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

Υποβοηθούμενες και Κατανεμημένες Τεχνικές Βαθιάς Μάθησης για τη Σημασιολογική Ταξινόμηση Εικόνων σε Κινητές Συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αθανάσιος Γ. Μπίμης

Επιβλέπων : Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2021

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

Υποβοηθούμενες και Κατανεμημένες Τεχνικές Βαθιάς Μάθησης για τη Σημασιολογική Ταξινόμηση Εικόνων σε Κινητές Συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αθανάσιος Γ. Μπίμης

Επιβλέπων : Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Ιουλίου 2021.

.....	.....	.....
Ι. Στ. Βενιέρης	Δ. -Θ. Κακλαμάνη	Αθ. Δ. Παναγόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2021

.....

Αθανάσιος Γ. Μπίμης

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

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν η σχεδίαση και ανάπτυξη μιας ολοκληρωμένης εφαρμογής αποθήκευσης και κατηγοριοποίησης εικόνας (AI Gallery App) για κινητές συσκευές με υποβοήθηση εξυπηρετητή.
Συγκεκριμένα, έγινε σχεδιασμός εφαρμογής Android για κινητές συσκευές που εμφανίζει σε μορφή πλέγματος κατηγοριοποιημένες εικόνες για κάθε χρήστη, όπως και ο εξυπηρετητής που βρίσκεται στο νέφος ή την άκρη του δικτύου, για την πραγματοποίηση της κατηγοριοποίησης με τη βοήθεια μοντέλων μηχανικής μάθησης.
Καθώς οι στόχοι του συστήματος που δημιουργήθηκε ήταν η υψηλή διεκπεραιωτική ικανότητα (εξυπηρετητής νέφους ή στην άκρη του δικτύου), η παραμετροποίηση ως προς το νευρωνικό δίκτυο, η υποστήριξη διαφορετικών συσκευών και η επεκτασιμότητα ως προς τον αριθμό των χρηστών, έγιναν συγκρίσεις με τη βοήθεια μετρήσεων που βοήθησαν στην εξαγωγή επιθυμητών συμπερασμάτων.

Λέξεις Κλειδιά

Python, Java, TensorFlow, Android, Firebase, Μηχανική Μάθηση, Βαθιά Μάθηση, Συνελικτικά Νευρωνικά Δίκτυα, Εξυπηρετητής Νέφους, Εξυπηρετητής στην Άκρη του Δικτύου, FastAPI

Abstract

The purpose of the thesis was to design and develop an integrated image storage and categorization application (AI Gallery App) for mobile devices with server assistance.
In particular, an Android application was designed for mobile devices that displays categorized images in a grid format for each user, as well as the server located on the cloud or the edge of the network, to perform the categorization with the help of machine learning models.
As the goals of the system created were high throughput (cloud server or network edge), neural network configuration, support for different devices, and scalability in terms of number of users, comparisons were made using measurements. which helped to draw the desired conclusions.

KeyWords

Python, Java, TensorFlow, Android, Firebase, Machine Learning, Deep Learning, Convolutional Neural Networks, Cloud Server, Edge Server, FastAPI

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαιτέρως τον επιβλέποντα καθηγητή Ε.Μ.Π.
κ. Ιάκωβο Στ. Βενιέρη, για την ευκαιρία που μου έδωσε να διευρύνω τις γνώσεις μου με το αρκετά ενδιαφέρον αντικείμενο της Μηχανικής Μάθησης στις κινητές συσκευές.
Επίσης, θα ήθελα να ευχαριστήσω τον κ. Ιωάννη Πανόπουλο, Υποψήφιο Διδάκτωρ Ε.Μ.Π., για τη συνεχή και πολύτιμη καθοδήγησή του, που χωρίς αυτή δεν θα ήταν δυνατή η ολοκλήρωση της διπλωματικής.
Κλείνοντας, θα ήθελα πω ένα μεγάλο ευχαριστώ στην οικογένεια μου, για την στήριξη, τη συμπαράσταση και την υπομονή, όλα αυτά τα χρόνια της ζωής και των σπουδών μου.

Αθήνα, Ιούλιος 2021

Αθανάσιος Μπίμης

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1: Θεωρητικό Υπόβαθρο

1.1 Μηχανική Μάθηση

Ως Μηχανική Μάθηση (Machine Learning) ορίζεται ο κλάδος της Τεχνητής Νοημοσύνης (Artificial Inteligence), που επικεντρώνεται στην κατασκευή εφαρμογών που μαθαίνουν από τα δεδομένα και βελτιώνουν την ακρίβειά τους με τον χρόνο, χωρίς την βοήθεια περαιτέρω προγραμματισμού.
Μερικά παραδείγματα χρήσης Μηχανικής Μάθησης υπάρχουν παντού γύρω μας. Ψηφιακοί βοηθοί αναζητούν στο διαδίκτυο και παίζουν μουσική σύμφωνα με ηχητικές εντολές. Ιστοσελίδες προτείνουν προϊόντα, ταινίες και τραγούδια σύμφωνα με αυτά που έχουμε αγοράσει, παρακολουθήσει ή ακούσει. Ρομπότ σκουπίζουν το πάτωμά μας. Ανιχνευτές spam σταματούν ανεπιθύμητα email από το να φτάσουν τα εισερχόμενά. Συστήματα Ιατρικής ανάλυσης εικόνων βοηθούν τους γιατρούς να εντοπίσουν όγκους που μπορεί να έχουν παραβλέψει και τα πρώτα αυτόματα αμάξια κάνουν την εμφάνισή τους στο δρόμο [1].

Οι αλγόριθμοι της μηχανικής μάθησης μπορούν να χωριστούν στις παρακάτω τρείς κατηγορίες, ανάλογα με τον τρόπο που λαμβάνεται η μάθηση:

- **Επιβλεπόμενη Μάθηση (Supervised Learning)**

  Ως Επιβλεπόμενη Μάθηση ορίζεται η διαδικασία εκμάθησης ενός αλγορίθμου από το σύνολο δεδομένων, που μπορεί να περιγραφεί ως ένας δάσκαλος που επιβλέπει τη διαδικασία μάθησης. Ο δάσκαλος γνωρίζει τις σωστές απαντήσεις και διορθώνει τις προβλέψεις του αλγορίθμου σε κάθε επανάληψη. Η εκμάθηση σταματά όταν ο αλγόριθμος φτάσει ένα επιθυμητό επίπεδο απόδοσης.

  Τα προβλήματα Επιβλεπόμενης Μάθησης μπορούν να χωριστούν σε δύο κατηγορίες:

  1. Ταξινόμησης (Classification): Προβλήματα όπου η έξοδος είναι διακριτή τιμή κατηγορίας.
  2. Παλινδρόμησης (Regression): Προβλήματα όπου η έξοδος είναι συνεχής πραγματική τιμή.

- **Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)**

  Ως Μη Επιβλεπόμενη Μάθηση ορίζεται η διαδικασία εκμάθησης ενός αλγορίθμου από ένα σύνολο δεδομένων, που στη συγκεκριμένη περίπτωση δεν υπάρχουν σωστές απαντήσεις και δεν υπάρχει δάσκαλος. Οι αλγόριθμοι αφήνονται στη δική τους

κρίση για να ανακαλύψουν και να παρουσιάσουν την δική τους εξήγηση για τα δεδομένα.

Τα προβλήματα Μη Επιβλεπόμενης Μάθησης μπορούν να χωριστούν σε δύο κατηγορίες:

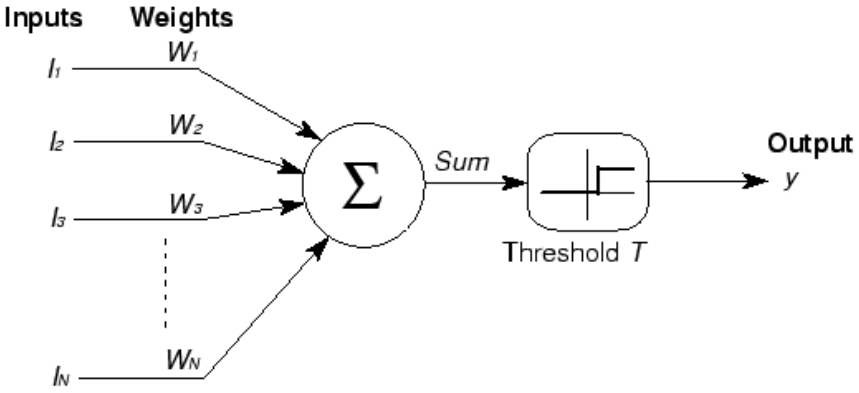
1. Ομαδοποίησης (Clustering): Προβλήματα που πρέπει να διακριθούν οι διάφορες ομάδες δεδομένων.
2. Συσχέτισης (Association): Προβλήματα που πρέπει να διακριθούν - περιγραφούν οι κανόνες μεταξύ των δεδομένων [2], [3].

- **Ενισχυτική Μάθηση (Reinforcement Learning)**

  Ως Ενισχυτική Μάθηση ορίζεται η μέθοδος Machine Learning που σχετίζεται με τον τρόπο που ένας πράκτορας (agent) πρέπει να λαμβάνει δράσεις σε ένα περιβάλλον. Οι αλγόριθμοι ενισχυτική μάθησης βοηθούν τον πράκτορα να φτάσει το στόχο, κάνοντας χρήση ενός αθροιστικού συστήματος επιβράβευσης. Ο στόχος του συστήματος είναι η μεγιστοποίηση του αθροίσματος για τις συνολικές ενέργειες του πράκτορα [4].

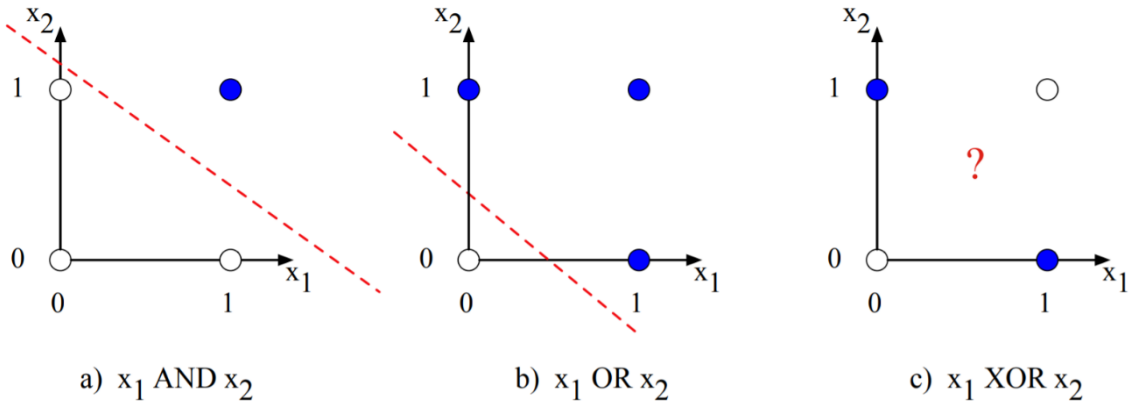
Η Μηχανική Μάθηση στις μέρες μας συνδέεται συνήθως με τη Βαθιά Μάθηση και τα νευρωνικά δίκτυα, αυτή όμως δεν είναι ολόκληρη η αλήθεια.
Τα πρώτα βήματα της Μηχανικής Μάθησης προτάθηκαν από τους McCulloch και Pitts το 1943 ως ένα απλό ηλεκτρικό κύκλωμα (Εικόνα 1.1), που απεικονίζει τον τρόπο που λειτουργεί ένας εγκέφαλος. Αργότερα, ο Hebb (1949) βάσει του μοντέλου McCulloch – Pitts, βρίσκει ένα μοτίβο στο νευρολογικό δίκτυο και προτείνει πως οι νευρολογικές οδοί δυναμώνουν με κάθε επιτυχή χρήση, ειδικά μεταξύ νευρώνων που ενεργοποιούνται την ίδια στιγμή [5], [6].
Ο πρώτος που επινόησε τη Μηχανική Μάθηση και αναφέρθηκε σε αυτή με τον συγκεκριμένο όρο, ήταν ο Arthur Samuel το 1952. Ο Samuel, μηχανικός της IBM, ανέπτυξε ένα πρόγραμμα παιχνιδιού ντάμας. Καθώς το πρόγραμμα είχε ελάχιστη διαθέσιμη μνήμη, ο Samuel υιοθέτησε τον αλγόριθμο αναζήτησης alpha-beta pruning [7], έτσι ώστε να μειώσει τον αριθμό των nodes που ελέγχονται, με τη βοήθεια του minimax αλγορίθμου για τα δέντρα αναζήτησης.

Το 1957, ο Frank Rosenblat – στο εργαστήριο αεροναυπηγικής του πανεπιστημίου Cornell, συνδυάζοντας το μοντέλο εγκεφαλικού κυττάρου του Donald Hebb (Εικόνα 1.1) με το μοντέλο Μηχανικής Μάθησης του Samuel, δημιούργησε το Perceptron. Το Perceptron ήταν ένα λογισμικό που αρχικώς σχεδιάστηκε για τον IBM 704 με στόχο την αναγνώριση εικόνων [5].



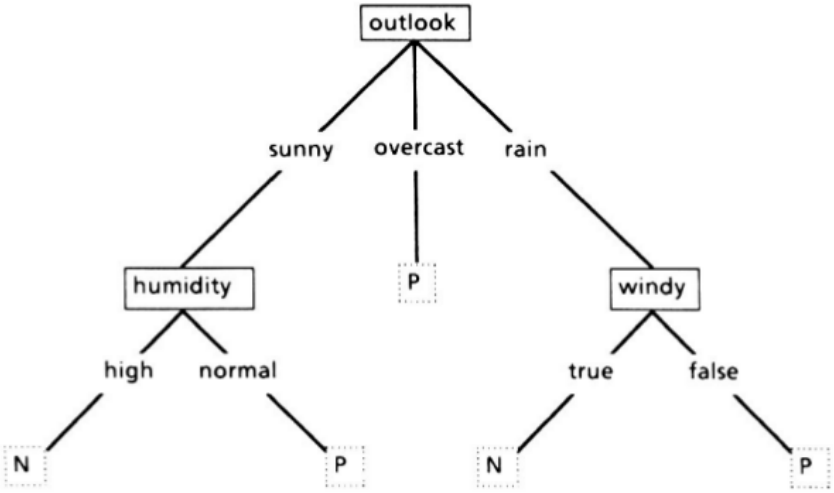
Εικόνα 1.1: Νευρώνας McCulloch - Pitts

Ωστόσο, ο ενθουσιασμός του Perceptron δεν κράτησε πολύ, καθώς το 1969 ο Minsky πρότεινε το διάσημο XOR πρόβλημα (Εικόνα 1.2) και την αδυναμία των Perceptron στα γραμμικά μη - διαχωρίσιμα δεδομένα. Έπειτα από αυτή την ανακάλυψη, οι έρευνες στον τομέα των νευρωνικών δικτύων έμειναν στάσιμες μέχρι την δεκαετία του 80 [5].



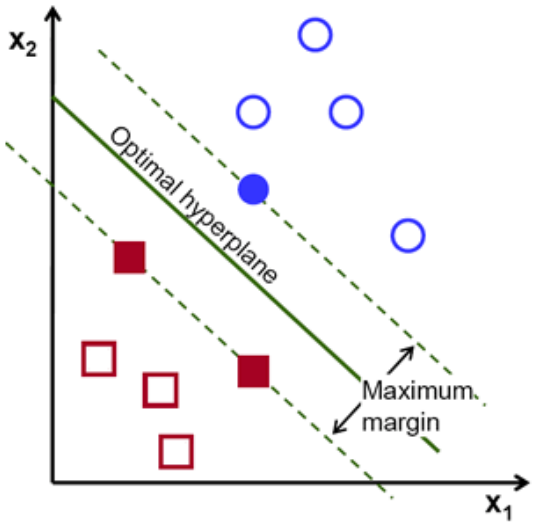
Εικόνα 1.2: XOR πρόβλημα του Minsky

Στις αρχές της δεκαετίας του 80 και συγκεκριμένα το 1981, ο Werbros πρότεινε την ιδέα του Πολυεπίπεδου Perceptron με τη βοήθεια του αλγορίθμου οπίσθιας διάδοσης (Backpropagation - BP) του Linnainmaa (1970).
Με τη μέθοδο αυτή γίνεται βελτίωση των βαρών βάσει της συχνότητας εμφάνισης λαθών σε κάθε επανάληψη. Ο αλγόριθμος του Backpropagation είναι κλειδί ακόμη και σήμερα για πολλούς αλγορίθμους Μηχανικής Μάθησης.
Την ίδια δεκαετία και συγκεκριμένα το 1986, προτάθηκε από τον J. R. Quinlan ο αλγόριθμος ταξινόμησης ID3 (Iterative Dichotomiser 3). Ο ID3 ακολουθεί μια άπληστη προσέγγιση για την κατασκευή ενός δέντρου απόφασης επιλέγοντας το χαρακτηριστικό που αποδίδει το μέγιστο κέρδος πληροφορίας (Information Gain) ή την ελάχιστη εντροπία. Κυκλοφόρησε ως λογισμικό για την εύρεση πραγματικών υποθέσεων λόγω των απλοϊκών του κανόνων, σε αντίθεση με τη λογική "μαύρου κουτιού" των νευρωνικών δικτύων [8].



Εικόνα 1.3: Απλό δέντρο απόφασης

Το 1995 προτάθηκε από τους Vapnik και Cortes μια από τις πιο σημαντικές ανακαλύψεις στη Μηχανική Μάθηση, τα Support Vector Machines (SVM). Η ανακάλυψη αυτή οδήγησε σε σχίσμα μεταξύ υποστηρικτών των νευρωνικών δικτύων και SVMs.
Τα SVMs είναι δυνατοί και ευέλικτοι αλγόριθμοι Επιβλεπόμενης Μηχανικής Μάθησης που χρησιμοποιούνται για ταξινόμηση και παλινδρόμηση. Οι μηχανές έγιναν γνωστές για τη δυνατότητά τους να διαχειρίζονται πολλαπλές και διαφορετικές τιμές κατηγοριών. Το μοντέλο των SVMs είναι μια απεικόνιση διαφορετικών κλάσεων σε ένα πολυδιάστατο χώρο, όπως φαίνεται στην Εικόνα 1.4.



Εικόνα 1.4: Ταξινόμηση με χρήση Support Vector Machine

Αυτός ο χώρος δημιουργείται με τη μορφή επαναλήψεων, έτσι ώστε το περιθώριο λάθους να μπορέσει να μεγιστοποιηθεί, με στόχο τον διαχωρισμό των δεδομένων σε διακριτές κλάσεις. Η πραγματική δύναμη του αλγόριθμου εξαρτάται από τον πυρήνα που χρησιμοποιείται μεταξύ γραμμικού, gauss, και πολυωνυμικού τύπου [8].
Μια ακόμη ενίσχυση για τη Μηχανική Μάθηση ήρθε στη συνέχεια από τους Freund και Schapire το 1997, με το Adaboost, μια ενισχυμένη μορφή αδύναμων ταξινομητών.
Το Adaboost εκπαιδεύει αδύναμους ταξινομητές που είναι εύκολοι στην εκπαίδευση, δίνοντας μεγαλύτερη προσοχή στις δύσκολες περιπτώσεις. Η βάση του μοντέλου αξιοποιείται ακόμη και σήμερα σε εργασίες όπως είναι η αναγνώριση και η ανίχνευση προσώπων [9].
Πλησιάζοντας στο σήμερα, λόγω της ευκολίας πρόσβασης σε μεγάλα σύνολα δεδομένων στο διαδίκτυο και της αύξησης των δυνατοτήτων των υπολογιστών, η Μηχανική Μάθηση έχει αρχίσει να κατακτάται όλο και περισσότερο από τα νευρωνικά δίκτυα και τη Βαθιά Μάθηση. Αν και τα νευρωνικά δίκτυα παίζουν πρωταγωνιστικό ρόλο σε πολλούς και διαφορετικούς τομείς της Μηχανικής Μάθησης (Αναγνώριση Αντικειμένων, Αναγνώριση Φωνής, Αναγνώριση Φυσικής Γλώσσας), πρέπει να σημειωθεί πως πολλοί κριτικάρουν το κόστος εκπαίδευσης και τους εξωγενείς παράγοντες αυτών των μοντέλων, επιλέγοντας ακόμη τα SVM λόγω απλότητας.

1.1.1 Ταξινόμηση Εικόνων (Image Classification)

Η Ταξινόμηση Εικόνων (Image Classification) είναι ένα ειδικό πρόβλημα Μηχανικής Μάθησης και μέρος της Επιβλεπόμενης Μάθησης. Πιο συγκεκριμένα, ορίζεται ένα σύνολο αντικειμένων τα οποία θέλουμε να αναγνωρίσουμε και εκπαιδεύουμε το μοντέλο ώστε να αναγνωρίζει νέες εικόνες βάσει του αρχικού συνόλου. Τα πρώτα μοντέλα που δημιουργήθηκαν με τεχνικές Όρασης Υπολογιστών (Computer Vision) βασίζονταν καθαρά στα δεδομένα των pixels. Ωστόσο, τα δεδομένα των pixels δεν έδιναν αρκετή σταθερότητα στα αποτελέσματα για να μπορεί να προβλεφθεί η ποικιλία των απεικονίσεων των αντικειμένων στις διάφορες εικόνες.

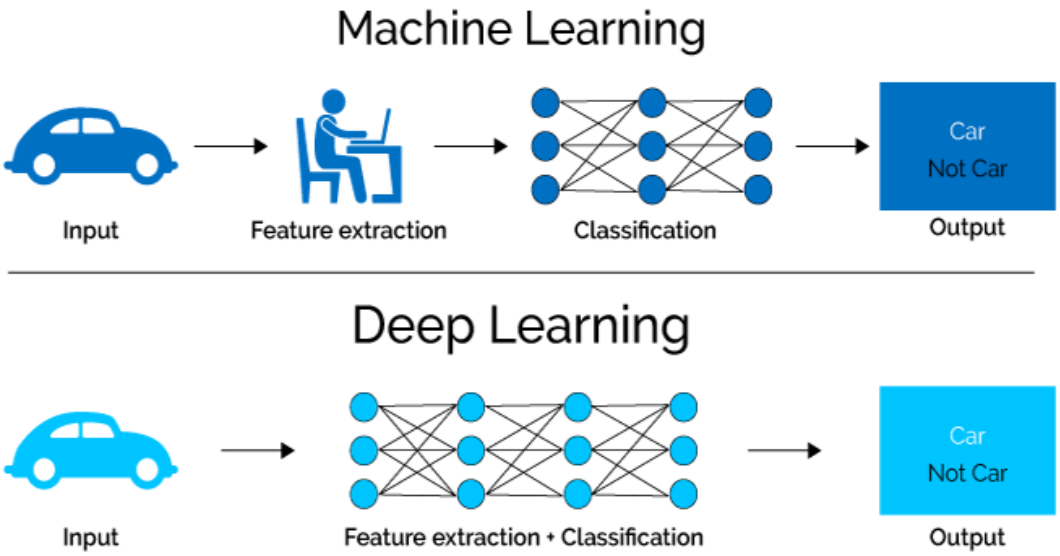
Όραση Υπολογιστών (Computer Vision)

Όραση Υπολογιστών ορίζεται ως ο τομέας που αναζητά την ανάπτυξη τεχνικών, ώστε να βοηθήσει τους υπολογιστές να «δουν» και να καταλάβουν το περιεχόμενο ψηφιακών εικόνων, όπως φωτογραφίες και βίντεο [10].
Ο τομέας αυτός, μέχρι και την έλευση των δικτύων Βαθιάς Μάθησης και συγκεκριμένα των Συνελικτικών Νευρωνικών Δικτύων (Convolutional Neural Networks – CNNs ή ConvNets), έπαιζε πρωταρχικό ρόλο στην ταξινόμηση εικόνων κάνοντας χρήση των τεχνικών Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), Hough transforms, Geometric hashing, κ.α.
Οι παραπάνω τεχνικές βασίζονταν στον εντοπισμό συγκεκριμένων «σημείων» στις εικόνες όπως ακμές (edge detection), γωνίες (corner detection) και γραμμές (line detection) [11].

Πρώιμες τεχνικές βασίζονταν σε εκτεταμένες προσπάθειες των ερευνητών να σχεδιάσουν κανόνες ταξινόμησης που θα εντοπίζουν συγκεκριμένες ομάδες συνδυασμού pixels. Παρατηρήθηκε πως χαρακτηριστικά όπως η θέση του αντικειμένου, το παρασκήνιο πίσω από το αντικείμενο, ο φωτισμός του χώρου, η γωνία και η εστίαση της κάμερας προκαλούσαν αρκετά μεγάλες διαφοροποιήσεις στα δεδομένα. Έτσι, οι εικόνες δεν μπορούσαν να κατηγοριοποιηθούν κάνοντας χρήση βαρών στις RGB τιμές των pixels.
Για την επίλυση του προβλήματος τα μοντέλα αυτά άρχισαν να εξάγουν πληροφορίες από ιστογράμματα χρωμάτων, την υφή και τα σχήματα. Τελικώς, αποδείχθηκε πως αυτές οι συμπαγείς τεχνικές ήταν δύσκολες και χρονοβόρες σε περιπτώσεις αλλαγών για κάθε νέα εφαρμογή ή αντικείμενο που χρειάζεται να ταξινομηθεί [13].
Τα παραπάνω προβλήματα λύθηκαν με την έλευση της Μηχανικής Μάθησης, που σταμάτησε να βασίζεται στις παραδοσιακές τεχνικές Όρασης Υπολογιστών εισάγοντας τη Βαθιά Μάθηση στην εξίσωση και αυξάνοντας την αποτελεσματικότητα των ήδη υπαρχόντων αλγορίθμων σε προβλήματα που εμφανίζονται στον «πραγματικό κόσμο».

1.1.2 Βαθιά Μάθηση

Η Βαθιά Μάθηση (Deep Learning) είναι ένα υποσύνολο της Μηχανικής Μάθησης (Machine Learning), που αυτή με τη σειρά της είναι υποσύνολο της Τεχνητής Νοημοσύνης (Artificial Intelligence).
Η Τεχνητή Νοημοσύνη είναι ένας γενικός όρος που αναφέρεται σε τεχνικές που επιτρέπουν στους υπολογιστές να μιμούνται την ανθρώπινη συμπεριφορά. Η Μηχανική Μάθηση αντιπροσωπεύει ένα σύνολο αλγορίθμων που έχουν εκπαιδευτεί σε δεδομένα που καθιστούν όλα αυτά δυνατά.
Συμπληρώνοντας, η Βαθιά Μάθηση, είναι εμπνευσμένη από τη δομή του ανθρώπινου εγκεφάλου. Οι αλγόριθμοι Βαθιάς Μάθησης προσπαθούν να εξαγάγουν παρόμοια συμπεράσματα όπως οι άνθρωποι, αναλύοντας συνεχώς τα δεδομένα με μια δεδομένη λογική δομή. Για να επιτευχθεί αυτό, η Βαθιά Μάθηση χρησιμοποιεί μια πολυεπίπεδη δομή αλγορίθμων που ονομάζονται νευρωνικά δίκτυα.
Σημαντική διαφορά μεταξύ Μηχανικής Μάθησης και Βαθιάς Μάθησης είναι πως στη Μηχανική Μάθηση πρέπει ο κατασκευαστής του μοντέλου να εξάγει πληροφορίες για τα δεδομένα εισόδου πριν εισαχθούν στο νευρωνικό δίκτυο, ενώ στην Βαθιά Μάθηση τα δεδομένα εισάγονται ακατέργαστα και τα δίκτυα εξάγουν από μόνα τους ότι χρειάζονται για να πετύχουν τον σκοπό τους (Εικόνα 1.5).



Εικόνα 1.5: Διαφορά λειτουργίας Machine Learning σε σχέση με το Deep Learning

Πολλές εφαρμογές κέρδισαν από την εξέλιξη της Βαθιάς Μάθησης και των νευρωνικών δικτύων βαθιάς μάθησης (Deep Neural Networks - DNNs), από τομείς πολυμέσων μέχρι και τον χώρο της ιατρικής.

Πιο συγκεκριμένα τα DNNs δείχνουν να φέρνουν αλλαγή στους ακόλουθους χώρους:

- **Βίντεο και Εικόνα**: Τα βίντεο περιλαμβάνουν το μεγαλύτερο μέρος των big data, καθώς αποτελούν το 70% του σημερινού internet. Για παράδειγμα, πάνω από 800 εκατομμύρια ώρες βίντεο συλλέγονται καθημερινά για βίντεο ασφάλειας και μπορούν πολύ εύκολα να χρησιμοποιηθούν στη Βαθιά Μάθηση. Τα DNNs έχουν βελτιώσει σημαντικά την ακρίβεια πολλών διεργασιών όπως είναι η ταξινόμηση εικόνων, ο εντοπισμός αντικειμένων, η κατάτμηση εικόνων και η αναγνώριση κινήσεων [12].

- **Ομιλία και Γλώσσα**: Τα DNNs έχουν βελτιώσει σημαντικά την ακρίβεια της αναγνώρισης φωνής αλλά και πολλών σχετικών εργασιών όπως είναι η μετάφραση (machine translation), η επεξεργασία φυσικής γλώσσας (natural language processing) και η δημιουργία ήχου (audio generation) [12].

- **Ιατρική**: Τα DNNs έχουν παίξει σημαντικό ρόλο γενικά στην γονιδιωματική και συγκεκριμένα στην αναγνώριση γονιδιωμάτων ασθενειών όπως ο αυτισμός, ο καρκίνος και η ασθένεια μυϊκής ατροφίας της σπονδυλικής στήλης. Ακόμη, χρησιμοποιούνται στην ιατρική απεικόνιση για τον εντοπισμό του καρκίνου του δέρματος και τύπους καρκίνων στον εγκέφαλο και στο στήθος [12].

- **Παιχνίδια**: Τα DNNs έχουν βρει μεγάλη χρήση και σε παιχνίδια. Η επιτυχία οφείλεται σε καινοτομίες και νέες τεχνικές εκπαίδευσης των μοντέλων, με τη βοήθεια των αλγορίθμων Ενισχυτικής Μάθησης (Deep Reinforcement Learning) [12].

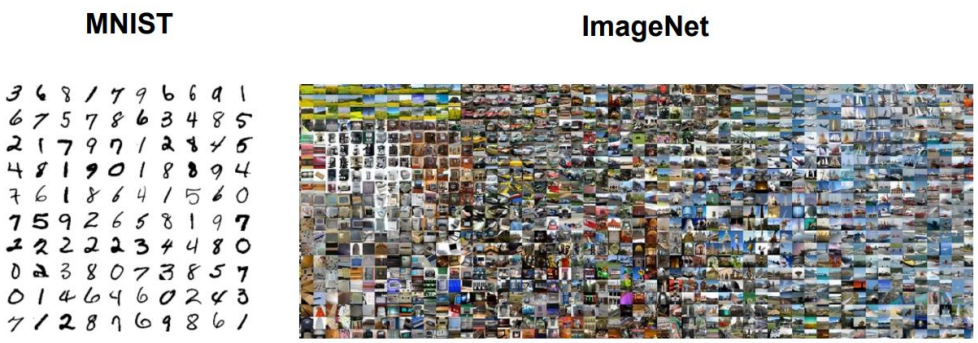
- **Ρομποτική**: Τα DNNs δείχνουν να είναι πολύ αποτελεσματικά στον τομέα της Ρομποτικής, σε διαδικασίες όπως είναι το κλείσιμο παλάμης ρομποτικού χεριού, ο σχεδιασμός κίνησης για ρομπότ εδάφους, η εικονική πλοήγηση, ο έλεγχος - σταθεροποίηση ελικοπτέρων και η οδήγηση αυτόνομων οχημάτων [12].

Σύνολα Δεδομένων

Τα δίκτυα Βαθιάς Μάθησης απαιτούν σύνολα δεδομένων για το στάδιο της εκπαίδευσης των δοθέντων DNNs. Καθώς υπάρχουν διαφορετικά μοντέλα για κάθε σκοπό, έχουν δημιουργηθεί και τα αντίστοιχα σύνολα δεδομένων. Τα πιο γνωστά σύνολα για ταξινόμηση εικόνων είναι τα ακόλουθα:

- Το **MNIST** (Εικόνα 1.6) είναι ένα σύνολο δεδομένων που χρησιμοποιείται ευρέως για ταξινόμηση ψηφίων. Αποτελείται από εικόνες χειρόγραφων ψηφίων 28x28 pixels. Υπάρχουν 10 κλάσεις, 60.000 εικόνες εκπαίδευσης και 10.000 εικόνες αξιολόγησης. Σήμερα, τα state-of-the-art μοντέλα έχουν φτάσει ακρίβεια 99.84% [12].

- Το **CIFAR** είναι ένα σύνολο δεδομένων που αποτελείται από έγχρωμες εικόνες 32x32 pixels διάφορων αντικειμένων που κυκλοφόρησε το 2009. Το CIFAR είναι ένα υποσύνολο ενός dataset 80 εκατομμυρίων εικόνων. Το CIFAR – 10 αποτελείται από 10 κλάσεις. Σε αυτό υπάρχουν 50.000 εικόνες εκπαίδευσης (5.000 ανά κλάση) και 10.000 εικόνες αξιολόγησης. Το μοντέλο Restricted Boltzman Machine έφτασε ακρίβεια 64.84%, με την θέσπιση του CIFAR – 10. Αργότερα, η ακρίβεια έφτασε στο 96.53% χρησιμοποιώντας τον αλγόριθμο του κλασματικού max pooling [12]. Σήμερα το ποσοστό ακρίβειας έχει φτάσει το 99.7% με την τεχνική SAM (Sharpness-Aware Minimization)



Εικόνα 1.6: (Αριστερά) Απεικόνιση MNIST, (Δεξιά) Απεικόνιση ImageNet

- Το **ImageNet** (Εικόνα 1.6) είναι ένα μεγάλο σύνολο δεδομένων εικόνων που παρουσιάστηκε πρώτη φορά το 2010 και σταθεροποιήθηκε το 2012. Περιλαμβάνει έγχρωμες εικόνες 256x256 pixels σε 1000 κλάσεις. Οι κλάσεις έχουν οριστεί με το WordNet για την διαχείριση δυσνόητων λέξεων και συνδυάζοντας τα συνώνυμα στην ίδια κατηγορία. Οι 1000 κλάσεις επιλέχθηκαν με τέτοιον τρόπο ώστε να μην υπάρχει επικάλυψη των κατηγοριών του ImageNet. Για την αξιολόγηση μοντέλων στο ImageNet χρησιμοποιούνται συνήθως δύο μετρικές: η Top-1 και η Top-5 ακρίβεια. Με βάση την Top-1 ακρίβεια, δεχόμαστε ότι το μοντέλο προέβλεψε σωστά αν η κλάση με την υψηλότερη πεποίθηση είναι και η σωστή. Αντίστοιχα, για την Top-5 ακρίβεια, δεχόμαστε ότι το μοντέλο προέβλεψε σωστά αν η σωστή κλάση βρίσκεται μέσα στις 5 με τις υψηλότερες πεποιθήσεις. Στον πρώτο σημαντικό διαγωνισμό του ImageNet το 2012, μοντέλο - νικητής αναδείχθηκε το AlexNet με Top-1 ακρίβεια 61.9%. Από τότε κάθε χρόνο η ακρίβεια αυξάνεται. Σήμερα, τα EfficientNets έχουν κατακτήσει τις πρώτες θέσεις στην κατάταξη, με το state-of-the-art να φτάνει Top-1 ακρίβεια 90.2% και Top-5 ακρίβεια 98.8% [12].

Στάδια Βαθιάς Μάθησης

Η Βαθιά Μάθηση αποτελείται από τρία βασικά στάδια: την εκπαίδευση του δικτύου (training), την αξιολόγηση του δικτύου (evaluation) και τη συμπερασματολογία (inference).

Παίρνοντας ως παράδειγμα ένα DNN ταξινόμησης εικόνων:

Κατά την εκπαίδευση (training), το μοντέλο κάνει χρήση ενός dataset και ο στόχος του είναι η δημιουργία βαρών, ώστε να επιτευχθεί το μέγιστο σκορ της σωστής κατηγορίας (που έχει δοθεί από δεδομένα του dataset) και το ελάχιστο σκορ στις υπόλοιπες λανθασμένες κατηγορίες.
Κατά το στάδιο της αξιολόγησης (evaluation), δίνεται μια εικόνα εισόδου (κάνοντας χρήση αντίστοιχου dataset) στο μοντέλο και εξάγονται οι κατάλληλοι πίνακες με τα σκορ κάθε κατηγορίας. Αξιολογώντας τα αποτελέσματα για το σκορ των πρώτων 5 κατηγοριών ή του σκορ 1 κατηγορίας, εξάγονται και τα ανάλογα ποσοστά για την αξιολόγηση του συστήματος (μετρικές Top-1 και Top-5).
Κατά το στάδιο της συμπερασματολογίας (inference), εισάγεται μια νέα άγνωστη εικόνα και κάνοντας χρήση των βαρών του ήδη εκπαιδευμένου μοντέλου εξάγεται μια απάντηση για την/τις κατηγορίες που ανήκει η εικόνα [12].

Για την καλύτερη κατανόηση της Βαθιάς Μάθησης και των εφαρμογών της παρατηρούμε κάποια δημοφιλή μοντέλα.

1.1.3 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs):

Καθώς τα μοντέλα που χρησιμοποιούνται για ταξινόμηση εικόνων στο AI Gallery App

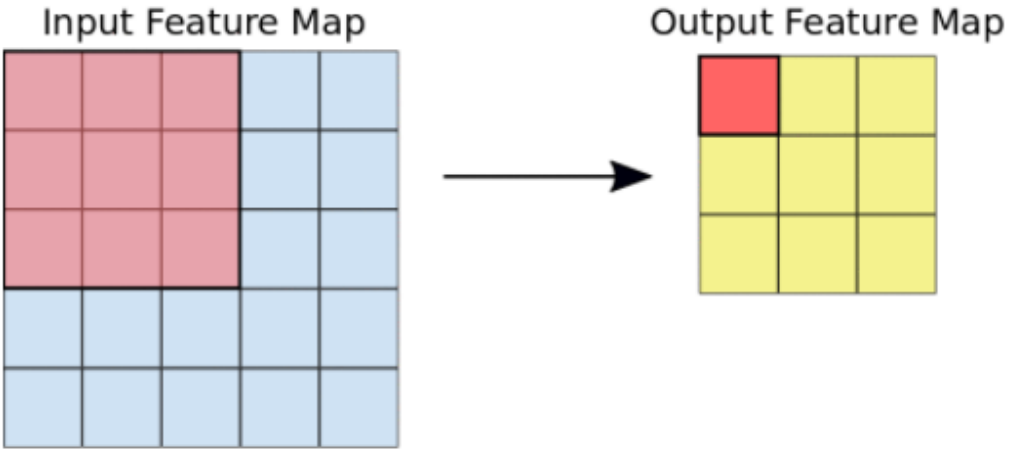
έχουν δημιουργηθεί με βάση τα CNNs, στη συνέχεια γίνεται μια περαιτέρω ανάλυση αυτών. Ξεκινώντας, το CNN λαμβάνει ένα χάρτη χαρακτηριστικών στην είσοδο: Έναν τρισδιάστατο πίνακα που το μέγεθος των δύο πρώτων διαστάσεων αντιστοιχεί στο μήκος και πλάτος των pixel της εικόνας. Το μέγεθος της τρίτης διάστασης είναι 3 (αντιστοιχώντας στα τρία κανάλια χρώματος). Τα CNNs αποτελούνται από πολλαπλά κομμάτια, που το καθένα απαρτίζεται από τρείς διαδικασίες (Επίπεδο Συνέλιξης, Επίπεδο Μη-Γραμμικότητας, Επίπεδο Συγκέντρωσης).

1. Επίπεδο Συνέλιξης

Το Επίπεδο Συνέλιξης (Convolutional Layer) εξάγει τετράγωνα από το χάρτη χαρακτηριστικών εισόδου, και εφαρμόζει φίλτρα, παράγοντας έναν χάρτη χαρακτηριστικών εξόδου (Εικόνα 1.7).
Οι συνελίξεις ορίζονται από δύο παράγοντες:

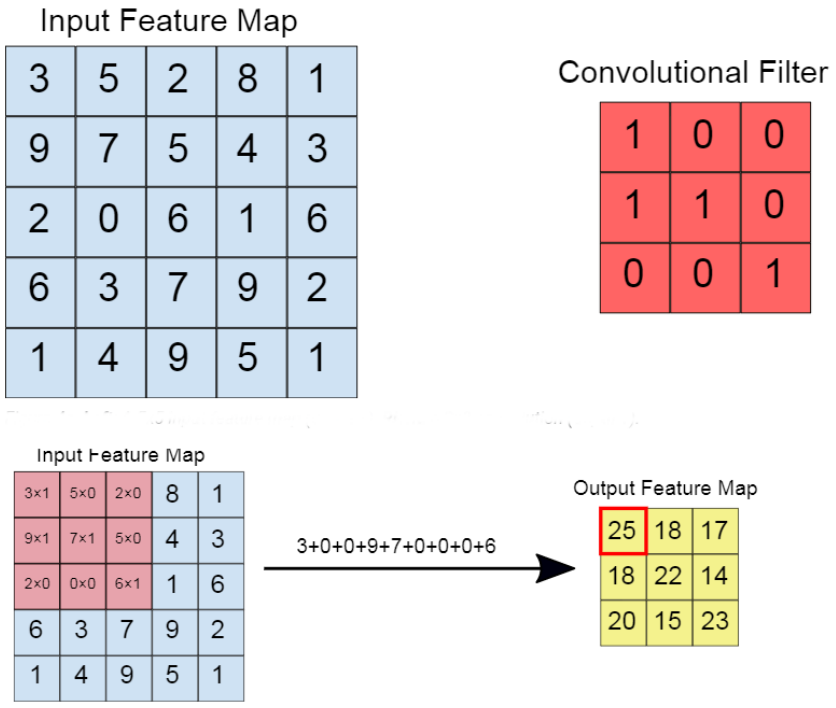
- Το μέγεθος των τετραγώνων που εξάγονται.
- Το βάθος του χάρτη χαρακτηριστικών εξόδου, που αντιστοιχεί στον αριθμό των φίλτρων που εφαρμόζονται.

Στο συνελικτικό επίπεδο, τα φίλτρα ελέγχουν όλα τα τετράγωνα του χάρτη εισόδου οριζοντίως και καθέτως, εξάγοντας ένα τελικό τετράγωνο.



Εικόνα 1.7: Χάρτης χαρακτηριστικών εισόδου που καταλήγει σε Χάρτη χαρακτηριστικών εξόδου (Στάδιο Συνέλιξης)

Αναλυτικότερα, για κάθε ζευγάρι τετραγώνου - φίλτρου, το CNN πολλαπλασιάζει τις τιμές μεταξύ φίλτρου και πίνακα εισόδου, αθροίζοντας όλα τα στοιχεία του νέου πίνακα (Εικόνα 1.8). Κάθε μία από αυτές τις τιμές είναι η έξοδος του πίνακα χαρακτηριστικών συνέλιξης.



Εικόνα 1.8 Είσοδος πριν την εφαρμογή φίλτρου (Πάνω), Έξοδος μετά την εφαρμογή φίλτρου (Κάτω)

Κατά τη διάρκεια της εκπαίδευσης, το CNN «μαθαίνει» τις βέλτιστες τιμές για τους πίνακες φίλτρων και τους ενεργοποιεί για την εξαγωγή σημαντικών χαρακτηριστικών (υφών, ακμών, σχημάτων) από τον χάρτη χαρακτηριστικών εισόδου (Input Feature Map). Όσο ο αριθμός των φίλτρων που εφαρμόζονται στην είσοδο αυξάνεται, τόσο αυξάνεται και ο αριθμός των χαρακτηριστικών που μπορεί να εξάγει το CNN. Βέβαια, ως μειονέκτημα, τα φίλτρα συνέλιξης αποτελούν την πλειοψηφία των πόρων που ξοδεύονται από τα CNN, έτσι ο χρόνος εκπαίδευσης αυξάνεται με την προσθήκη περισσότερων φίλτρων. Ακόμη, κάθε φίλτρο που προστίθεται στο δίκτυο παρέχει μικρότερη αξία από το προηγούμενο. Έτσι, οι ερευνητές προσπαθούν να δημιουργήσουν δίκτυα που χρησιμοποιούν το μικρότερο αριθμό φίλτρων για την ανάλογη ακρίβεια χαρακτηριστικών κατά την ταξινόμηση εικόνων.

2. Επίπεδο Μη-Γραμμικότητας

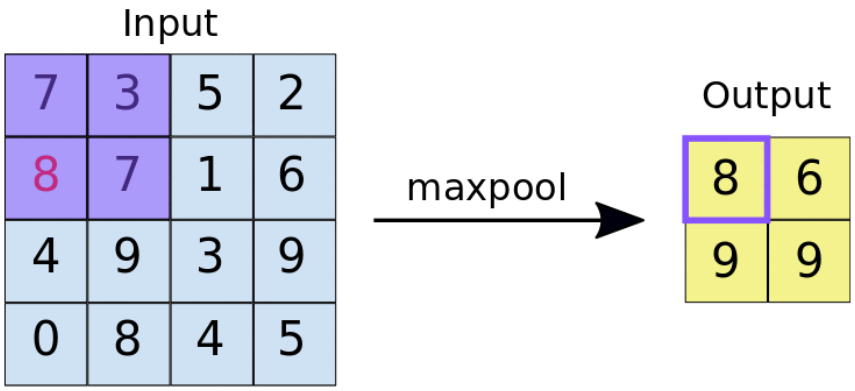
Ακολουθώντας κάθε πράξη συνέλιξης, το CNN εφαρμόζει έναν Rectified Linear Unit (ReLU) μετασχηματισμό στα χαρακτηριστικά του σταδίου συνέλιξης, έτσι ώστε να εισαχθεί η μη – γραμμικότητα στο μοντέλο. Η συνάρτηση ReLU F(x) = max (0, x), επιστρέφει x για όλες τις τιμές x>0, και 0 για τιμές $x \leq 0$.

3. Επίπεδο Συγκέντρωσης

Έπειτα από το ReLU, ακολουθεί το επίπεδο συγκέντρωσης (pooling), όπου το CNN μειώνει τα χαρακτηριστικά συνέλιξης για κέρδος χρόνου. Για να συμβεί αυτό, μειώνει τον αριθμό διαστάσεων του χάρτη χαρακτηριστικών, διατηρώντας τα σημαντικότερα χαρακτηριστικά αυτών. Ένας αλγόριθμος που χρησιμοποιείται για την επίτευξη της μείωσης είναι ο max

pooling.
Το max pooling λειτουργεί με παρόμοιο τρόπο με το επίπεδο συνέλιξης (Εικόνα 1.9). Ο χάρτης χαρακτηριστικών ελέγχεται με ένα φίλτρο για όλο τον χάρτη εισόδου. Με κάθε έλεγχο, η μέγιστη τιμή εισάγεται στο νέο χάρτη χαρακτηριστικών.
Το max pooling χαρακτηρίζεται από δύο παραμέτρους:

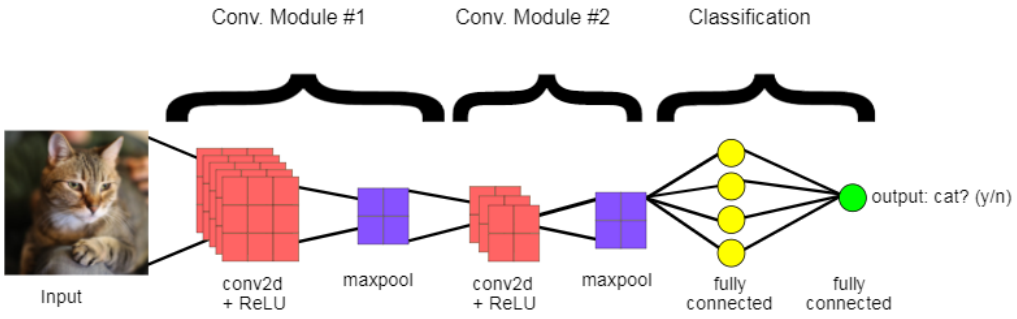
- Το **μέγεθος** του max pooling φίλτρου (συνήθως 2x2 pixels)
- Το **βήμα**, που είναι η απόσταση σε pixels του βήματος που κινείται το φίλτρο στο χάρτη εισόδου.



Εικόνα 1.9: Εφαρμογή max pooling στο χάρτη εισόδου

4. Πλήρως Συνδεδεμένο Επίπεδο

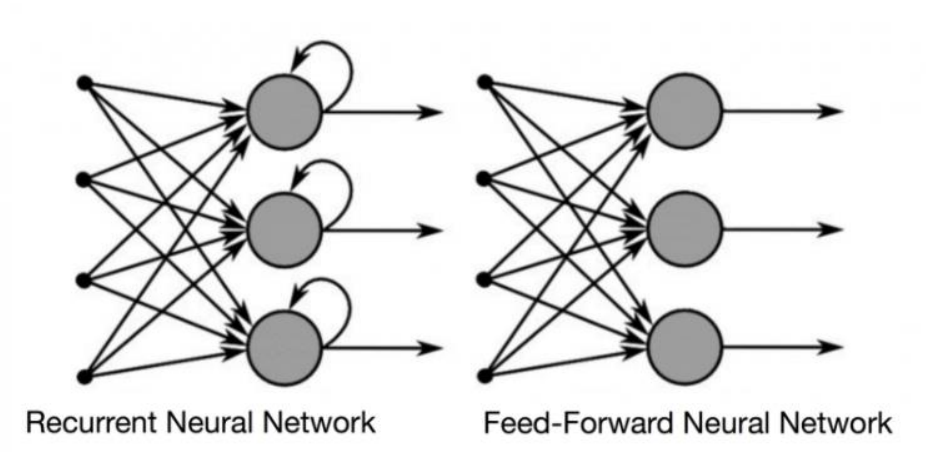
Στο τέλος ενός συνελικτικού νευρωνικού δικτύου υπάρχουν ένα ή περισσότερα πλήρως συνδεδεμένα επίπεδα. Όταν δύο στρώματα είναι "πλήρως συνδεδεμένα", κάθε κόμβος στο πρώτο στρώμα συνδέεται με κάθε κόμβο στο δεύτερο στρώμα. Η δουλειά τους είναι να πραγματοποιούν ταξινόμηση με βάση τα χαρακτηριστικά που εξήχθησαν από τις συνελίξεις (Εικόνα 1.10).
Συνήθως, το τελευταίο πλήρως συνδεδεμένο επίπεδο (fully connected layer) περιέχει μια συνάρτηση ενεργοποίησης softmax, η οποία εξάγει τιμή πιθανότητας από 0 έως 1 για καθεμία από τις ετικέτες ταξινόμησης που το μοντέλο προσπαθεί να προβλέψει [13].



Εικόνα 1.10: Μοντέλο Πλήρως Συνδεδεμένου Επιπέδου

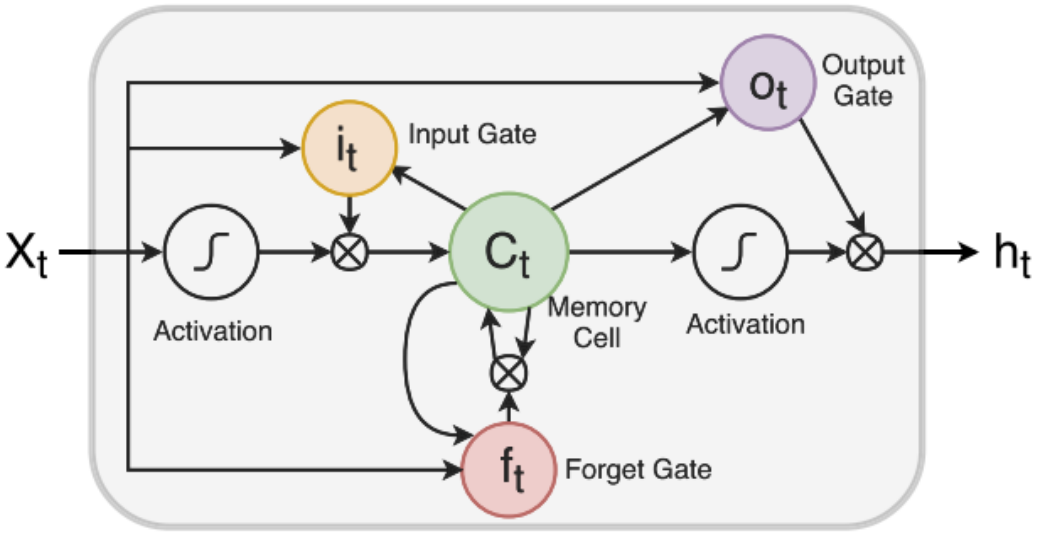
Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Network - RNN):

Τα RNNs (Εικόνα 1.11 - Αριστερά) αναπτύχθηκαν για την επίλυση του χρονικού προβλήματος που είχαν τα Feed Forward neural networks (FFNN). Στα FFNNs (Εικόνα 1.11 - Δεξιά) η πληροφορία κυκλοφορεί από το επίπεδο εισόδου μέχρι την έξοδο χωρίς να επισκεφθεί ξανά περασμένο κόμβο [14].



Εικόνα 1.11: Recurrent Neural Network - RNN (Αριστερά), Feed Forward Neural Network -FFNN (Δεξιά)

Κάθε νευρώνας του RNN έχει μια εσωτερική μνήμη που κρατάει πληροφορίες των υπολογισμών των περασμένων παραδειγμάτων. Η εκπαίδευση των RNN βασίζεται στον αλγόριθμο backpropagation.
Η Long Short – Term Memory (LSTM) είναι μια εκτεταμένη μορφή των RNNs.
Στα LSTM (Εικόνα 1.12), η πύλη χρησιμοποιείται για την εκπροσώπηση της βασικής μονάδας του νευρώνα. Κάθε νευρώνας του LSTM καλείται κύτταρο μνήμης (memory cell) και συμπεριλαμβάνει μια πύλη πολλαπλασιασμού forget gate. Αυτές οι πύλες χρησιμοποιούνται για τον έλεγχο της πρόσβασης στα memory cells και για την πρόληψη από διαταραχή άσχετων εισόδων [14].

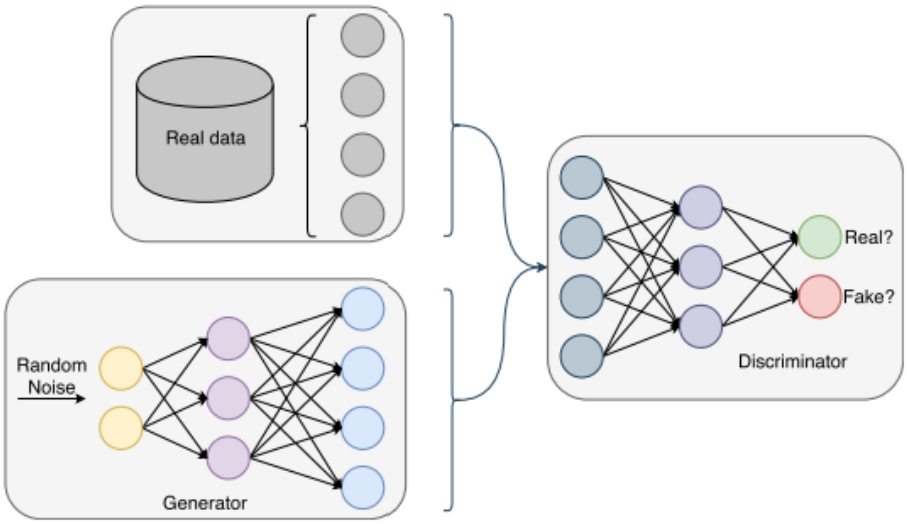


Εικόνα 1.12: Μοντέλο Long – Short Term Memory (LSTM)

Πληροφορίες προστίθενται ή αφαιρούνται μέσω της πύλης στο memory cell. Οι πύλες (input, output, forget) είναι διαφορετικά νευρωνικά δίκτυα που καθορίζουν ποιες πληροφορίες επιτρέπονται στο memory cell. Η forget gate μπορεί να μάθει ποιες πληροφορίες διατηρούνται ή ξεχνιούνται κατά τη διάρκεια της εκπαίδευσης.
Τα RNNs χρησιμοποιούνται συχνά σε τομείς όπως είναι η επεξεργασία φυσικής γλώσσας, η μοντελοποίηση, η πρόβλεψη λέξεων και η αυτόματη μετάφραση [14].

Γεννητικά Ανταγωνιστικά Δίκτυα (Generative Adversarial Networks - GANs)

Τα Γεννητικά Ανταγωνιστικά Δίκτυα (Εικόνα 1.13) αποτελούνται από δύο σημαντικά δίκτυα, τα δίκτυα γεννήτριας και διάκρισης (generator και discriminator). Ο generator είναι υπεύθυνος για την δημιουργία νέων δεδομένων βάσει συνόλου κατανεμημένων δεδομένων ή πραγματικών δεδομένων. Ο discriminator είναι υπεύθυνος για την διάκριση αληθινών δεδομένων από τα «πλαστά» δεδομένα που δημιουργεί ο generator. Στην ουσία τα δύο αυτά δίκτυα μπορούν να χαρακτηριστούν ως πλαστογράφος και αστυνόμος, με τον πλαστογράφο να φτιάχνει ψεύτικα δεδομένα προσπαθώντας να φτάσει στο σημείο να φαίνονται τόσο αληθινά, που θα ξεγελάνε τον αστυνόμο [14].

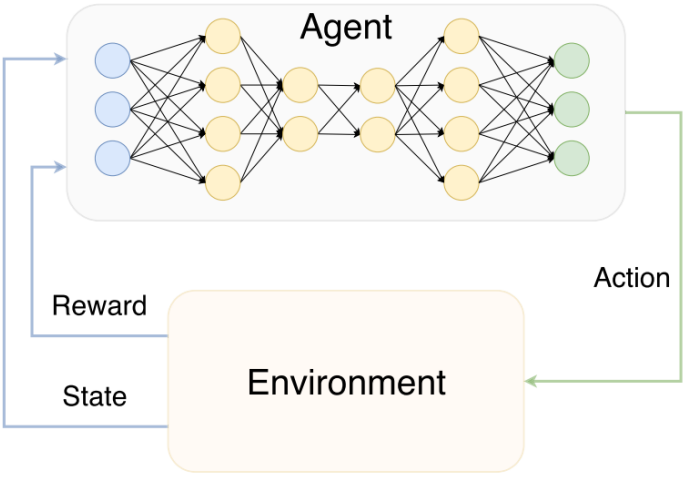


Εικόνα 1.13: Μοντέλο Generative Adversarial Network - GAN

Τα GANs χρησιμοποιούνται ευρέως στη παραγωγή, μετατροπή, σύνθεση και υπερ – ανάλυση εικόνων.

Βαθιά Ενισχυτική Μάθηση (Deep Reinforcement Learning - DRL)

Η Βαθιά Ενισχυτική Μάθηση (Εικόνα 1.14) αποτελείται από DNNs και εμπίπτει στην κατηγορία της Ενισχυτικής Μάθησης. Ο στόχος των DRLs είναι η δημιουργία ευφυών πρακτόρων (agents) που μπορούν να εφαρμόζουν πολιτικές που μεγιστοποιούν τις ανταμοιβές σε διαδικασίες με πολλά στάδια. Στην DRL προσέγγιση, η ενισχυτική μάθηση αναζητά τις καλύτερες κινήσεις σε καταστάσεις περιβάλλοντος, με το DNN να αναλαμβάνει τον έλεγχο μεγάλου αριθμού καταστάσεων αξιολογώντας τις τιμές δράσης, και επιλέγοντας την δράση με το μέγιστο βαθμό αξιολόγησης.



Εικόνα 1.14: Μοντέλο Deep Reinforcement Learning

Οι εφαρμογές που εφαρμόζονται συνήθως τα DRLs είναι η επίλυση προβλημάτων προγραμματισμού (scheduling), θέματα αποφάσεων σε παιχνίδια και η αξιολόγηση μετάδοσης ροής βίντεο [14].

1.2 Μηχανική Μάθηση στις Κινητές Συσκευές

Οι κινητές συσκευές είναι ένας αναπτυσσόμενος χώρος για οποιαδήποτε τεχνολογία. Νέα μοντέλα συσκευών κυκλοφορούν συνεχώς, αναβαθμίζοντας τις δυνατότητες επεξεργασίας δεδομένων και απεικόνισης γραφικών. Το γεγονός ότι έχουν γίνει μέρος της καθημερινότητας κάθε χρήστη και εξάγεται μεγάλος όγκος δεδομένων από αυτές, δημιουργεί συζήτηση ένταξης όλης της διαδικασίας Βαθιάς Μάθησης (εκπαίδευση, αξιολόγηση, συμπερασματολογία) σε αυτές.
Η εκπαίδευση νευρωνικών δικτύων Βαθιάς Μάθησης περιέχει εκατοντάδες εκατομμύρια παραμέτρους και παρουσιάζει τεράστιες απαιτήσεις πόρων. Έτσι, είναι αρκετά μακριά από τις δυνατότητες που έχουν οι κινητές συσκευές. Αυτή τη στιγμή οι έρευνες επικεντρώνονται στο τρόπο που μπορούν να χρησιμοποιηθούν στη Βαθιά Μάθηση τα δεδομένα που παράγουν οι κινητές συσκευές εκτός αυτής με ασφάλεια, εμπιστευτικότητα και ιδιωτικότητα. Για την έμμεση εμπλοκή των κινητών συσκευών στην εκπαίδευση, γίνονται προσπάθειες στους τομείς:

Κατανεμημένη Εκπαίδευση (Distributed Training): Εκπαίδευση των μοντέλων εκτός συσκευής. Η εκπαίδευση γίνεται στο νέφος (cloud) ή στην άκρη του δικτύου (edge).

Διαμοιραζόμενη Εκπαίδευση (Federated Training): Επιτρέπει την εκπαίδευση ενός κοινόχρηστου μοντέλου μέσω διαφορετικών συσκευών τοπικά.

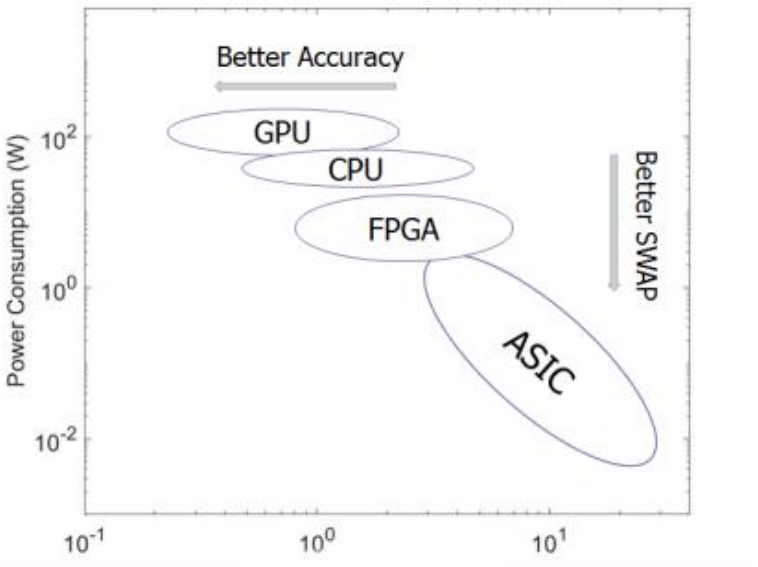
Εκπαίδευση Διατήρησης Ασφάλειας (Privacy – Preserving Training): Τεχνικές ασφάλειας που βελτιώνουν την εκπαίδευση του Federated Training [15].

Η συμπερασματολογία είναι μια πιο ελαφριά διαδικασία σε σχέση με την εκπαίδευση, καθώς κάνει χρήση ενός ήδη εκπαιδευμένου μοντέλου. Αν και είναι δυνατή η εφαρμογή της συμπερασματολογίας στις κινητές συσκευές, η διαδικασία δεν παύει να είναι ενεργοβόρα και να απαιτεί πόρους της συσκευής. Για αυτό το λόγο υπάρχουν δύο επιλογές για την εκτέλεση ενός εκπαιδευμένου μοντέλου: συμπερασματολογία απομακρυσμένα (cloud, edge) ή συμπερασματολογία τοπικά στη συσκευή.

Ενσωμάτωση της Βαθιάς Μάθησης στις Κινητές Συσκευές

Η συμπερασματολογία στην τοπική συσκευή προσπαθεί να εκμεταλλευτεί οποιοδήποτε πηγή υπολογισμού για την εκτέλεση της συμπερασματολογίας. Οι GPUs είναι σημαντικοί αρωγοί στην επίτευξη αυτού του σκοπού καθώς μπορούν να επιτύχουν πολύ υψηλή απόδοση. Βέβαια, οι GPUs μπορεί να δαπανήσουν αρκετή ενέργεια. Τα ASICs είναι μια

αρκετά καλύτερη επιλογή ενεργειακά, καθώς το hardware έχει δημιουργηθεί ειδικά για συγκεκριμένους υπολογισμούς. Βέβαια, ο σχεδιασμός τους και η ανάπτυξή τους είναι πολύ χρονοβόρες διαδικασίες. Έτσι τα ASICs χρησιμοποιούνται όταν ο αλγόριθμος έχει βρεθεί και το σύστημα έχει πολύ χαμηλό κόστος ενέργειας. Από την άλλη, τα FPGAs δείχνουν να προσφέρουν μια καλύτερη ταχύτητα ανάπτυξης συστήματος με τη σχέση των συστημάτων σε ενέργεια/ακρίβεια να φαίνεται παρακάτω (Εικόνα 1.15).



Εικόνα 1.15: Σύγκριση τεσσάρων αρχιτεκτονικών υλικού (GPA, CPU, FPGA, ASIC) (Κατανάλωση Ενέργειας/Ακρίβειας)

Στο κομμάτι της συμπερασματολογίας, τα μοντέλα Βαθιάς Μάθησης μπορούν να γίνουν επίσης πολύ απαιτητικά, αν και έχει γίνει πλέον εφικτή η εφαρμογή τους στις κινητές συσκευές, μέσω βιβλιοθηκών όπως το Tensorflow Lite της Google, το Caffe2 του Facebook και το Core ML της Apple [16].

Βελτιστοποίηση μοντέλων

Πέρα από τις μελέτες που παρατηρούνται στην αρχιτεκτονική του υλικού, βλέπουμε χρήσεις τεχνικών όπως pruning, quantization και Huffman coding για την μείωση του μεγέθους των μοντέλων χωρίς μεταβολή στην ακρίβειά τους.
Στο **pruning** αφαιρούνται όλες οι συνδέσεις μικρών βαρών μετά την εκπαίδευση των μοντέλων. Αν και χάνεται κάποια πληροφορία από το στάδιο της εκπαίδευσης, στην ουσία αφαιρούνται βάρη που δεν θα αξιοποιούνταν και ίσως να επιβράδυναν το μοντέλο. Συνήθως τα βάρη μπορούν να μειωθούν κατά 10% χρησιμοποιώντας pruning.
Το επόμενο βήμα είναι το **quantization**. Σε αυτό το βήμα ομαδοποιούνται όλα τα βάρη με παρόμοιες τιμές, και γίνεται η αποθήκευση τους σε ένα αρχείο. Το βήμα αυτό όχι μόνο μειώνει το μέγεθος των μοντέλων, αλλά μειώνει και την καθυστέρηση απόκρισης, αν και δέχεται μια μικρή υποβάθμιση στην ακρίβεια.
Το τελευταίο βήμα εφαρμόζει την τεχνική του **Huffman**, κατά την οποία συχνά

εμφανιζόμενα σύμβολα αποτυπώνονται με μικρότερα κομμάτια κώδικα.
Τέλος, για τις εφαρμογές σε κινητές συσκευές, συνηθίζεται η μείωση της ακρίβειας από 32 σε 16 bits για τα μοντέλα Βαθιάς Μάθησης [16].

1.2.1 Server στο Νέφος (Cloud) ή στην Άκρη του Δικτύου (Edge);

Server στο Νέφος (Cloud)

Η εφαρμογή της Μηχανικής Μάθησης στο **νέφος (cloud)** είναι η κυρίαρχη, τυπική μέθοδος σήμερα. Οι υπηρεσίες που προσφέρονται από μεγάλους παρόχους πλατφόρμας cloud επιτρέπουν στους επιστήμονες δεδομένων και τους προγραμματιστές να κατασκευάζουν, εκπαιδεύουν και αναπτύσσουν μοντέλα Μηχανικής Μάθησης γρήγορα και εύκολα. Η υποστήριξη στα πλαίσια Βαθιάς Μάθησης επιτρέπει ένα ανοιχτό, ευέλικτο περιβάλλον και η προετοιμασία και η χρήση δεδομένων απευθείας από υπηρεσίες αποθήκευσης δεδομένων που προσφέρονται από αντίστοιχο πάροχο cloud είναι εύκολη.
Ο πρωταρχικός περιορισμός αυτής της προσέγγισης είναι η πρόκληση της μετακίνησης των δεδομένων εισόδου από το σημείο που βρίσκονται σε ένα κέντρο δεδομένων cloud, ώστε να μπορούν να χρησιμοποιηθούν για την προετοιμασία και την ανάπτυξη μοντέλων. Η καθυστέρηση (latency), οι περιορισμοί εύρους ζώνης και τα ζητήματα κόστους καθιστούν συχνά ανέφικτη τη μεταφορά μεγάλου όγκου δεδομένων σε ένα απομακρυσμένο server στο νέφος (cloud) [17].

Server στην άκρη του δικτύου (Edge)

Με την αύξηση του όγκου δεδομένων και την αυξανόμενη ανάγκη για χρήση Βαθιάς Μάθησης σε πραγματικό ή σχεδόν πραγματικό χρόνο, καθίσταται ολοένα και πιο σημαντικό να αναλυθεί το ενδεχόμενο μετατόπισης πτυχών της διαδικασίας στην άκρη του δικτύου, αποκλειστικά ή και σε συνδυασμό με το νέφος.
Είναι εφικτή η ανάπτυξη και εκπαίδευση ενός αλγορίθμου μηχανικής μάθησης στο cloud και η εκτέλεσή του να γίνει στην ίδια τη συσκευή. Αυτή η μέθοδος διατηρεί την ευκολία και την ευελιξία της ανάπτυξης μοντέλων Μηχανικής Μάθησης στο cloud, αλλά εκτελεί τον αλγόριθμο κοντά στην πηγή δεδομένων.
Μια νεότερη προσέγγιση είναι η εκτέλεση Μηχανικής Μάθησης στην άκρη (edge) του δικτύου. Αυτό ενσωματώνει πόρους αιχμής που βρίσκονται κοντά - αλλά όχι μέσα - στη συσκευή και ξεπερνά τα προβλήματα ανάγκης μεταφοράς δεδομένων σε κεντρική τοποθεσία. Τα δεδομένα μπορούν να διανεμηθούν στην άκρη του δικτύου - αναπτύσσοντας υπολογιστές και χώρο αποθήκευσης σε πολύ κοντινή απόσταση από συσκευές που βρίσκονται σε μέρη όπως εργοστάσια, νοσοκομεία, εγκαταστάσεις πετρελαίου, τράπεζες και καταστήματα λιανικής.
Η επιλογή του τόπου ανάπτυξης και εκτέλεσης αλγορίθμων Μηχανικής Μάθησης είναι μια κρίσιμη απόφαση. Η τελική επιλογή θα επηρεάσει σε μεγάλο βαθμό τον τρόπο με τον οποίο οι εφαρμογές μπορούν να λάβουν αποφάσεις και πόσο γρήγορα μπορούν να ανταποκριθούν σε γεγονότα. Βέβαια, κάποια μειονεκτήματα που μπορούν να εντοπιστούν

στην edge εκτέλεση είναι πως υπάρχει κίνδυνος ασφάλειας λόγω του μεγάλου όγκου δεδομένων και το μεγάλο κόστος, καθώς είναι αναγκαία η δημιουργία προηγμένης δομής για την εκτέλεση και διαχείριση στην άκρη του δικτύου [17].

Συγκρίσεις με την τοπική εκτέλεση (on - device)

Ένα από τα επιχειρήματα εκτέλεσης του inference τοπικά είναι πως οι χρήστες μπορούν να κάνουν χρήση των μοντέλων χωρίς πρόσβαση στο internet. Επίσης, τα δεδομένα δεν θα χρειάζεται να μεταδοθούν εκτός συσκευής, σιγουρεύοντας το κομμάτι της ασφάλειας. Βέβαια, υπάρχουν και εδώ προβλήματα.
Τα μοντέλα σε συνδυασμό με την εφαρμογή θα αρχίσουν να γίνονται αρκετά μεγάλα σε μέγεθος, που θα κάνει δύσκολη την αναβάθμιση της εφαρμογής. Επίσης, εκτελώντας το inference τοπικά στην συσκευή, χρειάζεται μεγάλη υπολογιστική δύναμη, που γρήγορα θα απορροφήσει την περιορισμένη ενέργεια των κινητών συσκευών.

1.3 Αντικείμενο Διπλωματικής Εργασίας

Το αντικείμενο της εργασίας είναι η υλοποίηση μιας εφαρμογής AI Gallery App με υποβοήθηση από απομακρυσμένο εξυπηρετητή, που θα αναλαμβάνει το κομμάτι ταξινόμησης, και αποθήκευσης των εικόνων που εισάγονται μέσω της κάμερας ή του αποθηκευτικού χώρου.
Παρά την εξέλιξη σε hardware και software, κατά την on – device εκτέλεση, τα μοντέλα Βαθιάς Μάθησης απαιτούν μεγάλο ποσοστό πόρων της συσκευής και δεν είναι αξιόπιστα κατά το στάδιο της εκτέλεσης, λόγω της ανομοιογένειας που διακρίνει τις διαφορετικές κινητές συσκευές.
Η εργασία αποτελεί ένα μέρος της συλλογικής προσπάθειας μελέτης του ερωτήματος κατά πόσο μπορεί η Βαθιά Μάθηση να ενσωματωθεί σε κινητές συσκευές, εξετάζοντας τα πλεονεκτήματα – μειονεκτήματα, καθώς και τα διάφορα εμπόδια που εμφανίζονται κατά τη χρήση ενός cloud ή edge server.
Τέλος, γίνεται αξιολόγηση κάθε μοντέλου Βαθιάς Μάθησης που χρησιμοποιείται, μελετώντας μετρικές όπως η διεκπεραιωτική ικανότητα (throughput) και η χρονική καθυστέρηση (latency) εξάγοντας ανάλογα συμπεράσματα για την τελική χρήση της εφαρμογής που υλοποιήθηκε.

Κεφάλαιο 2: Εργαλεία – Βιβλιοθήκες

2.1 Client

2.1.1 Android Studio

Το Android Studio είναι ένα ενοποιημένο περιβάλλον ανάπτυξης κώδικα για τις ανάγκες του Android App development, και βασίζεται στο IntelliJ IDEA.
Πέρα από την ανάπτυξη κώδικα, το Android Studio δίνει την δυνατότητα χρήσης συστημάτων όπως είναι το Gradle και ο Android Emulator για τις ανάγκες μεταγλώττισης και εκτέλεσης της εφαρμογής χωρίς την απαραίτητη χρήση συσκευής. Ακόμη, υπάρχει διασύνδεση με το Firebase της Google, αλλά και το Github, δίνοντας στον developer απαραίτητα εργαλεία για την γρήγορη και ευέλικτη ανάπτυξη εφαρμογών [18].
Αναλυτικότερα το Android Studio προσφέρει:

- Ένα ευέλικτο σύστημα μεταγλώττισης (Gradle).
- Ένα γρήγορο εξομοιωτή πλούσιο σε χαρακτηριστικά.
- Ένα ενοποιημένο περιβάλλον για ανάπτυξη σε όλες τις συσκευές Android.
- Εφαρμογή και προώθηση των αλλαγών κώδικα και πόρων στην τρέχουσα εφαρμογή χωρίς επανεκκίνηση της εφαρμογής.
- Πρότυπα κώδικα και ενσωμάτωση GitHub.
- Εκτεταμένα εργαλεία δοκιμής και πλαίσια.
- Εργαλεία για την απόδοση, τη χρηστικότητα, τη συμβατότητα έκδοσης και άλλα συχνά θέματα.
- Υποστήριξη C ++ και NDK.
- Ενσωματωμένη υποστήριξη για το Google Cloud Platform, διευκολύνοντας την ενσωμάτωση του Google Cloud Messaging και του App Engine.

2.1.2 Android

Το Android είναι λειτουργικό σύστημα για κινητές και tablet συσκευές. Ξεκίνησε το 2003 ως εγχείρημα της εταιρίας τεχνολογίας Android Inc. για την δημιουργία λειτουργικού συστήματος ψηφιακών καμερών. Το 2004 το εγχείρημα άλλαξε, ώστε να γίνει λειτουργικό σύστημα για smartphones και το 2005 η εταιρεία εξαγοράστηκε από την Google.
Η Google, μετά την απόκτηση της εταιρείας, αποφάσισε να βασίσει το εγχείρημα σε Linux, λειτουργικό σύστημα για υπολογιστές. Στις 5 Νοεμβρίου 2007 η Google ανακοίνωσε την ίδρυση της Open Handset Alliance, μιας κοινοπραξίας δεκάδων εταιρειών τεχνολογίας και κινητής τηλεφωνίας, συμπεριλαμβανομένων των Intel Corporation, Motorola, NVIDIA

Corporation, Texas Instruments Incorporated, LG Electronics, Samsung Electronics, Sprint Nextel Corporation και T-Mobile (Deutsche Telekom). Η κοινοπραξία δημιουργήθηκε για να αναπτύξει και να προωθήσει το Android ως ένα ελεύθερο λειτουργικό σύστημα ανοιχτού κώδικα με υποστήριξη για εφαρμογές τρίτων. Οι συσκευές που βασίζονται σε Android χρησιμοποιούν ασύρματα δίκτυα για να επωφεληθούν πλήρως από λειτουργίες όπως αναζητήσεις Google με ένα άγγιγμα, Google Docs (π.χ. προγράμματα επεξεργασίας κειμένου, υπολογιστικά φύλλα) και το Google Earth (λογισμικό χαρτογράφησης δορυφόρων) [19].

2.1.3 Java

Η Java είναι μια γλώσσα προγραμματισμού γενικής χρήσης με βάση τον αντικειμενοστραφή προγραμματισμό και έχει σχεδιαστεί για να έχει λιγότερες εξαρτήσεις από βιβλιοθήκες. Είναι μια πλατφόρμα υπολογιστών για ανάπτυξη εφαρμογών.
Η Java είναι γρήγορη, ασφαλής και αξιόπιστη. Χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών σε φορητούς υπολογιστές, κέντρα δεδομένων, κονσόλες παιχνιδιών, επιστημονικούς υπερυπολογιστές και κινητά τηλέφωνα [20].
Η γλώσσα ξεκίνησε από τον James Gosling τον Ιούνιο του 1991, έτσι ώστε να την χρησιμοποιήσει σε ένα από τα πολλά του project για set-top κουτιά τηλεοράσεων. Η γλώσσα αρχικώς είχε το όνομα 'Oak', βάσει ενός δέντρου βελανιδιάς που βρισκόταν έξω από γραφείο του Gosling. Κάποια στιγμή η γλώσσα είχε και το όνομα 'Green' αν και αργότερα πήρε το όνομα Java, από μια τυχαία λίστα λέξεων.
Η Sun κυκλοφόρησε την πρώτη δημόσια υλοποίηση της γλώσσας ως Java 1.0 το 1995 και υποσχέθηκε πως θα μπορεί να γραφεί και να τρέχει παντού, χωρίς επιπλέον κόστος στους χρόνους εκτέλεσης δημοφιλών συστημάτων.
Στις 13 Νοεμβρίου 2006, η Sun κυκλοφόρησε το μεγαλύτερο μέρος της Java ως δωρεάν και ανοιχτού κώδικα λογισμικό υπό τους όρους του GNU General Public License (GPL).
Στις 8 Μαΐου 2007, η Sun ολοκλήρωσε την διαδικασία, διαθέτοντας όλο τον πηγαίο κώδικα της Java δωρεάν, εκτός από ένα μικρό μέρος του οποίου η Sun δεν διέθετε τα δικαιώματα.
Τελικώς, στις 20 Απριλίου 2009, η Oracle αγοράζει την Sun Microsystems μετατρέποντας τη Java ως τη δημοφιλέστερη γλώσσα για αντικειμενοστραφή προγραμματισμό [21].

2.1.4 Picasso

Η Picasso είναι μια γρήγορη, αποδοτική και ανοιχτού κώδικα βιβλιοθήκη φόρτωσης εικόνων (caching library) για το Android, όπου πακετάρει δυνατότητες αποκωδικοποίησης, μνήμης disk caching και resource pooling σε μια απλή και εύκολη στη χρήση διεπαφή.
Η Picasso υποστηρίζει την μεταφορά, αποκωδικοποίηση και απεικόνιση στιγμιότυπων βίντεο, εικόνων και κινούμενων GIFs. Η Picasso περιλαμβάνει ένα ευέλικτο API, που επιτρέπει στους προγραμματιστές να την εφαρμόσουν σε οποιοδήποτε δίκτυο. Από προεπιλογή, η Picasso κάνει χρήση ενός ειδικού HttpUrlConnection stack, είτε την OkHttp

βιβλιοθήκη της Square.
Το κύριο εγχείρημα της Picasso είναι να κάνει το scrolling οποιασδήποτε λίστας εικόνων όσο πιο ομαλό και γρήγορο γίνεται, όμως τα καταφέρνει το ίδιο καλά οποιαδήποτε στιγμή χρειαστεί να φέρει, αλλάξει μέγεθος ή να απεικονίσει μια απομακρυσμένη εικόνα [22].

2.1.5 Firebase

Το Firebase είναι μια δυνατή πλατφόρμα για κινητά και εφαρμογές web. Το Firebase τροφοδοτεί το backend πολλών εφαρμογών, δίνοντας δυνατότητες όπως αποθήκευση δεδομένων, ταυτοποίηση χρήστη και στατική φιλοξενία. Η πλατφόρμα κάνει δυνατή την ανάπτυξη εφαρμογών για κινητά και ιστό, που μπορούν να ικανοποιήσουν από έναν μέχρι ένα εκατομμύριο χρήστες.
Η ιστορία του Firebase ξεκινά το 2011, από μια startup με όνομα Envolve. Η εταιρεία αυτή εξόπλιζε τους developers της με ένα API όπου έδινε δυνατότητες online chat μέσω του site της. Το ενδιαφέρον της υπόθεσης, είναι πως οι χρήστες δεν περιορίζονταν μόνο στα μηνύματα, αλλά αντάλλασσαν δεδομένα εφαρμογών, όπως η κατάσταση ενός παιχνιδιού, σε πραγματικό χρόνο. Αυτό το συμβάν οδήγησε τους developers του Envolve, James Tamplin και Andrew Lee, να διαχωρίσουν το chat σύστημα από την real time αρχιτεκτονική. Έτσι, το 2012, δημιουργήθηκε η Firebase ως ξεχωριστή εταιρεία, που έδινε στους χρήστες της δυνατότητες Backend-as-a-Service με λειτουργίες πραγματικού χρόνου.
Το 2014 η εταιρεία αποκτήθηκε από την Google, εξελίσσοντας αστραπιαία την υπηρεσία ως έναν πολυεργαλείο για κινητά και εφαρμογές ιστού [23].

Κάποιες από τις λειτουργίες που γίνονται χρήση:

- Cloud Firestore: Εύκολη στη χρήση NoSQL βάση δεδομένων με δομή δέντρου που χρησιμοποιεί αρχεία JSON.
- Authentication: Εύκολο στη χρήση authentication για την εφαρμογή, ενοποιώντας το email χρήστη με μοναδικό userID
- Firebase Storage: Εύκολη και γρήγορη πρόσβαση σε δεδομένα μεγαλύτερου όγκου, όπως εικόνες, ήχος και βίντεο [24].

2.2 Server Ταξινόμησης

2.2.1 Python

Η Python είναι μια μεταγλωττισμένη, αντικειμενοστραφής, γλώσσα προγραμματισμού υψηλού επιπέδου με δυναμική σημασιολογία. Οι ενσωματωμένες δομές δεδομένων υψηλού επιπέδου, σε συνδυασμό με τη δυναμική πληκτρολόγηση και τη δυναμική δέσμευση, την καθιστούν πολύ ελκυστική για την ταχεία ανάπτυξη εφαρμογών, καθώς και για χρήση ως γλώσσα scripting ή ως συνδετικό κρίκο υπαρχόντων στοιχείων. Η απλή και εύχρηστη

σύνταξη της Python δίνει έμφαση στην αναγνωσιμότητα, και συνεπώς μειώνει το κόστος συντήρησης ενός προγράμματος. Η Python υποστηρίζει λειτουργικές μονάδες και πακέτα, τα οποία ενθαρρύνουν τη διαμόρφωση του προγράμματος και την επαναχρησιμοποίηση κώδικα. Ο διερμηνέας Python και η εκτεταμένη τυπική βιβλιοθήκη διατίθενται σε πηγαίο κώδικα ή δυαδική μορφή χωρίς χρέωση για όλες τις μεγάλες πλατφόρμες και μπορεί να διανεμηθεί ελεύθερα.
Συχνά, οι προγραμματιστές ερωτεύονται την Python λόγω της αυξημένης παραγωγικότητας που παρέχει. Επειδή δεν υπάρχει βήμα σύνταξης, ο κύκλος επεξεργασίας-δοκιμής-εντοπισμού σφαλμάτων είναι απίστευτα γρήγορος. Ο εντοπισμός σφαλμάτων των προγραμμάτων Python είναι εύκολος: ένα σφάλμα ή κακή είσοδος δεν θα προκαλέσει ποτέ σφάλμα τμηματοποίησης. Αντ 'αυτού, όταν ο διερμηνέας ανακαλύπτει ένα σφάλμα, δημιουργεί μια εξαίρεση. Όταν το πρόγραμμα δεν έχει την εξαίρεση, ο διερμηνέας εκτυπώνει ένα ίχνος στοίβας. Ένα πρόγραμμα εντοπισμού σφαλμάτων σε επίπεδο πηγής επιτρέπει την επιθεώρηση τοπικών και παγκόσμιων μεταβλητών, την αξιολόγηση αυθαίρετων εκφράσεων, τον καθορισμό σημείων διακοπής, τον έλεγχο του κώδικα μια γραμμή κάθε φορά και ούτω καθεξής. Το πρόγραμμα εντοπισμού σφαλμάτων είναι γραμμένο στην ίδια την Python, μαρτυρώντας την ενδοσκοπική ισχύ της γλώσσας. Από την άλλη πλευρά, συχνά ο γρηγορότερος τρόπος για την αποσφαλμάτωση ενός προγράμματος είναι η πρόσθεση μερικών δηλώσεων εκτύπωσης στην πηγή: ο γρήγορος κύκλος επεξεργασίας-δοκιμής-εντοπισμού σφαλμάτων καθιστά αυτήν την απλή προσέγγιση πολύ αποτελεσματική [25].
Η γλώσσα σχεδιάστηκε από τον Guido van Rossum το 1991 και αναπτύχθηκε από την Python Software Foundation. Κύριο μέλημα της ήταν η ευκολία στην ανάγνωση του κώδικα και της σύνταξής του, επιτρέποντας τους προγραμματιστές να εκφράσουν έννοιες με λιγότερες γραμμές κώδικα.
Σήμερα η Python έχει φτάσει να θεωρείται η πιο δημοφιλής γλώσσα κώδικα στον κόσμο, κλείνοντας τα 30 της χρόνια [26].

2.2.2 FastAPI

Το FastAPI είναι ένα μοντέρνο, γρήγορο και υψηλών επιδόσεων web framework, κατάλληλο για την κατασκευή APIs με τη βοήθεια της Python 3.6+ και βασισμένο στη σύνταξη της κλασικής Python [27].
Από την πρώτη του κυκλοφορία στα τέλη του 2018, το FastAPI διαφοροποιεί τον εαυτό του από άλλα Python frameworks, προσφέροντας ένα μοντέρνο και γρήγορο τρόπο για δημιουργία αξιόπιστων REST APIs. Αν και είναι ένα από τα νεότερα ανοιχτού - κώδικα Python frameworks που είναι διαθέσιμα, το framework έχει ήδη αρκετούς πιστούς ακόλουθους με πάνω από 22.000 αστέρια στο Github, και μια ενεργή κοινότητα που το συντηρεί και συνεχίζει να το εξελίσσει [28], [29].

2.2.3 Uvicorn

Το Uvicorn είναι μια ταχύτατη υλοποίηση ASGI εξυπηρετητή, κάνοντας χρήση των βιβλιοθηκών uvloop και httptools. Συναντώντας την ελάχιστη κάλυψη low-level server/application διεπαφής για ασύγχρονα frameworks, το Uvicorn δίνει τα κατάλληλα εργαλεία για την χρήση ασύγχρονων frameworks [30].
Από τις 30 Μαΐου 2017, το Uvicorn βοηθά στην ενεργοποίηση του οικοσυστήματος των web frameworks της Python, ανταγωνιζόμενο αυτά των Node και Go στον τομέα του high throughput. Ακόμη, δίνει την δυνατότητα υποστήριξης HTTP/2 και WebSockets που δεν είναι διαχειρίσιμα από WSGI εξυπηρετητές.

2.2.4 Tensorflow

Το Tensorflow είναι μια ανοιχτού κώδικα βιβλιοθήκη της Google που έχει γίνει πολύ δημοφιλής στον τομέα του Machine Learning, καθώς προσφέρει διάφορα APIs για πολλούς τομείς δεδομένων. Το Tensorflow προσφέρει τους γρηγορότερους χρόνους μεταγλώττισης σε σχέση με άλλες βιβλιοθήκες όπως Keras, Touch και υποστηρίζει υπολογισμούς σε CPU και GPU.
Για την δημιουργία – εκπαίδευση νέων μοντέλων, η βιβλιοθήκη δέχεται ως είσοδο μια μορφή πολυδιάστατων πινάκων, οι οποίοι ονομάζονται Tensors. Στη συνέχεια, ο μηχανισμός εκτέλεσης γίνεται με την μορφή γράφων: κάθε κόμβος στον γράφο αντιπροσωπεύει μια μαθηματική πράξη (πρόσθεση, αφαίρεση, πολλαπλασιασμό) και κάθε σύνδεση μεταξύ των κόμβων αντιπροσωπεύει τους πολυδιάστατους πίνακες [31].
Κάθε μοντέλο μπορεί να δημιουργηθεί σε πολλών ειδών μηχανήματα και μπορεί να χρησιμοποιηθεί είτε από GPUs είτε από CPUs.
Αν και αρχικά οι GPUs σχεδιάστηκαν για βίντεο παιχνίδια, το 2010 ερευνητές στο Stanford ανακάλυψαν πως είναι πολύ καλές και γρήγορες σε πράξεις πινάκων. Καθώς το Deep Learning βασίζεται σε μεγάλο βαθμό σε ανάλογες πράξεις, οι GPUs είναι ιδανικές για τη χρήση στο κομμάτι του Machine Learning [32].

Κεφάλαιο 3: Σχεδίαση Συστήματος

Στο τρίτο Κεφάλαιο περιγράφεται η λειτουργία του συστήματος ταξινόμησης εικόνων που αποτελείται από την Android εφαρμογή απεικόνισης και διαχείρισης των εικόνων (AI Gallery App), τον Server Αρχείων και τον Server Ταξινόμησης. Επίσης, θα γίνει περιγραφή των χαρακτηριστικών του συστήματος, της διασύνδεσης μεταξύ τους αλλά και διάφορων στιγμιοτύπων από τις οθόνες της εφαρμογής Android.

3.1 Περιγραφή

Το σύστημα ταξινόμησης εικόνων είναι μια ολοκληρωμένη εφαρμογή αποθήκευσης και κατηγοριοποίησης εικόνας (AI Gallery App) για κινητές συσκευές με υποβοήθηση εξυπηρετητή.
Ο σχεδιασμός του συστήματος χωρίστηκε σε τρία στάδια. Το στάδιο αποθήκευσης των φωτογραφιών που προέρχονται από την κινητή συσκευή μαζί με την ετικέτα ταξινόμησης (File Server), το στάδιο ταξινόμησης των εικόνων (Classification Server) και το στάδιο δημιουργίας, φόρτωσης, παρουσίασης και διαγραφής των εικόνων που εισάγονται στο σύστημα (AI Gallery App).
Η ιδιαιτερότητα της εφαρμογής εντοπίζεται στην υλοποίηση του συστήματος με απομακρυσμένο εξυπηρετητή, με στόχο τη μη συμμετοχή των κινητών συσκευών στο στάδιο της ταξινόμησης. Ο απομακρυσμένος Server Ταξινόμησης χρησιμοποιείται για την αύξηση του εύρους των συσκευών που μπορούν να αξιοποιηθούν, αλλά και τη δυνατότητα ταξινόμησης μεγάλου πακέτου εικόνων ταυτόχρονα, με κέρδος σε χρόνο και ευστοχία.

3.2 Server Αρχείων

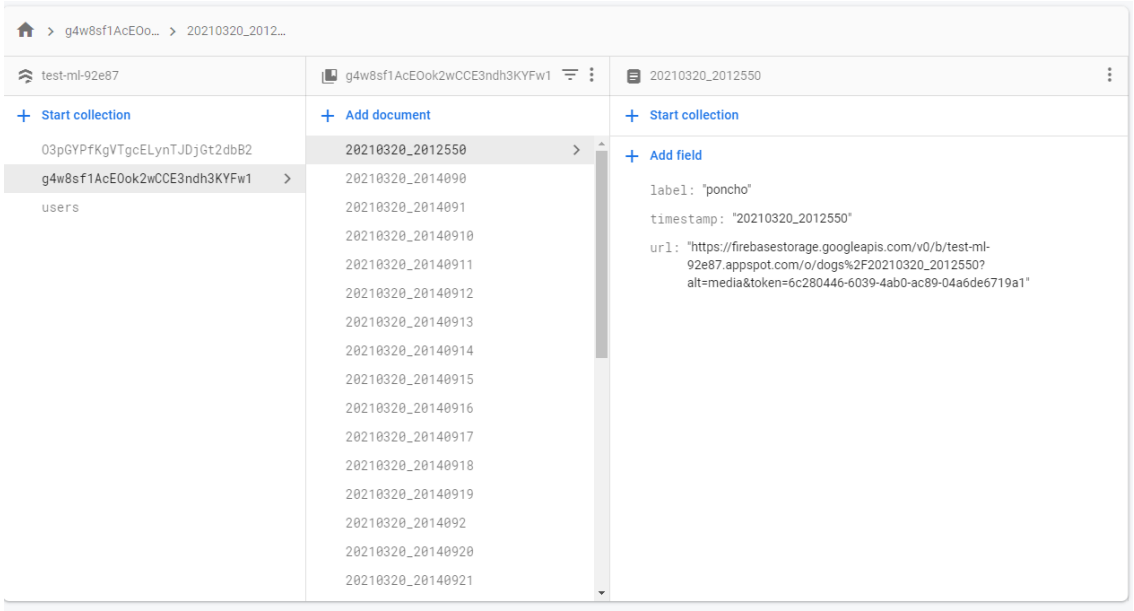
Το Firebase της Google έκανε δυνατή την άμεση δημιουργία δομής δεδομένων για την αποθήκευση εικόνων και ετικετών ταξινόμησης.
Αναλυτικότερα, έγινε χρήση των υπηρεσιών Firebase FireStore (αποθήκευση πληροφοριών ταξινόμησης), Firebase Storage (αποθήκευση φωτογραφιών) και Firebase Authentication (διαχείριση χρηστών).

Η δομή αποθήκευσης (Εικόνα 3.1) στο Firestore αποθηκεύει τα δεδομένα με τον παρακάτω τρόπο:

• Κάθε χρήστης δημιουργείται ως συλλογή, με αναγνωριστικό το uid του χρήστη.

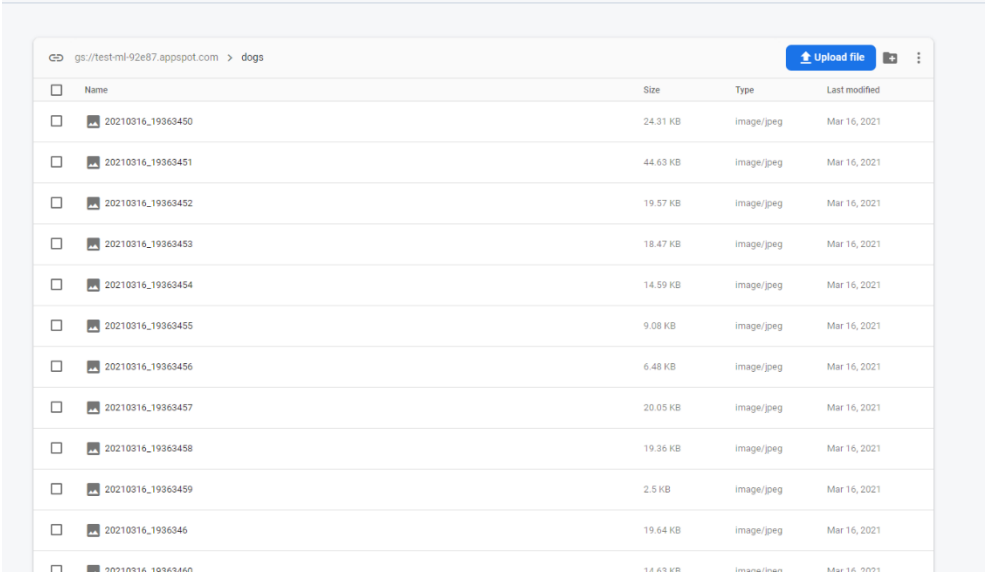
• Κάθε χρήστης – συλλογή έχει πολλά αρχεία - φωτογραφίες με αναγνωριστικό το χρόνο δημιουργίας της φωτογραφίας (timestamp).

• Κάθε αρχείο - φωτογραφία περιέχει σε μορφή κλειδί : τιμή τις τιμές ετικέτα ταξινόμησης, timestamp εικόνας, url εικόνας.



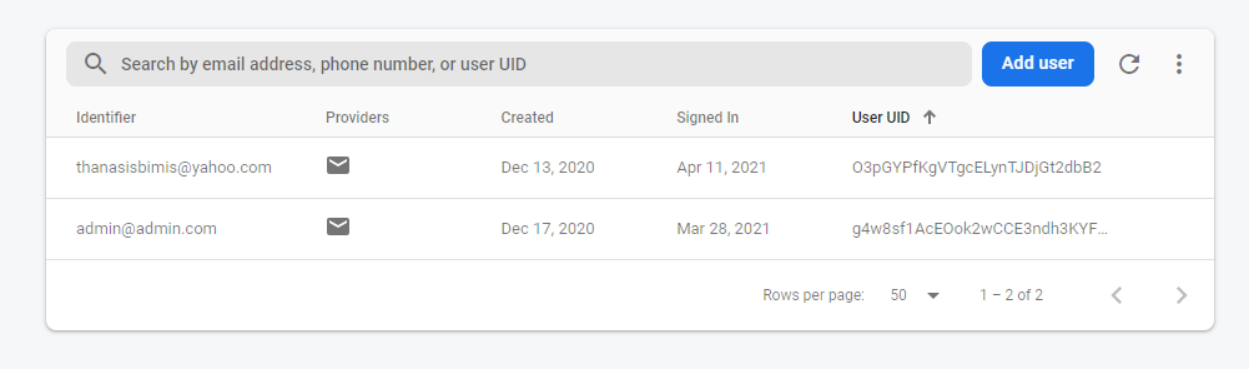
Εικόνα 3.1: Δομή πληροφοριών στο Firebase Firestore

Το url που περιέχεται στο FireStore δείχνει τον σύνδεσμο που είναι αποθηκευμένη η φωτογραφία στο Firebase Storage (Εικόνα 3.2).



Εικόνα 3.2: Αποθηκευμένες εικόνες στο Firebase Storage

Ο Server Αρχείων χρησιμεύει και ως μέσο ταυτοποίησης χρήστη, καθώς το Firebase έχει ενσωματωμένη υπηρεσία Authentication χρηστών (Εικόνα 3.3), δίνοντας μοναδικό αναγνωριστικό (uid) σε κάθε χρήστη, με τη δημιουργία λογαριασμού με email και κωδικό.



Εικόνα 3.3: Firebase Authentication και λογαριασμοί

3.3 Server Ταξινόμησης

Ο Server Ταξινόμησης (Classification Server) είναι ένα πολύ σημαντικό κομμάτι της υλοποίησης του συστήματος ταξινόμησης εικόνων. Η δουλειά του server είναι να δημιουργεί ετικέτες ταξινόμησης για τις εικόνες που εισάγονται στο σύστημα μέσω της εφαρμογής στο κινητό (AI Gallery App).
Για τις ανάγκες υλοποίησης του Server έγινε χρήση του web framework FastAPI σε γλώσσα Python και περικλείεται ως web server με χρήση του Uvicorn.

Δομή

Το FastApi επιτρέπει εύκολο και γρήγορο στήσιμο κώδικα για τον server, έχοντας ως βασικό κορμό τον κώδικα που είναι εμφανής στην παρακάτω εικόνα (Εικόνα 3.4).

```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
def read_root():
    return {"Hello": "World"}
```

Εικόνα 3.4: Βασικός κορμός λειτουργίας του FastAPI

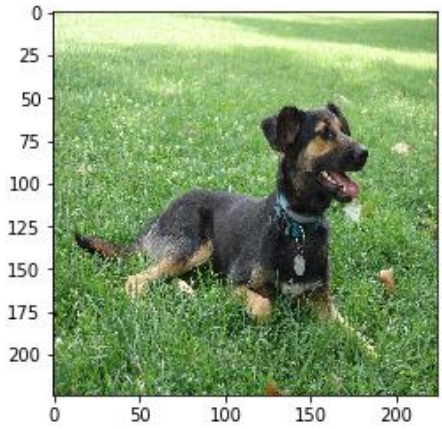
Ουσιαστικά επιτρέπει την άμεση δημιουργία λειτουργικού server, που είναι προσβάσιμος σε επιθυμητό url. Επίσης, δίνει την δυνατότητα εκτέλεσης ασύγχρονων συναρτήσεων για web servers που το επιτρέπουν.
Με την εκκίνηση του server αρχικοποιούνται όλα τα μοντέλα Μηχανικής Μάθησης του Tensorflow, με κάθε μοντέλο να αντιστοιχεί σε διαφορετικό url.
Η λειτουργία ταξινόμησης του server ενεργοποιείται κάθε φορά που καλείται post request

σε url αντίστοιχου μοντέλου ταξινόμησης από την εφαρμογή Android (AI Gallery App). Τότε, μια συνάρτηση δέχεται ως είσοδο μία ή πολλές εικόνες για ταξινόμηση, και τελικώς επιστρέφει ένα JSON αρχείο με μία ή πολλές ετικέτες ταξινόμησης αντίστοιχα.

Στάδια Ταξινόμησης

Αναλυτικότερα στα στάδια ταξινόμησης ο server:

1. Διαβάζει την εικόνα που φτάνει (Εικόνα 3.5).



Εικόνα 3.5: Παράδειγμα εικόνας ανάλυσης 224x224 που εισάγεται στον Server Ταξινόμησης

2. Μετατρέπει την εικόνα σε πίνακα Bytes.

3. Πριν την είσοδο των δεδομένων της εικόνας στην προεπεξεργασία, η βιβλιοθήκη numpy μετατρέπει την είσοδο σε μονοδιάστατο πίνακα.

4. Τα δεδομένα εισάγονται στο στάδιο προεπεξεργασίας του συγκεκριμένου μοντέλου ταξινόμησης εικόνας, με την έξοδο να εξάγει πίνακα όπου μετατρέπει τη σειρά των bytes από RGB σε BGR, και έπειτα κάθε κανάλι χρώματος κεντράρεται στο μηδέν σε σχέση με το σύνολο δεδομένων ImageNet (τιμές από το -1 έως το 1).

5. Έπειτα ο πίνακας εισάγεται στο μοντέλο και αρχίζει η συμπερασματολογία (Εικόνα 3.6).

```
preds = modelRes.predict(dedomena, batch_size=megethos)
```

Εικόνα 3.6: Εκτέλεση συμπερασματολογίας

6. Τα αποτελέσματα της συμπερασματολογίας γίνονται decode και ως έξοδος δίνονται τα πέντε πρώτα αποτελέσματα σε αυτή (Εικόνα 3.7) τη μορφή.

```
[[('n02093754', 'Border_terrier', 0.37348402),
  ('n02106662', 'German_shepherd', 0.1644547),
  ('n02105162', 'malinois', 0.123688936),
  ('n02106550', 'Rottweiler', 0.07267013),
  ('n02096051', 'Airedale', 0.04084967)]]
```

Εικόνα 3.7: Αποτελέσματα συμπερασματολογίας μετά το στάδιο decode

7. Τέλος, τα δεδομένα δίνονται ως έξοδος σε μορφή JSON για την δημοφιλέστερη απάντηση.

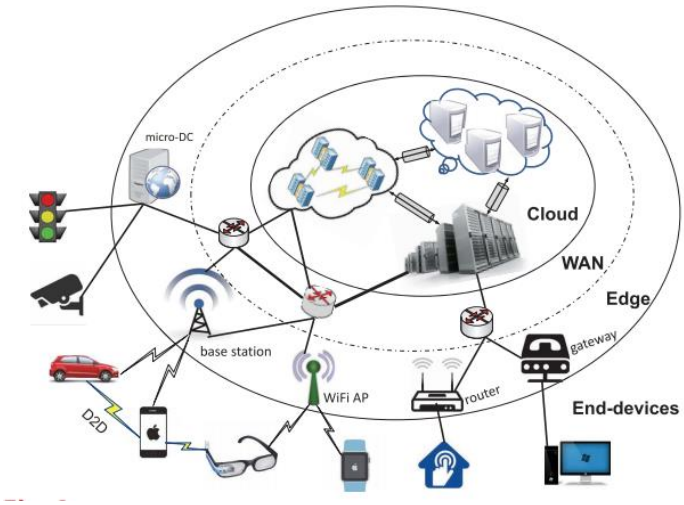
{ “label”: ‘Border_terrier’, “confidence”: 0.37348402 }

3.3.1 Δημιουργία τοπολογιών

Για τις ανάγκες της εργασίας κρίθηκε απαραίτητη η εξέταση διαφορετικών τοπολογιών δικτύων για τον απομακρυσμένο server. Συγκεκριμένα, γίνεται αξιολόγηση της διεκπεραιωτικής ικανότητας του συστήματος με χρήση του υπολογιστικού νέφους (cloud computing), σε σύγκριση με τον υπολογισμό στα άκρα του δικτύου (edge computing).

Server Νέφους (Cloud Server)

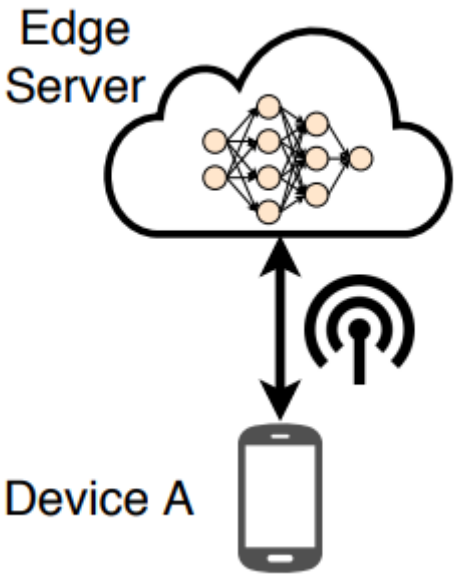
Ο Server Νέφους (Cloud Server), δημιουργήθηκε σε Virtual Machine (Windows 10) μηχάνημα που παραχωρήθηκε από το εργαστήριο. Ο server αυτός έχει πρόσβαση στο διαδίκτυο μέσω του δικτύου του Εργαστήριου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης και είναι προσβάσιμος από την συσκευή κινητού μόνο μέσω διαδικτύου (Εικόνα 3.8). Ακόμη, οι υπολογισμοί για την συμπερασματολογία (inference) των μοντέλων ταξινόμησης γίνονται με CPU.
Για την εκτέλεση του κώδικα (που δημιουργήθηκε με το FastAPI web framework), γίνεται χρήση του Uvicorn, κάνοντας διαθέσιμο τον server στην public IP (host = 0.0.0.0) του VM και την επιλεγμένη πύλη (port).



Εικόνα 3.8: Απεικόνιση θέσης των Edge οντοτήτων σε σχέση με το Cloud

Server Στην Άκρη του Δικτύου (Edge Server)

Ο Server στην Άκρη του Δικτύου (Edge Server) δημιουργήθηκε σε μηχάνημα (Windows 7) που είναι μέλος του τοπικού δικτύου της κινητής συσκευής, με το μηχάνημα του server να κάνει χρήση GPU για τους υπολογισμούς της συμπερασματολογίας των μοντέλων ταξινόμησης. Για την λειτουργία επιλέχθηκε αρχιτεκτονική που βασίζεται στην άκρη του δικτύου (Edge - based). Δηλαδή η συσκευή στέλνει τα δεδομένα (Εικόνα 3.9) για ανάλυση στον server. Μόλις η συμπερασματολογία τελειώσει, τα αποτελέσματα επιστρέφονται στην συσκευή. Υπό αυτή την αρχιτεκτονική, είναι εύκολη η εφαρμογή του συστήματος σε οποιαδήποτε συσκευή, με το μεγαλύτερο μειονέκτημα να εμφανίζεται στην καθυστέρηση (latency) επικοινωνίας μεταξύ της κινητής συσκευής και της άκρης του δικτύου.



Εικόνα 3.9: Μοντέλο βασιζόμενο στην άκρη του δικτύου (Edge-Based Model)

Ο server εκτελεί τον κώδικα του FastAPI με τη χρήση του uvicorn, κάνοντας τον διαθέσιμο στην local ip (host = 0.0.0.0) και την επιλεγμένη πύλη.

3.4 Εφαρμογή Android

Η εφαρμογή Android αναλαμβάνει την διαχείριση - προβολή των εικόνων που είτε δημιουργούνται, είτε φορτώνονται σε αυτή για ταξινόμηση.
Η ανάπτυξη της εφαρμογής έγινε με τη βοήθεια των Android SDK και Android Studio, που έπαιξαν μεγάλο ρόλο στη γρήγορη και αυτοματοποιημένη δημιουργία του .apk.

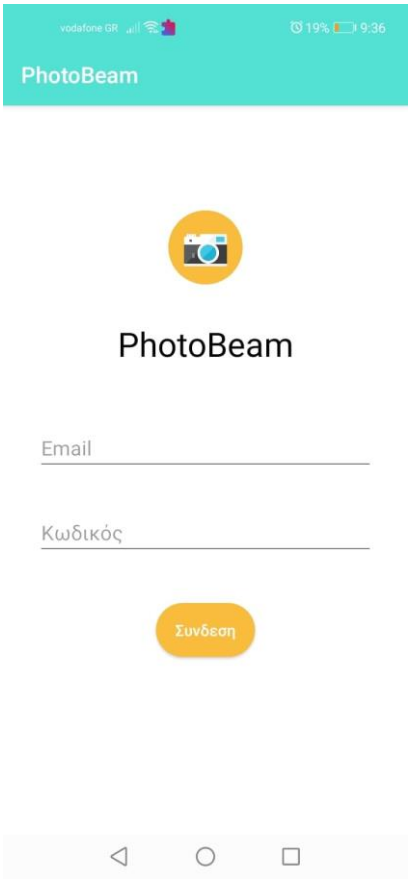
3.4.1 Οθόνες Εφαρμογής

Η εφαρμογή (AI Gallery App) αποτελείται από τις ακόλουθες οθόνες:

1. Σύνδεση Χρήστη
2. Κύρια Οθόνη με πλέγμα (Gallery)
3. Προβολή Εικόνας

Σύνδεση Χρήστη

Με την εκκίνηση της εφαρμογής, εάν δεν έχει ξαναγίνει σύνδεση, απαιτείται η προσθήκη στοιχείων χρήστη (ηλ.ταχυδρομείο, κωδικός) για την είσοδο στην εφαρμογή (Εικόνα 3.10). Αν ο χρήστης έχει συνδεθεί ξανά από τη τρέχουσα συσκευή, η εφαρμογή συνδέεται με τον τελευταίο λογαριασμό που είχε συνδεθεί επιτυχώς. Η παραπάνω δυνατότητα πραγματοποιείται μέσω των στοιχείων χρηστών, που βρίσκονται στον Server Αρχείων. Κάθε λογαριασμός (και αντίστοιχα οι κατηγοριοποιημένες εικόνες) είναι προσβάσιμος από οποιαδήποτε συσκευή με τη χρήση σύνδεσης στο διαδίκτυο.



Εικόνα 3.10: Οθόνη σύνδεσης χρήστη

Κυρία Οθόνη με Πλέγμα (Gallery)

Έπειτα από την επιτυχημένη είσοδο, ο χρήστης συναντά μια οθόνη (Εικόνα 3.11) που του δίνει τις δυνατότητες:

- Προβολής των αποθηκευμένων online εικόνων σε πλέγμα μαζί με την ετικέτα ταξινόμησης.
- Αναζήτησης εικόνων βάσει της ετικέτας ταξινόμησης (Εικόνα 3.12).
- Φόρτωσης εικόνας/ων στο πλέγμα για εύρεση ετικέτας ταξινόμησης μέσω του Server Ταξινόμησης.
- Δημιουργίας στιγμιότυπου εικόνας μέσω του αισθητήρα της κάμερας και εύρεση της ετικέτας ταξινόμησης.
- Ανανέωσης του πλέγματος φωτογραφιών.
- Αποσύνδεσης από το λογαριασμό.

Εικόνα 3.11: Κύρια οθόνη με πλέγμα

Εικόνα 3.12: Κύρια οθόνη με πλέγμα και αναζήτηση

Προβολή Εικόνας

Στην οθόνη Προβολής Εικόνας (Εικόνα 3.13) ο χρήστης βλέπει την εικόνα που επέλεξε μέσω του πλέγματος, την ετικέτα της και ένα κουμπί που επιτρέπει την διαγραφή της εικόνας.
Η οθόνη Προβολής Εικόνας ενεργοποιείται με το αντίστοιχο πάτημα του τετραγώνου στο πλέγμα και εμφανίζεται με τη βοήθεια του Intent του Android SDK.

Εικόνα 3.13: Οθόνη προβολής φωτογραφίας

3.4.2 Στάδια Επεξεργασίας

Κατά τη δημιουργία στιγμιοτύπου εικόνας μέσω της κάμερας, η εφαρμογή δημιουργεί δύο αρχεία.
Το πρώτο αρχείο διοχετεύεται στον Server Ταξινόμησης, ενώ το δεύτερο στον Server Αρχείων και συγκεκριμένα στην υπηρεσία Firebase Storage.

Περιστροφή Εικόνων

Καθώς τα αρχεία που δημιουργούνται σε κάθε διαφορετικό αισθητήρα κάμερας μπορεί να έχουν διαφορετικό προεπιλεγμένο προσανατολισμό, οι εικόνες ελέγχονται και περιστρέφονται ώστε να εμφανίζονται σωστά. Με τον τρόπο αυτό διασφαλίζεται η σωστή

είσοδος των εικόνων για το στάδιο της ταξινόμησης, αλλά και η σωστή αποθήκευση και εμφάνιση στο πλέγμα της εφαρμογής.

Κλιμάκωση Εικόνων

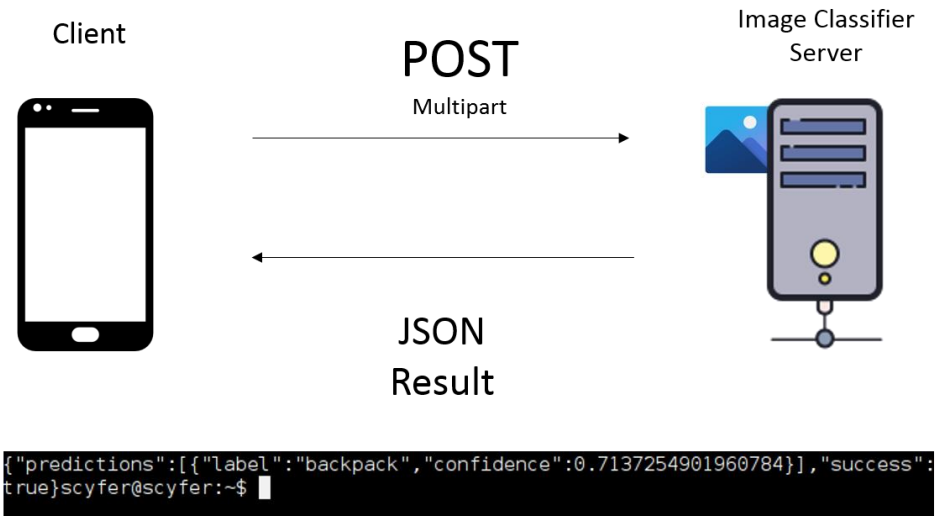
Ο Server Ταξινόμησης (Classify Server) αναμένει ως είσοδο εικόνες μικρότερης ανάλυσης από τις αρχικές. Αυτό συμβαίνει διότι τα μοντέλα ταξινόμησης εικόνων και γενικότερα Βαθιάς Μάθησης λειτουργούν με αρκετά μικρές αναλύσεις.
Έτσι, το πρώτο αρχείο δέχεται κλιμάκωση και συμπιέζεται με ποιότητα μικρότερη του αρχικού μεγέθους σύμφωνα με τα συμπεράσματα του Κεφαλαίου 4, ώστε να μειωθεί ο χρόνος αποστολής στον Server Ταξινόμησης.

Συμπίεση Εικόνων

Στο δεύτερο αρχείο η αρχική ανάλυση της κάμερας διατηρείται, καθώς δεν δέχεται κλιμάκωση και θα εμφανίζεται στο πλέγμα της εφαρμογής. Ακόμη, θα πρέπει να παρουσιάζεται σωστά και στην πλήρη οθόνη της κινητής συσκευής. Για κέρδος χρόνου κατά την αποστολή στον Server Αρχείων, το δεύτερο αρχείο δέχεται επίσης μεγάλη συμπίεση, σε σημείο όμως που η εμφάνιση δεν δείχνει να επηρεάζεται.

Αποστολή στον Server Ταξινόμησης

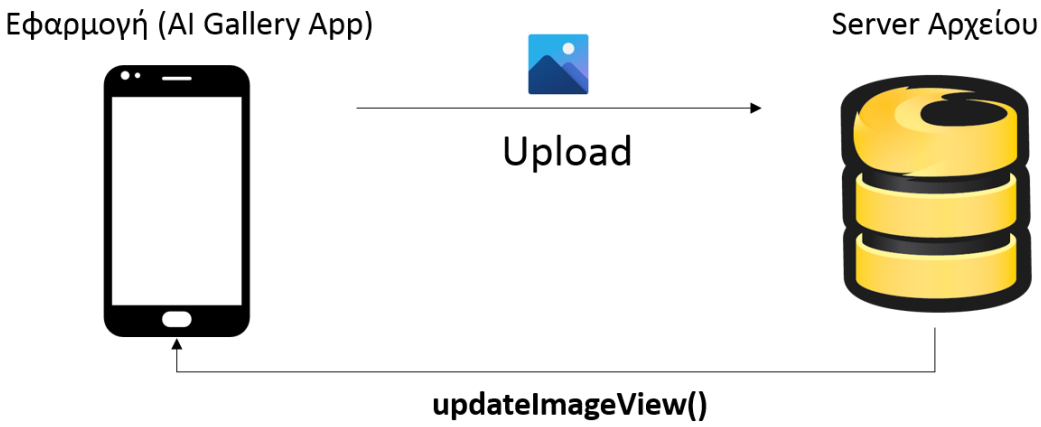
Την στιγμή που ο χρήστης στείλει κάποια εικόνα για ταξινόμηση (Εικόνα 3.14), η εφαρμογή επικοινωνεί με τον Server Ταξινόμησης ασύγχρονα (threading), αναμένοντας την απάντηση στο παρασκήνιο. Μόλις η εφαρμογή λάβει απάντηση, η ετικέτα ταξινόμησης μαζί με την εικόνα αποστέλλονται στο Server Αρχείων (FireStore και Firebase Storage αντίστοιχα).



Εικόνα 3.14: Διάγραμμα αποστολής εικόνας στον Server Ταξινόμησης

Αποστολή στον Server Αρχείων

Για να δει ο χρήστης τις ταξινομημένες φωτογραφίες του, η εφαρμογή επικοινωνεί με τον Server Αρχείων (Εικόνα 3.15) κατεβάζοντας τα αντίστοιχα αρχεία, που στη συνέχεια φορτώνονται για προβολή στο πλέγμα φωτογραφιών μαζί με την ετικέτα ταξινόμησης.



Εικόνα 3.15: Διάγραμμα αποστολής εικόνων στο Firebase

Η επικοινωνία με τον Server Αρχείων συμβαίνει με κάθε ενέργεια σχετική με το σύνολο των αρχείων της συλλογής, έτσι ώστε να υπάρχει πιστή αναπαράσταση των νέων, διαγραμμένων ή τροποποιημένων εικόνων.

Παραλληλία Εργασιών

Για την ομαλή λειτουργία της εφαρμογής, κάθε ενέργεια που εκτελείται συμβαίνει όσο αυτό είναι δυνατό στο παρασκήνιο. Με αυτό τον τρόπο διασφαλίζεται και ο ελάχιστος δυνατός χρόνος για την εμφάνιση, διαγραφή και δημιουργία ταξινόμησης για τις εικόνες που περιέχονται στον λογαριασμό χρήστη.

Κεφάλαιο 4: Ανάπτυξη Συστήματος

Το Κεφάλαιο αυτό αναλύει τα βασικά στοιχεία ανάπτυξης του συστήματος. Η διαδικασία αυτή χωρίστηκε σε δύο μέρη. Στο πρώτο μέρος αναπτύχθηκε η εφαρμογή που θα δημιουργεί, φορτώνει και προβάλλει τις εικόνες που εισάγονται σε αυτή μαζί με την ετικέτα κατηγοριοποίησης. Στο δεύτερο μέρος αναπτύχθηκαν οι δύο Servers Ταξινόμησης (Cloud, Edge) για τις εικόνες που εισάγονται στο σύστημα.
Στη συνέχεια βλέπουμε τις κλάσεις που βοήθησαν στην ανάπτυξη των δύο αυτών κομματιών που ο σχεδιασμός τους αναλύθηκε στο Κεφάλαιο 3.

4.1 Java Κλάσεις

4.1.1 Προβολής Εικόνων

Για την εμφάνιση των εικόνων στην εφαρμογή κρίθηκε απαραίτητη η δημιουργία των κλάσεων ImageAdapterGridView και UpdateImageViews.

Κλάση ImageAdapterGridView

Η κλάση ImageAdapterGridView επιτρέπει την φόρτωση των εικόνων στο πλέγμα διαδοχικά. Αυτό σημαίνει πως κάθε στοιχείο του πλέγματος δημιουργείται και συμπληρώνεται μεμονωμένα με τη βοήθεια της βιβλιοθήκης Picasso και της κλάσης BaseAdapter από την οποία κληρονομεί. Η Picasso κάνει δυνατή την αυτόματη αποκοπή φωτογραφιών στο μέγεθος του τετραγώνου στο πλέγμα και την τοποθέτηση της στο κέντρο, γεμίζοντας το πλέγμα με κάθε φωτογραφία που προσθέτει ή έχει αποθηκευμένη ο κάθε χρήστης στον προσωπικό του λογαριασμό.
Καθώς η φόρτωση εικόνων σε πλέγμα είναι μια ενεργοβόρα διαδικασία για να ολοκληρωθεί στο προσκήνιο, η ImageAdapterGridView καλείται μέσω της κλάσης UpdateImageViews.

Κλάση UpdateImageViews

Η UpdateImageViews κληρονομεί δυνατότητες από την κλάση AsyncTask [33] της βιβλιοθήκης android.os και δημιουργήθηκε για την εκτέλεση ολόκληρης της διαδικασίας εμφάνισης των εικόνων στο παρασκήνιο. Οι εικόνες, όπως και η ετικέτα ταξινόμησης, φορτώνονται από το σημείο αποθήκευσης (Server Αρχείων).

4.1.2 Προετοιμασίας Εικόνων

Το δεύτερο μέρος της εφαρμογής Android (AI Gallery App) επιλύει το κομμάτι της προετοιμασίας και αποστολής της/ων εικόνας/ων στον server ταξινόμησης. Το δεύτερο μέρος συμπιέζει όλα τα αρχεία εικόνας παράλληλα και μόλις ολοκληρωθεί η διαδικασία συμπίεσης, δημιουργείται ένα ArrayList αρχείων που αποστέλλεται στον Server Ταξινόμησης. Η παραπάνω διαδικασία περικλείεται στην κλάση CloseItAll που κληρονομεί από την κλάση Runnable και τρέχει ως Thread στο παρασκήνιο.

Κλάση CloseItAll

Η κλάση CloseItAll επιλύει την προετοιμασία αποστολής των αρχείων και την αποστολή τους στον Server Ταξινόμησης. Αρχικά, οι εικόνες μετατρέπονται σε Bitmap για να πραγματοποιηθεί η αποκοπή και συμπίεσή τους με τη βοήθεια των εργαλείων που κάνει διαθέσιμα η κλάση Bitmap [34] της βιβλιοθήκης android.graphics και στη συνέχεια με τη βοήθεια της κλάσης AndroidNetworking δημιουργείται post request τύπου multipart, που αποστέλλει το/α αρχείο/α αναμένοντας το ανάλογο response.
Για την ομαλή λειτουργεία της εφαρμογής η προετοιμασία και αποστολή των εικόνων γίνεται στο παρασκήνιο με τη βοήθεια της κλάσης Thread [35] του Android SDK.

4.2 Ανάπτυξη Server Ταξινόμησης

Για τις ανάγκες ανάπτυξης του Server Ταξινόμησης, έγινε χρήση των βιβλιοθηκών PIL, Numpy, Tensorflow και FastAPI.
Το FastAPI επέτρεψε την δημιουργία της δομής του server υλοποιώντας μια συνάρτηση για κάθε μοντέλο του Tensorflow.
Η βιβλιοθήκες PIL και Numpy κατέστησαν δυνατό το διάβασμα και την διαχείριση πινάκων για τα δεδομένα που προετοιμάζονται ως είσοδος της συμπερασματολογίας.
Εφόσον οι πίνακες με τα δεδομένα εικόνων έχουν πλέον μετατραπεί σε byte μορφή – ανάλογη του μοντέλου που εισάγονται, η βιβλιοθήκη του Tensorflow εκτελεί τη συμπερασματολογία μέσω της μεθόδου predict() και εξάγει αποτελέσματα για τις εικόνες εισόδου.
Τελικώς, δίνονται ως έξοδος τα αποτελέσματα του classification σε μορφή JSON αρχείου {“labels”:”value”} για κάθε είσοδο εικόνας/ων.

4.3 Μοντέλα Ταξινόμησης

Για να διαπιστωθεί η αποτελεσματικότητα και η διεκπεραιωτική ικανότητα της εφαρμογής σε cloud και edge server, έγινε χρήση τεσσάρων συνελικτικών δικτύων για τις δοκιμές πολλαπλών πακέτων εικόνων (batch). Τα μοντέλα αυτά ήταν τα EfficientNetB0, ResNetV50,

InceptionV3 και XceptionNet [36].
Τα παραπάνω μοντέλα δοκιμάστηκαν σε πολλαπλά μεγέθη batches.
Ακόμη, προστέθηκαν για χρήση τα μοντέλα NasNetLarge, VGG16, InceptionResNetV2 και EfficientNetB7, καθώς εξάγουν αποτελέσματα μεγάλης ακρίβειας. Τα χαρακτηριστικά των παραπάνω μοντέλων διακρίνονται στον πίνακα 4.1.
Ο λόγος που δεν έγιναν μετρήσεις με πολλαπλά batches και με τα υπόλοιπα μοντέλα, έχει να κάνει με τον χρόνο συμπερασματολογίας αυτών, που είναι αρκετά χρονοβόρος σε σχέση με τα τέσσερα αρχικά μοντέλα που ενδείκνυνται για πολλαπλά batch.

Μοντέλο	Ανάλυση Εισόδου	Top – 1 Ακρίβεια	Αριθμός Παραμέτρων
EfficientNetB0	224x224	71.3%	5.3 εκ.
ResNet50	224x224	74.9%	25.6 εκ.
InceptionV3	299x299	77.9%	23.8 εκ.
InceptionResNetV2	299x299	80.3%	55.8 εκ.
Xception	299x299	79%	22.9 εκ.
NASNetLarge	331x331	82.5%	88.9 εκ.
VGG16	224x224	71.3%	138.3 εκ.
EfficientNetB7	600x600	84.3%	66.6 εκ.

Πίνακας 4.1 Μοντέλα Ταξινόμησης

Κεφάλαιο 5: Αξιολόγηση

Σε αυτό το κεφάλαιο αναφέρονται οι μετρικές και οι συσκευές που χρησιμοποιήθηκαν για την αξιολόγηση της απόδοσης του συστήματος. Ακόμη, γίνεται σχολιασμός των αποτελεσμάτων για τα διάφορα μοντέλα ταξινόμησης εικόνων που χρησιμοποιούνται, όπως και των διαφορετικών τοπολογιών server ταξινόμησης μέσω του συνολικού χρόνου καθυστέρησης (latency).

5.1 Μετρικές και Συσκευές

Για την μέτρηση απόδοσης των μοντέλων ταξινόμησης εικόνων του συστήματος, έγιναν μετρήσεις του χρόνου συμπερασματολογίας (inference latency) και του συνολικού χρόνου καθυστέρησης στον Server στην Άκρη του Δικτύου (Edge Server) συγκριτικά με τον Server Νέφους (Cloud Server). Ακόμη, αξιολογήθηκαν συσκευές διαφορετικής παλαιότητας στο κομμάτι της συμπίεσης και εξετάσθηκε ο βαθμός που επηρεάζει η συμπίεση τα αποτελέσματα της συμπερασματολογίας.
Για την παραπάνω αξιολόγηση έγινε χρήση των μετρικών και συσκευών που παρουσιάζονται στην συνέχεια.

5.1.1 Μετρικές

Για τις ανάγκες των μετρήσεων της εφαρμογής έγινε χρήση των μετρικών:

Minimum: Η ελάχιστη μετρήσιμη τιμή του συνόλου των αποτελεσμάτων.

Maximum: Η μέγιστη μετρήσιμη τιμή του συνόλου των αποτελεσμάτων.

Average: Ο μέσος όρος του συνόλου των αποτελεσμάτων.

Median: Η τιμή που βρίσκεται στη μέση του συνόλου των αποτελεσμάτων.

90th percentile: Το 90% των αποτελεσμάτων βρίσκεται κάτω από αυτή την τιμή.

Throughput: Πλήθος εικόνων που επεξεργάζονται ανά 1 δευτερόλεπτο.

Ποιότητα (Quality): Η παράμετρος της συνάρτησης του Android SDK Bitmap.compress (bmp, quality), που ελέγχει την ποιότητα της JPEG συμπίεσης. Στο εύρος 0 – 100, 0 είναι η μικρότερη δυνατή ποιότητα-μέγεθος και 100 η καλύτερη [34].

Εμπιστοσύνη (Confidence): Είναι η πεποίθηση ότι το αποτέλεσμα που δίνει το μοντέλο συμπερασματολογίας (inference) εικόνας είναι σωστό.

5.1.2 Κινητές Συσκευές

Για τις ανάγκες της εφαρμογής ταξινόμησης εικόνων έγινε χρήση των συσκευών με τα παρακάτω χαρακτηριστικά:

Συσκευή	**Huawei P40 lite E (A)**	**Huawei P9 lite (B)**
Μοντέλο	ART-L29	VNS-L21
Κυκλοφορία	4 Μαρτίου, 2020	20 Απριλίου, 2016
System on Chip (SoC)	HUAWEI Kirin 710F	HUAWEI Kirin 650
CPU	Octa-core (4xCortex-A73 Based 2,2 GHz +4xCortex-A53 Based 1,7 GHz)	Octa-core (4x2.0 GHz Cortex-A53 & 4x1.7 GHz Cortex-A53)
GPU	4 x 650 MHz Mali G51-MP4	2 x 600 Mali-T830MP2
Συνολική RAM	4 GB (1833 MHz)	2 GB (1700 MHz)
Έκδοση Android	10 (API level 29)	7 (API level 24)
Wifi Hardware	Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot	Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot

Πίνακας 5.1: Συσκευές Android

5.1.3 Servers Ταξινόμησης

Για τις ανάγκες του server ταξινόμησης εικόνων έγινε χρήση των συσκευών με τα παρακάτω χαρακτηριστικά:

Τύπος Server	**Cloud (Σύστημα Α)**	**Edge (Σύστημα Β)**
Λειτουργικό Σύστημα	Windows 10 Pro 64-bit (Virtual Machine)	Windows 7 Ultimate 64-bit SP1
CPU	Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz.	Intel Core i5 2500K @ 3.30GHz
GPU	-	2047MB NVIDIA GeForce GTX 660
RAM	8 GB FPG @976Mhz	12.0GB Dual-Channel DDR3 @ 668MHz (9-9-9-24)
Disk	200 GB QEMU HARDDISK (SATA (SSD))	111GB KINGSTON SA400S37120G ATA Device (SATA-2 (SSD))

Πίνακας 5.2: Συστήματα Server

5.2 Μετρήσεις

Οι μετρήσεις στην εφαρμογή βασίζονται στη βιβλιοθήκη System και τη συνάρτηση nanoTime() της Java. Με αυτή εντοπίζεται η καθυστέρηση (latency) των διαδικασιών κλιμάκωσης-συμπίεσης, Inference Latency, Συνολικός Χρόνος.

Οι μετρήσεις στον Server βασίζονται στην συνάρτηση time.time() από την αντίστοιχη βιβλιοθήκη της γλώσσας Python και την αποστολή των πληροφοριών αυτών μέσω του Post Response.

Για να ληφθούν σωστά οι μετρήσεις χρειάστηκε αρχικώς να γίνουν κάποια warm up runs της συσκευής σε χρήση (CPU ή GPU), ώστε να αποφευχθεί η λειτουργία εξοικονόμησης ενέργειας που μπαίνει η συσκευή σε περίπτωση που βρισκόταν εκτός λειτουργίας. Επιπρόσθετα, για τις μετρήσεις δημιουργήθηκε ένα σύνολο εικόνων με ποικίλες αναλύσεις από το COCO dataset και για κάθε περίπτωση που εξετάσθηκε έγινε επανάληψη της μέτρησης 10 φορές.

5.2.1 Κλιμάκωση – Συμπίεση

Στο κομμάτι της κλιμάκωσης, τα μοντέλα ταξινόμησης εικόνων του Tensorflow απαιτούν διαστάσεις αρκετά μικρότερες από αυτές της αρχικής φωτογραφίας.
Συγκεκριμένα, για κάθε μοντέλο που χρησιμοποιείται από την εφαρμογή, απαιτούνται οι αναλύσεις εισόδου που φαίνονται στον Πίνακα 4.1.

Χρόνοι Συμπίεσης Συσκευών

Καθώς είναι επιθυμητή η εύρυθμη λειτουργία της εφαρμογής σε μεγάλο εύρος κινητών συσκευών, γίνεται έλεγχος των διαφορών στην ενεργοβόρα διαδικασία συμπίεσης – κλιμάκωσης για τις Συσκευές Α, Β.

Συσκευή Α

Batch	Mean (s)	Median (s)	90th Percentile (s)
1	0.095	0.070	0.121
10	0.258	0.211	0.328
20	0.464	0.398	0.601
40	0.843	0.753	1.119
60	1.223	0.996	1.675
80	1.585	1.361	1.977
100	2.001	1.789	2.470

Πίνακας 5.3: Μετρήσεις Κλιμάκωσης – Συμπίεσης (Συσκευή Α - πρόσφατη)

Συσκευή Β

Batch	Mean (s)	Median (s)	90th Percentile (s)
1	0.075	0.056	0.136
10	0.516	0.557	0.773
20	0.758	0.658	1.182
40	1.574	1.677	1.694
60	1.64	1.604	1.797
80	2.023	1.896	2.997
100	2.811	2.572	3.711

Πίνακας 5.4: Μετρήσεις Κλιμάκωσης – Συμπίεσης (Συσκευή Β - πενταετίας)

Από τις παραπάνω μετρήσεις (Πίνακες 5.3 – 5.4) είναι εμφανές πως αν και η Συσκευή Β είναι πενταετίας και με χειρότερα χαρακτηριστικά, τα καταφέρνει αρκετά καλά συγκρίνοντας τη με την πιο πρόσφατη κινητή συσκευή (Συσκευή Α). Αυτό δείχνει πως η εφαρμογή μπορεί να ανταπεξέλθει σε αξιοπρεπέστατο βαθμό από ένα μεγάλο εύρος συσκευών.

Φυσικά, εάν γίνει χρήση της εφαρμογής με χρήση Server Ταξινόμησης στην Άκρη του Δικτύου (Edge Server), είτε μελλοντικά υπάρχει υποδομή για καλύτερες ταχύτητες μεταφορτώσεως που θα επιτρέπουν την αποστολή ακατέργαστης φωτογραφίας χωρίς να δαπανά πολλούς πόρους, μπορεί να μεταφερθεί και το κομμάτι κλιμάκωσης - συμπίεσης στη μεριά του server, ώστε να ανεξαρτητοποιηθεί τελείως η εφαρμογή του κινητού.
Με αυτό τον τρόπο, δίνεται η δυνατότητα επέκτασης του κύκλου ζωής της εφαρμογής, ακόμη και σε συσκευές παλαιότητας δεκαετίας.

Κλιμάκωση

Κατά το στάδιο κλιμάκωσης, μειώνεται η ανάλυση της εικόνας στο επιθυμητό μέγεθος. Έπειτα από τη μείωση της ανάλυσης υπάρχει μεγάλο κέρδος χρόνου, καθώς επιτυγχάνεται σημαντική μείωση στο μέγεθος του αρχείου. Ενώ το αρχικό μέγεθος κινούνταν στα επίπεδα 1-3 Megabytes, τελικώς τα μεγέθη των αρχείων φτάνουν μεγέθη λίγων kilobytes (20-40 kb). Το γεγονός αυτό μειώνει αρκετά και τον χρόνο αποστολής της φωτογραφίας στον Server Ταξινόμησης.

Συμπίεση

Καθώς ήταν απαραίτητη η επίτευξη βέλτιστης απόδοσης του συνολικού χρόνου του συστήματος ταξινόμησης εικόνων, ήταν σημαντικό να υπάρξει περαιτέρω συμπίεση των εικόνων εφόσον αυτό ήταν δυνατό.
Για τον σκοπό αυτό έγινε προσπάθεια αναζήτησης της ελάχιστης ποιότητας με την οποία

δίνεται η μέγιστη δυνατή Εμπιστοσύνη (confidence) από τη συμπερασματολογία (inference) του μοντέλου Μηχανικής Μάθησης.

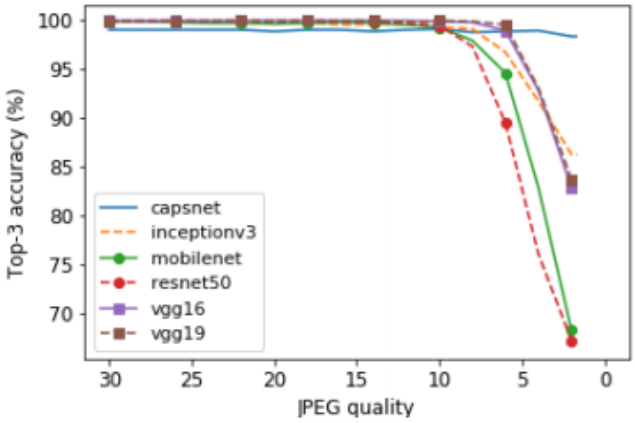
Ποιότητα	Εμπιστοσύνη
100	0.660242123
80	0.688666667
60	0.603
40	0.801
35	0.851666667
30	0.833
25	**0.853333333**
20	0.710333333
10	0.843333333
5	0.743
1	Λανθασμένο label

Πίνακας 5.5: Μέσος Όρος Confidence ανά ποιότητα εικόνας

Όπως φαίνεται στον Πίνακα 5.5, η επιθυμητή τιμή ποιότητας εικόνας προσεγγίζεται στο 25%, καθώς επιτυγχάνεται η μεγαλύτερη εμπιστοσύνη αποτελέσματος από τη συμπερασματολογία (inference) του μοντέλου.
Από τις μετρήσεις, παρατηρείται πως με την συμπίεση της φωτογραφίας, όχι μόνο θα κερδίσουμε χρόνο λόγω της μειωμένης ποιότητας της εικόνας κατά την αποστολή, αλλά θα πάρουμε ακόμη μεγαλύτερο confidence.
Οι μετρήσεις είναι σύμφωνες με συμπεράσματα για το κομμάτι σχέση συμπίεσης–εμπιστοσύνης, και επιβεβαιώνει πως στις περισσότερες των περιπτώσεων η Ποιότητα (Quality) μπορεί να φτάσει στο 15, χωρίς να επηρεάσει την εμπιστοσύνη στο αποτέλεσμα της συμπερασματολογίας [37].

Ακρίβεια

Το στάδιο κλιμάκωσης – συμπίεσης δείχνει να μην επηρεάζει τα αποτελέσματα που δίνουν τα μοντέλα στην ακρίβεια. Αλλαγές ξεκινούν να εμφανίζονται σε ποιότητα εικόνας κάτω του 2%.
Το συμπέρασμα αυτό υποστηρίζεται από έρευνα [38] σχετική με την υποβάθμιση των εικόνων από τύπους συμπίεσης, που δείχνει πως με τύπο συμπίεσης JPEG, η ακρίβεια (accuracy) δεν δείχνει να επηρεάζεται, μέχρι η ποιότητα συμπίεσης να φτάσει το 20 για πληθώρα μοντέλων (Εικόνα 5.6).

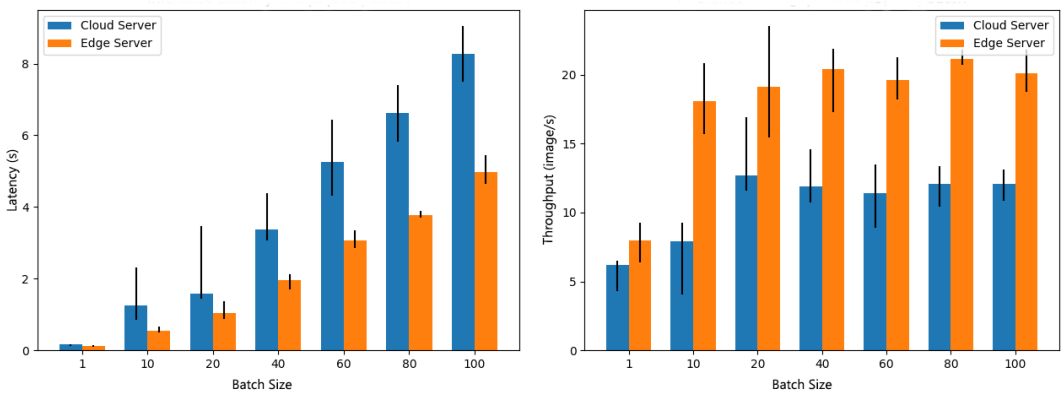


Εικόνα 5.6: Σύγκριση ακρίβειας ταξινόμησης διαφορετικών CNN αρχιτεκτονικών υπό συμπίεση JPEG για dataset φυσικών εικόνων [27].

5.2.2 Συμπερασματολογία

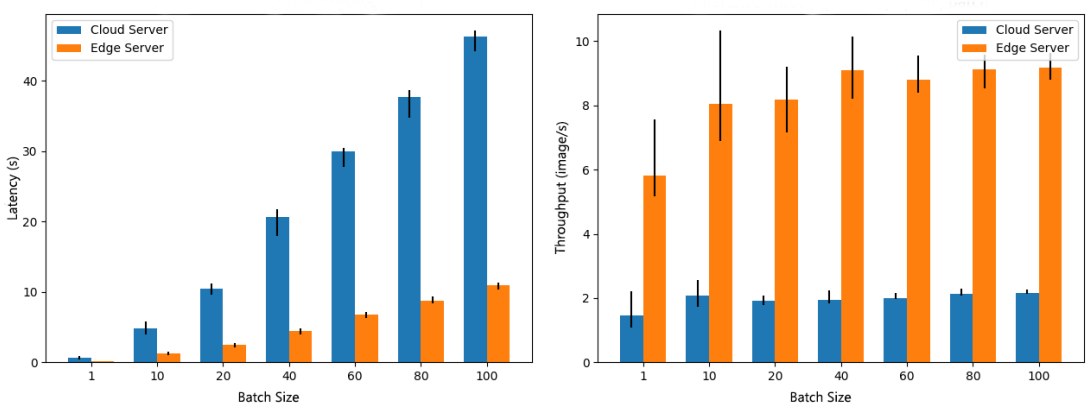
Για να αξιολογηθεί ο Server Ταξινόμησης ανά μοντέλο ταξινόμησης, παρατηρείται ο αντίστοιχος χρόνος συμπερασματολογίας (inference) για τις τοπολογίες των Συστημάτων Α (Cloud Server) και Β (Edge Server).
Στα χαρακτηριστικά των συστημάτων για τη συμπερασματολογία, το Σύστημα Β κάνει χρήση GPU, ενώ στο Σύστημα Α οι υπολογισμοί γίνονται στη CPU.
Η αξιολόγηση των μοντέλων EfficientNetB0, ResNetV50, InceptionV3 και XceptionNet που φαίνεται στα παρακάτω γραφήματα γίνεται βάσει της μετρικής 90th percentile για τους χρόνους συμπερασματολογίας (inference Latency) και διεκπεραιωτικής ικανότητας (Throughput).
Αναλυτικοί πίνακες για τις μετρήσεις κάθε μοντέλου μπορούν να βρεθούν στα Παραρτήματα.

EfficientNetB0 – Top 1 Accuracy = 77.1%



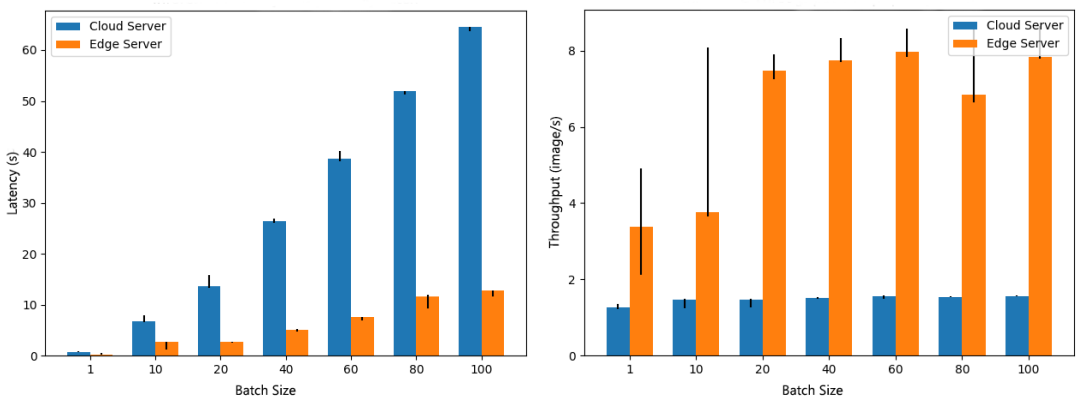
Εικόνα 5.7: Inference Latency - Throughput για το μοντέλο EfficientNetB0

ResNet50 – Top 1 Accuracy = 74.9%



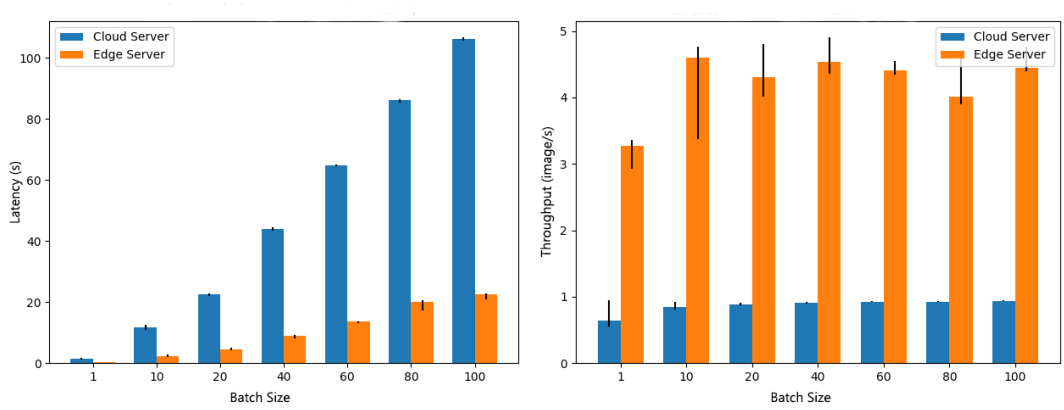
Εικόνα 5.8: Inference Latency - Throughput για το μοντέλο ResNet50

InceptionV3 – Top 1 Accuracy = 77,9%



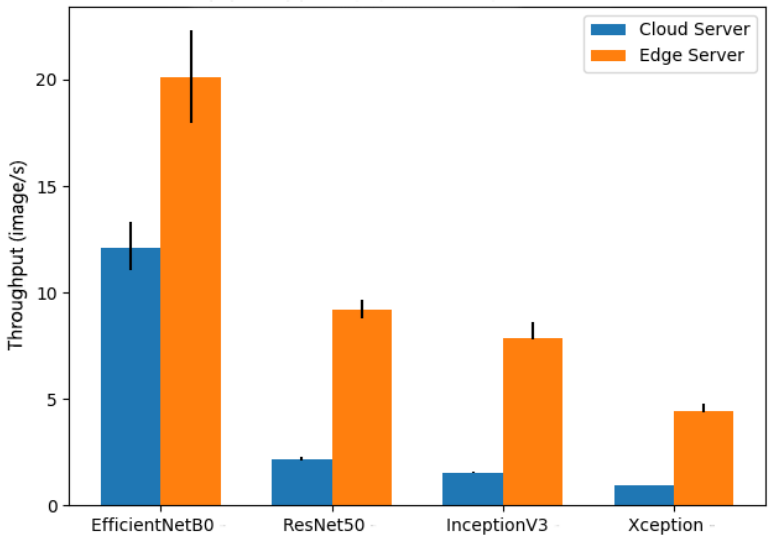
Εικόνα 5.9: Inference Latency - Throughput για το μοντέλο InceptionV3

Xception – Top 1 Accuracy = 79%



Εικόνα 5.10: Inference Latency - Throughput για το μοντέλο Xception

Σύμφωνα με τα αποτελέσματα συμπερασματολογίας των μοντέλων (Εικόνες 5.7 – 5.10), παρατηρείται πως το μοντέλο με τη μεγαλύτερη διεκπεραιωτική ικανότητα (throughput) είναι το EfficientNetB0. Συγκρίνοντας τους χρόνους μεταξύ των Συστημάτων Α και Β (Εικόνες 5.7 – 5.10) για το πιο αποδοτικό μοντέλο EfficientNetB0, παρατηρείται πως το σύστημα Β έχει throughput κοντά στο διπλάσιο του Συστήματος Α.
Το παραπάνω αποτέλεσμα είναι αναμενόμενο για το Σύστημα Β, καθώς γίνεται χρήση κάρτας γραφικών.



Εικόνα 5.11: Συγκρίσεις μεταξύ μοντέλων για Batch Size 100

Ακόμη, κάτι που διακρίνεται στα πιο απαιτητικά μοντέλα (Εικόνα 5.11), όπως τα InceptionV3 και XceptionNet, είναι πως όσο αυξάνονται οι παράμετροι του μοντέλου σε χρήση, τόσο μεγαλώνει και η διαφορά στο throughput μεταξύ των συστημάτων Α και Β. Η

διαφορά αυτή δημιουργείται λόγω της χρήσης κάρτας γραφικών στο Σύστημα Β.
Για παράδειγμα, το EfficentNetB0 με 5.3 εκ. παραμέτρους γίνεται 1.66 φορές αποδοτικότερο στο Σύστημα Β, ενώ τα μοντέλα ResNet50, InceptionV3, Xception (πάνω από 22 εκ. παράμετροι) βελτιώνονται κατά μέσο όρο 4.5 φορές περισσότερο όταν εισέρχονται στο Σύστημα Β.
Αυτό συμβαίνει καθώς οι GPUs είναι βελτιστοποιημένες για την εκπαίδευση μοντέλων τεχνητής νοημοσύνης και Βαθιάς Μάθησης και έχουν τη δυνατότητα να κάνουν ταυτόχρονα πολλαπλούς υπολογισμούς πινάκων. Τέλος, έχουν μεγάλο αριθμό πυρήνων, κάτι που επιτρέπει καλύτερο υπολογισμό πολλαπλών παράλληλων διεργασιών [39].

5.2.3 Συνολικός Χρόνος

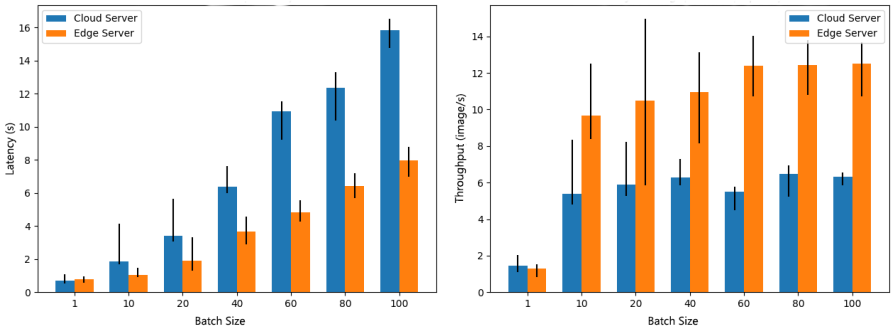
Για την αξιολόγηση του συνόλου του συστήματος θεωρείται ως Συνολικός Χρόνος, ο χρόνος από τη στιγμή δημιουργίας ή φόρτωσης εικόνας μέχρι και την εμφάνιση της εικόνας με την ετικέτα συμπερασματολογίας στο πλέγμα.

Αναλυτικότερα, ο Συνολικός Χρόνος περιλαμβάνει τους χρόνους:

1. Συμπίεσης.
2. Αποστολής στο server ταξινόμησης.
3. Διαβάσματος εικόνων από τον Server Ταξινόμησης.
4. Συμπερασματολογίας.
5. Επιστροφής αποτελεσμάτων και εμφάνισης στη συσκευή.

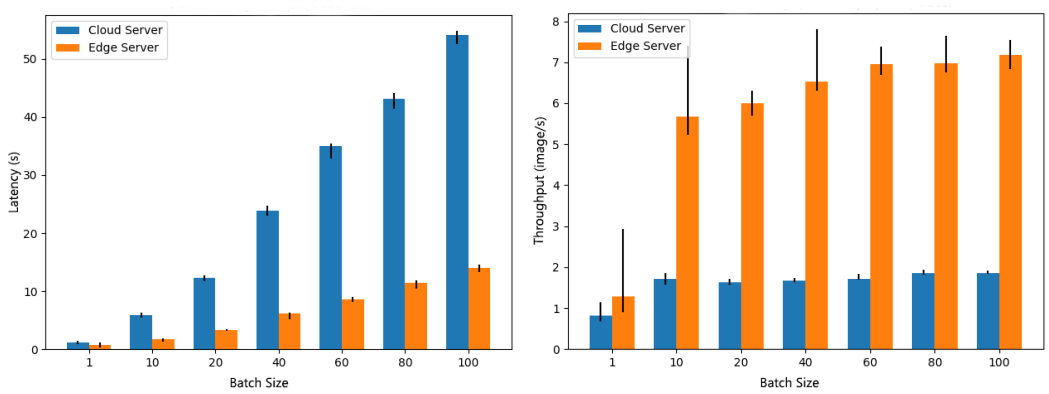
Όπως αναφέρθηκε και στην Ενότητα 5.4.2, έχουμε δύο υλοποιήσεις server με τα χαρακτηριστικά των Συστημάτων Α, Β. Εκτός από τις διαφορές στο τρόπο με τον οποίο επιχειρείται η συμπερασματολογία (inference) μεταξύ των συστημάτων, υπάρχουν και διαφορές στην τοπολογία. Το Σύστημα Α βρίσκεται στο νέφος (Cloud), ενώ το Σύστημα Β βρίσκεται στην άκρη του τοπικού δικτύου (Edge) που βρίσκεται η κινητή συσκευή.
Γνωρίζοντας τους χρόνους συμπερασματολογίας (inference) και τις διαφορές που υπάρχουν λόγω υλικού (CPU vs. GPU), γίνεται παρατήρηση των μετρήσεων (Εικόνες 5.12 – 5.15) του συνολικού χρόνου των μοντέλων EfficientNetB0, ResNetV50, InceptionV3 και XceptionNet:

EfficientNetB0 – Top 1 Accuracy = 77.1%



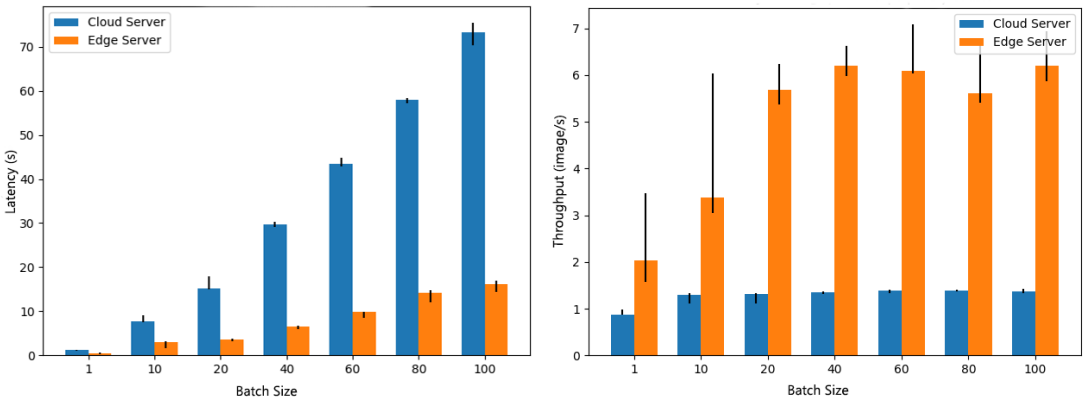
Εικόνα 5.12: Total Latency - Throughput για το μοντέλο EfficientNetB

ResNet50 – Top 1 Accuracy = 74.9%



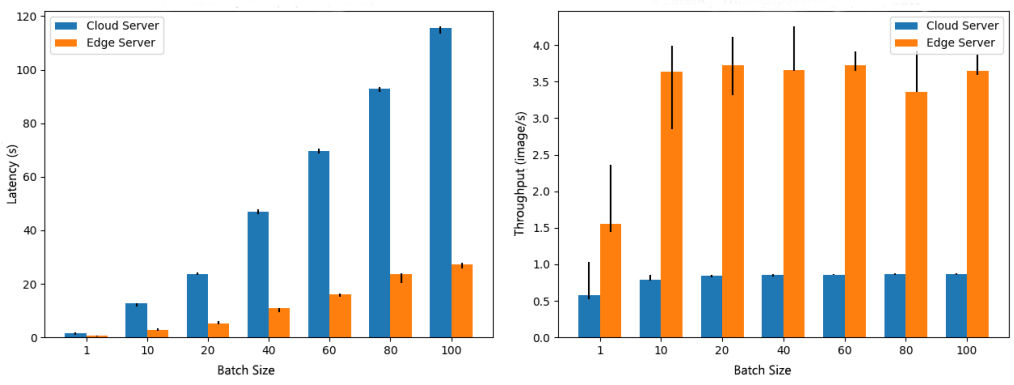
Εικόνα 5.13 : Total Latency - Throughput για το μοντέλο ResNet50

InceptionV3 – Top 1 Accuracy = 77,9%



Εικόνα 5.14: Total Latency - Throughput για το μοντέλο InceptionV3

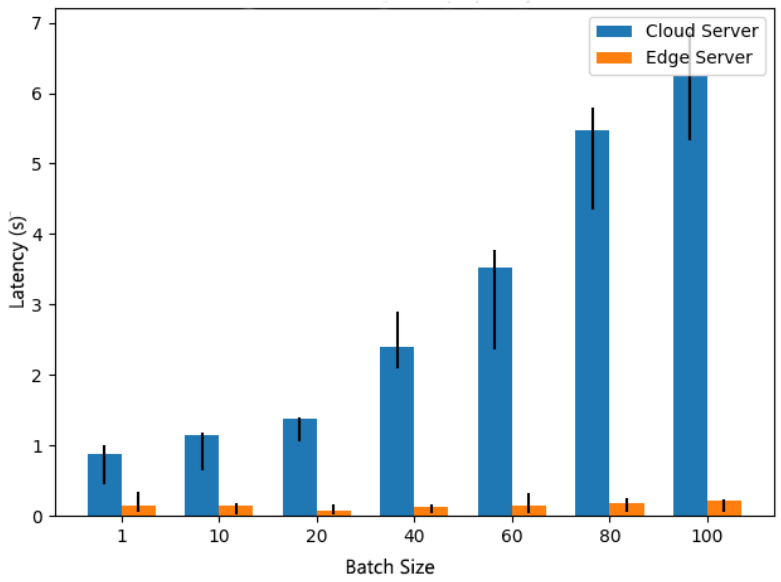
Xception – Top 1 Accuracy = 79%



Εικόνα 5.15: Total Latency - Throughput για το μοντέλο Xception

5.2.4 Χρόνος Μεταφόρτωσης

Στην προσπάθεια εύρεσης βέλτιστης τοπολογίας για την λειτουργία του συστήματος ταξινόμησης εικόνων, έγινε έλεγχος στις δύο εν λειτουργία τοπολογίες. Ο χρόνος αποστολής αποτελεσμάτων δεν μπόρεσε να μετρηθεί αποκλειστικά, όμως έγινε υπολογισμός μέσω του συνολικού χρόνου της εφαρμογής.
Καθώς είναι γνωστοί οι χρόνοι συμπίεσης, διαβάσματος αρχείων από server, ταξινόμησης, επιστροφής αποτελεσμάτων και ο συνολικός χρόνος, είναι δυνατή η εύρεση του χρόνου μεταφόρτωσης στον server αφαιρώντας από τον συνολικό τα προηγούμενα.
Ο χρόνος επιστροφής αποτελεσμάτων αν και δεν είναι γνωστός δεν τον λαμβάνουμε υπ' όψη καθώς ακόμη και για ταξινόμηση εκατό εικόνων, επιστρέφεται αρχείο περίπου 200B που ακόμη και στα πιο αργά δίκτυα η απάντηση είναι σχεδόν στιγμιαία (μικρότερη του 1ms).



Εικόνα 5.16: Upload Latency ανά μέγεθος batch

Ο χρόνος μεταφόρτωσης των εικόνων στον Server Νέφους (Cloud Server) δείχνει να αυξάνεται ανάλογα με το πλήθος των εικόνων που αποστέλλονται ταυτόχρονα (Εικόνα 5.16).
Στον Server στην άκρη του δικτύου (Edge Server) παρατηρείται μικρή διακύμανση στον χρόνο μεταφόρτωσης, σχεδόν ανεξάρτητα από το μέγεθος του batch εικόνων (Εικόνα 4.16). Αυτό μπορεί να εξαρτάται από πολλούς παράγοντες, όπως το RTT latency με τον server ταξινόμησης, διαχείριση άλλων συσκευών στο δίκτυο ή παρεμβολή άλλων σημάτων στην συχνότητα 2.4GHz.
Είναι σίγουρο πως ο χρόνος μεταφόρτωσης δείχνει αρκετά μεγάλος στην περίπτωση του Server Νέφους, καθώς στο δίκτυο από το οποίο στάλθηκαν τα batch εικόνων, είχε ως peak

ταχύτητα upload τα 0.77Mbps. Εάν η ταχύτητα μεταφόρτωσης της κινητής συσκευής μεγαλώσει, αναμένεται και κλείσιμο της ψαλίδας μεταξύ των χρόνων Cloud και Edge Server.
Φυσικά, όσο μεγάλη ταχύτητα πρόσβασης στο διαδίκτυο και να υπάρχει, η διαδρομή της πληροφορίας θα είναι μικρότερη προς τον Edge, σε σχέση με τον Cloud Server.

Κεφάλαιο 6: Επίλογος

Σε αυτό το κεφάλαιο αναλύονται τα συμπεράσματα που εξήχθησαν από τη δημιουργία και μελέτη του AI Gallery App με υποβοήθηση όπως και δίνονται μερικά παραδείγματα μελλοντικών επεκτάσεων για την εφαρμογή.

6.1 Συμπεράσματα

Οι βασικοί στόχοι της εργασίας ήταν πρώτα η επίτευξη της μεγαλύτερης ακρίβειας στην ταξινόμηση και έπειτα η μεγαλύτερη διεκπεραιωτική ικανότητα του συστήματος. Καθώς η εφαρμογή AI Gallery δεν είναι εφαρμογή πραγματικού χρόνου, δόθηκε μεγαλύτερη προσοχή στην ακρίβεια του αποτελέσματος ταξινόμησης και όχι τόσο στην καθυστέρηση (latency) των μοντέλων και την διεκπαιρεωτική ικανότητα του συστήματος.
Παρατηρώντας τα μοντέλα που μετρήθηκαν μέσω του συστήματος AI Gallery App, παρατηρείται πως το μοντέλο που δίνει την καλύτερη σχέση ακρίβειας και απόδοσης είναι το EfficientNetB0 (ακρίβεια 71.3%). Δηλαδή, το EfficientNetB0 δείχνει μεγάλη διεκπεραιωτική ικανότητα στην ταξινόμηση εικόνων και αποκλειστικά στο κομμάτι της συμπερασματολογίας, αλλά και στην εξαγωγή αποτελέσματος από την στιγμή δημιουργίας της εικόνας μέχρι και την ταξινομημένη απεικόνισή της στην Android εφαρμογή. Το μοντέλο δείχνει να πλησιάζει ένα βέλτιστο throughput στον Cloud Server (Σύστημα Α) για πακέτο εικόνων ίσο με 20, ενώ στον Edge Server (Σύστημα Β) το ίδιο συμβαίνει για πακέτο εικόνων ίσο με 10.
Όπως αναφέρθηκε, αν και ο πρωταρχικός στόχος ήταν η ακρίβεια των μοντέλων, κάνοντας χρήση πιο απαιτητικών μοντέλων όπως τα NASNetLarge και EfficientNetB7, παρατηρήθηκε μέχρι και δέκα φορές μεγαλύτερη καθυστέρηση στη συμπερασματολογία. Παράλληλα, για την δυνατότητα χρήσης των πιο απαιτητικών μοντέλων, η συμπερασματολογία επιλέχθηκε να γίνει εκτός συσκευής. Αυτό επέτρεψε με μια μικρή σχετικά καθυστέρηση, να μπορούν να χρησιμοποιηθούν πιο απαιτητικά μοντέλα, λόγω της επεκτάσιμης φύσης του απομακρυσμένου server, αλλά και την αποσυμφόρηση της κινητής συσκευής από διεργασίες που είναι απαιτητικές για τα χαρακτηριστικά τους και καταναλώνουν αρκετή ενέργεια.
Επίσης, μεταξύ των δύο επιλογών server, διαπιστώνεται πως η χρήση server στα άκρα του δικτύου (edge server) είναι αποδοτικότερη από τη χρήση server στο νέφος (cloud server), καθώς η καθυστέρηση δείχνει να είναι μικρότερη. Βέβαια, με την επέκταση του συστήματος είναι σίγουρο πως η λύση στα άκρα του δικτύου τείνει να κοστίζει παραπάνω, και θα χρειαστεί ειδική μελέτη για τη διαχείριση μεγάλου όγκου χρηστών από το δίκτυο.
Σημαντικό αναφοράς είναι πως σύμφωνα με τις μετρήσεις της καθυστέρησης της συμπερασματολογίας για τα μοντέλα που ελέγχθηκαν, η διεκπεραιωτική ικανότητα δείχνει να μεγιστοποιείται για batch size μεταξύ 10-20 εικόνων, οπότε φαίνεται πως υπάρχει δυνατότητα να γίνει βελτιστοποίηση στον τρόπο αποστολής, με τις εικόνες να μπορούν να αποσταλούν αποδοτικότερα στο server ταξινόμησης σε πακέτα των 10-20 εικόνων τη φορά.
Εκτός από τη θετική πλευρά του απομακρυσμένου server, με την υλοποίηση ενός τέτοιου συστήματος, είναι σίγουρο πως εγείρονται ερωτήματα σχετικά με την ασφάλεια των

δεδομένων που μεταδίδονται μεταξύ της εφαρμογής και του server, αλλά και σχετικά με το γεγονός ότι απαιτείται συνεχής σύνδεση στο διαδίκτυο για τις ανάγκες της ταξινόμησης. Τέλος, κάνοντας έναν έλεγχο συμβατότητας με παλαιότερες και νεότερες συσκευές, παρατηρήθηκε πως η υλοποίηση μπορεί να υποστηρίξει ακόμη και κινητές συσκευές πενταετίας. Αυτό συμβαίνει διότι ο υπολογιστικός φόρτος της συμπερασματολογίας έχει μετατεθεί πλέον στον εξυπηρετητή, που βρίσκεται εκτός της κινητής συσκευής.

6.2 Μελλοντικές Επεκτάσεις

Συγκρίσεις επικοινωνίας εφαρμογής – server με διαφορετικούς τύπους σύνδεσης.

Ο τρόπος με τον οποίο ελέγχθηκε το σύστημα ήταν, η εφαρμογή να στέλνει τις εικόνες και μέσω WiFi στον απομακρυσμένο server. Θα μπορούσε να γίνει έλεγχος και με διαφορετικές συνδεσιμότητες όπως 3G, 4G, 5G και να γίνει σύγκριση μεταξύ τους, ώστε να βρεθεί η πιο αποδοτική.

Διαχείριση διαμοιρασμού requests σε πολλαπλούς edge servers.

Το σύστημα μπορεί να επεκταθεί με την αύξηση των edge servers, λαμβάνοντας ακόμη περισσότερα requests την ίδια στιγμή, εξυπηρετώντας περισσότερους χρήστες σε μικρότερο χρόνο. Ακόμη, το σύστημα μπορεί να βελτιώσει το throughput για κάθε μοναδικό χρήστη, δημιουργώντας πακέτα εικόνων εκτιμώντας τον αριθμό των edge servers που είναι αδρανείς.

Χρήση συστήματος για αναγνώριση αντικειμένων, προσώπων, στάσης σώματος και τμηματοποίησης εικόνας.

Καθώς το μέγεθος των νευρωνικών δικτύων που μπορούν να αξιοποιηθούν είναι αρκετά μεγάλο λόγω της χρήσης εξωτερικού server για την συμπερασματολογία, το σύστημα μπορεί να αξιοποιηθεί αναγνωρίζοντας πιο περίπλοκα tasks Όρασης Υπολογιστών, όπως είναι ο εντοπισμός αντικειμένων και προσώπων, η εκτίμηση στάσης σώματος, η τμηματοποίηση εικόνων κ.α.
Πιο συγκεκριμένα, καθώς τα μοντέλα αυτά δεν έχουν βελτιστοποιηθεί κατάλληλα για εκτέλεση στη συσκευή, θα ήταν καλή πρακτική η προσθήκη τους στο server στην Άκρη του Δικτύου.

Αξιοποίηση πληροφοριών χρήστη για δημιουργία προσωποποιημένων album βάσει φωτογραφιών του χρήστη.

Το σύστημα μπορεί να αξιοποιήσει την δυνατότητα που έχει να εξάγει πληροφορίες για το περιεχόμενο των εικόνων, και να κάνει προτάσεις για δημιουργία θεματικών album φωτογραφιών σύμφωνα με τοποθεσίες ή αντικείμενα που έχουν αναγνωριστεί από τη συμπερασματολογία.

Εμφάνιση πληροφοριών βάσει φωτογραφιών σε σημεία ενδιαφέροντος

Το σύστημα μπορεί να αξιοποιήσει την δυνατότητα που έχει να εξάγει πληροφορίες για το περιεχόμενο των εικόνων, και να δίνει πληροφορίες για σημεία ενδιαφέροντος όπως και προτάσεις για σημεία που μπορεί να επισκεφθεί κανείς στη συνέχεια.

Επέκταση με εκτέλεση της συμπερασματολογίας στο κινητό

Στο σύστημα μπορεί να προστεθεί ένα ελαφρύ-βελτιστοποιημένο μοντέλο στη συσκευή, που θα γίνεται χρήση όταν η συσκευή βρίσκεται εκτός σύνδεσης. Μόλις η συσκευή βρίσκει σύνδεση στο διαδίκτυο, η ακρίβεια της συμπερασματολογίας θα μπορεί να επιστρέφει στην αρχική και θα διορθώνει εάν είναι απαραίτητο, το αποτέλεσμα της εκτός σύνδεσης συμπερασματολογίας.

Χρήση Cloud ή Edge Server για κατανεμημένη εκπαίδευση μοντέλων εικόνας

Έχοντας ήδη μια λειτουργική δομή ισχυρής υπολογιστικής δύναμης cloud/edge server, συνδεδεμένη με κινητές συσκευές, δίνεται η δυνατότητα επέκτασης σε δίκτυο κατανεμημένης εκπαίδευσης μοντέλων Βαθιάς Μάθησης. Το δίκτυα αυτά χρησιμοποιώντας την διασύνδεση με το κινητό, μπορούν να δημιουργήσουν εύκολα dataset για το κομμάτι εκπαίδευσης και αξιολόγησης.

Προσωποποιημένη Εφαρμογή κρισιμότητας τύπων καρκίνου του δέρματος.

Ο καρκίνος του δέρματος έχει περιγραφεί ως μια ασθένεια που αρκετοί ασθενείς αγνοούν, αναβάλλοντας την επίσκεψη στο δερματολόγο, αναβάλλοντας μαζί και τον έλεγχο και της έγκαιρης πρόγνωσης. Η έγκαιρη πρόγνωση του καρκίνου έχει γίνει απαραίτητη, καθώς μπορεί να έχει ως αποτέλεσμα την καλύτερη κλινική διαχείριση των ασθενών [40]. Η σοβαρότητα της ταξινόμησης των τύπων καρκίνου του δέρματος σε ομάδες υψηλού και χαμηλού ρίσκου μπορούν να δώσουν μια καλή και άμεση απάντηση για ασθενείς που αναβάλλουν συνεχώς το δερματολόγο, κρούοντας τον κώδωνα του κινδύνου όπου αυτό είναι αναγκαίο, συστήνοντας μια επίσκεψη στον ειδικό. Ένας καλός τρόπος να γίνει αυτό είναι η Μηχανική Μάθηση και οι κινητές συσκευές που έχουμε όλη μέρα στα χέρια μας. Το AI Gallery App είναι μια καλή βάση για δημιουργία προσωποποιημένης εφαρμογής χρήστη, που θα μπορεί με χρήση του ανάλογου μοντέλου Βαθιάς Μάθησης να αναγνωρίζει

(εάν υπάρχει) και έπειτα να καταγράφει τα ποσοστά επικινδυνότητας κάποιου καρκινώματος κατά την εξέλιξή του σε μέγεθος, χρώμα και σχήμα.

Παραρτήματα

Α. Μετρήσεις Χρόνου Συμπερασματολογίας

EfficientNetB0 – Top 1 Accuracy = 77.1%

Σύστημα Α

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.124	0.140	0.140	0.162	0.171	6.173
10	0.843	1.243	1.094	1.259	2.312	7.943
20	1.453	2.040	1.453	1.578	3.468	12.674
40	3.078	3.500	3.356	3.372	4.374	11.862
60	4.312	5.172	4.594	5.259	6.437	11.409
80	5.828	6.362	5.925	6.628	7.406	12.070
100	7.5	8.049	7.578	8.278	9.046	12.080

Πίνακας Α1: Μετρήσεις για το inference στο μοντέλο EfficientNetB0 – Σύστημα Α

Σύστημα Β

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.123	0.140	0.101	0.105	0.212	9.524
10	0.487	0.511	0.526	0.554	0.645	18.051
20	0.967	1.015	1.021	1.047	1.236	19.102
40	1.721	1.912	1.922	1.962	2.342	20.387
60	2.489	2.753	2.868	3.062	3.517	19.595
80	3.417	3.678	3.752	3.781	3.981	21.158
100	4.481	4.738	4.706	4.975	5.566	20.101

Πίνακας Β1: Μετρήσεις για το inference στο μοντέλο EfficientNetB0 – Σύστημα Β

ResNet50 – Top 1 Accuracy = 74.9%

Σύστημα Α

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.452	0.593	0.624	0.681	0.934	1.468
10	3.921	4.73	4.781	4.839	5.767	2.067
20	9.568	10.374	10.15	10.484	11.176	1.908
40	17.891	19.312	19.421	20.67	21.721	1.935
60	27.785	28.482	28.627	29.903	30.41	2.006
80	34.782	37.2	35.247	37.627	38.654	2.126
100	44.189	45.116	45.42	46.317	47.113	2.159

Πίνακας Α2: Μετρήσεις για το inference στο μοντέλο ResNet50 – Σύστημα Α

Σύστημα Β

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.132	0.177	0.163	0.172	0.193	5.814
10	0.968	1.095	1.175	1.243	1.453	8.045
20	2.172	2.2	2.304	2.441	2.789	8.193
40	3.939	4.336	4.352	4.401	4.867	9.089
60	6.286	6.584	6.673	6.818	7.156	8.800
80	8.356	8.674	8.561	8.776	9.365	9.116
100	10.379	10.7676	10.697	10.905	11.376	9.170

Πίνακας Β2 Μετρήσεις για το inference στο μοντέλο ResNet50– Σύστημα Β

InceptionV3 – Top 1 Accuracy = 77,9%

Σύστημα Α

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	0.778	0.781	0.796	0.734	0.823	1.256
10	6.928	6.781	6.845	6.671	7.99	1.461
20	13.847	13.499	13.657	13.312	15.87	1.464
40	26.443	26.402	26.486	26.15	26.903	1.510
60	38.741	38.645	38.704	38.211	40.147	1.550
80	51.725	51.786	51.965	51.187	51.967	1.539
100	64.026	63.88	64.488	63.619	64.518	1.551

Πίνακας Α3: Μετρήσεις για το inference στο μοντέλο InceptionV3 – Σύστημα Α

Σύστημα Β

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	0.2436	0.2095	0.295	0.204	0.471	3.390
10	1.6196	1.316	2.663	1.237	2.743	3.755
20	2.627	2.62	2.677	2.534	2.757	7.471
40	4.9742	4.9435	5.159	4.799	5.197	7.753
60	7.2037	7.0795	7.528	6.997	7.66	7.970
80	10.5137	10.097	11.679	9.254	12.031	6.850
100	12.3499	12.3795	12.752	11.574	12.836	7.842

Πίνακας Β3: Μετρήσεις για το inference στο μοντέλο InceptionV3 – Σύστημα Β

XceptionNet – Top 1 Accuracy = 79%

Σύστημα Α

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	1.156	1.347	1.547	1.052	1.815	0.646
10	11.437	11.389	11.871	10.871	12.467	0.842
20	22.015	22.214	22.498	21.871	22.816	0.889
40	43.547	43.879	43.984	43.261	44.626	0.909
60	64.617	64.521	64.668	64.398	65.074	0.928
80	86.023	85.872	86.245	85.436	86.421	0.928
100	106.14	105.781	106.192	105.657	106.672	0.942

Πίνακας Α4: Μετρήσεις για το inference στο μοντέλο Xception – Σύστημα Α

Σύστημα Β

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	0.3055	0.3025	0.306	0.298	0.341	3.268
10	2.2246	2.1525	2.176	2.099	2.966	4.596
20	4.3452	4.2295	4.645	4.167	4.991	4.306
40	8.6094	8.6635	8.831	8.155	9.182	4.529
60	13.4628	13.4625	13.63	13.198	13.794	4.402
80	18.4556	17.898	19.944	17.383	20.556	4.011
100	22.1906	22.3615	22.477	20.994	22.736	4.449

Πίνακας Β4: Μετρήσεις για το inference στο μοντέλο Xception – Σύστημα Β

Β. Μετρήσεις Συνολικού χρόνου

EfficientNetB0 – Top 1 Accuracy = 77.1%

Σύστημα Α

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.546	0.778	0.639	0.681	1.092	1.565
10	1.679	2.148	1.887	1.857	4.145	5.299
20	3.062	3.577	3.253	3.395	5.643	6.148
40	5.981	6.324	6.220	6.390	7.646	6.431
60	9.234	9.909	9.229	10.923	11.529	6.501
80	10.367	12.454	11.609	12.355	13.292	6.891
100	14.765	15.683	15.010	15.863	16.535	6.662

Πίνακας Α5: Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο EfficientNetB0 – Σύστημα Α

Σύστημα Β

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.565	0.618	0.626	0.774	0.965	1.597
10	0.913	0.971	1.026	1.034	1.467	9.747
20	1.319	1.627	1.770	1.905	3.312	11.299
40	2.912	3.141	3.399	3.656	4.567	11.768
60	4.267	4.549	4.718	4.841	5.572	12.717
80	5.676	5.987	6.093	6.427	7.2	13.130
100	6.967	7.718	7.687	7.983	8.787	13.009

Πίνακας Β5: Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο EfficientNetB0 – Σύστημα Β

ResNet50 – Top 1 Accuracy = 74.9%

Σύστημα Α

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.878	1.231	1.123	1.222	1.473	0.890
10	5.421	6.095	5.574	5.836	6.345	1.794
20	11.667	12.110	11.950	12.301	12.775	1.674
40	11.956	22.536	22.484	23.888	24.672	1.779
60	32.783	33.220	33.262	34.966	35.413	1.804
80	41.458	43.292	41.930	43.154	44.056	1.908
100	52.446	52.750	52.852	54.102	54.789	1.892

Πίνακας Α6: Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο ResNet50 – Σύστημα Α

Σύστημα Β

Batch Size	Min	Average	Median	90th Percentile	Max	90th Percentile Throughput
1	0.342	0.543	0.727	0.780	1.121	1.376
10	1.352	1.594	1.714	1.762	1.913	5.834
20	3.873	4.251	3.093	3.337	3.612	6.466
40	5.124	5.604	5.868	6.134	6.342	6.817
60	8.123	8.418	8.562	8.635	8.978	7.008
80	10.478	11.021	10.941	11.461	11.867	7.312
100	13.241	13.786	13.717	13.952	14.652	7.290

Πίνακας Β6 Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο ResNet50 – Σύστημα Β

InceptionV3 – Top 1 Accuracy = 77,9%

Σύστημα Α

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	1.0712571	1.0498	1.142	1.014	1.153	0.876
10	7.8301429	7.624	7.673	7.553	9.07	1.303
20	15.552857	15.169	15.271	15.047	17.984	1.310
40	29.603714	29.548	29.708	29.162	30.37	1.346
60	43.389143	43.083	43.451	42.936	44.891	1.381
80	57.876714	57.97	58.022	57.248	58.371	1.379
100	72.368571	71.943	73.323	70.419	75.449	1.364

Πίνακας Α7: Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο InceptionV3 – Σύστημα Α

Σύστημα Β

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	0.3862	0.3505	0.491	0.288	0.633	2.037
10	2.0824	1.831	2.964	1.658	3.281	3.374
20	3.358	3.269	3.515	3.207	3.721	5.690
40	6.2984	6.275	6.456	6.042	6.683	6.196
60	9.2126	9.208	9.841	8.47	9.93	6.097
80	13.1936	13.074	14.255	11.96	14.81	5.612
100	**15.6621**	**15.788**	**16.124**	**14.402**	**17.039**	6.202

Πίνακας Β7: Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο InceptionV3 – Σύστημα Β

XceptionNet – Top 1 Accuracy = 79%

Σύστημα Α

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	1.441	1,452	1.742	0.972	1.891	0.574
10	12.329	12.388	12.644	11.678	12.785	0.791
20	23.74	23.655	23.676	23.251	24.246	0.845
40	46.816	46.521	46.979	46.105	47.898	0.851
60	69.627	69.548	69.675	68.864	70.533	0.861
80	92.611	91.863	92.778	91.649	93.527	0.862
100	114.778	114.781	115.683	113.577	116.182	0.864

Πίνακας Α8 Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο Xception – Σύστημα Α

Σύστημα Β

Batch Size	Average	Median	90th Percentile	Min	Max	90th Percentile Throughput
1	0.5339	0.4965	0.642	0.423	0.693	1.558
10	2.7475	2.692	2.753	2.504	3.516	3.632
20	5.2353	5.1865	5.375	4.866	6.035	3.721
40	10.2273	10.1945	10.947	9.401	10.968	3.654
60	15.8228	15.6995	16.123	15.345	16.467	3.721
80	21.9518	21.522	23.803	20.39	23.842	3.361
100	26.8781	27.072	27.387	25.848	27.86	3.651

Πίνακας Β8: Μετρήσεις για το Συνολικό Χρόνο στο μοντέλο Xception – Σύστημα Β

Αναφορές

[1] IBM Cloud Education (2020). *What is machine learning?* https://www.ibm.com/cloud/learn/machine-learning (Πρόσβαση: 8 Απριλίου 2021).

[2] Jason Brownlee (2016). *Supervised and Unsupervised Machine Learning Algorithms*.https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ (Πρόσβαση: 12 Μαΐου 2021).

[3] Ian J. Goodfellow, Yoshua Bengio and Aaron C. Courville (2016). *Deep Learning*. MIT Press.

[4] *Reinforcement Learning: What_is, Algorithms, Applications, Example*. https://www.guru99.com/reinforcement-learning-tutorial.html (Πρόσβαση: 12 Μαΐου, 2021).

[5] Shaw G.L. (1986). *Donald Hebb: The Organization of Behavior*. Brain Theory, Springer.

[6] Jaspreet (2014). *A Concise History of Neural Networks*. https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec (Πρόσβαση: 12 Μαΐου, 2021).

[7] Carl Felstiner (2019). *Alpha-Beta Pruning*. Whitman College.

[8] Kavish Sanghvi (2020). *Image Classification Techniques*. https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac (Πρόσβαση: 12 Μαΐου, 2021).

[9] Eren Gölge (2015). *Brief History of Machine Learning*. https://erogol.com/brief-history-machine-learning (Πρόσβαση: 12 Μαΐου 2021).

[10] Jason_Brownlee (2019). *A Gentle Introduction to Computer Vision*. https://machinelearningmastery.com/what-is-computer-vision (Πρόσβαση: 12 Μαΐου 2021).

[11] Niall O' Mahony, Sean Campbell, Anderson Carvalho et al. (2020). *Deep Learning vs. Traditional Computer Vision*. Advances in Computer Vision. CVC 2019. Advances in Intelligent Systems and Computing, vol 943. Springer, Cham.

[12] Google Developers. *ML Practicum: Image Classification*. https://developers.google.com/machine-learning/practica/image-classification (Πρόσβαση: 22 Ιουνίου 2021).

[13] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang et al. (2017). *Efficient Processing of Deep Neural Networks: A Tutorial and Survey*. Proc. IEEE, 105(12), 2295–2329.

[14] Zhi Zhou, Xu Chen, En Li et al. (2019). *Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing*. Proc. IEEE, 107(8), 1738–1762.

[15] Ji Wang, Bokai Cao, Philip S. Yu et al. (2018). *Deep Learning towards Mobile Applications*. In 38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018 (pp. 1385–1393). IEEE Computer Society.

[16] Yunbin Deng (2019). *Deep Learning on Mobile Devices - A Review*. SPIE Defense + Commercial Sensing, Invited Paper, Baltimore, MD.

[17] insideBIGDATA Editorial Team (2019). *Machine Learning Deployment Options: in the Cloud vs. at the Edge*. https://insidebigdata.com/2019/02/13/machine-learning-deployment-options-in-the-cloud-vs-at-the-edge/ (Πρόσβαση: 28 Απριλίου 2021).

[18] Android Developers. *Meet Android Studio*. https://developer.android.com/studio/intro (Πρόσβαση: 28 Απριλίου 2021).

[19] The Editors of Encyclopaedia Britannica(2020). Android operating system. https://www.britannica.com/technology/Android-operating-system (Πρόσβαση: 28 Απριλίου 2021)

[20] *What is Java? Definition, Meaning & Features of Java_Platforms*. https://www.guru99.com/java-platform.html (Πρόσβαση: 28 Απριλίου 2021).

[21] *Java - Overview*. https://www.tutorialspoint.com/java/java_overview.htm (Πρόσβαση: 28 Απριλίου 2021).

[22] *Picasso*. https://square.github.io/picasso (Πρόσβαση: 28 Απριλίου 2021).

[23] Vrijraj Singh (2018). *Introduction to Firebase*. https://medium.com/codingurukul/introduction-to-firebase-f9f6ccc8a785 (Πρόσβαση: 28 Απριλίου 2021).

[24] *Firebase*. https://firebase.google.com (Πρόσβαση: 28 Απριλίου 2021)

[25] *What is Python? Executive Summary*. https://www.python.org/doc/essays/blurb (Πρόσβαση: 28 Απριλίου 2021).

[26] GeeksforGeeks (2019). *History of Python*. https://www.geeksforgeeks.org/history-of-python/ (Πρόσβαση: 28 Απριλίου 2021).

[27] *FastAPI*. https://fastapi.tiangolo.com (Πρόσβαση: 28 Απριλίου 2021).

[28] *FastAPI Github_Releases*. https://github.com/tiangolo/fastapi/releases?after=0.2.0 (Πρόσβαση: 28 Απριλίου 2021).

[29] Karl Hughes (2020). Build and Secure an API in Python with_FastAPI. https://developer.okta.com/blog/2020/12/17/build-and-secure-an-api-in-python-with-fastapi (Πρόσβαση: 28 Απριλίου 2021).

[30] *Uvicorn*. https://www.uvicorn.org (Πρόσβαση: 28 Απριλίου 2021).

[31] (2018). *What exactly is TensorFlow?* https://medium.datadriveninvestor.com/what-exactly-is-tensorflow-80a90162d5f1 (Πρόσβαση: 28 Απριλίου 2021).

[32] *What is TensorFlow? How it Works? Introduction & Architecture*. https://www.guru99.com/what-is-tensorflow.html (Πρόσβαση: 28 Απριλίου 2021).

[33] Android Reference *AsyncTask class*. https://developer.android.com/reference/android/os/AsyncTask (Πρόσβαση: 28 Απριλίου 2021).

[34] Android Reference. *Bitmap.class*. https://developer.android.com/reference/android/graphics/Bitmap (Πρόσβαση: 28 Απριλίου 2021).

[35] Android Reference. *Thread class*. https://developer.android.com/reference/java/lang/Thread (Πρόσβαση: 28 Απριλίου 2021).

[36] Keras API reference. *Keras Applications*. https://keras.io/api/applications (Πρόσβαση: 28 Απριλίου 2021).

[37] Matt Poyser, Amir Atapour Abarghouei and Toby P. Breckon (2020). *On the Impact of Lossy Image and Video Compression on the Performance of Deep Convolutional Neural Network Architectures*. In 25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021 (pp. 2830–2837). IEEE.

[38] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya et al. (2018). *Effects of Degradations on Deep Neural Network Architectures*. CoRR, abs/1807.10108.

[39] Jason_Dsouza (2020). *What is a GPU and do you need one in Deep Learning?* https://towardsdatascience.com/what-is-a-gpu-and-do-youneed-one-in-deep-learning-718b9597aa0d (Πρόσβαση: 28 Απριλίου 2021).

[40] Konstantina Kourou, Themis P. Exarchos, Konstantinos P. Exarchos et al. (2015) *Machine learning applications in cancer prognosis and prediction*. Computational and Structural Biotechnology Journal, vol. 13, pp. 8-17.