ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Αυτο-υποστηριζόμενο Ευφυές Σύστημα Αποθήκευσης και Κατηγοριοποίησης Εικόνων για Κινητές Συσκευές με χρήση Μοντέλων Βαθιάς Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΟΣ Σ. ΚΑΡΑΓΙΩΡΓΟΣ

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Αυτο-υποστηριζόμενο Ευφυές Σύστημα Αποθήκευσης και Κατηγοριοποίησης Εικόνων για Κινητές Συσκευές με χρήση Μοντέλων Βαθιάς Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΟΣ Σ. ΚΑΡΑΓΙΩΡΓΟΣ

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Νοεμβρίου 2021.

.....	.....	.....
Ιάκωβος Βενιέρης	Δήμητρα-Θεοδώρα Κακλαμάνη	Αθανάσιος Παναγόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2021

.....
Γεώργιος Σ. Καραγιώργος

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

Περίληψη

Στην παρούσα διπλωματική εργασία αναπτύχθηκε μια εφαρμογή αποθήκευσης και κατηγοριοποίησης εικόνων για κινητές συσκευές με λειτουργικό σύστημα Android, βασισμένη σε τεχνολογίες Βαθιάς Μάθησης.

Πιο συγκεκριμένα, η εφαρμογή επιτρέπει στον χρήστη να προσθέσει φωτογραφίες που βρίσκονται αποθηκευμένες στη συσκευή του ή νέες φωτογραφίες από τη κάμερα της συσκευής. Στη συνέχεια οι φωτογραφίες κατηγοριοποιούνται με βάση το περιεχόμενό τους και ο χρήστης έχει τη δυνατότητα να δει όλες τις φωτογραφίες που έχει προσθέσει στην εφαρμογή, την κατηγορία στην οποία έχουν ενταχθεί και τη βεβαιότητα με την οποία έχει γίνει η κατηγοριοποίηση, καθώς και να κάνει αναζήτηση φωτογραφιών με βάση το περιεχόμενο.

Η κατηγοριοποίηση γίνεται με χρήση προεκπαιδευμένων βαθιών νευρωνικών δικτύων, τα οποία ενσωματώνονται στην εφαρμογή μέσω της πλατφόρμας TensorFlow Lite. Η βασική διαφορά σε σχέση με άλλες αντίστοιχες εφαρμογές είναι ότι η διαδικασία κατηγοριοποίησης γίνεται τοπικά, χρησιμοποιώντας αποκλειστικά επεξεργαστικούς πόρους της ίδιας της συσκευής, και όχι μέσω κάποιας υπηρεσίας «cloud». Με αυτόν τον τρόπο αποφεύγονται οι προβληματισμοί σχετικά με την ιδιωτικότητα που προκύπτουν κάθε φορά που τα δεδομένα του χρήστη χρειάζεται να μεταφερθούν μέσω διαδικτύου, καθώς δεν υπάρχει η απαίτηση για σύνδεση στο διαδίκτυο. Αυτή είναι μια δυνατότητα που μας δίνεται χάρη στην τεράστια αύξηση της επεξεργαστικής ισχύος των κινητών συσκευών τα τελευταία χρόνια.

Το βασικό μέρος της εργασίας μετά την ανάπτυξη της εφαρμογής ήταν η αναζήτηση και χρήση διαφορετικών αρχιτεκτονικών δικτύων για την κατηγοριοποίηση, και η συγκριτική αξιολόγηση τους ως προς τον χρόνο εκτέλεσης, τη χρήση μνήμης και τη χρήση επεξεργαστικής ισχύος, λαμβάνοντας πάντα υπόψιν και τις διαφορές στα επίπεδα ακρίβειας κάθε αρχιτεκτονικής.

Λέξεις Κλειδιά

Κινητές Συσκευές, Android, Βαθιά Μάθηση, TensorFlow Lite, Κατηγοριοποίηση, Αναγνώριση Εικόνας, Εφαρμογή Gallery

Abstract

In this diploma thesis, an image storing and labeling application, based on Deep Learning technologies, was developed for mobile devices running Android.

More specifically, the application allows the user to add pictures stored in their device or new pictures taken using the device's camera. These pictures are then classified based on their content and the user can see all of their pictures, their label and the level of confidence of the classification, and search pictures based on their label.

As mentioned above, the classification is done using pretrained deep neural networks, which are integrated into the app using the TensorFlow Lite platform. The main difference compared to other similar applications is that the whole labelling process happens locally, using solely the device's computational resources, and not by using some kind of remote «cloud» service. This helps avoid any privacy concerns that exist any time user data has to be transferred over the internet, while avoiding the need for an internet connection altogether. On-device labeling is possible thanks to the enormous increase in mobile devices' processing power over the past few years.

The main objective of the thesis after developing the application was to find and use different Image Classification models and assess them in terms of latency, memory usage and CPU usage, while taking into consideration the difference in the accuracy levels of each architecture.

Keywords

Mobile Devices, Android, Deep Learning, TensorFlow Lite, Classification, Image Recognition, Gallery Application

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον κ. Ιάκωβο Βενιέρη, καθηγητή της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, για την επίβλεψη αυτής της διπλωματικής εργασίας 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. Αθανάσιο Παναγόπουλο για τη συμμετοχή τους στην τριμελή εξεταστική επιτροπή.

Επίσης ευχαριστώ τον Ιωάννη Πανόπουλο, ΥΔ ΕΜΠ και μέλος της ερευνητικής ομάδας του Εργαστηρίου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης για την συνεργασία μας και τη βοήθεια που μου παρείχε σε κάθε φάση ανάπτυξης της εφαρμογής και συγγραφής της παρούσας διπλωματικής εργασίας.

Φυσικά δε θα μπορούσα να μην ευχαριστήσω την οικογένεια μου και τους φίλους μου για τη στήριξη τους όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να αφιερώσω την παρούσα διπλωματική εργασία στη μνήμη του θείου και καθηγητή μου Παύλου Καραγιώργου που έφυγε πρόσφατα από τη ζωή.

Αθήνα, Νοέμβριος 2021
Γεώργιος Σ. Καραγιώργος

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1. Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται συνοπτικά το θεωρητικό υπόβαθρο που χρειάζεται κάποιος για να κατανοήσει την εργασία και αφορά κυρίως τη Μηχανική Μάθηση.

1.1. Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning) είναι κλάδος της Τεχνητής Νοημοσύνης και της επιστήμης των υπολογιστών που βασίζεται στη χρήση δεδομένων και αλγορίθμων για τη μίμηση του τρόπου που μαθαίνουν οι άνθρωποι [1]. Η βασική ιδέα της Μηχανικής Μάθησης είναι η χρήση μεθόδων στατιστικής και βελτιστοποίησης για την ανάλυση συνόλων δεδομένων και την εξαγωγή συμπερασμάτων [2].

Ανάλογα με τα διαθέσιμα δεδομένα οι μέθοδοι της Μηχανικής Μάθησης χωρίζονται στις παρακάτω κατηγορίες:

- Επιβλεπόμενη Μάθηση (Supervised Learning)

Ένα μοντέλο Επιβλεπόμενης Μάθησης εκπαιδεύεται με δεδομένα για τα οποία η σωστή έξοδος είναι γνωστή και παράγει μια συνάρτηση που μπορεί να παράγει έξοδο και για είσοδο την οποία δεν έχει συναντήσει ξανά. Τα 3 βασικά της συστατικά είναι η διαδικασία απόφασης, που εφαρμόζεται στην είσοδο και παράγει την έξοδο, η συνάρτηση σφάλματος με την οποία ποσοτικοποιεί τα λάθη και η διαδικασία βελτιστοποίησης, η οποία αξιολογεί τα αποτελέσματα σε περίπτωση εσφαλμένου αποτελέσματος και προσαρμόζει τη διαδικασία εξαγωγής του αποτελέσματος ώστε την επόμενη φορά να μικρύνει η απόσταση από τη σωστή απάντηση. Σε αυτή τη μέθοδο βασίστηκε και η παρούσα εργασία. Όλα τα μοντέλα έχουν εκπαιδευτεί σε σύνολα δεδομένων για τα οποία γνωρίζουμε τη σωστή κατηγοριοποίηση ώστε να μπορούν να κατηγοριοποιήσουν στη συνέχεια άγνωστες εικόνες.

- Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

Σε αντίθεση με την Επιβλεπόμενη Μάθηση, η Μη Επιβλεπόμενη Μάθηση χρησιμοποιείται όταν δεν υπάρχουν διαθέσιμα σύνολα δεδομένων με γνωστά αποτελέσματα. Προσπαθεί να αναλύσει τα δεδομένα και να εξάγει συμπεράσματα και κρυφές συσχετίσεις. Συχνά χρησιμοποιείται για ομαδοποίηση δεδομένων και για δημιουργία μηχανών προτάσεων (recommendation engines), που μπορούν να συγκρίνουν τις προτιμήσεις διαφορετικών καταναλωτών για να προβλέψουν άλλα προϊόντα που θα ενδιέφεραν κάθε χρήστη.

- Ημι-επιβλεπόμενη Μάθηση (Semisupervised Learning)

Πρόκειται για συνδυασμό των 2 προηγούμενων κατηγοριών. Χρησιμοποιείται ένα μεγάλο σύνολο δεδομένων χωρίς γνωστά αποτελέσματα και ένα μικρό σύνολο δεδομένων με γνωστά αποτελέσματα που βοηθάει στην επιτάχυνση της

εκπαίδευσης και τη βελτίωση της ακρίβειας. Προτιμάται όταν είναι εύκολη η εύρεση δεδομένων αλλά δύσκολη και χρονοβόρα η χειροκίνητη ταξινόμησή τους για να δημιουργηθεί ένα επαρκώς μεγάλο σύνολο εκπαίδευσης για Επιβλεπόμενη Μάθηση.

- Ενισχυτική Μάθηση (Reinforcement Learning)

Σε αυτή την περίπτωση η εκπαίδευση έχει ως στόχο τη δημιουργία ενός «πράκτορα» που παίρνει τις κατάλληλες αποφάσεις σε κάθε κατάσταση. Η εκπαίδευση γίνεται μέσω της χρήσης επιβραβεύσεων και ποινών. Ανάλογα με το αποτέλεσμα στο οποίο οδήγησε ένα σύνολο αποφάσεων προσαρμόζεται η διαδικασία που οδήγησε σε αυτές ώστε να οδηγούν στο βέλτιστο δυνατό αποτέλεσμα. Παράδειγμα εφαρμογής αποτελεί η Τεχνητή Νοημοσύνη που συναντάται σε ηλεκτρονικά παιχνίδια και μπορεί να προκύψει μετά από εκατομμύρια επαναλήψεις του παιχνιδιού, αξιοποιώντας την έκβαση του παιχνιδιού για να ενισχυθεί ή να περιοριστεί η τάση της μηχανής να λάβει συγκεκριμένες αποφάσεις.

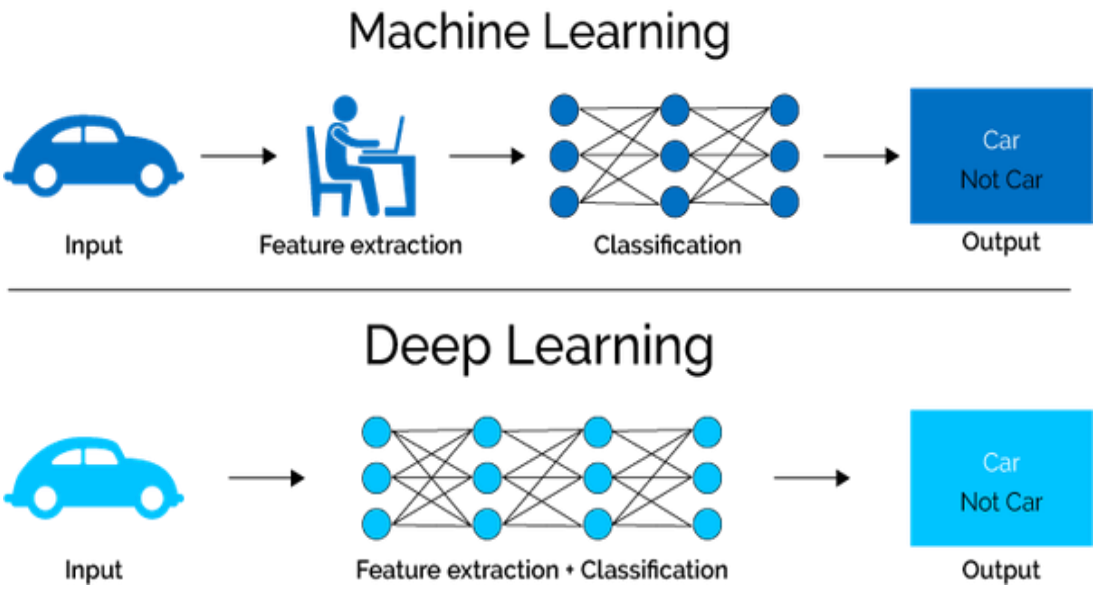
1.2. Βαθιά Μάθηση

Η Βαθιά Μάθηση (Deep Learning) είναι υποσύνολο της Μηχανικής Μάθησης. Το βασικό πλεονέκτημα είναι ότι οι αλγόριθμοι Βαθιάς Μάθησης μπορούν να δεχτούν σαν είσοδο ακατέργαστα δεδομένα (π.χ. κείμενο ή εικόνες), χωρίς να έχει προηγηθεί χειροκίνητη διαδικασία εξαγωγής χαρακτηριστικών που απαιτείται σε άλλες μεθόδους. Εάν για παράδειγμα θέλαμε να χρησιμοποιήσουμε μεθόδους Μηχανικής Μάθησης για να αναγνωρίσουμε το είδος οχημάτων, θα έπρεπε να τροφοδοτήσουμε τον αλγόριθμο μας με δεδομένα όπως το βάρος του οχήματος, τον αριθμό των τροχών και ούτω καθεξής, ενώ αντιθέτως με χρήση Βαθιάς Μάθησης θα μπορούσαμε να εκπαιδεύσουμε έναν αλγόριθμο που θα έπαιρνε σαν είσοδο ακατέργαστες φωτογραφίες οχημάτων.

Η Βαθιά Μάθηση υλοποιείται με βαθιά νευρωνικά δίκτυα, τα οποία προσπαθούν να μιμηθούν τον τρόπο λειτουργίας του ανθρωπίνου εγκεφάλου. Αυτά αποτελούνται από μια σειρά επιπέδων (layers), που καθένα με τη σειρά του απαρτίζεται από ένα σύνολο κόμβων. Ανάμεσα σε κόμβους διαδοχικών επιπέδων υπάρχουν συνδέσεις που χαρακτηρίζονται από κάποιο βάρος, μέσω των οποίων μεταφέρονται δεδομένα. Το πρώτο και το τελευταίο επίπεδο λέγονται επίπεδα εισόδου και εξόδου αντίστοιχα και τα δυο μαζί λέγονται ορατά επίπεδα. Ενδιάμεσα υπάρχουν αρκετά «κρυφά επίπεδα» (τουλάχιστον τρία, συνήθως αρκετά περισσότερα, εξού και βαθιά νευρωνικά δίκτυα). Το επίπεδο εισόδου είναι αυτό στο οποίο εισάγεται με κάποιο τρόπο η αναπαράσταση των δεδομένων στα οποία θέλουμε να επιδράσει το νευρωνικό δίκτυο, ενώ το επίπεδο εξόδου είναι αυτό στο οποίο γίνεται η τελική πρόβλεψη ή ταξινόμηση.

Κατά τη διαδικασία συμπερασματολογίας του νευρωνικού δικτύου για κάποια είσοδο η διάδοση των δεδομένων γίνεται από κάθε επίπεδο προς τα επόμενα του (διάδοση προς τα εμπρός, forward propagation). Κατά την εκπαίδευση του δικτύου γίνεται και διάδοση προς τα πίσω (backward propagation) με αλγόριθμους όπως ο αλγόριθμος απότομης καθόδου (gradient descent), που έχει ως στόχο σε περίπτωση λάθους αποτελέσματος να διορθώσει τα σφάλματα προσαρμόζοντας τα βάρη των συνδέσεων ανάλογα με το πόσο

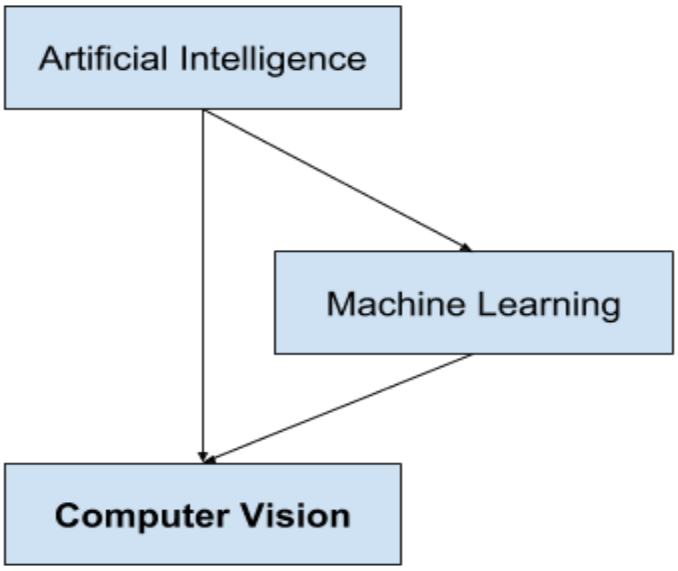
μεγάλη ήταν η απόκλιση από το σωστό αποτέλεσμα και πόσο έχει συνεισφέρει κάθε σύνδεση σε αυτό.



Εικόνα 1: Διαφορά μεταξύ Μηχανικής και Βαθιάς Μάθησης

1.3. Όραση Υπολογιστών

Η Όραση Υπολογιστών (Computer Vision) είναι τομέας της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης που επιτρέπει στους υπολογιστές να εξάγουν πληροφορίες και συμπεράσματα από εικόνες, βίντεο και άλλες οπτικές εισόδους [3].



Εικόνα 2: Σχέση Όρασης Υπολογιστών με Τεχνητή Νοημοσύνη και Μηχανική Μάθηση

Οι πρώτες προσπάθειες ανάπτυξης τρόπων ώστε οι μηχανές να καταλαβαίνουν οπτικά δεδομένα ξεκίνησαν ήδη από το 1959, όταν οι νευροφυσιολόγοι David Huber και Torsten Wiesel τοποθέτησαν ηλεκτρόδια στον εγκέφαλο μιας γάτας ώστε να παρατηρήσουν τις αντιδράσεις των νευρώνων της ενώ της έδειχναν μια σειρά από εικόνες και από τις αντιδράσεις του μυαλού της κατάλαβαν ότι η επεξεργασία εικόνων βασίζεται σε απλά σχήματα όπως είναι ευθείες γραμμές [4]. Την ίδια χρονιά ο Russell Kirsch και οι συνάδελφοί του δημιούργησαν τον πρώτο σαρωτή, μια συσκευή που επέτρεπε την μετατροπή εικόνων σε ένα πλέγμα από αριθμούς, δηλαδή μια μορφή που μπορούσαν να καταλάβουν οι υπολογιστές. Η επόμενη σημαντική πρόοδος ήρθε το 1963, όταν ο Lawrence Roberts στα πλαίσια της διδακτορικής του διατριβής με τίτλο «Μηχανική αντίληψη τρισδιάστατων στερεών» («Machine Perception of three-dimensional solids») ανέπτυξε μεθόδους εξαγωγής πληροφοριών για τρισδιάστατα στερεά από φωτογραφίες.

Το 1982 ο Βρετανός νευροεπιστήμονας David Marr έκανε μια ακόμα σημαντική δημοσίευση με τίτλο «Όραση: Μια υπολογιστική έρευνα σχετικά με την ανθρώπινη αναπαράσταση και επεξεργασία οπτικών πληροφοριών» («Vision: A computational investigation into the human representation and processing of visual information»). Βασιζόμενος στις ιδέες των Hubel και Wiesel έδειξε ότι η όραση είναι ιεραρχική και εισήγαγε ένα πλαίσιο όπου αλγόριθμοι χαμηλού επιπέδου για ανίχνευση ακμών, καμπυλών, γωνιών, κοκ. χρησιμοποιούνται ως εφαλτήρια προς μια υψηλότερου επιπέδου αναπαράσταση οπτικών δεδομένων. Ωστόσο παρόλο που η δουλειά του ήταν πολύ επαναστατική για την εποχή του, ήταν πολύ αφηρημένη και δεν περιείχε πληροφορίες για τη μοντελοποίηση που θα απαιτούνταν ώστε να εφαρμοστεί στην πράξη σε ένα υπολογιστικό σύστημα. Την ίδια περίπου εποχή ο Ιάπωνας επιστήμονας Kunihiko Fusushima δημιούργησε ένα αυτό-οργανούμενο τεχνητό δίκτυο από απλά και σύνθετα κελιά/κύτταρα (cells) που μπορούσε να αναγνωρίζει μοτίβα χωρίς να επηρεάζεται από μετακινήσεις. Το δίκτυο αυτό ονομάστηκε Neocognitron και περιλάμβανε πολλαπλά συνελικτικά επίπεδα των οποίων τα (συνήθως ορθογώνια) δεκτικά επίπεδα είχαν διανύσματα βάρους (γνωστά ως φίλτρα). Το Neocognitron θεωρείται το πρώτο νευρωνικό δίκτυο που αξίζει τον χαρακτηρισμό «βαθύ» και είναι κατά μια έννοια ο παππούς των σύγχρονων νευρωνικών δικτύων. Λίγα χρόνια αργότερα και συγκεκριμένα το 1989 ο νεαρός Γάλλος επιστήμονας Yann LeCun εφάρμοσε έναν τρόπο μάθησης βασισμένο στη διάδοση προς τα πίσω στην αρχιτεκτονική του Fukushima και μετά από λίγα χρόνια δουλειάς δημιούργησε το LeNet-5, το πρώτο μοντέρνο συνελικτικό δίκτυο που εισήγαγε ορισμένα από τα βασικά συστατικά των συνελικτικών δικτύων που χρησιμοποιούμε ως σήμερα. Εφάρμοσε την εφεύρεσή του στην αναγνώριση χαρακτήρων και αποτέλεσμα ήταν η δημιουργία του συνόλου δεδομένων από χειρόγραφα ψηφία MNIST.

Στο τέλος της δεκαετίας του 1990 έγινε μια στροφή στον τομέα της Όρασης Υπολογιστών και οι επιστήμονες σταμάτησαν να προσπαθούν να επαναδημιουργήσουν μοντέλα φτιάχνοντας τρισδιάστατα μοντέλα και αντίθετα κατεύθυναν τις προσπάθειες τους προς την αναγνώριση αντικειμένων βασισμένη σε χαρακτηριστικά. Το 2001 δημιουργήθηκε από τους Paul Viola και Michael Jones ο πρώτος αλγόριθμος αναγνώρισης προσώπων που λειτουργούσε σε πραγματικό χρόνο, ο οποίος αν και δεν βασίζεται στη Βαθιά Μάθηση χρησιμοποιεί κάποιες ιδέες της, αφού κατά τη λειτουργία του μαθαίνει ποια

χαρακτηριστικά είναι σημαντικά. Το 2006 ξεκίνησε το έργο Pascal VOC που παρείχε ένα προτυποποιημένο σύνολο δεδομένων για αναγνώριση αντικειμένων και για τα επόμενα 6 χρόνια υπήρχε ένας ετήσιος διαγωνισμός για την αξιολόγηση διαφορετικών μεθόδων αναγνώρισης αντικειμένων. Το 2010 ξεκίνησε ο διαγωνισμός ILSVRC («ImageNet Large Scale Visual Recognition Competition») που αύξησε το πλήθος των κατηγοριών για τα αντικείμενα από τις 20 του Pascal VOC σε πάνω από 1000. Το 2012 πήρε μέρος στο διαγωνισμό μια ομάδα από το Πανεπιστήμιο του Τορόντο με το μοντέλο συνελικτικού νευρωνικού δικτύου AlexNet που πέτυχε ποσοστό λάθους πολύ μικρότερο από οποιοδήποτε προηγούμενο, και από τότε και στο εξής μέχρι και την τελευταία χρονιά του διαγωνισμού το 2017 οι νικητές ήταν κάθε χρόνο συνελικτικά νευρωνικά δίκτυα.

1.4. Συνελικτικά Νευρωνικά Δίκτυα

Τα παραδοσιακά νευρωνικά δίκτυα δέχονται μια εικόνα σαν είσοδο και την μεταμορφώνουν μέσω μιας ακολουθίας από κρυφά επίπεδα, χρησιμοποιώντας συχνά μη γραμμικές συναρτήσεις ενεργοποίησης [5]. Κάθε επίπεδο αποτελείται από ένα σύνολο νευρώνων και κάθε νευρώνας είναι πλήρως συνδεδεμένος με κάθε νευρώνα του προηγούμενου επιπέδου. Είναι εμφανές ότι όταν έχουμε να κάνουμε με εικόνες το πλήθος των νευρώνων που χρειάζονται είναι πολύ μεγάλο, γεγονός που κάνει τη χρήση νευρωνικών δικτύων για εφαρμογές σχετικές με εικόνες πολύ δύσκολη. Εάν για παράδειγμα είχαμε μια εικόνα ανάλυσης 250×250 εικονοστοιχείων (με 3 κανάλια για κάθε pixel, ένα για καθένα από τα 3 βασικά χρώματα) τότε μόνο για το επίπεδο εισόδου θα χρειαζόμασταν 250×250×3 = 187.500 νευρώνες.

Για να αντιμετωπίσουμε το παραπάνω πρόβλημα χρησιμοποιούμε τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs ή ConvNets). Τα CNNs 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. Σε ένα CNN τα επίπεδα είναι διατεταγμένα σε έναν τρισδιάστατο όγκο όπου οι τρεις διαστάσεις είναι πλάτος, ύψος και βάθος (αναφέρεται στην τρίτη διάσταση, για παράδειγμα στο πλήθος καναλιών μιας εικόνας ή το πλήθος των φίλτρων σε ένα επίπεδο). Τα δίκτυα αυτά αξιοποιούν την τοπικότητα της πληροφορίας και περιορίζουν το πλήθος των συνδέσεων, καθώς κάθε νευρώνας συνδέεται μόνο με τους νευρώνες του προηγούμενου επιπέδου που αντιστοιχούν σε μια μικρή περιοχή (τοπική συνδεσιμότητα).

Τα CNNs παίρνουν το όνομά τους από το γεγονός ότι ένα ή περισσότερα από τα επίπεδά τους βασίζονται στην πράξη της συνέλιξης. Η συνέλιξη μιας δισδιάστατης εικόνας I με έναν επίσης δυσδιάστατο πίνακα K συμβολίζεται με $I * K$ και ορίζεται μαθηματικά από τον ακόλουθο τύπο:

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

Συνήθως η εικόνα I αναφέρεται ως η είσοδος (input) της συνέλιξης και ο πίνακας K ως ο πυρήνας (kernel) ή το φίλτρο (filter) αυτής. Στις επόμενες παραγράφους θα αναλυθούν οι βασικοί τύποι επιπέδων των CNNs [6].

Συνελικτικό Επίπεδο

Το συνελικτικό επίπεδο (convolutional layer) είναι το βασικό δομικό στοιχείο ενός συνελικτικού νευρωνικού δικτύου. Οι παράμετροί του αποτελούνται από ένα σύνολο φίλτρων που το δίκτυο μαθαίνει κατά την εκπαίδευσή του. Τα φίλτρα αυτά είναι μικρά σε σχέση με την εικόνα εισόδου. Η εφαρμογή κάθε φίλτρου γίνεται συνελλίσοντάς το με την εικόνα εισόδου και σαν αποτέλεσμα προκύπτει ένας χάρτης ενεργοποίησης (activation map) ή χάρτης χαρακτηριστικών (feature map), που περιέχει είτε χαμηλού είτε υψηλού επιπέδου χαρακτηριστικά. Ο κάθε νευρώνας εξόδου συνδέεται με μια μικρή περιοχή της εικόνας εισόδου που ονομάζεται τοπικό δεκτικό πεδίο (local receptive field) του νευρώνα.

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. Μεγαλύτερες τιμές βηματισμού έχουν ως αποτέλεσμα μικρότερες διαστάσεις στην έξοδο.

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

Οι παραπάνω παράμετροι δε μαθαίνονται κατά την εκπαίδευση αλλά επιλέγονται από τον σχεδιαστή του δικτύου και γι' αυτό το λόγο ονομάζονται συνήθως υπερπαράμετροι.

Ένας τρόπος να περιοριστεί το πλήθος των βαρών των νευρώνων σε ένα συνελικτικό επίπεδο και συνεπώς οι απαιτήσεις μνήμης του δικτύου είναι το λεγόμενο σχήμα διαμοιρασμού των παραμέτρων (parameter sharing). Σύμφωνα με αυτό οι παράμετροι κάθε φίλτρου είναι οι ίδιες για τον υπολογισμό οποιουδήποτε από τους νευρώνες εξόδου. Το σχήμα αυτό βασίζεται στην υπόθεση ότι αν ένα χαρακτηριστικό βρεθεί σε μια συγκεκριμένη θέση, τότε είναι χρήσιμο το ίδιο χαρακτηριστικό να αναζητηθεί και σε όλες τις υπόλοιπες θέσεις.

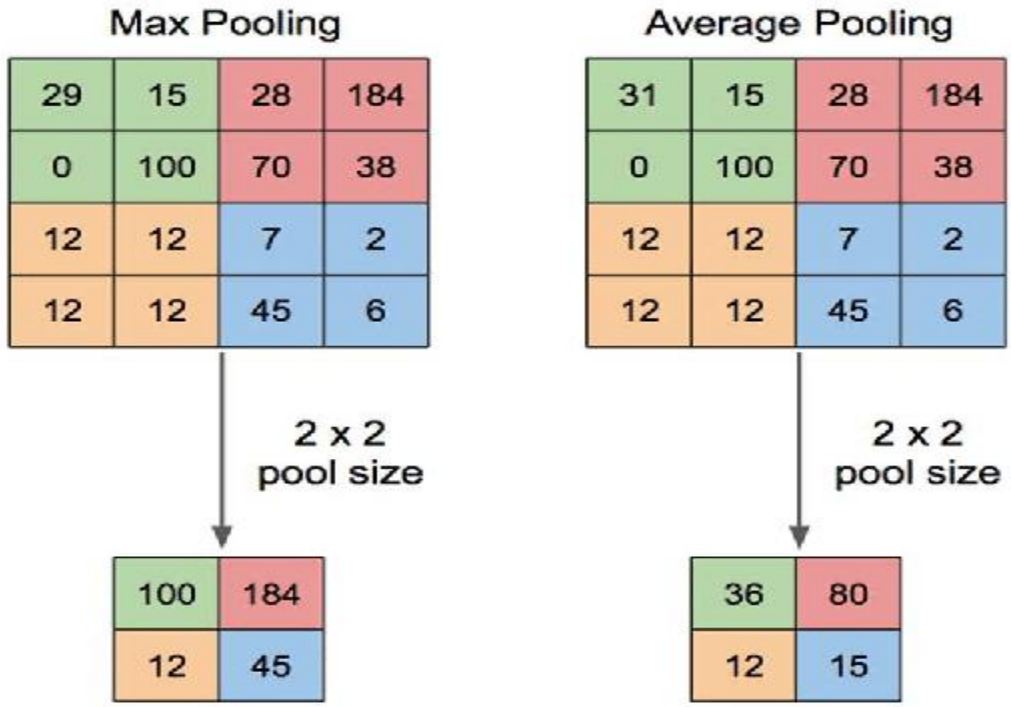
Επίπεδο Συγκέντρωσης

Τα επίπεδα συγκέντρωσης (pooling layers) χρησιμοποιούνται για να μειώσουν το μέγεθος του πίνακα εισόδου τους. Συχνά τοποθετούνται ανάμεσα σε διαδοχικά επίπεδα συνέλιξης ώστε να μειώσουν το πλήθος των παραμέτρων και συνεπώς την πολυπλοκότητα του δικτύου. Αυτό το κάνουν αντικαθιστώντας μια μικρή περιοχή νευρώνων με μια στατιστική τιμή που υπολογίζεται εφαρμόζοντας

μια συνάρτηση στις τιμές των νευρώνων της περιοχής. Δημοφιλείς συναρτήσεις είναι:

- Η μέγιστη συγκέντρωση (max pooling), που επιστρέφει τη μέγιστη τιμή από μια (συνήθως τετραγωνική) γειτονιά.
- Η μέση συγκέντρωση (average pooling), που επιστρέφει τον μέσο όρο των τιμών μιας γειτονιάς.
- Η συγκέντρωση L^2 (L^2 -norm pooling), που επιστρέφει την τετραγωνική ρίζα του αθροίσματος των τετραγώνων των τιμών μιας γειτονιάς.

Συνήθως γίνεται ομαδοποίηση περιοχών 2×2, ωστόσο όταν το μέγεθος εισόδου είναι πολύ μεγάλο μπορεί να επιλεχθεί και 3×3. Ο βηματισμός είναι συνήθως 1 pixel και πιο σπάνια 2.



Εικόνα 3: Παράδειγμα Max και Average Pooling [7]

Συναρτήσεις Ενεργοποίησης

Μετά από κάθε συνελικτικό επίπεδο εφαρμόζεται συνήθως μια μη γραμμική συνάρτηση ενεργοποίησης (activation function). Οι συναρτήσεις αυτές είναι μη γραμμικές επειδή είναι αδύνατο σε ένα πρόβλημα κατηγοριοποίησης οι κλάσεις να εξαρτώνται γραμμικά από τα δεδομένα. Ορισμένες συχνά χρησιμοποιούμενες τέτοιες συναρτήσεις είναι οι εξής:

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

Η συνάρτηση ενεργοποίησης εφαρμόζεται σε κάθε στοιχείο του πίνακα εισόδου της και άρα η έξοδος της έχει ίδια διάσταση με την είσοδο.

Πλήρως Συνδεδεμένο Επίπεδο

Τα πλήρως συνδεδεμένα επίπεδα (fully connected layers) βρίσκονται μόνο στο τέλος του δικτύου, και συνήθως είναι ένα ή δυο, ακολουθούμενα από έναν ταξινομητή Softmax. Πρόκειται για επίπεδα των οποίων οι νευρώνες είναι συνδεδεμένοι με όλους τους νευρώνες του προηγούμενου επιπέδου. Το πρώτο από τα δυο μετασχηματίζει τα χαρακτηριστικά που έχουν εντοπιστεί σε ένα ενιαίο διάνυσμα, ενώ το τελευταίο έχει μήκος ίσο με το πλήθος των κλάσεων του προβλήματος και εξάγει το επίπεδο βεβαιότητας για κάθε κλάση.

Είναι ενδιαφέρον ότι παρόλο που όπως αναφέρθηκε προηγουμένως τα CNNs υπάρχουν ουσιαστικά εδώ και περίπου 4 δεκαετίες έγιναν πολύ δημοφιλή μόλις τα τελευταία χρόνια. Οι βασικοί λόγοι είναι η μεγάλη αύξηση της διαθέσιμης υπολογιστικής ισχύος των υπολογιστών και η ύπαρξη πλέον μεγάλων συνόλων δεδομένων που μπορούν να αξιοποιηθούν για την εκπαίδευση των δικτύων.

1.5. Γνωστές Αρχιτεκτονικές

Όπως αναφέρθηκε προηγουμένως, αρχιτεκτονικές που μοιάζουν στα σύγχρονα CNNs πρωτοεμφανίστηκαν τη δεκαετία του 1980, ωστόσο οι πρώτες αρχιτεκτονικές που μπορούν να θεωρηθούν σύγχρονα CNNs εμφανίστηκαν στο τέλος της δεκαετίας του 1990. Τα πρώτα συνελικτικά νευρωνικά δίκτυα περιλάμβαναν μερικά συνελικτικά επίπεδα με κάποια συνάρτηση ενεργοποίησης ακολουθούμενα από κάποια επίπεδα συγκέντρωσης για να μικρύνει το μέγεθος της εικόνας. Στη συνέχεια υπήρχαν κάποια πλήρως συνδεδεμένα επίπεδα που κατέληγαν σε αυτό που έδινε τις προβλέψεις για κάθε κλάση. Στην πορεία όμως άρχισαν να εισάγονται νέα δομικά συστατικά, όπως τα residual blocks [8] και τα depthwise convolution blocks [9] με αποτέλεσμα οι αρχιτεκτονικές να γίνονται όλο και πιο περίπλοκες. Μερικές αρχιτεκτονικές ορόσημα για την εξέλιξη των συνελικτικών δικτύων είναι οι παρακάτω:

- LeNet (1998, Yann Lecun et al.) [10]
- AlexNet (2012, Alex Krizhevsky) [11]
- GoogLeNet (2014, Christian Szegedy et al.) [12]
- VGG (2015, Karen Stmonyan και Andrew Zisserman) [13]
- Inception-v3 (2015, Christian Szegedy et al.) [14]
- ResNet (2015, Kaiming He et al.) [15]
- EfficientNet (2019, Mingxing Tan και Qoc V. Le) [16]

Τα περισσότερα νέα μοντέλα που εμφανίζονται βασίζονται σε συνδυασμούς δομικών στοιχείων των παραπάνω αρχιτεκτονικών. Το κορυφαίο αυτή τη στιγμή μοντέλο στο σύνολο δεδομένων ImageNet είναι το CoAtNet-7 που πετυχαίνει top-1 ακρίβεια 90.88%. Μέχρι πριν μερικά χρόνια τα δίκτυα ήταν όλο και πιο περίπλοκα, ωστόσο η εισαγωγή της Βαθιάς Μάθησης σε κινητές συσκευές υπήρξε καθοριστική για να υπάρξει μια στροφή προς πιο ελαφριά δίκτυα, όπως το

SqueezeNet και τα MobileNets, καθώς και σε τρόπους συμπίεσης των ήδη υπαρχουσών αρχιτεκτονικών.

1.6. Αντικείμενο Διπλωματικής Εργασίας

Το αντικείμενο της διπλωματικής εργασίας είναι η μελέτη και αξιολόγηση διαφορετικών μοντέλων Βαθιάς Μάθησης για κατηγοριοποίηση εικόνων σε κινητές συσκευές. Η περίπτωση που μελετάται είναι της εκτέλεσης της συμπερασματολογίας τοπικά στη συσκευή και όχι μέσω της επικοινωνίας με κάποιο εξωτερικό σύστημα.

Στα πλαίσια της παραπάνω μελέτης αναπτύχθηκε μια εφαρμογή για κινητές συσκευές με λειτουργικό σύστημα Android με χρήση της γλώσσας Java. Η επιλογή του Android έγινε λόγω του μεγάλου μεριδίου αγοράς (πάνω από 70% παγκοσμίως [17]) αλλά και λόγω της ευκολίας ανάπτυξης εφαρμογών για αυτό χωρίς να είναι απαραίτητη η κατοχή συγκεκριμένου τύπου υπολογιστών όπως συμβαίνει για άλλα λειτουργικά συστήματα.

Ουσιαστικά μέσω της εφαρμογής αξιολογήθηκε η απόδοση των διαφορετικών μοντέλων σε δυο διαφορετικές συσκευές και με διαφορετικές κάθε φορά παραμέτρους (επεξεργαστής εκτέλεσης, πλήθος επεξεργαστικών νημάτων, πλήθος εικόνων εισόδου, κοκ).

2. Τεχνολογίες

2.1. Android

Το Android είναι ένα λειτουργικό σύστημα για κινητές συσκευές, η πρώτη έκδοση του οποίου κυκλοφόρησε το 2008 [18]. Αναπτύσσεται από την εταιρία Google, η οποία εξαγόρασε το 2005 την εταιρία Android Inc που είχε ξεκινήσει την ανάπτυξή του ως λογισμικό για χρήση σε κάμερες. Πρόκειται για λογισμικό ανοιχτού κώδικα, που είναι διαθέσιμο ακόμα και για εμπορική χρήση από τον οποιονδήποτε, ενώ η βάση του είναι μια τροποποιημένη έκδοση του πυρήνα του Linux.

Είναι το δημοφιλέστερο λειτουργικό σύστημα του κόσμου, με το πλήθος των συσκευών που το χρησιμοποιούν να υπολογίζεται περίπου στα 2.5 δισεκατομμύρια. Σε αυτές συμπεριλαμβάνονται κυρίως smartphones (έξυπνα τηλέφωνα) και tablets, αλλά υπάρχουν εκδόσεις του Android μεταξύ άλλων για smart TVs (έξυπνες τηλεοράσεις) και smartwatches (έξυπνα ρολόγια). Υπάρχουν πάνω από 3 εκατομμύρια εφαρμογές για Android, πολλές από τις οποίες είναι διαθέσιμες μέσω του επίσημου Google Play Store, ωστόσο ο κάθε χρήστης μπορεί να εγκαταστήσει εφαρμογές που έχει κατεβάσει από το διαδίκτυο.

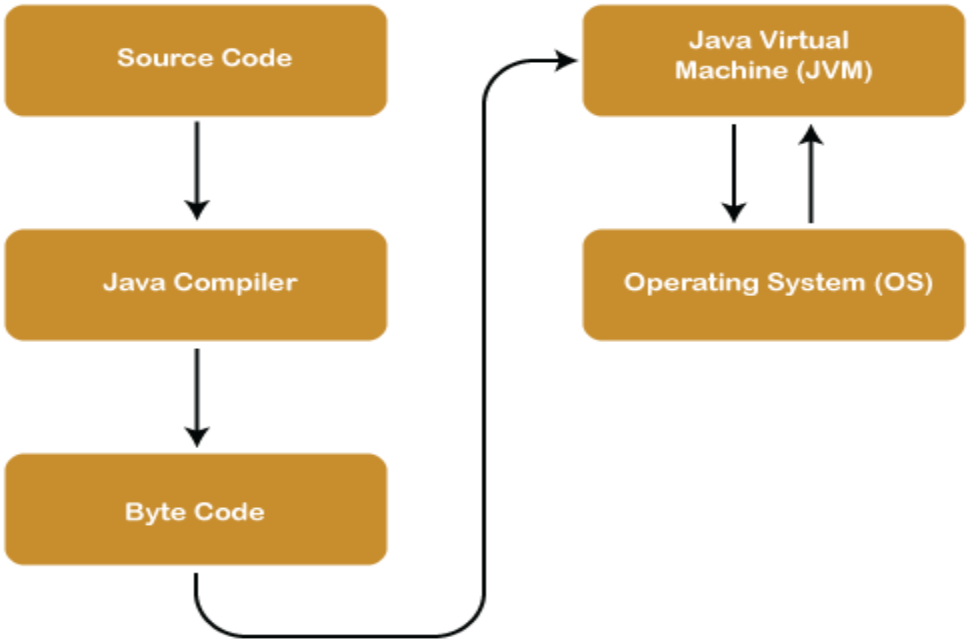
Νέες εκδόσεις του Android κυκλοφορούν κάθε χρόνο. Η πιο πρόσφατη είναι η έκδοση Android 11, ενώ σύντομα αναμένεται να κυκλοφορήσει η έκδοση Android 12. Στο παρελθόν η Google έδινε στις εκδόσεις ονόματα από γλυκά (4.0 Ice Cream Sandwich, 4.1 Jelly Bean, 5.0 Lollipop, 6.0 Marshmallow, 8.0 Oreo και άλλα). Ωστόσο από την έκδοση 10 και μετά αυτό σταμάτησε και οι νέες εκδόσεις χαρακτηρίζονται πλέον μόνο από τον αριθμό τους.

2.2. Java

Η Java είναι μια μοντέρνα αντικειμενοστρεφής γλώσσα προγραμματισμού. Η ανάπτυξη της ξεκίνησε το 1991 στη Sun Microsystems ως μια γλώσσα που θα επέτρεπε σε διάφορες ηλεκτρονικές συσκευές να επικοινωνούν μεταξύ τους [19]. Επικεφαλής της ομάδας ανάπτυξης ήταν ο James Gosling και πρώτο της όνομα το «Oak». Σύντομα η ομάδα συνειδητοποίησε ότι η δυνατότητα της Java να προσδίδει διαδραστικότητα και να υποστηρίζει εφαρμογές πολυμέσων την καθιστούσε κατάλληλη για χρήση σε διαδικτυακές εφαρμογές.

Κατά τη διάρκεια της δεκαετίας του 1990 η Java επεκτάθηκε πέρα από το διαδίκτυο και χρησιμοποιήθηκε σε συσκευές όπως προσωπικοί υπολογιστές, κινητά τηλέφωνα και ακόμα και σε ερευνητικές αποστολές της NASA. Η δημοφιλία της Java οδήγησε στη δημιουργία διαφόρων εκδόσεων για οικιακούς υπολογιστές (Java SE), ενσωματωμένες συσκευές (Java ME), διαδικτυακές εφαρμογές και υπερυπολογιστές (super computers) (Java EE). Το 2010 η εταιρία Oracle εξαγόρασε τη Sun Microsystems και ανέλαβε τη διαχείριση της Java, η οποία σύμφωνα με την εταιρία έφτασε να είναι εγκατεστημένη σε τουλάχιστον 3 δισεκατομμύρια συσκευές. Ακόμα και σήμερα που δεν είναι πλέον στην περίοδο της ακμής της η Java παραμένει μια πολύ δημοφιλής γλώσσα προγραμματισμού για μεγάλα συστήματα επιχειρήσεων όπως τράπεζες, εφαρμογές για κινητά τηλέφωνα και άλλα.

Το βασικό χαρακτηριστικό της Java που της επέτρεψε να ξεχωρίσει από άλλες γλώσσες ήταν η επαναστατική ιδέα της χρήσης ενός περιβάλλοντος εκτέλεσης γνωστού ως Java Runtime Environment (JRE) ή Java Virtual machine (JVM). Αυτό επιτρέπει στον κώδικα των εφαρμογών της Java να γράφεται μια φορά και να μπορεί να εκτελεστεί σε πολλές συσκευές με διαφορετικά χαρακτηριστικά υλικού και λογισμικού. Στις περισσότερες γλώσσες ο κώδικας μεταφράζεται με τη χρήση ενός μεταγλωτιστή σε εντολές χαμηλού επιπέδου που μπορούν να εκτελεστούν σε έναν συγκεκριμένο τύπο υπολογιστή. Αντίθετα, στην περίπτωση της Java ο κώδικας μεταφράζεται στη μορφή του «Bytecode», που εκτελείται στη συνέχεια από το JRE [20]. Έτσι μπορεί ο ίδιος κώδικας να γραφτεί μια φορά και να τρέξει σε πολλές διαφορετικές πλατφόρμες, γι' αυτό και ένα από τα «συνθήματα» της Java είναι το «write once, run anywhere».



Εικόνα 4: Αρχιτεκτονική της Java

2.3. Android Studio

Το Android Studio [21] είναι το επίσημο Ενσωματωμένο Περιβάλλον Ανάπτυξης (IDE) για εφαρμογές Android. Ανακοινώθηκε από τη Google το 2013 ως αντικαταστάτης του Eclipse και η πρώτη του έκδοση κυκλοφόρησε στο τέλος του 2014. Βασίζεται στο IntelliJ IDEA της JetBrains και είναι διαθέσιμο δωρεάν. Διαθέτει λειτουργίες συγγραφής κώδικα, αποσφαλμάτωσης (debugging) και συνοδεύεται από έναν εξομοιωτή (emulator) βασισμένο στο Qemu για εκτέλεση εφαρμογών Android σε εικονικές μηχανές.

2.4. TensorFlow Lite

Το TensorFlow Lite [22] είναι ένα σύνολο εργαλείων που επιτρέπει την ενσωμάτωση εφαρμογών Βαθιάς Μάθησης σε κινητές συσκευές, ενσωματωμένα συστήματα και συσκευές Διαδικτύου των Πραγμάτων (Internet of Things).

Τα βασικά χαρακτηριστικά που υπόσχεται είναι τα εξής:

- Βελτιστοποίηση για Μηχανική Μάθηση για συμπερασματολογία στη συσκευή (on-device) λαμβάνοντας υπόψιν πέντε βασικούς περιορισμούς: την καθυστέρηση (δεν υπάρχει μεταφορά δεδομένων προς κάποιον εξυπηρετητή), την ιδιωτικότητα (τα δεδομένα του χρήστη δε φεύγουν από τη συσκευή του), τη συνδεσιμότητα (δεν είναι απαραίτητη η ύπαρξη σύνδεσης με το διαδίκτυο), το μέγεθος (τα μοντέλα και ο εκτελέσιμος κώδικας έχουν το μικρότερο δυνατό μέγεθος) και την κατανάλωση ενέργειας (η συμπερασματολογία είναι όσο πιο ενεργειακά αποδοτική γίνεται).

- Υποστήριξη πολλαπλών πλατφορμών, όπως είναι οι συσκευές Android και iOS, ενσωματωμένα συστήματα με λειτουργικό Linux ή μικροελεγκτές.

- Υποστήριξη διαφορετικών γλωσσών προγραμματισμού, μεταξύ των οποίων οι Java, Swift, Objective-C, C++ και Python.

- Υψηλή απόδοση, με επιτάχυνση υλικού και βελτιστοποίηση μοντέλων.

- Ολοκληρωμένα παραδείγματα για συχνές εφαρμογές όπως είναι η κατηγοριοποίηση εικόνων, η αναγνώριση αντικειμένων, η απάντηση ερωτήσεων κοκ σε πολλαπλές πλατφόρμες.

Οι διάφορες αρχιτεκτονικές που μπορούν να χρησιμοποιηθούν με το TensorFlow Lite ονομάζονται μοντέλα και αναπαρίστανται σε μια ειδική, φορητή μορφή γνωστή ως FlatBuffers (αρχεία με κατάληξη «.tflite»). Η μορφή αυτή προσφέρει μικρότερο μέγεθος και πιο αποδοτική εκτέλεση, ενώ μπορεί να περιλαμβάνει και μεταδεδομένα με πληροφορίες σχετικές με το μοντέλο, όπως είναι η περιγραφή του μοντέλου σε μορφή κατανοητή από τους ανθρώπους ή πληροφορίες κατανοητές από τις μηχανές που μπορούν να επιτρέψουν την αυτόματη δημιουργία κώδικα.

Ένα μοντέλο TensorFlow Lite μπορεί να βρεθεί με τους εξής τρόπους:

- Με χρήση ενός υπάρχοντος μοντέλου. Αρκετά μοντέλα φιλοξενούνται στην ιστοσελίδα του TensorFlow Lite και στο TFHub [23].

- Με δημιουργία ενός νέου μοντέλου TensorFlow Lite, χρησιμοποιώντας το εργαλείο δημιουργίας μοντέλων TensorFlow Lite Model Maker [24]. Από προεπιλογή τα μοντέλα αυτά περιέχουν μεταδεδομένα.

- Με μετατροπή ενός μοντέλου TensorFlow σε μορφή συμβατή με το TensorFlow Lite, χρησιμοποιώντας το εργαλείο TensorFlow Lite Converter. Κατά την μετατροπή μπορούν να εφαρμοστούν

βελτιστοποιήσεις όπως κβαντοποίηση για τον περιορισμού του μεγέθους του μοντέλου και του χρόνου εκτέλεσης, με ελάχιστη ή και καθόλου απώλεια ακρίβειας. Εν γένει αυτά τα μοντέλα δεν περιλαμβάνουν μεταδεδομένα.

Με όποιον τρόπο και να προκύψει ένα μοντέλο στη συνέχεια μπορεί να χρησιμοποιηθεί ο TensorFlow Lite Interpreter για την εκτέλεση του σε μια συσκευή. Η συμπερασμοτολογία μπορεί να εκτελεστεί με δυο τρόπους:

- Για μοντέλα χωρίς μεταδεδομένα, χρησιμοποιώντας απευθείας το TensorFlow Lite Interpreter, που υποστηρίζει πολλές πλατφόρμες και γλώσσες.

- Για μοντέλα με μεταδεδομένα μπορούν να χρησιμοποιηθούν βοηθητικές βιβλιοθήκες όπως η TensorFlow Lite Task Library και η TensorFlow Lite Support Library. Επίσης μπορεί να παραχθεί αυτόματα κώδικας για εφαρμογές Android χρησιμοποιώντας το Android Studio ML Model Binding ή το TensorFlow Lite Code Generator. Αυτός ο τρόπος υποστηρίζεται προς το παρόν μόνο με χρήση της γλώσσας Java και μόνο στο Android, ενώ προετοιμάζεται η υποστήριξη για τις γλώσσες Swift (iOS) και C++.

Σε συσκευές Android και iOS μπορεί να χρησιμοποιηθεί για βελτίωση της απόδοσης επιτάχυνση υλικού, καθώς και ένα σύνολο από «εκπροσώπους» (delegates), που αποστέλλουν μέρη από το δίκτυο σε κάποιον επιταχυντή όπως οι GPUs ή οι NPUs.

3. Ανάλυση Εφαρμογής

Το Smart Gallery είναι μια εφαρμογή τύπου «Gallery» για κινητές συσκευές με λειτουργικό Android όπου ο χρήστης μπορεί να προσθέσει και να δει τις φωτογραφίες του. Οι φωτογραφίες που προστίθενται κατηγοριοποιούνται με χρήση μεθόδων Βαθιάς Μάθησης σε 1000 κατηγορίες και στη συνέχεια ο χρήστης μπορεί να δει το αποτέλεσμα της ταξινόμησης, τη βεβαιότητα με την οποία έχει ενταχθεί μια φωτογραφία σε κάποια κατηγορία καθώς και να αναζητήσει φωτογραφίες με βάση τις ετικέτες τους.

3.1. Η Διεπαφή Χρήστη

Στην παρούσα ενότητα γίνεται μια σύντομη αναφορά στις βασικές οθόνες και λειτουργίες της εφαρμογής.

3.1.1. Αρχική Οθόνη

Στην αρχική οθόνη της εφαρμογής ο χρήστης βλέπει όλες τις φωτογραφίες που έχει προσθέσει με αντίστροφη χρονολογική σειρά.

Στο κάτω μέρος της οθόνης υπάρχουν δυο κουμπιά. Το πρώτο ανοίγει την εφαρμογή της Κάμερας του κινητού για λήψη φωτογραφίας η οποία προστίθεται αυτόματα στην εφαρμογή. Το δεύτερο κουμπί επιτρέπει στον χρήστη να επιλέξει για προσθήκη στην εφαρμογή μια ή περισσότερες φωτογραφίες αποθηκευμένες στη συσκευή του (ή σε κάποιον συνδεδεμένο με τη συσκευή αποθηκευτικό χώρο στο Cloud). Στην περίπτωση επιλογής μιας μόνο εικόνας αυτή εμφανίζεται άμεσα στην αρχική οθόνη, ενώ αν επιλεγούν περισσότερες τότε ο χρήστης μεταφέρεται αυτόματα σε μια βοηθητική οθόνη όπου βλέπει μικρογραφίες των εικόνων που επέλεξε έως ότου να αποθηκευτούν όλες, και τότε μπορεί να επιστρέψει στην αρχική οθόνη.

Στην αρχική οθόνη με παρατεταμένο πάτημα σε μια φωτογραφία ενεργοποιείται η διαδικασία επιλογής όπου ο χρήστης μπορεί να επιλέξει μια ή περισσότερες φωτογραφίες για διαγραφή.

3.1.2. Προβολή Μεμονωμένης Φωτογραφίας

Με απλό πάτημα σε μια φωτογραφία αυτή εμφανίζεται σε μεγαλύτερο μέγεθος. Στο πάνω μέρος της οθόνης εμφανίζονται σαν τίτλος η ημερομηνία προσθήκης της φωτογραφίας και η ετικέτα της. Δίπλα από αυτά υπάρχουν ένα κουμπί για προβολή ορισμένων πληροφοριών για τη φωτογραφία (όνομα αρχείου, ανάλυση, ημερομηνία προσθήκης, μέγεθος αρχείου) και ένα κουμπί για διαγραφή, ενώ πατώντας πάνω στην ετικέτα εμφανίζονται οι 5 ετικέτες με τις οποίες ταιριάζει περισσότερο η φωτογραφία και το αντίστοιχο επίπεδο βεβαιότητας. Επίσης ο χρήστης μπορεί να κάνει zoom στη φωτογραφία και να μετακινηθεί σε επόμενες ή προηγούμενες φωτογραφίες κάνοντας «swipe» αριστερά ή δεξιά.

3.1.3. Κοινοποίηση Φωτογραφιών

Τέλος, υπάρχει ένας 3ος τρόπος προσθήκης εικόνων στην εφαρμογή. Από οποιαδήποτε εφαρμογή της συσκευής με δυνατότητα "κοινοποίησης" φωτογραφιών υπάρχει η δυνατότητα προσθήκης μιας ή περισσότερων φωτογραφιών απευθείας στην εφαρμογή. Σε αυτή την περίπτωση εμφανίζεται η ίδια οθόνη όπως στην περίπτωση εισαγωγής πολλαπλών φωτογραφιών απευθείας μέσα από την εφαρμογή.

3.2. Αποθήκευση Φωτογραφιών

Οι φωτογραφίες που προσθέτει ο χρήστης με οποιονδήποτε τρόπο αποθηκεύονται στον ιδιωτικό αποθηκευτικό χώρο της εφαρμογής. Παράλληλα αποθηκεύονται στη βάση δεδομένων πληροφορίες όπως το όνομα του αρχείου, το μέγεθος και οι διαστάσεις της εικόνας.

3.3. Κατηγοριοποίηση Φωτογραφιών

Κάθε φωτογραφία που προστίθεται στην εφαρμογή δίνεται σαν είσοδος σε ένα μοντέλο ταξινόμησης εικόνων ώστε να αποκτήσει μια ετικέτα. Η διαδικασία αυτή δε γίνεται τη στιγμή που προστίθεται η εικόνα, αλλά προγραμματίζεται να γίνει σε δεύτερο χρόνο χωρίς να επηρεάζει τη λειτουργία της εφαρμογής.

4. Ανάπτυξη Εφαρμογής

Σε αυτό το κεφάλαιο αναλύεται σε μεγαλύτερο βάθος η δομή της εφαρμογής και παρουσιάζονται συνοπτικά οι βιβλιοθήκες που χρησιμοποιήθηκαν για να διευκολύνουν την ανάπτυξή της.

4.1. Δομή Εφαρμογής

Η εφαρμογή αναπτύχθηκε με χρήση της γλώσσας Java που είναι μια από τις προτεινόμενος γλώσσες για προγραμματισμό εφαρμογών για Android (μαζί εδώ και λίγα χρόνια με την Kotlin). Για την οργάνωση του κώδικα έχει γίνει εκτεταμένη χρήση της δυνατότητας της Java για πακέτα (packages). Στη συνέχεια αναλύονται τα βασικότερα πακέτα και ορισμένες σημαντικές κλάσεις.

- Το πακέτο ui. Εδώ βρίσκονται οι κλάσεις που αφορούν το γραφικό περιβάλλον της εφαρμογής. Περιλαμβάνει κάποια υπο-πακέτα, όπως είναι τα εξής:
    - Home. Περιλαμβάνει τον κώδικα που είναι υπεύθυνος για την εμφάνιση της αρχικής οθόνης της εφαρμογής και της λίστας με όλες τις φωτογραφίες.
    - Pictures. Περιλαμβάνει τον κώδικα που αφορά την προβολή μιας μεμονωμένης φωτογραφίας και των λεπτομερειών που την αφορούν.
    - Share. Εδώ βρίσκεται η κλάση ShareActivity, που εκτελείται όταν ο χρήστης επιλέξει να κοινοποιήσει κάποια ή κάποιες φωτογραφίες από τη συλλογή φωτογραφιών του στην εφαρμογή και είναι υπεύθυνη για την αποθήκευσή τους και τον προγραμματισμό της κατηγοριοποίησής τους.
    - Benchmarks. Το πακέτο αυτό περιλαμβάνει κώδικα για μια οθόνη που επιτρέπει την επιλογή παραμέτρων για εκτέλεση του μοντέλου και δημιουργήθηκε για την υποστήριξη των μετρήσεων (περισσότερα στο Κεφάλαιο 6).
- Το πακέτο ml. Είναι το πιο σημαντικό πακέτο, καθώς υλοποιεί το κομμάτι της εφαρμογής που αφορά τη Μηχανική Μάθηση. Ορισμένες κλάσεις της είναι η ClassificationCategory που αναπαριστά μια κατηγορία στην οποία μπορεί να ταξινομηθεί μια φωτογραφία, η ClassificationResult που αναπαριστά τα αποτελέσματα του inference για μια φωτογραφία και η ClassifierConfiguration που περιλαμβάνει τις παραμέτρους εκτέλεσης του μοντέλου (όπως το ποιο μοντέλο θέλουμε να χρησιμοποιηθεί, τον επεξεργαστή στον οποίο θα εκτελεστεί (CPU, GPU, NNAPI) και το πλήθος νημάτων που θα χρησιμοποιηθούν).

Η βασικότερη κλάση είναι η ImageClassifier, καθώς είναι αυτή που τελικά χρησιμοποιεί τις μεθόδους του Tensorflow Lite για την εκτέλεση του νευρωνικού δυκτύου. Ο τρόπος που δουλεύει είναι ο εξής: παίρνει σαν παράμετρο ένα αντικείμενο της κλάσης ClassifierConfiguration που αναφέρθηκε παραπάνω, διαβάζει τη λίστα με τις ετικέτες από το αρχείο labels.txt και αρχικοποιεί έναν Interpreter του Tensorflow Lite. Στη συνέχεια όταν κληθεί η μέθοδος classifyImagesAsBatch με όρισμα μια λίστα από φωτογραφίες σε μορφή Bitmap δημιουργεί έναν ByteBuffer που περιέχει όλες τις φωτογραφίες και είναι η είσοδος της μεθόδου run του Interpreter. Στη συνέχεια μετατρέπει τα αποτελέσματα που επιστρέφονται σε μορφή ενός δεύτερου ByteBuffer σε μια λίστα από αντικείμενα τύπου ClassificationResult ώστε να είναι εύκολα διαχειρίσιμα.

- Το πακέτο data. Περιλαμβάνει τις κλάσεις που αναπαριστούν τους πίνακες της βάσης δεδομένων (Picture και Benchmark), τις αντίστοιχες κλάσεις για ανάγνωση και εγγραφή αντίστοιχων δεδομένων στη βάση (PictureDao και BenchmarkDao), την κλάση AppDatabase που είναι το κεντρικό σημείο εισόδου για τη βάση και το πακέτο converters με χρήσιμες μεθόδους για μετατροπή διαφόρων τύπων δεδομένων από και προς μορφές που μπορούν να αποθηκευτούν σαν στήλες των πινάκων στη βάση.

- Το πακέτο utils. Πρόκειται για μια συλλογή χρήσιμων μεθόδων που επιτρέπει την επαναχρησιμοποίηση του κώδικα. Περιλαμβάνει κλάσεις όπως η CollectionUtils (με μεθόδους όπως η partition, που τεμαχίζει μια λίστα σε ισοπληθή κομμάτια), StringUtils (με βοηθητικές μεθόδους για Strings), IOStreams και PictureFileUtils. Οι τελευταίες 2 κλάσεις είναι οι πιο σημαντικές, καθώς είναι αυτές που πραγματοποιούν την αποθήκευση, φόρτωση και διαγραφή των φωτογραφιών.

- Το πακέτο work. Περιλαμβάνει τον κώδικα που επιτρέπει τον προγραμματισμό της κατηγοριοποίησης φωτογραφιών στο παρασκήνιο με χρήση του WorkManager API (περισσότερα στην επόμενη ενότητα).

- Το πακέτο di. Ο κώδικας που περιλαμβάνει είναι υπεύθυνος για την υλοποίηση του dependency injection με χρήση της βιβλιοθήκης Hilt που περιγράφεται παρακάτω.

4.2. Βιβλιοθήκες

4.2.1. Navigation Component

Η βιβλιοθήκη Navigation Component [25] διευκολύνει την πλοήγηση μεταξύ διαφορετικών προορισμών στην εφαρμογή και το πέρασμα δεδομένων από τη

μια οθόνη στην άλλη εξασφαλίζοντας ότι το πλήθος και οι τύποι των παραμέτρων είναι τα σωστά. Μας επιτρέπει να χρησιμοποιούμε για το μεγαλύτερο μέρος της εφαρμογής ένα μόνο Activity, το οποίο περιλαμβάνει ένα στοιχείο τύπου FragmentContainerView που αντικαθίσταται κάθε φορά με το Fragment που αντιστοιχεί στην εκάστοτε οθόνη.

4.2.2. Room

Η βιβλιοθήκη Room [26] κάνει πιο απλή τη χρήση της βάσης δεδομένων SQLite του Android. Χρησιμοποιώντας ορισμένα Annotations, Interfaces και Classes που μας προσφέρει μπορούμε να ορίσουμε με εύκολο τρόπο τους πίνακες της βάσης μας - οντότητες, τις μεταξύ τους συσχετίσεις, τα ερωτήματα που μπορούμε να εκτελέσουμε κ.ο.κ. Επίσης προσφέρει δυνατότητα ελέγχου ορθότητας των ερωτημάτων μας από τη στιγμή που γράφουμε τον κώδικα, χωρίς να χρειαστεί να τρέξουμε την εφαρμογή και να αντιμετωπίσουμε τυχόν προβλήματα την ώρα της εκτέλεσης.

4.2.3. Glide

Η βιβλιοθήκη Glide [27] είναι μια από τις πιο διαδεδομένες βιβλιοθήκες για το Android. Φροντίζει για την φόρτωση των φωτογραφιών μας, εξασφαλίζει την υψηλή απόδοση της εφαρμογής μας μέσω διαχείρισης του Caching των φωτογραφιών, και έχει επιπλέον δυνατότητες προβολής προσωρινών φωτογραφιών ώσπου να φορτωθεί η φωτογραφία, μετασχηματισμού των φωτογραφιών (πχ περικοπή, περιστροφή) και άλλα.

4.2.4. PhotoView

Το PhotoView [28] είναι μια βιβλιοθήκη που προσθέτει στις εικόνες μας δυνατότητες Zoom με τρόπο ίδιο με των εφαρμογών που έχουν συνηθίσει οι χρήστες Android (δηλαδή pinch to zoom και double tap).

4.2.5. RX Java/RX Android

Αποτελεί υλοποίηση της πασίγνωστης βιβλιοθήκης ReactiveX για Java και Android [29]. Διευκολύνει την ανάπτυξη κώδικα που βασίζεται στα «Observable» και τις ασύγχρονες ροές δεδομένων. Χρησιμοποιείται για την επικοινωνία με τη βάση δεδομένων και την αποθήκευση των φωτογραφιών, εξασφαλίζοντας ότι θα εκτελούνται στο παρασκήνιο χωρίς να προκαλούν καθυστερήσεις και κολλήματα στην εφαρμογή.

4.2.6. Hilt

Αυτή η βιβλιοθήκη χρησιμοποιείται για την υλοποίηση του Design Pattern του Dependency Injection [30]. Μας επιτρέπει να ορίζουμε άλλες κλάσεις από τις οποίες εξαρτάται κάθε κλάση ώστε να τις παρέχονται χωρίς να χρειάζεται να τα αρχικοποιεί μόνη της τα αντίστοιχα αντικείμενα. Έτσι μπορεί κάθε κλάση να

εξαρτάται μόνο από Interfaces, οπότε μπορούμε εύκολα να αλλάζουμε τις υλοποιήσεις που χρησιμοποιούμε (π.χ. για τη συγγραφή Unit Tests).

4.2.7. WorkManager

Το WorkManager [31] είναι μια βιβλιοθήκη που επιτρέπει τον προγραμματισμό εργασιών για ασύγχρονη εκτέλεση με αξιοπιστία. Οι εργασίες που προγραμματίζονται με αυτόν τον τρόπο εκτελούνται ακόμα και αν η λειτουργία της εφαρμογής τερματιστεί ή η συσκευή επανεκκινηθεί. Επίσης δίνεται η δυνατότητα ορισμού συνθηκών που πρέπει να πληρούνται για να γίνει εκτέλεση της εργασίας, όπως για παράδειγμα να υπάρχει σύνδεση στο διαδίκτυο, να είναι αρκετά υψηλή η στάθμη της μπαταρίας ή να βρίσκεται η συσκευή σε κατάσταση φόρτισης. Στην εφαρμογή χρησιμοποιείται για τον προγραμματισμό της διαδικασίας κατηγοριοποίησης κάθε φωτογραφίας.

4.3. Βάση Δεδομένων

Η εφαρμογή χρησιμοποιεί για να διατηρεί αποθηκευμένα τα δεδομένα που πρέπει να διατηρούνται μια βάση δεδομένων SQLite [32] που παρέχεται από το Android, με την οποία επικοινωνεί με χρήση της βιβλιοθήκη Room που αναφέρθηκε παραπάνω. Η βάση δεδομένων αποτελείται από δυο βασικούς πίνακες:

- Τον πίνακα Pictures, όπου αποθηκεύονται τα μεταδεδομένα των πληροφοριών και τα αποτελέσματα της διαδικασίας ταξινόμησης (συγκεκριμένα αποθηκεύονται οι 5 πιο πιθανές κλάσεις μαζί με τα αντίστοιχα ποσοστά πεποίθησης).

- Τον πίνακα Benchmarks, όπου αποθηκεύονται τα αποτελέσματα των εκτελέσεων του μοντέλου που γίνονται για λόγους αξιολόγησης της απόδοσής του. Συγκεκριμένα κάθε γραμμή του πίνακα περιλαμβάνει το όνομα της συσκευής όπου πραγματοποιήθηκε η μέτρηση, το όνομα του μοντέλου που χρησιμοποιήθηκε, τον επεξεργαστή όπου έγινε η εκτέλεση, το πλήθος νημάτων, το μέγεθος δέσμης και το συνολικό χρόνο που χρειάστηκε. Η επεξεργασία αυτών των δεδομένων έγινε σε δεύτερο χρόνο και έδωσε τα αποτελέσματα που φαίνονται αναλυτικά στο επόμενο Κεφάλαιο.

4.4. Επεξεργασία Μοντέλων

4.4.1. Αλλαγή Διάστασης Εισόδου

Τα περισσότερα από τα μοντέλα που μελετήθηκαν υπήρχαν έτοιμα στο διαδίκτυο σε μορφή «.tflite», ωστόσο είχαν τον περιορισμό ότι δεν μπορούσαν να χρησιμοποιηθούν με μέγεθος δέσμης μεγαλύτερο του 1, καθώς αυτό ήταν το

μέγεθος της πρώτης διάστασης της εισόδου τους κατά τη μετατροπή τους σε αυτή τη μορφή. Αυτό το πρόβλημα αντιμετωπίστηκε κατεβάζοντας τα μοντέλα σε μορφή TensorFlow και ακολουθώντας τις οδηγίες για την εκ νέου μετατροπή τους σε μορφή «.tflite» με ορισμό του μεγέθους της πρώτης διάστασης εισόδου ως «None». Αυτό επιτρέπει στα μοντέλα να δεχτούν ως είσοδο πολλαπλές εικόνες.

4.4.2. Κβαντοποίηση

Σε αντίθεση με τα υπόλοιπα μοντέλα, στην περίπτωση των μοντέλων ResNet v2 101 και MnasNet-A1 δεν υπήρχαν διαθέσιμες στο διαδίκτυο οι κβαντισμένες εκδοχές τους. Γι' αυτό το λόγο έγινε μετατροπή τους σε μορφή «.tflite» με εφαρμογή Κβαντοποίησης Δυναμικού Εύρους (Dynamic Range Quantization). Υπάρχουν λοιπόν ενδεχομένως διαφορές στον τρόπο που κβαντοποιήθηκαν τα μοντέλα αυτά σε σχέση με τα υπόλοιπα, ωστόσο η μέθοδος αυτή προτιμήθηκε σε σχέση με την Πλήρη Ακέραιη Κβαντοποίηση (Full Integer Quantization) για λόγους απλότητας. Αντίθετα, οι κβαντισμένες μορφές των υπόλοιπων μοντέλων που βρέθηκαν έτοιμες στο διαδίκτυο έχουν προκύψει με Πλήρη Ακέραιη Κβαντοποίηση.

4.4.3. Μέτρηση Ορθότητας

Τα μοντέλα MobileNet v1 1.0.192, MobileNet v2 1.0.224 και Inception v3 συνοδεύονται από πληροφορίες για την ορθότητα τους (top-1 και top-5). Αντίθετα για τα MnasNet-A1 και ResNet v2 101 δεν υπήρχε διαθέσιμη αυτή η πληροφορία. Γι' αυτό το λόγο η ορθότητα αυτών των μοντέλων υπολογίστηκε στα πλαίσια της αξιολόγησης των μοντέλων με χρήση του εργαλείου ImageNet Image Classification Evalution [33].

5. Αξιολόγηση

Το παρόν κεφάλαιο είναι ένα από τα πιο σημαντικά ολόκληρης της διπλωματικής εργασίας, καθώς παρουσιάζει τη διαδικασία αξιολόγησης των μοντέλων και τα συμπεράσματα που προέκυψαν.

5.1. Μετρικές

Η απόδοση κάθε μοντέλου αξιολογήθηκε βάση μετρήσεων που αφορούν το χρόνο εκτέλεσης (latency) της συμπερασματολογίας και τη χρήση μνήμης και CPU κατά τη διάρκειά της. Επειδή στις μετρήσεις μας εκτελούμε συμπερασματολογία θέτοντας περισσότερες από μια εικόνες ως είσοδο χρησιμοποιούμε το μέγεθος διαπερατότητα (throughput) που ορίζεται ως ο λόγος του συνολικού χρόνου εκτέλεσης προς το πλήθος των εικόνων στην είσοδο και συμβολίζει το χρόνο που χρειάζεται για την ταξινόμηση κάθε εικόνας.

Οι μετρικές που χρησιμοποιούμε είναι το μέσο (mean) throughput και το 90th percentile throughput, δηλαδή η τιμή πάνω από την οποία βρίσκεται το 90% των μετρήσεων, η χρήση μνήμης σε απόλυτο αριθμό και η ποσοστιαία χρήση CPU.

5.2. Μοντέλα

Στον παρακάτω πίνακα παρουσιάζονται όλα τα μοντέλα τα οποία αξιολογήθηκαν και τα βασικά τους χαρακτηριστικά (μέγεθος εισόδου, Top-1 Ορθότητα, Top-5 Ορθότητα και Μέγεθος του μοντέλου), ταξινομημένα ως προς την Top-1 Ορθότητα.

Α/Α	**Μοντέλο**	**Κβαντοποίηση**	**Μέγεθος εισόδου**	**Top-1 Ορθότητα**	**Top-5 Ορθότητα**	**Μέγεθος**
1	Inception v3	-	299x299	77.9	93.8%	95.3MB
2	Inception v3	Full	299x299	77.5%	93.7%	23MB
3	ResNet v2 101	-	299x299	77.2%	93.9%	174MB
4	ResNet v2 101	DR	299x299	77.2%	93.8%	45MB
5	MobileNet v2 1.0.224	-	224x224	71%	89.9%	16.9MB
6	MobileNet v2 1.0.224	Full	224x224	70.8	89.9	3.4MB
7	MobileNet v1 1.0.192	-	192x192	69.9%	89.1%	16.9MB
8	MobileNet v1 1.0.192	Full	192x192	69.1%	88.1%	4.3MB
9	MnasNet-A1	-	224x224	66%	87.2%	15MB
10	MnasNet-A1	DR	224x224	64.4%	86.2%	4MB

Πίνακας 1: Μοντέλα που χρησιμοποιήθηκαν για την αξιολόγηση του συστήματος

5.3. Παράμετροι

Για κάθε μοντέλο πραγματοποιήθηκαν μετρήσεις για όλες τις διαφορετικές τιμές ενός πλήθους παραμέτρων, οι οποίες ήταν:

- Ο επεξεργαστής εκτέλεσης του μοντέλου, με πιθανές τιμές CPU, GPU και NNAPI. Το NNAPI (Neural Networks API) είναι στην πραγματικότητα ένας «εκπρόσωπος» που είναι διαθέσιμος στις εκδόσεις του Android από την 8.1 και μετά και προσφέρει επιτάχυνση στα μοντέλα χρησιμοποιώντας τη GPU, το DSP (Digital Signal Processor) ή το NPU (Neural Processing Unit), ανάλογα με το διαθέσιμο υλικό στην εκάστοτε συσκευή

- Το πλήθος επεξεργαστικών νημάτων, με τιμές 1, 2, 4 ή 8 (μόνο όταν η εκτέλεση γίνεται σε CPU).

- Το μέγεθος δέσμης (batch size), δηλαδή το πλήθος των εικόνων που εισάγονται ταυτόχρονα σαν είσοδος στο μοντέλο. Οι τιμές που χρησιμοποιήθηκαν ήταν οι δυνάμεις του 2 από 1 έως και 512 (ή όσο μεγαλύτερο ήταν δυνατό σε κάθε μοντέλο και κάθε συσκευή ανάλογα με τις απαιτήσεις σε μνήμη).

5.4. Μέθοδος Μετρήσεων

Για να διευκολυνθεί η διαδικασία των μετρήσεων προστέθηκε στην εφαρμογή μια οθόνη που επιτρέπει την επιλογή των παραμέτρων και του μοντέλου και κάνει πολλαπλές μετρήσεις ώστε να εξαχθούν οι μετρικές μας. Η χρονική διάρκεια κάθε εκτέλεσης υπολογίζεται με χρήση της μεθόδου System.currentTimeMillis() της Java, η οποία επιστρέφει τη τρέχουσα ώρα σε milliseconds. Η μέθοδος αυτή καλείται ακριβώς πριν και μετά την εκτέλεση και η διαφορά των δυο τιμών μας δίνει τη χρονική διάρκεια.

Για να είναι πιο αντιπροσωπευτικές οι μετρήσεις ήταν απαραίτητο να υπάρξει ένα πλήθος από warm up runs, δηλαδή ένα πλήθος αναγνωρίσεων που γίνονται πριν αρχίσουμε να καταγράφουμε τους χρόνους ώστε να προλάβει να προσαρμοστεί η συσκευή στις απαιτήσεις της εφαρμογής. Η τιμή που επιλέχθηκε ήταν 5. Επίσης έπρεπε να υπάρξει ένα πλήθος συνολικών runs μετά τα warm up runs. Αυτό καθορίστηκε ως 25, που αποτελεί συμβιβασμό ανάμεσα σε τιμές που θα έκαναν τις μετρήσεις μη αντιπροσωπευτικές ή το πλήθος των μετρήσεων απαγορευτικά μεγάλο.

5.4.1. Συσκευές Δοκιμών

Για να εξασφαλιστεί ότι τα αποτελέσματα των δοκιμών δεν εξαρτώνται από συγκεκριμένα χαρακτηριστικά μιας και μόνο συσκευής, χρησιμοποιήθηκαν για τις μετρήσεις δυο διαφορετικές συσκευές. Οι συσκευές κυκλοφόρησαν την ίδια χρονιά ωστόσο η πρώτη ήταν μια από τις ισχυρότερες συσκευές της χρονιάς ενώ

η δεύτερη μια συσκευή χαμηλότερων επιδόσεων, οπότε τα αποτελέσματα θα είναι πιο αντιπροσωπευτικά για ένα μεγαλύτερο εύρος συσκευών.

Μοντέλο	**Samsung Galaxy Note 9**	**Xiaomi Redmi Note 5**
Έκδοση Android	10	9
System on Chip (SoC)	Exynos 9810	Qualcomm Snapdragon 636
CPU	Octa-core (4x2.7 GHz Mongoose M3 & 4x1.8 GHz Cortex-A55)	Octa-core (4x1.8 GHz Kryo 260 Gold & 4x1.6 GHz Kryo 260 Silver)
GPU	Mali-G72 MP18	Adreno 509
Μνήμη RAM	6 GB	4 GB
Ημερομηνία κυκλοφορίας	Αύγουστος 2018	Μάρτιος 2018

Πίνακας 2: Χαρακτηριστικά Συσκευών Δοκιμών

Όπως φαίνεται και στον παραπάνω πίνακα η πρώτη συσκευή έχει 2 GB περισσότερη μνήμη RAM, γεγονός που θα της επιτρέψει να ανταπεξέλθει καλύτερα σε μεγαλύτερες εισόδους και πιο βαριά μοντέλα.

Στο εξής η συσκευή της Samsung θα αναφέρεται ως συσκευή A και η συσκευή της Xiaomi ως συσκευή B.

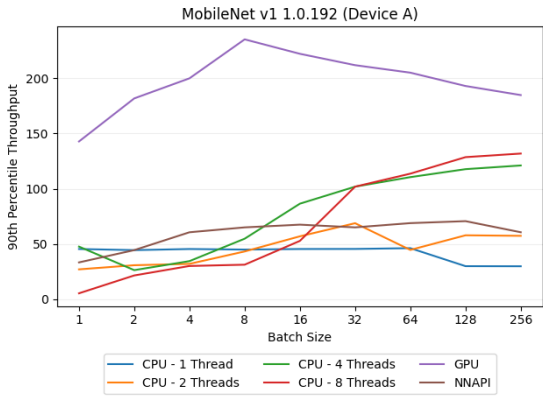
5.5. Αποτελέσματα

5.5.1. Διαπερατότητα

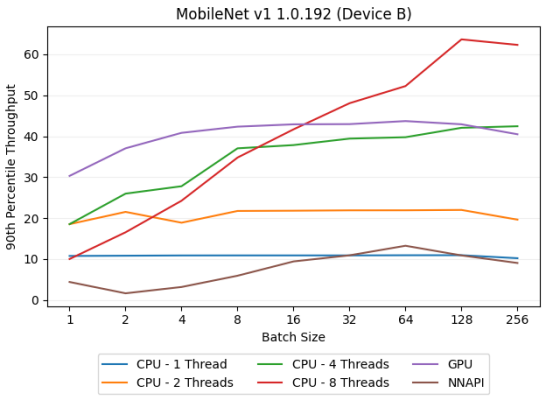
Παρακάτω παρουσιάζονται τα αποτελέσματα των μετρήσεων για κάθε μοντέλο ξεχωριστά, με τη μορφή γραφικών παρουσιάσεων του 90th percentile throughput ως συνάρτηση του μεγέθους δέσμης για κάθε μοντέλο σε κάθε συσκευή και με κάθε δυνατό επεξεργαστή και πλήθος νημάτων (στην περίπτωση της CPU).

Για κάθε μοντέλο παρουσιάζονται τα αποτελέσματα για την FP και για την κβαντισμένη εκδοχή του για κάθε συσκευή και ακολουθεί ένας σύντομος σχολιασμός.

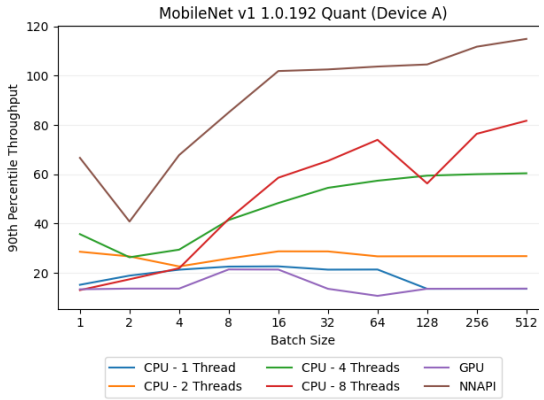
Mobilenet v1 1.0.192



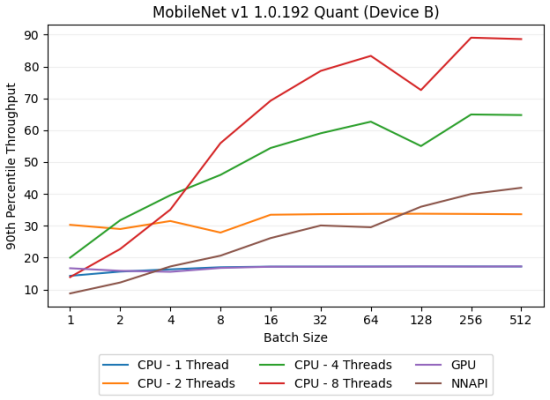
Εικόνα 5: MobileNet v1 1.0.192 - Συσκευή Α



Εικόνα 8: MobileNet v1 1.0.192 - Συσκευή Β



Εικόνα 7: MobileNet v1 1.0.192 Quant - Συσκευή Α

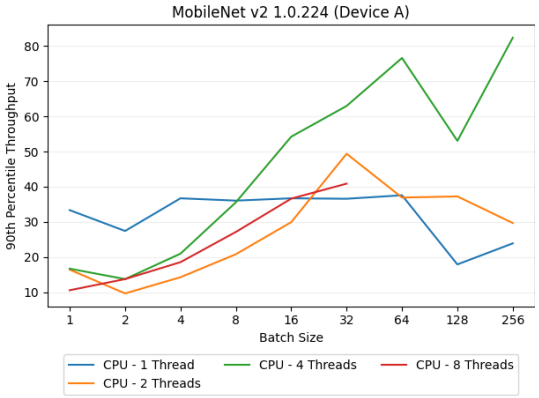


Εικόνα 6: MobileNet v1 1.0.192 Quant - Συσκευή Β

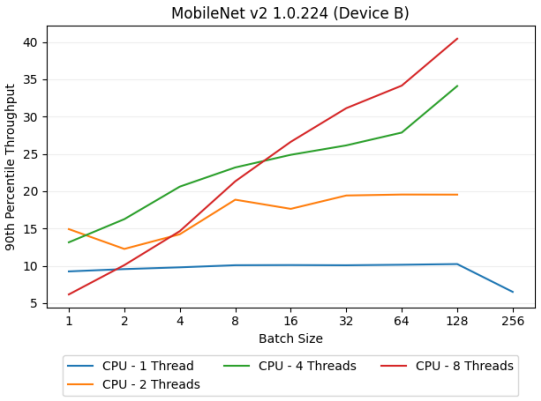
Παρατηρούμε ότι για την FP εκδοχή του μοντέλου η κορυφαία με διαφορά επίδοση προκύπτει στη συσκευή Α με χρήση GPU και για μέγεθος δέσμης 8, ενώ μετά αρχίζει η πτώση. Αντίστοιχα στη κβαντισμένη εκδοχή του για τη συσκευή Α οι καλύτερες επιδόσεις επιτυγχάνονται με χρήση του NNAPI, με ραγδαία αύξηση μέχρι και την τιμή 16 για το μέγεθος δέσμης και συνεχιζόμενη ανοδική πορεία έως και για την τιμή 512.

Όσον αφορά τη συσκευή Β, βλέπουμε ότι ξεχωρίζει και στις 2 εκδοχές του μοντέλου η CPU με χρήση νημάτων, ενώ φαίνεται ξεκάθαρα μια κορυφή της διαπερατότητας για μέγεθος δέσμης 128 και 256 αντίστοιχα, όπου και αρχίζει σταδιακά η πτώση.

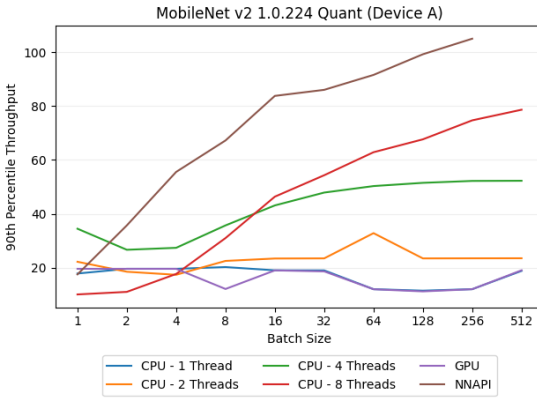
Mobilenet v2 1.0.224



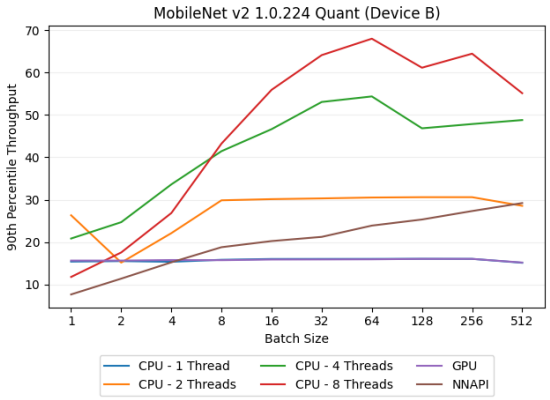
Εικόνα 10: MobileNet v2 1.0.224 - Συσκευή Α



Εικόνα 11: MobileNet v2 1.0.224 - Συσκευή Β



Εικόνα 9: MobileNet v2 1.0.224 Quant - Συσκευή Α



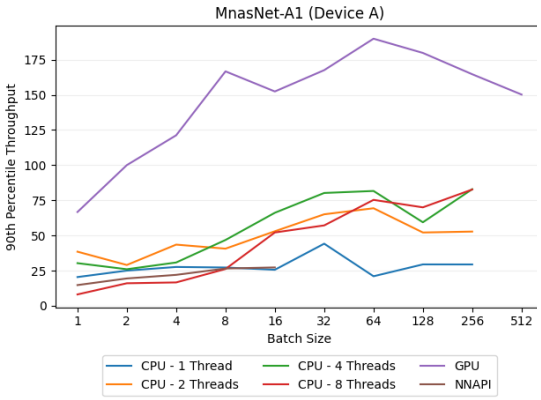
Εικόνα 12: MobileNet v2 1.0.224 Quant - Συσκευή Β

Στην περίπτωση αυτού του μοντέλου βλέπουμε καταρχάς ότι λόγω ασυμβατότητας με κάποια λειτουργία δεν υπάρχουν μετρήσεις για τις περιπτώσεις χρήσης GPU και NNAPI για την FP εκδοχή του.

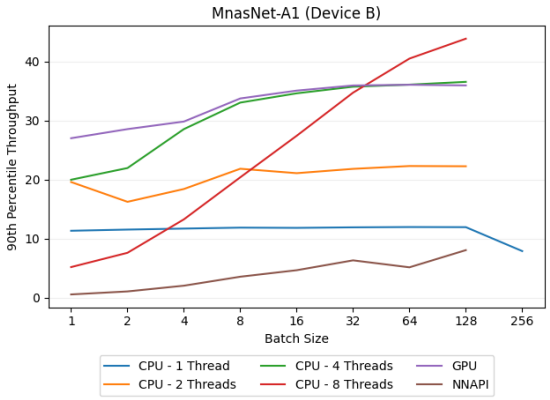
Για τη συσκευή Α βλέπουμε τα καλύτερα αποτελέσματα στην FP εκδοχή με χρήση CPU και 4 νημάτων, ωστόσο ίσως παίζει κάποιο ρόλο ότι σε όσες προσπάθειες έγιναν για μετρήσεις με χρήση 8 νημάτων και μέγεθος δέσμης πάνω από 32 υπήρχαν προβλήματα και άρα δεν υπάρχουν αποτελέσματα. Όσον αφορά την κβαντισμένη εκδοχή βλέπουνε να ξεχωρίζει ξανά το NNAPI.

Για τη συσκευή Β βλέπουμε και στις 2 περιπτώσεις να ξεχωρίζει η περίπτωση χρήσης CPU με 8 νήματα, ενώ GPU και NNAPI δίνουν στην περίπτωση της κβαντισμένης εκδοχής τα χειρότερα αποτελέσματα.

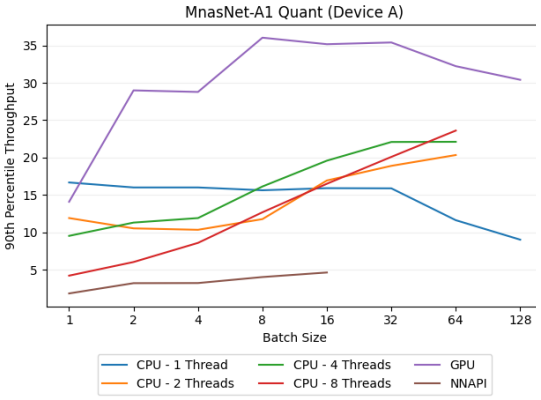
MnasNet-A1



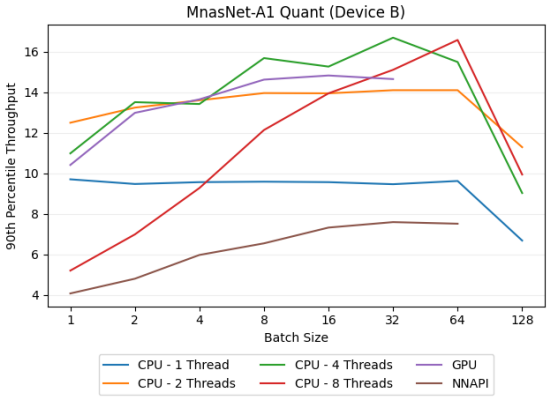
Εικόνα 15: MnasNet-A1 - Συσκευή Α



Εικόνα 16: MnasNet-A1 - Συσκευή Β



Εικόνα 13: MnasNet-A1 Quant - Συσκευή Α

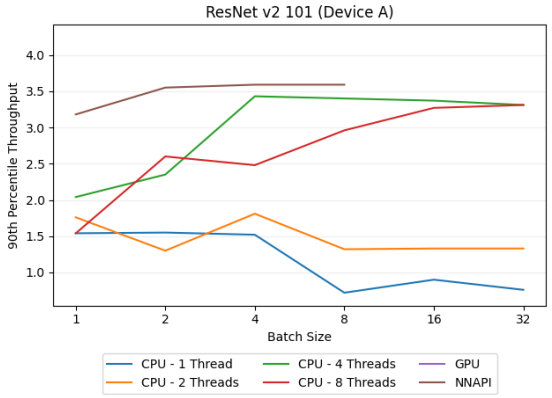


Εικόνα 14: MnasNet-A1 Quant - Συσκευή Β

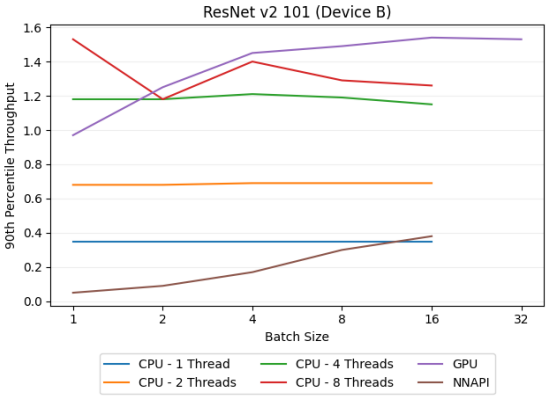
Στην περίπτωση της συσκευής Α βλέπουμε και στις δυο εκδοχές πολύ παρόμοια αποτελέσματα, με τη GPU να δίνει τα καλύτερα αποτελέσματα (με κορυφαία τιμή για μέγεθος δέσμης 64 και 8 αντίστοιχα και στη συνέχεια φθίνουσα πορεία). Το αξιοσημείωτο σε σχέση με τα προηγούμενα μοντέλα είναι ότι οι επιδόσεις στη κβαντισμένη εκδοχή με χρήση NNAPI είναι αυτή τη φορά οι χαμηλότερες, ενώ στα 2 μοντέλα της οικογένειας MobileNet το NNAPI ξεχώριζε θετικά.

Όσον αφορά τη συσκευή Β οι καλύτερες επιδόσεις είναι και πάλι και στις 2 περιπτώσεις με χρήση CPU και 8 νημάτων. Στην περίπτωση FP έχουμε αύξουσα πορεία έως και το μέγεθος δέσμης 128 όπου αρχίζει να μειώνεται η θετική κλίση, ενώ στην κβαντισμένη εκδοχή έχουμε απότομη μείωση μετά την μέγιστη τιμή που σημειώθηκε για μέγεθος δέσμης 64.

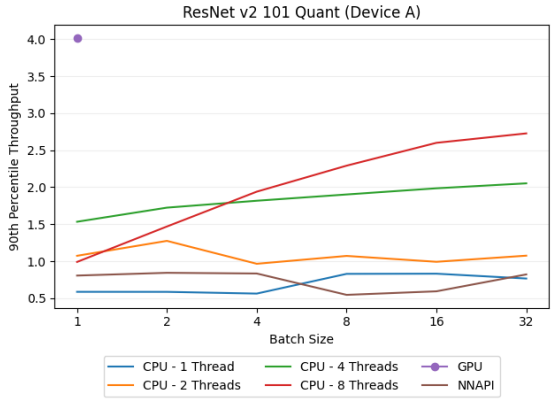
ResNet v2 101



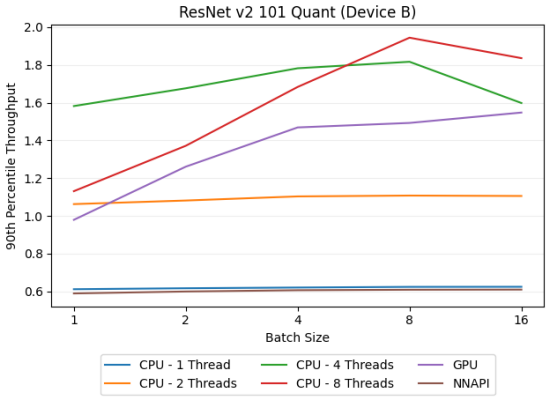
Εικόνα 18: ResNet v2 101 - Συσκευή Α



Εικόνα 17: ResNet v2 101 - Συσκευή Β



Εικόνα 20: ResNet v2 101 Quant - Συσκευή Α



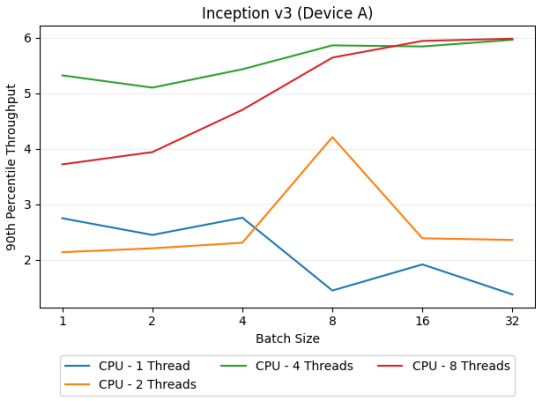
Εικόνα 19: ResNet v2 101 Quant - Συσκευή Β

Σε αυτό το μοντέλο βλέπουμε με ότι οι κορυφαίες επιδόσεις διαπερατότητας είναι μονοψήφιοι αριθμοί, ενώ το μέγιστο μέγεθος δέσμης που μετρήθηκε επιτυχώς (λόγω πολύ μεγάλης χρήσης μνήμης) ήταν μόλις 32.

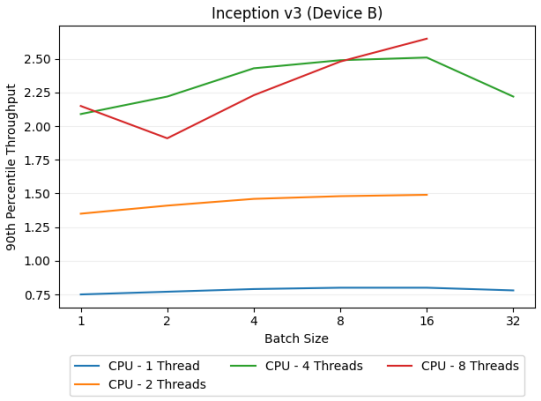
Στην περίπτωση της συσκευής Α οι καλύτερες επιδόσεις είναι για NNAPI με μέγεθος δέσμης 4 στην FP εκδοχή, ενώ στην κβαντισμένη είναι για GPU με μέγεθος δέσμης 1 (για τιμές μεγαλύτερες του 1 η συσκευή επανεκκινούταν κατά τη διάρκεια εκτέλεσης) .

Αντίστοιχα στη συσκευή Β οι καλύτερες επιδόσεις ήταν με χρήση GPU στην FP εκδοχή και με CPU και 8 νήματα στη κβαντισμένη, με μέγεθος δέσμης 16 και 8 αντίστοιχα.

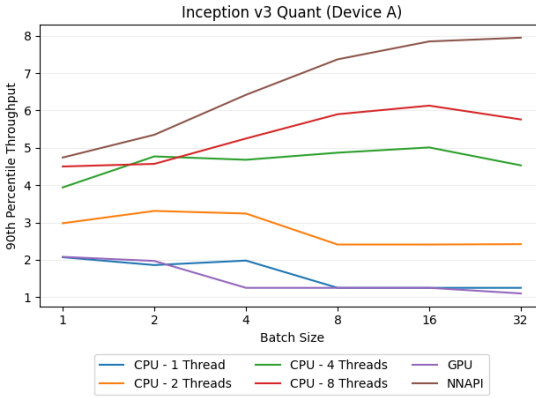
Inception v3



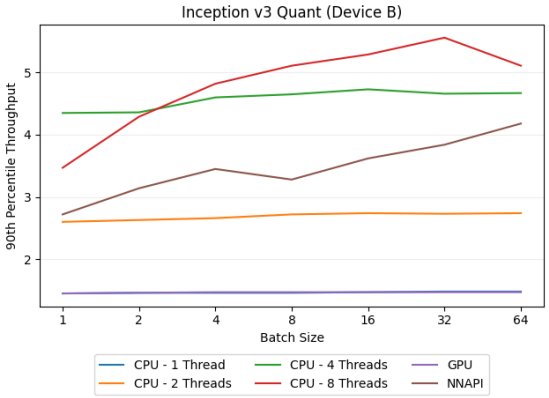
Εικόνα 22: Inception v3 - Συσκευή Α



Εικόνα 21: Inception v3 - Συσκευή Β



Εικόνα 24: Inception v3 Quant - Συσκευή Α



Εικόνα 23: Inception v3 Quant - Συσκευή Β

Και σε αυτό το μοντέλο οι τιμές της διαπερατότητας κυμαίνονται σε μονοψήφια νούμερα, ενώ υπήρξε πάλι αδυναμία εκτέλεσης της FP εκδοχής με χρήση GPU και NNAPI.

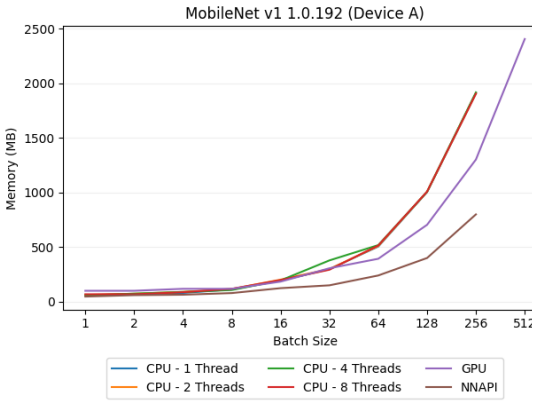
Για τη συσκευή Α τα καλύτερα αποτελέσματα ήταν με χρήση 4 και 8 νημάτων στην FP εκδοχή και με χρήση NNAPI στη κβαντισμένη με μέγεθος δέσμης κοντά στο 32.

Αντίστοιχα για τη συσκευή Β οι καλύτερες επιδόσεις ήταν και στις δυο περιπτώσεις με χρήση CPU και 8 νημάτων, με μέγεθος δέσμης 16 και 32 αντίστοιχα.

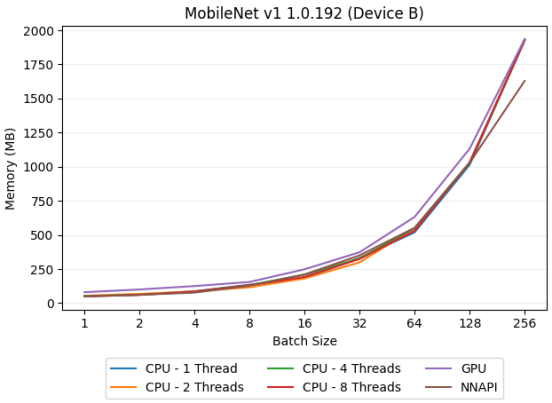
5.5.2. Μνήμη

Για τη μελέτη της χρήσης μνήμης επιλέχθηκαν οι δυο εκδοχές ενός από τα πιο ελαφριά μοντέλα, καθώς και η FP εκδοχή ενός από τα πιο βαριά μοντέλα. Οι μετρήσεις έγιναν με χρήση του Profiler του Android Studio. Παρακάτω παρουσιάζονται οι γραφικές παραστάσεις της χρήσης μνήμης ως συνάρτηση του μεγέθους δέσμης για τα τρία αυτά μοντέλα.

MobileNet v1 1.0.192

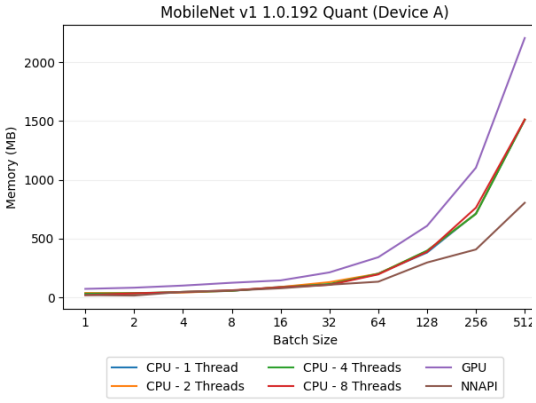


Εικόνα 26: MobileNet v1 1.0.192 (Μνήμη) - Συσκευή A

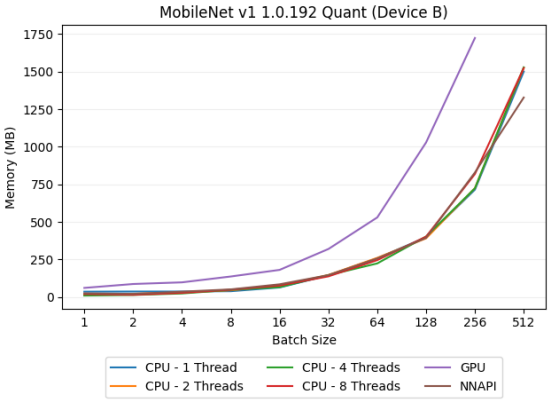


Εικόνα 25: MobileNet v1 1.0.192 (Μνήμη) - Συσκευή B

MobileNet v1 1.0.192

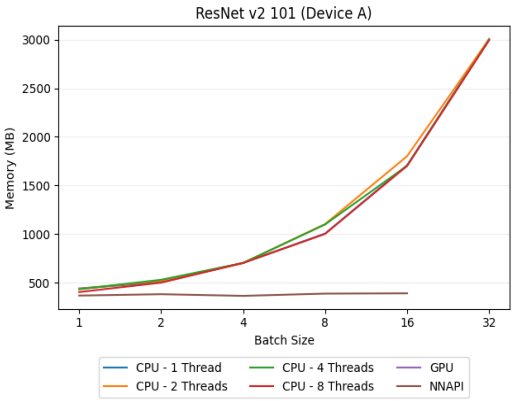


Εικόνα 27: MobileNet v1 1.0.192 Quant (Μνήμη) - Συσκευή A

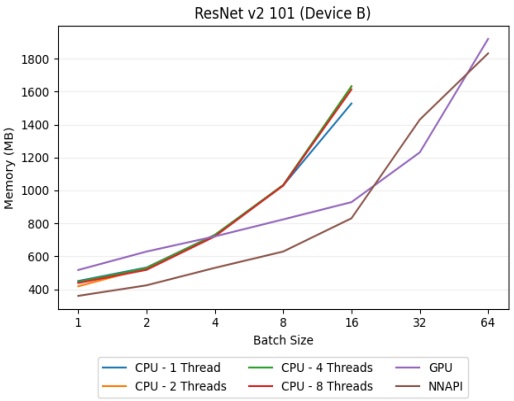


Εικόνα 28: MobileNet v1 1.0.192 Quant (Μνήμη) - Συσκευή B

ResNet v2 101



Εικόνα 30: ResNet v2 101 (Μνήμη) - Συσκευή A



Εικόνα 29: ResNet v2 101 (Μνήμη) - Συσκευή A

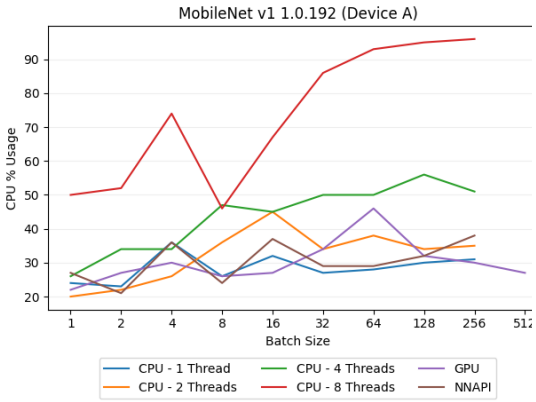
Ανάμεσα στις δυο εκδοχές του MobileNet βλέπουμε ότι όπως ήταν αναμενόμενο η FP εκδοχή έχει μεγαλύτερες απαιτήσεις μνήμης. Επίσης παρατηρούμε εν γένει μια τάση το NNAPI να έχει τις μικρότερες απαιτήσεις μνήμης και η GPU τις μεγαλύτερες.

Όσον αφορά το ResNet, είναι εμφανές ότι οι απαιτήσεις μνήμης του είναι τάξεις μεγέθους μεγαλύτερες, οπότε επιβεβαιώνεται ότι πράγματι είναι ένα πολύ βαρύ μοντέλο.

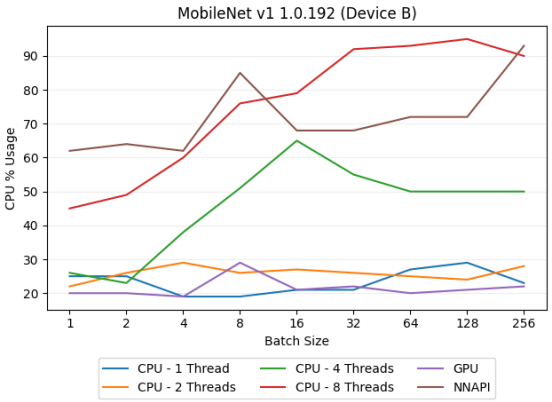
5.5.3. Χρήση CPU

Παρακάτω παρουσιάζονται οι γραφικές παραστάσεις της χρήσης CPU ως συνάρτηση του μεγέθους δέσμης, όπως μετρήθηκε με χρήση του Profiler του Android Studio.

MobileNet v1 1.0.192

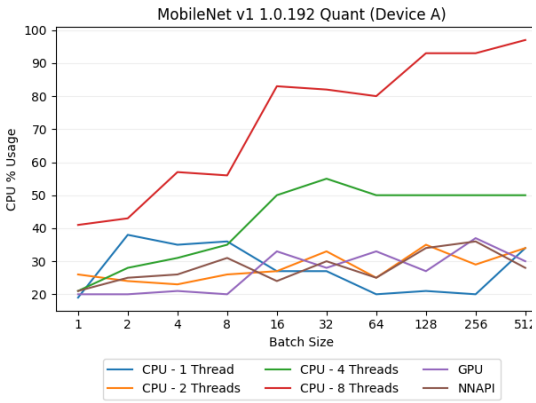


Εικόνα 32: MobileNet v1 1.0.192 (CPU) - Συσκευή A

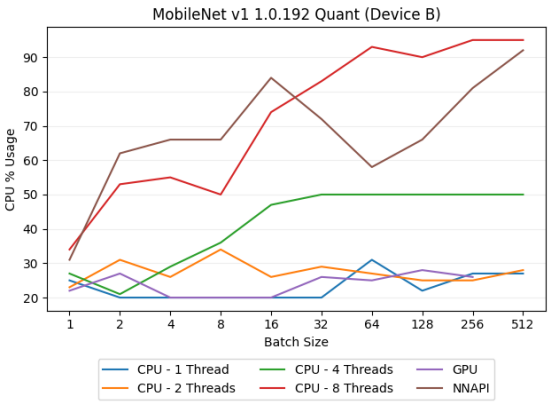


Εικόνα 31: MobileNet v1 1.0.192 (CPU) - Συσκευή B

MobileNet v1 1.0.192 Quant

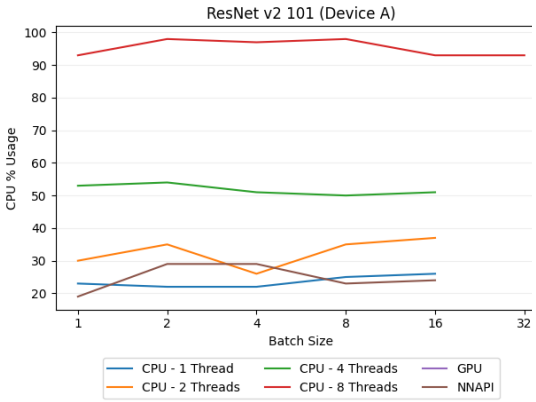


Εικόνα 33: MobileNet v1 1.0.192 Quant (CPU) - Συσκευή A

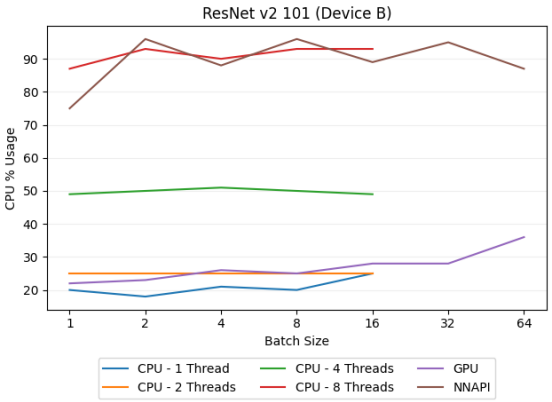


Εικόνα 34: MobileNet v1 1.0.192 Quant (CPU) -Συσκευή B

ResNet v2 101



Εικόνα 36: ResNet v2 101 (CPU) - Συσκευή Α



Εικόνα 35: ResNet v2 101 (CPU) - Συσκευή Β

Παρατηρούμε ότι σε όλες τις περιπτώσεις καθώς αυξάνουμε το πλήθος των νημάτων που χρησιμοποιούνται στην περίπτωση της χρήσης CPU αυξάνεται και η μέγιστη χρήση CPU, γεγονός που περιμέναμε.

Όσον αφορά την περίπτωση όπου χρησιμοποιείται η GPU, το ποσοστό χρήσης της CPU είναι αρκετά χαμηλό ακόμα και σε βαριά μοντέλα με μεγάλα μεγέθη δέσμης, γεγονός που είναι επίσης αναμενόμενο, αφού το μεγαλύτερο μέρος του φορτίου εκτελείται στη GPU.

Αυτό που έχει περισσότερο ενδιαφέρον είναι ότι στην περίπτωση του NNAPI η συσκευή Α παρουσιάζει σταθερά χαμηλά ποσοστά χρήσης CPU, ενώ η συσκευή Β πολύ υψηλά. Αυτό ίσως σημαίνει ότι η συσκευή Β δεν διαθέτει εξειδικευμένους επεξεργαστές όπως η NPU που αναφέρθηκε παραπάνω ή δεν μπορεί για λόγους συμβατότητας να τους χρησιμοποιήσει, και αναγκαστικά χρησιμοποιεί και πάλι σε μεγάλο βαθμό τη CPU.

5.5.4. Κορυφαίες Επιδόσεις

Από τις προηγούμενες ενότητες είναι σαφές ότι το κάθε μοντέλο έχει διαφορετική συμπεριφορά για κάθε συνδυασμό παραμέτρων και σε κάθε συσκευής. Στην παρούσα ενότητα θα παρουσιαστεί συνοπτικά η μέγιστη τιμή της διαπερατότητας για κάθε μοντέλο σε κάθε συσκευή, καθώς και οι τιμές των παραμέτρων για τις οποίες επιτεύχθηκε.

Μοντέλο	Μέγιστη Διαπερατότητα	Επεξεργαστής	Νήματα	Μέγεθος Δέσμης
MobileNet v1 1.0.192	235,29	GPU	-	8
MnasNet-A1	189,91	GPU	-	64
MobileNet v1 1.0.192 Quant	114,95	NNAPI	-	512
MobileNet v2 1.0.224 Quant	105	NNAPI	-	256
MobileNet v2 1.0.224	82,42	CPU	4	256
MnasNet-A1 Quant	36,04	GPU	-	8
Inception v3 Quant	7,95	NNAPI	-	32
Inception v3	5,98	CPU	8	32
ResNet v2 101	4,24	GPU	-	1
ResNet v2 101 Quant	4,02	GPU	-	1

Πίνακας 3: Κορυφαία διαπερατότητα ανά μοντέλο - Συσκευή Α

Μοντέλο	Μέγιστη Διαπερατότητα	Επεξεργαστής	Νήματα	Μέγεθος Δέσμης
MobileNet v1 1.0.192 Quant	89,04	CPU	8	256
MobileNet v2 1.0.224 Quant	68,01	CPU	8	64
MobileNet v1 1.0.192	63,65	CPU	8	128
MnasNet-A1	43,88	CPU	8	128
MobileNet v2 1.0.224	40,44	CPU	8	128
MnasNet-A1 Quant	16,69	CPU	4	32
Inception v3 Quant	5,56	CPU	8	32
Inception v3	2,65	CPU	8	16
ResNet v2 101 Quant	1,94	CPU	8	8
ResNet v2 101	1,54	GPU	-	16

Πίνακας 4: Κορυφαία διαπερατότητα ανά μοντέλο - Συσκευή Β

Παρατηρούμε ότι στη συσκευή Α στα περισσότερα FP μοντέλα οι κορυφαίες επιδόσεις επιτυγχάνονται με χρήση GPU, όπως ήταν αναμενόμενο λόγω των υψηλών επιδόσεων που πετυχαίνουν εν γένει οι GPU σε πράξεις με τέτοιους αριθμούς. Αντίθετα σε 3 από τα 5 κβαντισμένα μοντέλα οι καλύτερες επιδόσεις επιτεύχθηκαν με χρήση NNAPI, που μας δείχνει ότι η συσκευή Α έχει καλύτερη συμβατότητα με το NNAPI. Εξαίρεση αποτελούν τα 2 μοντέλα τα οποία δεν ήταν εφικτό να εκτελεστούν με χρήση NNAPI ή GPU, όπου οι καλύτερες επιδόσεις

επιτεύχθηκαν με 4 και 8 νήματα CPU αντίστοιχα, επιβεβαιώνοντας την εκτίμηση ότι η χρήση περισσότερων νημάτων βελτιώνει τις επιδόσεις.

Τα αποτελέσματα για τη συσκευή Β είναι εντελώς διαφορετικά, καθώς σε 9 από τα 10 μοντέλα οι καλύτερες επιδόσεις επιτεύχθηκαν με χρήση 8 νημάτων CPU. Αυτό μας δείχνει ότι ενδεχομένως οι σχετικές επιδόσεις της GPU είναι χαμηλότερες σε σχέση με της CPU, ενώ και το NNAPI δε δείχνει να υποστηρίζεται επαρκώς. Τα δυο αυτά γεγονότα ίσως εξηγούνται από το ότι η συσκευή Β ήταν μια οικονομική συσκευή, οπότε δεν εξοπλίζεται με GPU υψηλών επιδόσεων και υποστήριξη για το NNAPI, σε αντίθεση με τη συσκευή Α που ήταν μια συσκευή υψηλών επιδόσεων.

Όσον αφορά το μέγεθος δέσμης για το οποίο επιτυγχάνονται οι κορυφαίες επιδόσεις, συνήθως ήταν σε τιμές μεταξύ 32 και 256, με εξαιρέσεις κυρίως στα ResNet και Inception που λόγω απαιτήσεων μνήμης δεν μπορούσαν να εκτελεστούν με τόσο μεγάλα μεγέθη δέσμης.

Τα μοντέλα με τις καλύτερες επιδόσεις είναι και στις δυο συσκευές τα MobileNet και το MnasNet-A1, ενώ τις χειρότερες επιδόσεις έχουν τα ResNet και Inception. Λόγω της πολύ καλής GPU της συσκευής Α την απόλυτη πρωτιά πέτυχε ένα FP MobileNet, ενώ στη συσκευή Β και τις δυο θέσεις της κορυφής τις κατέλαβαν κβαντισμένα μοντέλα.

6. Επίλογος

Στο τελευταίο Κεφάλαιο αναλύονται τα συμπεράσματα, η συνεισφορά της διπλωματικής εργασίας και ορισμένες παρατηρήσεις. Τέλος, γίνεται αναφορά σε ορισμένες πιθανές κατευθύνσεις μελλοντικής επέκτασης της διπλωματικής εργασίας

6.1. Συμπεράσματα

Βλέποντας τον πίνακα με τα χαρακτηριστικά των μοντέλων αλλά και τα αποτελέσματα των μετρήσεων στο Κεφάλαιο 6, είναι προφανές ότι δεν μπορεί να ανακηρυχθεί ένα μοντέλο ως κορυφαίο, καθώς το καθένα υπερτερεί σε διαφορετικές κατηγορίες, και η επιλογή του καταλληλότερου μοντέλου για κάθε εφαρμογή εξαρτάται από τις ανάγκες και τους περιορισμούς που υπάρχουν.

Για παράδειγμα τα Inception v3 και ResNet v2 101 ξεχωρίζουν όσον αφορά την ακρίβεια, ωστόσο για να το πετύχουν αυτό έχουν με διαφορά το μεγαλύτερο μέγεθος, το χαμηλότερο throughput και τις υψηλότερες απαιτήσεις μνήμης. Από πλευρά ακρίβειας ακολουθούν τα 2 MobileNet, τα οποία είναι πολύ καλύτερα ως προς το throughput και τη χρήση μνήμης. Τελευταίο έρχεται το MnasNet, το οποίο δείχνει συνολικά να υστερεί έναντι των υπολοίπων μοντέλων, αφού η χαμηλή ακρίβεια του δε δείχνει να συνοδεύεται στις μετρήσεις μας από ταχύτητα υψηλότερη έναντι των MobileNet.

Όσον αφορά την εξάρτηση της διαπερατότητας από τον επεξεργαστή εκτέλεσης, βλέπουμε ότι στην περίπτωση της χρήσης CPU έχουμε όπως ήταν αναμενόμενο καλύτερη απόδοση όσο αυξάνεται το πλήθος των νημάτων. Στην περίπτωση της χρήσης GPU έχουμε συνήθως καλύτερες επιδόσεις σε μοντέλα floating point παρά σε κβαντισμένα, ωστόσο αυτό είναι πιο ορατό στην περίπτωση της συσκευής A που λόγω υψηλότερης κατηγορίας τιμής έχει και ισχυρότερη GPU. Τέλος, τα συμπεράσματα σχετικά με τις επιδόσεις με χρήση NNAPI δεν μπορούν να γενικευτούν, αφού εξαρτώνται σε μεγάλο βαθμό από τη συμβατότητα κάθε συσκευής με διάφορες λειτουργίες του και από τις ιδιαιτερότητες κάθε μοντέλου.

Αντίστοιχα για την εξάρτηση της διαπερατότητας από το μέγεθος δέσμης βλέπουμε ότι καθώς αυξάνεται το μέγεθος δέσμης αυξάνεται και η διαπερατότητα, με τάση από ένα σημείο και μετά να αρχίσει ξανά να μειώνεται. Ωστόσο αυτή η καμπή δεν είναι πάντα ορατή επειδή οι περιορισμοί μνήμης και χρόνου δεν επέτρεπαν να αυξάνουμε επ’ αόριστον το μέγεθος δέσμης.

Ως προς τη χρήση μνήμης έχουμε εν γένει μεγαλύτερη χρήση στην περίπτωση εκτέλεση στη GPU και μικρότερη στο NNAPI,

Τέλος, αξίζει να σημειωθεί ότι πολλές φορές υπήρξαν προβλήματα κατά τις μετρήσεις με διάφορους συνδυασμούς συσκευών, επεξεργαστών και μοντέλων που οδήγησαν σε προβλήματα από διακοπή λειτουργίας της εφαρμογής έως και τερματισμό λειτουργίας της ίδιας της συσκευής, γι’ αυτό και υπάρχουν ορισμένα κενά στις μετρήσεις.

6.2. Μελλοντικές επεκτάσεις

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής μπορεί να αναπτυχθεί περεταίρω και προς διαφορετικές κατευθύνσεις. Ενδεικτικά αναφέρονται οι εξής κατευθύνσεις:

6.2.1. Μελέτη περισσότερων Μοντέλων

Μια ενδιαφέρουσα επέκταση θα ήταν η μελέτη περισσότερων και νεότερων μοντέλων. Για παράδειγμα μια οικογένεια πολλά υποσχόμενων μοντέλων είναι τα EfficientNets. Ωστόσο η προσπάθεια να μελετηθούν απέτυχε, καθώς δεν ήταν δυνατή η μετατροπή τους ώστε να υποστηρίξουν μεγέθη δέσμης μεγαλύτερα του 1 για να αξιολογηθούν επί ίσοις όροις με τα υπόλοιπα μοντέλα.

6.2.2. Εξειδίκευση Μοντέλων

Όλα τα μοντέλα που μελετήθηκαν χρησιμοποιούν το ίδιο σύνολο 1000 (ή 1001) κατηγοριών. Ωστόσο πολλές από αυτές τις κατηγορίες αφορούν αντικείμενα που σπάνια εμφανίζονται σε φωτογραφίες ενός μέσου χρήστη. Θα μπορούσαν με χρήση μεθόδων μεταφοράς μάθησης (transfer learning) να εκπαιδευτούν μοντέλα που να αναγνωρίζουν με μεγάλη ακρίβεια συγκεκριμένες κατηγορίες (π.χ. οικιακές συσκευές, φυτά, ζώα, φαγητά). Έτσι ο χρήστης θα μπορούσε να επιλέγει εξαρχής την ευρύτερη κατηγορία στην οποία ανήκει το αντικείμενο της φωτογραφίας του και να πάρει μια πιο ακριβή κατηγοριοποίηση, ή θα μπορούσε να δοκιμαστεί μια αρχιτεκτονική με μοντέλα στη σειρά, ένα γενικό για μια πρώτη κατηγοριοποίηση και ένα πιο ειδικό που θα επιλεγόταν με βάση το αποτέλεσμα του 1ου και θα έδινε πιο ακριβή αποτελέσματα.

6.2.3. Άλλες Εφαρμογές Όρασης Υπολογιστών

Η κατηγοριοποίηση φωτογραφιών είναι μια πολύ σημαντική εφαρμογή της Όρασης Υπολογιστών, ωστόσο απέχει πολύ από το να είναι η μοναδική.

Ένα βασικό μειονέκτημα όλων των μοντέλων κατηγοριοποίησης εικόνων είναι ότι αναθέτουν σε ολόκληρη τη φωτογραφία μια μοναδική ετικέτα, χωρίς να μπορούν να λάβουν υπόψιν το ενδεχόμενο να εμφανίζονται στην ίδια φωτογραφία περισσότερα από ένα αντικείμενα. Αυτός ο περιορισμός μπορεί να ξεπεραστεί με χρήση μοντέλων ανίχνευσης αντικειμένων (object detection) [34], τα οποία μπορούν να εντοπίσουν περισσότερα του ενός αντικείμενα στην ίδια φωτογραφία.

Μια άλλη χρήσιμη λειτουργία για την εφαρμογή θα ήταν η δυνατότητα εξαγωγής κειμένου από τις φωτογραφίες με χρήση τεχνικών αναγνώρισης κειμένου (Optical Character Recognition)., που ήταν όπως αναφέρθηκε στο πρώτο Κεφάλαιο και μια από τις πρώτες εφαρμογές Βαθιάς Μάθησης.

Ένα μεγάλο ποσοστό των φωτογραφιών που έχει ο μέσος χρήστης στο κινητό του τηλέφωνο είναι φωτογραφίες με οικεία του πρόσωπα από διάφορες δραστηριότητες. Αντί να ανιχνεύονται απλά τα αντικείμενα που υπάρχουν σε μια φωτογραφία θα ήταν χρήσιμο να χρησιμοποιείται κάποιο μοντέλο αναγνώρισης προσώπων (face recognition) ώστε να γίνεται αυτόματα ομαδοποίηση των

φωτογραφιών ανάλογα με τα πρόσωπα που εμφανίζονται σε αυτές, ή κάποιο μοντέλο για αναγνώριση σκηνής (scene recognition) ώστε να ομαδοποιούνται οι φωτογραφίες με βάση το χώρο όπου έχουν ληφθεί.

Μια ακόμα εφαρμογή είναι η «Υπερανάλυση» (super resolution) [35], δηλαδή η δημιουργία μιας εικόνας υψηλής ανάλυσης που προκύπτει από μια εικόνα χαμηλότερης ανάλυσης. Ένα τέτοιο μοντέλο είναι το ESRGAN, που καταφέρνει λόγο κλιμάκωσης 4 με πολύ καλύτερα αποτελέσματα ως προς τη ποιότητα εικόνας σε σχέση με κλασικές μεθόδους. Χρησιμοποιώντας ένα τέτοιο μοντέλο θα μπορούσε η εφαρμογή να βελτιώνει αυτόματα την ποιότητα των φωτογραφιών που προσθέτει ο χρήστης. Αντίστοιχες μέθοδοι χρησιμοποιούνται ήδη για τη βελτίωση της ποιότητας εικόνας σε ηλεκτρονικά παιχνίδια σε πραγματικό χρόνο χωρίς να επιβαρύνονται οι επιδόσεις τους (π.χ. NVIDIA DLSS [36]).

Τέλος, μια πολύ σημαντική εφαρμογή που έχει τις προοπτικές να κάνει τη ζωή πολλών ανθρώπων πιο εύκολη είναι η χρήση τεχνικών όπως είναι η εκτίμηση πόζας (pose estimation) για αναγνώριση της νοηματικής γλώσσας σε πραγματικών, ώστε να γίνει πιο εύκολη η επικοινωνία ατόμων με προβλήματα ομιλίας.

6.2.4. Υβριδικό Σύστημα

Στο 1ο Κεφάλαιο αναφέρθηκε για ποιο λόγο προτιμήθηκε η συμπερασματολογία να γίνεται τοπικά στη συσκευή. Ωστόσο μια πιο σύνθετη εφαρμογή θα μπορούσε να συνδυάσει τη συμπερασματολογία που πραγματοποιείται τοπικά με συμπερασματολογία που γίνεται με χρήση κάποιας απομακρυσμένης υπηρεσίας. Η επιλογή μπορεί να γίνεται είτε από τον χρήστη μιας τέτοιας εφαρμογής είτε με αυτόματο τρόπο. Για παράδειγμα μπορεί από προεπιλογή να προτιμάται η τοπική λειτουργία, ωστόσο εάν εντοπίζεται σύνδεση στο διαδίκτυο και η συσκευή έχει μικρή επεξεργαστική ισχύ ή χαμηλή στάθμη μπαταρίας να προτιμάται η χρήση ενός απομακρυσμένου συστήματος ώστε να εξοικονομηθούν οι πόροι του συστήματος, εφόσον ο χρήστης έχει δώσει τη συγκατάθεσή του. Αντίστοιχα θα μπορούσε η πρώτη επιλογή να είναι η χρήση ενός απομακρυσμένου συστήματος αλλά σε περίπτωση μη ύπαρξης σύνδεσης με το διαδίκτυο να χρησιμοποιείται προσωρινά κάποιο μοντέλο που θα εκτελείται τοπικά, έστω και με μικρότερη ακρίβεια, και όταν επανέλθει η σύνδεση στο διαδίκτυο να γίνεται τότε χρήση του απομακρυσμένου συστήματος.

7. Βιβλιογραφία

[1] "What is Machine Learning? | IBM." https://www.ibm.com/cloud/learn/machine-learning (accessed Nov. 03, 2021).

[2] "What Is Machine Learning? - I School Online." https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/ (accessed Nov. 04, 2021).

[3] "What is Computer Vision? | IBM." https://www.ibm.com/topics/computer-vision (accessed Nov. 04, 2021).

[4] "A Brief History of Computer Vision (and Convolutional Neural Networks) | Hacker Noon." https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3 (accessed Nov. 04, 2021).

[5] "Convolutional Neural Networks (CNNs) and Layer Types - PyImageSearch." https://www.pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/ (accessed Nov. 03, 2021).

[6] "CS231n Convolutional Neural Networks for Visual Recognition." https://cs231n.github.io/convolutional-networks/#conv (accessed Nov. 04, 2021).

[7] "Illustration of Max Pooling and Average Pooling Figure 2 above shows an... | Download Scientific Diagram." https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451 (accessed Nov. 04, 2021).

[8] "Residual blocks — Building blocks of ResNet | by Sabyasachi Sahoo | Towards Data Science." https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec (accessed Nov. 04, 2021).

[9] "2D Depthwise convolution." https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-depthwise-convolution (accessed Nov. 04, 2021).

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012, Accessed: Nov. 04, 2021. [Online]. Available: http://code.google.com/p/cuda-convnet/

[12] C. Szegedy *et al.*, "[googLenet]Going deeper with convolutions Christian," *Population Health Management*, vol. 18, no. 3, pp. 186–191, 2015, Accessed: Nov. 04, 2021. [Online]. Available: http://online.liebertpub.com/doi/10.1089/pop.2014.0089

[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, Accessed: Nov. 04, 2021. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*,

vol. 2016-December, pp. 2818–2826, Dec. 2015, Accessed: Nov. 04, 2021. [Online]. Available: https://arxiv.org/abs/1512.00567v3

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, Dec. 2015, Accessed: Nov. 04, 2021. [Online]. Available: https://arxiv.org/abs/1512.03385v1

[16] M. Tan and Q. v. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, May 2019, Accessed: Nov. 04, 2021. [Online]. Available: https://arxiv.org/abs/1905.11946v5

[17] "Mobile Operating System Market Share Worldwide | Statcounter Global Stats." https://gs.statcounter.com/os-market-share/mobile/worldwide (accessed Nov. 04, 2021).

[18] "What is Android? Everything you need to know about Google's OS." https://www.androidauthority.com/what-is-android-328076/ (accessed Nov. 04, 2021).

[19] "Java | Definition & Facts | Britannica." https://www.britannica.com/technology/Java-computer-programming-language (accessed Nov. 04, 2021).

[20] "Java Architecture - Javatpoint." https://www.javatpoint.com/java-architecture (accessed Nov. 04, 2021).

[21] "Meet Android Studio | Android Developers." https://developer.android.com/studio/intro (accessed Nov. 04, 2021).

[22] "TensorFlow Lite." https://www.tensorflow.org/lite/guide (accessed Nov. 04, 2021).

[23] "Home | TensorFlow Hub." https://tfhub.dev/ (accessed Nov. 03, 2021).

[24] "TensorFlow Lite Model Maker." https://www.tensorflow.org/lite/guide/model_maker (accessed Nov. 04, 2021).

[25] "Navigation | Android Developers." https://developer.android.com/guide/navigation (accessed Nov. 04, 2021).

[26] "Save data in a local database using Room | Android Developers." https://developer.android.com/training/data-storage/room (accessed Nov. 04, 2021).

[27] "Glide v4 : Fast and efficient image loading for Android." https://bumptech.github.io/glide/ (accessed Nov. 04, 2021).

[28] "photoview/photoview: Photo gallery for self-hosted personal servers." https://github.com/photoview/photoview (accessed Nov. 04, 2021).

[29] "ReactiveX/RxJava: RxJava – Reactive Extensions for the JVM – a library for composing asynchronous and event-based programs using observable sequences for the Java VM." https://github.com/ReactiveX/RxJava (accessed Nov. 04, 2021).

[30] "Hilt." https://dagger.dev/hilt/ (accessed Nov. 04, 2021).

[31] "Schedule tasks with WorkManager | Android Developers." https://developer.android.com/topic/libraries/architecture/workmanager (accessed Nov. 04, 2021).

[32] "SQLite Home Page." https://www.sqlite.org/index.html (accessed Nov. 04, 2021).

[33] "tensorflow/tensorflow/lite/tools/evaluation/tasks/imagenet_image_classificati

on at master · tensorflow/tensorflow.” https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/tools/evaluation/tasks/imagenet_image_classification (accessed Nov. 04, 2021).
[34] “Object detection | TensorFlow Lite.” https://www.tensorflow.org/lite/examples/object_detection/overview (accessed Nov. 04, 2021).
[35] “Super resolution with TensorFlow Lite.” https://www.tensorflow.org/lite/examples/super_resolution/overview (accessed Nov. 04, 2021).
[36] “Deep Learning Super Sampling (DLSS) Technology | NVIDIA.” https://www.nvidia.com/en-eu/geforce/technologies/dlss/ (accessed Nov. 04, 2021).

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

κ.	κυρία, κύριος
κοκ	και ούτω καθεξής
π.χ.	παραδείγματος χάρη
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DSP	Digital Signal Processor
GPU	Graphics Processing Unit
NNAPI	Neural Network API
ReLU	rectified linear unit
RAM	Random Access Memory
SoC	System on a Chip

Απόδοση ξενόγλωσσων όρων

Απόδοση	**Ξενόγλωσσος Όρος**
Ανίχνευση Αντικειμένων	Object Detection
Βαθιά Μάθηση	Deep Learning
βάρος	weight
βηματισμός	stride
δεδομένα	data
Διεπαφή Χρήστη	User Interface
διερμηνέας	interpreter
δομικό στοιχείο	component
δραστηριότητα	activity
εκπρόσωπος	delegate
Εκτίμηση Πόζας	Pose Estimation
ελαφρύ	light-weight
εφαρμογή	application
είσοδος	input
ετικέτα	label
κάμερα	camera
κανονικοποίηση	normalization
κβαντοποίηση	quantization
Κεντρική Μονάδα Επεξεργασίας	Central Processing Unit
κινητή συσκευή	mobile device
μέγεθος	size
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
μοντέλο	model
νήμα	thread
Ολοκληρωμένο Προγραμματιστικό Περιβάλλον	Integrated Development Environment
ορθότητα	accuracy
πεποίθηση	confidence
πλήρως συνδεδεμένο	fully-connected
πυρήνας	kernel
σιγμοειδής	sigmoid
συμπερασματολογία	inference
στρώμα	layer
συνάρτηση ενεργοποίησης	activation function
Συνελικτικά Νευρωνικά Δίκτυα	Convolutional Neural Networks
συγκέντρωση	pooling
Σύνολο δεδομένων	dataset
ταξινομητής	classifier

Τεχνητή Νοημοσύνη	Artificial Intelligence
τεχνητό νευρωνικό δίκτυο	artificial neural network
Τοπικό δεκτικό πεδίο	local receptive field
υλικό	hardware
φίλτρο	filter
χαμηλού επιπέδου	low-level
χαρακτηριστικό	feature
χάρτης	map
χιλιοστό του δευτερολέπτου	millisecond
χρόνος εκτέλεσης	latency