιση γραφικών,

ωστόσο η αρχιτεκτονική τους έχει πολλά χαρακτηριστικά που τις κάνει καταλληλότερες για την εκπαίδευση βαθιών νευρωνικών δικτύων από μια CPU. Το 2016 η Google άρχισε να αναπτύσσει μονάδες ειδικά για την επεξεργασία πολυεπίπεδων πινάκων (τανυστών), οι οποίες αποκαλούνται Μονάδες Επεξεργασίας Τανυστών (Tensor Processing Units - TPUs) και οι οποίες έχουν καταφέρει πολύ χαμηλούς χρόνους εκπαίδευσης.

Οι εφαρμογές της Βαθιάς Μάθησης αφορούν αναγνώριση αντικειμένων σε εικόνες, αναγνώριση φωνής, επεξεργασία φυσικής γλώσσας, ανάλυση συναισθήματος, επεξεργασία ιατρικών εικόνων, συστήματα συστάσεων, αυτόνομα αυτοκίνητα, οικονομικές προβλέψεις κ.ά. Όλες οι παραπάνω εφαρμογές σήμερα είναι εξαιρετικά κερδοφόρες και πολλές κορυφαίες εταιρείες τεχνολογίας όπως είναι η Google, η Amazon, η NVIDIA, η Facebook [5], η Apple, η Netflix, η IBM κ.ά. χρησιμοποιούν τη Βαθιά Μάθηση στα προϊόντα ή και τις υπηρεσίες τους [1].

1.2 Σύνολα δεδομένων

Τα σύνολα δεδομένων (datasets) είναι αναπόσπαστο κομμάτι της Βαθιάς Μάθησης διότι η ίδια η μάθηση ξεκινά βρίσκοντας τα σωστά δεδομένα και κάνοντας εξάσκηση και πειραματισμούς πάνω σε αυτά. Με το πέρασμα του χρόνου τα σύνολα δεδομένων γίνονται όλο και μεγαλύτερα λόγω της ψηφιοποίησης της κοινωνίας. Σήμερα μπορούν να βρεθούν σύνολα δεδομένων με πάνω από ένα δισεκατομμύριο δείγματα με ετικέτα. Η εποχή των «Μεγάλων Δεδομένων» (Big Data) έχει δώσει τη δυνατότητα στους αλγορίθμους Βαθιάς Μάθησης να γενικεύουν πιο εύκολα σε νέα - άγνωστα δεδομένα [1]. Ωστόσο, παρά τη διευκόλυνση που έφεραν τα μεγάλα σύνολα δεδομένων, πολλοί ερευνητές έχουν στρέψει το ενδιαφέρον τους στην μελέτη προβλημάτων με λίγα διαθέσιμα δεδομένα χρησιμοποιώντας τεχνικές όπως η Μεταφορά Μάθησης (Transfer Learning).

Σε ερευνητικό και ακαδημαϊκό επίπεδο, υπάρχει ένας μεγάλος αριθμός από διαθέσιμα σύνολα δεδομένων. Τα πιο γνωστά ανά ερευνητικό πεδίο παρατίθενται παρακάτω:

- Εικόνες: MNIST, CIFAR, ImageNet
- Φυσική Γλώσσα: IMDB Reviews, Sentiment140, WordNet
- Ήχος - Ομιλία: Free Spoken Digit Dataset, Free Music Archive (FMA)

1.3 Συνελικτικά Νευρωνικά Δίκτυα

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs ή ConvNets) είναι ένα εξειδικευμένο είδος βαθιών νευρωνικών δικτύων που επεξεργάζονται δεδομένα τα οποία μπορούν να αναπαρασταθούν σε μορφή πλέγματος. Αυτό τα καθιστά κατάλληλα για αναγνώριση εικόνων αλλά και για γενικότερα προβλήματα Όρασης Υπολογιστών (Computer Vision). Η ονομασία τους προέρχεται από το γεγονός ότι ένα ή περισσότερα από τα επίπεδά τους βασίζονται στην πράξη της συνέλιξης, η οποία είναι και η κύρια λειτουργία τους [1].

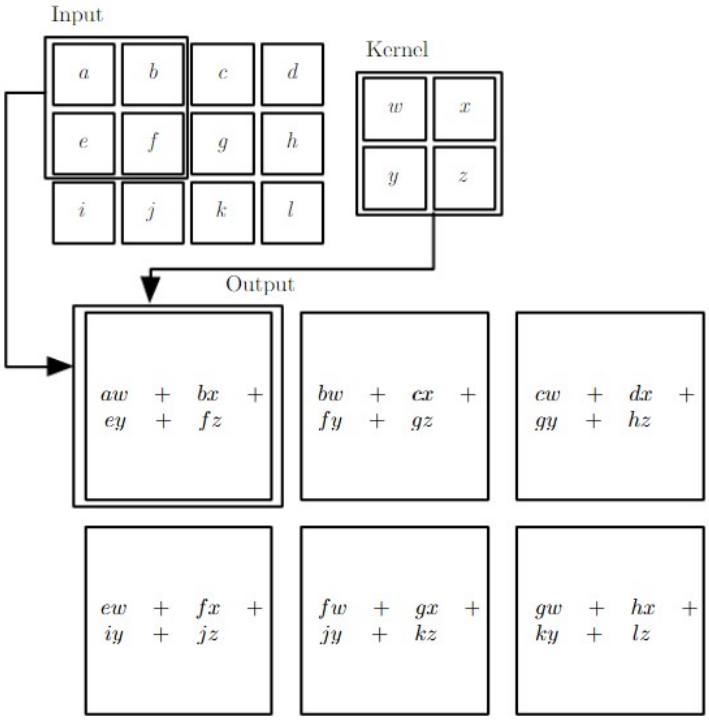
Η συνέλιξη δύο συναρτήσεων συνεχούς χρόνου f και g συμβολίζεται με $f * g$ και ορίζεται μαθηματικά από τον ακόλουθο τύπο:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Συνήθως η συνάρτηση f αναφέρεται ως η είσοδος (input) της συνέλιξης και η συνάρτηση g ως ο πυρήνας (kernel) ή το φίλτρο (filter) αυτής. Στην περίπτωση μιας δισδιάστατης εικόνας I, έχουμε δισδιάστατη συνέλιξη με τον επίσης δισδιάστατο πυρήνα K, η οποία ορίζεται ανάλογα ως εξής:

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

Ένα παράδειγμα υπολογισμού δισδιάστατης συνέλιξης φαίνεται στο Σχήμα 1.2. Τα επίπεδα που χρησιμοποιούνται σε ένα συνελικτικό νευρωνικό δίκτυο αναλύονται παρακάτω.



Σχήμα 1.2: *Παράδειγμα δισδιάστατης συνέλιξης*

Συνελικτικό Επίπεδο

Το συνελικτικό επίπεδο (convolutional layer) είναι το βασικό δομικό στοιχείο ενός συνελικτικού νευρωνικού δικτύου καθώς επιτελεί το σημαντικότερο και μεγαλύτερο υπολογιστικό μέρος. Οι παράμετροι ενός τέτοιου επιπέδου αποτελούνται από ένα σύνολο φίλτρων, τα οποία είναι μικρά σε μέγεθος σε σχέση με την είσοδο και τα οποία το δίκτυο μαθαίνει κατά την εκπαίδευση. Καθώς τα φίλτρα συνελλίσονται με την εικόνα εισόδου παράγονται οι λεγόμενοι χάρτες ενεργοποίησης (activation maps) ή χάρτες χαρακτηριστικών (feature maps),

που περιέχουν είτε χαμηλού είτε υψηλού επιπέδου χαρακτηριστικά [6]. Έτσι, ο κάθε νευρώνας εξόδου συνδέεται με μια μικρή περιοχή της εικόνας εισόδου, η οποία ονομάζεται τοπικό δεκτικό πεδίο (local receptive field) του νευρώνα. Κάθε φίλτρο εντοπίζει και από ένα διαφορετικό χαρακτηριστικό [7].

Οι παράμετροι που καθορίζουν το μέγεθος εξόδου ενός συνελικτικού επιπέδου είναι:

1. Το πλήθος των φίλτρων, που ισούται με το πλήθος των διαφορετικών χαρακτηριστικών που θα εντοπιστούν.

2. Το μέγεθος των φίλτρων, των οποίων το σχήμα συνήθως είναι τετράγωνο.

3. Ο βηματισμός (stride) με τον οποίο θα ολισθαίνουν τα φίλτρα πάνω στην εικόνα. Συνήθεις τιμές είναι 1 ή 2 pixels.

4. Το πλήθος των μηδενικων παραγεμίσματος (zero-padding) γύρω από το σύνορο της εικόνας. Αυτή η ενέργεια βοηθάει στην διατήρηση των διαστάσεων της εικόνας στην έξοδο του επιπέδου [6].

Οι παραπάνω παράμετροι συνήθως αποκαλούνται υπερπαράμετροι του επιπέδου επειδή δεν μαθαίνονται κατά την εκπαίδευση αλλά επιλέγονται από τον σχεδιαστή του δικτύου με τεχνικές βελτιστοποίησης.

Στα συνελικτικά επίπεδα χρησιμοποιείται επίσης το σχήμα διαμοιρασμού των παραμέτρων (parameter sharing), το οποίο πηγάζει από την υπόθεση ότι αν ένα χαρακτηριστικό βρεθεί σε μια συγκεκριμένη θέση, τότε είναι χρήσιμο το ίδιο χαρακτηριστικό να αναζητηθεί και σε όλες τις υπόλοιπες θέσεις [6]. Έτσι, οι παράμετροι κάθε φίλτρου είναι οι ίδιες για τον υπολογισμό οποιουδήποτε από τους νευρώνες εξόδου. Η χρήση του παραπάνω σχήματος μειώνει αρκετά τις απαιτήσεις μνήμης.

Τέλος, αξίζει να αναφερθεί ότι η δισδιάστατη συνέλιξη μπορεί να γραφεί και ως κλασικός πολλαπλασιασμός πινάκων χρησιμοποιώντας έναν πίνακα Toeplitz. Λόγω του μικρού μεγέθους των φίλτρων ένας τέτοιος πίνακας Toeplitz είναι αραιός, καθιστώντας τον υπολογισμό της συνέλιξης γρήγορο και αποδοτικό [1].

Επίπεδο Συγκέντρωσης

Η λειτουργία ενός επιπέδου συγκέντρωσης (pooling layer) είναι να συμπυκνώνει χωρικά κάθε χάρτη διατηρώντας παράλληλα τα χαρακτηριστικά του. Αυτό γίνεται αντικαθιστώντας μικρές περιοχές νευρώνων με μια στατιστική τιμή που υπολογίζεται από τις τιμές των νευρώνων αυτής της περιοχής. Δημοφιλείς συναρτήσεις είναι:

- Η μέγιστη συγκέντρωση (max pooling), που επιστρέφει τη μέγιστη τιμή από μια τετραγωνική γειτονιά.

- Η μέση συγκέντρωση (average pooling), που επιστρέφει τον μέσο όρο των τιμών μιας τετραγωνικής γειτονιάς.

- Η L^2 συγκέντρωση (L^2-norm pooling), που επιστρέφει την τετραγωνική ρίζα του αθροίσματος των τετραγώνων των τιμών μιας τετραγωνικής γειτονιάς [1].

Επίπεδο Κανονικοποίησης

Πολλοί διαφορετικοί τύποι επιπέδων κανονικοποίησης (normalization layer) έχουν προταθεί, όμως τις περισσότερες φορές η συνεισφορά τους στην απόδοση έχει αποδειχθεί ότι είναι ελάχιστη [6]. Τα εν λόγω επίπεδα συνήθως εφαρμόζουν έναν μετασχηματισμό στις ενεργοποιήσεις ώστε ο μέσος όρος των τιμών τους να διατηρείται κοντά στο 0 και η τυπική απόκλιση κοντά στο 1.

Συναρτήσεις Ενεργοποίησης

Κάθε συνάρτηση ενεργοποίησης (activation function) ή μη-γραμμικότητα (non-linearity) δέχεται έναν αριθμό και εκτελεί μια σταθερή μη-γραμμική μαθηματική πράξη σε αυτόν. Η μη-γραμμικότητα είναι απαραίτητη σε ένα πρόβλημα κατηγοριοποίησης διότι είναι αδύνατο οι κλάσεις ενός προβλήματος να εξαρτώνται γραμμικά από τα δεδομένα. Οι πιο γνωστές συναρτήσεις ενεργοποίησης είναι:

1. Η σιγμοειδής (sigmoid) με συνάρτηση $\sigma(x) = 1/(1 + e^{-x})$.
2. Η ReLU (Rectified Linear Unit) με συνάρτηση $f(x) = max(0, x)$.
3. Η υπερβολική εφαπτομένη με συνάρτηση $tanh(x) = 2\sigma(2x) - 1$ [6].

Πλήρως Συνδεδεμένο Επίπεδο

Στο τέλος ενός συνελικτικού δικτύου υπάρχουν ένα η περισσότερα πλήρως συνδεδεμένα (fully-connected) επίπεδα, των οποίων σκοπός είναι να εξάγουν την τελική πρόβλεψη για την ετικέτα της εικόνας εισόδου. Το πρώτο πλήρως συνδεδεμένο επίπεδο μετασχηματίζει τα χαρακτηριστικά που έχουν εντοπιστεί σε ένα ενιαίο διάνυσμα, ενώ το τελευταίο που έχει μήκος όσες είναι και οι κλάσεις του προβλήματος, εξάγει τις πιθανότητες για την κάθε κλάση. Τα πλήρως συνδεδεμένα επίπεδα πλην του τελευταίου συνήθως ακολουθούνται από κάποια συνάρτηση ενεργοποίησης.

1.3.1 Αρχιτεκτονικές

Η κλασική μορφή αρχιτεκτονικής ενός συνελικτικού νευρωνικού δικτύου περιλαμβάνει μερικά συνελικτικά επίπεδα με κάποια συνάρτηση ενεργοποίησης ακολουθούμενα από επίπεδα συγκέντρωσης, τα οποία επαναλαμβάνονται μέχρι το μέγεθος της εικόνας να γίνει μικρό. Από εκεί και μετά εισάγονται τα πλήρως συνδεδεμένα επίπεδα με το τελευταίο να δίνει τις προβλέψεις ανά κλάση. Με την πάροδο του χρόνου όμως, η αρχιτεκτονική των συνελικτικών δικτύων έγινε πιο περίπλοκη, ειδικά με την εισαγωγή νέων μονάδων, όπως για παράδειγμα των residual blocks η των depthwise convolution blocks. Παρακάτω αναφέρονται οι πιο γνωστές αρχιτεκτονικές συνελικτικών δικτύων:

1. LeNet (1998, Yann LeCun et al.) [8]
2. AlexNet (2012, Alex Krizhevsky) [9]
3. GoogLeNet (2014, Christian Szegedy et al.) [10]

4. VGG (2015, Karen Simonyan και Andrew Zisserman) [11]

5. Inception-v3 (2015, Christian Szegedy et al.) [12]

6. ResNet (2015, Kaiming He et al.) [13]

7. Xception (2017, François Chollet) [14]

8. ResNext (2017, Saining Xie et al.) [15]

Νέα μοντέλα που εμφανίστηκαν από το 2017 και μετά βασίζονται σε συνδυασμούς των δομικών στοιχείων των μοντέλων που αναφέρθηκαν παραπάνω. Κατά τη στιγμή συγγραφής της παρούσας διπλωματικής, στην κορυφή του πίνακα κατάταξης κατηγοριοποίησης εικόνων του συνόλου δεδομένων ImageNet βρίσκονται τα EfficientNets (2020, Mingxing Tan και Quoc V. Le) [16] και παραλλαγές τους.

Η γενική τάση μέχρι και το 2016 ήταν να δημιουργούνται όλο και μεγαλύτερα συνελικτικά δίκτυα για όλο και υψηλότερη απόδοση, ενώ παράλληλα εκείνη την περίοδο ξεκίνησαν οι πρώτες δοκιμές για την εισαγωγή της Βαθιάς Μάθησης σε συσκευές αιχμής (edge devices) και κινητές συσκευές (mobile devices). Ωστόσο, τα δίκτυα που υπήρχαν ήταν τεράστια σε σχέση με τις περιορισμένες δυνατότητες τέτοιων συσκευών, οπότε δημιουργήθηκε η ανάγκη για πιο ελαφριά (lightweight) δίκτυα με πρώτα να εμφανίζονται το SqueezeNet (2016, Forrest N. Iandola et al.) [17] και τα MobileNets (2017, Andrew G. Howard et al.) [18], τα οποία και επικράτησαν.

1.4 Βαθιά Μάθηση σε Κινητές Συσκευές

Πολλοί ερευνητές προσπαθούν τα τελευταία χρόνια να γεφυρώσουν το χάσμα που έχει δημιουργηθεί μεταξύ της έρευνας πάνω στην Βαθιά Μάθηση και πρακτικών εφαρμογών για τον τελικό χρήστη, ο οποίος συνήθως βρίσκεται μπροστά από ένα πρόγραμμα περιήγησης ιστού (web browser) ή ένα κινητό τηλέφωνο (smartphone) [19].

Λόγω της συνεχώς αυξανόμενης αλληλεπίδρασης των χρηστών με τις κινητές συσκευές, ένας μεγάλος όγκος δεδομένων σχετικά με τη συμπεριφορά, τις προτιμήσεις και τις συνήθειες των χρηστών συλλέγεται καθημερινά. Αν και αυτά τα δεδομένα αποτελούν σημαντικούς πόρους για τη Βαθιά Μάθηση, υπάρχουν πολλές προκλήσεις στην ενσωμάτωσή της σε κινητές συσκευές, όπως είναι οι περιορισμένες δυνατότητες - πόροι των κινητών συσκευών, οι υψηλές απαιτήσεις των βαθιών νευρωνικών δικτύων ή θέματα απορρήτου και ασφάλειας [20].

Η ενσωμάτωση της Βαθιάς Μάθησης σε κινητές συσκευές μπορεί να διαχωριστεί σε δύο βασικές διεργασίες, την εκπαίδευση (training) και τη συμπερασματολογία (inference). Γενικά, κατά την εκπαίδευση, ένα μεγάλο σύνολο δεδομένων χρησιμοποιείται για την προσαρμογή των βαρών του δικτύου, συνήθως μέσω αλγορίθμων κατάβασης κλίσης (gradient descent), ενώ κατά τη συμπερασματολογία ζητάμε από ένα ήδη εκπαιδευμένο δίκτυο να εξάγει μια ετικέτα ή να κάνει μια πρόβλεψη από άγνωστα δεδομένα [20].

1.4.1 Εκπαίδευση

Η εκπαίδευση ενός βαθιού νευρωνικού δικτύου με έως και εκατοντάδες εκατομμύρια παραμέτρους γίνεται συνήθως σε κέντρα δεδομένων υπολογιστικού νέφους (cloud data centers) ή χρησιμοποιώντας υπολογιστές υψηλών αποδόσεων (high-performance computers). Σε μια κινητή συσκευή η εκπαίδευση δεν είναι εφικτή διότι υπερβαίνει κατά πολύ τόσο τη διαθέσιμη μνήμη όσο και την υπολογιστική ισχύ της. Για αυτόν τον λόγο, συνήθως η εκπαίδευση του δικτύου γίνεται πριν την ενσωμάτωση του σε κάποια εφαρμογή κινητής συσκευής. Αν παρ' όλα αυτά είναι απαραίτητο η εκπαίδευση να γίνει με δεδομένα που παράγονται από την κινητή συσκευή, τότε θα πρέπει αυτά να αποστέλνονται κάθε φορά σε κάποιον απομακρυσμένο διακομιστή νέφους (cloud server). Οι λύσεις που έχουν προταθεί είναι (α) η κατανεμημένη εκπαίδευση (distributed training), (β) η ομοσπονδιακή εκπαίδευση (federated training) και (γ) μέθοδοι εκπαίδευσης με στόχο την προστασία της ιδιωτικότητας των δεδομένων του χρήστη (privacy-preserving training) [20].

1.4.2 Συμπερασματολογία

Σε αντίθεση με την εκπαίδευση, η συμπερασματολογία απαιτεί λιγότερους πόρους και άρα είναι εφικτό να εκτελεστεί σε μια κινητή συσκευή, ωστόσο, μπορεί να καταλάβει σημαντικό ποσοστό από την ενέργεια, την ισχύ και τη μνήμη της συσκευής. Λόγω αυτού, αντί της εκτέλεσης τοπικά (on-device) μπορεί να επιλεγεί και η εκτέλεση σε κάποιον διακομιστή νέφους (on-cloud) [20].

Σε αυτή την περίπτωση, το εκπαιδευμένο δίκτυο βρίσκεται σε έναν απομακρυσμένο διακομιστή και παρέχεται μια διεπαφή (interface) για την επικοινωνία του με τη συσκευή μέσω του διαδικτύου. Ο χρήστης στέλνει τα δεδομένα στον διακομιστή, αυτός εκτελεί τη συμπερασματολογία και επιστρέφει το αποτέλεσμα στην εφαρμογή. Τα πλεονεκτήματα αυτού του τρόπου είναι ότι (α) η εφαρμογή παραμένει απλή, (β) ο χρόνος και το κόστος ανάπτυξης μειώνονται δραματικά και (γ) το νευρωνικό δίκτυο που βρίσκεται στον διακομιστή μπορεί να τροποποιηθεί ανεξάρτητα από την εφαρμογή. Από την άλλη προκύπτουν τρία σοβαρά προβλήματα:

1. Η απαίτηση για συνεχή σύνδεση στο διαδίκτυο.
2. Η παραβίαση της ιδιωτικότητας του χρήστη, η οποία μπορεί να προκύψει κατά την ανταλλαγή των δεδομένων του με τον απομακρυσμένο διακομιστή.
3. Η επιπρόσθετη χρονική καθυστέρηση και η ενδεχόμενη αναξιοπιστία του διαύλου επικοινωνίας, οι οποίες μπορεί να έχουν καταστροφικές συνέπειες σε εφαρμογές πραγματικού χρόνου (real-time) [20].

Οι παραπάνω τρεις λόγοι είναι σοβαροί και έχουν κάνει τους ερευνητές να στρέψουν το ενδιαφέρον τους κυρίως στην τοπική εκτέλεση, όπου τα εκπαιδευμένα βάρη του δικτύου βρίσκονται στην εφαρμογή και δεν υπάρχει καμία αλληλεπίδραση με άλλες συσκευές. Οι πιο γνωστές πλατφόρμες που χρησιμοποιούνται σήμερα για συμπερασματολογία στην συσκευή (on-device inference) είναι οι:

- Tensorflow Lite από τη Google

- Caffe2 από το Facebook
- Core ML από την Apple
- Snapdragon neural processing SDK από την Qualcomm
- Mobile AI Compute Engine (MACE) από την Xiaomi

Με την τοπική εκτέλεση τα τρία βασικά προβλήματα της εκτέλεσης σε διακομιστή νέφους εξαλείφονται, όμως όπως αναφέρθηκε και παραπάνω μπορεί να εξαντληθεί σημαντικό ποσοστό από τους πόρους της κινητής συσκευής [20]. Επίσης, συνήθως τα μοντέλα που προορίζονται για κινητές συσκευές έχουν χαμηλότερη ορθότητα (accuracy). Αν θέλαμε να κάνουμε μια γενική σύγκριση των δύο μεθόδων, θα προέκυπτε ο Πίνακας 1.1.

	Αξιοπιστία	**Ιδιωτικότητα**	**Ορθότητα**
On-cloud	Χαμηλή	Χαμηλή	Υψηλή
On-device	Υψηλή	Υψηλή	Χαμηλή

Πίνακας 1.1: *Σύγκριση της on-cloud και on-device εκτέλεσης*

Για την αποφυγή κατασπατάλησης μεγάλου ποσοστού πόρων κατά την τοπική εκτέλεση (α) άρχισε να χρησιμοποιείται υλικό όπως είναι οι GPUs ή οι DSPs ώστε να εκτελούνται ορισμένες λειτουργίες πιο αποτελεσματικά από ό,τι σε μια CPU γενικότερου σκοπού (hardware acceleration) και (β) αναπτύχθηκαν αρκετοί τρόποι προσαρμογής και μέθοδοι βελτιστοποίησης των βαθιών νευρωνικών δικτύων. Αυτοί οι δύο τρόποι αναλύονται περισσότερο ακολούθως. Ένα εργαλείο που χρησιμοποιείται για την μέτρηση της απόδοσης κινητών συσκευών σε διάφορα Deep Learning tasks είναι το AI Benchmark [21], ενώ ενδιαφέροντα συμπεράσματα εξάγονται και από το Embench [22] σχετικά με το ποια μοντέλα ευδοκιμούν κάτω από ποιες πλατφόρμες, προσδιορίζοντας τα σημεία συμφόρησης (bottlenecks) και τις πηγές αναποτελεσματικότητας τους.

Hardware acceleration

Η επιλογή της υπολογιστικής μονάδας που θα αναλάβει να εκτελέσει τη συμπερασματολογία εξαρτάται από πολλούς παράγοντες, όπως είναι η απόδοση, το μέγεθος, το βάρος, το κόστος ανάπτυξης ή η ενέργεια που αυτή καταναλώνει.

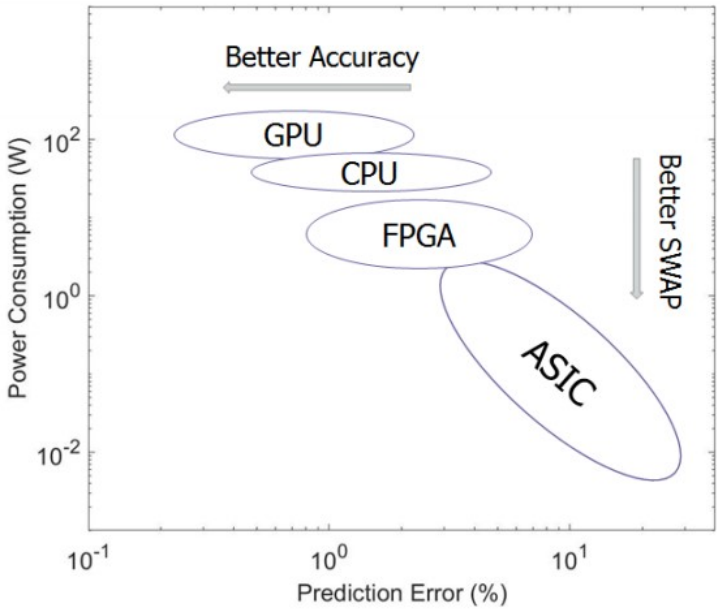
Στην αρχή χρησιμοποιήθηκαν κατά κόρον οι CPUs και οι GPUs επειδή είναι ευέλικτες και διαθέσιμες σε κάθε κινητή συσκευή, οπότε δεν ήταν αναγκαία η κατασκευή κάποιου πρόσθετου υλικού. Αν αξιοποιηθούν πολυπύρηνες CPUs και γίνει εκμετάλλευση πολλαπλών νημάτων (threads) για παράλληλους υπολογισμούς, τότε η απόδοσή τους είναι ικανοποιητική μερικές φορές. Οι GPUs είναι από τη φύση τους κατασκευασμένες για παράλληλη επεξεργασία και επαναλαμβανόμενους υπολογισμούς, οπότε συνήθως προτιμώνται για τη συμπερασματολογία βαθιών νευρωνικών δικτύων.

Ωστόσο, οι παραπάνω υπολογιστικές μονάδες δεν κατασκευάστηκαν για αυτόν τον σκοπό και συνήθως χρησιμοποιούνται από την κινητή συσκευή για άλλες λειτουργίες, οπότε η συμπερασματολογία είναι επιπρόσθετος φόρτος. Για αυτόν τον λόγο τα τελευταία χρόνια έχουν αναπτυχθεί ιδιαίτερα οι NPUs (Neural Processing Units), οι οποίες έγκεινται στα

ASICs (Application-Specific Integrated Circuits) και οι οποίες μπορούν να έχουν μικρότερα μέγεθος, βάρος και κατανάλωση ενέργειας, όμως μεγαλύτερο χρόνο κατασκευής. Από το 2017 και μετά, όλοι οι μεγάλοι κατασκευαστές ολοκληρωμένων κυκλωμάτων (chips) για κινητές συσκευές ενσωματώνουν σε αυτά NPUs αφιερωμένες στην Μηχανική Μάθηση.

Κάπου στην μέση των CPUs και των ASICs βρίσκονται τα FPGAs, τα οποία μπορούν να επαναπρογραμματιστούν από υλικολογισμικό για την εκτέλεση διαφόρων συγκεκριμένων λειτουργιών αποδοτικά. Τα πιο σύγχρονα FPGAs τείνουν στην SoC (System on a Chip) προσέγγιση με σκοπό να ενσωματώσουν λειτουργίες επεξεργασίας, επικοινωνίας και μνήμης σε ένα μοναδικό ολοκληρωμένο κύκλωμα.

Αν συγκρίνουμε όλες τις παραπάνω μονάδες, θα δούμε ότι οι CPUs και GPUs είναι πιο ευέλικτες και μπορούν να υποστηρίξουν υπολογιστικά έντονα μοντέλα, όμως είναι λιγότερο αποδοτικές σε θέματα ενέργειας και φυσικών χαρακτηριστικών. Τα ASICs έχουν πολύ χαμηλή κατανάλωση ενέργειας όμως η ανάπτυξή τους είναι αρκετά χρονοβόρα και τα FPGAs βρίσκονται κάπου στην μέση. Στο Σχήμα 1.3 φαίνεται ένα πρόχειρο διάγραμμα των επιδόσεων των τεσσάρων αρχιτεκτονικών [23]. Ο όρος SWAP αναφέρεται στο μέγεθος (size), το βάρος (weight), την επιφάνεια (area) και την κατανάλωση ενέργειας (power) κάθε αρχιτεκτονικής.



Σχήμα 1.3: *Απόδοση λειτουργίας των τεσσάρων αρχιτεκτονικών υλικού*

Βελτιστοποίηση

Εκτός από την πρόοδο στο υλικό, παράλληλα έχει γίνει και σημαντική πρόοδος στο λογισμικό με την έννοια μεθόδων βελτιστοποίησης βαθιών νευρωνικών δικτύων. Η πιο σημαντική είναι η κβαντοποίηση (quantization) που αφορά την ακρίβεια ή ισοδύναμα τον αριθμό των bits με τα οποία θέλουμε να αναπαραστήσουμε τα βάρη ενός εκπαιδευμένου δικτύου. Η προκαθορισμένη τιμή για τα βαθιά νευρωνικά δίκτυα συνήθως είναι τα 32 bits για αριθμούς κινητής υποδιαστολής, όμως φαίνεται ότι μπορούν να λειτουργήσουν και με λιγότερα, όπως για παράδειγμα με ακέραιους αριθμούς των 8 bits, με μειωμένη ωστόσο ακρίβεια. Τα πλε-

ονεκτήματα της κβαντοποίησης είναι ότι η αριθμητική με λιγότερα bits είναι πιο γρήγορη, οι απαιτήσεις μνήμης και αποθήκευσης μειώνονται δραματικά, λιγότερα bits σημαίνουν ότι τα περισσότερα βάρη μπορούν να αποθηκευθούν στις ίδιες κρυφές μνήμες - καταχωρητές, μειώνοντας έτσι τη συχνότητα πρόσβασης στη μνήμη.

Το παραπάνω λειτουργεί διότι τα βαθιά δίκτυα, εφόσον εκπαιδευτούν, είναι σχετικά εύρωστα σε θόρυβο ή μικρές διαταραχές και επίσης τα βάρη στην έξοδο ενός συγκεκριμένου δικτύου βρίσκονται σε ένα σχετικά μικρό εύρος τιμών, το οποίο μάλιστα μπορεί να προβλεφθεί. Αν τελικά η κβαντοποίηση γίνει σωστά, τότε η απώλεια ορθότητας θα είναι πολύ μικρή ή και ελάχιστη αν το δίκτυο εκπαιδευτεί ξανά ώστε να προσαρμοστεί στην κβαντοποίηση [24].

1.5 Αντικείμενο Διπλωματικής Εργασίας

Αντικείμενο της διπλωματικής εργασίας είναι η διερεύνηση του τρόπου με τον οποίο η κάμερα μιας κινητής συσκευής και ένα μοντέλο Βαθιάς Μάθησης για αναγνώριση εικόνας σε πραγματικό χρόνο μπορούν να βρίσκονται σε ταυτόχρονη χρήση. Πιο συγκεκριμένα, μελετάται η περίπτωση κατά την οποία η συμπερασματολογία του μοντέλου γίνεται τοπικά (on-device) και οι εικόνες που αυτό αναγνωρίζει δεν είναι άλλες παρά καρέ (frames) από την κάμερα.

Το παραπάνω γίνεται χρησιμοποιώντας μια Android εφαρμογή, η οποία σχεδιάστηκε και αναπτύχθηκε στα πλαίσια της εργασίας. Αυτή τη στιγμή, το λειτουργικό σύστημα Android καταλαμβάνει μερίδιο που ανέρχεται στο 74.43 % της παγκόσμιας αγοράς [25], οπότε τα αποτελέσματα που προκύπτουν από αυτή την εργασία αντιπροσωπεύουν ένα μεγάλο ποσοστό χρηστών.

Μέσα από την εφαρμογή καταμετράται η απόδοση του ολοκληρωμένου συστήματος κάμερας - αναγνώρισης και στη συνέχεια εξάγονται τα απαραίτητα συμπεράσματα. Η εργασία στοχεύει σε μια συγκριτική προτυποποίηση (benchmarking) του συστήματος στα πλαίσια διαφορετικών μοντέλων Βαθιάς Μάθησης, διαφορετικών κινητών συσκευών και διαφορετικών επεξεργαστών.

Κεφάλαιο 2

Τεχνολογίες

Στο δεύτερο Κεφάλαιο περιγράφονται οι τεχνολογίες - τα εργαλεία που αξιοποιήθηκαν στα πλαίσια της εργασίας, από το λειτουργικό σύστημα της εφαρμογής μέχρι το περιβάλλον ανάπτυξης αυτής.

2.1 Android

Το Android είναι ένα λειτουργικό σύστημα ανοιχτού-κώδικα (open-source) βασισμένο στον πυρήνα του Linux και σχεδιασμένο για να τρέχει σε κινητές συσκευές. Κάθε εταιρεία κινητών συσκευών που χρησιμοποιεί το Android μπορεί να το τροποποιήσει και να το προσαρμόσει στις δικές της συσκευές.

Η αρχή έγινε το 2005 όταν η Google αποφάσισε να αγοράσει την αρχική εταιρεία Android Inc και να χρησιμοποιήσει τον πυρήνα Linux ως βάση. Η πρώτη δημόσια παρουσίαση του λειτουργικού συστήματος για προγραμματιστές έγινε τον Νοέμβριο του 2007 μαζί με την ανακοίνωση της ίδρυσης της Open Handset Alliance, μιας κοινοπραξίας πολλών εταιριών τηλεπικοινωνίας, εταιριών λογισμικού καθώς και κατασκευής υλικού, η οποία θα αναλάμβανε από εκεί και έπειτα την ανάπτυξη του λογισμικού.

Κάθε χρόνο η Google αναβαθμίζει το Android βγάζοντας νέες εκδόσεις με βελτιώσεις και νέες δυνατότητες. Τον Απρίλιο του 2009 μαζί με την κυκλοφορία της έκδοσης 1.5, ξεκίνησε και η παράδοση ονομασίας των εκδόσεων με γλυκά ή επιδόρπια (π.χ. Cupcake, Eclair, Gingerbread, KitKat, Marshmallow, Pie), η οποία παρέμεινε μέχρι τον Σεπτέμβριο του 2019, όπου η Google αποφάσισε να ανανεώσει ριζικά τη μάρκα Android. Έτσι, η επόμενη έκδοση, δηλαδή η έκδοση Q, έγινε γνωστή απλά ως Android 10. Η πιο πρόσφατη έκδοση είναι η Android 11 που κυκλοφόρησε τον Σεπτέμβριο του 2020. Σήμερα, το Android είναι το πιο δημοφιλές λειτουργικό σύστημα κινητών συσκευών παγκοσμίως με μοναδικό ανταγωνιστή το λειτουργικό σύστημα iOS της Apple [26].

2.2 Android Studio

Το Android Studio είναι το επίσημο Ολοκληρωμένο Προγραμματιστικό Περιβάλλον (Integrated Development Environment - IDE) για την ανάπτυξη Android εφαρμογών και είναι βασισμένο στο IntelliJ IDEA [27]. Ανακοινώθηκε τον Μάιο του 2013 στο Google I/O και η πρώτη έκδοση κυκλοφόρησε τον Δεκέμβριο του 2014. Μεταξύ άλλων, προσφέρει στον προ-

γραμματιστή τη δυνατότητα ανάπτυξης εφαρμογών για κάθε συσκευή Android, έναν γρήγορο και με πολλά χαρακτηριστικά εξομοιωτή, εκτενή εργαλεία δοκιμών και μέτρησης απόδοσης και διασύνδεση με το Github για εύκολη μεταφορά και αποθήκευση του κώδικα [28]. Παραδοσιακά, η γλώσσα προγραμματισμού που χρησιμοποιούνταν στο Android Studio ήταν η Java, όμως από τον Μάιο του 2019, προτεινόμενη γλώσσα άρχισε να είναι η Kotlin.

2.3 Java

Η Java είναι μια σύγχρονη, ισχυρή, αντικειμενοστραφής γλώσσα προγραμματισμού γενικού σκοπού. Σε αντίθεση με άλλες γλώσσες, ο μεταγλωττιστής της Java μετατρέπει τον κώδικα σε bytecode, το οποίο στην συνέχεια διερμηνεύεται από το Java Runtime Environment (JRE) ή τη Java Virtual Machine (JVM), τα οποία δρουν σαν ένας εικονικός υπολογιστής που μεταφράζει το bytecode σε κάθε υπολογιστή. Αποτέλεσμα είναι ο κώδικας να μπορεί να γράφεται με τον ίδιο τρόπο για πολλές πλατφόρμες (write once, run anywhere) [29].

Η ανάπτυξη της Java ξεκίνησε με την ονομασία Oak το 1991 από τη Sun Microsystems, ενώ το 1995 έλαβε την επίσημη ονομασία της. Το πρωταρχικό κίνητρο ήταν η ανάγκη για μια γλώσσα ανεξάρτητη πλατφόρμας, η οποία θα μπορούσε να χρησιμοποιηθεί για τη δημιουργία λογισμικού για διάφορες ηλεκτρονικές συσκευές, όπως τοστιέρες, φούρνους μικροκυμάτων ή τηλεχειριστήρια. Το πρόβλημα ήταν ότι εκείνη την εποχή οι περισσότερες γλώσσες προγραμματισμού ήταν σχεδιασμένες να μεταγλωττίζονται σε γλώσσα μηχανής που στόχευε σε έναν συγκεκριμένο τύπο CPU, οπότε έπρεπε η κάθε συσκευή να έχει τον δικό της μεταγλωττιστή. Όμως, οι μεταγλωττιστές είναι ακριβοί και η κατασκευή τους χρονοβόρα, άρα τελικά η λύση που προτάθηκε ήταν η δημιουργία μια φορητής, ανεξάρτητης πλατφόρμας γλώσσας, η οποία θα δημιουργούσε κώδικα ο οποίος θα έτρεχε σε μια πληθώρα από CPUs.

Παρ' όλα αυτά, η ομάδα που εργαζόταν για την ανάπτυξη της Java άλλαξε γρήγορα κατεύθυνση διότι εκείνη την εποχή άρχισε να αναπτύσσεται παράλληλα και ο Παγκόσμιος Ιστός (World Wide Web), ο οποίος είχε επίσης ανάγκη από φορητά προγράμματα. Μέσα από αυτή την αλλαγή κατεύθυνσης, η Java έπλασε τη σημερινή μορφή του Διαδικτύου τόσο από πλευράς πελάτη (client), όσο και από πλευράς εξυπηρετητή (server). Πολλοί θα έλεγαν ότι ο τρόπος λειτουργίας της Java ήταν επαναστατικός και ότι γενικά στην Επιστήμη των Υπολογιστών, λίγες τεχνολογίες είχαν το αντίκτυπο που είχε η Java [30].

Τον Νοέμβριο του 2006 ένα μεγάλο μέρος της Java ξεκίνησε να διατίθεται ως λογισμικό ανοιχτού κώδικα, ενώ μέχρι τον Μάιο του 2007 αυτή η διεργασία είχε ολοκληρωθεί για ολόκληρο τον κώδικα πυρήνα της Java. Το 2009 η Oracle αγόρασε τη Sun Microsystems και για πολλά χρόνια η Java ήταν η πιο δημοφιλής και πιο χρησιμοποιούμενη γλώσσα προγραμματισμού μέχρι το 2015 όπου άρχισε να αυξάνεται σε δημοσιότητα η γλώσσα Python, η οποία σήμερα έχει αντικαταστήσει τη Java σε δημοτικότητα [31]. Σήμερα η Java χρησιμοποιείται σε εφαρμογές κινητών συσκευών, ιστού, επιτραπέζιων υπολογιστών, παιχνιδιών, επιχειρήσεων, σε κατανεμημένες εφαρμογές, εφαρμογές νέφους, κ.α. Νέες εκδόσεις κυκλοφορούν κάθε Μάρτιο και Σεπτέμβριο, με την πιο πρόσφατη να είναι η Java SE 15 που κυκλοφόρησε τον Σεπτέμβριο του 2020.

2.4 TensorFlow Lite

Το TensorFlow Lite είναι μια ολοκληρωμένη, ανεξάρτητη πλατφόρμας, έτοιμη για προϊόντα και ανοιχτού κώδικα δομή Βαθιάς Μάθησης (Deep Learning framework) που δίνει τη δυνατότητα σε εκπαιδευμένα μοντέλα TensorFlow να ενσωματωθούν αποτελεσματικά σε συσκευές αιχμής, όπως είναι οι κινητές συσκευές ή οι μικροελεγκτές, για συμπερασματολογία στην συσκευή (on-device) ή στην αιχμή (at the edge).

Τα βασικά συστατικά του TensorFlow Lite είναι δύο:

1. Ο TensorFlow Lite converter, του οποίου η κύρια λειτουργία είναι να μετατρέπει τα TensorFlow μοντέλα σε FlatBuffers μειώνοντας το μέγεθός τους χωρίς να μειώνεται παράλληλα και η ορθότητά τους. Ταυτόχρονα ο converter μπορεί να εισάγει βελτιστοποιήσεις, όπως είναι η κβαντοποίηση ή το κλάδεμα (pruning) των παραμέτρων ώστε να μειωθεί περαιτέρω το μέγεθος ή και ο χρόνος εκτέλεσης.

2. Ο TensorFlow Lite interpreter, ο οποίος αναλαμβάνει την εκτέλεση των light-weight μοντέλων που παράγει ο converter σε πολλούς και διαφορετικούς τύπους υλικού. Για να μειωθεί περαιτέρω ο χρόνος εκτέλεσης, ο interpreter μπορεί να χρησιμοποιήσει «εκπροσώπους» (delegates), οι οποίοι αποστέλλουν μέρη από το δίκτυο σε κάποιον επιταχυντή, όπως είναι οι GPUs ή οι NPUs για υψηλότερη απόδοση [32] [33].

Κεφάλαιο 3

Αρχιτεκτονική Εφαρμογής

Στο τρίτο Κεφάλαιο περιγράφεται η αρχιτεκτονική - διαστρωμάτωση του ολοκληρωμένου συστήματός και αναλύονται τα χαρακτηριστικά και οι δυνατότητες των επιμέρους επιπέδων, αλλά και η διασύνδεση μεταξύ τους.

3.1 Γενικές αρχές

Τα βασικά χαρακτηριστικά που έπρεπε να ληφθούν υπόψη κατά τη σχεδίαση του συστήματος ήταν δύο: η ανεξαρτησία από τη συσκευή (device independence) και η ανεξαρτησία από το μοντέλο (model independence).

Η ανεξαρτησία από τη συσκευή είναι ένα σχετικά πρόσφατο χαρακτηριστικό των συστημάτων που παλιότερα δεν υπήρχε στην Επιστήμη των Υπολογιστών καθώς για πολλές δεκαετίες τα προγράμματα ήταν αρκετά εξαρτώμενα από τη συσκευή ή το λειτουργικό σύστημα στο οποίο έτρεχαν. Στην δική μας περίπτωση, αυτό το χαρακτηριστικό σχετίζεται κυρίως με την έκδοση Android των κινητών συσκευών και με το υλικό τους.

Με την κυκλοφορία νέων εκδόσεων κάθε χρόνο, το Android αναβαθμίζεται και οι εφαρμογές οφείλουν να προσαρμόζονται στις αλλαγές ώστε να λειτουργούν σωστά σε όλες τις συσκευές. Αυτή τη στιγμή παγκοσμίως το 33.29% των Android χρηστών χρησιμοποιεί την έκδοση 10, το 26.09% την έκδοση 9 (Pie), το 10.96% την έκδοση 8.1 (Oreo), ενώ το 7.41% χρησιμοποιεί ακόμη την έκδοση 6.0 (Marshmallow) [34].

Το Android σαν λειτουργικό σύστημα χρησιμοποιείται από πολλούς και διαφορετικούς κατασκευαστές με αποτέλεσμα κάθε κινητή συσκευή Android να έχει διαφορετικά τεχνικά χαρακτηριστικά όπως κάμερες, οθόνες, CPUs, GPUs, NPUs κ.α. Το σύστημα πρέπει να λαμβάνει υπόψη τα χαρακτηριστικά της εκάστοτε συσκευής και να προσαρμόζει την εφαρμογή σε αυτά.

Η ανεξαρτησία από το μοντέλο αναφέρεται στη δυνατότητα του συστήματος να λειτουργεί σωστά με μια πληθώρα από μοντέλα Βαθιάς Μάθησης ή ακόμη και χωρίς, όπου φυσικά τότε θα λείπει από την εφαρμογή η λειτουργία της αναγνώρισης σε πραγματικό χρόνο, όμως όλες οι άλλες λειτουργίες θα υπάρχουν.

Με βάση τα παραπάνω, κρίθηκε απαραίτητος ο διαχωρισμός του συστήματος σε δύο επίπεδα - στρώματα: το Ανεξάρτητο Υπηρεσίας Επίπεδο (**S**ervice **I**ndependent **L**ayer - SIL) και το Επίπεδο Σύγκλισης (**C**onvergence **L**ayer - CL), τα οποία αναλύονται παρακάτω.

3.2 Το Ανεξάρτητο Υπηρεσίας Επίπεδο (SIL)

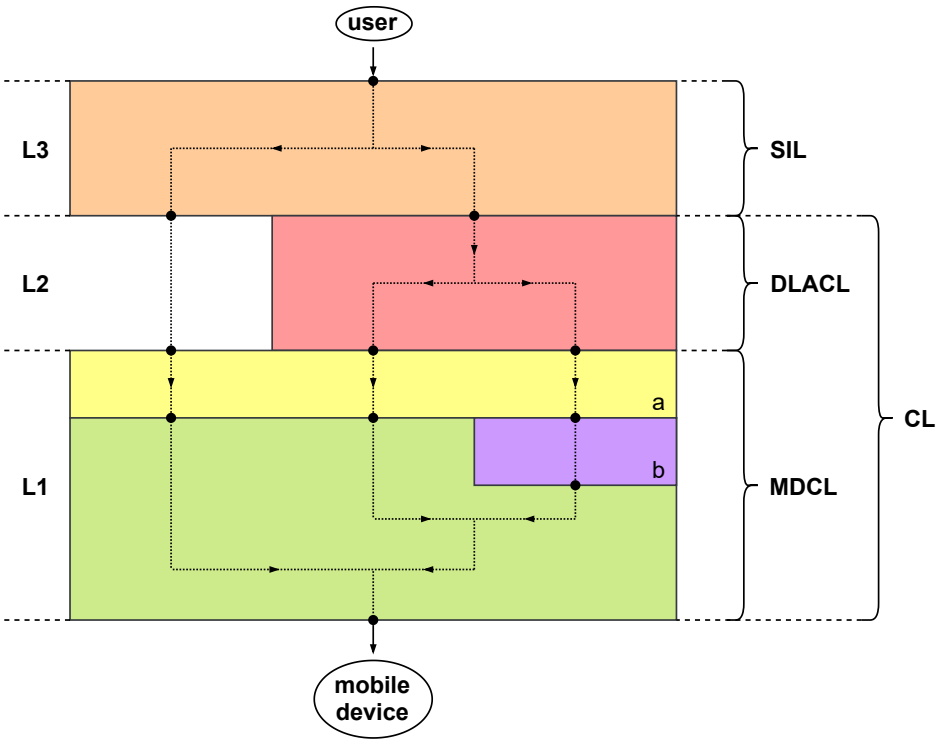
Το SIL είναι το πρώτο επίπεδο που αλληλεπιδρά με τον χρήστη και παρέχει την κυρίως λειτουργικότητα του συστήματος ανεξάρτητα από την κινητή συσκευή και το μοντέλο, παρέχοντας με αυτόν τον τρόπο αρθρωτότητα (modularity), ευελιξία (flexibility) και συντηρησιμότητα (maintainability). Έτσι, το σύστημα έχει μια σπονδυλωτή δομή και μπορεί εύκολα να διαχωριστεί σε διακριτά μέρη, τα οποία είναι αρκετά ανεξάρτητα μεταξύ τους.

3.3 Το Επίπεδο Σύγκλισης (CL)

Το CL λαμβάνει στην συνέχεια υπόψη του τόσο τα χαρακτηριστικά της συσκευής όσο και τα χαρακτηριστικά του μοντέλου και πραγματοποιεί τις απαραίτητες προσαρμογές. Για αυτό, μπορεί να διαιρεθεί σε δύο υποεπίπεδα, το Επίπεδο Προσαρμογής Αρχιτεκτονικής Βαθιάς Μάθησης (**D**eep **L**earning **A**rchitecture **C**onvergence **L**ayer - DLACL) και το Επίπεδο Προσαρμογής Κινητής Συσκευής (**M**obile **D**evice **C**onvergence **L**ayer - MDCL). Οι λειτουργίες του πρώτου εξαρτώνται από το εκάστοτε μοντέλο, ενώ οι λειτουργίες του δεύτερου από την εκάστοτε συσκευή.

Ολοκληρωμένη αρχιτεκτονική

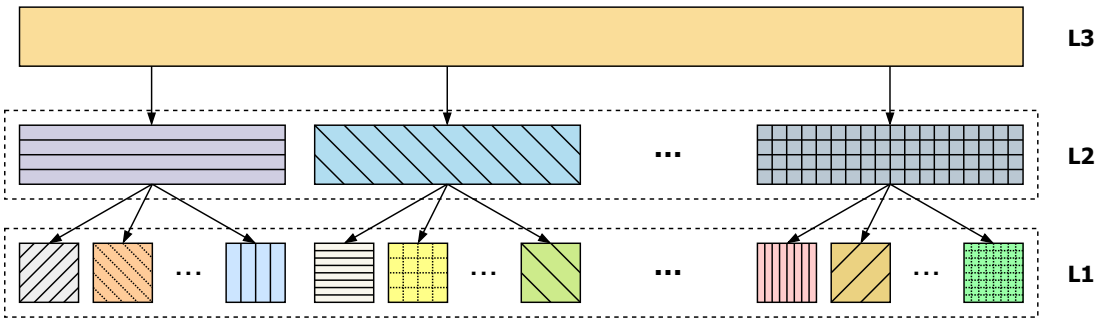
Στο Σχήμα 3.1 φαίνεται ολόκληρη η διαστρωμάτωση του συστήματος. Για λόγους απλότητας και ομοιομορφίας, το MDCL θα αναφέρεται και ως **L**ayer **1** (L1), το DLACL ως **L**ayer **2** (L2) και το SIL ως **L**ayer **3** (L3).



Σχήμα 3.1: *Διαστρωμάτωση συστήματος*

Όπως είναι εμφανές, η ύπαρξη του 2ου επιπέδου δεν είναι υποχρεωτική, καθώς όπως αναφέρθηκε το σύστημα θα πρέπει να μπορεί να λειτουργεί και χωρίς αυτό. Το middleware a προσφέρει λειτουργίες οι οποίες είναι απαραίτητες είτε το σύστημα εκτελεί αναγνώριση είτε όχι, ενώ το middleware b μπορεί να προσφέρει επιπλέον λειτουργίες με βάση το υλικό εκμεταλλευόμενο το αποτέλεσμα της αναγνώρισης. Μια πιο αναλυτική σχεδίαση του συστήματος περιγράφεται στο Κεφάλαιο 4.

Η αρχιτεκτονική του συστήματος είναι σπονδυλωτή, όπως ήταν επιθυμητό, κι αυτό φαίνεται καλύτερα στο Σχήμα 3.2, στο οποίο εξετάζουμε την περίπτωση πολλών διαφορετικών μοντέλων και συσκευών. Το L3 παρέχει λειτουργίες για οποιαδήποτε μοντέλο και συσκευή, το L2 για συγκεκριμένο μοντέλο αλλά για πολλές συσκευές και το L1 για συγκεκριμένο μοντέλο και συγκεκριμένη συσκευή. Το L2 φυσικά μπορεί είτε να υπάρχει, είτε όχι.



Σχήμα 3.2: *Αρχιτεκτονική συστήματος στην περίπτωση πολλών μοντέλων και συσκευών*

Κεφάλαιο 4

Αναλυτική Σχεδίαση

Στο τέταρτο Κεφάλαιο αναλύονται τα βασικά δομικά στοιχεία της Android εφαρμογής και συγκεκριμένα ποια είναι τα χαρακτηριστικά τους, πώς συνδέονται μεταξύ τους, πως το ένα εξαρτάται από το άλλο και πώς εμπίπτουν στην αρχιτεκτονική του προηγούμενου Κεφαλαίου. Επίσης, περιγράφονται οι λειτουργίες που προσφέρει η εφαρμογή στον χρήστη και παρατίθενται στιγμιότυπα από την οθόνη της.

4.1 Περιγραφή εφαρμογής

Η εφαρμογή, όπως αναφέρθηκε και παραπάνω, είναι μια «έξυπνη» κάμερα, η οποία αναγνωρίζει σε πραγματικό χρόνο οτιδήποτε βρίσκεται μπροστά από τον κεντρικό φακό της κινητής συσκευής. Τα κύρια συστατικά στοιχεία της εφαρμογής είναι δύο: η κάμερα και ο ταξινομητής.

4.1.1 Κάμερα

Για τη δημιουργία της κάμερας χρησιμοποιήθηκε το Camera2 API, το οποίο παρουσιάστηκε μαζί με την έκδοση 5.0 (Lollipop) του Android τον Νοέμβριο του 2014. Στόχος του συγκεκριμένου API ήταν να προσφέρει στους προγραμματιστές προηγμένες χρήσεις της κάμερας, όπως είναι ο έλεγχος της έκθεσης, η χειροκίνητη εστίαση ή η λήψη φωτογραφιών σε μορφή RAW. Ωστόσο, παρά τα πλεονεκτήματα, πρόκειται για ένα αρκετά περίπλοκο και εκ πρώτης όψης δυσνόητο API, ειδικά σε σύγκριση με το παλιό Camera API που χρησιμοποιούνταν μέχρι τότε, καθώς το νέο έφερε μαζί του δεκάδες κλάσεις, νέες έννοιες και μια σωλήνωση (pipeline) με πολλές ασύγχρονες κλήσεις (callbacks) που προκαλούν σύγχυση. Εκτός από αυτά, οποιοδήποτε API σχετίζεται άμεσα με το υλικό είναι απαιτητικό.

Αξίζει να αναφερθεί ότι η δυνατότητα εκτέλεσης του Camera2 API σε κάθε συσκευή εξαρτάται αποκλειστικά από τον κατασκευαστή. Αυτό σημαίνει πως δεν υπάρχει εγγύηση ότι το συγκεκριμένο API μπορεί να χρησιμοποιηθεί σε όλες τις συσκευές με έκδοση Android 5.0 και πάνω. Για αυτόν τον λόγο κάθε συσκευή μπορεί να χρησιμοποιεί το Camera2 API σε 4 επίπεδα: το LEGACY, που υποστηρίζει μόνο τις λειτουργίες του παλιού Camera API, το LIMITED, το οποίο υποστηρίζει κάποιες απλές λειτουργίες, το FULL που υποστηρίζει όλες τις καθιερωμένες λειτουργίες του νέου API και το LEVEL 3, στο οποίο προστίθενται επιπλέον λειτουργίες, όπως είναι η λήψη φωτογραφιών σε μορφή RAW [35].

Όταν ένας χρήστης ανοίγει μια Android εφαρμογή, δημιουργείται μια νέα διεργασία Linux μαζί με το κεντρικό νήμα εκτέλεσης (main - execution thread), το οποίο είναι υπεύθυνο για οτιδήποτε συμβαίνει στην οθόνη, γι' αυτό συχνά αναφέρεται και ως User Interface ή UI thread. Όλα τα κομμάτια κώδικα που πρέπει να εκτελεστούν στο main thread τοποθετούνται πρώτα στην ουρά μηνυμάτων αυτού. Στην περίπτωση που αυτή η ουρά μηνυμάτων περιέχει διεργασίες οι οποίες είναι είτε πάρα πολλές είτε πολύ χρονοβόρες για να εκτελεστούν στο main thread, θα πρέπει να χρησιμοποιούνται άλλα νήματα που ονομάζονται νήματα εργάτη (worker threads), ώστε να μην επηρεάζεται η ομαλή απόδοση και η ταχεία απόκριση στις εισόδους του χρήστη [36].

Με βάση τα παραπάνω και επειδή οι λειτουργίες του Camera2 API είναι χρονοβόρες, χρησιμοποιούμε στην εφαρμογή ένα worker thread, το οποίο ονομάζουμε background thread και το οποίο αναλαμβάνει όλες τις λειτουργίες της κάμερας, καθώς και όλες τις λειτουργίες του συστήματος αναγνώρισης όπως θα περιγραφεί παρακάτω.

Είναι λογικό ότι η ανάπτυξη μιας εφαρμογής κάμερας εξαρτάται από την κινητή συσκευή, δηλαδή από χαρακτηριστικά όπως είναι ο τύπος του αισθητήρα ή των αισθητήρων, οι διαθέσιμες διαστάσεις λήψης φωτογραφιών ή βίντεο από τους αισθητήρες, η ύπαρξη η μη φλας, το μέγιστο ποσοστό μεγέθυνσης (zoom), κ.α. Όλες αυτές οι πληροφορίες που είναι διαφορετικές από συσκευή σε συσκευή παρέχονται από το middleware a του L1 προς το L3, το οποίο τελικά αναπτύσσει την κάμερα για την εκάστοτε συσκευή (βλ. Σχήμα 3.1).

4.1.2 Ταξινομητής

Ο ταξινομητής (classifier) ανήκει στο L2, ένα επίπεδο εντελώς ανεξάρτητο, αφού δεν απαιτείται καμία ανταλλαγή πληροφορίας ούτε από την κάμερα, ούτε και από τη συσκευή. Στην αρχιτεκτονική του Σχήματος 3.1 ωστόσο, παρατηρείται και το middleware b, το οποίο αν και δεν αξιοποιείται στην παρούσα εργασία, ενσωματώθηκε ως επέκταση - μελλοντική εργασία. Η ιδέα είναι ότι σε μια πιο ολοκληρωμένη εφαρμογή τα χαρακτηριστικά του pipeline της κάμερας θα βελτιστοποιούνται με βάση το αποτέλεσμα της αναγνώρισης (βλ. Ενότητα 7.2).

Για την ανάπτυξη του ταξινομητή χρησιμοποιήθηκε το TensorFlow Lite. Μαζί με τη δημιουργία του background thread, δημιουργείται και ο Interpreter για το εκάστοτε μοντέλο Βαθιάς Μάθησης, στον οποίο πρέπει επίσης να προσδιοριστεί πού θα εκτελείται η συμπερασματολογία. Οι επιλογές είναι τρεις: η CPU, η GPU ή κάποιος άλλος επιταχυντής (DSP ή NPU).

Με την πρώτη επιλογή δίνεται επίσης η δυνατότητα καθορισμού του αριθμού των παράλληλων νημάτων (threads) τα οποία θα χρησιμοποιηθούν από τη CPU. Η δεύτερη επιλογή απαιτεί τα βάρη των μοντέλων που χρησιμοποιούνται να είναι κινητής υποδιαστολής (floating point), δηλαδή να αναπαριστώνται από 32 bits, άρα δεν μπορεί να χρησιμοποιηθεί για κβαντισμένα μοντέλα. Η τρίτη επιλογή σημαίνει χρήση του NNAPI (Neural Networks API), ενός API γραμμένου σε γλώσσα C και σχεδιασμένου για την αποδοτική εκτέλεση των υπολογιστικά απαιτητικών λειτουργιών της Μηχανικής Μάθησης σε συσκευές Android. Το εν λόγω API είναι διαθέσιμο σε συσκευές με έκδοση Android 8.1 και πάνω, οπότε θα πρέπει το σύστημα να εκτελεί τον κατάλληλο έλεγχο πριν επιχειρήσει να το χρησιμοποιήσει. Αν και

κανείς θα περίμενε ότι οι επιδόσεις κατά τη χρήση του NNAPI είναι θεαματικές, αυτό δεν συμβαίνει πάντα διότι κάθε κινητή συσκευή Android δεν έχει τον ίδιο DSP ή την ίδια NPU και είναι δύσκολο ένα μοντέλο να καταφέρει να βελτιστοποιηθεί για όλους τους διαφορετικούς επιταχυντές που κυκλοφορούν ;;.

4.1.3 Στάδια αναγνώρισης

Η διαδικασία ή το pipeline της αναγνώρισης μπορεί να διασπαστεί σε τέσσερα στάδια, τα οποία περιγράφονται ακολούθως.

1. Η προετοιμασία της εικόνας από τον αισθητήρα της κάμερας

Τα περισσότερα, αν όχι όλα, συνελικτικά νευρωνικά δίκτυα έχουν περιορισμούς ως προς τις διαστάσεις των εικόνων εισόδου που δέχονται. Οι διαστάσεις αυτές είναι πολύ μικρότερες από τις διαστάσεις στις οποίες αποθηκεύεται συνήθως μια φωτογραφία, οπότε είναι απαραίτητη μια προ-επεξεργασία της εικόνας που μας παρέχει ο αισθητήρας της κάμερας.

Πιο συγκεκριμένα, τα περισσότερα συνελικτικά δίκτυα, όπως και αυτά που χρησιμοποιούνται σε αυτήν την εργασία, δέχονται τετράγωνες εικόνες. Αντίθετα, οι εικόνες που παράγουν οι κινητές συσκευές έχουν σχήμα ορθογώνιου παραλληλόγραμμου. Σε αυτή την εργασία εξετάζονται δύο τρόποι προσαρμογής: η περικοπή (cropping) και η κλιμάκωση (scaling). Το μειονέκτημα της πρώτης μεθόδου είναι ότι ενδεχομένως χάνουμε πληροφορία για το αντικείμενο προς αναγνώριση καθώς απορρίπτουμε τα pixels που βρίσκονται εκτός του κεντρικού τετραγώνου της εικόνας, ενώ της δεύτερης είναι ότι το αντικείμενο παραμορφώνεται και άρα μπορεί να χάσει εντελώς τη μορφή του με αποτέλεσμα η αναγνώριση να είναι ανεπιτυχής. Το Σχήμα 4.1 δείχνει ένα παράδειγμα εφαρμογής των δύο τεχνικών. Στην Ενότητα 6.3 αξιολογούμε και τις δύο τεχνικές σε τρεις διαφορετικές αναλογίες (ratios) φωτογραφιών.

Σχήμα 4.1: *Οι τεχνικές περικοπής και κλιμάκωσης της εικόνας εισόδου*

2. Οι ενέργειες που εκτελούνται πριν τη συμπερασματολογία

Αφορούν τη μετατροπή της τετράγωνης εικόνας του προηγούμενου σταδίου από τη μορφή ARGB_8888 που παρέχει ο αισθητήρας στην μορφή ByteBuffer. Στην αρχική μορφή, το κάθε pixel της εικόνας αποθηκεύεται σε 4 bytes με αποτέλεσμα κάθε ένα από τα τέσσερα κανάλια (RGB και A για τη διαφάνεια) να αποθηκεύεται με ακρίβεια 8 bits. Η τελική μορφή επιτρέπει πολύ γρήγορη και αποδοτική τυχαία πρόσβαση στα δυαδικά δεδομένα κυρίως επειδή εξαλείφει την ανάγκη για αντιγραφή των δεδομένων σε κάποιον άλλον προσωρινό buffer. Γενικά προτιμάται σε περιπτώσεις όπου απαιτείται να γίνουν γρήγορα εγγενείς λειτουργίες εισόδου / εξόδου χαμηλού επιπέδου (low-level I/O).

3. Η εκτέλεση της συμπερασματολογίας

Η εκτέλεση της συμπερασματολογίας από τον Interpreter είναι η σημαντικότερη και συνήθως η πιο χρονοβόρα εργασία. Το συνελικτικό δίκτυο δέχεται ως είσοδο τον ByteBuffer εισόδου από το προηγούμενο στάδιο και επιστρέφει ένα διάνυσμα εξόδου με τις πιθανότητες κάθε κλάσης.

4. Οι ενέργειες που εκτελούνται μετά τη συμπερασματολογία

Το διάνυσμα πιθανοτήτων από το τρίτο στάδιο επεξεργάζεται ώστε να παραχθεί το επιθυμητό αποτέλεσμα. Στην δική μας περίπτωση, στον χρήστη εμφανίζεται μόνο η κλάση με την υψηλότερη πιθανότητα, γι' αυτό ψάχνουμε στο διάνυσμα εξόδου την υψηλότερη πιθανότητα και στην συνέχεια την αντιστοιχίζουμε με το όνομα της κλάσης της. Η επιλογή της εμφάνισης ενός μόνο αποτελέσματος στον χρήστη μειώνει και τον συνολικό χρόνο του τέταρτου σταδίου, καθώς διαφορετικά το διάνυσμα των πιθανοτήτων θα έπρεπε πρώτα να ταξινομηθεί.

4.1.4 Ενσωμάτωση του ταξινομητή στην κάμερα

Για να μην επηρεάζει ο ταξινομητής τις λειτουργίες της κάμερας, η αναγνώριση γίνεται παράλληλα με αυτές. Κάθε φορά που ο ταξινομητής ξεκινά την αναγνώριση μιας νέας εικόνας, παίρνει ως είσοδο το frame που δείχνει εκείνη τη στιγμή η προεπισκόπηση της κάμερας. Επίσης, για να ξεκινήσει η αναγνώριση μιας νέας εικόνας, πρέπει αρχικά να έχει τελειώσει η προηγούμενη μιας και αυτή η διεργασία πραγματοποιείται στο ίδιο νήμα.

Η συχνότητα ή ο ρυθμός με τον οποίο ο αισθητήρας της κάμερας παρέχει νέες εικόνες είτε για καταγραφή βίντεο είτε στην δική μας περίπτωση για την προεπισκόπηση της κάμερας ονομάζεται frame rate και μετριέται σε FPS (frames per second). Μια καλή τιμή για ομαλή κίνηση στο ανθρώπινο μάτι είναι τα 24-30 FPS, με οτιδήποτε μικρότερο να κάνει την κίνηση ασταθή και με εμφανείς «διακοπές». Κάποιες κινητές συσκευές σήμερα χρησιμοποιούν 60 FPS για ακόμα περισσότερο ομαλή κίνηση.

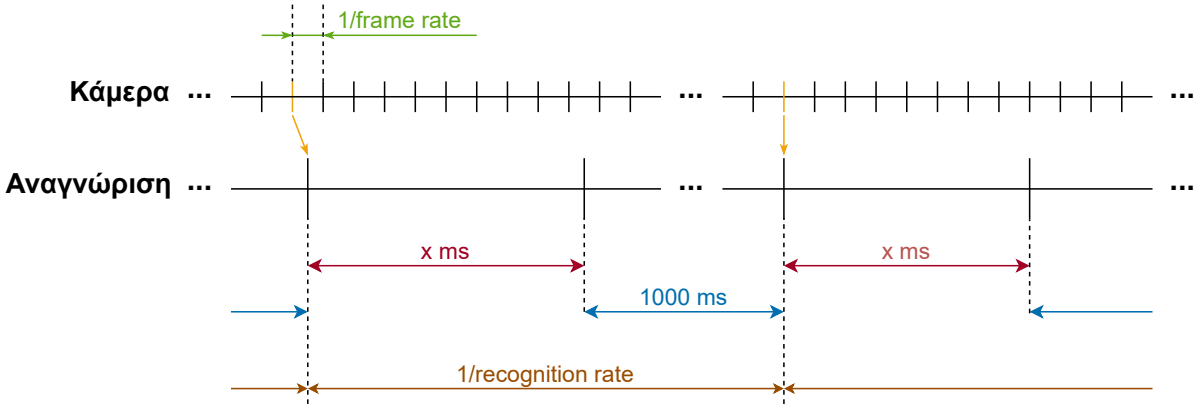
Η συχνότητα με την οποία το σύστημα θα εκτελεί αναγνώριση είναι μια παράμετρος που έπρεπε να ρυθμιστεί. Στο ένα άκρο, η αναγνώριση κάθε frame, δηλαδή 30 ή 60 εικόνων το δευτερόλεπτο είναι άσκοπη διότι είναι πολύ πιθανό όλες αυτές οι εικόνες να απεικονίζουν το ίδιο ακριβώς αντικείμενο και άρα τα περισσότερα αποτελέσματα αναγνώρισης να είναι περιττά. Εκτός από αυτό, η υπερβολικά συχνή χρήση του ταξινομητή οδηγεί σε κατασπατάληση

των υπολογιστικών πόρων με αποτέλεσμα η συσκευή να υπερθερμαίνεται ή ακόμη και να μην αποκρίνεται στον χρήστη. Στο άλλο άκρο, αν η συχνότητα αναγνώρισης γίνει πολύ αραιή, όπως για παράδειγμα 1 frame κάθε 4 δευτερόλεπτα (0.25 FPS), τότε το αποτέλεσμα της αναγνώρισης θα είναι επίσης περιττό καθώς πιθανότατα θα αναφέρεται σε κάποια προηγούμενη σκηνή ή αντικείμενο.

Ο χρόνος αναγνώρισης, δηλαδή ο χρόνος που παρέρχεται από τη στιγμή που το σύστημα δέχεται ως είσοδο μια εικόνα μέχρι τη στιγμή που θα επιστρέψει μια ετικέτα παρουσιάζει μικρές διακυμάνσεις λόγω διαφόρων παραγόντων όπως π.χ. άλλες εφαρμογές που μπορεί να τρέχουν ταυτόχρονα στην συσκευή. Επομένως, ο ρυθμός αναγνώρισης (recognition rate) δεν μπορεί να καθοριστεί επακριβώς, δηλαδή ορίζοντας ανά πόσα καρέ θα εκκινούμε το σύστημα αναγνώρισης.

Παρ' όλα αυτά η συχνότητα μπορεί να προσαρμοστεί μέσω της εισαγωγής μιας χρονικής καθυστέρησης μεταξύ διαδοχικών αναγνωρίσεων. Μετά από μετρήσεις, η καθυστέρηση αυτή ορίστηκε στο 1 δευτερόλεπτο και εξυπηρετεί ένα ευρύ φάσμα μοντέλων, όπως και συσκευών. Το Σχήμα 4.2 δείχνει τη σύνδεση του ταξινομητή με την κάμερα με βάση την παραπάνω ανάλυση. Με πράσινο χρώμα σημειώνεται ο χρόνος προβολής του κάθε frame στην προεπισκόπηση, με πορτοκαλί τα frames τα οποία τελικά θα ταξινομηθούν, με κόκκινο ο χρόνος αναγνώρισης, με γαλάζιο η καθυστέρηση του ενός δευτερολέπτου και με καφέ ο συνολικός χρόνος αναγνώρισης, ο οποίος καθορίζει και τον αντίστοιχο συνολικό ρυθμό:

$$recognition\ rate = \frac{1}{(x + 1000)\ ms}$$



Σχήμα 4.2: *Ενσωμάτωση του ταξινομητή στην κάμερα*

4.2 Λειτουργίες

Η αρχική και κύρια οθόνη της εφαρμογής φαίνεται στο Σχήμα 4.3 της επόμενης σελίδας. Εκτός από την κύρια λειτουργικότητα της κάμερας, η οποία είναι να δείχνει ό,τι βρίσκεται μπροστά από τον φακό και να αποθηκεύει φωτογραφίες, προστέθηκαν και άλλες λειτουργίες ώστε να ενισχυθεί η ευελιξία και η ευρωστία του συστήματος. Αυτές είναι (α) η χρήση flash, (β) η δυνατότητα αλλαγής της αναλογίας (ratio) της προεπισκόπησης και (γ) το zoom.

Σχήμα 4.3: *Κύρια οθόνη*

Flash

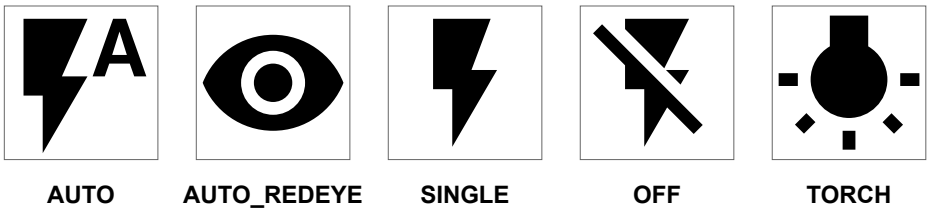
Κατά την αρχική ρύθμιση της κάμερας, πρέπει να ελεγχθεί αν η συσκευή διαθέτει flash και στην περίπτωση που διαθέτει, το αντίστοιχο κουμπί να γίνει ορατό. Εφόσον υπάρχει flash, είναι σίγουρο ότι υποστηρίζονται οι τρεις παρακάτω κύριες λειτουργίες του.

- SINGLE: Το flash ενεργοποιείται μόνο κατά τη λήψη φωτογραφίας.
- OFF: Το flash είναι συνεχώς απενεργοποιημένο.
- TORCH: Το flash είναι συνεχώς ενεργοποιημένο (σαν φακός).

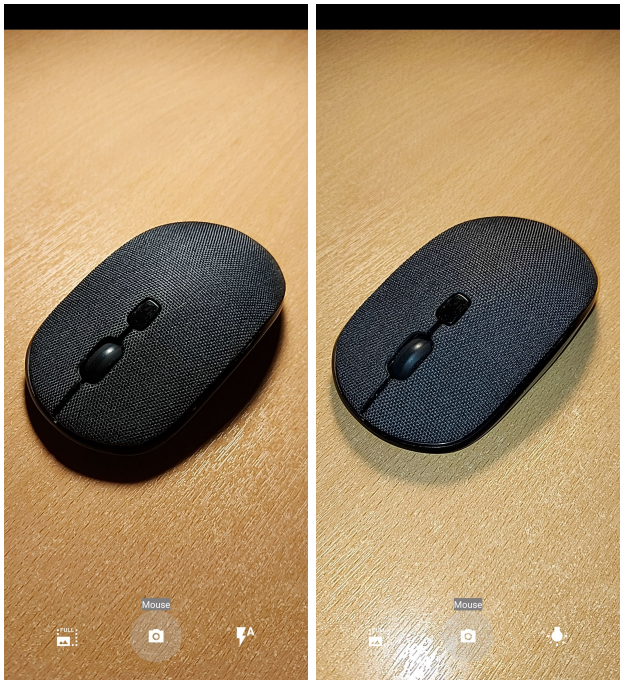
Εκτός από αυτές, κάποιες συσκευές υποστηρίζουν και άλλες δύο λειτουργίες του flash που σχετίζονται με την αυτόματη έκθεση (auto-exposure), οπότε πριν αυτές χρησιμοποιηθούν θα πρέπει να γίνει ο κατάλληλος έλεγχος.

- AUTO: Το flash ενεργοποιείται μόνο κατά τη λήψη φωτογραφίας σε συνθήκες χαμηλού φωτισμού.
- AUTO_REDEYE: Σαν τη λειτουργία AUTO, αλλά με αυτόματη μείωση των κόκκινων ματιών.

Τελικά, ο μέγιστος αριθμός λειτουργιών είναι πέντε με τις τρεις από αυτές να υποστηρίζονται από κάθε συσκευή. Στα Σχήματα 4.4 και 4.5 της επόμενης σελίδας φαίνονται τα εικονίδια που χρησιμοποιούνται για κάθε λειτουργία και ένα παράδειγμα χρήσης της λειτουργίας TORCH αντίστοιχα.



Σχήμα 4.4: *Εικονίδια για τις λειτουργίες του flash*

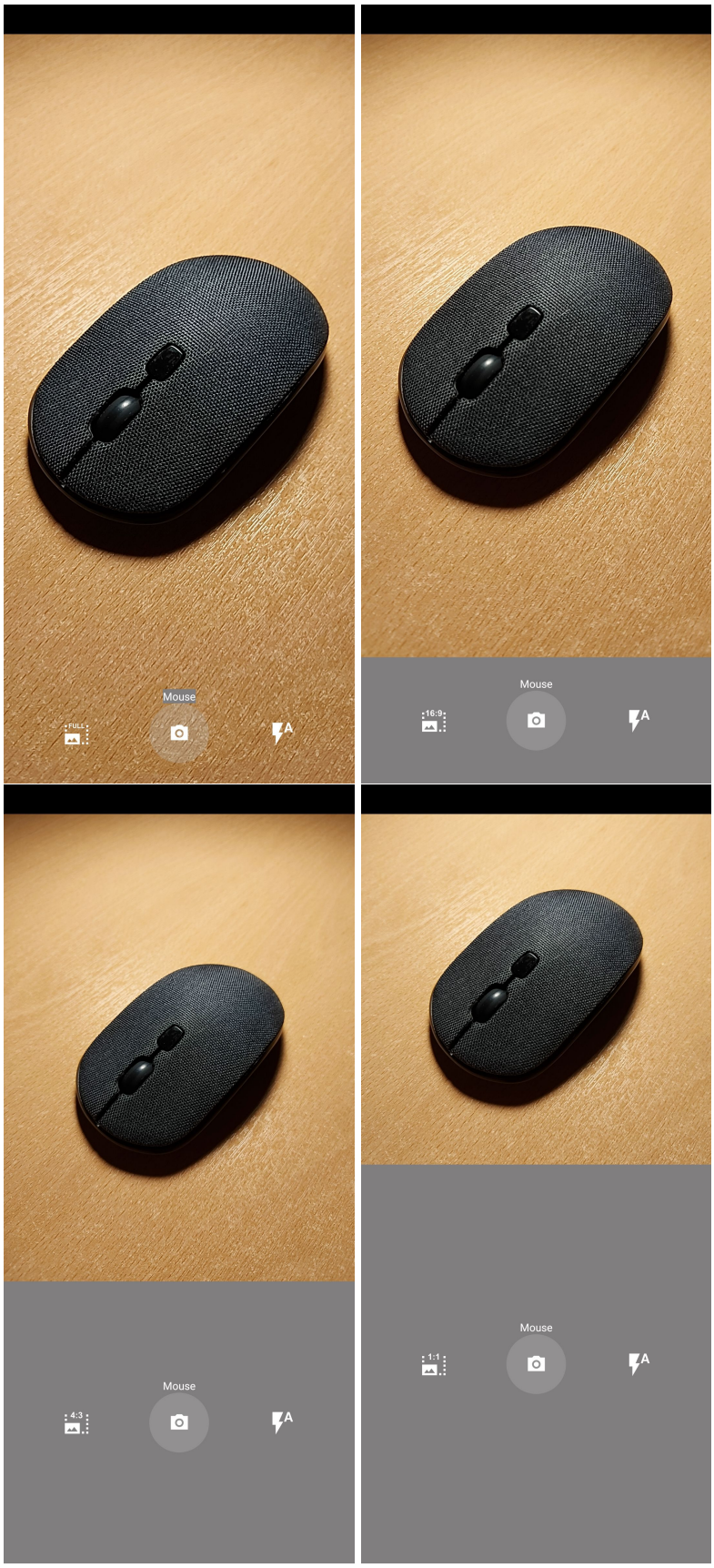


Σχήμα 4.5: *Παράδειγμα χρήσης της λειτουργίας TORCH*

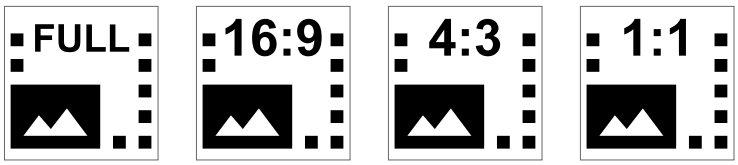
Ratios

Κάθε αισθητήρας έχει προκαθορισμένα μεγέθη όσον αφορά στην προεπισκόπηση της κάμερας και στην αποθήκευση φωτογραφιών. Αυτές οι λίστες μεγεθών μπορεί να είναι οι ίδιες ή να διαφέρουν από συσκευή σε συσκευή. Από αυτά τα μεγέθη, μπορούν εύκολα να υπολογιστούν οι υποστηριζόμενες αναλογίες (ratios). Στην εφαρμογή, αποφασίστηκε να υπάρχουν μέχρι τέσσερις διαφορετικές αναλογίες, εφόσον αυτές υποστηρίζονται από τη συσκευή. Αυτές είναι οι 16:9, 4:3, 1:1 και η αναλογία της οθόνης, η οποία ονομάστηκε FULL, εφόσον δεν συμπίπτει με κάποια από τις προηγούμενες. Κατά την αρχική ρύθμιση, η προεπισκόπηση της κάμερας επιλέγεται να έχει την αναλογία FULL και στην συνέχεια ελέγχεται αν ο αισθητήρας υποστηρίζει και τις άλλες τρεις αναλογίες. Αν η αναλογία FULL ταυτίζεται με κάποια από τις άλλες τρεις, τότε ο έλεγχος προφανώς παραλείπεται.

Για παράδειγμα, αν μια συσκευή έχει αναλογία οθόνης 16:9 και υποστηρίζει επιπλέον μόνο την αναλογία 4:3, τότε θα είναι διαθέσιμες μόνο αυτές οι δύο αναλογίες. Στα Σχήματα 4.6 και 4.7 των επόμενων σελίδων φαίνονται οι οθόνες για διαφορετικές αναλογίες προεπισκόπησης σε μια συσκευή με αναλογία οθόνης 20:9 και τα εικονίδια που χρησιμοποιούνται για κάθε αναλογία αντίστοιχα.



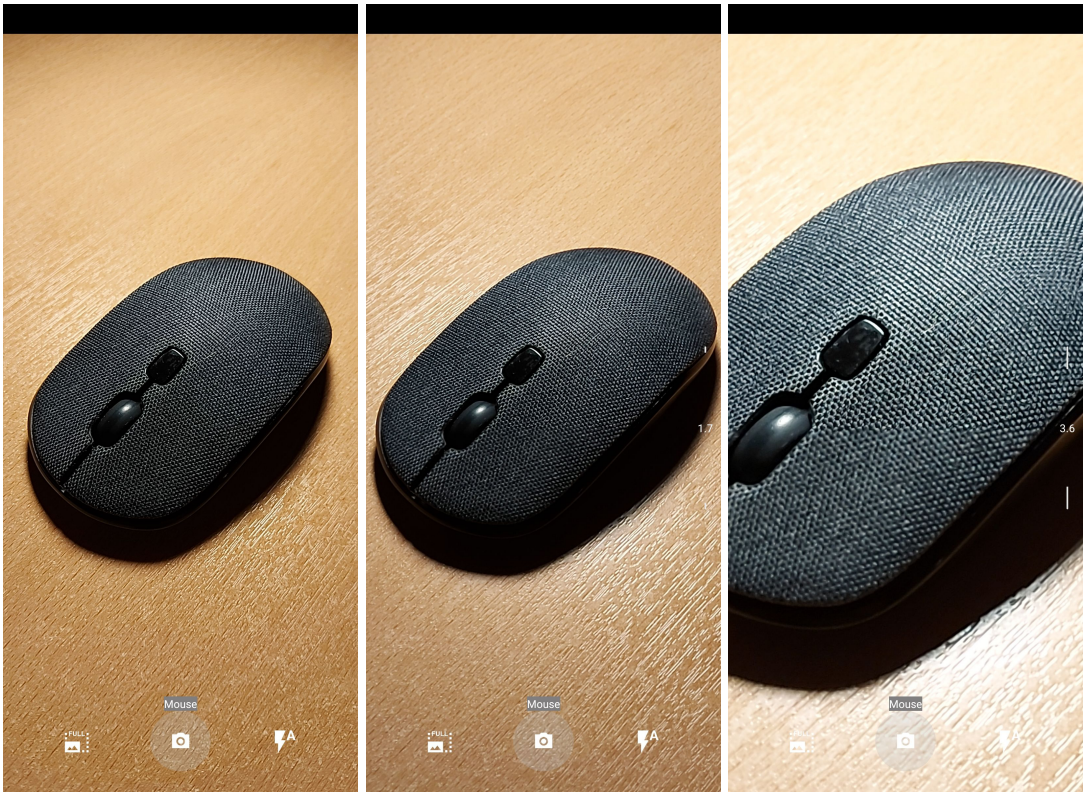
Σχήμα 4.6: *Οθόνες της εφαρμογής με διαφορετικές αναλογίες*



Σχήμα 4.7: *Εικονίδια για τις διαφορετικές αναλογίες*

Zoom

Το zoom πραγματοποιείται με τη μέθοδο pinch-to-zoom, για την οποία χρειάζονται δύο δάχτυλα στην οθόνη της συσκευής. Για τη μεγέθυνση της προεπισκόπησης (zoom in) τα δύο δάχτυλα πρέπει να μετακινηθούν προς αντίθετη κατεύθυνση, ενώ για τη σμίκρυνση (zoom out) τα δάχτυλα πρέπει να μετακινηθούν προς την ίδια κατεύθυνση. Όταν κάποια ενέργεια από τις παραπάνω συμβαίνει, εμφανίζεται στην οθόνη του χρήστη μια μπάρα προόδου (progress bar) που τον ενημερώνει για το τρέχον ποσοστό μεγέθυνσης. Στην αρχική κατάσταση η προεπισκόπηση βρίσκεται σε ποσοστό 1x, ενώ μπορεί να φτάσει μέχρι και το μέγιστο ποσοστό μεγέθυνσης που υποστηρίζει ο κάθε αισθητήρας, για παράδειγμα 8x. Στο Σχήμα 4.8 φαίνεται ένα παράδειγμα μεγέθυνσης.



Σχήμα 4.8: *Παράδειγμα μεγέθυνσης*

Κεφάλαιο 5

Ανάπτυξη

Στο πέμπτο Κεφάλαιο αναφέρονται βασικά στοιχεία γύρω από την ανάπτυξη του συστήματος, δηλαδή μεταφράζεται η ανάλυση που έγινε στο προηγούμενο Κεφάλαιο σε Java κλάσεις και παρουσιάζονται οι λεπτομέρειες για τα μοντέλα Βαθιάς Μάθησης και τις κινητές συσκευές που χρησιμοποιήθηκαν.

5.1 Κλάσεις

Η εφαρμογή απαρτίζεται από τρεις Java κλάσεις, οι οποίες αντιστοιχούν στα τρία επίπεδα της αρχιτεκτονικής του συστήματος. Η περιγραφή που παρατίθεται παρακάτω είναι συνοπτική με τον κώδικα καθαυτό να παραλείπεται διότι δεν αποτελεί το βασικό μέρος εστίασης. Κάθε ενδιαφερόμενος μπορεί να αναζητήσει ολόκληρο το project της εφαρμογής με τον κώδικα, τους πόρους, τις ρυθμίσεις και ένα demo APK στο Github [37].

5.1.1 CameraActivity

Η CameraActivity αντιστοιχεί στο L1, δηλαδή στο Ανεξάρτητο Υπηρεσίας Επίπεδο που ασχολείται κυρίως με την ανάπτυξη και τις λειτουργίες της κάμερας και αποτελεί την κύρια δραστηριότητα (main activity) της εφαρμογής.

Οι δραστηριότητες (activities) είναι ένας από τους τέσσερις βασικούς τύπους δομικών στοιχείων (components) μιας Android εφαρμογής. Κάθε τύπος εξυπηρετεί έναν ξεχωριστό σκοπό και έχει έναν ξεχωριστό κύκλο ζωής που καθορίζει τον τρόπο δημιουργίας και καταστροφής του κάθε στοιχείου [38]. Για τη μετάβαση μεταξύ των σταδίων του κύκλου ζωής μιας δραστηριότητας υπάρχει ένα σύνολο από έξι βασικές κλήσεις (callbacks): onCreate(), onStart(), onResume(), onPause(), onStop() και onDestroy(). Το σύστημα καλεί κάθε μια από αυτές τις μεθόδους καθώς η δραστηριότητα εισέρχεται σε μια νέα κατάσταση [39]. Για παράδειγμα, όταν μια δραστηριότητα δημιουργείται, καλείται η onCreate(), η οποία είναι και η πρώτη μέθοδος που υλοποιείται συνήθως από τους προγραμματιστές.

Μια δραστηριότητα είναι το σημείο εισόδου για την αλληλεπίδραση του χρήστη με την εφαρμογή και αντιπροσωπεύει μια οθόνη της [38]. Η κύρια δραστηριότητα είναι η πρώτη οθόνη που εμφανίζεται στον χρήστη όταν αυτός ανοίγει την εφαρμογή. Στην δική μας περίπτωση, μια οθόνη ήταν επαρκής, οπότε η κύρια δραστηριότητα είναι και η μοναδική.

5.1.2 MDSpecs

Η συγκεκριμένη κλάση αντιστοιχεί στο L3, δηλαδή στο Επίπεδο Προσαρμογής Κινητής Συσκευής. Όπως αναφέρει και το όνομά της, παρέχει όλες τις πληροφορίες σχετικά με την κινητή συσκευή.

Η συγκεκριμένη κλάση εξυπηρετεί και ως ViewModel της κύριας δραστηριότητας CameraActivity. ViewModel είναι μια κλάση σχεδιασμένη για να αποθηκεύει και να διαχειρίζεται δεδομένα που σχετίζονται με το UI με συνειδητό τρόπο ως προς τον κύκλο ζωής μιας δραστηριότητας. Αυτό σημαίνει ότι δεδομένα που απαιτούνται στην κύρια δραστηριότητα μπορούν να επιβιώσουν σε αλλαγές της διαμόρφωσης του UI, όπου συνήθως απαιτείται η επανεκκίνηση της δραστηριότητας και η διαγραφή όλων των δεδομένων της [40]. Στην δική μας περίπτωση, αλλαγή στο UI έχουμε κάθε φορά που αλλάζει η αναλογία της προεπισκόπησης της κάμερας, όπου η κύρια δραστηριότητα διαγράφεται και δημιουργείται από την αρχή. Στην κλάση MDSpecs υπάρχει μια μεταβλητή που παρακολουθεί την τρέχουσα αναλογία και παρέχει τη νέα κατά την αναδημιουργία της δραστηριότητας, καθώς και μια μεταβλητή που παρακολουθεί την τρέχουσα λειτουργία του flash.

5.1.3 Classifier

Η τρίτη κλάση αντιστοιχεί στο L2, δηλαδή στο Επίπεδο Προσαρμογής Αρχιτεκτονικής Βαθιάς Μάθησης. Κατά τη δημιουργία ενός αντικειμένου τύπου Classifier, δημιουργείται και ο Interpeter για το μοντέλο Βαθιάς Μάθησης που έχει επιλεγεί. Στον κατασκευαστή (constructor) της κλάσης προσδιορίζονται το μοντέλο και οι παράμετροι αυτού, καθώς και οι παράμετροι που σχετίζονται με το υλικό που θα χρησιμοποιηθεί για την εκτέλεση της συμπερασματολογίας.

Η βασική μέθοδος αυτής της κλάσης είναι η Classifier#classify(Bitmap bitmap), η οποία δέχεται ως όρισμα την εικόνα που προκύπτει από το στάδιο 1 της διαδικασίας αναγνώρισης σε μορφή Bitmap και εκτελεί τα υπόλοιπα 3 στάδια της αναγνώρισης. Το δεύτερο στάδιο εκτελεί η μέθοδος Classifier#convertBitmapToByteBuffer(Bitmap bitmap), το τρίτο η μέθοδος Interpreter#run(Object input, Object output) και το τέταρτο είτε η Classifier#getTopResultByte(byte[][] labelProbArray) για κβαντισμένα μοντέλα, είτε η Classifier#getTopResultFloat(float[][] labelProbArray) για floating point μοντέλα, οι οποίες επιστρέφουν την πιο πιθανή κλάση.

Στην περίπτωση όπου το αντικείμενο προς αναγνώριση δεν υπάρχει στις κλάσεις με τις οποίες έχει εκπαιδευτεί το μοντέλο, αναμένεται ότι η μεγαλύτερη πιθανότητα θα είναι πολύ μικρή, οπότε προτού επιλεγεί η πιο πιθανή κλάση, ελέγχεται αν η πιθανότητα της ξεπερνά ένα κατώφλι. Αυτό το κατώφλι είναι διαφορετικό για κάθε μοντέλο και στα μοντέλα που δοκιμάστηκαν σε αυτή την εργασία κυμαίνεται μεταξύ 10-30 %.

5.2 Μοντέλα

Για τη διερεύνηση της ευρωστίας του συστήματος επιλέχτηκαν τέσσερις γνωστές και διαφορετικές αρχιτεκτονικές συνελικτικών δικτύων: τα κλασικά MobileNetV2 [41], το ResNetV2 [42], το InceptionV3 [42] και τα πιο πρόσφατα EfficientNets στην lite εκδοχή τους [43].

Συνολικά επιλέχτηκαν δέκα μοντέλα τα οποία φαίνονται στον Πίνακα 5.1 ταξινομημένα σε αύξουσα σειρά Top-1 ορθότητας (accuracy), η οποία υποδηλώνει το ποσοστό των δειγμάτων για τα οποία η πιο πιθανή κλάση είναι και η σωστή. Αντίστοιχα, η Top-5 ορθότητα υποδηλώνει το ποσοστό των δειγμάτων για τα οποία η σωστή κλάση βρίσκεται στις 5 πιο πιθανές κλάσεις. Παρουσιάζεται επίσης το πλήθος των παραμέτρων κάθε δικτύου, από το οποίο προκύπτει και το μέγεθος αυτού, ανάλογα με το αν το δίκτυο είναι κβαντισμένο ή όχι.

Α/Α	**Μοντέλο**	**Μέγεθος εισόδου**	**Top-1 Ορθότητα**	**Top-5 Ορθότητα**	**Αριθμός παραμέτρων**	**Μέγεθος**
1	MobileNetV2 1.0 quantized	224x224	70.8%	89.9%	3.47 M	3.309 MB
2	MobileNetV2 1.0	224x224	71.8%	91.0%	3.47 M	13.237 MB
3	EfficientNetLite0 quantized	224x224	74.4%	N/A	4.7 M	4.482 MB
4	MobileNetV2 1.4	224x224	75.0%	92.5%	6.06 M	23.117 MB
5	EfficientNetLite0	224x224	75.1%	92.3%	4.7 M	17.929 MB
6	ResNetV2 101	299x299	76.8%	93.6%	44.5 M	169.754 MB
7	InceptionV3 quantized	299x299	77.5%	93.7%	23.9 M	22.793 MB
8	InceptionV3	299x299	77.9%	93.8%	23.9 M	91.171 MB
9	EfficientNetLite4 quantized	300x300	80.2%	N/A	13.0 M	12.398 MB
10	EfficientNetLite4	300x300	81.5%	95.6%	13.0 M	49.591 MB

Πίνακας 5.1: *Τα μοντέλα που χρησιμοποιήθηκαν για την αξιολόγηση του συστήματος*

Όλα τα μοντέλα είναι προεκπαιδευμένα στο σύνολο δεδομένων ImageNet, το οποίο αποτελείται από περισσότερες από 14 εκατομμύρια εικόνες και περιέχει περισσότερες από 20 χιλιάδες κατηγορίες. Ωστόσο, για την εκπαίδευση συνελικτικών δικτύων χρησιμοποιείται ένα υποσύνολο των 1000 κατηγοριών.

Τα πέντε πρώτα μοντέλα είναι πιο «ελαφριά» σε σχέση με τα πέντε τελευταία καθώς έχουν μικρότερο αριθμό παραμέτρων και δέχονται εικόνες εισόδου με μικρότερο μέγεθος. Σε τέσσερα μοντέλα έχει συμπεριληφθεί τόσο η floating point, όσο και η κβαντισμένη εκδοχή τους. Βλέπουμε ότι το μέγεθος των κβαντισμένων μοντέλων είναι περίπου 4 φορές μικρότερο, ενώ η ορθότητα τους έχει μειωθεί κατά ένα μικρό ποσοστό.

5.3 Συσκευές

Τα παραπάνω μοντέλα αξιολογήθηκαν σε δύο κινητές συσκευές Android των οποίων τα βασικότερα χαρακτηριστικά φαίνονται στον Πίνακα 5.2. Οι δύο συσκευές είναι αρκετά διαφορετικές μεταξύ τους καθώς η μια είναι πιο παλιά, ενώ η άλλη φετινή. Αυτό μας βοηθάει πολύ στο να ερευνήσουμε την ευελιξία του συστήματος μας ως προς διαφορετικές συσκευές και αναμένουμε ενδιαφέροντα αποτελέσματα και συγκρίσεις μεταξύ τους. Για λόγους απλότητας και διευκόλυνσης, η Sony συσκευή θα αναφέρεται ως συσκευή Α και η Samsung ως συσκευή Β.

Μοντέλο	**Sony Xperia C5 Ultra (A)**	**Samsung Galaxy A71 (B)**
Κυκλοφορία	2015, Αύγουστος	2020, Ιανουάριος
System on Chip (SoC)	MediaTek MT6752	Qualcomm Snapdragon 730
CPU	8x 1.69 GHz ARM Cortex-A53	2x 2.2 GHz Kryo 470 Gold 6x 1.8 GHz Kryo 470 Silver
GPU	2x 700 MHz ARM Mali-T760	Qualcomm Adreno 618
Συνολική RAM	2 GB, 800 MHz	6 GB, 1866 MHz
Έκδοση Android	6.0 (API level 23)	10 (API level 29)
Back Facing Camera Hardware Level	LEGACY	LEVEL_3

Πίνακας 5.2: *Χαρακτηριστικά συσκευών*

Παρατηρώντας τα παραπάνω χαρακτηριστικά, αναμένουμε ότι η συσκευή Β θα έχει υψηλότερες αποδόσεις από τη συσκευή Α διότι διαθέτει πιο σύγχρονες CPU και GPU και περισσότερη μνήμη RAM. Επίσης, η συσκευή Β υποστηρίζει το NNAPI μιας και λειτουργεί με την έκδοση Android 10, η οποία είναι νεότερη από την 8.1, καθώς και επιπρόσθετες λειτουργίες κάμερας σε αντίθεση με τη συσκευή Α, η οποία υποστηρίζει μόνο βασικές λειτουργίες από το παλιό Camera API.

Κεφάλαιο 6

Αξιολόγηση

Το κεφάλαιο της αξιολόγησης μαζί με τα γενικά συμπεράσματα που αναλύονται στο Κεφάλαιο 7 είναι δύο από τα πιο σημαντικά σημεία της διπλωματικής εργασίας. Παρακάτω παρουσιάζονται οι μετρικές που χρησιμοποιήθηκαν για την αξιολόγηση της απόδοσης, οι μετρήσεις χρόνου και μνήμης, καθώς και η εξέταση των μεθόδων περικοπής και κλιμάκωσης για την προετοιμασία της εικόνας προς αναγνώριση.

6.1 Μετρικές

Η απόδοση του κάθε μοντέλου βασίστηκε σε μετρήσεις που έγιναν σχετικά με (α) τον χρόνο εκτέλεσης (latency) τόσο της συμπερασματολογίας όσο και των υπόλοιπων τριών σταδίων του pipeline αναγνώρισης και (β) τη χρήση μνήμης της εφαρμογής όσο το σύστημα λειτουργεί. Οι μετρικές που χρησιμοποιήθηκαν για το latency είναι πέντε:

1. Minimum: Η ελάχιστη τιμή όλων των μετρήσεων.
2. Maximum: Η μέγιστη τιμή όλων των μετρήσεων.
3. Mean: Η μέση τιμή όλων των μετρήσεων.
4. Median: Η τιμή που χωρίζει το ταξινομημένο σύνολο των μετρήσεων στην μέση.
5. 90^{th} percentile: Η τιμή κάτω από την οποία βρίσκεται το 90% των μετρήσεων.

6.2 Μετρήσεις

Οι μετρήσεις για τον χρόνο πραγματοποιήθηκαν απομονώνοντας τα επιθυμητά κομμάτια κώδικα με τη μέθοδο System#currentTimeMillis() της Java, η οποία επιστρέφει την τρέχουσα ώρα σε milliseconds. Ο χρόνος που παρήλθε προκύπτει αφαιρώντας τον αρχικό χρόνο από τον τελικό. Πριν ξεκινήσουν οι μετρήσεις, έπρεπε να καθοριστούν:

- Ο αριθμός των warm-up runs, δηλαδή το πλήθος των αναγνωρίσεων μέχρι η συσκευή να προσαρμοστεί στις λειτουργίες της εφαρμογής. Παρατηρήθηκε ότι μετά τα 10-15 πρώτα runs οι χρόνοι ήταν πιο σταθεροί, οπότε επιλέχτηκε η τιμή 15.

- Ο αριθμός των συνολικών (total) runs (μετά το πέρας των warm-up), ώστε να προκύψουν οι μετρικές που αναφέρθηκαν παραπάνω. Αυτή η παράμετρος επιλέχτηκε 200 διότι φάνηκε ότι δεν υπάρχει σημαντική διαφορά στα αποτελέσματα με περισσότερα runs.

6.2.1 Pipeline αναγνώρισης

Όπως αναφέρθηκε και στην Υποενότητα 4.1.4, ο χρόνος αναγνώρισης παρουσιάζει διακυμάνσεις, οπότε για να έχουμε μια καλύτερη εικόνα, καλό θα ήταν να είναι γνωστή η κατανομή αυτού του χρόνου στα 4 στάδια που απαρτίζουν το pipeline της αναγνώρισης. Στον Πίνακα 6.1 έχουν συγκεντρωθεί οι χρόνοι σε ms των 5 μετρικών για τα στάδια 1 (προετοιμασία εικόνας από τον αισθητήρα), 2 (ενέργειες πριν τη συμπερασματολογία) και 4 (ενέργειες μετά τη συμπερασματολογία). Το στάδιο 3 αναλύεται ξεχωριστά παρακάτω. Μικρό μέγεθος εικόνας σημαίνει 224x224, ενώ μεγάλο σημαίνει 299x299 ή 300x300.

Πρέπει να αναφερθεί ότι οι μετρήσεις έγιναν θεωρώντας τη χειρότερη δυνατή περίπτωση (worst-case scenario). Αυτό σημαίνει ότι στο στάδιο 1 εφαρμόστηκε η μέθοδος περικοπής και όχι η μέθοδο κλιμάκωσης, καθώς η πρώτη είναι πιο χρονοβόρα, ενώ στο στάδιο 4 ταξινομούμε τον πίνακα των πιθανοτήτων, αντί να κάνουμε αναζήτηση για τη μέγιστη πιθανότητα.

Στάδιο	**Μέγεθος εικόνας**	**Συσκευή**	**Επεξεργαστής**	**Min**	**Max**	**Mean**	**Median**	**90th percentile**
1	Μικρό	A	CPU	9	33	15.8	14-16	22-24
		B	CPU	5	47	17.7	14-18	22-30
			GPU	6	46	19.2	16-20	26-31
			NNAPI	7	78	16.8	13-18	20-27
	Μεγάλο	A	CPU	15	44	24.3	22-26	29-34
		B	CPU	5	54	23.6	19-26	29-38
			GPU	6	45	21.9	21-22	30-32
			NNAPI	7	76	23.6	19-25	31-36
2	Μικρό	A	CPU	26	78	39.9	33-41	49-56
		B	CPU	3	42	15.1	11-18	15-26
			GPU	6	33	18.1	17-18	24-25
			NNAPI	4	36	15.2	10-18	17-27
	Μεγάλο	A	CPU	46	95	62.9	51-72	64-81
		B	CPU	4	48	21.5	17-25	22-33
			GPU	8	66	23.6	22-23	30-32
			NNAPI	5	41	20.8	18-24	24-31
4	Οποιοδήποτε	A	CPU	2	9	3.1	3	3-4
		B	CPU	0	8	0,8	0-1	1-2
			GPU	0	24	3.0	1-4	1-13
			NNAPI	0	13	2.0	1-3	1-5

Πίνακας 6.1: *Latency για τα στάδια 1, 2 και 4*

Ο χρόνος προετοιμασίας (στάδιο 1) εξαρτάται από το μέγεθος της εικόνας εισόδου διότι όσο πιο μεγάλο είναι το frame που παρέχει η προεπισκόπηση της κάμερας, τόσο μεγαλύτερος είναι και ο χρόνος ανάκτησης. Δεν παρατηρείται κάποια σημαντική διαφορά στον χρόνο μεταξύ των δύο συσκευών, ούτε και μεταξύ των διάφορων επεξεργαστών.

Ο χρόνος για την εκτέλεση των λειτουργιών πριν τη συμπερασματολογία (στάδιο 2) φαίνεται να εξαρτάται σημαντικά τόσο από το μέγεθος της εικόνας όσο και από τη συσκευή.

Όσο μεγαλύτερη είναι η εικόνα, τόσο περισσότερα pixels χρειάζεται να διασπαστούν στα τρία κανάλια RGB και να αποθηκευτούν στον ByteBuffer. Στην συσκευή Α παρατηρείται ο τριπλάσιος χρόνος από αυτόν στην συσκευή Β κι αυτό ίσως οφείλεται στο γεγονός ότι η μετατροπή που λαμβάνει χώρα σε αυτό το στάδιο είναι low-level και άρα η CPU έχει σημαντικό ρόλο. Σχετικά με τους επεξεργαστές, παρατηρείται ότι με τη χρήση της GPU ο χρόνος αυξάνεται λίγο.

Ο χρόνος για την εκτέλεση των λειτουργιών μετά τη συμπερασματολογία (στάδιο 3) δεν εξαρτάται από το μέγεθος της εικόνας εισόδου, μιας και η κύρια λειτουργία του είναι η διαχείριση των πιθανοτήτων εξόδου, το πλήθος των οποίων είναι σταθερό για κάθε μέγεθος εισόδου. Φαίνεται όμως ότι αυτός ο χρόνος τριπλασιάζεται στην CPU της συσκευής Α σε σύγκριση με τη CPU της συσκευής Β. Επίσης, παρατηρείται αύξηση του χρόνου με τη χρήση του NNAPI και ακόμα μεγαλύτερη με χρήση της GPU.

6.2.2 Συμπερασματολογία

Το latency του τρίτου (**3**) σταδίου καθώς και το συνολικό (**T**otal) latency και των τεσσάρων σταδίων αποτελούν τις πιο σημαντικές μετρήσεις. Αναλυτικοί πίνακες με τις εν λόγω μετρήσεις για κάθε μοντέλο, συσκευή και επεξεργαστή παρατίθενται στο Παράρτημα Α'. Στον Πίνακα 6.2 παρακάτω έχουν συγκεντρωθεί οι καλύτεροι χρόνοι για κάθε μοντέλο με βάση τη μετρική 90^{th} percentile. Οι μετρικές Mean και Median εξάγουν παρόμοια συμπεράσματα. Τις δύο μετρήσεις συνοδεύουν ο επεξεργαστής στον οποίο παρατηρήθηκαν οι συγκεκριμένοι χρόνοι, καθώς και το ποσοστό που καταλαμβάνει η συμπερασματολογία σε ολόκληρο το pipeline αναγνώρισης.

Α/Α	**Μοντέλο**	**Συσκευή Α**				**Συσκευή Β**			
		Επεξεργαστής	**3**	**T**	**%**	**Επεξεργαστής**	**3**	**T**	**%**
1	MobileNetV2 1.0 quantized	CPU (2)	94	157	59.9	NNAPI	11	54	20.4
2	MobileNetV2 1.0	CPU (3)	120	194	61.9	GPU	22	71	31.0
3	EfficientNetLite0 quantized	CPU (2)	117	183	63.9	CPU (2)	41	78	52.6
4	MobileNetV2 1.4	CPU (4)	189	249	75.9	GPU	26	77	33.8
5	EfficientNetLite0	CPU (3)	155	223	69.5	GPU	21	75	28.0
6	ResNetV2 101	-	-	-	-	CPU (4)	709	766	92.6
7	InceptionV3 quantized	CPU (3)	534	619	86.3	NNAPI	116	173	67.1
8	InceptionV3	-	-	-	-	NNAPI	222	279	79.6
9	EfficientNetLite4 quantized	CPU (3)	346	431	80.3	CPU (2)	98	145	67.6
10	EfficientNetLite4	CPU (2)	785	890	88.2	NNAPI	166	226	73.5

Πίνακας 6.2: *Latency για το στάδιο 3 και ολικό latency*

Όπως παρατηρείται, στην πιο παλιά συσκευή (Α) για όλες τις καλύτερες μετρήσεις χρησιμοποιείται η CPU επειδή από τη μια το NNAPI δεν είναι διαθέσιμο και από την άλλη η χρήση της GPU στα floating point μοντέλα, αν και είχε καλύτερη απόδοση, οδηγούσε σε «διακοπές» στην προεπισκόπηση της κάμερας με αποτέλεσμα μη ομαλή κίνηση. Στην πιο νέα συσκευή Β, η χρήση της GPU στα floating point μοντέλα έδινε πάντα τον καλύτερο χρόνο, όμως στα

πιο «βαριά» μοντέλα, δηλαδή στα μοντέλα 6, 8 και 10 οδηγούσε επίσης σε «διακοπές» στην προεπισκόπηση.

Οι μετρήσεις που λείπουν για τα μοντέλα 6 και 8 στην συσκευή Α υποδηλώνουν ότι τα μοντέλα αυτά είναι ακατάλληλα για τη συγκεκριμένη συσκευή, καθώς υπήρχαν «διακοπές» ακόμη και με τη χρήση της CPU. Αξίζει, τέλος, να αναφερθεί ότι κατά την εκτέλεση της εφαρμογής στην συσκευή Α με οποιοδήποτε από τα μοντέλα, η συσκευή υπερθερμαινόταν, με τη θερμότητα να είναι αρκετά αισθητή στην παλάμη με χρήση των πιο «βαριών» μοντέλων.

6.2.3 Χρήση μνήμης

Ο Πίνακας 6.3 είναι συνέχεια του Πίνακα 6.2, οπότε αναφέρεται επίσης στους καλύτερους χρόνους. Δείχνει τη χρησιμοποίηση της μνήμης RAM (memory footprint) κατά τη διάρκεια που η εφαρμογή αυτή εκτελείται στην αντίστοιχη συσκευή. Η εν λόγω μέτρηση έγινε χρησιμοποιώντας τον Profiler του Android Studio και παρέμενε σταθερή καθ' όλη τη διάρκεια εκτέλεσης.

Α/Α	**Μοντέλο**	**Συσκευή Α**		**Συσκευή Β**	
		Επεξεργαστής	**Μνήμη**	**Επεξεργαστής**	**Μνήμη**
1	MobileNetV2 1.0 quantized	CPU (2)	80 MB	NNAPI	120 MB
2	MobileNetV2 1.0	CPU (3)	98 MB	GPU	210 MB
3	EfficientNetLite0 quantized	CPU (2)	84 MB	CPU (2)	125 MB
4	MobileNetV2 1.4	CPU (4)	110 MB	GPU	240 MB
5	EfficientNetLite0	CPU (3)	101 MB	GPU	210 MB
6	ResNetV2 101	-	-	CPU (4)	330 MB
7	InceptionV3 quantized	CPU (3)	115 MB	NNAPI	140 MB
8	InceptionV3	-	-	NNAPI	208 MB
9	EfficientNetLite4 quantized	CPU (3)	100 MB	CPU (2)	135 MB
10	EfficientNetLite4	CPU (2)	150 MB	NNAPI	173 MB

Πίνακας 6.3: *Χρήση μνήμης*

Στις περιπτώσεις των μοντέλων 3 και 9, όπου και οι δύο συσκευές τα πηγαίνουν καλύτερα με τη χρήση της CPU, παρατηρούμε ότι στη συσκευή Β χρησιμοποιείται λίγο περισσότερη μνήμη, γεγονός το οποίο ίσως οφείλεται στην μεγαλύτερη οθόνη που διαθέτει η συσκευή Β. Στις περιπτώσεις όπου στη συσκευή Β χρησιμοποιείται το NNAPI, παρατηρούμε παρόμοια συμπεριφορά με ίσως μικρότερες αυξήσεις. Τέλος, στις περιπτώσεις χρήσης GPU, η μνήμη αυξάνεται πολύ, κάτι που δεν είναι επιθυμητό.

Αποτελέσματα

Για τα πρώτα πέντε «ελαφριά» μοντέλα με μέγεθος εισόδου 224x224 θα μπορούσαμε να πούμε ότι για έναν καλό συμβιβασμό μεταξύ ορθότητας και latency:

- Στην συσκευή Α, καλύτερο μοντέλο φαίνεται να είναι το κβαντισμένο EfficientNetLite0

με χρήση της CPU και 2 νημάτων. Εκτός από το γεγονός ότι το συγκεκριμένο μοντέλο παρουσιάζει πολύ καλό latency, χρησιμοποιεί και λίγα νήματα σε σύγκριση με τα υπόλοιπα μοντέλα.

- Στην συσκευή Β, προτιμάται το EfficientNetLite0 στην GPU. Γενικά όπου μπορούμε να χρησιμοποιήσουμε τη GPU χωρίς η προεπισκόπηση να «κολλάει», τα floating point μοντέλα προτιμώνται.

Για τα υπόλοιπα πέντε μοντέλα με μέγεθος εισόδου 299x299 ή 300x300, τόσο για τη συσκευή Α όσο και για τη Β, καλύτερο φαίνεται να είναι το κβαντισμένο μοντέλο EfficientNetLite4 στην CPU με 2 και 3 νήματα αντίστοιχα.

6.3 Περικοπή ή Κλιμάκωση;

Για να συγκριθούν οι δύο μέθοδοι προσαρμογής της εικόνας του αισθητήρα που περιγράφηκαν στην Υποενότητα 4.1.3, δημιουργήθηκαν τρία μικρά σύνολα δεδομένων των 35 εικόνων με φωτογραφίες οικιακών αντικειμένων. Οι αναλογίες των εικόνων στα τρία αυτά σύνολα ήταν 20:9, 16:9 και 4:3. Στην συνέχεια, αυτές οι εικόνες ταξινομήθηκαν δύο φορές με το κβαντισμένο MobileNetV2 1.0, την πρώτη φορά εφαρμόζοντας περικοπή και τη δεύτερη κλιμάκωση. Η έξοδος της αναγνώρισης κάθε εικόνας ήταν οι τρεις πιο πιθανές κατηγορίες μαζί με την πιθανότητα ή πεποίθηση (confidence) του νευρωνικού για κάθε μια.

Στον Πίνακα 6.4 έχουν αρχικά υπολογιστεί οι Top-1 και Top-3 ορθότητες. Η στήλη «Κοινά» δείχνει πόσα από τα 35 αντικείμενα ταξινομήθηκαν σωστά από κοινού, δηλαδή και με τις δύο μεθόδους και η στήλη «Μεγαλύτερη πεποίθηση» δείχνει τα ποσοστά των κοινών σωστών κατηγοριοποιήσεων στις οποίες η κάθε μέθοδος είχε τη μεγαλύτερη πεποίθηση.

Αναλογία	Μέθοδος	Top-1 Ορθότητα	Top-3 Ορθότητα	Κοινά	Μεγαλύτερη πεποίθηση
20:9	Περικοπή	82.8 %	97.1 %	17/35	52.9 % (9/17)
	Κλιμάκωση	60.0 %	74.3 %		35.3 % (6/17)
16:9	Περικοπή	65.7 %	85.7 %	18/35	66.7 % (12/18)
	Κλιμάκωση	65.7 %	85.7 %		27.8 % (5/18)
4:3	Περικοπή	71.2 %	82.8 %	21/35	38.1 % (8/21)
	Κλιμάκωση	68.6 %	85.7 %		57.1 % (12/21)

Πίνακας 6.4: *Αποτελέσματα σύγκρισης των μεθόδων περικοπής και κλιμάκωσης*

Όπως φαίνεται από τις παραπάνω μετρήσεις, στις αναλογίες 20:9 και 16:9 θα πρέπει να προτιμηθεί η μέθοδος περικοπής, ενώ στην αναλογία 4:3 η μέθοδος της κλιμάκωσης. Αυτό το αποτέλεσμα είναι αναμενόμενο, λογικό και μπορεί να γενικευτεί, καθώς στις πιο μακρόστενες εικόνες, η κλιμάκωση αλλοιώνει τις διαστάσεις του αντικειμένου, οπότε η αναγνώριση αποτυγχάνει, ενώ στις εικόνες με διαστάσεις πιο κοντά στο τετράγωνο, η κλιμάκωση μεταβάλλει ελάχιστα τις διαστάσεις της εικόνας και άρα ο κίνδυνος να απορριφθούν σημαντικά pixels της εικόνας με την περικοπή είναι μεγαλύτερος.

Κεφάλαιο 7

Επίλογος

Στο τελευταίο Κεφάλαιο αναλύονται τα συμπεράσματα, η συνεισφορά της διπλωματικής εργασίας αλλά και οι τελικές παρατηρήσεις - σημειώσεις. Ο Επίλογος κλείνει με την αναφορά κατευθύνσεων προς τις οποίες θα μπορούσε να επεκταθεί στο μέλλον η διπλωματική εργασία.

7.1 Συμπεράσματα

Όπως φάνηκε ιδιαίτερα στο Κεφάλαιο 5, δεν δόθηκε ιδιαίτερη έμφαση στην εφαρμογή καθαυτή αλλά στην λειτουργικότητά της. Ο σκοπός ανάπτυξης της εφαρμογής ήταν κυρίως η χρήση της ως ένα εργαλείο που θα έδινε τη δυνατότητα να πραγματοποιηθούν οι μετρήσεις, ώστε να συγκριθούν τα μοντέλα και οι συσκευές.

Τα τρία μοντέλα που ξεχώρισαν στην Ενότητα 6.2 συνοψίζονται στον Πίνακα 7.1. Όπως ήταν αναμενόμενο, βλέπουμε ότι κανένα μοντέλο δεν είναι τέλειο και ότι υπάρχουν σχέσεις αλληλεξάρτησης (trade-offs) μεταξύ τους. Για παράδειγμα, αν ενδιαφέρει μόνο η ορθότητα, τότε προτείνεται το κβαντισμένο EfficientNetLite4, ενώ αν εδιαφέρει το μέγεθος του μοντέλου, τότε το κβαντισμένο EfficientNetLite0 είναι μονόδρομος. Ωστόσο, ακόμα κι αν επιζητούμε την καλύτερη ορθότητα, μπορεί τα 431 ms του συνολικού χρόνου αναγνώρισης στην συσκευή Α να μην είναι αποδεκτά. Αν έπρεπε να επιλέξουμε ένα μοντέλο που να ικανοποιεί σε κάποιο βαθμό όλα τα κριτήρια, τότε αυτό θα ήταν το κβαντισμένο EfficientNetLite0 με χρήση της CPU και 2 νημάτων, το οποίο μας παρέχει και μεγαλύτερη ασφάλεια από τα υπόλοιπα.

Α/Α	Μοντέλο	Top-1 Ορθότητα	Μέγεθος	Συσκευή Α		Συσκευή Β	
				Ολικό	Μνήμη	Ολικό	Μνήμη
3	EfficientNetLite0 quantized	74.4 %	4.482 MB	183	84 MB	78	125 MB
5	EfficientNetLite0	75.1 %	17.929 MB	223	101 MB	75	210 MB
9	EfficientNetLite4 quantized	80.2 %	12.398 MB	431	100 MB	145	135 MB

Πίνακας 7.1: *Τα βέλτιστα μοντέλα*

Μια ακόμη σημαντική διαπίστωση αποτελεί η δυνατότητα της Βαθιάς Μάθησης να εισχωρήσει ακόμη και σε πιο παλαιές συσκευές, οι οποίες δεν έχουν ισχυρούς επεξεργαστές ή επιταχυντές και όταν κατασκευάστηκαν δεν υπήρχαν καν σκέψεις για την υποστήριξη τέτοιου είδους εφαρμογών. Συγκεκριμένα, όπως αναφέρθηκε στα αποτελέσματα της Υποε-

νότητας 6.2.2, η συσκευή Α αδυνατεί να εκτελέσει τα πιο «βαριά» μοντέλα δίχως η εφαρμογή να «κολλάει» και άρα τα μόνα μοντέλα που μπορεί να εκτελέσει είναι είτε σχεδιασμένα εξ αρχής με σκοπό το να είναι «ελαφριά» (MobileNetV2, EfficientNetLite0) ή είναι κβαντισμένα. Επομένως, οι πιο πρόσφατες τεχνικές σχεδίασης «ελαφριών» δικτύων, καθώς και οι μέθοδοι κβαντισμού είναι αποφασιστικοί παράγοντες πίσω από την εισχώρηση μοντέλων Βαθιάς Μάθησης ακόμη και σε παλαιότερες συσκευές.

Αξίζει να σημειωθεί ότι τα αποτελέσματα που προβάλλονται στην παρούσα εργασία αφορούν αποκλειστικά αυτή, δηλαδή για παράδειγμα οι χρόνοι που μετρήθηκαν για το inference latency αφορούν την εκτέλεση της συμπερασματολογίας σε μια εφαρμογή στην οποία χρησιμοποιείται παράλληλα και η κάμερα της συσκευής. Οι μετρήσεις θα ήταν διαφορετικές σε οποιαδήποτε άλλη περίπτωση ή εφαρμογή. Από τους αναλυτικούς πίνακες του Παραρτήματος Α΄ επιβεβαιώνεται τα ακόλουθα:

- Η χρήση του NNAPI δεν έχει πάντα τα καλύτερα αποτελέσματα. Για παράδειγμα, στα κβαντισμένα EfficientNet το NNAPI έχει τους χειρότερους χρόνους, ενώ στα ίδια floating point μοντέλα έχει πολύ καλές επιδόσεις. Αυτό σημαίνει ότι τα κβαντισμένα EfficientNet δεν είναι βελτιστοποιημένα είτε για τον επιταχυντή της συγκεκριμένης συσκευής στην οποία αξιολογήθηκαν, είτε γενικότερα για κανέναν επιταχυντή.

- Η χρήση της GPU για τη συμπερασματολογία έχει τα καλύτερα αποτελέσματα και στις δύο συσκευές. Ωστόσο, στη συγκεκριμένη εφαρμογή τη GPU απασχολεί και η κάμερα, οπότε σε κάποια μοντέλα αυτή η ταυτόχρονη χρήση επηρεάζει την προεπισκόπηση της κάμερας, οπότε η κάμερα θέτει έναν επιπλέον περιορισμό στην ενσωμάτωση μοντέλων Βαθιάς Μάθησης σε κινητές συσκευές.

- Όσον αφορά στη χρησιμοποίηση της μνήμης, παρατηρείται αρχικά ότι αυτή παραμένει σταθερή σε οποιονδήποτε αριθμό νημάτων. Συγκρίνοντας τις τρεις επιλογές επεξεργαστών, βλέπουμε ότι στο ίδιο μοντέλο, με τη χρήση του NNAPI χρησιμοποιείται λιγότερη μνήμη από την περίπτωση χρήσης της CPU, ενώ με τη χρήση της GPU η μνήμη σχεδόν διπλασιάζεται σε σχέση με αυτή που χρησιμοποιείται με τη CPU. Επομένως, εκτός από τις «διακοπές» που μπορεί να προκαλεί στην προεπισκόπηση, η μεγάλη χρήση μνήμης είναι ακόμη ένα στοιχείο που αποθαρρύνει την επιλογή τους. Βέβαια, η χρησιμοποίηση της μνήμης είναι ένα σχετικό μέγεθος, δηλαδή εξαρτάται τόσο από την συσκευή, όσο και από τον έλεγχο που έχει ο προγραμματιστής στους πόρους αυτής.

Συμπερασματικά, θα λέγαμε ότι η επιλογή της μονάδας επεξεργασίας εξαρτάται και από την πλατφόρμα, αλλά και από τις απαιτήσεις απόδοσης της εφαρμογής και τον βαθμό ελέγχου των πόρων από τον προγραμματιστή.

7.2 Μελλοντική εργασία

Το σύστημα που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας μπορεί να βελτιωθεί και να επεκταθεί περαιτέρω, με προσθήκες ή δοκιμάζοντας κάποιες παραλλαγές των βασικών δομικών στοιχείων της τουλάχιστον ως προς πέντε κατευθύνσεις, οι οποίες παρουσιάζονται παρακάτω.

Προσαρμογή παραμέτρων κάμερας

Όπως αναφέρθηκε και στην Υποενότητα 4.1.2, το αποτέλεσμα της αναγνώρισης μπορεί να αξιοποιηθεί περαιτέρω με την έννοια βελτιστοποίησης των παραμέτρων της κάμερας. Για παράδειγμα, αν μπροστά από την κάμερα βρίσκεται κάποιο φαγητό, θέλουμε πιο ζωηρά χρώματα ή σε συνθήκες χαμηλού φωτισμού θέλουμε υψηλή φωτεινότητα. Η αρχιτεκτονική του συστήματος επιτρέπει την συγκεκριμένη επέκταση, λόγω του σχεδιασμού της.

Μηχανισμός Επιλογής Μοντέλου και Προσαρμογής Παραμέτρων

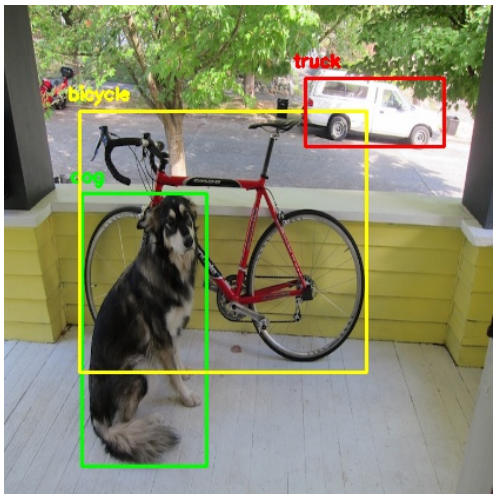
Από την αξιολόγηση των δέκα μοντέλων φάνηκε ότι πολλά από αυτά δεν έχουν τη δυνατότητα να υποστηριχθούν από όλες τις συσκευές και επίσης δεν αποδίδουν όλα καλά με την ίδια διαμόρφωση, δηλαδή επεξεργαστή ή/και αριθμό νημάτων (βλ. Πίνακα 6.2). Στην περίπτωση που η εφαρμογή έβγαινε στην αγορά και προσέγγιζε πολλούς χρήστες και άρα πολλές συσκευές, ο από πριν καθορισμός του μοντέλου και του επεξεργαστή τον οποίο θα χρησιμοποιήσει το μοντέλο σίγουρα δεν είναι μια καλή τακτική. Ήδη από τις μετρήσεις στο πλαίσιο της παρούσας διπλωματικής εργασίας με δύο μόνο συσκευές, είδαμε πόσο διαφορετικά συμπεριφέρονται τα μοντέλα.

Για τον παραπάνω λόγο προτείνεται η ενσωμάτωση δύο μοντέλων στην εφαρμογή, ενός πιο «ελαφριού» κι ενός πιο «βαριού» και η ανάπτυξη ενός ανεξάρτητου στοιχείου (component) - μηχανισμού, ο οποίος κάθε φορά που η εφαρμογή θα εγκαθίσταται σε μια νέα συσκευή, θα εκτελεί μια σειρά από δοκιμές αναγνώρισης σε ένα μικρό σύνολο εικόνων, ώστε να αποφασίσει ποιο από τα δύο μοντέλα είναι πιο κατάλληλο καθώς και σε ποια διαμόρφωση.

Μοντέλα

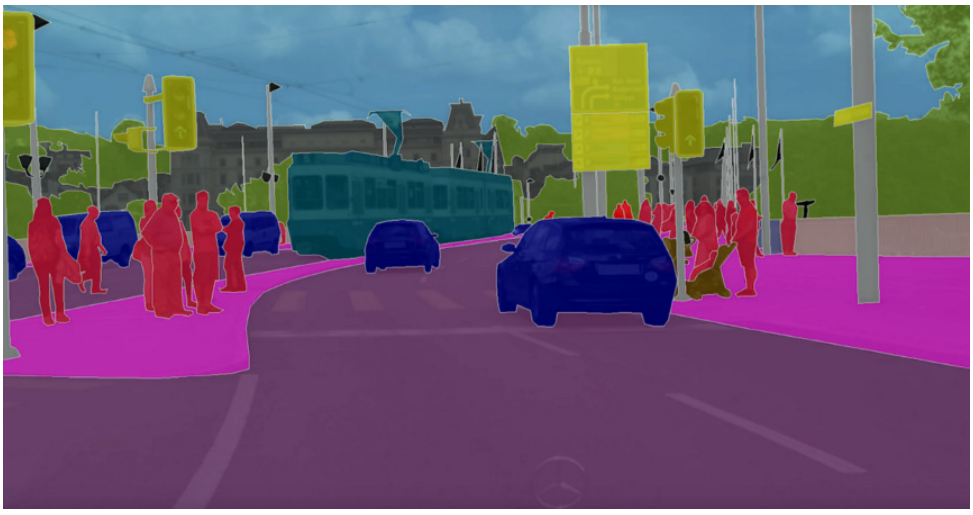
Όλα τα μοντέλα που χρησιμοποιήθηκαν στην παρούσα εργασία ήταν συνελικτικά νευρωνικά δίκτυα (CNNs), τα οποία είχαν εκπαιδευτεί στην διεργασία (task) της αναγνώρισης εικόνας, δηλαδή στο να δέχονται ως είσοδο μια εικόνα και να έχουν ως έξοδο μια ετικέτα από τις κλάσεις στις οποίες έχουν εκπαιδευτεί. Τι θα συμβεί ωστόσο αν σε μια εικόνα υπάρχουν δύο ή περισσότερα αντικείμενα; Αυτό ίσως συμβαίνει στις περισσότερες περιπτώσεις, οπότε σε μια πρακτική εφαρμογή, τα μοντέλα αναγνώρισης δεν είναι και τόσο κατάλληλα. Ευτυχώς η δύναμη των συνελικτικών νευρωνικών δικτύων δεν σταματά εκεί. Στην Όραση Υπολογιστών, εκτός από την αναγνώριση - ταξινόμηση εικόνων, υπάρχουν κι άλλα tasks στα οποία τα συνελικτικά δίκτυα τα πηγαίνουν αρκετά καλά, όπως:

- Η ανίχνευση - εντοπισμός αντικειμένων σε εικόνες (object detection), που έχει εφαρμοσθεί ευρέως σε παρακολούθηση βίντεο (surveilance), αυτόνομα αυτοκίνητα (self-driving cars), κ.α. Με αυτή την τεχνική, μπορεί να εντοπιστεί η παρουσία ενός αντικειμένου σε μια εικόνα και να σχεδιαστεί ένα πλαίσιο γύρω από αυτό. Κάποιες γνωστές αρχιτεκτονικές που χρησιμοποιούνται για ανίχνευση αντικειμένων είναι το R-CNN και οι εξελίξεις του (Fast R-CNN, Faster R-CNN, Mask R-CNN), το SSD (Single Shot MultiBox Defender) και το YOLO (You Only Look Once) [44] [45].



Σχήμα 7.1: *Ανίχνευση αντικειμένων*

- Η κατάτμηση εικόνων (segmentation), ένα πιο περίπλοκο task, κατά το οποίο η εικόνα πρέπει να διαιρεθεί σε πολλαπλά τμήματα ή με άλλα λόγια κάθε pixel της εικόνας πρέπει να συσχετιστεί με κάποιον τύπο αντικειμένου. Οι εφαρμογές της κατάτμησης περιλαμβάνουν αυτόνομα αυτοκίνητα, συστήματα ελέγχου κυκλοφορίας, εντοπισμός αντικειμένων σε δορυφορικές εικόνες, η ανίχνευση καρκινικών κυττάρων, κ.α. Γνωστές αρχιτεκτονικές δικτύων για αυτό το task είναι το U-Net, το FastFCN, το Gated-SCNN, το Gated-SCNN και το DeepLab [46] [47] [48].



Σχήμα 7.2: *Κατάτμηση*

- Η εκτίμηση πόζας - στάσης (pose estimation), η οποία στοχεύει στην αντιστοίχηση των pixels ενός ανθρώπου σε μια εικόνα στην τρισδιάστατη επιφάνεια του ανθρώπινου σώματος με εφαρμογές επαυξημένης πραγματικότητας (augmented reality - AR), αλληλεπίδρασης ανθρώπου-υπολογιστή (human-computer interaction - HCI) κ.α. Κάποιες από τις αρχιτεκτονικές που χρησιμοποιούνται είναι το DensePose, το OpenPose, το AlphaPose και το DeepPose [49] [50].

Σχήμα 7.3: *Εκτίμηση πόζας*

Εκτός από τα συνελικτικά δίκτυα, τα τελευταία χρόνια έχουν προσεγγίσει την Όραση Υπολογιστών και τα αναδρομικά δίκτυα (RNNs). Τα RNNs παραδοσιακά είναι κατάλληλα για την επεξεργασία ακολουθιακών δεδομένων, όπως είναι ο λόγος, προφορικός ή γραπτός. Ωστόσο, ακολουθία είναι και ένα βίντεο, δηλαδή ένα σύνολο από εικόνες. Πολλοί επιστήμονες έχουν στραφεί στην ανάπτυξη μοντέλων που συνδυάζουν συνελικτικά και αναδρομικά δίκτυα και αυτό ίσως είναι το επόμενο βήμα για την Όραση Υπολογιστών. Οι εφαρμογές που έχουν προκύψει αφορούν την αυτόματη περιγραφή εικόνων (image captioning), την εξαγωγή συναισθήματος από βίντεο, κ.α. [51]

Gallery

Την «έξυπνη» κάμερα θα μπορούσε να πλαισιώσει ένα γκαλερί (gallery), στο οποίο θα αποθηκεύονται οι φωτογραφίες του χρήστη μαζί με την ετικέτα τους με στόχο την αναζήτηση εικόνων με βάση το περιεχόμενο, ή και τη δημιουργία κολάζ ή συλλογών. Η κάμερα δεν θα είναι η μόνη πηγή φωτογραφιών για το gallery, αλλά ο χρήστης θα μπορεί να εισάγει φωτογραφίες και από άλλες πηγές, όπως για παράδειγμα εικόνες που έχει ήδη στην συσκευή του από άλλες εφαρμογές. Η κατηγοριοποίηση αυτών τον εικόνων δεν είναι απαραίτητο να γίνεται online, δηλαδή με την αλληλεπίδραση του χρήστη, οπότε το μοντέλο που θα εκτελεί αυτή την κατηγοριοποίηση θα έχει διαφορετικά χαρακτηριστικά από αυτό ή αυτά που εκτελούν την αναγνώριση σε πραγματικό χρόνο. Με μια πρώτη σκέψη, ένα κατάλληλο δίκτυο για την εφαρμογή πραγματικού χρόνου θα εκτελεί ανίχνευση αντικειμένων, ενώ για το gallery θα ταίριαζε ένα μοντέλο που θα εξάγει μια λεκτική περιγραφή για την κάθε εικόνα.

Κατηγορίες

Όπως αναφέρθηκε και προηγουμένως, τα μοντέλα που χρησιμοποιήθηκαν έχουν προεκπαιδευτεί στο σύνολο δεδομένων ImageNet, οπότε μπορούν να αναγνωρίσουν μέχρι 1000 αντικείμενα. Σε πραγματικές εφαρμογές, πολλά από αυτά τα αντικείμενα εμφανίζονται σπάνια ή ακόμη και ποτέ μπροστά από τον φακό μιας κινητής συσκευής, οπότε οι αντίστοιχες κλάσεις «περισσεύουν» και το νευρωνικό δίκτυο δεν χρησιμοποιείται στο μέγιστο.

Αντί λοιπόν για ένα τέτοιο νευρωνικό, θα μπορούσε να εκπαιδευτεί ένας αριθμός από νευρωνικά δίκτυα, τα οποία θα ειδικεύονται σε μια συγκεκριμένη κατηγορία αντικειμένων, όπως για παράδειγμα οικιακά αντικείμενα, ζώα, φυτά, αντικείμενα εκτός σπιτιού, φαγητά, κ.α. Με αυτόν τον τρόπο, η ορθότητα θα αυξηθεί και ο χρήστης θα μπορεί να εναλλάσσει μεταξύ των νευρωνικών ανάλογα με την κατηγορία που τον ενδιαφέρει κάθε φορά.

Παραρτήματα

Παράρτημα Α΄

Αναλυτικοί Πίνακες Μετρήσεων

Παρακάτω οι παρατίθενται αναλυτικοί πίνακες μετρήσεων του σταδίου **3** του pipeline αναγνώρισης, του ολικού latency (**T**otal) και της χρήσης μνήμης. Τα πράσινα κελιά δείχνουν τους καλύτερους χρόνους για τις μετρικές Mean, Median και 90th percentile, τα κίτρινα τους καλύτερους χρόνους των οποίων όμως η διαμόρφωση προκαλεί «διακοπές» στην ομαλή προεπισκόπηση και τα κόκκινα τις αντίστοιχες απαγορευτικές διαμορφώσεις. Τα χαρακτηριστικά των δέκα μοντέλων και τα χαρακτηριστικά των δύο συσκευών Α και Β παρουσιάστηκαν στους Πίνακες 5.1 και 5.2 αντίστοιχα.

Α΄.1 Συσκευή Α

MobileNetV2 1.0 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	114	156	195	258	127.2	182.1	122	175	146	211	80 MB
CPU (2)	62	104	147	207	80.0	136.4	78	136	94	157	
CPU (3)	45	86	177	243	72.9	129.9	66	119	119	182	
CPU (4)	35	75	198	255	76.4	137.5	51	118	162	211	
CPU (5)	31	72	197	259	96.4	155.1	77	140	174	229	

Πίνακας Α΄.1: *MobileNetV2 1.0 quantized - Συσκευή Α*

MobileNetV2 1.0

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	190	240	257	330	206.3	271.2	200	267	225	302	98 MB
CPU (2)	109	157	163	249	127.8	191.2	131	191	144	216	
CPU (3)	82	131	167	245	104.6	168.4	103	165	120	194	
CPU (4)	69	118	226	285	113.8	177.2	92	156	201	264	
CPU (5)	63	112	228	298	125.1	190.2	113	188	207	266	

Πίνακας Α΄.2: *MobileNetV2 1.0 - Συσκευή Α*

MobileNetV2 1.4

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	326	380	384	437	348.8	408.0	351	409	359	423	110 MB
CPU (2)	182	231	281	345	200.7	257.9	194	252	222	287	
CPU (3)	137	188	203	276	161.8	221.3	156	220	189	253	
CPU (4)	115	165	260	320	145.6	205.5	139	199	189	249	
CPU (5)	106	155	258	342	155.6	218.7	134	196	232	295	

Πίνακας Α΄.3: *MobileNetV2 1.4 - Συσκευή Α*

EfficientNet Lite0 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	151	194	209	277	167.9	225.6	160	223	190	254	84 MB
CPU (2)	82	123	138	204	99.7	156.2	100	152	117	183	
CPU (3)	59	103	193	247	87.9	145.1	81	135	129	197	
CPU (4)	45	87	206	274	90.1	146.0	65	123	177	225	
CPU (5)	41	82	205	261	100.9	158.5	95	160	173	223	

Πίνακας Α΄.4: *EfficientNet Lite0 quantized - Συσκευή Α*

EfficientNet Lite0

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	243	296	306	372	260.4	319.9	259	318	272	339	101 MB
CPU (2)	135	184	207	269	155.8	217.8	149	218	174	242	
CPU (3)	107	157	201	267	132.4	192.1	134	192	155	223	
CPU (4)	88	138	239	296	128.0	188.6	114	176	202	267	

Πίνακας Α΄.5: *EfficientNet Lite0 - Συσκευή Α*

InceptionV3 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	1315	1381	1387	1488	1359.4	1440.4	1362	1440	1376	1462	115 MB
CPU (2)	700	776	762	847	729.3	810.0	727	811	746	828	
CPU (3)	503	572	559	644	519.9	602.9	517	603	534	619	
CPU (4)	398	466	475	567	418.6	500.5	409	498	454	531	

Πίνακας Α΄.6: *InceptionV3 quantized - Συσκευή Α*

EfficientNet Lite4 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	838	908	904	991	864.4	943.1	865	943	880	964	100 MB
CPU (2)	436	505	486	579	459.4	540.6	454	540	477	560	
CPU (3)	310	379	371	460	328.1	411.3	323	412	346	431	
CPU (4)	238	307	352	424	262.2	346.8	252	345	289	373	

Πίνακας Α΄.7: *EfficientNet Lite4 quantized - Συσκευή Α*

EfficientNet Lite4

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	1372	1457	1457	1552	1419.8	1513.9	1423	1515	1437	1535	150 MB
CPU (2)	758	847	851	939	775.6	873.6	774	872	785	890	
CPU (3)	556	638	631	748	578.0	681.2	574	681	601	701	

Πίνακας Α΄.8: *EfficientNet Lite4 - Συσκευή Α*

Α΄.2 Συσκευή Β

MobileNetV2 1.0 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	16	39	60	96	37.3	70.3	38	70	46	83	128 MB
CPU (2)	12	25	61	97	28.9	63.8	28	64	35	76	
CPU (3)	24	60	87	132	49.5	83.6	49	81	61	100	
CPU (4)	19	45	84	125	57.4	92.7	59	93	69	111	
CPU (5)	16	39	60	96	37.3	70.3	38	70	46	83	
NNAPI	7	26	15	64	8.9	44.2	8	44	11	54	120 MB

Πίνακας Α΄.9: *MobileNetV2 1.0 quantized - Συσκευή Β*

MobileNetV2 1.0

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	39	57	109	148	66.1	104.4	66	105	82	128	137 MB
CPU (2)	32	57	76	113	44.2	81.3	44	81	50	92	
CPU (3)	46	79	85	127	66.0	104.1	66	103	74	117	
CPU (4)	42	78	95	144	69.3	107.6	70	107	82	123	
NNAPI	21	44	59	127	26.9	68.8	25	69	33	81	127 MB
GPU	16	41	29	81	19.4	60.5	19	60	22	71	210 MB

Πίνακας Α'.10: *MobileNetV2 1.0 - Συσκευή B*

MobileNetV2 1.4

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	65	89	130	180	101.3	139.8	109	144	122	167	153 MB
CPU (2)	47	78	102	143	62.9	101.9	62	99	73	118	
CPU (3)	64	97	100	152	83.1	124.6	83	124	92	137	
CPU (4)	66	96	108	154	84.8	124.6	84	124	96	139	
NNAPI	32	56	52	108	37.3	80.1	37	80	41	88	136 MB
GPU	20	43	34	87	23.2	63.9	23	63	26	77	240 MB

Πίνακας Α'.11: *MobileNetV2 1.4 - Συσκευή B*

EfficientNet Lite0 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	22	45	65	112	47.5	80.6	48	80	56	96	125 MB
CPU (2)	14	39	58	100	33.5	67.3	33	66	41	78	
CPU (3)	35	61	84	135	60.6	95.7	60	95	72	112	
CPU (4)	33	60	92	132	64.6	99.8	65	99	78	119	
NNAPI	346	368	396	444	375.5	407.2	375	406	388	422	132 MB

Πίνακας Α'.12: *EfficientNet Lite0 quantized - Συσκευή B*

EfficientNet Lite0

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	48	77	110	162	78.4	119.2	77	118	97	143	140 MB
CPU (2)	33	57	76	125	52.3	91.0	52	89	62	107	
CPU (3)	63	83	106	153	78.1	118.8	78	118	87	133	
CPU (4)	56	80	107	164	79.5	121.3	79	122	90	137	
NNAPI	27	53	42	105	31.2	69.0	30	68	36	79	132 MB
GPU	16	37	29	88	18.5	63.0	18	63	21	75	210 MB

Πίνακας Α΄.13: *EfficientNet Lite0 - Συσκευή Β*

InceptionV3 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	135	159	222	287	169.4	214.7	158	203	209	262	155 MB
CPU (2)	90	117	191	222	111.2	151.0	109	149	126	170	
CPU (3)	97	137	141	209	119.6	166.1	120	165	131	184	
CPU (4)	97	127	153	209	118.0	165.1	118	165	129	182	
NNAPI	85	117	126	186	103.2	152.1	103	151	116	173	140 MB

Πίνακας Α΄.14: *InceptionV3 quantized - Συσκευή Β*

InceptionV3

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	3	T	3	T	3	T	3	T	3	T	
CPU (1)	494	523	611	648	509.1	553.0	506	550	520	568	262 MB
CPU (2)	305	336	478	521	323.1	373.6	318	369	342	394	
CPU (3)	308	343	473	518	336.2	386.9	333	386	356	409	
CPU (4)	296	344	365	430	331.8	385.2	331	386	348	407	
NNAPI	198	230	236	304	210.3	264.0	208	264	222	279	208 MB
GPU	132	173	156	235	138.1	194.3	136	194	146	205	400 MB

Πίνακας Α΄.15: *InceptionV3 - Συσκευή Β*

ResnetV2 101

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	Σ	Τ	Σ	Τ	Σ	Τ	Σ	Τ	Σ	Τ	
CPU (1)	1070	1110	1331	1376	1220.2	1269.4	1293	1339	1303	1360	330 MB
CPU (2)	645	691	1960	2015	758.0	807.5	756	804	791	838	
CPU (3)	671	724	830	870	744.8	792.6	744	792	772	821	
CPU (4)	653	699	764	817	686.7	738.2	685	737	709	766	
CPU (5)	596	635	735	768	638.6	690.7	635	688	665	722	
NNAPI	2460	2495	2959	3015	2801.7	2853.3	2817	2869	2890	2941	300 MB
GPU	434	474	464	535	449.1	503.6	449	503	455	521	565 MB

Πίνακας Α'.16: *ResnetV2 101 - Συσκευή Β*

EfficientNet Lite4 quantized

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	Σ	Τ	Σ	Τ	Σ	Τ	Σ	Τ	Σ	Τ	
CPU (1)	110	134	166	228	139.0	183.9	136	180	160	214	135 MB
CPU (2)	68	94	166	193	88.3	130.7	87	128	98	145	
CPU (3)	121	161	186	229	142.1	185.8	141	183	153	201	
CPU (4)	116	145	170	222	143.0	191.0	143	191	156	207	
NNAPI	2306	2338	2348	2406	2330.0	2376.2	2330	2377	2339	2393	170 MB

Πίνακας Α'.17: *EfficientNet Lite4 quantized - Συσκευή Β*

EfficientNet Lite4

Επεξεργαστής	Min		Max		Mean		Median		90th percentile		Χρήση μνήμης
	Σ	Τ	Σ	Τ	Σ	Τ	Σ	Τ	Σ	Τ	
CPU (1)	257	285	348	425	275.2	330.0	271	324	280	344	190 MB
CPU (2)	253	279	348	425	276.0	329.9	271	324	281	346	
CPU (3)	215	246	326	385	245.9	297.5	245	295	255	316	
CPU (4)	227	273	278	328	245.5	298.3	245	296	256	316	
NNAPI	142	168	182	247	156.0	209.0	156	207	166	226	173 MB
GPU	92	120	119	191	98.7	153.6	98	152	104	170	325 MB

Πίνακας Α'.18: *EfficientNet Lite4 - Συσκευή Β*

Βιβλιογραφία

[1] Ian J. Goodfellow, Yoshua Bengio και Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[2] *Let's Understand the Difference between Machine Learning Vs. Deep Learning*. https://www.inteliment.com/blog/our-thinking/lets-understand-the-difference-between-machine-learning-vs-deep-learning/. Ημερομηνία πρόσβασης: 20 Οκτωβρίου 2020.

[3] Li Deng και Dong Yu. *Deep Learning: Methods and Applications*. Τεχνική Αναφορά με αριθμό MSR-TR-2014-21, Microsoft, 2014.

[4] *What Is Deep Learning?* https://www.mathworks.com/discovery/deep-learning.html. Ημερομηνία πρόσβασης: 15 Αυγούστου 2020.

[5] Carole-Jean Wu, *et al.* *Machine Learning at Facebook: Understanding Inference at the Edge*. *25th IEEE International Symposium on High Performance Computer Architecture, HPCA 2019, Washington, DC, USA, February 16-20, 2019*, σελίδες 331–344. IEEE, 2019.

[6] *Notes for the Stanford CS class "CS231n: Convolutional Neural Networks for Visual Recognition"*. https://cs231n.github.io. Ημερομηνία πρόσβασης: 19 Αυγούστου 2020.

[7] Michael Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/.

[8] Y. Lecun, L. Bottou, Y. Bengio και P. Haffner. *Gradient-based learning applied to document recognition*. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Alex Krizhevsky, Ilya Sutskever και Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, σελίδες 1106–1114, 2012.

[10] Christian Szegedy, *et al.* *Going deeper with convolutions*. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, σελίδες 1–9. IEEE Computer Society, 2015.

[11] Karen Simonyan και Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[12] Christian Szegedy, *et al.* *Rethinking the Inception Architecture for Computer Vision*. *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, σελίδες 2818–2826. IEEE Computer Society, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren και Jian Sun. *Deep Residual Learning for Image Recognition*. *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, σελίδες 770–778. IEEE Computer Society, 2016.

[14] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, σελίδες 1800–1807. IEEE Computer Society, 2017.

[15] Saining Xie, *et al.* *Aggregated Residual Transformations for Deep Neural Networks*. *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, σελίδες 5987–5995. IEEE Computer Society, 2017.

[16] Mingxing Tan και Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, τόμος 97 στο *Proceedings of Machine Learning Research*, σελίδες 6105–6114. PMLR, 2019.

[17] Forrest N. Iandola, *et al.* *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size*. *CoRR*, abs/1602.07360, 2016.

[18] Andrew G. Howard, *et al.* *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. *CoRR*, abs/1704.04861, 2017.

[19] Anirudh Koul, Siddha Ganju και Meher Kasam. *Practical Deep Learning for Cloud, Mobile, and Edge: Real-World AI Computer-Vision Projects Using Python, Keras TensorFlow*. O'Reilly Media, 1η έκδοση, 2019. https://www.practicaldeeplearning.ai.

[20] Ji Wang, *et al.* *Deep Learning towards Mobile Applications*. *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, σελίδες 1385–1393. IEEE Computer Society, 2018.

[21] Andrey Ignatov, *et al.* *AI Benchmark: All About Deep Learning on Smartphones in 2019*. *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, σελίδες 3617–3635. IEEE, 2019.

[22] Mario Almeida, Stefanos Laskaridis, Ilias Leontiadis, Stylianos I. Venieris και Nicholas D. Lane. *EmBench: Quantifying Performance Variations of Deep Neural Networks across Modern Commodity Devices*. *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, EMDL '19, σελίδα 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

[23] Yunbin Deng. *Deep Learning on Mobile Devices - A Review*. *CoRR*, abs/1904.09274, 2019.

[24] Manas Sahni. *8-Bit Quantization and TensorFlow Lite: Speeding up mobile inference with low precision*. https://heartbeat.fritz.ai/8-bit-quantization-and-tensorflow-lite-speeding-up-mobile-inference-with-low-precision-a882dfcafbbd. Ημερομηνία πρόσβασης: 8 Οκτωβρίου 2020.

[25] *Mobile Operating System Market Share Worldwide*. https://gs.statcounter.com/os-market-share/mobile/worldwide. Ημερομηνία πρόσβασης: 9 Οκτωβρίου 2020.

[26] John Callaham. *The history of Android: The evolution of the biggest mobile OS in the world*. https://www.androidauthority.com/history-android-os-name-789433/. Ημερομηνία πρόσβασης: 9 Οκτωβρίου 2020.

[27] *What is Android Studio?* https://searchmobilecomputing.techtarget.com/definition/Android-Studio. Ημερομηνία πρόσβασης: 9 Οκτωβρίου 2020.

[28] *Meet Android Studio*. https://developer.android.com/studio/intro. Ημερομηνία πρόσβασης: 9 Οκτωβρίου 2020.

[29] *Java*. https://www.britannica.com/technology/Java-computer-programming-language. Ημερομηνία πρόσβασης: 9 Οκτωβρίου 2020.

[30] Herbert Schildt. *Java: A Beginner's Guide*. McGraw-Hill Education, 7η έκδοση, 2017.

[31] *PYPL PopularitY of Programming Language*. https://pypl.github.io/PYPL.html. Ημερομηνία πρόσβασης: 9 Οκτωβρίου 2020.

[32] *TensorFlow Lite guide*. https://www.tensorflow.org/lite/guide. Ημερομηνία πρόσβασης: 10 Οκτωβρίου 2020.

[33] Renu Khandelwal. *A Basic Introduction to TensorFlow Lite*. https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292. Ημερομηνία πρόσβασης: 10 Οκτωβρίου 2020.

[34] *Mobile Tablet Android Version Market Share Worldwide*. https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide. Ημερομηνία πρόσβασης: 10 Οκτωβρίου 2020.

[35] *What is Camera2 and how to know if your mobile has it*. https://quickhax.com/what-is-camera2-how-to-know-if-your-mobile-has-it. Ημερομηνία πρόσβασης: 15 Οκτωβρίου 2020.

[36] *Better performance through threading*. https://developer.android.com/topic/performance/threads. Ημερομηνία πρόσβασης: 15 Οκτωβρίου 2020.

[37] *AI Camera*. https://github.com/ioannispan/ai-camera.

[38] *Application Fundamentals*. https://developer.android.com/guide/components/fundamentals. Ημερομηνία πρόσβασης: 17 Οκτωβρίου 2020.

[39] *Understand the Activity Lifecycle*. https://developer.android.com/guide/components/activities/activity-lifecycle. Ημερομηνία πρόσβασης: 18 Οκτωβρίου 2020.

[40] *ViewModel Overview*. https://developer.android.com/topic/libraries/architecture/viewmodel. Ημερομηνία πρόσβασης: 18 Οκτωβρίου 2020.

[41] *Mobilenet V2 Imagenet Checkpoints*. https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet#mobilenet-v2-imagenet-checkpoints. Ημερομηνία πρόσβασης: 18 Οκτωβρίου 2020.

[42] *Hosted models*. https://www.tensorflow.org/lite/guide/hosted_models. Ημερομηνία πρόσβασης: 18 Οκτωβρίου 2020.

[43] *EfficientNet-lite*. https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet/lite#efficientnet-lite. Ημερομηνία πρόσβασης: 18 Οκτωβρίου 2020.

[44] Derrick Mwiti. *A 2019 Guide to Object Detection*. https://heartbeat.fritz.ai/a-2019-guide-to-object-detection-9509987954c3. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

[45] Arun Ponnusamy. *YOLO Object Detection with OpenCV and Python*. https://www.arunponnusamy.com/yolo-object-detection-opencv-python.html. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

[46] Derrick Mwiti. *Image Segmentation in 2020: Architectures, Losses, Datasets, and Frameworks*. https://neptune.ai/blog/image-segmentation-in-2020. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

[47] Pulkit Sharma. *Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques (Part 1)*. https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

[48] Mario Salazarde Torres. *Semantic segmentation for dummies*. https://medium.com/@mario.sb93/semantic-segmentation-for-dummies-30206769383e. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

[49] *Deep Learning Models and Code for Pose Estimation*. https://modelzoo.co/blog/deep-learning-models-and-code-for-pose-estimation. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

[50] Herin Zhao. *Human Pose Estimation Model HRNet Breaks Three COCO Records; CVPR Accepts Paper*. https://syncedreview.com/2019/03/05/human-pose-estimation-model-hrnet-breaks-three-coco-records-cvpr-accepts-paper/. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

[51] Jeremy Cohen. *RNNs in Computer Vision*. https://thinkautonomous.medium.com/rnns-in-computer-vision-b04b438d805c. Ημερομηνία πρόσβασης: 21 Οκτωβρίου 2020.

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

βλ.	βλέπε
κ.	κύριος, κυρία
κ.α.	και άλλα
π.χ.	παραδείγματος χάρη
AI	Artificial Intelligence
ANN	artificial neural network
API	Application Programming Interface
AR	augmented reality
ASIC	Application-Specific Integrated Circuit
CL	Convergence Layer
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DLACL	Deep Learning Architecture Convergence Layer
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
FPS	frames per second
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
I/O	input/outpout
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MDCL	Mobile Device Convergence Layer
ms	millisecond
NNAPI	Neural Networks API
NPU	Neural Processing Unit
RAM	random-access memory
ReLU	rectified linear unit
RNN	Recurrent Neural Network
SIL	Service Independent Layer
SoC	System on a Chip
TPU	Tensor Processing Unit
UI	user interface
YOLO	You Only Look Once

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
ακατέργαστος	raw
Αναδρομικά Νευρωνικά Δίκτυα	Recurrent Neural Networks
αναλογία	ratio
ανεξαρτησία	independence
Ανίχνευση Αντικειμένων	Object Detection
απομακρυσμένος διακομιστής νέφους	cloud server
αρθρωτότητα	modularity
αυτόματη έκθεση	auto-exposure
Αυτόματη Περιγραφή Εικόνων	Image Captioning
αυτόνομο αυτοκίνητο	self-driving car
Βαθιά Μάθηση	Deep Learning
βάρος	weight
βηματισμός	stride
γκαλερί	gallery
δεδομένα	data
διάμεσος	median
Διεπαφή Προγραμματισμού Εφαρμογών	Application Programming Interface
Διεπαφή Χρήστη	User Interface
διεργασία	task
διερμηνέας	interpreter
δομικό στοιχείο	component
δομή	framework
δραστηριότητα	activity
εκατοστημόριο	percentile
εκπαίδευση	training
εκπρόσωπος	delegate
Εκτίμηση Πόζας	Pose Estimation
ελαφρύ	light-weight
ελάχιστο	minimum
ενέργεια	power
επαυξημένη πραγματικότητα	augmented reality
επίπεδο	level
επιταχυντής	accelerator
επιφάνεια	area

εφαρμογή	application
είσοδος	input
ετικέτα	label
ευελιξία	flexibility
κάμερα	camera
κανονικοποίηση	normalization
καρέ	frame
κατάβαση κλίσης	gradient descent
κατανεμημένος	distributed
κατασκευαστής	constructor
Κατάτμηση Εικόνων	Image Segmentation
κβαντοποίηση	quantization
κέντρα δεδομένων υπολογιστικού νέφους	cloud data centers
Κεντρική Μονάδα Επεξεργασίας	Central Processing Unit
κεντρικό νήμα εκτέλεσης	main execution thread
κλήση	callback
κλιμάκωση	scaling
κινητή συσκευή	mobile device
κινητή υποδιαστολή	floating point
κινητό τηλέφωνο	smartphone
κλάδεμα	pruning
με στόχο την προστασία της ιδιωτικότητας	privacy-preserving
μέγεθος	size
μεγέθυνση	zoom
μέγιστο	maximum
μεσισμικό	middleware
μέσος	mean
μετατροπέας	converter
Μεταφορά Μάθησης	Transfer Learning
μη-γραμμικότητα	non-linearity
μηδενικά παραγεμίσματος	zero-padding
Μηχανική Μάθηση	Machine Learning
Μονάδα Επεξεργασίας Γραφικών	Graphics Processing Unit
Μονάδα Επεξεργασίας Νευρώνων	Neural Processing Unit
Μονάδα Επεξεργασίας Τανυστών	Tensor Processing Unit
μοντέλο	model
μπάρα προόδου	progress bar
νήμα	thread
νήμα εργάτη	worker thread
ολοκληρωμένο κύκλωμα	chip
Ολοκληρωμένο Κύκλωμα Συγκεκριμένης Εφαρμογής	Application-Specific Integrated Circuit
Ολοκληρωμένο Προγραμματιστικό	Integrated Development

Περιβάλλον	Environment
οπίσθιος	back facing
ορθότητα	accuracy
παρακολούθηση	surveilance
πελάτης	client
πεποίθηση	confidence
περικοπή	cropping
πλήρως συνδεδεμένο	fully-connected
πραγματικού χρόνου	real-time
πρόγραμμα περιήγησης ιστού	web browser
πυρήνας	kernel
σιγμοειδής	sigmoid
συμπερασματολογία	inference
στρώμα	layer
συνάρτηση ενεργοποίησης	activation function
Συνελικτικά Νευρωνικά Δίκτυα	Convolutional Neural Networks
συνέλιξη	convolution
συντηρησιμότητα	maintainability
συσκευή στην άκρη του δικτύου	edge device
Συστοιχία Επιτόπια Προγραμματιζόμενων Πυλών	Field-Programmable Gate Array
σχεδιασμένο με το χέρι	hand-designed
σχέση αλληλεξάρτησης	trade-off
σχήμα διαμοιρασμού παραμέτρων	parameter sharing
συγκέντρωση	pooling
συνολικός	total
σύνολο δεδομένων	dataset
σωλήνωση	pipeline
ταξινομητής	classifier
Τεχνητή Νοημοσύνη	Artificial Intelligence
τεχνητό νευρωνικό δίκτυο	artificial neural network
τοπικό δεκτικό πεδίο	local receptive field
υλικό	hardware
υπολογιστής υψηλών αποδόσεων	high-performance computer
φλας	flash
φίλτρο	filter
χαμηλού επιπέδου	low-level
χαρακτηριστικό	feature
χάρτης	map
χειρότερη δυνατή περίπτωση	worst-case scenario
χιλιοστό του δευτερολέπτου	millisecond
χρόνος εκτέλεσης	latency