ΠΑΝΕΠΙΣΤΉΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ

ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΑΞΙΝΟΜΗΣΗ ΕΙΚΟΝΩΝ ΜΕ ΜΕΘΟΔΟΥΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

ΜΠΡΑΧΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΜ 41667

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΔΗΜΗΤΡΙΟΣ ΚΑΝΤΖΟΣ

ΑΘΗΝΑ 2023

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΑΞΙΝΟΜΗΣΗ ΕΙΚΟΝΩΝ ΜΕ ΜΕΘΟΔΟΥΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

ΜΠΡΑΧΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΜ 41667

Η Πτυχιακή Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Δ. Κάντζος **(Καθηγητής)**	**Ε. Α. Λελίγκου** **(Αν. Καθηγήτρια)**	**Γ. Νικολάου** **(Λέκτορας)**
Dimitrios Kantzos Digitally signed by Dimitrios Kantzos Date: 2023.07.18 14:25:56 +03'00'	Eleni Aikaterini Leligkou Digitally signed by Eleni Aikaterini Leligkou Date: 2023.07.19 19:37:25 +03'00'	GRIGORIOS NIKOLAOU Digitally signed by GRIGORIOS NIKOLAOU Date: 2023.07.19 20:16:27 +03'00'
(Υπογραφή)	**(Υπογραφή)**	**(Υπογραφή)**

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ

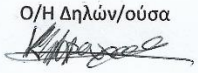
ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος/η Μπράχος Κωνσταντίνος του Ευαγγέλου με αριθμό μητρώου 41667 φοιτητής/τρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Βιομηχανικής Σχεδίασης και Παραγωγής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο/Η Δηλών/ούσα



ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της πτυχιακής εργασίας μου, κ. Δημήτριο Κάντζο για την καθοδήγηση του και τις πολύτιμες διορθώσεις του, τους φίλους μου και την οικογένειά μου για όλη την υποστήριξη που μου πρόσφεραν όλα τα χρόνια των σπουδών μου.

Περιεχόμενα

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια η τεχνητή νοημοσύνη και συγκεκριμένα η βαθιά μάθηση έχει γίνει αρκετά δημοφιλής, παρουσιάζοντας εντυπωσιακά αποτελέσματα σε διάφορους τομείς, και ειδικότερα στην μηχανική όραση, στην επεξεργασία φυσικής γλώσσας και στην ανίχνευση απάτης.

Σε αυτή την πτυχιακή εργασία θα ασχοληθούμε με τη μηχανική όραση και το πως μπορούμε να χρησιμοποιήσουμε μεθόδους βαθιάς μάθησης για την ταξινόμηση εικόνων.

Σκοπός της πτυχιακής εργασίας είναι ο σχεδιασμός και η υλοποίηση μιας web εφαρμογής για την ταξινόμηση εικόνων με τη χρήση νευρωνικών δικτύων για να δούμε στην πράξη ένα πραγματικό πρόβλημα όπως η αναγνώριση αντικειμένων. Η εφαρμογή θα επιτρέπει στους χρήστες να ανεβάζουν εικόνες και στη συνέχεια θα τους επιστρέφει κάποιες ετικέτες βασισμένες στο περιεχόμενο της εικόνας και τις αντίστοιχες πιθανότητες τους. Το νευρωνικό δίκτυο που θα χρησιμοποιηθεί για την ταξινόμηση εικόνων είναι ένα συνελικτικό νευρωνικό δίκτυο (Convolutional Neural Network - CNN) που εκπαιδεύτηκε στο σύνολο δεδομένων Cifar-10 που περιέχει μερικές δεκάδες χιλιάδες εικόνες και τις αντίστοιχες ετικέτες τους. Η εφαρμογή επίσης παρέχει μια διεπαφή για τους χρήστες για να ανεβάζουν εικόνες καθώς επίσης και να βλέπουν τα αποτελέσματα της ταξινόμησης. Στόχος είναι η ανάδειξη της χρησιμότητας τέτοιων μοντέλων βαθιάς μάθησης στην πράξη, με ένα εργαλείο φιλικό προς τους χρήστες που παρέχει πρόσβαση σε τέτοια μοντέλα.

Λέξεις-Κλειδιά: Τεχνητή Νοημοσύνη, Βαθιά Μάθηση, Συνελικτικό Νευρωνικό Δίκτυο, Μηχανική Όραση, Ταξινόμηση Εικόνων, Αναγνώριση Αντικειμένων

ABSTRACT

In recent years, artificial intelligence and specifically deep learning has become quite popular, showing impressive results in various fields and especially in computer vision, natural language processing and fraud detection.

In this thesis we will focus on computer vision and how deep learning can be used to classify images.

The purpose of this thesis is the design and implementation of a web application for image classification using neural networks in order to tackle a real-world problem like object detection. This application will allow users to upload images and subsequently provide them with some labels based on the content of the image and their respective probabilities. The neural network that will be used to classify images is a convolutional neural network trained on the Cifar-10 dataset that contains a few tens of thousands of images and their respective labels. The application also provides a user interface for users to upload their images as well as view the classification results. The goal is to highlight the utility of such deep learning models in practice, with a user-friendly tool that provides access to such models.

Keywords: Artificial Intelligence, Deep Learning, Convolutional Neural Network (CNN), Computer Vision, Image Classification, Object Detection

1. ΕΙΣΑΓΩΓΗ

Η μηχανική όραση είναι από τους πιο δημοφιλείς κλάδους της τεχνητής νοημοσύνης, που στόχο έχει να προσδώσει την ικανότητα της όρασης σε υπολογιστικά συστήματα όπως ένας ηλεκτρονικός υπολογιστής, ένας ρομποτικός βραχίονας ή ακόμα και ένα αυτόνομο όχημα. Μερικοί από τους λόγους που η μηχανική όραση άρχισε να τραβάει τα βλέμματα της επιστημονικής κοινότητας είναι η τεχνολογική εξέλιξη των υπολογιστικών συστημάτων, όπως οι ταχύτητες και οι επιδόσεις των επεξεργαστών ηλεκτρονικών υπολογιστών και των καρτών γραφικών που συνήθως προτιμώνται για τέτοιου είδους εργασίες αλλά και ο τεράστιος όγκος δεδομένων που πλέον υπάρχει.

Στην παρούσα εργασία θα δούμε μια πρακτική εφαρμογή της μηχανικής όρασης, την ταξινόμηση (ή κατηγοριοποίηση) εικόνων με βάση το περιεχόμενό τους, με χρήση βαθιάς μάθησης και νευρωνικών δικτύων. Στα πρώτα κεφάλαια θα δούμε εισαγωγικά κάποιες βασικές έννοιες της βαθιάς μάθησης, της μηχανικής όρασης και της ταξινόμησης εικόνων καθώς και τις εφαρμογές της σε πραγματικά προβλήματα και όσο προχωράμε στα επόμενα κεφάλαια θα δούμε μια πρακτική εφαρμογή, την υλοποίηση της και τις τεχνολογίες που χρησιμοποιήθηκαν.

1.1 Σκοπός της πτυχιακής εργασίας

Η ταξινόμηση εικόνων ή αλλιώς και κατηγοριοποίηση, είναι ένα πεδίο της μηχανικής όρασης που γίνεται όλο και πιο σημαντική τα τελευταία χρόνια λόγω της αυξανόμενης ανάγκης για αυτοματοποιημένα συστήματα που έχουν την δυνατότητα να επεξεργαστούν και να κατηγοριοποιήσουν δεδομένα εικόνων. Με τα νευρωνικά δίκτυα, και συγκεκριμένα τα συνελικτικά νευρωνικά δίκτυα, να έχουν αναπτυχθεί σε ένα πανίσχυρο εργαλείο για εργασίες ταξινόμησης εικόνων, με ευρύ φάσμα εφαρμογών όπως στον τομέα της υγείας, της γεωργίας και της ασφάλειας, η ενσωμάτωσή τους σε εφαρμογές φιλικές προς τον χρήστη μπορούν να τα κάνουν πιο προσιτά επιτρέποντας στον καθένα να τα χρησιμοποιήσει.

Ο κύριος σκοπός της πτυχιακής εργασίας είναι να δείξει πως ένα νευρωνικό δίκτυο μπορεί να ενσωματωθεί σε μια εφαρμογή ιστού καθιστώντας το προσβάσιμο στους χρήστες για εργασίες ταξινόμησης εικόνων. Επίσης σκοπεύει να δείξει την πρακτική πλευρά δημιουργίας και ανάπτυξης τέτοιων εφαρμογών παρουσιάζοντας τη χρησιμότητά τους σε διάφορος τομείς. Παρουσιάζοντας λοιπόν ένα λειτουργικό παράδειγμα, αυτή η εργασία στοχεύει να χρησιμοποιηθεί και ως εκπαιδευτικός πόρος για ερευνητές, προγραμματιστές και φοιτητές που ενδιαφέρονται να ασχοληθούν με αυτόν τον κλάδο και να πειραματιστούν με τις δικές τους ιδέες και εφαρμογές.

Εκτός από την εκπαιδευτική της αξία, η εργασία αυτή στοχεύει επίσης στη σημασία του πως πολύπλοκα μοντέλα μηχανικής μάθησης, όπως τα νευρωνικά δίκτυα, μπορούν να γίνουν πιο

προσιτά στο ευρύ κοινό δημιουργώντας εφαρμογές φιλικές προς το χρήστη έτσι ώστε άτομα που δεν έχουν τεχνικές γνώσεις να επωφεληθούν από τις δυνατότητες αυτών των μοντέλων, προωθώντας έτσι την καινοτομία την υιοθέτηση λύσεων που βασίζονται στην τεχνητή νοημοσύνη σε διάφορους κλάδους.

Επιπλέον, παρουσιάζοντας την ενσωμάτωση ενός τέτοιου μοντέλου σε μια web εφαρμογή, αυτή η εργασία δείχνει την ευελιξία τέτοιων λύσεων που μπορεί να εμπνεύσουν την ανάπτυξη παρόμοιων εφαρμογών προσαρμοσμένων σε συγκεκριμένες περιπτώσεις ή προβλήματα.

1.2 Στόχοι

Οι κύριοι στόχοι της πτυχιακής εργασίας παρέχουν μια ολοκληρωμένη κατανόηση της διαδικασίας δημιουργίας μιας web εφαρμογής και την ενσωμάτωση ενός μοντέλου μηχανικής μάθησης για εργασία ταξινόμησης εικόνων.

Πιο αναλυτικά:

- Δημιουργία ενός συνελικτικού νευρωνικού δικτύου για ταξινόμηση εικόνων
    - Επιλογή και επεξεργασία ενός dataset για την εκπαίδευση του μοντέλου.
    - Σχεδίαση της αρχιτεκτονικής του νευρωνικού δικτύου για την εργασία της ταξινόμησης εικόνων.
    - Εκπαίδευση του μοντέλου και βελτίωση της απόδοσης του με χρήση βελτιστοποίησης παραμέτρων.
    - Αξιολόγηση της απόδοσης του μοντέλου.
- Δημιουργία μιας φιλικής προς το χρήστη web εφαρμογής που επιτρέπει στον χρήστη να ανεβάζει εικόνες και να βλέπει τα αποτελέσματα της ταξινόμησης εικόνων.
    - Δημιουργία της διεπαφής χρήστη για να μπορεί ο χρήστης να την χρησιμοποιήσει εύκολα.
    - Υλοποιώντας τα επιμέρους κομμάτια της εφαρμογής, από το ανέβασμα εικόνας μέχρι και της παρουσίασης των αποτελεσμάτων και αποθήκευσής τους σε μια βάση δεδομένων.
- Υλοποίηση του κώδικα για την χρησιμοποίηση του μοντέλου μέσα από την web εφαρμογή.
    - Προεπεξεργασία της εικόνας που ανέβασε ο χρήστης, μεταβίβαση της εικόνας στο μοντέλο και επιστροφή των αποτελεσμάτων της ταξινόμησης στον χρήστη.
- Παρουσίαση της υλοποίησης και των αποτελεσμάτων.
    - Παρουσίαση της υλοποίησης της εφαρμογής και του μοντέλου.
    - Παρουσίαση των αποτελεσμάτων και την απόδοση του μοντέλου και πιθανών εφαρμογών.
    - Αναγνώριση περιορισμών και βελτιώσεις της εφαρμογής στο μέλλον.

1.3 Δομή της πτυχιακής εργασίας

Σε αυτή την πτυχιακή εργασία θα δούμε πως μπορούμε να χρησιμοποιήσουμε τα νευρωνικά δίκτυα για εργασίες ταξινόμησης εικόνων όπως επίσης και την υλοποίηση ενός μοντέλου και ενσωμάτωσή του σε μια web εφαρμογή. Συγκεκριμένα θα δούμε τα συνελικτικά νευρωνικά δίκτυα.

Στο κεφάλαιο 2 γίνεται η βιβλιογραφική αναφορά σχετικά με την ταξινόμηση εικόνων και τα νευρωνικά δίκτυα, την αρχιτεκτονική τους και τρόπο λειτουργίας τους αλλά και την εκπαίδευση τους.

Στο κεφάλαιο 3 θα δούμε συγκεκριμένα τα συνελικτικά νευρωνικά δίκτυα (cnn), την αρχιτεκτονική τους, τα επίπεδα, και την εκπαίδευση τους. Επίσης θα δούμε και την υλοποίηση του μοντέλου που θα χρησιμοποιήσουμε και το πως βελτιώσαμε την απόδοσή του.

Στο κεφάλαιο 4 θα αναλύσουμε την υλοποίηση της εφαρμογής και τα επιμέρους κομμάτια της, όπως βάση δεδομένων, καθώς και τις βιβλιοθήκες και εργαλεία που χρησιμοποιήσαμε.

Στο κεφάλαιο 5 θα αναφέρουμε τα συμπεράσματα στα οποία καταλήξαμε από τη δημιουργία της εφαρμογής μας και τις δυσκολίες που είχαμε. Επίσης θα αναφέρουμε και ποιες βελτιώσεις θα μπορούσαν να γίνουν.

1.4 Συνεισφορά

Παρόλο που δεν προσπαθούμε να λύσουμε ένα συγκεκριμένο πρόβλημα, με αυτή την πτυχιακή εργασία προσπαθούμε να συνεισφέρουμε στον «εκδημοκρατισμό» της τεχνητής νοημοσύνης μέσω μιας απλής και φιλικής προς τον χρήστη εφαρμογή.

2. Βιβλιογραφική Επισκόπηση

2.1 Image Classification

Ταξινόμηση εικόνων ή αλλιώς και κατηγοριοποίηση, είναι η εργασία κατά την οποία ένα σύστημα να αντιστοιχίσει ετικέτες σε εικόνες με βάση το περιεχόμενό τους. Ο κύριος στόχος είναι να μπορέσει το σύστημα να αντιληφθεί και να αναγνωρίσει το περιεχόμενο μιας εικόνας και να τις κατηγοριοποιήσει παρόμοια με τον τρόπο που το κάνουν οι άνθρωποι.

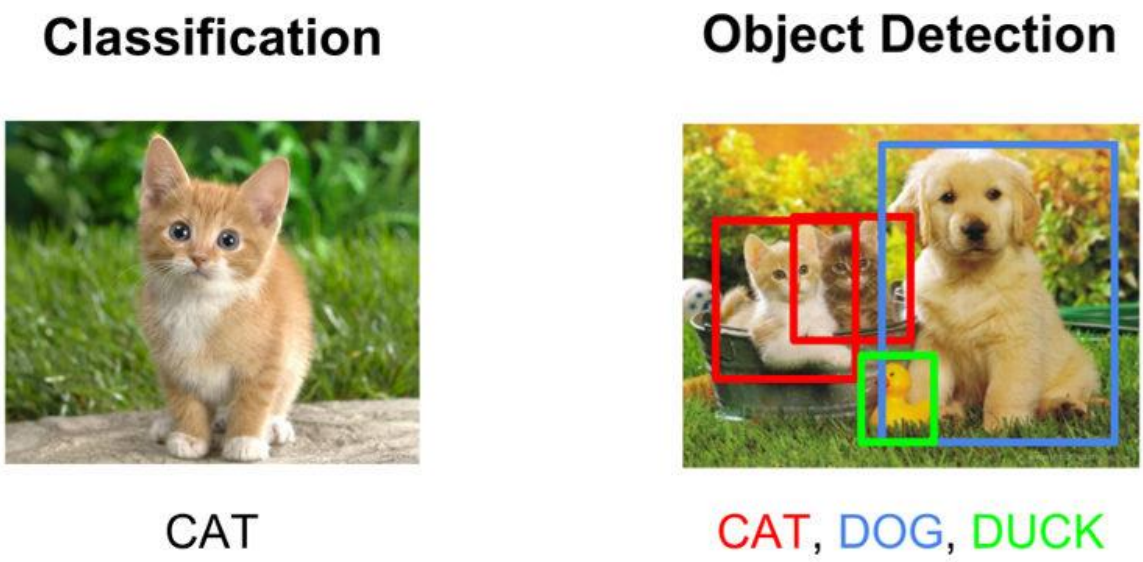
Όπως και στην αναγνώριση αντικειμένων, έτσι κι εδώ υπάρχουν κάποιες προκλήσεις για το σύστημα όπως για παράδειγμα ο φωτισμός, η παραμόρφωση του αντικειμένου, η αλλοίωση των χρωμάτων ή ακόμα και η δυσκολία αναγνώρισης στο περιβάλλον.

Επίσης, η ταξινόμηση εικόνων έχει αρκετές εφαρμογές στην ιατρική, για αναγνώριση καρκινικών κυττάρων, στα αυτοκινούμενα οχήματα για αναγνώριση φαναριών ή λωρίδων στους δρόμους, μέχρι και αναγνώριση προσώπων για λόγους ασφαλείας.

Παρόλο που σαν τεχνική μοιάζει αρκετά με την αναγνώριση αντικειμένων είναι σημαντικό να ξεχωρίσουμε αυτούς τους όρους, παρακάτω θα δούμε τις κύριες διαφορές τους.

2.1.1 Διαφορές ταξινόμησης εικόνων και αναγνώρισης αντικειμένων

- Στόχος
    - Στην ταξινόμηση εικόνων στόχος είναι η αντιστοίχιση μιας ετικέτας σε ολόκληρη τη φωτογραφία και η κατηγοριοποίησή της με βάση το περιεχόμενο που απεικονίζει.
    - Στην αναγνώριση αντικειμένων όμως ο στόχος είναι να αναγνωρίσει όλα τα αντικείμενα μιας συγκεκριμένης κλάσης ή και περισσότερων και να τα βάλει σε πλαίσια μαζί με την ετικέτα τους.
- Χειρισμός πολλαπλών αντικειμένων
    - Τα μοντέλα ταξινόμησης εικόνων δεν παρέχουν πληροφορίες σχετικά με τον αριθμό των αντικειμένων ή την τοποθεσία τους μέσα στην εικόνα. Είναι σχεδιασμένα να κατηγοριοποιούν το κύριο αντικείμενο της εικόνας.
    - Τα μοντέλα αναγνώρισης αντικειμένων όμως είναι σχεδιασμένα να χειρίζονται πολλαπλά αντικείμενα μέσα σε μια εικόνα και παρέχουν πληροφορίες σχετικά με την κλάση τους, την τοποθεσία τους και μερικές φορές και το μέγεθός τους.
- Αρχιτεκτονική
    - Τα μοντέλα ταξινόμησης εικόνων, όπως τα συνελικτικά δίκτυα, συνήθως αποτελούνται από μια σειρά από συνελικτικά επίπεδα και επίπεδα συγκέντρωσης που ακολουθούνται από πλήρως διασυνδεδεμένα επίπεδα που εξάγουν τις πιθανότητες τις κλάσης.
    - Τα μοντέλα αναγνώρισης αντικειμένων από την άλλη, μπορεί να έχουν σαν βάση μοντέλα ταξινόμησης εικόνων, αλλά επεκτείνονται με επιπρόσθετα κομμάτια, όπως region proposal networks πχ στα Fast R-CNN, ή άγκυρες (anchors) πχ στα SSD και YOLO μοντέλα, για να προβλέψουν τα πλαίσια και τις κλάσεις ταυτόχρονα.



Σχήμα 2.1 Image Classification vs Object Detection

Συνοπτικά όπως βλέπουμε στην εικόνα 2.1, η ταξινόμηση εικόνων εστιάζει στον προσδιορισμό του συνολικού περιεχομένου της εικόνας ενώ η αναγνώριση αντικειμένων στοχεύει στον εντοπισμό της κλάσης και της τοποθεσίας αντικειμένων μέσα σε μια εικόνα.

2.1.2 Τεχνικές Ταξινόμησης Εικόνων

Υπάρχουν διάφορες προσεγγίσεις για την ταξινόμηση εικόνων από τεχνικές μηχανικής μάθησης μέχρι και πιο καινούριες τεχνικές βαθιάς μάθησης που βασίζονται στα νευρωνικά δίκτυα.

1. Μέθοδοι που χειροκίνητα εξάγουμε χαρακτηριστικά από εικόνες, όπως άκρες, γωνίες, υφές, και τις χρησιμοποιούμε ως εισόδους σε παραδοσιακούς αλγόριθμους μηχανικής μάθησης όπως Support Vector Machines, Decision Trees, k-Nearest Neighbors.
2. Μέθοδοι Shallow Learning που χρησιμοποιούν απλά νευρωνικά δίκτυα, με μικρό αριθμό κρυφών επιπέδων, για την εκμάθηση των χαρακτηριστικών των εικόνων και την ταξινόμηση, αλλά η απόδοσή τους δεν είναι αρκετά καλή.
3. Μέθοδοι βαθιάς μάθησης που χρησιμοποιούν πιο πολύπλοκα νευρωνικά δίκτυα, και ιδιαίτερα τα συνελικτικά νευρωνικά δίκτυα που έχουν δείξει πολύ καλύτερη απόδοση από άλλες μεθόδους στην ταξινόμηση εικόνων.

Σε αυτή την πτυχιακή εργασία θα χρησιμοποιήσουμε τη μέθοδο βαθιάς μάθησης με συνελικτικά νευρωνικά δίκτυα για την ταξινόμηση εικόνων.

2.1.3 Δημοφιλή datasets για την ταξινόμηση εικόνων

Υπάρχουν αρκετά έτοιμα datasets που μπορεί κάποιος να χρησιμοποιήσει για την ταξινόμηση εικόνων, κάποια από τα πιο κοινά είναι τα εξής:

1. MNIST: Αποτελείται από 70000 ασπρόμαυρες εικόνες που απεικονίζουν ψηφία γραμμένα στο χέρι και χρησιμοποιούνται κυρίως σε μοντέλα για εργασίες αναγνώρισης χειρόγραφων ψηφίων.
2. CIFAR-10 / CIFAR-100: Αποτελείται από 60000 μικρές έγχρωμες εικόνες χωρισμένες σε 10 και 100 κλάσεις αντίστοιχα. Χρησιμοποιούνται κυρίως σε γενικά μοντέλα για αναγνώριση αντικειμένων και ταξινόμησης εικόνων.

Στη παρούσα πτυχιακή εργασία θα χρησιμοποιήσουμε το CIFAR-10 και θα το δούμε πιο αναλυτικά σε επόμενο κεφάλαιο.

2.2 Βαθιά Μάθηση και Νευρωνικά Δίκτυα

2.2.1 Βαθιά Μάθηση

Η βαθιά μάθηση είναι μια υποκατηγορία της μηχανικής μάθησης και της τεχνητής νοημοσύνης που βασίζεται στα νευρωνικά δίκτυα για να διδάξει έναν υπολογιστή να μάθει πολύπλοκα μοτίβα και αναπαραστάσεις από τεράστιο όγκο δεδομένων. Ο όρος «βαθιά» στη βαθιά μάθηση αναφέρεται στον αριθμό επιπέδων των νευρωνικών δικτύων που χρησιμοποιούνται. Οι αρχιτεκτονικές αυτές επιτρέπουν να εξάγουν αυτόματα χαρακτηριστικά από ακατέργαστα δεδομένα όπως εικόνες, κείμενο και ήχο.

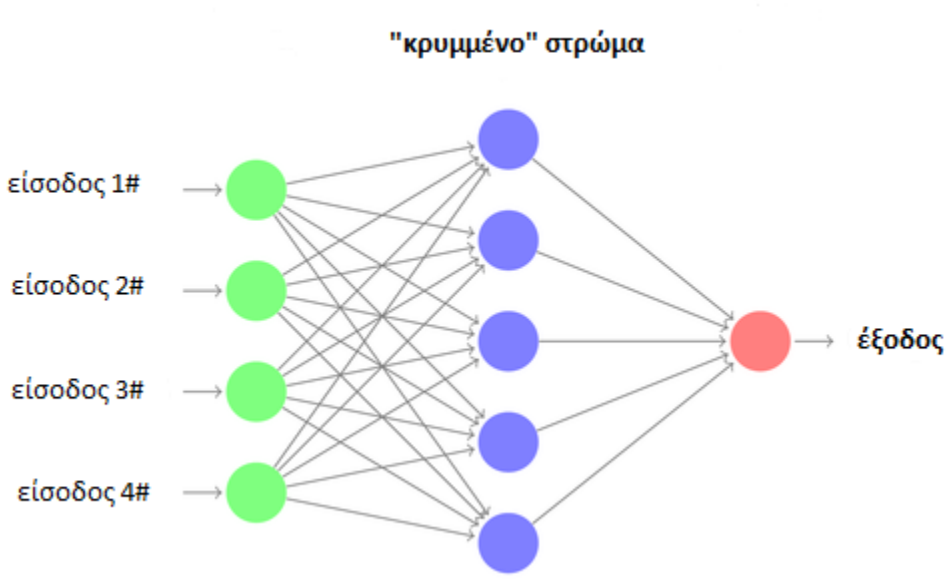
Με την τεράστια διαθεσιμότητα τόσο σε δεδομένα αλλά και σε επεξεργαστική ισχύ, η βαθιά μάθηση έχει φτάσει σε ένα σημείο που ερευνητές έχουν καταφέρει τεράστιες επιτυχίες σε εργασίες όπως η αναγνώριση αντικειμένων σε εικόνες και η αναγνώριση ομιλίας που μέχρι πριν ήταν αδιανόητο να συμβούν. Μοντέλα βαθιάς μάθησης μπορούν να επιτύχουν τεράστια ακρίβεια που μερικές φορές ξεπερνούν και την επίδοση των ανθρώπων.

Τα νευρωνικά δίκτυα εμπνέονται από τη δομή και τη λειτουργία του ανθρώπινου εγκεφάλου και αποτελούνται από επίπεδα διασυνδεδεμένων κόμβων, με κάθε κόμβο να επεξεργάζεται πληροφορίες και να τις μεταβιβάζει στο επόμενο επίπεδο.

2.2.2 Νευρωνικό Δίκτυο

Το νευρωνικό δίκτυο, ή αλλιώς τεχνητό νευρωνικό δίκτυο, είναι μια κατηγορία αλγορίθμων βαθιάς μηχανικής μάθησης, που αποτελούν μια προσπάθεια μοντελοποίησης των νευρώνων του εγκεφάλου και να μιμηθούν τον τρόπο λειτουργίας τους. Είναι σχεδιασμένα να αναγνωρίζουν μοτίβα και αναπαραστάσεις από δεδομένα.

Το νευρωνικό δίκτυο αποτελείται από διασυνδεδεμένα στρώματα τεχνητών νευρώνων που επεξεργάζονται και μεταδίδουν πληροφορίες. Η βασική δομή ενός νευρωνικού δικτύου περιλαμβάνει ένα ή περισσότερα επίπεδα εισόδου, ένα ή περισσότερα κρυφά επίπεδα και το επίπεδο εξόδου.

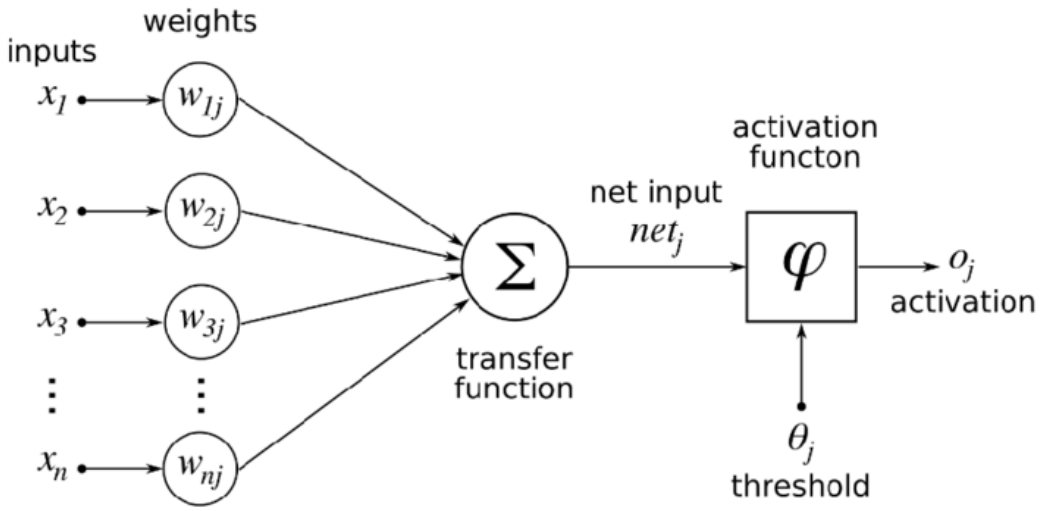


Σχήμα 2.2 Τεχνητό Νευρωνικό Δίκτυο

Παρακάτω θα δούμε κάποια βασικά στοιχεία και έννοιες των τεχνητών νευρωνικών δικτύων.

2.2.3 Τεχνητός Νευρώνας

Ο τεχνητός νευρώνας είναι το θεμελιώδες δομικό στοιχείο των τεχνητών νευρωνικών δικτύων. Είναι ένα υπολογιστικό μοντέλο εμπνευσμένο από τον βιολογικό νευρώνα του εγκεφάλου. Δέχεται σαν είσοδο εξωτερικά δεδομένα ή από άλλους τεχνητούς νευρώνες, επεξεργάζονται τα δεδομένα και παράγουν μια έξοδο που μεταβιβάζονται σε άλλους τεχνητούς νευρώνες του νευρωνικού δικτύου.



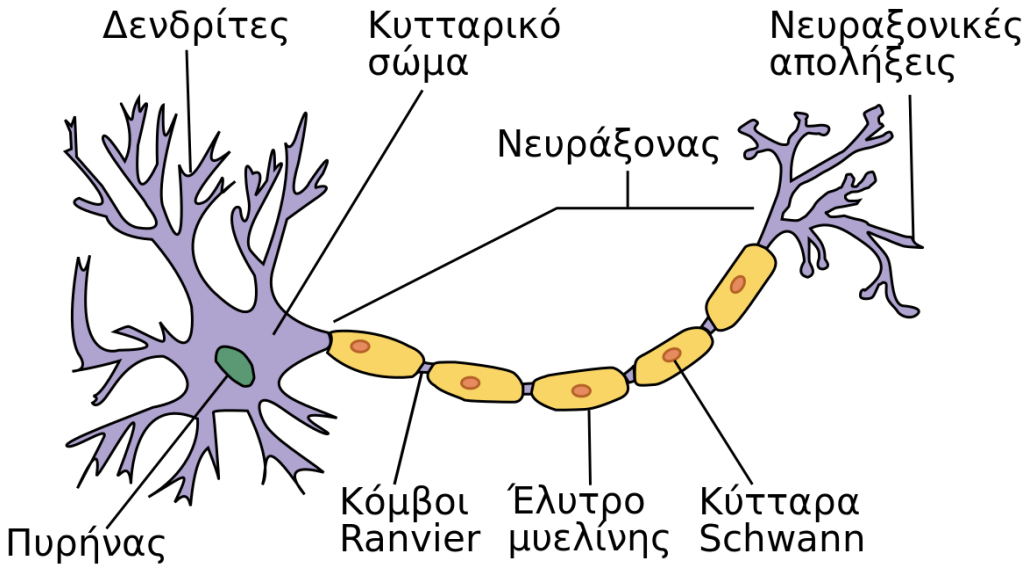
Σχήμα 2.3 Δομή Τεχνητού Νευρώνα

Η δομή του τεχνητού νευρώνα έχει τα εξής στοιχεία

1. Είσοδος (Inputs): Λαμβάνει πολλαπλές εισόδους είτε από εξωτερικά ακατέργαστα δεδομένα, είτε από την έξοδο άλλων νευρώνων.
2. Βάρη (Weights): Τα βάρη είναι παράμετροι που μαθαίνει το δίκτυο κατά τη διάρκεια της εκπαίδευσης και καθορίζουν την επιρροή της εισόδου στην έξοδο του νευρώνα.
3. Άθροισμα των βαρών: Ο νευρώνας πολλαπλασιάζει κάθε είσοδο με το αντίστοιχο βάρος και υπολογίζει το άθροισμα των αποτελεσμάτων. Μερικές φορές υπάρχει και ένας όρος bias.
4. Bias: Bias είναι μια παράμετρος που επιτρέπει τη μετατόπιση της συνάρτησης ενεργοποίησης κατά μήκος του άξονα εισόδου. Κάποιες φορές είναι σταθερή με τιμή 1 πολλαπλασιασμένη με το δικό της βάρος. Επίσης επιτρέπει στον νευρώνα να παράγει μη μηδενικές εξόδους ακόμα και όταν όλες οι είσοδοι είναι 0.
5. Συνάρτηση Ενεργοποίησης (Activation Function): Η συνάρτηση ενεργοποίησης εισάγει τη μη γραμμικότητα στον τεχνητό νευρώνα και του επιτρέπει να μαθαίνει περίπλοκες σχέσεις μεταξύ εισόδων και εξόδων. Η συνάρτηση ενεργοποίησης εφαρμόζεται στο άθροισμα των βαρών για να φτιάξει την έξοδο του νευρώνα. Κάποιες κοινές συναρτήσεις ενεργοποίησης είναι οι ReLU (Rectified Linear Unit), η σιγμοειδή και η υπερβολική εφαπτομένη που θα τις δούμε πιο αναλυτικά παρακάτω.
6. Έξοδος (Output): Η έξοδος του τεχνητού νευρώνα είναι το αποτέλεσμα της συνάρτησης ενεργοποίησης και μεταδίδεται σε άλλους νευρώνες ή χρησιμεύει ως τελική έξοδος του δικτύου, ανάλογα με το επίπεδο που βρίσκεται ο νευρώνας.

Οι τεχνητοί νευρώνες λοιπόν είναι το κύριο συστατικό των νευρωνικών δικτύων και χάρη σε αυτούς το νευρωνικό δίκτυο μπορεί να επεξεργαστεί πληροφορίες και να τις μεταβιβάσει στο επόμενο επίπεδο.

Οι τεχνητοί και οι βιολογικοί νευρώνες έχουν κάποιες ομοιότητες στη βασική δομή και λειτουργία τους αλλά υπάρχουν και αρκετές βασικές διαφορές. Υπολογίζεται ότι ο άνθρωπος έχει 86 δισεκατομμύρια νευρώνες οι οποίοι είναι ένα δομικό μέρος του νευρικού μας συστήματος. Ένας απλός νευρώνας παίρνει τα ηλεκτρικά σήματα εισόδου από τους δενδρίτες, τα επεξεργάζεται και στέλνει το σήμα εξόδου μέσω του νευράξονα σε άλλους νευρώνες. Αντίστοιχα και στον τεχνητό νευρώνα, οι είσοδοι πολλαπλασιάζονται με τα συναπτικά βάρη, προσθέτονται μεταξύ τους και περνάνε μέσα από μια συνάρτηση ενεργοποίησης όπου παράγει και το σήμα εξόδου. Κάποιες από τις διαφορές τους είναι οι εξής:



Σχήμα 2.4 Βιολογικός Νευρώνας

Οι βιολογικοί νευρώνες είναι απίστευτα πολύπλοκα κύτταρα με πολύπλοκη δομή. Όπως βλέπουμε και στην εικόνα 2.4, αποτελούνται από δενδρίτες, ένα κυτταρικό σώμα, έναν άξονα και περιλαμβάνουν νευροδιαβιβαστές και υποδοχείς. Οι τεχνητοί νευρώνες από την άλλη έχουν μια πιο απλοποιημένη δομή όπως είδαμε παραπάνω.

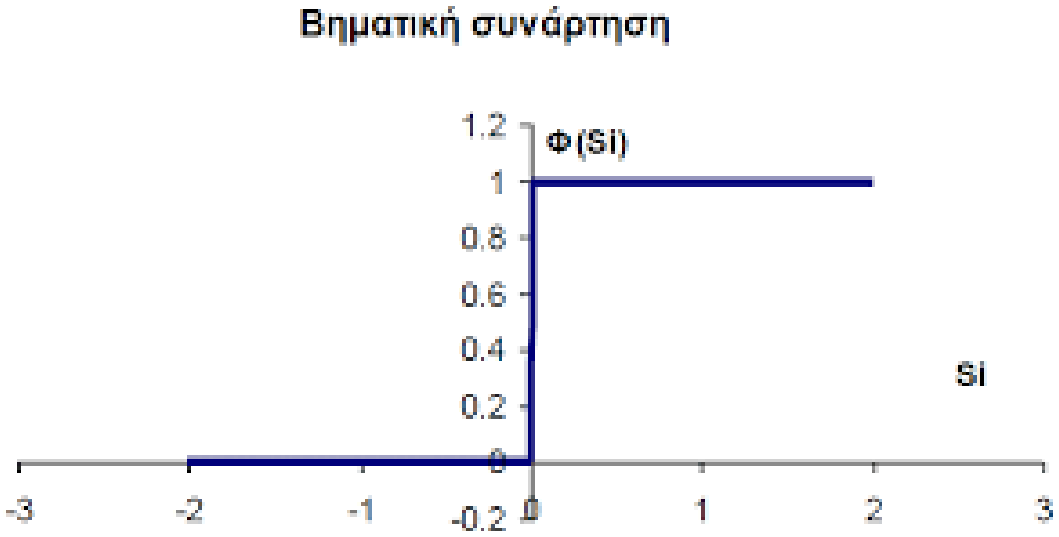
Επίσης στους βιολογικούς νευρώνες, οι πληροφορίες μεταδίδονται μέσω ηλεκτροχημικών σημάτων ενώ οι τεχνητοί νευρώνες μέσω μαθηματικών πράξεων.

Μια άλλη πολύ σημαντική διαφορά είναι η ταχύτητα, οι τεχνητοί νευρώνες μπορούν να επεξεργαστούν δεδομένα πολύ πιο γρήγορα από έναν βιολογικό νευρώνα λόγω της διαφοράς στο «hardware» που στην περίπτωση των τεχνητών νευρώνων είναι ηλεκτρονικά κυκλώματα.

Προφανώς υπάρχουν και άλλες μεγάλες διαφορές που δεν θα αναφέρουμε στην παρούσα πτυχιακή μιας και δεν είναι αυτός ο σκοπός της.

2.2.4 Συναρτήσεις Ενεργοποίησης Νευρωνικού Δικτύου

- Βηματική συνάρτηση ενεργοποίησης



Σχήμα 2.3 Γραφική Βηματικής Συνάρτησης

Η βηματική συνάρτηση είναι η πιο απλή συνάρτηση ενεργοποίησης που χαρακτηρίζεται από 2 τιμές α και β και χρησιμοποιείται κυρίως για τον διαχωρισμό της εισόδου σε 2 κατηγορίες. Είναι μια δυαδική συνάρτηση ενεργοποίησης που παράγει έξοδο 1 αν η τιμή εισόδου είναι μεγαλύτερη ή ίση με και 0 αν είναι μικρότερη ή ίση με 0. Μιας και η μάθηση στα τεχνητά νευρωνικά δίκτυα είναι η μεταβολή των βαρών που σχετίζεται με την παράγωγο και η βηματική συνάρτηση έχει ως μειονέκτημα ότι η παράγωγός της είναι μηδέν, δεν θεωρείται χρήσιμη ως συνάρτηση ενεργοποίησης.

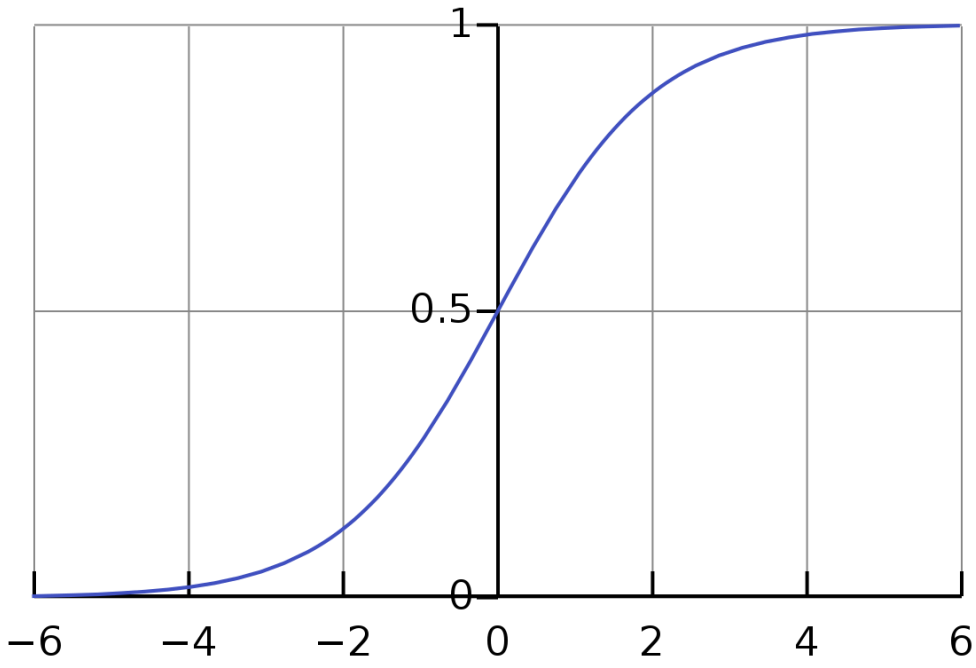
Μαθηματικά ορίζεται ως:

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 1 \end{cases} \qquad (2.1)$$

Λόγω της ασυνέχειας και της μη διαφοροποίησής της, δεν χρησιμοποιείται τόσο στις σύγχρονες αρχιτεκτονικές νευρωνικών δικτύων προτιμώντας πιο προηγμένες συναρτήσεις. Παρόλα αυτά η βηματική συνάρτηση μπορεί να χρησιμοποιηθεί για

εκπαιδευτικούς σκοπούς για την κατανόηση των νευρωνικών δικτύων όπως και σε ρηχά νευρωνικά δίκτυα για εργασίες δυαδικής ταξινόμησης.

- Σιγμοειδής συνάρτηση ενεργοποίηση



Σχήμα 2.4 Γραφική Σιγμοειδής Συνάρτησης

Η σιγμοειδής συνάρτηση, ή αλλιώς λογιστική συνάρτηση, είναι μια από τις πιο δημοφιλείς συναρτήσεις και αποτελεί συνεχή και παραγωγίσιμη προσέγγιση της βηματικής συνάρτησης. Έχει σχήμα S, εικόνα 2.5, και κανονικοποιεί την τιμή εισόδου στο διάστημα [0, 1] αλλά έχει ως μειονέκτημα ότι για πολύ μεγάλες ή μικρές τιμές της κλίσης της γίνεται βηματική. Είναι μη γραμμική συνάρτηση και συμβάλει σημαντικά στην εισαγωγή μη γραμμικότητας στα νευρωνικά δίκτυα, που επιτρέπει στο νευρωνικό δίκτυο να μοντελοποιεί σύνθετες σχέσεις μεταξύ χαρακτηριστικών εισόδου και των εξόδων, που δεν μπορούν τα γραμμικά μοντέλα.

Μαθηματικά ορίζεται ως:

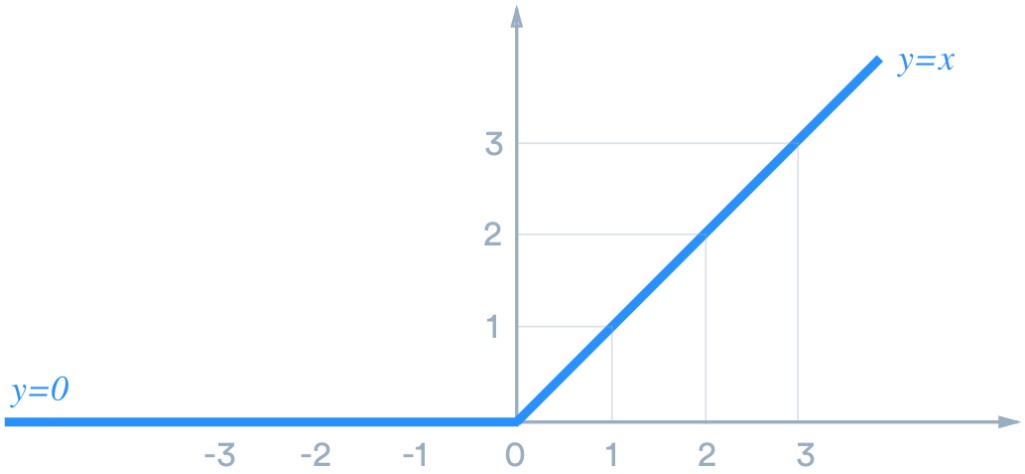
$$f(x) = \frac{1}{1+e^{-x}} \qquad (2.2)$$

Επίσης είναι διαφοροποιήσιμη όπου είναι απαραίτητη για τεχνικές βελτιστοποίησης που βασίζονται σε κλίση, όπως η οπισθοδιάδοση. Η παράγωγός της ως προς την είσοδό της μπορεί να εκφραστεί μαθηματικά ως:

$$f'(x) = f(x) * \big(1 - f(x)\big) \qquad (2.3)$$

Παρά τη δημοτικότητά της, έχει και τα μειονεκτήματά της όπως το πρόβλημα που εξαφανίζεται η κλίση της και η έλλειψη μηδενικής κεντρικότητας και γι'αυτό άλλες συναρτήσεις ενεργοποίησης, όπως η ReLU που θα δούμε στη συνέχεια, κερδίζουν έδαφος σαν συνάρτηση ενεργοποίησης ειδικά στα κρυφά επίπεδα των νευρωνικών δικτύων.

- Συνάρτηση ReLU (Rectified Linear Unit)



Σχήμα 2.5 Γραφική Συνάρτησης Ράμπας (ReLU)

Η συνάρτηση ReLU (Rectified Linear Unit ReLU) ή αλλιώς συνάρτηση ράμπας, είναι ίσως η πιο δημοφιλής συνάρτηση ενεργοποίησης ειδικά σε μοντέλα βαθιάς μάθησης εισάγοντας τη μη γραμμικότητα στο νευρωνικό δίκτυο ορίζοντας το όριο εισόδου στο 0, έτσι ώστε οι αρνητικές τιμές να αντιστοιχίζονται στο 0 και οι θετικές τιμές να παραμείνουν αμετάβλητες, επιτρέποντάς του να μαθαίνει πολύπλοκες σχέσεις μεταξύ εισόδων και εξόδων. Επίσης είναι και γραμμική ανά τμήματα και διαφοροποιήσιμη παντού εκτός από x = 0 όπου είναι ασυνεχείς. Η παράγωγός της είναι 1 για x > 0 και 0 για x < 0. Ένα άλλο πλεονέκτημα της ReLU είναι αρκετά αποδοτική λόγω της απλότητάς της, έτσι κάνει πιο

αποδοτική και γρήγορη την εκπαίδευση του μοντέλου. Πέρα από τα θετικά της έχει και κάποια μειονεκτήματα. Το πιο σημαντικό μειονέκτημα της είναι ότι μπορεί κατά τη διάρκεια της εκπαίδευσης να αδρανοποιήσει τον νευρώνα αν η έξοδός της είναι σταθερά 0, αλλά υπάρχουν παραλλαγές της οι οποίες παρέχουν κάποιες λύσεις σε αυτό το πρόβλημα.

Μαθηματικά ορίζεται ως:

$$f(x) = \max(0, x) \tag{2.4}$$

- Η συνάρτηση Softmax

Μία ακόμα δημοφιλής συνάρτηση είναι η συνάρτηση Softmax, ή αλλιώς κανονικοποιημένη εκθετική συνάρτηση, που χρησιμοποιείται στο επίπεδο εξόδου των νευρωνικών δικτύων για προβλήματα ταξινόμησης πολλαπλών κλάσεων. Παίρνει σαν είσοδο ένα διάνυσμα τιμών (logits) και σκοπός της είναι να το μετατρέπει σε κανονικοποιημένες πιθανότητες στις πιθανές κλάσεις εξασφαλίζοντας ότι το άθροισμα των πιθανοτήτων για όλες τις κλάσεις είναι ίσο με 1. Η έξοδος αυτής της συνάρτησης είναι ένα ν-διάστατο διάνυσμα, όπου ν είναι ο αριθμός των κλάσεων όπου το νευρωνικό δίκτυο έχει να διαλέξει. Γι' αυτό το λόγο είναι κατάλληλη σε εργασίες ταξινόμησης και τη συναντάμε πολύ στα συνελικτικά δίκτυα.
Επίσης είναι δύσκολο να αναπαρασταθεί άμεσα γραφικά. Η γραφική της παράσταση θα είναι μια επιφάνεια N-διαστάσεων σε χώρο N-διαστάσεων, όπου N ο αριθμός κλάσεων. Η οπτικοποίηση αυτής της επιφάνειας είναι δύσκολη για N > 3, γιατί είναι δύσκολο να αντιληφθούμε χώρους περισσότερων διαστάσεων.

Μαθηματικά ορίζεται ως:

$$\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.5}$$

Όπου $\vec{x}$ είναι το διάνυσμα εισόδου, e^{x_i} είναι τυπική εκθετική συνάρτηση για το διάνυσμα εισόδου, K ο αριθμός των κλάσεων και e^{x_j} η τυπική εκθετική συνάρτηση για το διάνυσμα εξόδου.

Επίσης η συνάρτηση Softmax είναι διαφοροποιήσιμη, που είναι απαραίτητο για τεχνικές βελτιστοποίησης που βασίζονται σε κλίση, όπως η οπισθοδιάδοση. Η παράγωγός της σε σχέση με την είσοδό της δίνεται από την ίδια τη συνάρτηση πολλαπλασιασμένη επί
$1 - \sigma(\vec{x})_i$ και με $-\sigma(x)_i * \sigma(x)_j$, όπου $i \neq j$.
Η διαφοροποίησή της επιτρέπει στον υπολογισμό των κλίσεων και την ενημέρωση των βαρών κατά τη διάρκεια της εκπαίδευσης.

Ένα άλλο πλεονέκτημα είναι ότι η έξοδός της μπορεί να ερμηνευτεί άμεσα ως πιθανότητες με τις οποίες να μπορέσουμε να κατανοήσουμε το confidence του μοντέλου στις προβλέψεις του.
Παρά τα πλεονεκτήματα αυτής της συνάρτησης, υπάρχουν και κάποια μειονεκτήματα.
Η εκθετική συνάρτηση που έχει η Softmax μπορεί να οδηγήσει σε μεγάλες τιμές και να προκαλέσουν προβλήματα υπερχείλισης. Μια λύση σε αυτό είναι η αφαίρεση της μέγιστης τιμής από όλες τις τιμές εισόδου πριν τον υπολογισμό της Softmax για να σταθεροποιήσει τον υπολογισμό χωρίς να αλλάξει την κατανομή.
Ένα άλλο μειονέκτημα είναι ότι δεν μπορεί να χρησιμοποιηθεί σε εργασίες μη ταξινόμησης μιας και έχει σχεδιαστεί ειδικά για εργασίες ταξινόμησης πολλών κλάσεων και δεν είναι κατάλληλη για προβλήματα δυαδικής ταξινόμησης ή και regression.

2.3 Εκπαίδευση Νευρωνικών Δικτύων

Η πιο βασική ιδιότητα των νευρωνικών δικτύων είναι η εκπαίδευση, δηλαδή η ικανότητα του να «μαθαίνει». Η εκπαίδευση επιτυγχάνεται μέσω της προσαρμογής των βαρών, που ανανεώνονται σε κάθε επανάληψη, προσπαθώντας να ελαχιστοποιήσει την συνάρτηση σφάλματος, η οποία συνάρτηση είναι η απόκλιση της επιθυμητής εξόδου με την έξοδο. Αυτό επιτυγχάνεται χρησιμοποιώντας συνδυασμό μπροστινής διάδοσης, υπολογισμό της συνάρτησης σφάλματος, οπισθοδιάδοσης και βελτιστοποίησης. Γενικά είναι μια επαναληπτική διαδικασία με πολλές εποχές (επαναλήψεις).

Γενική τα βήματα της διαδικασίας της εκπαίδευσης:

1. Προεπεξεργασία δεδομένων και διαχωρισμός σε δεδομένα εκπαίδευσης, επικύρωσης και δοκιμής.
2. Για κάθε εποχή (επανάληψη):
    a. Μπροστινή διάδοση: Κατά τη διάρκεια της μπροστινής διάδοσης η είσοδος περνάει μέσα από το νευρωνικό δίκτυο υπολογίζοντας την έξοδο σε κάθε επίπεδο χρησιμοποιώντας τα βάρη, τη σταθερά bias αν υπάρχει και τις συναρτήσεις ενεργοποίησης και την περνάει στο επόμενο επίπεδο.
    b. Υπολογισμός σφάλματος: Ο υπολογισμός του σφάλματος γίνεται συγκρίνοντας τις προβλέψεις του δικτύου με τις πραγματικές τιμές μέσω μιας συνάρτησης σφάλματος, όπως η mean squared error για regression προβλήματα ή cross-entropy loss για εργασίες ταξινόμησης. Αυτή η συνάρτηση μετράει την απόκλιση της τιμής εξόδου με την πραγματική τιμή και παρέχει μια τιμή σφάλματος που κατά την εκπαίδευση ο στόχος είναι να ελαχιστοποιηθεί.
    c. Οπισθοδιάδοση (backpropagation): Ο αλγόριθμος οπισθοδιάδοσης χρησιμοποιείται για τον υπολογισμό της κλίσης της συνάρτησης σφάλματος σε σχέση με τα βάρη στο νευρωνικό δίκτυο. Ο αλγόριθμος ξεκινάει από το επίπεδο

εξόδου υπολογίζοντας τις διαβαθμίσεις (gradients) της συνάρτησης σφάλματος σε σχέση με τις παραμέτρους του επιπέδου και συνεχίζει προς τα πίσω χρησιμοποιώντας τον κανόνα της αλυσίδας για τον υπολογισμό των διαβαθμίσεων για τις παραμέτρους κάθε επιπέδου.

d. Ενημέρωση των βαρών: Οι διαβαθμίσεις που υπολογίστηκαν χρησιμοποιούνται στην ενημέρωση των βαρών και του bias χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης, όπως ο SGD (Stochastic Gradient Descent), ο RMSprop ή ο Adam. Τα βάρη ενημερώνονται προς την κατεύθυνση της αρνητικής κλίσης οπου οδηγεί στη μείωση του σφάλματος. Στον αλγόριθμο βελτιστοποίησης έχουμε και μια νέα υπερπαράμετρο, τον ρυθμός εκμάθησης, που ελέγχει το μέγεθος του βήματος των ενημερώσεων.
e. Επανάληψη: Η μπροστινή διάδοση, ο υπολογισμός σφάλματος, η οπισθοδιάδοση και η ενημέρωση των βαρών επαναλαμβάνονται για πολλές εποχές, όπου μια εποχή είναι το πέρασμα όλου του training set μέσα από το νευρωνικό δίκτυο.
f. Επικύρωση (validation): Κατά τη διάρκεια της εκπαίδευσης αξιολογείται η απόδοση του μοντέλου στο validation set. Το validation set δεν χρησιμοποιείται για την ενημέρωση των βαρών αλλά χρησιμοποιείται για την εκτίμηση του μοντέλου σε δεδομένα που δεν έχει «δει». Η επικύρωση του μοντέλου βοηθάει στη παρακολούθηση υπερπροσαρμογής (overfitting) και στην απόφαση για να σταματήσει νωρίτερα η εκπαίδευση (early stopping) ή την προσαρμογή του ρυθμού εκμάθησης.
g. Regularization: Για να αποφύγουμε την υπερπροσαρμογή του νευρωνικού δικτύου χρησιμοποιούνται τεχνικές regularization κατά τη διάρκεια της εκπαίδευσης. Αυτές οι τεχνικές, όπως οι L1 ή L2 regularization, dropout και batch normalization βοηθούν στον περιορισμό της χωρητικότητας του μοντέλου και το βοηθούν να μάθει πιο γενικά και ισχυρά.
h. Fine tuning: Η απόδοση του μοντέλου μπορεί να βελτιωθεί περισσότερο ρυθμίζοντας τις παραμέτρους, τον ρυθμό εκμάθησης και προσαρμόζοντας την αρχιτεκτονική του μοντέλου.

3. Τέλος έχουμε την αξιολόγηση της απόδοσης του μοντέλου στο testing set. Χρησιμοποιώντας κατάλληλες μετρήσεις, για παράδειγμα την ακρίβεια (accuracy) ή ανάκληση (recall) για εργασίες ταξινόμησης και μέσο απόλυτο σφάλμα (mean absolute error) ή μέσο τετραγωνικό σφάλμα (mean squared error) για regression εργασίες.

2.3.1 Αλγόριθμος Οπισθοδιάδοσης Σφάλματος (Backpropagation)

Ο αλγόριθμος backpropagation είναι ένας supervised αλγόριθμος μάθησης για την εκπαίδευση νευρωνικών δικτύων. Όταν συνδυάζεται με μια τεχνική βελτιστοποίησης που βασίζεται σε κλίση, προσαρμόζει τις παραμέτρους του μοντέλου για να ελαχιστοποιήσει τη συνάρτηση σφάλματος και

να βελτιώσει την απόδοση του μοντέλου. Το όνομά του προέρχεται από τον τρόπο που υπολογίζει τις κλίσεις και διαδίδονται προς τα πίσω μέσω των επίπεδων του δικτύου.

Αναλυτική εξήγηση του αλγόριθμου backpropagation:

- Μπροστινό πέρασμα: Τα δεδομένα εισόδου περνάνε μέσα από τα επίπεδα του νευρωνικού δικτύου. Σε κάθε νευρώνα υπολογίζεται το άθροισμα των εισόδων και εφαρμόζεται μια συνάρτηση ενεργοποίησης και δημιουργεί την έξοδο του δικτύου η οποία συγκρίνεται με τις πραγματικές τιμές χρησιμοποιώντας μια συνάρτηση σφάλματος.
- Υπολογισμός σφάλματος: Η συνάρτηση σφάλματος υπολογίζει τη διαφορά των προβλέψεων του δικτύου με τις πραγματικές τιμές. Στόχος είναι με τη διαδικασία βελτιστοποίησης να μειωθεί στο ελάχιστο αυτή η διαφορά προσαρμόζοντας τις παραμέτρους του δικτύου.
- Οπισθοδιάδοση: Στη συνέχεια ο αλγόριθμος backpropagation υπολογίζει τη κλίση της συνάρτησης (μερική παράγωγος) σφάλματος σε σχέση με κάθε παράμετρο (βάρη και bias) στο δίκτυο εφαρμόζοντας τον κανόνα της αλυσίδας της διαφοροποίησης με αναδρομή ξεκινώντας από το επίπεδο εξόδου πηγαίνοντας προς τα πίσω μέσω του δικτύου. Γι’ αυτό και ονομάστηκε backpropagation.
- Υπολογισμός κλίσης:
    - Υπολογισμός της κλίσης της συνάρτησης σφάλματος σε σχέση με την έξοδο του επίπεδου εξόδου. Αυτή η κλίση δείχνει το ρυθμό μεταβολής του σφάλματος που αφορά την έξοδο κάθε νευρώνα στο επίπεδο εξόδου.
    - Υπολογισμός της κλίσης της συνάρτησης σφάλματος σε σχέση με το άθροισμα των βαρών σε κάθε νευρώνα στο επίπεδο εξόδου πολλαπλασιάζοντας τη κλίση του προηγούμενου βήματος με την κλίση της συνάρτησης ενεργοποίησης που εφαρμόζεται στο άθροισμα των βαρών του νευρώνα.
    - Υπολογισμός της κλίσης της συνάρτησης σφάλματος σε σχέση με τα βάρη και τα bias στο επίπεδο εξόδου πολλαπλασιάζοντας την κλίση που υπολογίζεται στο $2^{ο}$ βήμα, με τις αντίστοιχες τιμές εισόδου.
    - Διάδοση των κλίσεων στα προηγούμενα επίπεδα υπολογίζοντας τη κλίση της συνάρτησης σφάλματος σε σχέση με την έξοδο και το άθροισμα των βαρών κάθε νευρώνα στα προηγούμενα επίπεδα αναδρομικά προς τα πίσω επίπεδο προς επίπεδο, εφαρμόζοντας τον κανόνα της αλυσίδας της διαφοροποίησης και λαμβάνοντας υπόψη τα βάρη που συνδέουν τους νευρώνες.
    - Τέλος, γίνεται ο υπολογισμός της κλίσης της συνάρτησης σφάλματος σε σχέση με τα βάρη και τα bias σε κάθε επίπεδο καθώς η διαδικασία κινείται προς τα πίσω.
- Ενημέρωση παραμέτρων: Οι υπολογισμένες κλίσεις χρησιμοποιούνται για την ενημέρωση των παραμέτρων του μοντέλου με κάποιον αλγόριθμο βελτιστοποίησης. Οι ενημερώσεις γίνονται προς την κατεύθυνση που ελαχιστοποιεί την συνάρτηση σφάλματος με βάση το ρυθμό μάθησης που καθορίζει το μέγεθος του βήματος των ενημερώσεων.

- Επανάληψη: Τα περάσματα προς τα εμπρός και προς τα πίσω επαναλαμβάνονται για πολλές εποχές ή μέχρι να ικανοποιηθεί ένα κριτήριο διακοπής.

Backpropagation formula για τον υπολογισμό των κλίσεων της συνάρτησης σφάλματος σε σχέση με τα βάρη και bias:

$$\delta_i = f'(\alpha_i) \sum_k w_{ki} \delta_k \quad (2.6)$$

Όπου,

δ_i, είναι ο όρος σφάλματος για τον ι-οστό νευρώνα σε ένα επίπεδο του δικτύου,

$f'(\alpha_i)$, είναι η παράγωγος της συνάρτησης ενεργοποίησης του ι-οστού νευρώνα που αξιολογείται στην είσοδο a_i. Χρησιμοποιείται για τον υπολογισμό της ευαισθησίας της εξόδου του νευρώνα στις αλλαγές της εισόδου,

$\sum_k w_{ki} \delta_k$, είναι το άθροισμα των γινομένων των βαρών που συνδέουν τον ι-οστό νευρώνα στο επόμενο επίπεδο (k) και τους αντίστοιχους όρους σφάλματος (δ_k).

2.3.2 Συνάρτηση Σφάλματος (Loss Function)

Η συνάρτηση σφάλματος, ή αλλιώς συνάρτηση κόστους, χρησιμοποιούνται για να ποσοτικοποιήσουν τη διαφορά μεταξύ της πραγματικής τιμής και της πρόβλεψης του δικτύου. Παίζουν πολύ μεγάλο ρόλο στην εκπαίδευση του νευρωνικού δικτύου καθώς ο στόχος είναι να ελαχιστοποιηθεί η τιμή αυτής της συνάρτησης για τη βελτίωση της απόδοσης του μοντέλου. Υπάρχουν πολλές συναρτήσεις σφάλματος για διαφορετικές εργασίες η καθεμία.

Κάποιες από τις πιο γνωστές είναι:

- Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error - MSE): Η MSE, γνωστή και ως τετραγωνική, χρησιμοποιείται για εργασίες παλινδρόμησης (regression) όπου σκοπός είναι η πρόβλεψη συνεχών τιμών. Υπολογίζει τη μέση τετραγωνική διαφορά μεταξύ της πρόβλεψης (y_hat) και της πραγματικής τιμής (y).
  Μαθηματικά ορίζεται ως:

  $$L(y, \hat{y}) = \left(\frac{1}{N}\right) * \sum (y, \hat{y})^2 \quad (2.7)$$

  Όπου N είναι ο αριθμός των δειγμάτων, y η πραγματικές τιμές και $\hat{y}$ οι τιμές που πρόβλεψε το μοντέλο.

  Με τον τετραγωνισμό των διαφορών, η MSE δίνει έμφαση στα μεγαλύτερη σφάλματα, κάνοντας το μοντέλο πιο ευαίσθητο στις ακραίες τιμές.
  Μια άλλη ιδιότητα αυτής της συνάρτησης είναι ότι είναι ομαλή και διαφοροποιήσιμη κάτι που βοηθάει τους αλγόριθμους βελτιστοποίησης που βασίζονται σε διαβαθμίσεις.

Διασφαλίζει έτσι ότι οι μικρές αλλαγές στις παραμέτρους του μοντέλου έχουν ομαλές αλλαγές στη τιμή σφάλματος, κάνοντας τη διαδικασία βελτιστοποίησης πιο σταθερή.
Επίσης είναι πάντα θετική καθώς βασίζεται σε τετράγωνο και η ελάχιστη τιμή της μπορεί να είναι 0.

- Μέσο Απόλυτο Σφάλμα (Mean Absolute Error - MAE): Η MAE χρησιμοποιείται όπως και η MSE σε εργασίες παλινδρόμησης. Μετράει τις μέσες απόλυτες διαφορές μεταξύ των τιμών πρόβλεψης και των πραγματικών τιμών.

Μαθηματικά ορίζεται ως:

$$L(y, \hat{y}) = \left(\frac{1}{N}\right) * \sum |y, \hat{y}| \qquad (2.8)$$

Όπου N είναι ο αριθμός των δειγμάτων, y η πραγματικές τιμές και $\hat{y}$ οι τιμές που πρόβλεψε το μοντέλο.

Υπολογίζει τον μέσο όρο των απόλυτων διαφορών μεταξύ των τιμών που πρόβλεψε το μοντέλο και των πραγματικών τιμών και σε αντίθεση με την MSE δεν τετραγωνίζει τις διαφορές, καθιστώντας την λιγότερο ευαίσθητη σε ακραίες τιμές. Αυτή η ιδιότητα μπορεί να είναι χρήσιμη σε περιπτώσεις που η ανθεκτικότητα σε ακραίες τιμές είναι επιθυμητή.
Παρόλο που δεν είναι διαφοροποιήσιμη στο 0, λόγω της απόλυτης τιμής, είναι υποδιαφοροποιήσιμη, που σημαίνει ότι οι αλγόριθμοι βελτιστοποίησης μπορούν ακόμα να εφαρμοστούν για την ελαχιστοποίησή της.
Επίσης, όπως και η MSE, είναι πάντα θετική, λόγω της απόλυτης τιμής, οπότε η ελάχιστη τιμή που μπορεί να πάρει είναι 0.

- Συνάρτηση Σφάλματος Cross Entropy: Η συνάρτηση cross entropy είναι μια συνάρτηση σφάλματος που χρησιμοποιείται για εργασίες ταξινόμησης για δυαδική ταξινόμηση ή για ταξινόμηση πολλαπλών κλάσεων και μετράει την ανομοιότητα μεταξύ της προβλεπόμενης κατανομής πιθανότητας και της πραγματικής τιμής. Επίσης προέρχεται από την θεωρία της εντροπίας της θεωρίας πληροφοριών η οποία ποσοτικοποιεί την αβεβαιότητα μιας κατανομής πιθανοτήτων.

Για την ταξινόμηση πολλαπλών κλάσεων, μαθηματικά ορίζεται ως:

$$L(y, \hat{y}) = -\sum y_i * \log(\hat{y}_i) \qquad (2.9)$$

Όπου το y_i είναι η πραγματική ετικέτα για την κλάση i και το $\hat{y}_i$ είναι η προβλεπόμενη πιθανότητα για την κλάση i.

Για τη δυαδική ταξινόμηση, μαθηματικά ορίζεται ως:

$$L(y, \hat{y}) = -\sum(y * \log(\hat{y}) + (1 - y) * log(1 - \hat{y})) \quad (2.10)$$

Χαρακτηριστικά και ιδιότητες της cross entropy loss function

1. Ενθαρρύνει το μοντέλο να εκχωρήσει υψηλή πιθανότητα στη σωστή κλάση και τιμωρεί τις λάθος προβλέψεις.
2. Είναι ομαλή και διαφοροποιήσιμη που βοηθάει τους αλγόριθμους βελτιστοποίησης που βασίζονται σε διαβαθμίσεις. Η συνεχής φύση της συνάρτησης διασφαλίζει ότι μικρές αλλαγές στις παραμέτρους του μοντέλου έχουν ως αποτέλεσμα ομαλές αλλαγές στη τιμή σφάλματος, κάνοντας τη διαδικασία βελτιστοποίησης πιο σταθερή.
3. Έχει πάντα θετικές τιμές μιας και βασίζεται σε λογάριθμους πιθανοτήτων και η ελάχιστη τιμή που μπορεί να πάρει είναι 0, όταν η προβλεπόμενη κατανομή πιθανοτήτων είναι ίση με την πραγματική κατανομή.
4. Είναι ευαίσθητη σε λάθος ταξινομήσεις κλάσεων γιατί βάζει μεγάλη ποινή στις λάθος προβλέψεις. Αυτό είναι αρκετά σημαντικό για εργασίες που χρειάζονται μεγάλη ακρίβεια στη ταξινόμηση.
5. Και τέλος, λειτουργεί καλά σε συνδυασμό με τη συνάρτηση ενεργοποίησης Softmax στο επίπεδο εξόδου όπου κανονικοποιεί τις προβλεπόμενες πιθανότητες να αθροιστούν στο 1.

2.3.3 Ρυθμός Μάθησης

Ο ρυθμός μάθησης είναι μια σημαντική υπερπαράμετρος στην εκπαίδευση μοντέλων μηχανικής μάθησης και νευρωνικών δικτύων και ελέγχει το μέγεθος του βήματος κατά την ενημέρωση παραμέτρων τη διαδικασία της βελτιστοποίησης και επηρεάζει την ταχύτητα σύγκλισης και την τελική απόδοση του μοντέλου. Επίσης, αν ο ρυθμός μάθησης είναι πολύ μεγάλος, μπορεί να βιαστεί το μοντέλο να βρει μια λύση η οποία δεν θα είναι βέλτιστη, ή αν είναι πολύ μικρός να κολλήσει η διαδικασία και να πάρει υπερβολικά πολύ χρόνο για να βρει μια λύση.

Σημαντικές πτυχές του ρυθμού μάθησης:

- Σύγκλιση: Ο ρυθμός μάθησης επηρεάζει άμεσα τη σύγκλιση, τη διαδικασία μέσω της οποίας ένα νευρωνικό δίκτυο μαθαίνει , του αλγορίθμου βελτιστοποίησης. Ένας μεγάλος ρυθμός μάθησης μπορεί να προκαλέσει υπέρβαση της βέλτιστης τιμής και απόκλιση του αλγόριθμου, ενώ ένας μικρός ρυθμός μάθησης μπορεί να οδηγήσει σε αργή σύγκλιση ή και να κολλήσει σε τοπικά ελάχιστα.
- Trade-off ταχύτητας και ακρίβειας: Ένας μεγάλος ρυθμός μάθησης επιτρέπει στο μοντέλο να μαθαίνει πιο γρήγορα αλλά μπορεί να προκαλέσει ταλαντώσεις γύρω από τη βέλτιστη τιμή και να εμποδίσει το μοντέλο να φτάσει μεγάλη ακρίβεια. Ένας μικρός ρυθμός

μάθησης εξασφαλίζει πιο σταθερή σύγκλιση αλλά μπορεί να απαιτεί περισσότερες επαναλήψεις για να φτάσει στη βέλτιστη τιμή.

- Ευαισθησία στις αρχικές συνθήκες: Επίσης μπορεί να επηρεάσει την ευαισθησία του αλγόριθμου βελτιστοποίησης στις αρχικές τιμές των παραμέτρων του μοντέλου. Ο μεγάλος ρυθμός μάθησης μπορεί να προκαλέσει απόκλιση του αλγόριθμου, ενώ ο μικρός ρυθμός μάθησης να βοηθήσει τον αλγόριθμο να συγκλίνει πιο σταθερά ανεξάρτητα από το σημείο εκκίνησης.
- Προσαρμοστικός ρυθμός μάθησης: Στους προηγμένους αλγόριθμους βελτιστοποίησης, όπως ο ADAM, χρησιμοποιείται προσαρμοστικός ρυθμός μάθησης, δηλαδή προσαρμόζουν τον ρυθμό μάθησης για κάθε παράμετρο κατά τη διάρκεια εκπαίδευσης. Έτσι έχουμε ταχύτερη σύγκλιση και καλύτερη απόδοση.
- Προγραμματισμός ρυθμού μάθησης: Είναι σύνηθες να χρησιμοποιούνται τεχνικές προγραμματισμού του ρυθμού μάθησης για την προσαρμογή του κατά τη διάρκεια της εκπαίδευσης. Κάποιες δημοφιλείς τεχνικές είναι η εκθετική μείωση (exponential decay) και μείωση βήματος (step decay) και βοηθούν στη βελτίωση των ιδιοτήτων σύγκλισης του αλγόριθμου βελτιστοποίησης και καλύτερη απόδοση του μοντέλου.

Γενικά, η επιλογή κατάλληλου ρυθμού μάθησης είναι απαραίτητη για τη βέλτιστη σύγκλιση και απόδοση του μοντέλου.

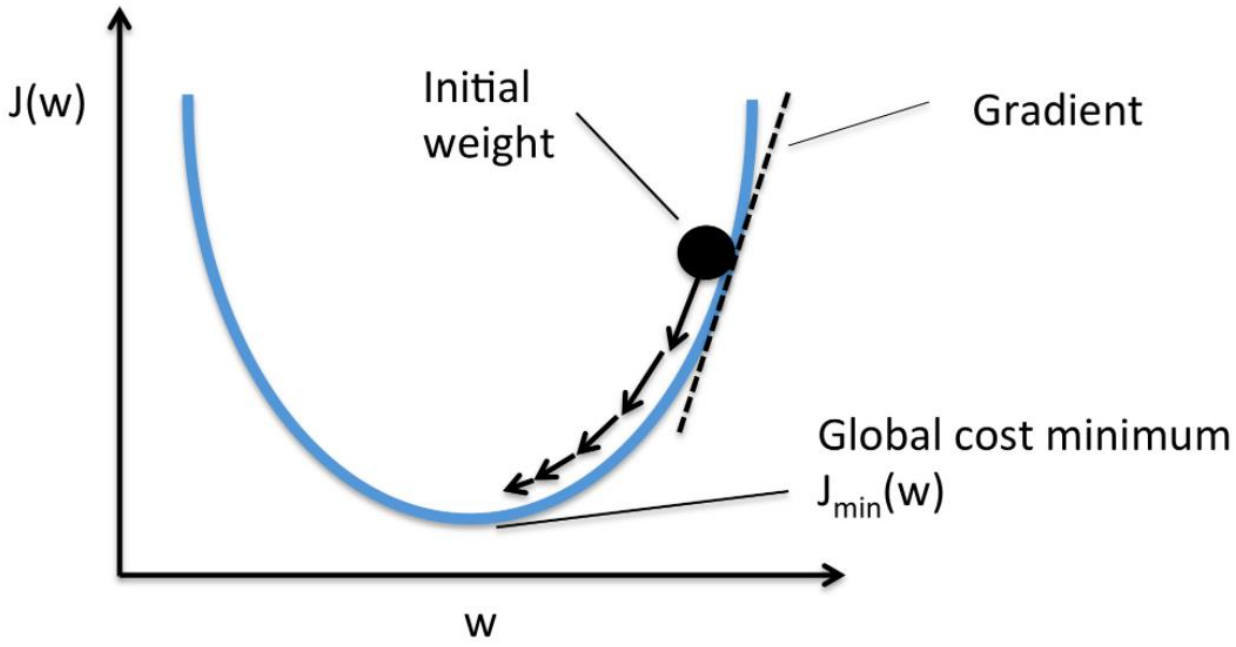
2.3.4 Αλγόριθμοι βελτιστοποίησης

Οι αλγόριθμοι βελτιστοποίησης αποτελούν ένα πολύ σημαντικό κομμάτι της εκπαιδευτικής διαδικασίας ενός νευρωνικού δικτύου. Σκοπός είναι να ελαχιστοποιηθεί η διαφορά μεταξύ των προβλέψεων από το μοντέλο και των πραγματικών ετικετών, προσαρμόζοντας τις παραμέτρους του μοντέλου, βάρη και bias, σε κάθε επανάληψη για να βελτιώσουν την απόδοση του μοντέλου στα δεδομένα εκπαίδευσης. Οι αλγόριθμοι αυτοί παίζουν κρίσιμο ρόλο στην εύρεση του καλύτερου δυνατού συνόλου αυτών των παραμέτρων που οδηγούν σε καλύτερες προβλέψεις σε δεδομένα που δεν έχει ξαναδεί το μοντέλο. Στην ουσία ο αλγόριθμος βελτιστοποίησης καθοδηγεί το νευρωνικό δίκτυο σε καλύτερες αναπαραστάσεις εισόδου άρα και καλύτερη απόδοση στη δεδομένη εργασία.

Παρακάτω θα δούμε κάποιους από τους πιο δημοφιλείς αλγόριθμους βελτιστοποίησης.

- Gradient Descent: Ο Gradient Descent είναι ένας αλγόριθμος βελτιστοποίησης πρώτης τάξης που χρησιμοποιείται ευρέως για την εκπαίδευση μοντέλων μηχανικής μάθησης. Ο κύριος στόχος του είναι να ελαχιστοποιήσει τη συνάρτηση σφάλματος ενημερώνοντας σε κάθε επανάληψη τα βάρη και το bias του μοντέλου. Η βασική ιδέα πίσω αυτόν τον αλγόριθμο είναι ότι η κλίση (παράγωγος) της συνάρτησης σφάλματος σε σχέση με τις παραμέτρους του μοντέλου, δείχνει την κατεύθυνση της πιο απότομης αύξησης στη

συνάρτηση σφάλματος, και κατά συνέπεια, η αρνητική κλίση δείχνει προς την πιο απότομη μείωση. Έτσι, μετακινώντας προς την κατεύθυνση της αρνητικής κλίσης, ο αλγόριθμος στοχεύει να φτάσει στο ελάχιστο σημείο της συνάρτησης σφάλματος.



Σχήμα 2.6 Gradient Descent

Πιο αναλυτικά:
1. Αρχικοποίηση των βαρών και bias.
2. Υπολογισμός της κλίσης της συνάρτησης σφάλματος σε σχέση με κάθε παράμετρο.
3. Ενημέρωση των παραμέτρων, βάρη και bias, κάνοντας ένα βήμα προς την κατεύθυνση της αρνητικής κλίσης με βάση τον ρυθμό μάθησης.
4. Επανάληψη των βημάτων 2 – 3 μέχρι ο αλγόριθμος να συγκλίνει ή να φτάσει σε ένα προκαθορισμένο σημείο διακοπής.

Ο κανόνας ενημέρωσης των παραμέτρων για αυτόν τον αλγόριθμο μαθηματικά ορίζεται ως:

$$\Theta = \Theta - \eta * \nabla L(\Theta) \qquad (2.11)$$

όπου:
Θ, είναι οι παράμετροι του μοντέλου (βάρη, bias),
η, είναι ο ρυθμός μάθησης,

∇L(Θ), είναι η κλίση της συνάρτησης σφάλματος ως προς τις παραμέτρους.

Χαρακτηριστικά του αλγόριθμου Gradient Descent:

Σύγκλιση: Ο αλγόριθμος Gradient Descent είναι εγγυημένο ότι συγκλίνει σε ένα συνολικό ελάχιστο για convex συναρτήσεις σφάλματος και σε τοπικό ελάχιστο για μη-convex συναρτήσεις, με βάση το ρυθμό μάθησης.

Ρυθμός Μάθησης: Ο ρυθμός μάθησης είναι μια σημαντική υπερπαράμετρος που καθορίζει το μέγεθος του βήματος κάθε ενημέρωσης παραμέτρου. Αν ο ρυθμός μάθησης είναι μεγάλος μπορεί να προκαλέσει υπέρβαση του ελάχιστου και απόκλιση, ενώ ένας μικρός ρυθμός μάθησης μπορεί να οδηγήσει σε αργή σύγκλιση.

Επεκτασιμότητα: Μπορεί να είναι υπολογιστικά ακριβός αλγόριθμος για μεγάλα datasets μιας και επεξεργάζεται ολόκληρο το dataset σε κάθε επανάληψη.

Ευαισθησία: Είναι αρκετά ευαίσθητος στις αρχικές τιμές των παραμέτρων, γι’ αυτό τεχνικές όπως η τυχαία αρχικοποίηση των παραμέτρων βοηθούν στην βελτίωση της σύγκλισης.

- Stochastic Gradient Descent (SGD): Ο αλγόριθμος SGD είναι μια επέκταση του Gradient Descent που λύνει κάποια από τα προβλήματά του, όπως ότι είναι υπολογιστικά ακριβό σε μεγάλα datasets. Όπως και ο Gradient Descent, κύριος στόχος είναι να ελαχιστοποιήσει τη συνάρτηση σφάλματος ενημερώνοντας σε κάθε επανάληψη τα βάρη και το bias. Η κύρια διαφορά τους βρίσκεται στη μέθοδο υπολογισμού της κλίσης. Ενώ ο Gradient Descent υπολογίζει την κλίση χρησιμοποιώντας ολόκληρο το dataset, ο SGD προσεγγίζει τη κλίση χρησιμοποιώντας ένα μόνο δείγμα σε κάθε επανάληψη που επιλέγεται τυχαία από το dataset. Αυτή η προσέγγιση μειώνει την υπολογιστική ισχύ και τη μνήμη που χρειάζεται κάνοντάς τον καταλληλότερο για μεγάλα datasets.

  Πιο αναλυτικά:

  1. Αρχικοποίηση των βαρών και bias.
  2. Τυχαία επιλογή ενός δείγματος από το dataset.
  3. Υπολογισμός της κλίσης της συνάρτησης σφάλματος σε σχέση με κάθε παράμετρο χρησιμοποιώντας το επιλεγμένο δείγμα.
  4. Ενημέρωση των παραμέτρων κάνοντας ένα βήμα προς την κατεύθυνση της αρνητικής κλίσης με βάση τον ρυθμό μάθησης.
  5. Επανάληψη των βημάτων 2 – 4 μέχρι ο αλγόριθμος να συγκλίνει ή να φτάσει σε ένα προκαθορισμένο σημείο διακοπής.

Βασικά χαρακτηριστικά του αλγόριθμου SGD:

Θόρυβος: Αφού ο αλγόριθμος SGD χρησιμοποιεί μόνο ένα δείγμα από το dataset για τον υπολογισμό της κλίσης, οι ενημερώσεις έχουν θόρυβο οδηγώντας ίσως σε πιο αργή σύγκλιση.

Ορμή: Για να μειωθεί ο θόρυβος στις ενημερώσεις των παραμέτρων και να σταθεροποιηθεί η βελτιστοποίηση, μπορεί να εισαχθεί ορμή. Η ορμή προσθέτει ένα κλάσμα της προηγούμενης ενημέρωσης στην τρέχουσα ενημέρωση για να ομαλοποιήσει τη βελτιστοποίηση και να επιταχύνει τη σύγκλιση.

Προγραμματισμός ρυθμού μάθησης: Αυτή η τεχνική προγραμματισμού του ρυθμού μάθησης (πχ step decay, exponential decay) χρησιμοποιείται για την προσαρμογή του ρυθμού μάθησης κατά τη διάρκεια της εκπαίδευσης βελτιώνοντας τις ιδιότητες σύγκλισης του αλγορίθμου.

Mini-batch SGD: Mini-batch SGD είναι μια παραλλαγή του SGD που επιτυγχάνει μια ισορροπία μεταξύ Gradient Descent και SGD υπολογίζοντας τη κλίση χρησιμοποιώντας ένα μικρό υποσύνολο των δεδομένων εκπαίδευσης σε κάθε επανάληψη. Αυτή η προσέγγιση συνδυάζει την υπολογιστική απόδοση του SGD με τις πιο σταθερές εκτιμήσεις κλίσης του Gradient Descent οδηγώντας σε ταχύτερη σύγκλιση.

ADAM (Adaptive Moment Estimation): Ο ADAM είναι ένας προηγμένος αλγόριθμος βελτιστοποίησης για την εκπαίδευση μοντέλων μηχανικής μάθησης και νευρωνικών δικτύων. Συνδυάζει τις έννοιες της ορμής και της προσαρμογής του ρυθμού μάθησης κάνοντάς τον έναν από τους πιο δημοφιλή αλγορίθμους βελτιστοποίησης.

Ο ADAM διατηρεί ξεχωριστούς κινητούς μέσους όρους για τις κλίσεις και τις τετράγωνες κλίσεις και στη συνέχεια υπολογίζει διορθωμένες με bias τις εκτιμήσεις πρώτης και δεύτερης στιγμής που χρησιμοποιούνται για τη προσαρμογή του ρυθμού μάθησης για κάθε παράμετρο ξεχωριστά. Αυτός ο μηχανισμός προσαρμογής του ρυθμού μάθησης επιτρέπει στον αλγόριθμο να συγκλίνει ταχύτερα και να επιτυγχάνει καλύτερη απόδοση σε σχέση με τις άλλες μεθόδους βελτιστοποίησης.

Πιο αναλυτικά:

- Αρχικοποίηση των βαρών και bias.

- Αρχικοποίηση των μεταβλητών πρώτης και δεύτερης στιγμής (m και v αντίστοιχα) για κάθε παράμετρο στο 0.
- Υπολογισμός της κλίσης της συνάρτησης σφάλματος σε σχέση με κάθε παράμετρο χρησιμοποιώντας τα δεδομένα εκπαίδευσης.
- Ενημέρωση των μεταβλητών πρώτης και δεύτερης στιγμής χρησιμοποιώντας της υπολογισμένη κλίση και τους ρυθμούς εκθετικής μείωσης (exponential decay rate) β1 και β2.
- Υπολογισμός των διορθωμένων με βάση το bias εκτιμήσεων της πρώτης και δεύτερης στιγμής.
- Ενημέρωση των παραμέτρων κάνοντας ένα βήμα προς την κατεύθυνση της αρνητικής κλίσης, κλιμακούμενης με τον προσαρμοσμένο ρυθμό μάθησης.
- Επανάληψη των βημάτων 3 – 6 μέχρι ο αλγόριθμος να συγκλίνει ή να φτάσει σε ένα προκαθορισμένο σημείο διακοπής.

Ο κανόνας ενημέρωσης των παραμέτρων για αυτόν τον αλγόριθμο μαθηματικά ορίζεται ως:

$$\Theta = \Theta - \eta * (\frac{\hat{m}}{\sqrt{\hat{v}} + \varepsilon}) \qquad (2.12)$$

Όπου,

Θ, είναι οι παράμετροι του μοντέλου,

η, είναι ο ρυθμός μάθησης,

$\hat{m}$ και $\hat{v}$ είναι οι διορθωμένες με βάση το bias εκτιμήσεις της πρώτης και δεύτερης στιγμής αντίστοιχα,

ε, μια μικρή σταθερά για που προστίθεται για αριθμητική σταθερότητα, (συνήθως 1e-8).

Βασικά χαρακτηριστικά του αλγόριθμου ADAM:

- Προσαρμοστικός ρυθμός μάθησης: Ο ADAM προσαρμόζει τον ρυθμό μάθησης για κάθε παράμετρο με βάση τις προηγούμενες κλίσεις και τα μεγέθη τους. Αυτό επιτρέπει στον αλγόριθμο να κάνει μεγαλύτερες ενημερώσεις για σπάνιες ή πιο σημαντικές παραμέτρους και μικρότερες ενημερώσεις για συχνές ή λιγότερο σημαντικές παραμέτρους και έχει ως αποτέλεσμα ταχύτερη σύγκλιση και καλύτερη απόδοση.
- Ορμή: Ενσωματώνει την έννοια της ορμής διατηρώντας έναν κινητό μέσο όρο των κλίσεων (πρώτη στιγμή). Αυτό βοηθάει στη σταθεροποίηση της βελτιστοποίησης αποτρέποντας τις ταλαντώσεις και επιτρέποντας ταχύτερη σύγκλιση.
- Σύγκλιση: Ο αλγόριθμος ADAM γενικά συγκλίνει πιο γρήγορα από άλλους αλγόριθμους βελτιστοποίησης ειδικά στη βαθιά μάθηση.

- Υπερπαράμετροι: Ο ADAM εισάγει 2 επιπλέον υπερπαράμετρους, τη β1 και τη β2, οι οποίες ελέγχουν τους εκθετικούς ρυθμούς μείωσης των μεταβλητών πρώτης και δεύτερης στιγμής. Οι προεπιλεγμένες τιμές είναι β1 = 0.9 και β2 = 0.999.
- Ανθεκτικότητα: Ο ADAM θεωρείται πιο ανθεκτικός στην επιλογή ρυθμού μάθησης και άλλων υπερπαραμέτρων σε σύγκριση με άλλους αλγόριθμους, κάνοντάς τον αρκετά δημοφιλή επιλογή για την εκπαίδευση μοντέλων βαθιάς μάθησης.

2.3.5 Τεχνικές Regularization

Οι τεχνικές regularization χρησιμοποιούνται για την αποφυγή της υπερπροσαρμογής στα μοντέλα μηχανικής μάθησης και νευρωνικά δίκτυα. Η υπερπροσαρμογή συμβαίνει όταν ένα μοντέλο μαθαίνει τον θόρυβο στα δεδομένα εκπαίδευσης και όχι το μοτίβο, με αποτέλεσμα κακή γενίκευση σε δεδομένα που δεν έχει δει το δίκτυο. Οι τεχνικές regularization προσθέτουν έναν όρο ποινής στη συνάρτηση σφάλματος, κάνοντας το μοντέλο να μάθει πιο απλά και σταθερά μοτίβα.

Μερικές δημοφιλείς τεχνικές που χρησιμοποιούνται στα νευρωνικά δίκτυα:

- L1 (Κανονικοποίηση Lasso): Η L1 τεχνική προσθέτει την απόλυτη τιμή των βαρών στη συνάρτηση σφάλματος πολλαπλασιασμένη με μια παράμετρο λάμδα. Αυτή η τεχνική τείνει να παράγει πίνακες αραιού βάρους, μηδενίζοντας αποτελεσματικά ορισμένα βάρη, καταλήγοντας σε ένα πιο απλό μοντέλο.

$$Loss = Original\ Loss + \lambda * \sum \left| W^{l_{ij}} \right| \tag{2.13}$$

- L2 (Ridge regularization): Η τεχνική L2 προσθέτει τη τετραγωνική τιμή των βαρών στη συνάρτηση σφάλματος πολλαπλασιασμένη με μια παράμετρο λάμδα. Αυτή η τεχνική αποθαρρύνει τα μεγάλα βάρη χωρίς να τα κάνει 0, με αποτέλεσμα ένα πιο ισορροπημένο μοντέλο.

$$Loss = Original\ Loss + \lambda * \sum \left(W^{l_{ij}} \right)^2 \tag{2.14}$$

- Dropout: Το Dropout είναι μια τεχνική ειδικά σχεδιασμένη για νευρωνικά δίκτυα για να αποτρέψει την υπερπροσαρμογή και παρουσιάστηκε το 2014. Αυτή η τεχνική συμβάλει στη βελτίωση της γενίκευσης του μοντέλου με την τυχαία απομάκρυνση ενός κλάσματος των εξόδων των νευρώνων σε κάθε επανάληψη κατά τη διάρκεια της εκπαίδευσης κάνοντάς το δίκτυο λιγότερο επιρρεπές σε υπερπροσαρμογή. Επίσης πολλές φορές συνδυάζεται με άλλες τεχνικές όπως L1 ή L2.

- Πιο αναλυτικά, κατά τη διάρκεια της εκπαίδευσης σε κάθε επανάληψη, ένα τυχαίο υποσύνολο της εξόδου των νευρώνων μηδενίζεται με καθορισμένη πιθανότητα (ρυθμός εγκατάλειψης). Το ποσοστό αυτό ορίζεται συνήθως από 0.20 έως 0.50 ανάλογα με τη πολυπλοκότητα του μοντέλου και τον όγκο δεδομένων εκπαίδευσης. Ρυθμίζοντας την έξοδο του νευρώνα στο μηδέν αφαιρείται από το δίκτυο για τη συγκεκριμένη επανάληψη, αναγκάζοντας τους υπόλοιπους νευρώνες να μάθουν πιο δυνατά χαρακτηριστικά και αποτρέπει έτσι την υπερβολική εξάρτηση από κάποιο μεμονωμένο νευρώνα. Όταν γίνεται το πίσω πέρασμα, γίνεται η ενημέρωση των βαρών λαμβάνοντας υπόψη μόνο τους νευρώνες που η έξοδός τους δεν έγινε μηδέν κατά τη διάρκεια του μπροστινού περάσματος.
- Κατά τη διάρκεια προβλέψεων σε δεδομένα που δεν έχει δει το δίκτυο χρησιμοποιούνται όλοι οι νευρώνες, παρόλα αυτά τα αποτελέσματά τους κλιμακώνονται με βάση το ποσοστό εγκατάλειψης, dropout, για να ληφθεί υπόψη ότι περισσότεροι νευρώνες είναι ενεργοί σε σύγκριση με την εκπαίδευση. Αυτή η κλιμάκωση είναι απαραίτητη για να διασφαλιστεί ότι τα συνολικά μεγέθη ενεργοποίησης είναι τα ίδια μεταξύ inference και εκπαίδευσης.

Πλεονεκτήματα του Dropout:

- Βελτιώνει τη γενίκευση: Αποτρέποντας την υπερβολική εξάρτηση από οποιονδήποτε μεμονωμένο νευρώνα, ενθαρρύνει το δίκτυο να μάθει πιο ισχυρά χαρακτηριστικά, το dropout βοηθάει στη βελτίωση της γενίκευσης του μοντέλου σε δεδομένα που δεν έχει δει κατά την εκπαίδευση.
- Μειώνει την υπερπροσαρμογή: Με το τυχαίο dropout νευρώνων κατά την διάρκεια της εκπαίδευσης, το dropout εκπαιδεύει αποτελεσματικά ένα σύνολο μικρότερων δικτύων με διαφορετικούς συνδυασμούς νευρώνων. Το τελικό μοντέλο θεωρείται και σύνολο αυτών των μικρότερων δικτύων που είναι πιο στιβαρό στην υπερπροσαρμογή.
- Το dropout δεν αυξάνει αρκετά τη πολυπλοκότητα της εκπαίδευσης του δικτύου.
- Επίσης, είναι αρκετά εύκολο στην υλοποίηση απλά με την προσθήκη επιπέδων dropout ανάμεσα στα υπόλοιπα επίπεδα.

- Batch Normalization: Το batch normalization είναι μια τεχνική για την αντιμετώπιση της εσωτερικής μετατόπισης συμμεταβλητών στα βαθιά νευρωνικά δίκτυα και παρουσιάστηκε το 2015. Η εσωτερική μετατόπιση συμμεταβλητής συμβαίνει όταν η κατανομή των δεδομένων εισόδων σε ένα επίπεδο αλλάζει κατά τη διάρκεια της εκπαίδευσης με αποτέλεσμα η εκπαίδευση να γίνεται πιο αργή και λιγότερο σταθερή. Η ομαλοποίηση αυτή που κάνει το batch normalization βοηθάει στο να γίνει πιο γρήγορη η εκπαίδευση του δικτύου και να γίνει πιο αποδοτικό κάνοντας κανονικοποίηση των εισόδων σε κάθε επίπεδο για να πετύχει σταθερό μέσο όρο και διακύμανση.

- Υπολογίζει τη μέση τιμή (μ) και τη διακύμανση (σ^2) για ένα δεδομένο υποσύνολο των δεδομένων εισόδων για κάθε χαρακτηριστικό ή ενεργοποίηση.
- Στη συνέχεια κανονικοποιεί τις εισόδους αφαιρώντας τη μέση τιμή και διαιρώντας με την τετραγωνική ρίζα της διακύμανσης συν μια μικρή σταθερά (ε) για να αποφύγει τη διαίρεση με το μηδέν. Αυτό το βήμα διασφαλίζει ότι οι κανονικοποιημένες είσοδοι έχουν μέσο όρο μηδέν και διακύμανση 1.

$$x_{normalized} = \frac{(X-\mu)}{\sqrt{\sigma^2+\varepsilon}} \quad (2.15)$$

- Εφαρμόζει μια παράμετρο κλίμακας (γ) και μετατόπισης (β) στις κανονικοποιημένες εισόδου. Αυτές οι παράμετροι επιτρέπουν στο δίκτυο να προσαρμόζει τον μέσο όρο και τη διακύμανση των ενεργοποιήσεων κατά τη διάρκεια της εκπαίδευσης. Η έξοδος του επιπέδου batch normalization υπολογίζεται έτσι:

$$Y = \gamma * X_{normalized} + \beta \quad (2.16)$$

και οι δύο αυτές παράμετροι, β και γ, ενημερώνονται κατά τη διάρκεια της εκπαίδευσης χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης όπως ο gradient descent ή adam. Η τελική έξοδος Y έχει τις ίδιες διαστάσεις με την είσοδο X.

- Early Stopping: Το early stopping είναι άλλη μια regularization τεχνική που χρησιμοποιείται για να αποφύγει την υπερπροσαρμογή στην εκπαίδευση νευρωνικών δικτύων. Η έγκαιρη διακοπή της εκπαίδευσης επιτρέπει στην εύρεση της σωστής ισορροπίας μεταξύ της υποπροσαρμογής και της υπερπροσαρμογής παρακολουθώντας την απόδοση του μοντέλου στο validation dataset και σταματάει τη διαδικασία εκπαίδευσης όταν η απόδοση του μοντέλου αρχίσει να πέφτει.

  Πως λειτουργεί:

  - Διαχωρίζουμε το dataset σε 3 υποσύνολα, training, validation και test set, για την εκπαίδευση, για την παρακολούθηση της απόδοσης κατά τη διάρκεια της εκπαίδευσης και για την αξιολόγηση του τελικού μοντέλου αντίστοιχα.
  - Ανά κάποια χρονικά διαστήματα κατά την εκπαίδευση (πχ μετά από κάθε εποχή), γίνεται αξιολόγηση του μοντέλου στο validation set. Μια συνηθισμένη μέτρηση είναι η ακρίβεια ή το μέσο τετραγωνικό σφάλμα, ανάλογα με το πρόβλημα.

- Ορίζουμε μια παράμετρο υπομονής, η οποία καθορίζει τον αριθμό εποχών πριν από τη διακοπή της εκπαίδευσης εφόσον η απόδοση του μοντέλου δεν βελτιωθεί. Επίσης, μπορούμε να ορίσουμε ένα ελάχιστο όριο βελτίωσης, για την ελάχιστη βελτίωση που χρειάζεται για να μηδενίσει ο μετρητής υπομονής.
- Κατά τη διάρκεια της εκπαίδευσης εάν η απόδοση του μοντέλου στο validation set δεν βελτιωθεί κατά το ελάχιστο όριο βελτίωσης για τον αριθμό συνεχόμενων εποχών που ορίσαμε, η εκπαίδευση σταματάει. Αυτό αποτρέπει το μοντέλο από την υπερπροσαρμογή του μοντέλου στα δεδομένα εκπαίδευσης.
- Τέλος, μετά τη διακοπή της εκπαίδευσης, επιλέγεται το μοντέλο με την καλύτερη απόδοση στο validation set και αξιολογείται στο test set για να γίνει μια εκτίμηση της απόδοσής του σε δεδομένα που δεν έχει ξαναδεί.

Επίσης το early stopping βοηθάει στη μείωση του χρόνου εκπαίδευσης και έτσι θα εξοικονομήσει χρόνο και υπολογιστικούς πόρους και είναι αρκετά απλό στη υλοποίηση χωρίς να χρειάζεται αλλαγές η αρχιτεκτονική του νευρωνικού δικτύου.

3. Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural networks - CNN)

3.1 Τι είναι τα συνελικτικά δίκτυα

Τα συνελικτικά δίκτυα είναι μια κατηγορία νευρωνικών δικτύων για εργασίες επεξεργασίας εικόνας και μηχανικής όρασης, όπως η ταξινόμηση εικόνων και η αναγνώριση αντικειμένων και μπορούν να μαθαίνουν αυτόματα ιεραρχικά χαρακτηριστικά από ακατέργαστα δεδομένα εικόνας. Το κύριο κίνητρο των συνελικτικών δικτύων είναι ότι δεδομένα όπως οι εικόνες έχουν μοναδικές ιδιότητες που μπορούν να αξιοποιηθούν με συγκεκριμένες αρχιτεκτονικές νευρωνικών δικτύων όπως τα συνελικτικά δίκτυα.

Τα συνελικτικά δίκτυα έχουν τρείς βασικές ιδέες:

1. Τα συνελικτικά δίκτυα εκμεταλλεύονται την ιδιότητα των pixel ότι δεν είναι ανεξάρτητα και σχετίζονται χωρικά συνδέοντας κάθε νευρώνα σε ένα συνελικτικό επίπεδο σε μια τοπική περιοχή της εισόδου και όχι σε ολόκληρη την είσοδο. Αυτή η τοπική συνδεσιμότητα επιτρέπει στο δίκτυο να μαθαίνει τοπικά μοτίβα και μειώνει τον αριθμό παραμέτρων που πρέπει να μάθει, κάνοντάς το δίκτυο πιο αποτελεσματικό.
2. Σε ένα cnn χρησιμοποιείται ένα σύνολο φίλτρων με δυνατότητα εκμάθησης σε ολόκληρη την εικόνα εισόδου. Αυτή η κοινή χρήση παραμέτρων μειώνει σημαντικά τον αριθμό των παραμέτρων με δυνατότητα εκμάθησης στο δίκτυο και επιτρέπει στο δίκτυο να μαθαίνει

χαρακτηριστικά που δεν μεταβάλλουν το μοτίβο. Το ίδιο φίλτρο μπορεί να χρησιμοποιηθεί για την ανίχνευση ενός συγκεκριμένου μοτίβου ανεξάρτητα από τη θέση του μέσα στην εικόνα.

3. Τα cnn αποτελούνται από πολλαπλά επίπεδα που το κάθε ένα είναι υπεύθυνο για την εκμάθηση όλο και πιο αφηρημένων χαρακτηριστικών από τις εικόνες εισόδου. Τα χαμηλότερα επίπεδα πιάνουν τοπικά μοτίβα όπως άκρες και υφές, και τα βαθύτερα επίπεδα μαθαίνουν να αναγνωρίζουν πιο πολύπλοκα χαρακτηριστικά, όπως μέρη αντικειμένων και ολόκληρα αντικείμενα.

Γι' αυτό το λόγο τα συνελικτικά δίκτυα είναι αρκετά δημοφιλή σε μεγάλο φάσμα εφαρμογών της μηχανικής όρασης, όπως:

- Ταξινόμηση εικόνας,
- Αναγνώριση αντικειμένων,
- Semantic Segmentation,
- Δημιουργία νέων εικόνων.

Σε σύγκριση με τις παραδοσιακές τεχνικές μηχανικής μάθησης για ταξινόμηση εικόνων, τα cnn προσφέρουν πολλά πλεονεκτήματα, όπως να μαθαίνουν αυτόματα σχετικές λειτουργίες από ακατέργαστα δεδομένα εικόνας χωρίς να βασίζονται σε χειροκίνητες εργασίες που είναι χρονοβόρες. Επίσης, έχουν τη δυνατότητα να μαθαίνουν πολύπλοκες μη γραμμικές σχέσεις εισόδου – εξόδου, κάνοντάς τα κατάλληλα για μοντελοποίηση των πολύπλοκων οπτικών δεδομένων.

Τα βασικά επίπεδα των συνελικτικών δικτύων είναι το συνελικτικό επίπεδο, από όπου πήρε και το όνομά του το δίκτυο, το επίπεδο συγκέντρωσης, και το πλήρως διασυνδεδεμένο επίπεδο. Παρακάτω θα δούμε περισσότερα για αυτά τα επίπεδα.

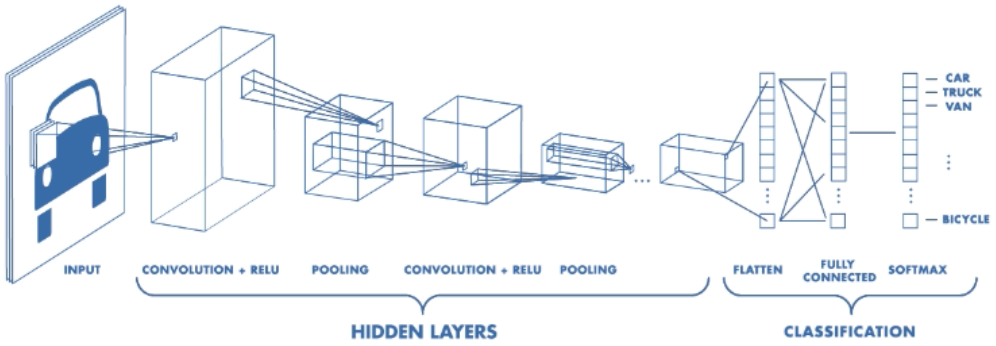
3.2 Αρχιτεκτονική CNN

Τα συνελικτικά νευρωνικά δίκτυα, cnn, όπως είδαμε έχουν σχεδιαστεί για την επεξεργασία δεδομένων που μοιάζουν με πλέγμα όπως οι εικόνες και οι χωρικές σχέσεις μεταξύ των pixel είναι σημαντικές.

Η αρχιτεκτονική ενός cnn αποτελείται από πολλά επίπεδα, το κάθε ένα με τη δική του λειτουργία, που είναι οργανωμένα σε τρεις κύριους τύπους, τα συνελικτικά επίπεδα, τα επίπεδα συγκέντρωσης και τα πλήρως συνδεδεμένα επίπεδα και που συνεργάζονται για να μετατρέψουν την εικόνα εισόδου σε μια τελική έξοδο, που συνήθως είναι μια κατανομή πιθανότητας σε διάφορες κλάσεις. Εκτός από τους τρεις βασικούς τύπους επιπέδων, υπάρχουν επίσης τα επίπεδα dropout, batch normalization και επίπεδα ενεργοποίησης που προσφέρουν καλύτερη απόδοση και σταθερότητα στο δίκτυο.

Μια τυπική αρχιτεκτονική cnn αποτελείται από τα παρακάτω επίπεδα και στοιχεία:

1. Επίπεδο εισόδου: Το επίπεδο εισόδου είναι υπεύθυνο για τη λήψη των ακατέργαστων δεδομένων εικόνας με τη μορφή πίνακα, με τις διαστάσεις του να είναι το ύψος, το πλάτος και ο αριθμός των καναλιών χρώματος (πχ 3 για έγχρωμες εικόνες). Το επίπεδο εισόδου δεν εκτελεί κάποια επεξεργασία απλά περνάει τα δεδομένα στα επόμενα επίπεδα.
2. Συνελικτικό επίπεδο: Το συνελικτικό επίπεδο είναι το βασικό επίπεδο των cnn. Εκτελούν τη λειτουργία της συνέλιξης, όπου περιλαμβάνει την ολίσθηση των φίλτρων πάνω από την εικόνα εισόδου, ανιχνεύοντας τοπικά μοτίβα, όπως άκρες, υφές και σχήματα. Κάθε φίλτρο είναι υπεύθυνο για την καταγραφή ενός συγκεκριμένου χαρακτηριστικού στα δεδομένα εισόδου. Η έξοδος του συνελικτικού επιπέδου ονομάζεται χάρτης χαρακτηριστικών (feature map) όπου αντιπροσωπεύει τη χωρική διάταξη των χαρακτηριστικών που ανιχνεύθηκαν. Πολλαπλά συνελικτικά επίπεδα μπορούν να στοιβάζουν σε ένα cnn, επιτρέποντας στο δίκτυο να μαθαίνει όλο και πιο πολύπλοκα χαρακτηριστικά σε πιο βαθιά επίπεδα.
3. Συνάρτηση ενεργοποίησης: Μετά τη λειτουργία της συνέλιξης, μια συνάρτηση ενεργοποίησης εφαρμόζεται κατά στοιχεία στους χάρτες χαρακτηριστικών για την εισαγωγή μη γραμμικότητας. Οι μη γραμμικές συναρτήσεις επιτρέπουν στο δίκτυο να μάθει πολύπλοκες μη γραμμικές σχέσεις εισόδου – εξόδου.
4. Επίπεδο συγκέντρωσης: Το επίπεδο συγκέντρωσης χρησιμοποιείται για τη μείωση των χωρικών διαστάσεων στους χάρτες χαρακτηριστικών, διατηρώντας παράλληλα τις πιο σημαντικές πληροφορίες. Αυτό βοηθάει στη μείωση της υπολογιστικής ισχύς που χρειάζεται και στην υπερπροσαρμογή. Η συγκέντρωση πραγματοποιείται εφαρμόζοντας μια συνάρτηση συνάθροισης σε μη επικαλυπτόμενες περιοχές του χάρτη. Μια δημοφιλής μέθοδός είναι η μέγιστη συγκέντρωση (max pooling).
5. Flatten επίπεδο: Το επίπεδο flatten χρησιμοποιείται για τη μετατροπή της 3D εξόδου ενός συνελικτικού ή συγκέντρωσης επίπεδου σε 1D διάνυσμα.
6. Πλήρως συνδεδεμένο επίπεδο: Το πλήρως συνδεδεμένο επίπεδο, γνωστό και ως πυκνό (dense) επίπεδο, χρησιμοποιούνται για την ενοποίηση των εξαγόμενων χαρακτηριστικών από τα προηγούμενα επίπεδα και την πραγματοποίηση της πρόβλεψης. Οι νευρώνες σε αυτό το επίπεδο συνδέονται με όλους τους νευρώνες του προηγούμενου επίπεδου και τα βάρη τους μαθαίνονται κατά τη διάρκεια της εκπαίδευσης. Επίσης, συνήθως είναι μετά το τελευταίο επίπεδο συνέλιξης ή συγκέντρωσης και χρησιμοποιούνται για την παραγωγή του διανύσματος εξόδου.
7. Επίπεδο εξόδου: Το επίπεδο εξόδου είναι το τελευταίο επίπεδο και είναι υπεύθυνο για την παραγωγή των τελικών πιθανοτήτων ταξινόμησης. Συνήθως αποτελείται από ένα πλήρως συνδεδεμένο επίπεδο με τον αριθμό των νευρώνων να είναι ίδιος με τον αριθμό κλάσεων σε κάποιο πρόβλημα ταξινόμησης. Επίσης, συνήθως χρησιμοποιείται και μια συνάρτηση Softmax για να μετατρέψει την έξοδο του δικτύου σε κατανομή πιθανότητας στις κλάσεις.



Σχήμα 3.1 Τυπική Αρχιτεκτονική CNN

Παρακάτω θα δούμε πιο αναλυτικά το κάθε επίπεδο των συνελικτικών νευρωνικών δικτύων.

3.3 Συνελικτικό επίπεδο

Τα συνελικτικά επίπεδα είναι τα πιο σημαντικά επίπεδα των συνελικτικών δικτύων και εκτελούν τη λειτουργία της συνέλιξης για να εξάγουν χαρακτηριστικά από τα δεδομένα εισόδου.

Ο ρόλος των συνελικτικών επιπέδων σε ένα cnn είναι να αναγνωρίζουν και να εξάγουν τοπικά μοτίβα και χαρακτηριστικά από τα δεδομένα εισόδου. Τα χαρακτηριστικά μπορεί να είναι απλά όπως άκρες και υφές αλλά και πιο σύνθετα όπως μέρη αντικειμένων ή και ολόκληρα αντικείμενα. Μαθαίνοντας να αναγνωρίζουν αυτά τα ιεραρχικά χαρακτηριστικά, τα cnn μπορούν να εκτελούν αποτελεσματικά εργασίες ταξινόμησης εικόνων ή άλλων εργασιών μηχανικής όρασης.

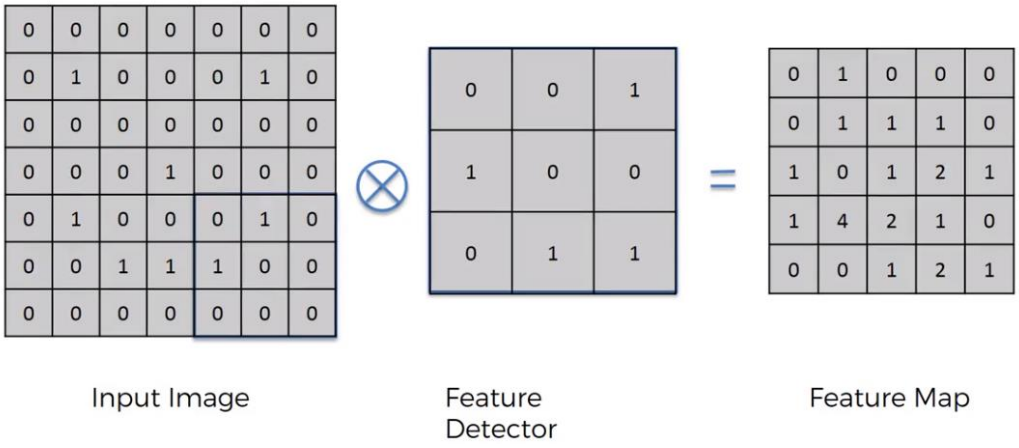
3.3.1 Λειτουργία συνέλιξης

Η λειτουργία της συνέλιξης είναι θεμελιώδης λειτουργία στα συνελικτικά επίπεδα. Περιλαμβάνει την ολίσθηση ενός ή περισσότερων φίλτρων πάνω στα δεδομένα εισόδου για την εξαγωγή χαρακτηριστικών.

Πιο αναλυτικά:

1. Ένα φίλτρο είναι ένας μικρός πίνακας με διαστάσεις μικρότερες από τα δεδομένα εισόδου οι τιμές του αντιπροσωπεύουν τα βάρη του. Τα φίλτρα έχουν σχεδιαστεί να ανιχνεύουν συγκεκριμένα χαρακτηριστικά ή μοτίβα στα δεδομένα εισόδου.

2. Το φίλτρο περνάει πάνω στα δεδομένα εισόδου, πατώντας πάνω σε μια μικρή περιοχή της εισόδου.
3. Πολλαπλασιάζεται το κάθε στοιχείο του φίλτρου με το αντίστοιχο στοιχείο της περιοχής εισόδου.
4. Γίνεται άθροισμα του γινομένου για να βγει μια μοναδική τιμή, που αντιπροσωπεύει την απόκριση του φίλτρου στη συγκεκριμένη θέση στην είσοδο.
5. Το φίλτρο, στη συνέχεια, μετακινείται με συγκεκριμένο βήμα, που ονομάζεται stride, και επαναλαμβάνεται η διαδικασία μέχρι να καλυφθεί όλη η είσοδος.
6. Ο πίνακας που παράγεται σαν έξοδος λέγεται χάρτης χαρακτηριστικών (feature map) και αντιπροσωπεύει τη χωρική διάταξη του χαρακτηριστικού που εντοπίστηκε στην είσοδο.



Σχήμα 3.2 Χάρτης Χαρακτηριστικών (Feature Map)

Μαθηματικά η λειτουργία της συνέλιξης μπορεί να περιγραφεί από τον τύπο:

$$(I * F)(i,j) = \sum_m \sum_n I(i+m, j+n) F(m,n) \qquad (3.1)$$

Όπου, I είναι ο πίνακας εισόδου, F ο πίνακας του φίλτρου, το (i,j) αντιπροσωπεύει τη θέση στο χάρτη χαρακτηριστικών εξόδου, το (m,n) αντιπροσωπεύει τις διαστάσεις του φίλτρου και το Σ υποδηλώνει τη λειτουργία της άθροισης. Η άθροιση γίνεται εντός των ορίων των δεδομένων εισόδου και των διαστάσεων φίλτρου. Από τον τύπο καταλαβαίνουμε ότι η λειτουργία της συνέλιξης είναι ο πολλαπλασιασμός των στοιχείων του φίλτρου και των δεδομένων εισόδου που ακολουθείται από το άθροισμα για την παραγωγή μιας μοναδικής τιμής εξόδου.

3.3.2 Padding

Το padding είναι μια τεχνική που βοηθάει στην αντιμετώπιση προκλήσεων που προκύπτουν κατά την εφαρμογή φίλτρων στα δεδομένα εισόδου. Είναι μια διαδικασία που προσθέτει επιπλέον σειρές ή και στήλες στα δεδομένα εισόδου, συνήθως με τιμή μηδέν. Αυτό γίνεται πριν τη λειτουργία της συνέλιξης και έχει στόχο τον έλεγχο των χωρικών διαστάσεων των feature maps.

Το padding έχει τρεις βασικούς σκοπούς:

- Τη διατήρηση των χωρικών διαστάσεων. Χωρίς padding, η λειτουργία της συνέλιξης δημιουργεί χάρτες χαρακτηριστικών με μικρότερες χωρικές διαστάσεις από τα δεδομένα εισόδου. Αυτό γίνεται γιατί τα φίλτρα δεν μπορούν να εφαρμοστούν στις άκρες της εισόδου χωρίς να βγουν εκτός ορίων. Το padding επιτρέπει στα φίλτρα να εφαρμόζονται στις άκρες διατηρώντας τις χωρικές διαστάσεις και εξασφαλίζει ότι δεν χάνονται σημαντικές πληροφορίες στις άκρες.
- Τον έλεγχο του οπτικού πεδίου. Προσθέτοντας επιπλέον σειρές και στήλες στα δεδομένα εισόδου, τα φίλτρα μπορούν να καλύψουν μια μεγαλύτερη περιοχή της εισόδου και να βρουν χαρακτηριστικά που θα χάνονταν.
- Την αποτροπή της γρήγορης συρρίκνωσης. Στα βαθιά cnn υπάρχουν πολλά συνελικτικά επίπεδα, χωρίς το padding οι χωρικές διαστάσεις των χαρτών χαρακτηριστικών θα συρρικνωθούν γρήγορα μέσω διαδοχικών επιπέδων, που μπορεί να οδηγήσει σε απώλεια πληροφορίας. Το padding βοηθάει στην αποφυγή αυτού του προβλήματος και διατηρεί τη χωρική δομή στους χάρτες χαρακτηριστικών.

Επίσης, υπάρχουν διάφοροι τύποι padding:

- Zero-padding: Αυτός είναι ο πιο διαδεδομένος τύπος padding όπου οι επιπλέον σειρές και στήλες γεμίζουν με μηδενικά και έχει το πλεονέκτημα ότι δεν εισάγει νέες πληροφορίες στην είσοδο.
- Same-padding: Αυτός ο τύπος διασφαλίζει ότι ο χάρτης χαρακτηριστικών έχει τις ίδιες χωρικές διαστάσεις με τα δεδομένα εισόδου. Η ποσότητα του padding εξαρτάται από το μέγεθος του φίλτρου και το βήμα (stride).
- Reflect-padding: Σε αυτόν τον τύπο padding, οι επιπλέον σειρές και στήλες συμπληρώνονται αντανακλώντας τις τιμές περιγράμματος της εισόδου. Αυτή η προσέγγιση βοηθάει στη διατήρηση της συνέχειας των χαρακτηριστικών των δεδομένων εισόδου.
- Constant-padding: Αυτός ο τύπος padding προσθέτει επιπλέον σειρές και στήλες με μια συγκεκριμένη τιμή. Είναι από τους λιγότερο διαδεδομένους τύπους padding.

3.3.3 Stride

Το stride είναι μια σημαντική παράμετρος στη λειτουργία της συνέλιξης στα cnn. Καθορίζει το μέγεθος του βήματος της εφαρμογής του φίλτρου σε διαφορετικές χωρικές θέσεις στα δεδομένα εισόδου. Το stride αναφέρεται στον αριθμό μονάδων με τις οποίες το φίλτρο κινείται στα δεδομένα εισόδου κατά τη συνέλιξη. Συνήθως αναπαρίσταται ως (s_x, s_y) όπου s_x είναι ο οριζόντιος βηματισμός και s_y ο κάθετος βηματισμός.

Για παράδειγμα, stride (1,1) δηλώνει ότι το φίλτρο κινείται μια μονάδα κάθε φορά οριζόντια και κάθετα, (2,2) δύο μονάδες προς τις δύο κατευθύνσεις κτλπ.

Το stride έχει τρεις βασικούς σκοπούς:

- Το βήμα επηρεάζει το μέγεθος των χαρτών χαρακτηριστικών. Όσο μεγαλύτερο το βήμα τόσο μικρότεροι θα είναι οι χάρτες χαρακτηριστικών, μιας και το φίλτρο θα εφαρμόζεται σε λιγότερες θέσεις στα δεδομένα εισόδου. Αυτό είναι χρήσιμο όταν θέλουμε να μειώσουμε την υπολογιστική πολυπλοκότητα του δικτύου και της ποσότητας μνήμης που χρειάζεται για την αποθήκευση των χαρτών χαρακτηριστικών.
- Ένας μεγαλύτερος βηματισμός αυξάνει το πεδίο λήψης της λειτουργίας της συνέλιξης, που είναι η περιοχή των δεδομένων εισόδου που επηρεάζει την έξοδο σε μια συγκεκριμένη θέση. Αυτό επιτρέπει στο δίκτυο να πιάνει περισσότερες πληροφορίες και μπορεί να βελτιώσει την ικανότητά του να αναγνωρίζει μοτίβα μεγαλύτερης κλίμακας.
- Σε κάποιες περιπτώσεις, όταν τα δεδομένα εισόδου μπορεί να περιέχουν περιττές πληροφορίες, ένα μεγαλύτερο βήμα μπορεί να βοηθήσει στη μείωση αυτού του πλεονασμού εφαρμόζοντας το φίλτρο σε λιγότερες επικαλυπτόμενες περιοχές στα δεδομένα εισόδου.

Η επιλογή του stride έχει και κάποια trade-offs, όπως:

- Όσο μεγαλύτερο το βήμα μειώνει το μέγεθος της εξόδου και την υπολογιστική πολυπλοκότητα του δικτύου, αλλά μπορεί να οδηγήσει σε απώλεια χωρικής πληροφορίας στα δεδομένα εισόδου.
- Ένας μεγαλύτερος βηματισμός αυξάνει το πεδίο λήψης της λειτουργίας της συνέλιξης, αλλά μειώνει τη χωρική ανάλυση των χαρτών χαρακτηριστικών, μιας και το φίλτρο θα εφαρμοστεί σε λιγότερες θέσεις στα δεδομένα εισόδου.
- Ένα μεγαλύτερο βήμα μπορεί να βοηθήσει στην αποφυγή της υπερπροσαρμογής μειώνοντας τον χωρικό πλεονασμό στα δεδομένα εισόδου, αλλά μπορεί να κάνει το δίκτυο λιγότερο ευαίσθητο σε λεπτομέρειες και μοτίβα μικρής κλίμακας.

Το stride λοιπόν είναι μια σημαντική παράμετρος στα cnn και η επιλογή του εξαρτάται από τις συγκεκριμένες απαιτήσεις του δικτύου και τα δεδομένα εισόδου.

3.4 Επίπεδο συγκέντρωσης

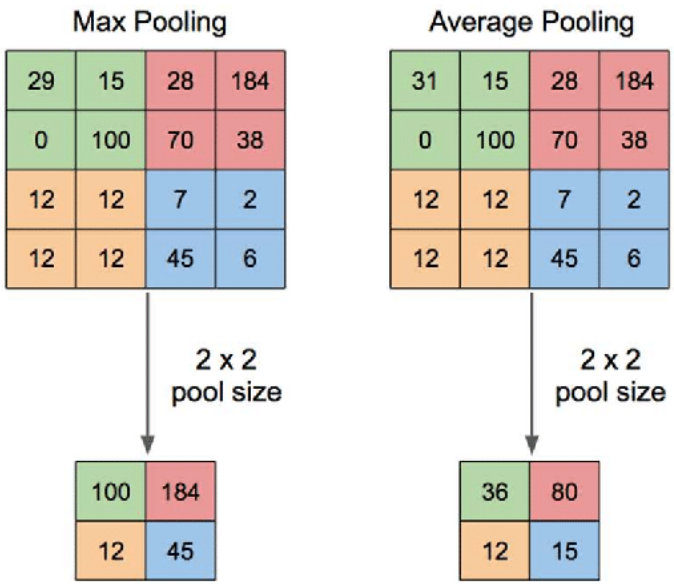
Το επίπεδο συγκέντρωσης είναι υπεύθυνο για τη μείωση των διαστάσεων του χάρτη χαρακτηριστικών που παράγονται από το συνελικτικό επίπεδο. Αυτό το επιτυγχάνουν συγκεντρώνοντας πληροφορίες από τους χάρτες χαρακτηριστικών εισόδου, συμβάλλοντας στη διατήρηση των πιο σημαντικών χαρακτηριστικών ενώ απορρίπτει λιγότερο σημαντικές λεπτομέρειες. Στην ουσία το επίπεδο συγκέντρωσης εκτελεί μια διαδικασία μείωσης δειγματοληψίας στους χάρτες χαρακτηριστικών, μειώνοντας τις χωρικές τους διαστάσεις ενώ παράλληλα διατηρούν τις πιο σημαντικές πληροφορίες. Αυτό γίνεται με την εφαρμογή μιας συνάρτησης συγκέντρωσης όπως οι μέγιστο ή μέσο όρο, σε μη επικαλυπτόμενες περιοχές των χαρτών χαρακτηριστικών.

Οι βασικοί σκοποί του επιπέδου χαρακτηριστικών είναι:

1. Η μείωση των χωρικών διαστάσεων. Τα επίπεδα συγκέντρωσης μειώνουν τις χωρικές διαστάσεις και με αυτό συμβάλει στη μείωση της υπολογιστικής πολυπλοκότητας του μοντέλου και της ποσότητας μνήμης που απαιτείται. Αυτή η μείωση της πολυπλοκότητας επιτρέπει τη χρήση βαθύτερων και πιο σύνθετων αρχιτεκτονικών που θα βελτιώσουν την απόδοση του μοντέλου.
2. Εισαγωγή αναλλοίωτης μετατόπισης: Τα επίπεδα συγκέντρωσης εισάγουν ένα επίπεδο μεταβλητότητας μετατόπισης, επιτρέποντας στο δίκτυο να αναγνωρίζει χαρακτηριστικά ανεξάρτητα από την θέση τους στα δεδομένα εισόδου. Αυτή η ιδιότητα είναι χρήσιμη για εργασίες ταξινόμησης όπου η θέση των χαρακτηριστικών εντός της εικόνας μπορεί να είναι διαφορετική.
3. Αποφυγή υπερπροσαρμογής: Με την απόρριψη λιγότερο σημαντικών χαρακτηριστικών και τη μείωση των χωρικών διαστάσεων, τα επίπεδα συγκέντρωσης βοηθούν στην αποφυγή της υπερπροσαρμογής βελτιώνοντας την γενίκευσή του.

Δύο βασικά είδη συναρτήσεων συγκέντρωσης είναι η συγκέντρωση μεγίστου και η συγκέντρωση μέσου όρου:

- Στη συγκέντρωση μεγίστου το φίλτρο σαρώνει την εικόνα με βήμα S και βρίσκει το μέγιστο από τα στοιχεία του πίνακα εισόδου που καλύπτει το φίλτρο και επιλέγεται ως έξοδος. Η συγκέντρωση μεγίστου είναι η πιο διαδεδομένη συνάρτηση συγκέντρωσης.
- Στη συγκέντρωση μέσου όρου το φίλτρο σαρώνει την εικόνα με βήμα S και βρίσκει τον μέσο όρο από τα στοιχεία του πίνακα που καλύπτει το φίλτρο και επιλέγεται ως έξοδος. Αυτή η συνάρτηση παρέχει μια πιο ομαλή αναπαράσταση των χαρτών χαρακτηριστικών.



Σχήμα 3.3 Max Pooling - Average Pooling

Οι κύριες παράμετροι που καθορίζουν τη συμπεριφορά του επίπεδου συγκέντρωσης είναι:

- Η συνάρτηση συγκέντρωσης, μέγιστο ή μέσος όρος, καθορίζει τον τρόπο με τον οποίο συνδυάζονται οι τιμές εισόδου σε μια περιοχή συγκέντρωσης.
- Το μέγεθος συγκέντρωσης καθορίζει τις χωρικές διαστάσεις των περιοχών συγκέντρωσης πχ 2x2 όπως στην εικόνα 3.3. Τα μεγαλύτερα μεγέθη συγκέντρωσης έχουν ως αποτέλεσμα πιο επιθετική μείωση δειγματοληψίας και μεγαλύτερη μείωση των χωρικών διαστάσεων.
- Stride: Όπως και στη λειτουργία της συνέλιξης, το βήμα (stride) στα επίπεδα συγκέντρωσης καθορίζει το μέγεθος βήματος ή την απόσταση μεταξύ της εφαρμογής της συνάρτησης συγκέντρωσης σε διαφορετικές χωρικές θέσεις στα δεδομένα εισόδου.

3.5 Πλήρως διασυνδεδεμένο επίπεδο

Συνήθως, το τελευταίο επίπεδο ενός συνελικτικού δικτύου πριν το επίπεδο εξόδου είναι ένα πλήρως διασυνδεδεμένο επίπεδο. Στο πλήρως συνδεδεμένο επίπεδο, γνωστό και ως πυκνό (dense), ο κάθε νευρώνας συνδέεται με κάθε νευρώνα στο προηγούμενο επίπεδο, έχουν δηλαδή πλήρεις συνδέσεις με όλες τις ενεργοποιήσεις στο προηγούμενο επίπεδο. Αυτό επιτρέπει στο επίπεδο να μαθαίνει σύνθετα μοτίβα και σχέσεις μεταξύ των χαρακτηριστικών.

Μαθηματικά ορίζεται ως εξής:

$$y = Wx + b \tag{3.2}$$

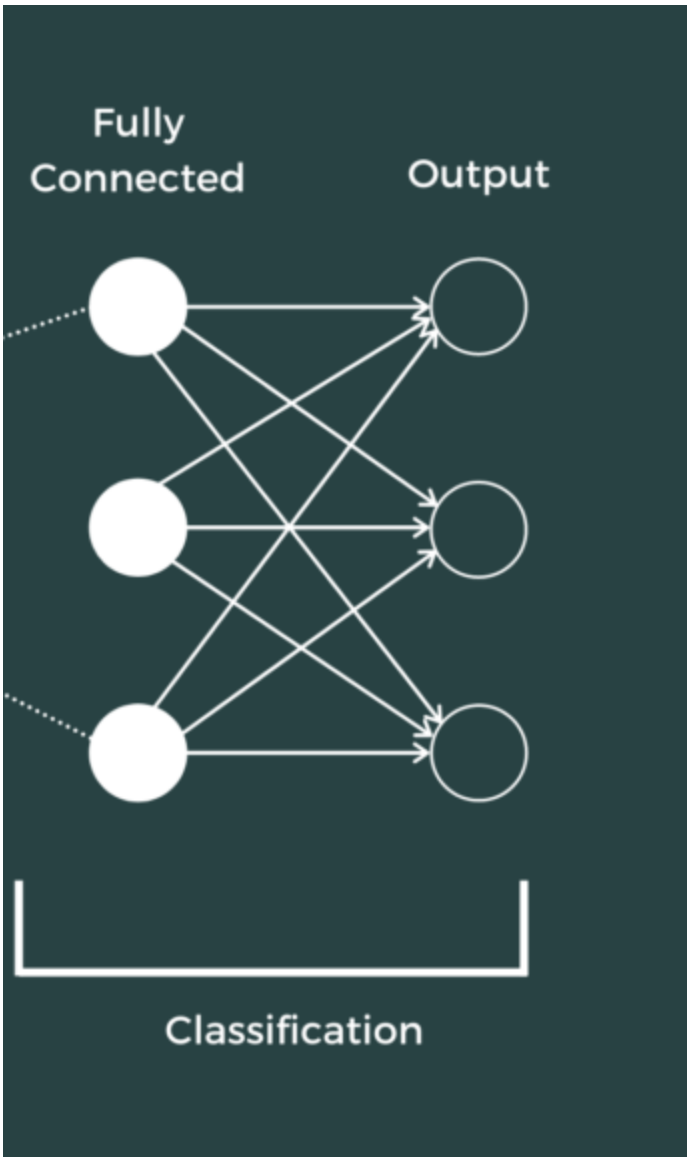
Όπου,

- Το y είναι το διάνυσμα εξόδου,
- το W είναι ο πίνακας με τα βάρη, που δείχνει τις συνδέσεις μεταξύ των νευρώνων του πλήρως συνδεδεμένου επίπεδου με το προηγούμενο επίπεδο,
- το x είναι το διάνυσμα εισόδου του επιπέδου
- και b είναι το διάνυσμα bias, που δείχνει έναν επιπλέον όρο που επιτρέπει στο δίκτυο να μαθαίνει πιο άνετα πολύπλοκες σχέσεις.

Μετά από αυτόν τον γραμμικό μετασχηματισμό, μια συνάρτηση ενεργοποίησης εφαρμόζεται στοιχείο προς στοιχείο στο διάνυσμα εξόδου. Η συνάρτηση ενεργοποίησης προσθέτει μια μη γραμμικότητα στο δίκτυο επιτρέποντάς του να μάθει μη γραμμικές σχέσεις μεταξύ των χαρακτηριστικών εισόδου και των κλάσεων εξόδου.

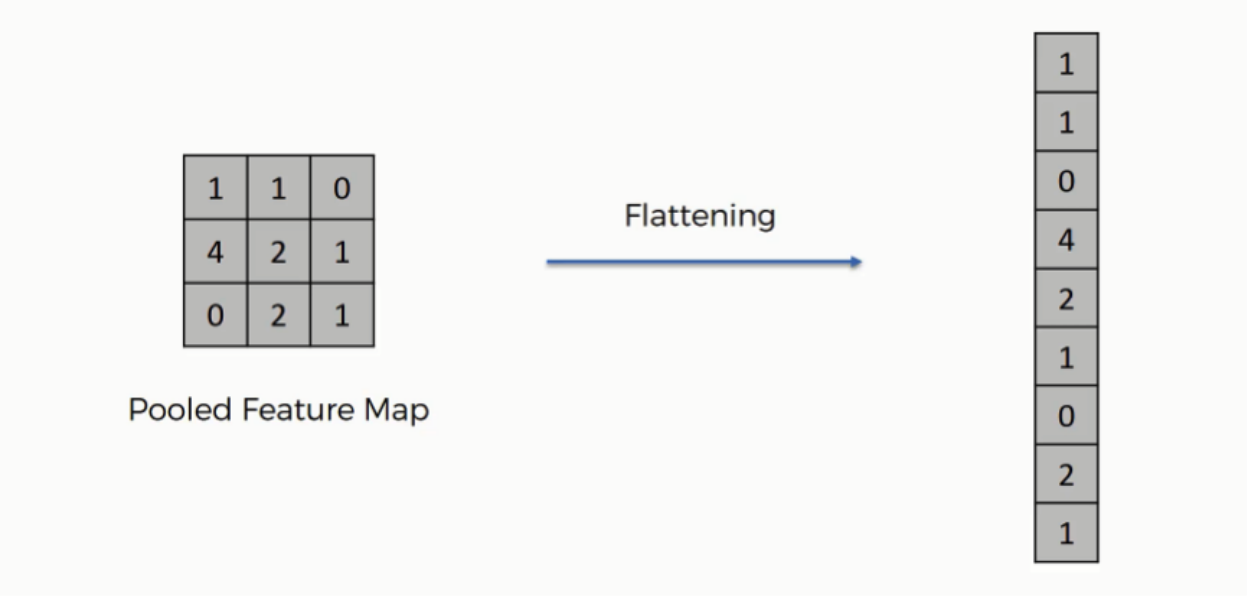
Σκοπός αυτού του επιπέδου είναι:

- Η ενσωμάτωση χαρακτηριστικών υψηλού επιπέδου. Αφού τα προηγούμενα επίπεδα συνέλιξης και συγκέντρωσης εξάγουν τα χαρακτηριστικά υψηλού επιπέδου, χρησιμοποιείται ένα πλήρως συνδεδεμένο επίπεδο για την ενοποίηση αυτών των χαρακτηριστικών και την εκμάθηση των μεταξύ τους σχέσεων. Αυτό βοηθάει το δίκτυο να κάνει πιο ακριβείς προβλέψεις και να εκτελεί πολύπλοκες εργασίες ταξινόμησης.
- Ένας άλλος σκοπός αυτού του επιπέδου είναι να εκτελεί την ταξινόμηση. Αυτό το επίπεδο συνήθως έχει αριθμό νευρώνων ίσο με τις κλάσεις την εργασία ταξινόμησης και η συνάρτηση ενεργοποίησης που χρησιμοποιείται συνήθως είναι η Softmax για να μετατρέψει την έξοδο σε πιθανότητες κλάσης, για να μπορούν να χρησιμοποιηθούν για τις προβλέψεις.
- Επίσης ενεργοποιεί την εκπαίδευση από άκρο σε άκρο. Τα πλήρως συνδεδεμένα επίπεδα μαζί με τα συνελικτικά και τα επίπεδα συγκέντρωσης, σχηματίζουν μια πλήρη αρχιτεκτονική από άκρο σε άκρο που μπορεί να εκπαιδευτεί από κοινού χρησιμοποιώντας backpropagation και gradient descent. Αυτό επιτρέπει στο δίκτυο να μάθει τις βέλτιστες λειτουργίες εξαγωγής και ταξινόμησης χαρακτηριστικών ταυτόχρονα.



Σχήμα 3.4 Πλήρως Συνδεδεμένο Επίπεδο

Επίσης, πριν πάρει σαν είσοδο την έξοδο του προηγούμενου επιπέδου, γίνεται flattening της εξόδου σε μονοδιάστατο διάνυσμα. Αυτό είναι απαραίτητο γιατί τα πλήρως συνδεδεμένα επίπεδα περιμένουν στην είσοδό τους ένα διάνυσμα, ενώ τα προηγούμενα επίπεδα, συνέλιξης και συγκέντρωσης, παράγουν στην έξοδό τους πολυδιάστατους χάρτες χαρακτηριστικών.



Σχήμα 3.5 Flattening σε 1D διάνυσμα

Τέλος, για να αποφύγει την υπερπροσαρμογή και να βελτιώσει την απόδοση της γενίκευσης του δικτύου, μπορεί να εφαρμοστεί και κάποια τεχνική regularization στο πλήρως συνδεδεμένο επίπεδο, όπως L1, L2 ή dropout.

3.6 Επίπεδο εξόδου

Το επίπεδο εξόδου σε ένα συνελικτικό δίκτυο είναι το τελικό επίπεδο που είναι υπεύθυνο για την παραγωγή προβλέψεων ή των πιθανοτήτων κλάσης με βάση τα χαρακτηριστικά που εξάγονται και επεξεργάζονται στα προηγούμενα επίπεδα, και παίζει σημαντικό ρόλο στη συνολική απόδοση και λειτουργικότητα ενός cnn.

Ο κύριος ρόλος του επιπέδου εξόδου είναι να μετασχηματίσει τα χαρακτηριστικά που επεξεργάστηκαν τα προηγούμενα επίπεδα σε μια κατάλληλη μορφή για την εκάστοτε εργασία, ταξινόμησης, αναγνώρισης αντικειμένων κλπ. Το επίπεδο εξόδου λοιπόν είναι υπεύθυνο για την παραγωγή προβλέψεων ή πιθανοτήτων που μπορούν να ερμηνευτούν και να χρησιμοποιηθούν από εξωτερικά συστήματα και εφαρμογές.

Ο αριθμός των νευρώνων στο επίπεδο εξόδου εξαρτάται από τη συγκεκριμένη εργασία που το δίκτυο σχεδιάστηκε να εκτελέσει. Για ένα πρόβλημα ταξινόμησης εικόνων πολλαπλών κλάσεων το επίπεδο εξόδου έχει τυπικά τόσους νευρώνες όσες είναι και οι κλάσεις. Αν υπάρχουν μόνο 2 κλάσεις, μπορεί να έχει έναν νευρώνα.

Επίσης, η συνάρτηση ενεργοποίησης που χρησιμοποιείται σε αυτό το επίπεδο εξαρτάται κι αυτή από την εργασία που έχει να εκτελέσει το δίκτυο. Για εργασία ταξινόμησης πολλαπλών κλάσεων χρησιμοποιείται συνήθως η συνάρτηση Softmax, ενώ για δυαδική ταξινόμηση μπορεί να χρησιμοποιηθεί και μια σιγμοειδή συνάρτηση ενεργοποίησης.

Το επίπεδο εξόδου συνδέεται απευθείας με τη συνάρτηση σφάλματος, που ποσοτικοποιεί τη διαφορά μεταξύ των προβλεπόμενων εξόδων και των πραγματικών τιμών. Η επιλογή της συνάρτησης σφάλματος κι αυτή με τη σειρά της εξαρτάται από την εργασία. Συνήθως για εργασίες ταξινόμησης χρησιμοποιείται η συνάρτηση cross-entropy.

Τέλος, η έξοδος του επιπέδου εξόδου μπορεί να ερμηνευτεί διαφορετικά ανάλογα με την εργασία που εκτελεί το δίκτυο. Για εργασία ταξινόμησης, η έξοδος μπορεί να ερμηνευτεί ως πιθανότητες κλάσης και η κλάση με την υψηλότερη πιθανότητα μπορεί να επιλεγεί ως η τελική πρόβλεψη του δικτύου.

Συνοπτικά, το επίπεδο εξόδου σε ένα cnn είναι το τελικό επίπεδο που ευθύνεται για την παραγωγή της τελική εξόδου, πρόβλεψης, με βάση όσα έχει μάθει από τα προηγούμενα επίπεδα. Ο σχεδιασμός του, αριθμός νευρώνων και συνάρτηση ενεργοποίησης, εξαρτάται από τη συγκεκριμένη εργασία που έχει να εκτελέσει το δίκτυο, και παίζει κρίσιμο ρόλο στη συνολική απόδοση του μοντέλου αφού συνδέεται άμεσα με την συνάρτηση σφάλματος που χρησιμοποιείται για την εκπαίδευση και βελτιστοποίηση.

3.7 Εκπαίδευση CNN

Η εκπαίδευση ενός συνελικτικού δικτύου μοιάζει αρκετά με την εκπαίδευση άλλων τύπων νευρωνικών δικτύων, αλλά έχει και μερικές μοναδικές πτυχές λόγω των επιπέδων συνέλιξης και συγκέντρωσης στην αρχιτεκτονική του. Η κύρια διαφορά δηλαδή, είναι στην αρχιτεκτονική του δικτύου και στα επίπεδα για να επεξεργάζεται εικόνες. Παρακάτω θα δούμε πιο αναλυτικά τα βήματα για την εκπαίδευση ενός cnn συγκεκριμένα για εργασίες ταξινόμησης εικόνων.

3.7.1 Προετοιμασία δεδομένων

Η προετοιμασία δεδομένων είναι ένα αρκετά σημαντικό κομμάτι για την εκπαίδευση ενός νευρωνικού δικτύου, αφού μπορούν να επηρεάσουν αρκετά την απόδοση και τη γενίκευση του μοντέλου.

- Το πρώτο βήμα είναι η συλλογή δεδομένων που σχετίζονται με το πρόβλημα που θέλουμε να λύσουμε. Το dataset αυτό, που θα χρησιμοποιηθεί στην εκπαίδευση του δικτύου θα

πρέπει να είναι όσο πιο μεγάλο γίνεται και να έχει μεγάλη ποικιλία ώστε να εκπαιδευτεί το δίκτυο σε όσο το δυνατόν διαφορετικές παραλλαγές.

- Μετά τη συλλογή δεδομένων, το επόμενο είναι καθαριστεί το dataset για να αφαιρεθούν διπλότυπες, κακής ποιότητας και άσχετες εικόνες ανάλογα με το τι θέλουμε το δίκτυο να ταξινομεί. Αυτό βοηθάει στο να εξασφαλίσει ένα καλό, υψηλής ποιότητας dataset για την εκπαίδευση του δικτύου.
- Κανονικοποίηση των δεδομένων. Αυτό είναι απαραίτητο για να είναι όλες οι τιμές εισόδου στην ίδια κλίμακα, για να είναι πιο σταθερή η εκπαίδευση. Για δεδομένα εικόνας, γίνεται κανονικοποίηση στις τιμές των pixels σε εύρος [0, 1].
- Στη συνέχεια, επειδή μπορεί οι εικόνες να έχουν διαφορετικό μέγεθος μεταξύ τους γίνεται αλλαγή στο μέγεθος όλων των εικόνων σε σταθερό μέγεθος για να είναι συμβατές με το επίπεδο εισόδου του cnn (πχ 224x224).
- Πολλές φορές είναι αρκετά δύσκολο να μαζέψουμε αρκετά δεδομένα και να φτιάξουμε ένα καλό dataset. Γι' αυτό το λόγο μπορούμε να χρησιμοποιήσουμε μια τεχνική που λέγεται data augmentation. Το data augmentation είναι μια τεχνική που χρησιμοποιείται για φτιάξει τεχνητά δεδομένα από τα υπάρχων. Παίρνει διάφορες εικόνες από το dataset και εφαρμόζει κάποιους μετασχηματισμούς, όπως περιστροφή, αλλαγή φωτεινότητας, αλλαγή αντίθεσης, για να παρέχει στο dataset μεγαλύτερη ποικιλομορφία. Αυτό βοηθάει το δίκτυο να μάθει πιο ισχυρά χαρακτηριστικά και να βελτιώσει τη γενίκευσή του σε δεδομένα που δεν έχει εκπαιδευτεί.
- Μια άλλη τεχνική που χρησιμοποιείται σχεδόν πάντα είναι ο διαχωρισμός του dataset. Το dataset χωρίζεται συνήθως σε τρία υποσύνολα, training, validation και test set, και συνήθως χωρίζονται 80%, 10%, 10% αντίστοιχα. Το training set όπως λέει και το όνομά του χρησιμοποιείται για την εκπαίδευση του μοντέλου, το validation set χρησιμοποιείται για το fine tuning των υπερπαραμέτρων και την παρακολούθηση της απόδοσής του κατά τη διάρκεια της εκπαίδευσης, και το test set χρησιμοποιείται για την αξιολόγηση της τελικής απόδοσης του μοντέλου.
- Τέλος, κατά τη διάρκεια της εκπαίδευσης τα δεδομένα τροφοδοτούνται στο μοντέλο ανά παρτίδες, batches, και όχι μία εικόνα κάθε φορά ούτε όλο το training set μαζί. Αυτό λέγεται data batching και βοηθάει στην χρήση των διαθέσιμων υπολογιστικών πόρων και παρέχει καλύτερη προσέγγιση της κλίσης, gradient, κατά τη βελτιστοποίηση. Η επιλογή του μεγέθους κάθε παρτίδας είναι μια υπερπαράμετρος και επηρεάζει την ταχύτητα εκπαίδευσης και τη σύγκλιση του μοντέλου.

Γενικά, τα δεδομένα πολλές φορές είναι το πιο σημαντικό κομμάτι για ένα μοντέλο. Υπάρχει μια προσέγγιση που λέγεται data centric ai, που εστιάζει στη συλλογή, διαχείριση και ανάλυση μεγάλων datasets. Σε αυτή την προσέγγιση, τα δεδομένα είναι το πιο σημαντικό κομμάτι και όχι η αρχιτεκτονική του μοντέλου.

3.7.2 Μπροστινό πέρασμα (Forward propagation)

Το επίπεδο εισόδου είναι το αρχικό σημείο για το μπροστινό πέρασμα και είναι ίδιο και με τα υπόλοιπα είδη νευρωνικών δικτύων με τη διαφορά ότι στα cnn το επίπεδο εισόδου παίρνει μια εικόνα ή παρτίδα με εικόνες στην είσοδό του.

Όσο περνάνε τα δεδομένα μέσα από το δίκτυο, κάθε επίπεδο εκτελεί μια συγκεκριμένη δουλειά στα δεδομένα εισόδου που παίρνει από το προηγούμενο επίπεδο και παράγει μια έξοδο η οποία χρησιμεύει ως είσοδος για το επόμενο επίπεδο. Αυτή η διαδικασία συνεχίζεται μέχρι τα δεδομένα να φτάσουν στο επίπεδο εξόδου. Η εργασία που εκτελείται σε κάθε επίπεδο εξαρτάται από τον τύπο και τον σκοπό του επίπεδου. Στα cnn τα δεδομένα συνήθως περνάνε από επίπεδα συνέλιξης, συγκέντρωσης και πλήρως συνδεδεμένα επίπεδα όπως είδαμε και παραπάνω και όταν φτάσουν στο επίπεδο εξόδου, παράγονται οι προβλέψεις ή πιθανότητες κλάσης για τα δεδομένα εισόδου και συγκρίνονται με τις πραγματικές τιμές χρησιμοποιώντας μια συνάρτηση σφάλματος ώστε να βρεθεί η διαφορά μεταξύ τους και να βρεθεί το σφάλμα.

3.7.3 Υπολογισμός σφάλματος και backpropagation

Μετά από το μπροστινό πέρασμα και την παραγωγή των προβλέψεων, αυτές οι προβλέψεις συγκρίνονται με τις πραγματικές τιμές και υπολογίζεται το σφάλμα. Μετράει δηλαδή τη διαφορά μεταξύ των πιθανοτήτων κλάσης που έβγαλε το δίκτυο με τις πραγματικές ετικέτες κλάσης, και συνήθως χρησιμοποιείται η συνάρτηση cross entropy.

$$cross\ entropy\ loss(y, y_{pred}) = -\sum y_i * \log\left(y_{pred_i}\right) \qquad (3.3)$$

- Μετά τον υπολογισμό του σφάλματος, ο αλγόριθμος backpropagation χρησιμοποιείται για τον υπολογισμό των κλίσεων της συνάρτησης σφάλματος σε σχέση με τα βάρη και το bias. Η οπισθοδιάδοση, όπως είδαμε και στο κεφάλαιο 2.3.1 είναι μια εφαρμογή του κανόνα της αλυσίδας για τον υπολογισμό των κλίσεων σε ένα νευρωνικό δίκτυο.
- Ξεκινώντας από το επίπεδο εξόδου και πηγαίνοντας προς τα πίσω, υπολογίζονται πρώτα οι κλίσεις της συνάρτησης σφάλματος σε σχέση με τις εξόδους, τα βάρη και το bias κάθε επιπέδου και επαναλαμβάνεται μέχρι να φτάσει στο επίπεδο εισόδου. Αυτή η διαδικασία περιλαμβάνει την εφαρμογή του κανόνα της αλυσίδας για τον υπολογισμό των κλίσεων σε σχέση με τις εισόδους του επιπέδου, που ακολουθείται από τον υπολογισμό των κλίσεων σε σχέση με τα βάρη και το bias του επιπέδου.
- Για τα συνελικτικά επίπεδα, η διαδικασία οπισθοδιάδοσης περιλαμβάνει τον υπολογισμό κλίσεων σε σχέση με τα βάρη του φίλτρου και τα ενημερώνει κατάλληλα, από τον υπολογισμό της συνέλιξης των χαρτών χαρακτηριστικών εισόδου με τις κλίσεις της συνάρτησης σφάλματος σε σχέση με τις εξόδους του επιπέδου.

- Υπολογίζονται επίσης οι κλίσεις σε σχέση με τις συναρτήσεις ενεργοποίησης με τον υπολογισμό της παραγώγου της συνάρτησης ενεργοποίησης σε σχέση με την είσοδό της και τον πολλαπλασιασμό της με την κλίση της συνάρτησης σφάλματος σε σχέση με την έξοδο του επιπέδου.
- Οι κλίσεις που υπολογίζονται κατά τη διαδικασία του backpropagation συσσωρεύονται σε ολόκληρη την παρτίδα δεδομένων εισόδου, αθροίζοντας τις κλίσεις για κάθε σημείο δεδομένων στη παρτίδα. Αυτές οι συσσωρευμένες κλίσεις χρησιμοποιούνται για την ενημέρωση των παραμέτρων του μοντέλου κατά τη διάρκεια της βελτιστοποίησης.

3.7.4 Βελτιστοποίηση και regularization στα CNN

Η διαδικασία της βελτιστοποίησης γίνεται για την ενημέρωση των παραμέτρων, βάρη και bias. Συνήθως χρησιμοποιούνται οι αλγόριθμοι Stochastic Gradient Descent (SGD) και Adaptive Moment Estimation (Adam) που είδαμε και στο κεφάλαιο 2.3.4.

- Με τον αλγόριθμο SGD οι παράμετροι ενημερώνονται χρησιμοποιώντας ένα μικρό υποσύνολο από τα δεδομένα εκπαίδευσης κάνοντας τη διαδικασία ενημέρωσης των παραμέτρων πιο αποδοτική.
- Με τον αλγόριθμο Adam προσαρμόζει τον ρυθμό μάθησης για κάθε παράμετρο με βάση τη πρώτη και τη δεύτερη στιγμή των κλίσεων, βοηθώντας στην ταχύτερη σύγκλιση και λιγότερη ευαισθησία στις ρυθμίσεις των υπερπαραμέτρων.

Οι τεχνικές regularization χρησιμοποιούνται, όπως είδαμε και στο κεφάλαιο 2.3.5 για να αποφύγουμε την υπερπροσαρμογή (overfitting) προσθέτοντας περιορισμού στο μοντέλο ή τροποποιώντας τη διαδικασία εκμάθησης. Συνήθως χρησιμοποιούνται οι τεχνικές L1 (Lasso), L2 (Ridge), Dropout και Batch Normalization.

- Οι τεχνικές L1 και L2 προσθέτουν έναν όρο ποινής στη συνάρτηση σφάλματος με βάση τα βάρη του μοντέλου. Η L1 προσθέτει το άθροισμα των απόλυτων τιμών των βαρών, ενώ η L2 το άθροισμα των τιμών τετραγώνου των βαρών. Αυτοί οι όροι ποινής κάνουν το μοντέλο να μάθει απλούστερες αναπαραστάσεις και αποτρέπουν την υπερπροσαρμογή.
- Το dropout απορρίπτει τυχαία ένα κλάσμα νευρώνων κατά τη διάρκεια της εκπαίδευσης, για να εμποδίσει το μοντέλο να βασίζεται πολύ σε οποιονδήποτε μεμονωμένο νευρώνα και συνήθως εφαρμόζεται στα πλήρως συνδεδεμένα επίπεδα αλλά μπορεί να εφαρμοστεί και στα συνελικτικά.
- Το batch normalization κανονικοποιεί την είσοδο σε ένα επίπεδο κατά τη διάρκεια της εκπαίδευσης, μειώνοντας την εσωτερική μετατόπιση συμμεταβλητής για να σταθεροποιήσει της μαθησιακή διαδικασία και επιτρέπει τη χρήση μεγαλύτερου ρυθμού μάθησης. Παρόλο που δεν είναι ακριβώς τεχνική regularization, μπορεί να βοηθήσει στη μείωση της υπερπροσαρμογής.

3.7.5 Παρακολούθηση και επικύρωση του μοντέλου

Τέλος έχουμε την παρακολούθηση και την επικύρωση του μοντέλου. Η παρακολούθηση (monitoring) περιλαμβάνει την παρακολούθηση διάφορων μετρήσεων απόδοσης του μοντέλου κατά τη διάρκεια της εκπαίδευσης ενώ η επικύρωση (validation) περιλαμβάνει την αξιολόγηση της απόδοσης του μοντέλου στο validation set που δεν χρησιμοποιείται για την εκπαίδευση. Παρακάτω θα δούμε πιο αναλυτικά σχετικά με την παρακολούθηση και την επικύρωση.

- Παρακολούθηση (Monitoring): Κατά την παρακολούθηση της εκπαίδευσης παρακολουθούνται διάφορες μετρήσεις για την παρατήρηση των τάσεων τους με την πάροδο του χρόνου. Οι πιο βασικές μετρήσεις είναι:
    - Training loss: Το training loss είναι η τιμή της συνάρτησης σφάλματος που υπολογίζεται στο σύνολο των δεδομένων εκπαίδευσης και μας δείχνει πόσο καλά ταιριάζει το μοντέλο στα δεδομένα εκπαίδευσης. Όταν σε κάθε εποχή το training loss πέφτει σημαίνει ότι το μοντέλο μαθαίνει αποτελεσματικά.
    - Validation loss: Το validation loss είναι η τιμή της συνάρτησης σφάλματος που υπολογίζεται στο σύνολο των δεδομένων επικύρωσης (validation set) και μας δείχνει πόσο καλή είναι η γενίκευση του μοντέλου σε νέα δεδομένα που δεν έχει ξαναδεί. Όταν σε κάθε εποχή το validation loss πέφτει δηλώνει ότι μοντέλο δεν ταιριάζει υπερβολικά στα δεδομένα εκπαίδευσης και υπάρχει καλή γενίκευση.
    - Training accuracy: Το training accuracy είναι το ποσοστό των σωστών προβλέψεων που γίνονται από το μοντέλο στα δεδομένα εκπαίδευσης και μας δείχνει την απόδοση του μοντέλου και βοηθάει στον εντοπισμό πιθανών προβλημάτων στη διαδικασία εκπαίδευσης, όπως υποπροσαρμογή (underfitting) ή υπερπροσαρμογή (overfitting).
    - Validation accuracy: Το validation accuracy είναι το ποσοστό των σωστών προβλέψεων που γίνονται από το μοντέλο στο validation set. Παρέχει ένα μέτρο της ικανότητας του μοντέλου να γενικεύει σε καινούργια δεδομένα και χρησιμοποιείται για την επιλογή του καλύτερου μοντέλου κατά τη διαδικασία του hyperparameter tuning ή την επιλογή αρχιτεκτονικής.

- Επικύρωση (Validation): Η επικύρωση είναι η διαδικασία αξιολόγησης της απόδοσης του μοντέλου σε ένα dataset που δεν έχει χρησιμοποιηθεί κατά τη διάρκεια της εκπαίδευσης. Σκοπός είναι να εκτιμηθεί η ικανότητα του μοντέλου να γενικεύει σε καινούργια δεδομένα και να εντοπίζει προβλήματα όπως η υπερπροσαρμογή.
  Βασικά χαρακτηριστικά της επικύρωσης:
    - Validation set: Το validation set είναι ένα υποσύνολο του αρχικού dataset που χρησιμοποιείται για την αξιολόγηση της απόδοσης του μοντέλου κατά τη διάρκεια της εκπαίδευσης και για το συντονισμό των υπερπαραμέτρων (hyperparameter tuning).

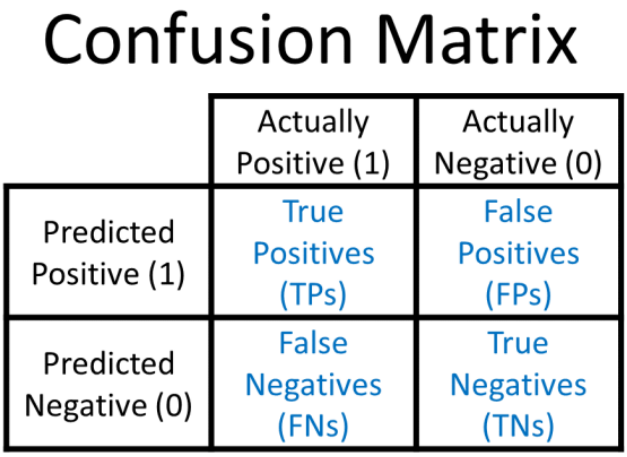
- Στρατηγικές επικύρωσης: Υπάρχουν διάφορες στρατηγικές για τη δημιουργία του validation set και την αξιολόγηση του μοντέλου όπως το k-fold cross-validation και το hold-out validation. Το k-fold cross-validation διαιρεί τα δεδομένα σε k ίσα υποσύνολα και σε κάθε επανάληψη εκπαιδεύεται το μοντέλο σε k-1 υποσύνολο επικυρώνεται στα υπόλοιπα υποσύνολα. Το hold-out validation χωρίζει το dataset σε σταθερά υποσύνολα εκπαίδευσης και επικύρωσης, συνήθως σε αναλογία 80 – 20 αντίστοιχα.
- Early stopping: Το early stopping είναι μια τεχνική που περιλαμβάνει την παρακολούθηση του validation loss κατά την εκπαίδευση του μοντέλου και σταματάει τη διαδικασία όταν αυτή η μετρική σταματά να βελτιώνεται. Αυτό βοηθάει στην αποφυγή υπερπροσαρμογής και μειώνει το υπολογιστικό κόστος της εκπαίδευσης.
- Επιλογή μοντέλου: Η παρακολούθηση της απόδοσης επικύρωσης χρειάζεται για να γίνει η επιλογή του καλύτερου μοντέλου από ένα σύνολο υποψήφιων μοντέλων με διαφορετικές αρχιτεκτονικές ή υπερπαραμέτρους. Το τελικό μοντέλο θα είναι αυτό με την καλύτερη απόδοση.

3.7.6 Αξιολόγηση του τελικού μοντέλου

Παρόλο που η αξιολόγηση του μοντέλου δεν είναι μέρος της εκπαιδευτικής διαδικασίας με την έννοια της ενημέρωσης των βαρών και άλλων υπερπαραμέτρων, είναι όμως μέρος της συνολικής διαδικασίας δημιουργίας ενός μοντέλου, που περιλαμβάνει τη συλλογή και προετοιμασία των δεδομένων, το σχεδιασμό της αρχιτεκτονικής και την εκπαίδευση του νευρωνικού δικτύου και τέλος την αξιολόγηση του τελικού μοντέλου πριν αρχίσουμε να το χρησιμοποιούμε.

Αυτό το βήμα έρχεται μετά την εκπαίδευση και επικύρωση του μοντέλου στο validation set και θεωρείται έτοιμο για δοκιμή, μιας και στην ουσία η αξιολόγηση είναι ένα είδους δοκιμής του μοντέλου σε δεδομένα που δεν έχει ξαναδεί. Η αξιολόγηση γίνεται σε ένα υποσύνολο του αρχικού dataset, το testing set, για να δώσει μια αμερόληπτη αξιολόγηση της απόδοσης του μοντέλου.

- Αξιολόγηση του μοντέλου στο test set: Στο test set γίνεται η αξιολόγηση της προγνωστικής απόδοσης του μοντέλου. Το μοντέλο κάνει προβλέψεις στο test set και τα αποτελέσματα συγκρίνονται με τις αληθινές ετικέτες. Οι ίδιες μετρήσεις που χρησιμοποιούνται και στην εκπαίδευση και επικύρωση, όπως οι ακρίβεια, ανάκληση, F1 score, υπολογίζονται για το test set για την τελική αξιολόγηση της απόδοσης του μοντέλου.
- Confusion matrix: Για προβλήματα ταξινόμησης κάποιες φορές χρησιμοποιείται ένας πίνακας, ο confusion matrix, για να απεικονίσει την απόδοση του μοντέλου. Κάθε γραμμή του πίνακα αντιπροσωπεύει τα στιγμιότυπα μιας πραγματικής κλάσης, ενώ κάθε στήλη αντιπροσωπεύει τα στιγμιότυπα της κλάσης που προέβλεψε το μοντέλο και παρέχει μια επισκόπηση του μοντέλου και των περιοχών λάθος ταξινόμησης.



Σχήμα 3.6 Confusion Matrix

- Precision: Το precision, ή αλλιώς θετική προγνωστική τιμή (Positive Predictive Value), μετρά την αναλογία των σωστών προβλέψεων θετικών παρατηρήσεων από τις συνολικές θετικές προβλέψεις. Είναι ένα μέτρο της ποιότητας ενός ταξινομητή και όσο πιο υψηλή είναι τόσο πιο χαμηλό θα είναι το ποσοστό των ψευδώς θετικών.
  Δίνεται από τον τύπο:

$$P = \frac{TP}{(TP+FP)} \tag{3.4}$$

  όπου στη δυαδική ταξινόμηση,
  TP (True Positives), είναι οι περιπτώσεις που το μοντέλο πρόβλεψε σωστά την κλάση,
  FP (False Positives), είναι οι περιπτώσεις που το μοντέλο πρόβλεψε λανθασμένα ότι η κλάση είναι η σωστή.

- Recall: Η ανάκληση μετρά τη αναλογία των πραγματικών θετικών που σωστά προσδιορίζονται ως θετικά. Είναι ένα μέτρο της πληρότητας ενός ταξινομητή. Η υψηλή ανάκληση δείχνει χαμηλό ποσοστό ψευδώς αρνητικών.
  Η ανάκληση δίνεται από τον τύπο:

$$R = \frac{TP}{(TP+FN)} \tag{3.5}$$

  όπου στη δυαδική ταξινόμηση,
  FN (False Negatives), είναι οι περιπτώσεις που το μοντέλο προβλέπει λανθασμένα ότι είναι η λάθος κλάση.

- F1 Score: Το F1 Score είναι ο αρμονικός μέσος όρος του precision και του recall και προσπαθεί να βρει την ισορροπία μεταξύ τους. Είναι χρήσιμο σε περιπτώσεις που η θετική κλάση είναι λιγότερο συχνή από την αρνητική.
  Το F1 score δίνεται από τον τύπο:

$$F1 = 2 * \frac{P*R}{P+R} \quad (3.6)$$

- Accuracy: Το accuracy είναι από τις πιο απλές μετρήσεις που χρησιμοποιούνται και υπολογίζει την αναλογία των σωστών προβλέψεων που έγιναν από το μοντέλο σε σχέση με όλες τις προβλέψεις που έγιναν.
  Επίσης, είναι πιο χρήσιμη όταν ο αριθμός των εικόνων για κάθε κλάση στο dataset είναι σχεδόν ίσος.
  Η ακρίβεια δίνεται από το τύπο:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.7)$$

όπου στη δυαδική ταξινόμηση,
TP (True Positives), είναι οι περιπτώσεις που το μοντέλο πρόβλεψε σωστά την κλάση,
TN (True Negatives), είναι οι περιπτώσεις που το μοντέλο πρόβλεψε σωστά ότι είναι η λάθος κλάση,
FP (False Positives), είναι οι περιπτώσεις που το μοντέλο πρόβλεψε λανθασμένα ότι η κλάση είναι η σωστή,
FN (False Negatives), είναι οι περιπτώσεις που το μοντέλο προβλέπει λανθασμένα ότι είναι η λάθος κλάση.

Στη ταξινόμηση πολλαπλών κλάσεων οι έννοιες αυτές είναι λίγο πιο πολύπλοκες από ότι στη δυαδική ταξινόμηση.
Ας πάρουμε ένα παράδειγμα ταξινόμησης πολλαπλών κλάσεων για να το δούμε λίγο καλύτερα, για παράδειγμα έχουμε τρεις κλάσεις, γάτα, σκύλος και αυτοκίνητο.

- TP (True Positives): Είναι ο αριθμός των φορών που το μοντέλο προβλέπει σωστά τη θετική κλάση. Έχουμε εικόνα με μια γάτα και το μοντέλο προβλέπει την κλάση γάτα.
- TN (True Negatives): Είναι ο αριθμός των φορών που το μοντέλο προβλέπει σωστά ότι μια εικόνα δεν ανήκει στη συγκεκριμένη κλάση. Έχουμε μια εικόνα με μια γάτα και το μοντέλο προβλέπει σωστά ότι δεν είναι γάτα, άρα είναι TN για την κλάση γάτα.

- FP (False Positives): Είναι ο αριθμός των φορών που το μοντέλο προβλέπει λανθασμένα τη σωστή κλάση. Έχουμε την εικόνα με τη γάτα και το μοντέλο προβλέπει τη κλάση σκύλος. Τότε αυτό είναι FP για την κλάση σκύλος.
- FN (False Negatives): Είναι ο αριθμός των φορών που το μοντέλο προβλέπει λανθασμένα την λάθος κλάση. Στην εικόνα με τη γάτα το μοντέλο προβλέπει ως σκύλο ή αυτοκίνητο. Τότε είναι FN για την κλάση γάτα.

Αυτοί οι ορισμοί μπορούν να επεκταθούν σε x αριθμό κλάσεων θεωρώντας μια κλάση θετική και όλες τις άλλες αρνητικές.

3.8 Υλοποίηση CNN

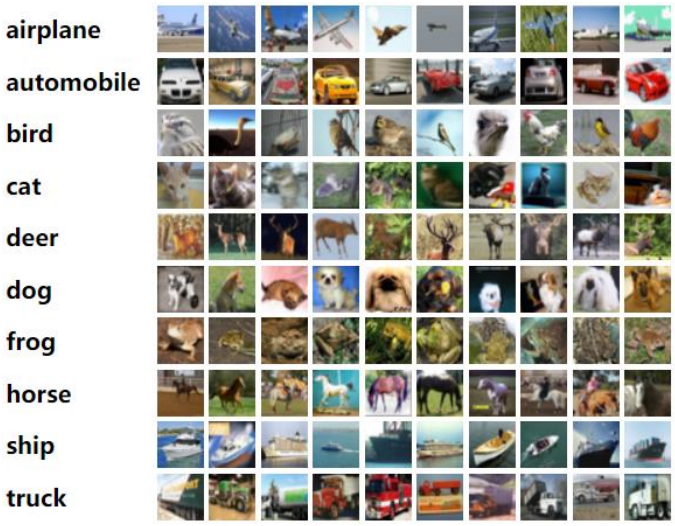
Σε αυτό το υπό κεφάλαιο θα δούμε το dataset που θα χρησιμοποιήσουμε για την εκπαίδευση και αξιολόγηση του μοντέλου αλλά και την υλοποίηση του cnn μοντέλου για ταξινόμηση εικόνων, με επεξήγηση του κώδικα που χρησιμοποιούμε σε αυτή την πτυχιακή εργασία. Η υλοποίηση του μοντέλου έγινε με τη γλώσσα προγραμματισμού python.

3.8.1 Δεδομένα (Dataset)

Στην παρούσα πτυχιακή εργασία το dataset που χρησιμοποιήθηκε για την εκπαίδευση και αξιολόγηση του μοντέλου είναι το CIFAR-10 (**C**anadian **I**nstitute **F**or **A**dvanced **R**esearch). Είναι ένα πολύ δημοφιλή dataset που χρησιμοποιείται σε εργασίες μηχανικής όρασης και ειδικά για εργασίες ταξινόμησης. Ο αριθμός 10 στο όνομά του αναφέρεται στον αριθμό των κατηγοριών που υπάρχουν στο dataset.

Χαρακτηριστικά του dataset:

Όπως αναφέρεται και στην επίσημη ιστοσελίδα, το cifar-10 αποτελείται από 60000 έγχρωμες εικόνες όπου οι 50000 (training set) είναι για την φάση της εκπαίδευσης και οι 10000 (test set) είναι για την φάση της αξιολόγησης της επίδοσης του μοντέλου, ανάλυσης 32x32 χωρισμένες σε 10 κατηγορίες με 6000 εικόνες ανά κατηγορία. Οι κατηγορίες που υπάρχουν στο dataset είναι **αεροπλάνο, αυτοκίνητο, πουλί, γάτα, ελάφι, σκύλος, βάτραχος, άλογο, πλοίο,** και **φορτηγό.**



Σχήμα 3.7 Cifar-10 Dataset

3.8.2 Βιβλιοθήκες

Για την υλοποίηση του μοντέλου χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες:

- Tensorflow: Το tensorflow είναι μια βιβλιοθήκη ανοιχτού κώδικα που μας παρέχει τα απαραίτητα εργαλεία για να δημιουργήσουμε και να εκπαιδεύσουμε νευρωνικά δίκτυα ή να χρησιμοποιήσουμε υπάρχων αλγόριθμους μηχανικής μάθησης.
- Keras: Το keras είναι μια βιβλιοθήκη του tensorflow, είναι ένα από τα εργαλεία που μας παρέχει μια φιλική προς το χρήστη διεπαφή για να δημιουργήσουμε και να εκπαιδεύσουμε εύκολα νευρωνικά δίκτυα. Μας παρέχει μεταξύ άλλων τα δομικά στοιχεία των νευρωνικών δικτύων όπως είναι τα επίπεδα, συναρτήσεις ενεργοποίησης, optimizers και άλλα εργαλεία για την προεπεξεργασία των δεδομένων.
- Visual keras: Είναι μια βιβλιοθήκη που μας βοηθάει στην οπτικοποίηση της αρχιτεκτονικής του μοντέλου.
- Keras tuner: Το keras tuner είναι μια βιβλιοθήκη για hyperparameter tuning που μας επιτρέπει να ψάχνουμε αυτόματα για τις καλύτερες υπερπαραμέτρους που θα οδηγήσουν στην εκπαίδευση του καλύτερου δυνατού μοντέλου.
- Matplotlib – Seaborn: Αυτές οι δύο βιβλιοθήκες μας παρέχουν εργαλεία για να δημιουργήσουμε γραφικές και visualizations.

```
!pip install keras-tuner -q
!pip install visualkeras

import visualkeras
import tensorflow as tf
import keras_tuner as kt
from tensorflow.keras.models import load_model, Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img , img_to_array
from tensorflow.keras.layers import Dense , Dropout , Conv2D , MaxPooling2D, Flatten , BatchNormalization
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.optimizers import Adam
from keras_tuner.engine.hyperparameters import HyperParameters
from keras_tuner import HyperModel
import matplotlib.pyplot as plt
import seaborn as sb
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

Σχήμα 3.8 Εισαγωγή Βιβλιοθηκών

Μετά τις βιβλιοθήκες γίνεται αρχικοποίηση των παρακάτω παραμέτρων:

```
num_classes = 10
input_shape = (32, 32, 3)
weight_decay = 0.0001
batch_size = 32
epoch = 100
```

Σχήμα 3.9 Αρχικοποίηση Μεταβλητών

- num_classes: είναι ο αριθμός των κλάσεων που υπάρχει στο dataset
- input_shape: είναι οι διαστάσεις των δεδομένων που περιμένει το δίκτυο ως είσοδο
- weight_decay: είναι μια υπερπαράμετρος που ελέγχει την ποσότητα μείωσης βάρους στην τεχνική regularization L2 κατά την εκπαίδευση
- batch_size: είναι ο αριθμός των δεδομένων εκπαίδευσης που χρησιμοποιούνται σε κάθε επανάληψη
- epoch: είναι ο αριθμός των επαναλήψεων σε ολόκληρο το training set.

3.8.3 Data Preprocessing

Το πρώτο βήμα που πρέπει να κάνουμε είναι να φορτώσουμε τα δεδομένα.

```
# Load the CIFAR10 dataset
(x_train_full , y_train_full), (x_test , y_test) = tf.keras.datasets.cifar10.load_data()

# Split the full training set into a validation set and a (smaller) training set
x_train, x_val, y_train, y_val = train_test_split(x_train_full, y_train_full, test_size=0.2, random_state=42)
```

Σχήμα 3.10 Load and Split Dataset

Όπως βλέπουμε και στην παραπάνω εικόνα, χρησιμοποιούμε τη βιβλιοθήκη keras που μας παρέχει το cifar10 dataset και καλώντας την function load_data() κατεβάζουμε και φορτώνουμε τις εικόνες και τις ετικέτες του cifar10 χωρισμένες σε training και test sets που αποτελούνται από 50000 και 10000 αντίστοιχα και μετά χωρίζουμε το training set σε μικρότερο training set και validation set σε αναλογία 80 – 20 αντίστοιχα.

Το x_train, x_val και x_test περιέχουν τις εικόνες του training, validation και test set αντίστοιχα στη μορφή πίνακα, όπου το περιέχει τον αριθμό των εικόνων και τις διαστάσεις των εικόνων, ύψος, πλάτος αλλά και τα κανάλια, που στην προκειμένη περίπτωση είναι 3 αφού έχουμε έγχρωμες (RGB) εικόνες.

Τα y_train, y_val και y_test από την άλλη, περιέχουν τις ετικέτες των εικόνων των training, validation και test set αντίστοιχα.

```
print('Train Images', x_train.shape[0])
print('Val Images', x_val.shape[0])
print('Test Images', x_test.shape[0])
```

```
Train Images 40000
Val Images 10000
Test Images 10000
```

Σχήμα 3.11 Σχήμα Υποσυνόλων Δεδομένων

Στη συνέχεια, γίνεται κανονικοποίηση των pixel των εικόνων για να έχουμε σταθερή κλίμακα των τιμών. Αυτό γίνεται με την normalize function που φτιάξαμε που αυτό που κάνει είναι να μετατρέπει τις τιμές σε δεκαδικό αριθμό (float) και να διαιρεί τις τιμές των pixel με το 255,0 για να τις μετατρέψει σε κλίμακα μεταξύ 0 και 1 αφού σε έγχρωμες (RGB) εικόνες 255 είναι η μέγιστη τιμή pixel.

```
def normalize(x):
    return x.astype('float32') / 255.0

datagen = ImageDataGenerator(rotation_range=15,
                             width_shift_range=0.1,
                             height_shift_range=0.1,
                             horizontal_flip=True)

x_train = normalize(x_train)
x_test = normalize(x_test)

y_train = to_categorical(y_train , num_classes)
y_test = to_categorical(y_test , num_classes)

datagen.fit(x_train)
```

Σχήμα 3.12 Κανονικοποίηση Δεδομένων

Επίσης, αμέσως μετά την κανονικοποίηση των pixel των εικόνων, μετατρέπουμε τις ετικέτες (y_train και y_test) σε κωδικοποιημένα (one-hot encoded) διανύσματα, όπου το στοιχείο που αντιστοιχεί στην ετικέτα κλάσης είναι 1 και όλα τα άλλα στοιχεία είναι 0.

Για παράδειγμα, αφού το dataset που έχουμε έχει 10 κλάσεις, σε μια συγκεκριμένη εικόνα από το training ή test set ανήκει στην κλάση 5 τότε το διάνυσμα του συγκεκριμένου στοιχείου – ετικέτας θα είναι [0, 0, 0, 0, 1, 0, 0, 0, 0, 0].

Η μετατροπή των ετικετών σε κωδικοποιημένα διανύσματα βοηθάει το μοντέλο να ερμηνεύει τις ετικέτες ως διακριτές κατηγορίες και όχι ως τακτικές τιμές.

Τέλος, χρησιμοποιούμε το ImageDataGenerator για data augmentation. Με αυτή την τεχνική παράγουμε καινούριες τεχνητές εικόνες από τις υπάρχουσες, αλλά είναι λίγο αλλαγμένες, όπως για παράδειγμα έχουν υποστεί περιστροφή έτσι ώστε να υπάρχουν ακόμα περισσότερα training data για καλύτερη εκπαίδευση.

3.8.4 Αρχιτεκτονική του μοντέλου

Στην παρακάτω φωτογραφία βλέπουμε την κλάση δημιουργίας του αρχικού μοντέλου.

```python
class CustomModel:
    def __init__(self, input_shape, num_classes, weight_decay):
        self.input_shape = input_shape
        self.num_classes = num_classes
        self.weight_decay = weight_decay

    def create_model(self):
      custom_model = Sequential([Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularizer=tf.keras.regularizers.l2(weight_decay),
                                 input_shape=input_shape),
                          BatchNormalization(),
                          Conv2D(64, (3, 3), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(weight_decay), padding='same'),
                          BatchNormalization(),
                          MaxPooling2D((2, 2)),
                          Dropout(0.2),

                          Conv2D(128, (3, 3), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(weight_decay), padding='same'),
                          BatchNormalization(),
                          Conv2D(128, (3, 3), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(weight_decay), padding='same'),
                          BatchNormalization(),
                          MaxPooling2D((2, 2)),
                          Dropout(0.3),

                          Conv2D(256, (3, 3), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(weight_decay), padding='same'),
                          BatchNormalization(),
                          Conv2D(256, (3, 3), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(weight_decay), padding='same'),
                          BatchNormalization(),
                          MaxPooling2D((2, 2)),
                          Dropout(0.4),

                          Flatten(),
                          Dense(256, activation='relu'),
                          BatchNormalization(),
                          Dropout(0.5),
                          Dense(num_classes, activation='softmax')])

      return custom_model
```

Σχήμα 3.13 Αρχιτεκτονική Μοντέλου

Το αρχικό μοντέλο είναι ένα cnn μοντέλο που αποτελείται από 6 συνελικτικά επίπεδα (Conv2D), 6 επίπεδα BatchNormalization, 3 επίπεδα συγκέντρωσης (MaxPooling2D), 4 επίπεδα dropout, 1 flatten επίπεδο και 2 πλήρως συνδεδεμένα επίπεδα (Dense).

Πιο αναλυτικά:

- Conv2D:
    - Στα πρώτα 2 επίπεδα έχουμε από 64 φίλτρα, το τρίτο και τέταρτο επίπεδο έχουν 128 φίλτρα και το πέμπτο και έκτο από 256 φίλτρα, με μέγεθος φίλτρων 3x3.
    - Σε όλα τα συνελικτικά επίπεδα χρησιμοποιούμε την συνάρτηση ενεργοποίησης relu.
    - Για συμπλήρωση, padding, χρησιμοποιούμε same για την προσθήκη μηδενικών στην εικόνα εισόδου για να μην αλλάζει τις χωρικές διαστάσεις της εικόνας.

    - Τεχνική regularization χρησιμοποιούμε την τεχνική L2 για αποφυγή υπερπροσαρμογής.
    - Και τέλος έχουμε και το input shape που ορίζουμε τις διαστάσεις εισόδου.
- BatchNormalization: Αυτά τα επίπεδα τα χρησιμοποιούμε για την κανονικοποίηση των εξόδων των συναρτήσεων ενεργοποίησης των προηγούμενων επιπέδων.
- MaxPooling2D: Εδώ χρησιμοποιούμε τη συγκέντρωση μέγιστου για τη μείωση δειγματοληψίας της εισόδου κατά ύψος και πλάτος και χρησιμοποιούμε μέγεθος 2x2 και λαμβάνεται η μέγιστη τιμή σε κάθε παράθυρο 2x2.
- Dropout: Για την αποφυγή της υπερπροσαρμογής χρησιμοποιούμε dropout που αυξάνεται από 0.2 σε 0.5 όσο προχωράμε σε βαθύτερα επίπεδα.
- Flatten: Χρησιμοποιούμε ένα επίπεδο flatten που όπως έχουμε δει χρησιμοποιείται για τη μετατροπή της 3D εξόδου του προηγούμενου επίπεδου σε 1D.
- Dense: Τέλος, έχουμε τα πλήρως συνδεδεμένα επίπεδα, το πρώτο έχει 256 νευρώνες που συνδέονται με το προηγούμενο επίπεδο και συνάρτηση ενεργοποίησης relu. Το τελικό επίπεδο όσους νευρώνες όσες και ο αριθμός κλάσεων, num_classes, και έχει συνάρτηση ενεργοποίησης softmax.

Τέλος φτιάχνουμε το μοντέλο με τη create_model που έχουμε φτιάξει.

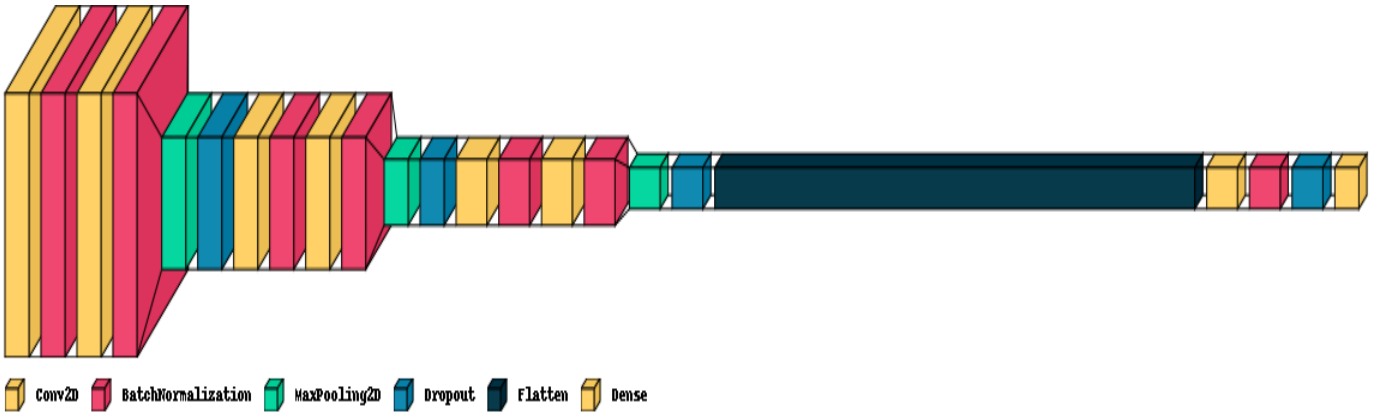
```
custom_model = CustomModel(input_shape=input_shape, num_classes=num_classes, weight_decay=weight_decay).create_model()
```

Σχήμα 3.14 Δημιουργία Μοντέλου

και χρησιμοποιούμε τη βιβλιοθήκη visualkeras για να οπτικοποιήσουμε την αρχιτεκτονική του μοντέλου.

```
visualkeras.layered_view(custom_model).show()
```

Σχήμα 3.15 Οπτικοποίηση Μοντέλου



Σχήμα 3.16 Αρχιτεκτονική 1ου Μοντέλου

Εδώ βλέπουμε την αρχιτεκτονική του αρχικού μοντέλου πριν γίνει hyperparameter optimization.

Επίσης η αρχιτεκτονική του μοντέλου σε μορφή πίνακα. Με αυτόν τον πίνακα μπορούμε να καταλάβουμε εύκολα την αρχιτεκτονική του μοντέλου, βλέποντας τα επίπεδα και τον τύπο τους, την έξοδό τους και τον αριθμό των παραμέτρων τους.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 64)        1792

 batch_normalization (BatchN  (None, 32, 32, 64)       256
 ormalization)

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        36928

 batch_normalization_1 (Batc  (None, 32, 32, 64)       256
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 dropout (Dropout)           (None, 16, 16, 64)        0

 conv2d_2 (Conv2D)           (None, 16, 16, 128)       73856

 batch_normalization_2 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 conv2d_3 (Conv2D)           (None, 16, 16, 128)       147584

 batch_normalization_3 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 8, 8, 128)         0

 conv2d_4 (Conv2D)           (None, 8, 8, 256)         295168

 batch_normalization_4 (Batc  (None, 8, 8, 256)        1024
 hNormalization)

 conv2d_5 (Conv2D)           (None, 8, 8, 256)         590080
```

Σχήμα 3.17 Πίνακας Επιπέδων Μοντέλου 1/2

```
batch_normalization_5 (Batc  (None, 8, 8, 256)        1024
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 4, 4, 256)        0
2D)

dropout_2 (Dropout)          (None, 4, 4, 256)        0

flatten (Flatten)            (None, 4096)             0

dense (Dense)                (None, 256)              1048832

batch_normalization_6 (Batc  (None, 256)              1024
hNormalization)

dropout_3 (Dropout)          (None, 256)              0

dense_1 (Dense)              (None, 10)               2570

=================================================================
Total params: 2,201,418
Trainable params: 2,199,114
Non-trainable params: 2,304
```

Σχήμα 3.18 Πίνακας Επιπέδων Μοντέλου 2/2

Το “Layer (Type)” αναφέρεται στο όνομα και τον τύπο του επιπέδου, για παράδειγμα conv2d, conv2d_1, το “Output Shape” είναι το σχήμα των δεδομένων που έχει ως έξοδο το επίπεδο και θα λάβει ως είσοδο το επόμενο, και “Param” είναι ο αριθμός παραμέτρων, και υπολογίζονται ως εξής:

- Οι παράμετροι στα συνελικτικά επίπεδα (conv2d) υπολογίζονται με τον τύπο (filter_height * filter_width * input_depth +1) * number_of_filters, όπου input_depth είναι ο αριθμός των καναλιών, 3 για RGB εικόνες και το +1 αναφέρεται στο bias.
- Οι παράμετροι στο batch normalization επίπεδο υπολογίζονται ως, 4 * 256, όπου 256 είναι η είσοδός του από το προηγούμενο επίπεδο, και το 4 είναι οι 4 παράμετροι της εικόνας που λαμβάνει υπόψη του, scale, offsetting, mean και variance.
- Οι παράμετροι του dense layer υπολογίζονται ως, (number of inputs (4096) +1 (bias)) * 256 (number of outputs)
- Τα υπόλοιπα επίπεδα, MaxPooling2D, Dropout, Flatten, δεν έχουν παραμέτρους που εκπαιδεύονται.

3.8.5 Εκπαίδευση CNN χωρίς hyperparameter optimization

Σε αυτό το στάδιο θα δούμε την εκπαίδευση του μοντέλου χωρίς hyperparameter optimization. Πριν ξεκινήσει η εκπαίδευση του μοντέλου χρησιμοποιούμε τη μέθοδο compile για τη διαμόρφωση της διαδικασίας εκμάθησης και ορίζει τη συνάρτηση σφάλματος, τον optimizer και τον ρυθμό μάθησής του, με αρχική τιμή **0.01**, και τη μετρική accuracy που παρακολουθείται κατά την εκπαίδευση.

```
custom_model.compile(loss='categorical_crossentropy',
                     optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
                     metrics=['accuracy'])
```

Σχήμα 3.19 Διαμόρφωση Παραμέτρων Μοντέλου

Στη συνέχεια κάνουμε αρχικοποίηση των callback.

```
callbacks = [ReduceLROnPlateau(monitor='val_loss',
                               patience=10,
                               cooldown=1,
                               verbose=1),
             EarlyStopping(monitor='val_loss',
                           patience=15,
                           verbose=1)]
```

Σχήμα 3.20 Αρχικοποίηση Callbacks

Τα callbacks χρησιμεύουν για 2 πράγματα:

- ReduceLROnPlateau: Το χρησιμοποιούμε για να παρακολουθεί το validation loss και για να μειώσει τον ρυθμό μάθησης αν σε 10 επαναλήψεις, patience, δεν βελτιωθεί. Επίσης το cooldown αναφέρεται σε πόσες επαναλήψεις θα επιστρέψει στην κανονική εκπαιδευτική διαδικασία μετά τη μείωση του ρυθμού μάθησης.
- EarlyStopping: Το χρησιμοποιούμε για να παρακολουθεί το validation loss και να σταματήσει τη διαδικασία της εκπαίδευσης αν δεν βελτιωθεί μετά από 15 επαναλήψεις, patience.

Αφού γίνει compile το μοντέλο και ορίσουμε τα callbacks, μπορεί να ξεκινήσει η εκπαίδευση του μοντέλου.

Για την εκκίνηση της εκπαίδευσης του μοντέλου χρησιμοποιούμε την εντολή που βλέπουμε στην παρακάτω εικόνα

```
history = custom_model.fit(
    datagen.flow(x_train , y_train , batch_size = batch_size),
    epochs = epoch,
    validation_data = (x_val, y_val),
    callbacks=callbacks)
```

Σχήμα 3.21 Έναρξη Διαδικασίας Εκπαίδευσης

- Η μέθοδος fit() προσαρμόζει τις παραμέτρους του μοντέλου για να ελαχιστοποιήσει το σφάλμα.
- Η μέθοδος datagen.flow() δημιουργεί μια γεννήτρια δεδομένων από το ImageDataGenerator και περνάει αυτόματα τις εικόνες εκπαίδευσης στο μοντέλο, x_train, y_train ανα παρτίδες μεγέθους batch_size, που έχουμε αρχικοποιήσει στην αρχή με 32.
- Επίσης περνάμε τον αριθμό επαναλήψεων (epochs) που έχουμε επίσης αρχικοποιήσει με τον αριθμό 100.
- Περνάμε τα validation_data για να γίνει αξιολόγηση του μοντέλου κατά την εκπαίδευση, x_val, y_val.
- Τέλος, περνάμε τα callbacks που όπως έχουμε δει, τα χρησιμοποιούμε για να μειώσει το ρυθμό μάθησης κατά την εκπαίδευση καθώς και να σταματήσει τη διαδικασία αν δεν βελτιώνεται το σφάλμα.

```
Epoch 15/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3648 - accuracy: 0.6203 - val_loss: 1.3913 - val_accuracy: 0.6276 - lr: 0.0100
Epoch 16/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.4560 - accuracy: 0.6012 - val_loss: 1.5738 - val_accuracy: 0.5816 - lr: 0.0100
Epoch 17/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3872 - accuracy: 0.6182 - val_loss: 1.2761 - val_accuracy: 0.6515 - lr: 0.0100
Epoch 18/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3552 - accuracy: 0.6199 - val_loss: 1.2956 - val_accuracy: 0.6497 - lr: 0.0100
Epoch 19/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3538 - accuracy: 0.6214 - val_loss: 1.3688 - val_accuracy: 0.6113 - lr: 0.0100
Epoch 20/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3595 - accuracy: 0.6227 - val_loss: 1.5357 - val_accuracy: 0.6003 - lr: 0.0100
Epoch 21/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3522 - accuracy: 0.6254 - val_loss: 1.4498 - val_accuracy: 0.6012 - lr: 0.0100
Epoch 22/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3383 - accuracy: 0.6276 - val_loss: 1.6006 - val_accuracy: 0.5800 - lr: 0.0100
Epoch 23/100
1561/1563 [============================>.] - ETA: 0s - loss: 1.3440 - accuracy: 0.6291
Epoch 23: ReduceLROnPlateau reducing learning rate to 0.0009999999776482583.
1563/1563 [==============================] - 29s 19ms/step - loss: 1.3439 - accuracy: 0.6291 - val_loss: 1.4999 - val_accuracy: 0.5967 - lr: 0.0100
Epoch 24/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.1406 - accuracy: 0.6836 - val_loss: 0.9790 - val_accuracy: 0.7304 - lr: 1.0000e-03
Epoch 25/100
1563/1563 [==============================] - 29s 19ms/step - loss: 1.0402 - accuracy: 0.7052 - val_loss: 0.8884 - val_accuracy: 0.7460 - lr: 1.0000e-03
```

Σχήμα 3.227 Epochs (Επαναλήψεις)

Στην παραπάνω φωτογραφία βλέπουμε τη μείωση του ρυθμού μάθησης, στην εποχή 23, που έχουμε ρυθμίσει στα callbacks.

Το ιστορικό της εκπαίδευσης αποθηκεύεται στη μεταβλητή history για να μπορούμε αργότερα να δούμε τις τιμές των μετρικών που θέλουμε καθώς και να βγάλουμε τα διαγράμματα που μας ενδιαφέρουν.

3.8.6 Εκπαίδευση CNN με hyperparameter optimization

Για να χρησιμοποιήσουμε την τεχνική hyperparameter optimization και να έχουμε ένα βελτιωμένο μοντέλο πρέπει να κάνουμε τις εξής αλλαγές:

```
class CustomModel(HyperModel):
    def __init__(self, input_shape, num_classes, weight_decay):
        self.input_shape = input_shape
        self.num_classes = num_classes
        self.weight_decay = weight_decay

    def create_model(self, hp):
      custom_model = Sequential([Conv2D(hp.Int('conv_1', min_value=64, max_value=128, step=64), (3, 3),
                                 activation='relu', padding='same', kernel_regularizer=tf.keras.regularizers.l2(self.weight_decay),
                                 input_shape=self.input_shape), BatchNormalization(),
                           Conv2D(hp.Int('conv_1', min_value=64, max_value=128, step=64), (3, 3),
                                 activation='relu', kernel_regularizer=tf.keras.regularizers.l2(self.weight_decay),
                                 padding='same'), BatchNormalization(),
                           MaxPooling2D((2, 2)),
                           Dropout(0.2),

                           Conv2D(hp.Int('conv_2', min_value=128, max_value=256, step=128), (3, 3),
                                 activation='relu', kernel_regularizer=tf.keras.regularizers.l2(self.weight_decay), padding='same'),
                           BatchNormalization(),
                           Conv2D(hp.Int('conv_2', min_value=128, max_value=256, step=128), (3, 3),
                                 activation='relu', kernel_regularizer=tf.keras.regularizers.l2(self.weight_decay), padding='same'),
                           BatchNormalization(),
                           MaxPooling2D((2, 2)),
                           Dropout(0.3),

                           Conv2D(hp.Int('conv_3', min_value=256, max_value=512, step=256), (3, 3),
                                 activation='relu', kernel_regularizer=tf.keras.regularizers.l2(self.weight_decay), padding='same'),
                           BatchNormalization(),
                           Conv2D(hp.Int('conv_3', min_value=256, max_value=512, step=256), (3, 3),
                                 activation='relu', kernel_regularizer=tf.keras.regularizers.l2(self.weight_decay), padding='same'),
                           BatchNormalization(),
                           MaxPooling2D((2, 2)),
                           Dropout(0.4),

                           Flatten(),
                           Dense(hp.Int('dense_units', min_value=256, max_value=1024, step=256), activation='relu'),
                           BatchNormalization(),
                           Dropout(0.5),
                           Dense(self.num_classes, activation='softmax')])
```

Σχήμα 3.23 Αρχιτεκτονική Μοντέλου με Hyperparameter Tuning

Η πρώτη αλλαγή γίνεται στη κλάση που φτιάχνει την αρχιτεκτονική του μοντέλου. Πλέον η κλάση CustomModel κληρονομεί από τη κλάση HyperModel του keras-tuner για να έχουμε πρόσβαση στις μεθόδους του για το optimization. Η δεύτερη αλλαγή όπως βλέπουμε και στην παραπάνω εικόνα είναι στον αριθμό των φίλτρων, πλέον δεν δηλώνουμε συγκεκριμένο αριθμό φίλτρων στα

επίπεδα, αλλά μια ελάχιστη και μια μέγιστη τιμή, 64 και 128 αντίστοιχα, και ένα βήμα, 64, για να δηλώσουμε ότι σε αυτό το επίπεδο θέλουμε να βρούμε την βέλτιστη τιμή φίλτρων για το επίπεδο ανάμεσα στις τιμές 64 και 128 με βήμα 64. Το ίδιο συμβαίνει και με τα υπόλοιπα επίπεδα με διαφορετικές τιμές.

Με αυτόν τον τρόπο μπορούμε να δοκιμάσουμε διαφορετικές τιμές φίλτρων σε μια εκπαιδευτική διαδικασία χωρίς να χρειάζεται να αλλάζουμε με το χέρι τις τιμές των φίλτρων και να εκπαιδεύουμε από την αρχή κάθε μοντέλο μέχρι να βρούμε τις βέλτιστες τιμές κάθε υπερπαράμετρου έτσι ώστε να παραχθεί το βέλτιστο μοντέλο.

Το ίδιο κάνουμε με το ρυθμό μάθησης, με τη διαφορά ότι εδώ του έχουμε δηλώσει 3 συγκεκριμένους ρυθμούς μάθησης να τους δοκιμάσει και να μας δώσει τον ρυθμό μάθησης που βγάζει μοντέλο με καλύτερα αποτελέσματα.

```
  model_lr = hp.Choice("learning_rate", values=[1e-2, 1e-3, 1e-4])

  custom_model.compile(optimizer=Adam(learning_rate=model_lr), loss='categorical_crossentropy', metrics=['accuracy'])


  return custom_model

def optimize(self, x_train, y_train, x_test, y_test, trials=5):
    tuner = kt.Hyperband(hypermodel=self.create_model, objective='val_accuracy', max_epochs=25, factor=3,
                         seed=42, overwrite=True,
                         directory='/content/gdrive/My Drive/Image Classification/with_tuning/', project_name='ImageClassification')

    tuner.search_space_summary()

    print('performing hyperparameter search...')
    tuner.search(x_train, y_train, validation_data=(x_test, y_test), epochs=5,
                 callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)])
    # Get the best hyperparameters
    best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
    print("Best learning rate: {:.4f}".format(best_hps.get("learning_rate")))
    print("[INFO] optimal number of filters in conv_1 layer: {}".format(best_hps.get("conv_1")))
    print("[INFO] optimal number of filters in conv_2 layer: {}".format(best_hps.get("conv_2")))
    print("[INFO] optimal number of filters in conv_3 layer: {}".format(best_hps.get("conv_3")))
    print("[INFO] optimal number of units in dense layer: {}".format(best_hps.get("dense_units")))

    best_model = tuner.get_best_models(num_models=1)[0]

    # Build the model with the best hyperparameters.
    best_model = tuner.hypermodel.build(best_hps)

    return best_model
```

Σχήμα 3.24 Μέθοδος Έναρξης Hyperparameter Tuning

Άλλη αλλαγή είναι ότι πλέον εκτός από την create_model() μέθοδο έχουμε και την optimize() μέθοδο η οποία ξεκινάει τη διαδικασία να ψάχνει και μας γυρνάει το βέλτιστο μοντέλο.

```
best_model = CustomModel(input_shape=input_shape,
                         num_classes=num_classes, weight_decay=weight_decay).optimize(x_train, y_train, x_val, y_val)
```

Σχήμα 3.25 Δημιουργία Μοντέλου και Έναρξη Optimization

Τέλος βλέπουμε ότι δημιουργούμε το custom_model object όπως κάναμε και πριν, αλλά μετά καλούμε την optimize() μέθοδο για να ξεκινήσει η διαδικασία του hyperparameter tuning και να πάρουμε το μοντέλο με την καλύτερη απόδοση.

Σε αυτή την πτυχιακή εργασία χρησιμοποιούμε την Hyperband μέθοδο, μια hyperparameter tuning μέθοδο, που χρησιμοποιεί τις αρχές τυχαίας αναζήτησης και early stopping για αποτελεσματική βελτιστοποίηση του μοντέλου. Η μέθοδος Hyperband χρησιμοποιεί την έννοια των brackets (αγκύλες), και κάθε μια από αυτές τις αγκύλες αντιστοιχεί σε έναν αριθμό πόρων που χρησιμοποιούνται στις διαμορφώσεις των υπερπαραμέτρων. Το Hyperband εκτελεί πολλές τυχαίες αναζητήσεις με διαφορετικά επίπεδα επιθετικού early stopping και εξισορροπεί την διαφορά μεταξύ εξερεύνησης και εκμετάλλευσης πόρων, βάζοντας περισσότερους πόρους στις πιο υποσχόμενες διαμορφώσεις των υπερπαραμέτρων. Στόχος μας είναι να πετύχουμε υψηλότερη ακρίβεια, accuracy, στο test set.

Στη συνέχεια γίνεται χρήση της μεθόδου search() που εκτελεί την αναζήτηση υπερπαραμέτρων με τον αλγόριθμο Hyperband. Δίνουμε τα δεδομένα εκπαίδευσης, τα δεδομένα επικύρωσης, τον αριθμό επαναλήψεων για κάθε δοκιμή και τα callbacks κυρίως για το early stopping.

Με λίγα λόγια:

- Η μέθοδος search() πρώτα φτιάχνει το σύνολο των διαμορφώσεων των υπερπαραμέτρων χρησιμοποιώντας την μέθοδο build() του αλγόριθμου Hyperband.
- Φτιάχνει και εκπαιδεύει ένα μοντέλο για κάθε μια από τις διαμορφώσεις (configurations).
- Μετά από κάθε κύκλο εκπαίδευσης υπολογίζει την ακρίβεια στο validation set για κάθε μοντέλο.
- Διατηρεί έναν αριθμό μοντέλων με τις καλύτερες αποδόσεις, στην προκειμένη περίπτωση 3 μοντέλα, που το ορίζουμε με την παράμετρο factor στον αλγόριθμο Hyperband, και απορρίπτει τα υπόλοιπα για να ελευθερώσει υπολογιστικούς πόρους.
- Και επαναλαμβάνει αυτή τη διαδικασία εκπαιδεύοντας κάθε configuration των υπερπαραμέτρων, κάθε συνδυασμό, για περισσότερες εποχές και απορρίπτει τις λιγότερο υποσχόμενες μέχρι να μείνει το καλύτερο δυνατό σύνολο υπερπαραμέτρων.

Έτσι βρίσκουμε τις καλύτερες υπερπαραμέτρους για το μοντέλο μας.

```
Trial 30 Complete [00h 01m 28s]
val_accuracy: 0.8055999875068665

Best val_accuracy So Far: 0.8598999977111816
Total elapsed time: 00h 39m 13s
Best learning rate: 0.0010
[INFO] optimal number of filters in conv_1 layer: 64
[INFO] optimal number of filters in conv_2 layer: 128
[INFO] optimal number of filters in conv_3 layer: 256
[INFO] optimal number of units in dense layer: 256
```

Σχήμα 3.26 Βέλτιστες Υπερπαράμετροι

Σε αυτή την εικόνα βλέπουμε το αποτέλεσμα του αλγορίθμου και τις βέλτιστες τιμές των φίλτρων και του ρυθμού μάθησης, δηλαδή τον καλύτερο συνδυασμό αυτών των υπερπαραμέτρων. Σε αυτή την περίπτωση το μόνο που άλλαξε είναι ο ρυθμός μάθησης σε 0.001.

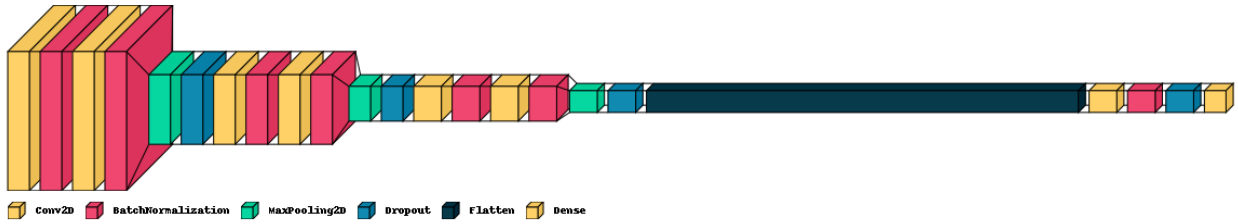
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 32, 32, 64)        1792

 batch_normalization_7 (Batc  (None, 32, 32, 64)       256
 hNormalization)

 conv2d_7 (Conv2D)           (None, 32, 32, 64)        36928

 batch_normalization_8 (Batc  (None, 32, 32, 64)       256
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 dropout_4 (Dropout)         (None, 16, 16, 64)        0

 conv2d_8 (Conv2D)           (None, 16, 16, 128)       73856

 batch_normalization_9 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 conv2d_9 (Conv2D)           (None, 16, 16, 128)       147584

 batch_normalization_10 (Bat  (None, 16, 16, 128)      512
 chNormalization)

 max_pooling2d_4 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 dropout_5 (Dropout)         (None, 8, 8, 128)         0

 conv2d_10 (Conv2D)          (None, 8, 8, 256)         295168

 batch_normalization_11 (Bat  (None, 8, 8, 256)        1024
 chNormalization)

 conv2d_11 (Conv2D)          (None, 8, 8, 256)         590080
```

Σχήμα 3.27 Πίνακας Επιπέδων Βέλτιστου Μοντέλου 1/2

```
batch_normalization_12 (Bat  (None, 8, 8, 256)        1024
chNormalization)

max_pooling2d_5 (MaxPooling  (None, 4, 4, 256)        0
2D)

dropout_6 (Dropout)          (None, 4, 4, 256)        0

flatten_1 (Flatten)          (None, 4096)             0

dense_2 (Dense)              (None, 256)              1048832

batch_normalization_13 (Bat  (None, 256)              1024
chNormalization)

dropout_7 (Dropout)          (None, 256)              0

dense_3 (Dense)              (None, 10)               2570

=================================================================
Total params: 2,201,418
Trainable params: 2,199,114
```

Σχήμα 3.28 Πίνακας Επιπέδων Βέλτιστου Μοντέλου 2/2

Και πλέον έχουμε το νέο μοντέλο, με τη νέα αρχιτεκτονική παρόλο που είναι το ίδιο.



Σχήμα 3.29 Αρχιτεκτονική Επιπέδων Μοντέλου

Τώρα που βρήκαμε τις βέλτιστες υπερπαράμετρους ήρθε η ώρα να εκπαιδεύσουμε το νέο μοντέλο με τις καινούριες, βέλτιστες υπερπαράμετρους.

```
print("Training the best model...")
best_model_history = best_model.fit(
    datagen.flow(x_train , y_train , batch_size = batch_size),
    epochs = epoch,
    validation_data = (x_val, y_val) ,
    callbacks=callbacks)
```

Σχήμα 3.30 Εκπαίδευση Βέλτιστου Μοντέλου

```
Epoch 20/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8812 - accuracy: 0.8181 - val_loss: 0.8821 - val_accuracy: 0.8147 - lr: 0.0010
Epoch 21/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8788 - accuracy: 0.8177 - val_loss: 0.7809 - val_accuracy: 0.8512 - lr: 0.0010
Epoch 22/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8719 - accuracy: 0.8188 - val_loss: 0.8846 - val_accuracy: 0.8210 - lr: 0.0010
Epoch 23/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8708 - accuracy: 0.8213 - val_loss: 0.8406 - val_accuracy: 0.8337 - lr: 0.0010
Epoch 24/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8634 - accuracy: 0.8248 - val_loss: 0.8816 - val_accuracy: 0.8226 - lr: 0.0010
Epoch 25/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8605 - accuracy: 0.8245 - val_loss: 0.8825 - val_accuracy: 0.8219 - lr: 0.0010
Epoch 26/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8517 - accuracy: 0.8268 - val_loss: 0.8896 - val_accuracy: 0.8197 - lr: 0.0010
Epoch 27/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8531 - accuracy: 0.8274 - val_loss: 0.8494 - val_accuracy: 0.8332 - lr: 0.0010
Epoch 28/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8492 - accuracy: 0.8292 - val_loss: 0.8035 - val_accuracy: 0.8451 - lr: 0.0010
Epoch 29/100
1248/1250 [============================>.] - ETA: 0s - loss: 0.8428 - accuracy: 0.8307
Epoch 29: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
1250/1250 [==============================] - 22s 18ms/step - loss: 0.8431 - accuracy: 0.8307 - val_loss: 0.8727 - val_accuracy: 0.8238 - lr: 0.0010
Epoch 30/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.7584 - accuracy: 0.8554 - val_loss: 0.7098 - val_accuracy: 0.8719 - lr: 1.0000e-04
Epoch 31/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.7141 - accuracy: 0.8694 - val_loss: 0.6821 - val_accuracy: 0.8782 - lr: 1.0000e-04
Epoch 32/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.6869 - accuracy: 0.8752 - val_loss: 0.6590 - val_accuracy: 0.8827 - lr: 1.0000e-04
Epoch 33/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.6747 - accuracy: 0.8743 - val_loss: 0.6486 - val_accuracy: 0.8847 - lr: 1.0000e-04
```

Σχήμα 3.31 Μείωση Ρυθμού Μάθησης κατά την Εκπαίδευση

Παρόλο που βρήκαμε το βέλτιστο ρυθμό μάθησης, συνεχίζουμε να χρησιμοποιούμε την τεχνική ReduceLROnPlateau, γιατί παρόλο που ο βέλτιστος ρυθμός μάθησης είναι ένα καλό αρχικό σημείο, η μείωσή του μπορεί να αποφέρει ακόμα καλύτερα αποτελέσματα.

```
Epoch 88/100
1249/1250 [============================>.] - ETA: 0s - loss: 0.3659 - accuracy: 0.9272
Epoch 88: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3660 - accuracy: 0.9272 - val_loss: 0.4592 - val_accuracy: 0.9055 - lr: 1.0000e-04
Epoch 89/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3516 - accuracy: 0.9329 - val_loss: 0.4469 - val_accuracy: 0.9097 - lr: 1.0000e-05
Epoch 90/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3461 - accuracy: 0.9349 - val_loss: 0.4455 - val_accuracy: 0.9098 - lr: 1.0000e-05
Epoch 91/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3401 - accuracy: 0.9364 - val_loss: 0.4406 - val_accuracy: 0.9116 - lr: 1.0000e-05
Epoch 92/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3459 - accuracy: 0.9343 - val_loss: 0.4416 - val_accuracy: 0.9101 - lr: 1.0000e-05
Epoch 93/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3383 - accuracy: 0.9372 - val_loss: 0.4416 - val_accuracy: 0.9109 - lr: 1.0000e-05
Epoch 94/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3416 - accuracy: 0.9369 - val_loss: 0.4417 - val_accuracy: 0.9103 - lr: 1.0000e-05
Epoch 95/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3388 - accuracy: 0.9363 - val_loss: 0.4403 - val_accuracy: 0.9120 - lr: 1.0000e-05
Epoch 96/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3386 - accuracy: 0.9364 - val_loss: 0.4446 - val_accuracy: 0.9104 - lr: 1.0000e-05
Epoch 97/100
1250/1250 [==============================] - 23s 18ms/step - loss: 0.3350 - accuracy: 0.9383 - val_loss: 0.4402 - val_accuracy: 0.9125 - lr: 1.0000e-05
Epoch 98/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3385 - accuracy: 0.9377 - val_loss: 0.4376 - val_accuracy: 0.9118 - lr: 1.0000e-05
Epoch 99/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3374 - accuracy: 0.9370 - val_loss: 0.4345 - val_accuracy: 0.9137 - lr: 1.0000e-05
Epoch 100/100
1250/1250 [==============================] - 22s 18ms/step - loss: 0.3323 - accuracy: 0.9374 - val_loss: 0.4340 - val_accuracy: 0.9138 - lr: 1.0000e-05
```

Σχήμα 3.32 Τέλος Εκπαίδευσης

Μέχρι το τέλος της εκπαίδευσης ο ρυθμός μάθησης πέφτει κι άλλο στο 0.00001.

3.8.7 Αποτελέσματα CNN

Τα μοντέλα εκπαιδεύτηκαν στο περιβάλλον Google Colab, λόγω έλλειψης καλής κάρτας γραφικών, άρα ο χρόνος εκπαίδευσης ήταν περιορισμένος μιας και υπάρχει χρονικό όριο στη χρήση κάρτας γραφικών από το Google Colab.

```
def save_results(history):
  # Plot training and validation accuracy.
  plt.figure(figsize=(10,6))
  sb.set_style("whitegrid")
  plt.plot(history.history['accuracy'],color="#E74C3C",marker='o')
  plt.plot(history.history['val_accuracy'],color='#641E16',marker='h')
  plt.title('Accuracy comparison between Validation and Train Data set',fontsize=15)
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Test'], loc='lower right')
  plt.savefig('/content/gdrive/My Drive/Image Classification/with_tuning/train_val_acc_2.png')
  plt.show()

  # Plot training and validation loss.
  plt.figure(figsize=(10,6))
  sb.set_style("whitegrid")
  plt.plot(history.history['loss'],color="Purple",marker='o')
  plt.plot(history.history['val_loss'],color='Orange',marker='h')
  plt.title('Loss comparison between Validation and Train Data set',fontsize=15)
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Test'], loc='best')
  plt.savefig('/content/gdrive/My Drive/Image Classification/with_tuning/train_val_loss_2.png')
  plt.show()
```

Σχήμα 3.33 Σχεδίαση Γραφικών

Στην παραπάνω εικόνα 3.33, έχουμε φτιάξει μια function για να μπορούμε να βγάζουμε τις γραφικές για training vs validation accuracy και training vs validation loss όποτε τη χρειαζόμαστε. Χρησιμοποιώντας τις βιβλιοθήκες matpotlib και seaborn ορίζουμε το μέγεθος του Σχήμα, το style, το title και τα labels και legend για κάθε γραφική και σχεδιάζουμε τις παραμέτρους που θέλουμε από το history και αποθηκεύουμε κάθε εικόνα στο google drive.

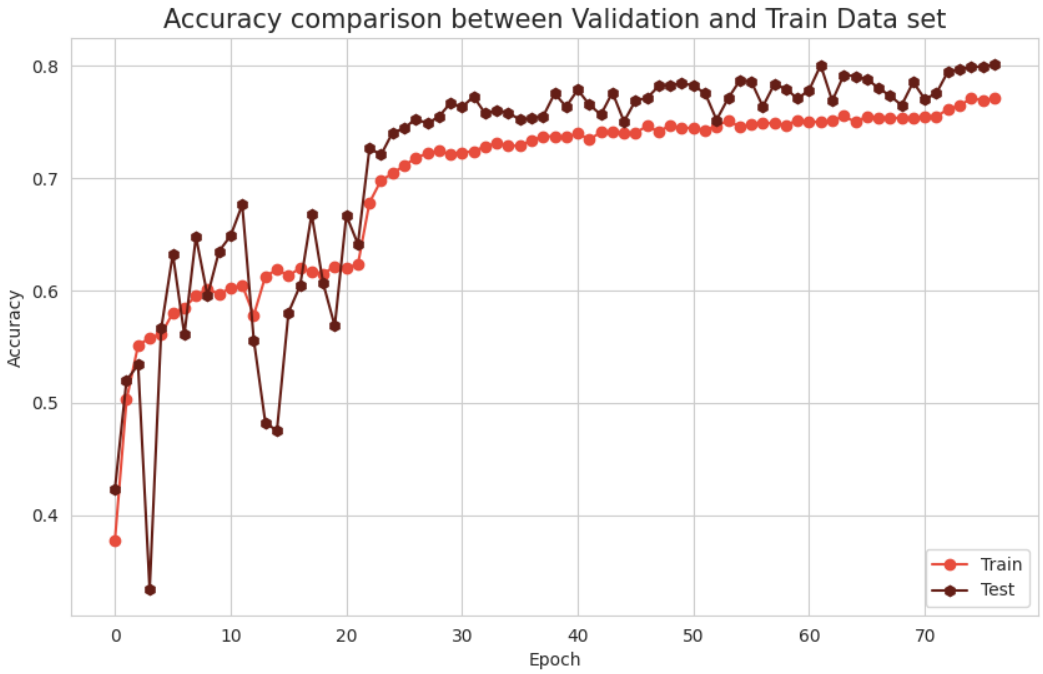
Στην πρώτη περίπτωση, χωρίς να κάνουμε hyperparameter tuning είχαμε τα εξής αποτελέσματα:

```
test_loss, test_acc = custom_model.evaluate(x_test, y_test, verbose=0)
print("Test Accuracy: %.2f%%" % (test_acc*100))
print("Test Loss: %.2f%%" % (test_loss*100))
```

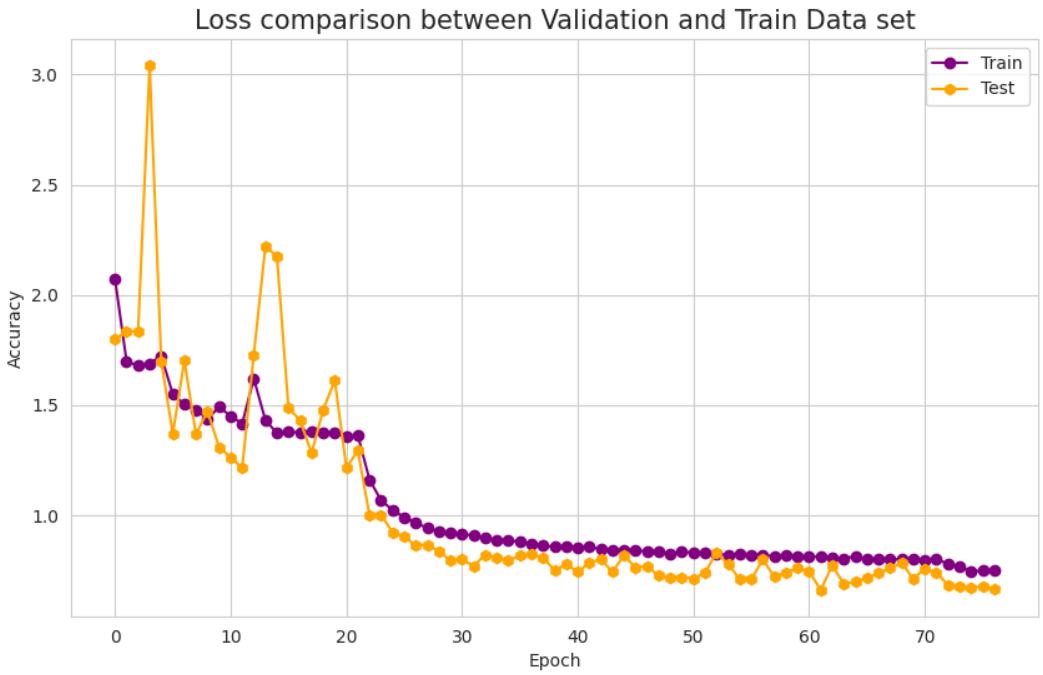
```
Test Accuracy: 79.48%
Test Loss: 69.02%
```

Σχήμα 3.34 Accuracy - Loss στο Test Set Μη Βέλτιστου Μοντέλου

Εδώ έχουμε accuracy στο test set 79,48%, δηλαδή προέβλεψε σωστά την κλάση της εικόνας το 79,48% των φορών, και test loss 69,02%, αν και συνήθως το test loss δεν παρουσιάζεται σαν ποσοστό, οπότε μπορούμε να πούμε ότι το test loss είναι 0.6902, και αυτό σημαίνει πόσο μακριά είναι η πρόβλεψη του μοντέλου από την πραγματική τιμή.



Σχήμα 3.35 Accuracy Validation - Train Set



Σχήμα 3.36 Loss Validation - Train Set

Βλέποντας και τις γραφικές που περιέχουν το train, validation accuracy και loss συμπεραίνουμε ότι δεν υπάρχει ούτε υπερπροσαρμογή αλλά ούτε υποπροσαρμογή, μιας και το accuracy των training και validation set δεν είναι πολύ υψηλό. Αυτό που μπορούμε να συμπεράνουμε από τα παραπάνω είναι ότι ναι μεν το μοντέλο γενικεύει σε δεδομένα που δεν έχει ξαναδεί, αλλά χρειαζόμαστε ένα πιο πολύπλοκο μοντέλο ή άλλες υπερπαραμέτρους για να γενικεύει καλύτερα.

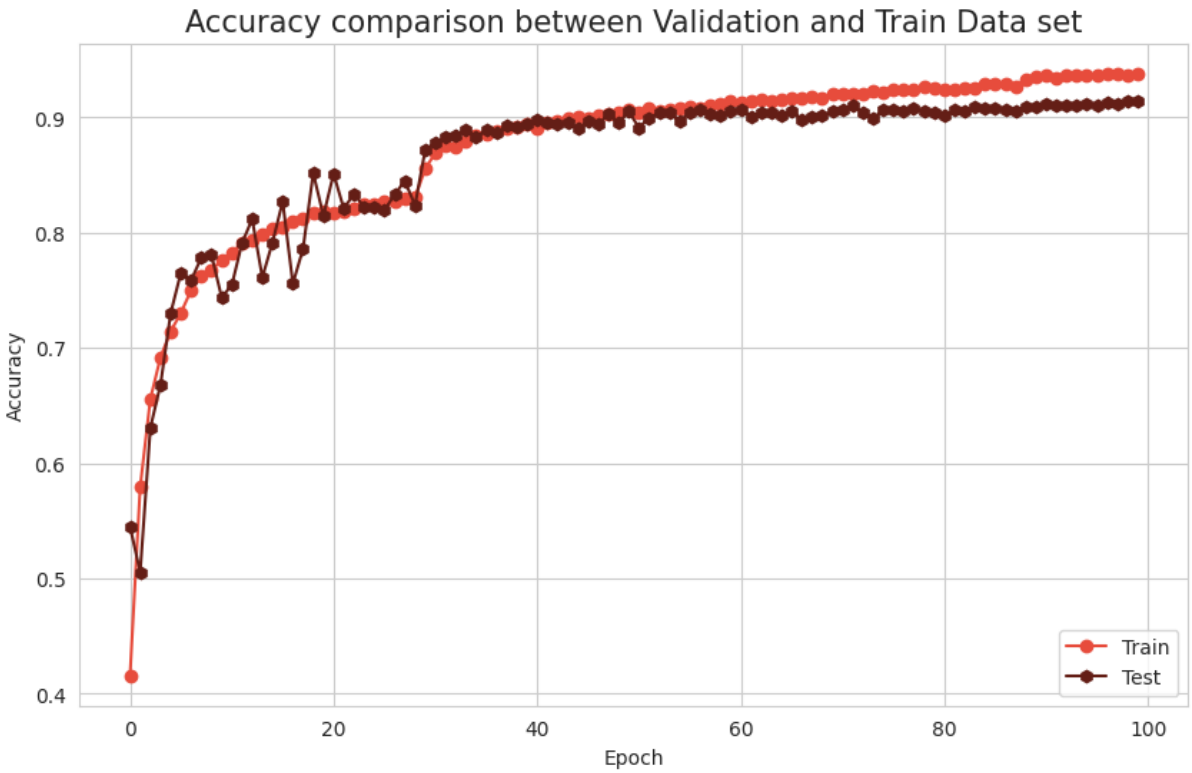
Στη δεύτερη περίπτωση που έχουμε hyperparameter tuning:

```
best_model_test_loss, best_model_test_acc = best_model.evaluate(x_test, y_test, verbose=0)
print("Test Accuracy: %.2f%%" % (best_model_test_acc*100))
print("Test Loss: %.2f%%" % (best_model_test_loss*100))
```

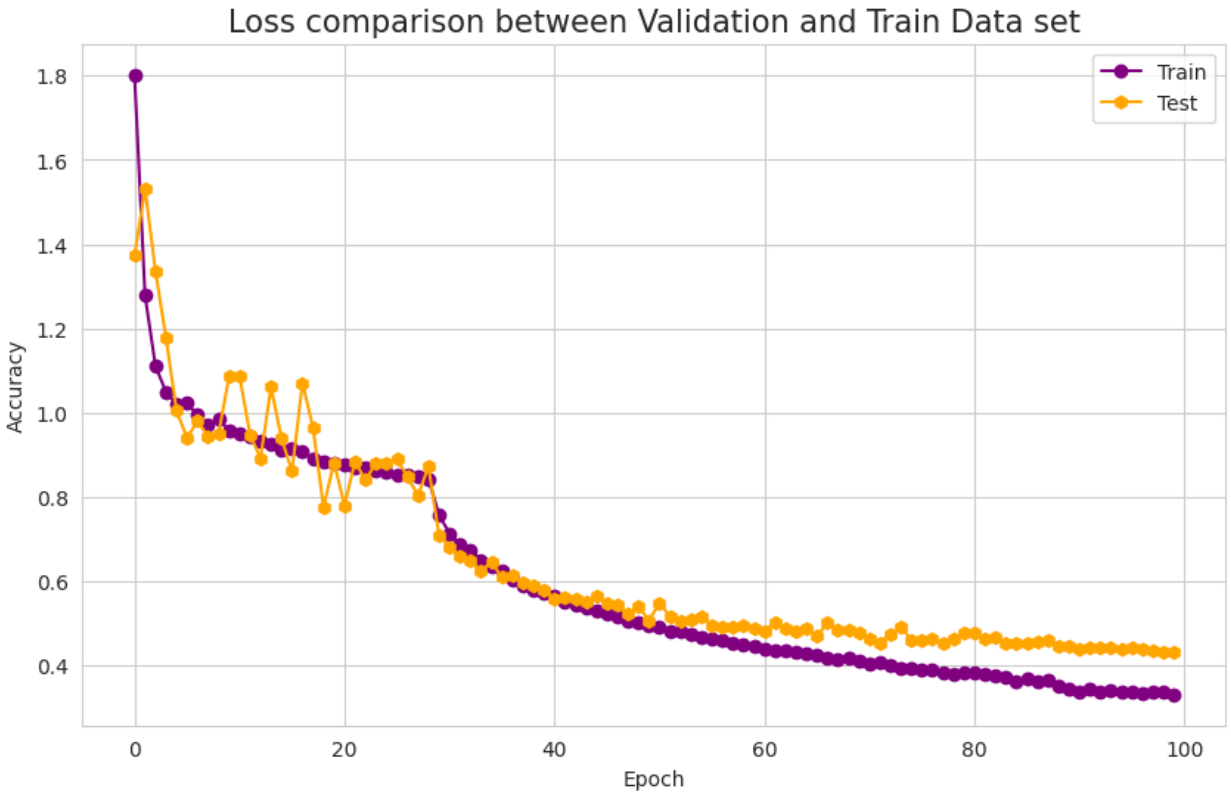
```
Test Accuracy: 91.12%
Test Loss: 44.39%
```

Σχήμα 3.37 Accuracy - Loss Βέλτιστου Μοντέλου

Μετά από hyperparameter tuning, όπως βλέπουμε και στην εικόνα, το accuracy στο test set ανέβηκε στο 91.12% και το σφάλμα έπεσε στο 0.4439. Απλά και μόνο με μία αλλαγή σε μία υπερπαράμετρο βλέπουμε μια αρκετά καλή διαφορά στο accuracy και loss.



Σχήμα 3.38 Accuracy Validation - Train Set



Σχήμα 3.39 Loss Validation - Train Set

Από τις γραφικές, βλέπουμε ότι το μοντέλο έχει ελάχιστα καλύτερη απόδοση στο training set από ότι στα validation και test sets και αυτό ίσως είναι ένα αρχικό σημάδι υπερπροσαρμογής. Παρόλα αυτά, η διαφορά δεν είναι τόσο μεγάλη, οπότε δεν μπορούμε να πούμε με σιγουριά ότι το υπάρχει υπερπροσαρμογή, μιας και αυτό συνήθως φαίνεται όταν υπάρχει πολύ μεγάλη διαφορά.

Πειραματιστήκαμε με λίγες τιμές στα φίλτρα και στο ρυθμό μάθησης, λόγω περιορισμών, καθώς και με ένα σχετικά μικρό dataset αλλά τα αποτελέσματα ήταν πολύ καλά. Για να είχαμε ακόμα καλύτερη γενίκευση του μοντέλου θα μπορούσαμε να προσθέταμε τεχνητές εικόνες με data augmentation και με περισσότερες διαφορετικές παραλλαγές, να αλλάζαμε κι άλλες υπερπαραμέτρους όπως το dropout ή το patience του early stopping ή ακόμα και με μεγαλύτερες τιμές στα φίλτρα και μικρότερο βήμα όταν ψάχνει για τις βέλτιστες τιμές.

Εδώ αξίζει να τονίσουμε ότι, το να πετύχουμε την τέλεια γενίκευση είναι αδύνατο οπότε μια τόσο μικρή διαφορά ανάμεσα στην απόδοση στα δεδομένα εκπαίδευσης και στα δεδομένα επικύρωσης και αξιολόγησης ήταν αναμενόμενη μιας και το μοντέλο μαθαίνει πάνω στα δεδομένα εκπαίδευσης και προσπαθεί να μειώσει το σφάλμα πάνω σε αυτά ενώ τα δεδομένα επικύρωσης και αξιολόγησης είναι δεδομένα που δεν έχει ξαναδεί.

4. Υλοποίηση της Εφαρμογής

4.1 Γενική Περιγραφή Εφαρμογής

Η υπό ανάπτυξη εφαρμογή είναι μία web εφαρμογή που αξιοποιεί τις δυνατότητες των συνελικτικών νευρωνικών δικτύων για την κατηγοριοποίηση των εικόνων που ανεβάζουν οι χρήστες και δημιουργήθηκε με το Flask Framework της γλώσσας προγραμματισμού Python. Οι χρήστες αλληλεπιδρούν με την εφαρμογή μέσω μιας διεπαφής (UI – User Interface) που τους επιτρέπει να ανεβάσουν εικόνες για ταξινόμηση. Όταν ο χρήστης ανεβάσει μια εικόνα, η εικόνα περνάει από ένα στάδιο επεξεργασίας από το εκπαιδευμένο cnn μοντέλο το οποίο θα μας επιστρέψει το αποτέλεσμα της ταξινόμησης σε μορφή ετικετών και των αντίστοιχων ποσοστών εμπιστοσύνης (confidence percentage) για κάθε ετικέτα.

Επίσης η εφαρμογή χρησιμοποιεί έναν server αποθήκευσης δεδομένων, το Minio, όπου βρίσκεται το εκπαιδευμένο μοντέλο και αποθηκεύονται οι εικόνες που ανεβάζουν οι χρήστες. Εκτός από το Minio, χρησιμοποιείται και μια μη σχεσιακή (NoSQL) βάση δεδομένων, MongoDB, για την αποθήκευση των αποτελεσμάτων της ταξινόμησης μαζί με τη διεύθυνση της αποθηκευμένης εικόνας στο Minio.

Για να εξασφαλίσουμε ότι η εφαρμογή θα δουλεύει σε όποιο μηχάνημα κι αν την βάλουμε, τρέχει μέσα σε ένα docker container, για να μην υπάρχουν προβλήματα συμβατότητας συστήματος.

Παρακάτω θα δούμε την υλοποίηση της web εφαρμογής και πως χρησιμοποιεί το εκπαιδευμένο μοντέλο που είδαμε στο προηγούμενο κεφάλαιο, καθώς και τα εργαλεία που χρησιμοποιήσαμε.

4.2 Σκοπός της Εφαρμογής

Σκοπός της web εφαρμογής ταξινόμησης εικόνων είναι να παρέχει στους χρήστες μια πλατφόρμα ώστε να χρησιμοποιούν εύκολα μοντέλα τεχνητής νοημοσύνης για ταξινόμηση εικόνων αξιοποιώντας συνελικτικά νευρωνικά δίκτυα και μια φιλική προς το χρήστη διεπαφή. Έτσι, η εφαρμογή εκδημοκρατίζει την πρόσβαση την πρόσβαση σε τέτοιου είδους μοντέλα τεχνητής νοημοσύνης.

4.3 Τεχνολογίες που χρησιμοποιήθηκαν

4.3.1 Flask

Το κύριο μέρος της εφαρμογής είναι γραμμένο γλώσσα προγραμματισμού Python. Το Flask είναι ένα web framework, δηλαδή μια συλλογή από βιβλιοθήκες, που μας βοηθάει να φτιάχνουμε γρήγορα και εύκολα web εφαρμογές χωρίς να μας ενδιαφέρουν λεπτομέρειες όπως τα διάφορα πρωτόκολλα.

Εδώ το χρησιμοποιούμε για να χτίσουμε τη διεπαφή προγραμματισμού εφαρμογών (API – Application Programming Interface) για να συνδέσουμε τη διεπαφή χρήστη με τα υπόλοιπα τμήματα της εφαρμογής. Ένα API είναι ένα σύνολο κανόνων και πρωτοκόλλων που καθορίζει τον τρόπο με τον οποίο διαφορετικά τμήματα της εφαρμογής αλληλεπιδρούν μεταξύ τους. Επίσης ορίζει τα είδη των αιτημάτων που μπορούν να γίνουν, τον τρόπο υποβολής τους και τις μορφές των δεδομένων που πρέπει να χρησιμοποιηθούν και τι συμβάσεις να ακολουθήσουν.

Στην υπό ανάπτυξη εφαρμογή το Flask Framework μας βοηθάει στο να φτιάξουμε API endpoints, δηλαδή συγκεκριμένα URLs όπου η εφαρμογή μπορεί να λάβει HTTP αιτήματα (requests), όπως GET, POST, PUT, DELETE, και να γυρνάει μια απάντηση. Με αυτόν τον τρόπο μπορούμε μέσω ενός browser να αλληλεπιδράσουμε με την web εφαρμογή μας, όπως να ανεβάζουμε εικόνες.

4.3.2 HTML/CSS

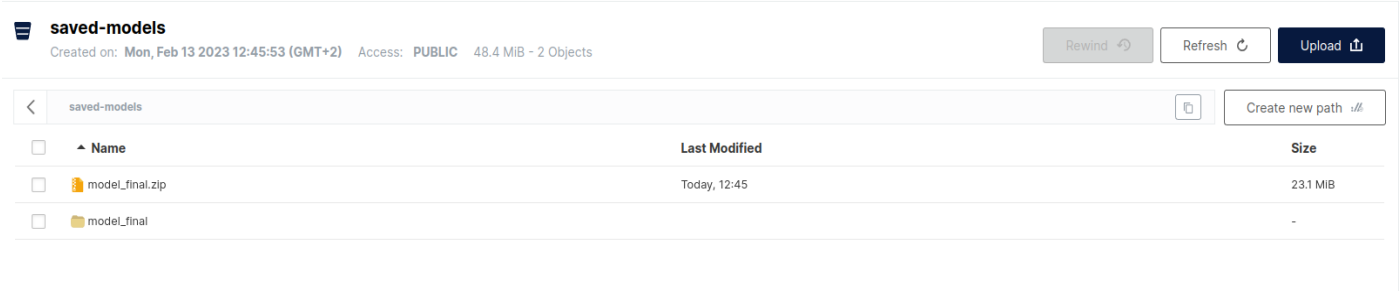
HTML (HyperText Markup Language) είναι η γλώσσα που περιγράφει τη δομή μιας ιστοσελίδας, όπως παραγράφους, εικόνες, συνδέσμους, κουμπιά κ.α, και CSS (Cascading Style Sheets) είναι η γλώσσα που περιγράφει το παρουσιαστικό και τη μορφοποίηση μιας ιστοσελίδας όπως χρώματα, γραμματοσειρές κ.α. Επίσης ελέγχει πως τα HTML στοιχεία απεικονίζονται στη οθόνη.

Αυτά τα εργαλεία τα χρησιμοποιούμε για να κατασκευάσουμε τη διεπαφή χρήστη (UI – User Interface) για να μπορεί ο χρήστης να ανεβάζει εικόνες και να βλέπει τα αποτελέσματα.

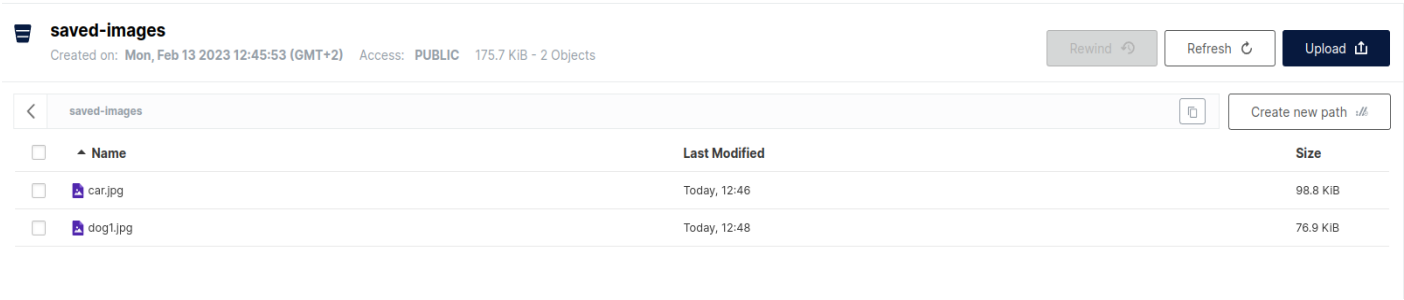
4.3.3 Minio

Το Minio είναι ένας server ανοιχτού κώδικα για αποθήκευση αρχείων και σχεδιάστηκε για να παρέχει τις ίδιες δυνατότητες και λειτουργίες με αντίστοιχους σε cloud περιβάλλον που είναι επί πληρωμή. Χρησιμοποιείται κυρίως για αποθήκευση μη δομημένων δεδομένων, όπως εικόνες, βίντεο, συμπιεσμένα αρχεία zip κ.α. και μπορεί να χειρίζεται μεγάλο όγκο δεδομένων.

Σε αυτήν την εφαρμογή την χρησιμοποιούμε για να αποθηκεύουμε τις εικόνες που ανεβάζει ο χρήστης και για να έχουμε αποθηκευμένα τα εκπαιδευμένα μοντέλα μας.



Σχήμα 4.1 Αποθήκευση μοντέλων στο Minio



Σχήμα 4.2 Αποθήκευση εικόνων στο Minio

Είναι πολύ χρήσιμο να υπάρχουν οι εικόνες αποθηκευμένες σε ένα κεντρικό σημείο έτσι ώστε να μπορούμε αν θέλουμε να βάλουμε λειτουργία αναζήτησης έτσι ώστε να μπορεί ο χρήστης να δει τι έχει κάνει στο παρελθόν.

Τα μοντέλα από την άλλη, θέλουμε να τα έχουμε αποθηκευμένα έτσι ώστε να μπορεί η εφαρμογή να τα χρησιμοποιήσει. Μπορούμε να έχουμε διαφορετικές εκδόσεις των μοντέλων και να χρησιμοποιούμε την τελευταία έκδοση και σε περίπτωση που πάει κάτι στραβά να έχουμε μια προηγούμενη έκδοση έτσι σαν backup.

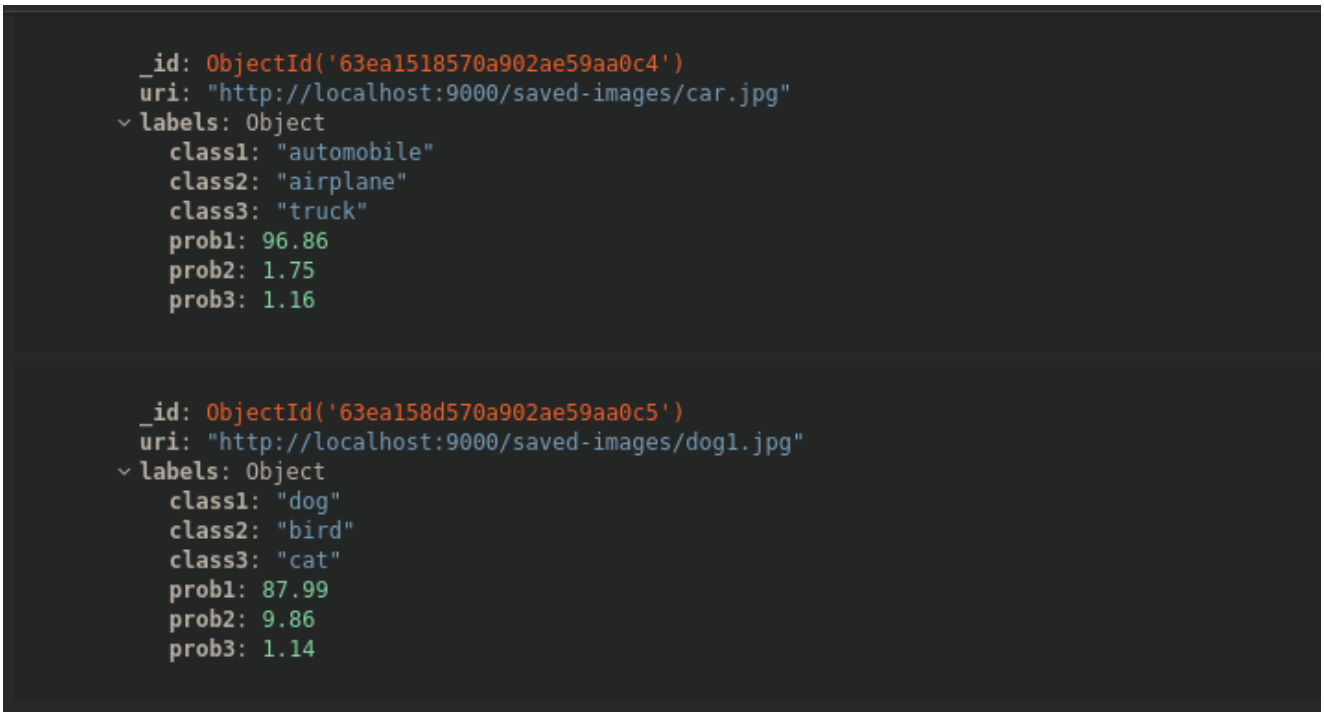
4.3.4 MongoDB

Η MongoDB είναι μια μη σχεσιακή (NoSQL) βάση δεδομένων που χρησιμοποιείται κυρίως για την αποθήκευση ημι-δομημένων δεδομένων τύπου JSON και συνεπώς δεν έχει συγκεκριμένο

σχήμα, σε αντίθεση με της σχεσιακές βάσεις που έχουν συγκεκριμένο σχήμα που μοιάζει με πίνακα για την αποθήκευση δεδομένων.

Η MongoDB οργανώνει τα δεδομένα ως «έγγραφα», σύνολα ζευγών κλειδιών – τιμών. Αυτά τα έγγραφα ομαδοποιούνται σε συλλογές, που είναι αντίστοιχα με τους πίνακες (tables) σε μια σχεσιακή βάση δεδομένων.

Σε αυτήν την εφαρμογή τη χρησιμοποιούμε για να αποθηκεύσουμε τη διεύθυνση μιας εικόνας αποθηκευμένης στο Minio μαζί με τις ετικέτες και τις πιθανότητες μετά από την ταξινόμηση, στο ίδιο «αντικείμενο» στη βάση.



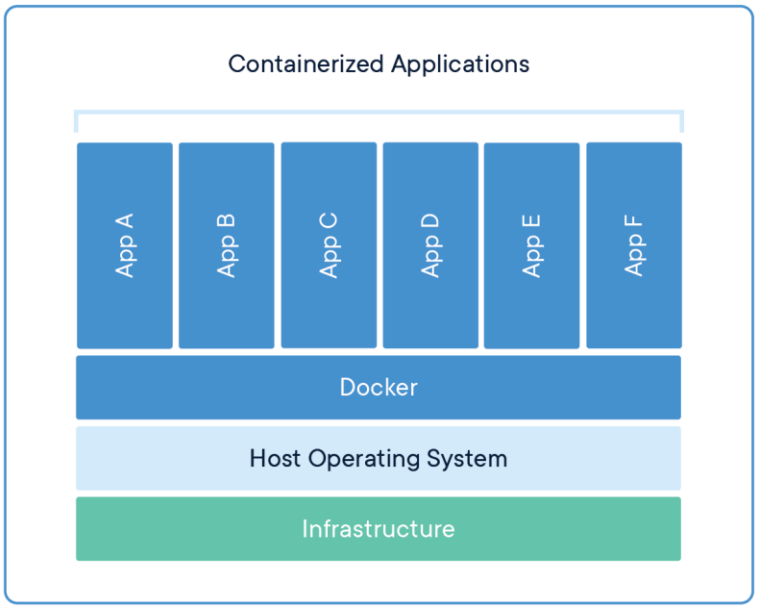
Σχήμα 4.3 MongoDB

Η βάση δεδομένων χρησιμοποιήθηκε για την αποθήκευση του URL της εικόνας στο minio μαζί με τις ετικέτες και τις πιθανότητες της ταξινόμησης για μελλοντική χρήση, πχ για την αναζήτηση. Στην παραπάνω εικόνα βλέπουμε πως είναι αποθηκευμένα στη βάση.

4.3.5 Docker – Docker Compose

Το docker είναι ένα εργαλείο, το οποίο μας επιτρέπει να χτίζουμε κάποια περιβάλλοντα που ονομάζονται containers, και να «τρέχει» η εφαρμογή μας μέσα σε αυτά και να τα διαχειρίζεται, όπως για παράδειγμα να τα ξεκινάει, να τα σταματάει κ.α. Ένα από τα μεγαλύτερα πλεονεκτήματα

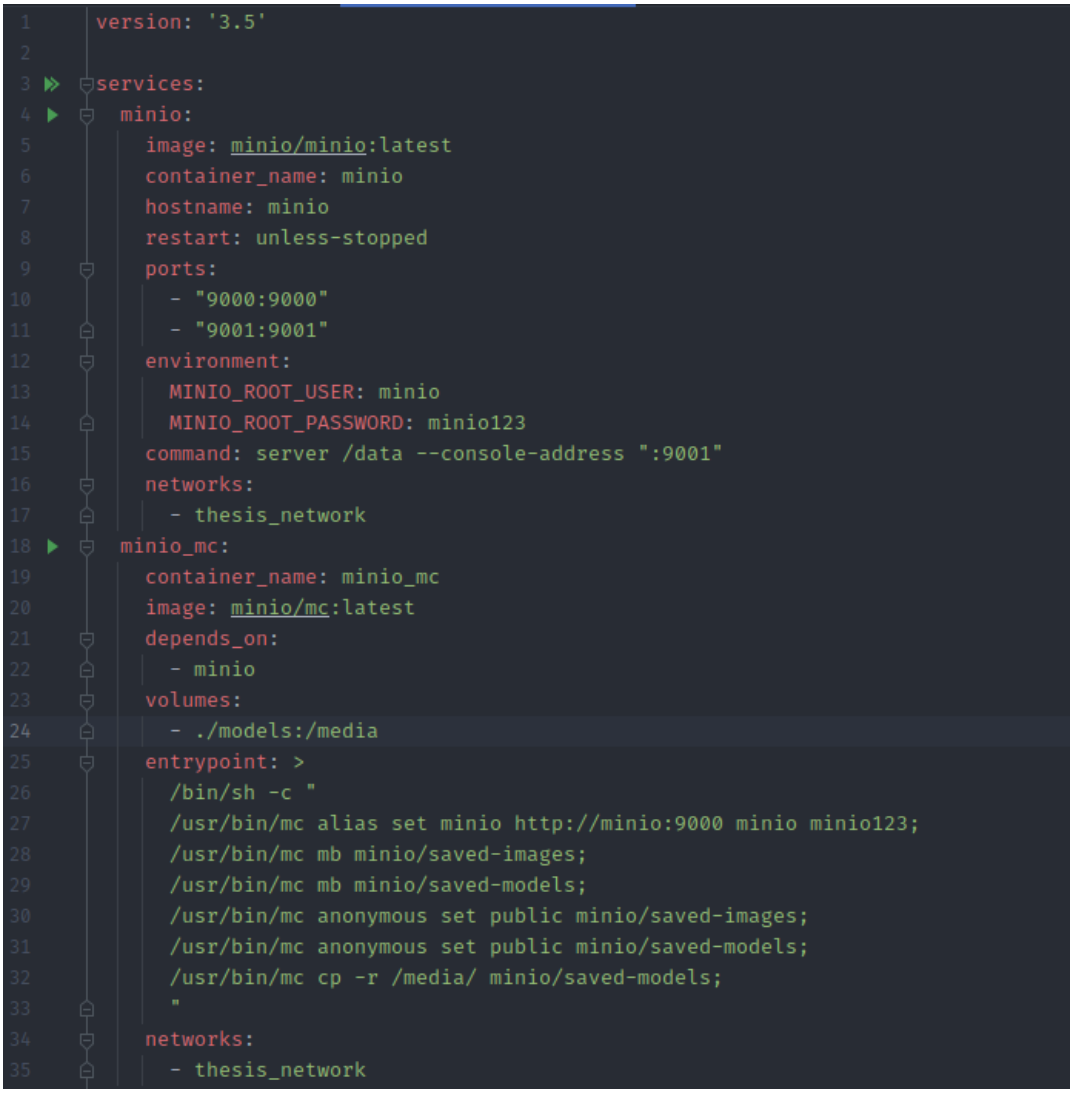
είναι ότι δεν χρειάζεται σε κάθε υπολογιστή ή server που θα βάλουμε μια εφαρμογή να «τρέχει» να εγκαταστήσουμε τη γλώσσα προγραμματισμού που είναι γραμμένη η εφαρμογή ή όλες τις βιβλιοθήκες και οτιδήποτε χρειάζεται μια εφαρμογή για να χρησιμοποιηθεί, μιας και όλα όσα χρειάζεται είναι μέσα στο container. Έτσι, μας βοηθάει επίσης όταν ένα άλλο μέλος της ομάδας που θέλει να προσθέσει καινούργια πράγματα στην εφαρμογή δεν θα χρειάζεται να φάει χρόνο για να εγκαταστήσει στο μηχάνημά του, πχ στο λάπτοπ του, οτιδήποτε δεν θα χρειαστεί μετά, αλλά και δεν παίζει και ιδιαίτερο ρόλο αν το λειτουργικό του σύστημα είναι διαφορετικό από κάποιου άλλου. Το μόνο που θα χρειαστεί είναι να εγκαταστήσει το docker και είναι έτοιμος. Για να δημιουργήσουμε ένα docker container το μόνο που χρειάζεται να κάνουμε είναι να φτιάξουμε ένα αρχείο με το όνομα Dockerfile και να γράψουμε μέσα την «εικόνα» που θέλουμε να φορτώσει, «εικόνα» μπορεί να είναι ένα λειτουργικό linux ή η «εικόνα» μιας γλώσσας προγραμματισμού, και μετά κάποιες εντολές για να αντιγράψουμε την εφαρμογή μας μέσα στο container ή/και εντολές για να εγκαταστήσουμε βιβλιοθήκες που χρειαζόμαστε μέσα στο container.



Σχήμα 4.4 Docker Container

Όπως βλέπουμε και στην παραπάνω εικόνα, μπορούμε να έχουμε πολλαπλά docker containers, τα οποία θα περιέχουν ένα διαφορετικό κομμάτι της συνολικής εφαρμογής, όπως για παράδειγμα το Minio ή η βάση δεδομένων μπορούν να είναι σε δικό τους ξεχωριστό container το κάθε ένα.

Το Docker Compose από την άλλη είναι ένα εργαλείο, που μας επιτρέπει να διαχειριζόμαστε πολλαπλά docker containers ταυτόχρονα. Σε πιο μεγάλες εφαρμογές που χρησιμοποιούν πολλά διαφορετικά containers όπως εμείς, πχ ένα για την βάση δεδομένων, ένα για το Minio, ένα άλλο για την εφαρμογή κλπ., μπορούμε να έχουμε ένα κεντρικό αρχείο, το docker-compose.yml, στο οποίο να έχουμε όλα τα διαφορετικά containers που θέλουμε να διαχειριστεί. Αυτό είναι αρκετά χρήσιμο γιατί με μία μόνο εντολή μπορούμε να χτίσουμε ή να σταματήσουμε όλα μας τα containers.



```yaml
version: '3.5'

services:
  minio:
    image: minio/minio:latest
    container_name: minio
    hostname: minio
    restart: unless-stopped
    ports:
      - "9000:9000"
      - "9001:9001"
    environment:
      MINIO_ROOT_USER: minio
      MINIO_ROOT_PASSWORD: minio123
    command: server /data --console-address ":9001"
    networks:
      - thesis_network
  minio_mc:
    container_name: minio_mc
    image: minio/mc:latest
    depends_on:
      - minio
    volumes:
      - ./models:/media
    entrypoint: >
      /bin/sh -c "
      /usr/bin/mc alias set minio http://minio:9000 minio minio123;
      /usr/bin/mc mb minio/saved-images;
      /usr/bin/mc mb minio/saved-models;
      /usr/bin/mc anonymous set public minio/saved-images;
      /usr/bin/mc anonymous set public minio/saved-models;
      /usr/bin/mc cp -r /media/ minio/saved-models;
      "
    networks:
      - thesis_network
```

Σχήμα 4.5 Docker Compose YAML File ½

```yaml
  flask_app:
    container_name: flask_app
    restart: unless-stopped
    build:
      context: ./api
      dockerfile: Dockerfile
    environment:
       FLASK_ENV: dev
       FLASK_APP: main.py
       FLASK_RUN_HOST: 0.0.0.0
    ports:
      - "5000:5000"
    volumes:
      - ./api:/api
    depends_on:
      - mongodb
      - minio_client
      - minio
    networks:
      - thesis_network
  mongodb:
    image: mongo:latest
    container_name: mongodb
    hostname: mongodb
    restart: unless-stopped
    environment:
      MONGO_INITDB_ROOT_USERNAME: root_user
      MONGO_INITDB_ROOT_PASSWORD: root_pwd
      MONGO_DB_NAME: image_data
    ports:
      - "27017:27017"
    networks:
      - thesis_network

networks:
  thesis_network:
    name: thesis_app_network
```

Σχήμα 4.6 Docker Compose YAML File 2/2

Όπως βλέπουμε και στις παραπάνω εικόνες, το docker-compose της εφαρμογής μας διαχειρίζεται 4 containers. Το minio, τη βάση και την εφαρμογή. Το δεύτερο container, minio mc, χρησιμοποιείται μόνο την πρώτη φορά που στήνουμε ολόκληρο το project σε ένα μηχάνημα, και το μόνο που κάνει είναι να φτιάχνει τους «φακέλους» που θα αποθηκεύονται οι εικόνες και το

εκπαιδευμένο μοντέλο, καθώς και να αντιγράφει το μοντέλο στο αντίστοιχο φάκελο. Μόλις κάνει την δουλειά του σταματάει να δουλεύει μιας και δεν είναι χρήσιμο πλέον.

Δομή του docker-compose:

- Version: Αυτή είναι η έκδοση του docker compose.
- Services: Τα services είναι τα ξεχωριστά docker containers που διαχειρίζεται το docker compose, εμείς εδώ έχουμε τα minio, flask app και mongodb καθώς και ένα βοηθητικό το minio mc.
- Image: Στο image βάζουμε το έτοιμο image που θέλουμε να χρησιμοποιήσουμε και βρίσκεται στο docker hub, μια πλατφόρμα που μπορεί ο οποιοσδήποτε να φτιάχνει και να ανεβάζει ένα image για ένα εργαλείο ή μια γλώσσα προγραμματισμού και να μπορεί να την χρησιμοποιεί όποιος θέλει.
- Container_name: Εδώ δηλώνουμε το όνομα που θέλουμε να έχει το συγκεκριμένο container. Με το να δηλώσουμε ένα όνομα για το κάθε container μας βοηθάει να αναγνωρίζουμε ποιο container περιέχει τι, πχ βάση δεδομένων, να τα χρησιμοποιούμε στις εντολές αντί για το ID του container που είναι πιο δύσκολο να το θυμόμαστε και τέλος το όνομα του container λειτουργεί ως hostname που επιτρέπει σε άλλα containers στο ίδιο docker δίκτυο να επικοινωνούν μεταξύ τους με τα ονόματά τους.
- Hostname: Το hostname είναι άλλο ένα αναγνωριστικό για τα containers για να μπορούν να επικοινωνούν μεταξύ τους αν βρίσκονται στο ίδιο δίκτυο και είναι αρκετά χρήσιμο για εσωτερική χρήση μεταξύ εφαρμογών ή όταν μπαίνει σε κάποιο url.
- Restart: Το restart δηλώνει κυρίως τι θα γίνει σε περίπτωση που κλείσει το container, για παράδειγμα η επιλογή unless-stopped που χρησιμοποιούμε σημαίνει ότι το container ότι και να γίνει θα ξεκινάει πάλι εκτός αν το σταματήσουμε εμείς συγκεκριμένα.
- Ports: Τα ports είναι οι πόρτες δικτύου που χρησιμοποιεί κάθε container για να επικοινωνεί με τα υπόλοιπα containers.
- Environment: Εδώ δηλώνονται παράμετροι που θέλουμε να περάσουμε μέσα στο container όπως ένα συνθηματικό χρήστη.
- Command / Entrypoint: Εδώ δηλώνεται κάποια εντολή που θέλουμε να τρέξει στο container.
- Networks: Το network είναι το docker network που υπάγονται τα containers για να μπορούν να επικοινωνούν μεταξύ τους.
- Depend_on: Αυτή η επιλογή δηλώνει ότι πρέπει να περιμένει ένα ή περισσότερα συγκεκριμένα containers πριν ξεκινήσει αυτό με αυτή την επιλογή.
- Volumes: Τα volumes χρησιμοποιούνται για να έχει πρόσβαση το container σε κάποιον φάκελο που είναι εκτός αυτού, είναι σαν μια σύνδεση ενός φακέλου στον σκληρό δίσκο του μηχανήματος με το container για να έχει πρόσβαση να διαβάζει ή να γράφει αρχεία το container.

- Build: Το build χρησιμοποιείται όταν δεν παίρνουμε έτοιμη docker εικόνα αλλά χτίζουμε εμείς μία δικιά μας. Αυτό το καταφέρνουμε δημιουργώντας ένα αρχείο που λέγεται Dockerfile.

Σχήμα 4.7 Dockerfile

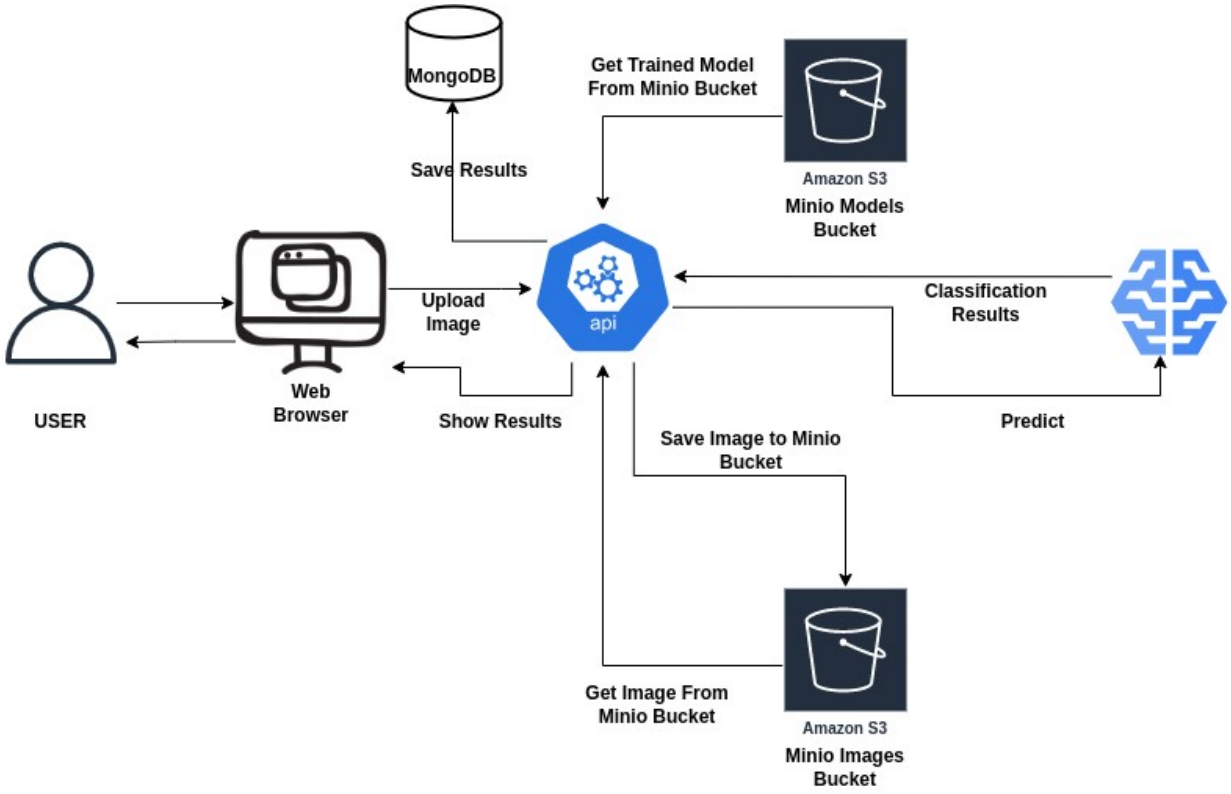
Όπως βλέπουμε και στην εικόνα 4.7 από πάνω, αυτό είναι το Dockerfile. Το Dockerfile το χρειαζόμαστε όταν θέλουμε να χτίσουμε τη δικιά μας docker εικόνα από την αρχή. Εδώ παίρνουμε την έτοιμη εικόνα python:3.10, δηλαδή το docker image της γλώσσας python με έκδοση 3.10 για να φτιάξουμε την δικιά μας εικόνα. Στη συνέχεια ορίζουμε σαν working directory τον φάκελο /api, αντιγράφουμε το αρχείο requirements.txt μέσα σε αυτόν τον φάκελο, τρέχουμε την εντολή να εγκαταστήσει τις βιβλιοθήκες από το αρχείο requirements.txt και αντιγράφουμε και τα υπόλοιπα αρχεία της εφαρμογής στον φάκελο api. Τέλος ορίζουμε σαν entrypoint του container το flask run για να ξεκινήσει η web εφαρμογή μας με το που ξεκινήσει να λειτουργεί το container.

4.3.6 Version Control – GIT

Version control είναι μια πρακτική για παρακολούθηση και διαχείριση των αλλαγών του κώδικα. Φτιάχτηκε για να υπάρχει σε ένα σημείο όλος ο κώδικας, για παράδειγμα μιας εφαρμογής, έτσι ώστε όλα τα μέλη μιας ομάδας να έχουν πρόσβαση τόσο στον κώδικα όσο και στις προσθήκες που γίνονται από όλα τα μέλη αλλά και να ειδοποιηθούν σε περίπτωση που μια προσθήκη ενός μέλους της ομάδας συγκρούεται με μια προσθήκη ή αλλαγή κάποιου άλλου. Επίσης ακόμα και σε ομάδα ενός ατόμου, βοηθάει σε περίπτωση που κάποιος θέλει να αλλάξει υπολογιστή, και έτσι δεν χρειάζεται να παίρνει μαζί του τον κώδικα σε εξωτερικούς σκληρούς ή οτιδήποτε, αλλά και για ασφάλεια σε περίπτωση που χαλάσει ο υπολογιστής κάποιου δεν χάνεται ο κώδικας. Στην

παρούσα εφαρμογή χρησιμοποιήθηκε το Git και για την αποθήκευση του κώδικα η πλατφόρμα GitHub. Στο GitHub δημιουργήθηκε ένα repository για το project και κάθε προσθήκη και αλλαγή ανέβαινε εκεί.

4.4 Ανάλυση Κώδικα Εφαρμογής



Σχήμα 4.8 Διάγραμμα της Εφαρμογής

4.4.1 Backend

```python
conn_str = "mongodb://root_user:root_pwd@mongodb:27017/?authMechanism=DEFAULT"

classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

ALLOWED_EXTENSIONS = ["png", "jpg", "jpeg", "gif"]

1 usage  Brachos Konstantinos
def allowed_file(filename: str) -> bool:
    """Returns true if file name contains allowed extension and false otherwise
    Parameters
    ----------
    filename : str
        the filename along with the extension"""
    return "." in filename and filename.rsplit(".", 1)[1].lower() in ALLOWED_EXTENSIONS
```

Σχήμα 4.9 config.py

Στο αρχείο config.py δηλώνουμε στατικά πράγματα όπως είναι το connection_string για τη βάση δεδομένων που χρησιμοποιούμε και δηλώνει πως θα συνδεθεί η εφαρμογή στη βάση δεδομένων. Δηλώνουμε επίσης τις classes του dataset και τους τύπους των αρχείων εικόνας που μπορούμε να ανεβάσουμε, όπως png, jpg, jpeg και gif.

Επίσης έχουμε και μια function που ελέγχει αν η εικόνα που ο χρήστης ανέβασε ανήκει στους παραπάνω τύπους εικόνας.

```python
import io
import os
import shutil
import zipfile
import boto3
from botocore.client import ClientError
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import load_model

s3_resource = boto3.resource(
    "s3",
    endpoint_url="http://minio:9000",
    aws_access_key_id="minio",
    aws_secret_access_key="minio123"
)

s3 = boto3.client("s3",
                  endpoint_url="http://minio:9000",
                  aws_access_key_id="minio",
                  aws_secret_access_key="minio123")


1 usage  Brachos Konstantinos
def upload_img_to_s3(img, filename, content_type):
    try:
        s3.put_object(Body=img,
                      Bucket="saved-images",
                      Key=filename,
                      ContentType=content_type)
    except ClientError as e:
        print("Something Happened: ", e)
        return e
    return "http://localhost:9000/saved-images/" + filename
```

Σχήμα 4.10 s3_ops.py 1/2

Εδώ έχουμε το αρχείο s3_ops.py. Σε αυτό το αρχείο έχουμε ότι κώδικα χρειάζεται σχετικά με την επικοινωνία με το minio όπου ορίζουμε τα resources και τον client που έχουν το url του minio και τα username και password.

Η function upload_img_to_s3 παίρνει σαν παραμέτρους την εικόνα, το όνομά της και τον τύπο της εικόνας και χρησιμοποιεί τον client για να ανεβάσει την εικόνα στον φάκελο saved-images.

```
1 usage  Brachos Konstantinos
def image_from_s3(filename: str):
    bucket = s3_resource.Bucket("saved-images")
    image = bucket.Object(filename)
    image = load_img(io.BytesIO(image.get()['Body'].read()), target_size=(32, 3
    image = img_to_array(image)
    image = image.reshape(1, 32, 32, 3)
    image = image.astype('float32')
    image = image / 255.0
    return image

1 usage  Brachos Konstantinos
def load_model_from_s3(model_name: str):
    bucket = s3_resource.Bucket("saved-models")
    # Fetch and save the zip file to the temporary directory
    bucket.download_file(f"{model_name}.zip", f"{model_name}.zip")
    # Extract the model zip file within the temporary directory
    with zipfile.ZipFile(f"{model_name}.zip") as zip_ref:
        zip_ref.extractall(f"{model_name}")
    # Load the keras model from the temporary directory
    model =  load_model(os.path.join("model_final/model_final", "custom_cnn_fin
    if os.path.exists(f"{model_name}.zip") and os.path.exists(f"{model_name}"):
        os.remove(f"{model_name}.zip")
        shutil.rmtree(f"{model_name}")
    return model
```

Σχήμα 4.11 s3_ops.py 2/2

Στην εικόνα 4.11 βλέπουμε 2 functions που αφορούν το minio. Η πρώτη, image_from_s3, χρησιμοποιείται για να πάρουμε την εικόνα που θέλουμε με το όνομά της από το minio, γίνεται επεξεργασία της εικόνας, όπως reshape και normalization για να έρθει στη μορφή που χρειάζεται για να τη χρησιμοποιήσει το μοντέλο και να βγάλει τα αποτελέσματα της ταξινόμησης.

Η δεύτερη function, load_model_from_s3, χρησιμοποιείται για να φέρει το εκπαιδευμένο μοντέλο από το minio που είναι σε μορφή zip, να το κάνει unzip έτσι ώστε να μπορέσει να χρησιμοποιηθεί από την εφαρμογή για τις εικόνες που ανεβάζουν οι χρήστες.

```python
from typing import Any

2 usages (1 dynamic)  Brachos Konstantinos
def predict(img, model) -> tuple[list[str], list[Any]]:
    result = model.predict(img)
    classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
    dict_result = {}
    for i in range(10):
        dict_result[result[0][i]] = classes[i]

    res = result[0]
    res.sort()
    res = res[::-1]
    prob = res[:3]

    prob_result = []
    class_result = []
    for i in range(3):
        prob_result.append((prob[i] * 100).round(2))
        class_result.append(dict_result[prob[i]])

    return class_result, prob_result
```

Σχήμα 4.12 predict.py

Στο αρχείο predict.py έχουμε την function που κάνει το prediction. Παίρνει σαν παράμετρο την εικόνα και το μοντέλο, και χρησιμοποιώντας την predict function του μοντέλου αποθηκεύει τα αποτελέσματα στη μεταβλητή results και στη συνέχεια αποκωδικοποιεί τα αποτελέσματα για να τα φέρει στη μορφή που θέλουμε για να τα δείξουμε πίσω στο χρήστη. Συγκεκριμένα φτιάχνει 2 λίστες, μια για τις ετικέτες και μια για τις πιθανότητες τους επί τις εκατό και επιστρέφει αυτές τις 2 λίστες.

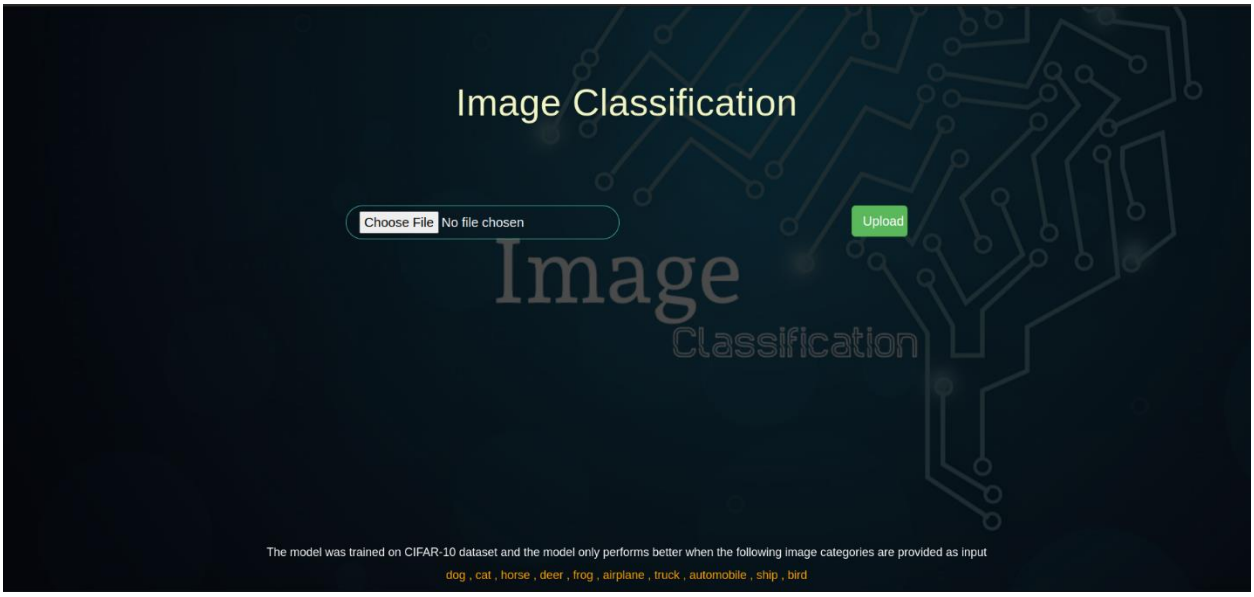
```python
import io
import requests
from flask import request, jsonify, send_file, render_template
from pymongo import MongoClient
from werkzeug.utils import secure_filename
from api import api, s3_ops, config, predict


client = MongoClient(config.conn_str)
collection = client.image_data.predictions


Brachos Konstantinos
@api.route("/alive", methods=["GET"])
def alive():
    """Liveliness Endpoint. Can be used as a probe."""
    return jsonify({"message": "I'm alive", "status": 200})


operatingfuture
@api.route("/", methods=["GET"])
def home():
    """Homepage"""
    return render_template("index.html")
```

Σχήμα 4.13 main.py

Το αρχείο main.py είναι το βασικό αρχείο της εφαρμογής μας, εδώ έχουμε το endpoint της εφαρμογής που μεταξύ άλλων επιτρέπει το ανέβασμα της εικόνας. Πριν από τα endpoints όμως ορίζουμε τον client της βάσης δεδομένων και το collection της βάσης, στο οποίο θέλουμε να αποθηκεύσουμε τα δεδομένα μας.

Στην αρχή όμως έχουμε ένα απλό endpoint, το alive, που το μόνο που κάνει είναι να επιστρέφει ένα μήνυμα ότι είναι ζωντανό κάθε φορά που το καλεί κάποιος μόνο και μόνο για να ξέρουμε ότι η εφαρμογή μας είναι σε λειτουργία.

Το δεύτερο endpoint που βλέπουμε είναι της αρχικής σελίδας. Το μόνο που κάνει αυτό το endpoint είναι, όταν ο χρήστης μπει μέσω browser στη σελίδα της εφαρμογής μας, να του δείχνει την αρχική σελίδα.



Σχήμα 4.14 Αρχική σελίδα της εφαρμογής

Εδώ βλέπουμε την αρχική οθόνη της εφαρμογής. Εδώ μπορεί ο χρήστης να επιλέξει ένα αρχείο εικόνας και να την ανεβάσει με το κουμπί upload.

Στο τέλος, αναφέρεται επίσης ποιο dataset χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου καθώς και σε ποιες κατηγορίες μπορεί να αποδώσει καλύτερα και με περισσότερη ακρίβεια.

```
Brachos Konstantinos +1
@api.route("/image", methods=["POST"])
def image_post():
    """This endpoint receives a form parameter called image with the image file to be classified,
    and saves the image uri to mongo.
    """
    model = s3_ops.load_model_from_s3("model_final")
    if "image" not in request.files:
        return (
            jsonify(
                {"message": "missing 'image' post form parameter", "status": "400"}
            ),
            400,
        )
    image_file = request.files['image']
    image_filename = secure_filename(image_file.filename)

    if image_filename == "":
        return (
            jsonify({"message": "image filename can not be empty", "status": "400"}),
            400,
        )

    if image_file and config.allowed_file(image_filename):
        content_type = image_file.content_type
        upload_image_url = s3_ops.upload_img_to_s3(image_file, image_filename, content_type)
        get_image = s3_ops.image_from_s3(image_filename)


        class_result, prob_result = predict.predict(get_image, model)

        predictions = {
            "class1": class_result[0],
            "class2": class_result[1],
            "class3": class_result[2],
            "prob1": prob_result[0],
            "prob2": prob_result[1],
            "prob3": prob_result[2],
        }
```

Σχήμα 4.15 Upload image endpoint 1/2

```
        obj = {"uri": upload_image_url, "labels": predictions}

        try:
            collection.insert_one(obj)
            return render_template('success.html', url=upload_image_url, predictions=predictions)
        except Exception as e:
            return jsonify({"Error": e.__str__()})

    else:
        return (jsonify({"message": "not allowed image type, allowed types are: 'png', 'jpg', 'jpeg', 'gif' ",
                         "status": "400"}),400)


if __name__ == "main":
    api.run()
```

Σχήμα 4.16 Upload image endpoint 2/2

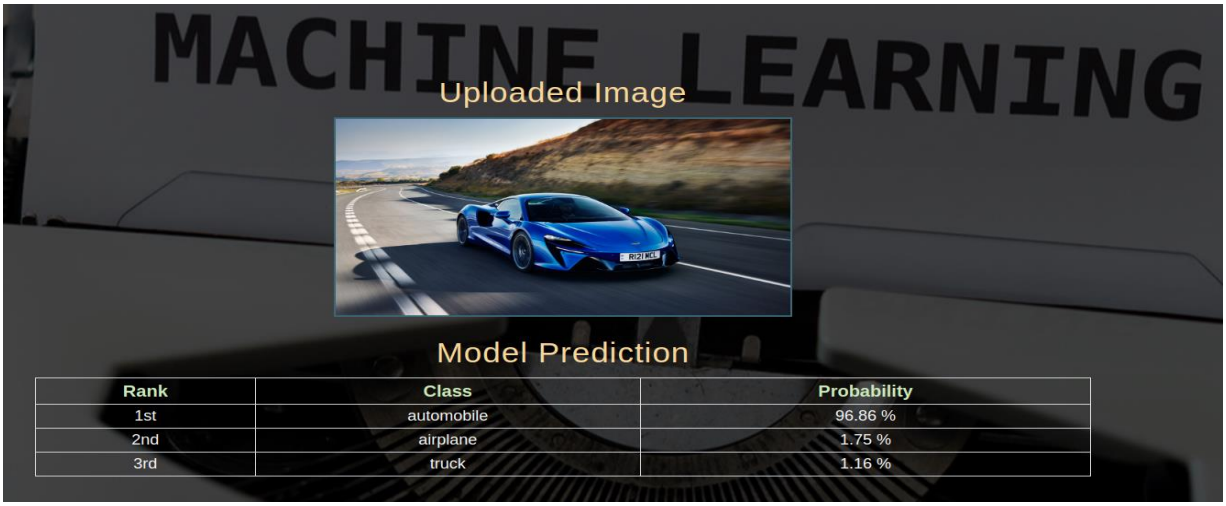
Στις εικόνες 4.15 και 4.16, έχουμε το endpoint που καλείται όταν πατάμε το upload από την αρχική σελίδα για να ανεβάσει ο χρήστης την εικόνα που θέλει.

Είναι ένα HTTP POST endpoint, που με λίγα λόγια σημαίνει ότι παίρνει δεδομένα από έξω όπως την εικόνα του χρήστη και το πρώτο πράγμα που κάνει είναι να φορτώσει το εκπαιδευμένο μοντέλο καλώντας την load_model_from_s3 function που είδαμε παραπάνω. Αφού φορτώσουμε το μοντέλο μας, ελέγχουμε εάν υπάρχει όντως εικόνα όταν έγινε το upload και τη φορτώνουμε στη μεταβλητή image_file και το όνομά της στη μεταβλητή image_filename, διαφορετικά γυρίζει ένα μήνυμα που λέει ότι λείπει η εικόνα από τις παραμέτρους του upload. Ελέγχουμε επίσης το όνομα της εικόνας να μην είναι κενό και γυρίζουμε το μήνυμα ότι δεν πρέπει να είναι κενό το όνομα της εικόνας σε περίπτωση που συμβεί κάτι τέτοιο. Ο τελευταίος έλεγχος που γίνεται είναι σε περίπτωση που όντως υπάρχει εικόνα, γίνεται έλεγχος στον τύπο της με βάση το file extension που έχει, χρησιμοποιώντας την function allowed_file που είδαμε και παραπάνω. Αν είναι στους επιτρεπόμενους τύπους τότε παίρνουμε τον τύπο της εικόνας, content_type, ανεβαίνει η εικόνα στο minio και παίρνουμε το url της χρησιμοποιώντας την upload_img_to_s3 και παίρνουμε πάλι την εικόνα έτοιμη για το μοντέλο χρησιμοποιώντας την image_from_s3.

Στη συνέχεια χρησιμοποιούμε την predict function και παίρνουμε τις λίστες που είδαμε παραπάνω με τα αποτελέσματα της ταξινόμησης, τις ετικέτες και τις πιθανότητές τους και φτιάχνουμε ένα dictionary με τα αποτελέσματα για να το χρησιμοποιήσουμε όταν θα το δείξουμε πίσω στη σελίδα αποτελεσμάτων.

Φτιάχνουμε επίσης το object, που περιέχει το url της εικόνας από το minio και τα αποτελέσματα της ταξινόμησης και το αποθηκεύουμε στη βάση δεδομένων μας και γυρίζουμε τα αποτελέσματα στη σελίδα αποτελεσμάτων για να τα δει ο χρήστης.

Εδώ ο χρήστης μπορεί να δει την εικόνα την οποία ανέβασε μαζί με τις ετικέτες με τις 3 μεγαλύτερες πιθανότητες. Επίσης υπάρχει και το κουμπί Homepage πάνω δεξιά το οποίο μας επιστρέφει στην αρχική σελίδα για να ανεβάσουμε νέα εικόνα.



Σχήμα 4.17 Σελίδα αποτελεσμάτων

4.4.2 Frontend

```
<!DOCTYPE html>
<html lang="eng">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href='../static/css/normalize.css'>
    <link rel="stylesheet" href='../static/css/grid.css'>
    <link rel='stylesheet' href='../static/css/style.css'>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <link href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;1,300&display=swap"
        rel="stylesheet" type='text/css'>
    <title>Image Classification WebApp</title>
    <meta name="keywords" content="custom file input, styling, label, cross-browser, accessible, input type file"/>
    <style>
        .content {
            max-width: 100%;
            margin: auto;
        }
        .center {
            margin: auto;
            width: 50%;
            padding: 10px;
            text-align: center;
        }
        .bg-img {
            width: 100%;
            height: 100vh;
            background-image: linear-gradient(rgba(0, 0, 0, 0.71), rgba(0, 0, 0, 0.7)),
                              url('../static/css/ImageClassification1080.jpg');
            background-size: cover;
            background-position: center;
        }
        .footer {
            width:100%;
            position:fixed;
            bottom: 0px;
            text-align:center;
        }
```

Σχήμα 4.18 Αρχική Σελίδα HTML 1/2

Στην εικόνα 4.18 βλέπουμε τα external resources και το styling της σελίδας. Χρησιμοποιούμε bootstrap, ένα open source framework που μας παρέχει με έτοιμα κομμάτια, όπως κουμπιά, για να μπορούμε εύκολα και γρήγορα να φτιάξουμε μια σελίδα. Εδώ λοιπόν χρησιμοποιούμε το link για να «συνδέσουμε» εξωτερικά css αρχεία, γραμματοσειρές και bootstrap JavaScript αρχεία στην αρχική σελίδα μας όπως επίσης και έναν τίτλο.

Στο style section έχουμε κάποιες κλάσεις της σελίδας, όπως το background image και το footer που θέλουμε με CSS και τα χρησιμοποιούμε για να αλλάξουμε τη μορφοποίηση της σελίδας όπου τα χρησιμοποιούμε.

```
<div class="bg-img">
    <div class="content">
        <div class="index-main">

            <div style="...">
                <div class="header-content">
                    <h1 style="...">
                        <span class="header-content-text">Image Classification</span>
                    </h1>
                </div>
            </div>

            <div class="upload-section center">

                <div class="upload-file">
                    <form class="file-form" action="/image" method="post" enctype="multipart/form-data">
                        <input class="file-form-input" type="file" name="image"/>
                        <button class="btn btn-success btn-lg upload-img">Upload</button>
                    </form>
                </div>

                <div class="header-content-sub">
                    <p style="..." class="header-content-info">{{error}}</p>
                </div>
            </div>

            <div class="footer">
                <div class="footer-content-sub">
                    <p style="..." class="header-content-info">The model was trained on
                        CIFAR-10 dataset and the model only performs better when the following image categories are
                        provided as input</p>
                </div>
                <div class="footer-content-sub">
                    <p style="..." class="header-content-info">dog , cat , horse , deer ,
                        frog , airplane , truck , automobile , ship , bird</p>
                </div>
            </div>
        </div>
    </div>
</div>
```

Σχήμα 4.19 Αρχική Σελίδα HTML 2/2

Στην εικόνα 4.19 βλέπουμε τον κώδικα HTML της αρχικής σελίδας, στην αρχή χρησιμοποιούμε την κλάση bg-img για να ορίσουμε το background της σελίδας και μέσα στο content έχουμε την index-main που χωρίζουμε τη σελίδα σε 3 κομμάτια.

Το πρώτο κομμάτι είναι το header, δηλαδή το πάνω μέρος της σελίδας. Σε αυτό το κομμάτι έχουμε απλά το κείμενο Image Classification που θα φαίνεται στην σελίδα.

Στο δεύτερο κομμάτι έχουμε το κομμάτι του uploading. Έχουμε μία φόρμα που όταν πατήσει ο χρήστης θα του ανοίξει ένα παράθυρο για να επιλέξει την εικόνα που θέλει για να ανεβάσει και το κουμπί που κάνει το ανέβασμα με το κείμενο Upload.

Στο τρίτο κομμάτι έχουμε το footer, το κάτω μέρος της σελίδας, όπου εμφανίζουμε το κείμενο σχετικά τα δεδομένα στα οποία εκπαιδεύτηκε το μοντέλο και τις κατηγορίες στις οποίες το μοντέλο αποδίδει καλύτερα.

```
<!DOCTYPE html>
<html lang="eng">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href='../static/css/normalize.css'>
    <link rel="stylesheet" href='../static/css/grid.css'>
    <link rel='stylesheet' href='../static/css/styleSucc.css'>
    <link href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;1,300&display=swap"
          rel="stylesheet" type='text/css'>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
          integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <title>Results Page</title>
    <style>
        th {
            text-align: center;
            font-size: 20px;
        }
        td {
            text-align: center;
            font-size: 18px;
        }
        img {
        width: 100%;
        height: 100%;
      }
    </style>
</head>
<body>
<div class="second-main">
    <nav>
        <ul style="..." class="main-nav">
            <li><a href="/">HomePage</a></li>
        </ul>

    </nav>
```

Σχήμα 4.20 Σελίδα Αποτελεσμάτων HTML 1/2

Στην εικόνα 4.20, όπως είδαμε και στην αρχική σελίδα, έχουμε τα external resources και το styling της σελίδας καθώς και τον τίτλο που έχουμε βάλει.

Επίσης σε αντίθεση με την αρχική σελίδα, στην σελίδα αποτελεσμάτων έχουμε κι έναν σύνδεσμο, Homepage, όπου όταν ο χρήστης βρίσκεται στη σελίδα με τα αποτελέσματα μπορεί να πατήσει τον σύνδεσμο για να επιστρέψει στην αρχική σελίδα.

```
    <div class="header">
        <row style="...">
            <h3 class="header-text">Uploaded Image</h3>
        </row>
        <div style="...">
            <img class="result-img" src="{{ url }}" alt={{ img_filename }}>
        </div>
    </div>

    <div class="info">

        <div style="...">
            <h3 class="header-text">Model Prediction</h3>
        </div>
        <div style="...">
            <table class="table-bordered text-light table-custom">
                <tr>
                    <th>Rank</th>
                    <th>Class</th>
                    <th>Probability</th>
                </tr>
                <tr>
                    <td>1st</td>
                    <td>{{ predictions.class1 }}</td>
                    <td>{{ predictions.prob1 }} %</td>
                </tr>
                <tr>
                    <td>2nd</td>
                    <td>{{ predictions.class2 }}</td>
                    <td>{{ predictions.prob2 }} %</td>
                </tr>
                <tr>
                    <td>3rd</td>
                    <td>{{ predictions.class3 }}</td>
                    <td>{{ predictions.prob3 }} %</td>
                </tr>
            </table>
        </div>
    </div>
</div>
</body>
```

Σχήμα 4.21 Σελίδα Αποτελεσμάτων HTML 2/2

Στη σελίδα 4.21 έχουμε τον HTML κώδικα της σελίδας αποτελεσμάτων. Στο header, πάνω μέρος της σελίδας, έχουμε το κείμενο Uploaded Image και ακριβώς από κάτω έχουμε την εικόνα που ανέβασε ο χρήστης την οποία την περνάμε με το url και το όνομά της.

Στο δεύτερο κομμάτι, info, έχουμε το κείμενο Model Prediction και έναν πίνακα που περιέχει τις 3 μεγαλύτερες πιθανότητες και τις αντίστοιχες κλάσεις που το μοντέλο έκανε την ταξινόμηση για την εικόνα που ανέβασε ο χρήστης.

4.5 Πως λειτουργεί η εφαρμογή

4.5.1 Στήσιμο της εφαρμογής

Πριν ξεκινήσουμε να χρησιμοποιούμε την εφαρμογή θα πρέπει να την στήσουμε σε ένα μηχάνημα, server ή ακόμα και στον δικό μας υπολογιστή, για να μπορούμε να έχουμε πρόσβαση στη σελίδα.

Για να μπορούμε να στήνουμε την εφαρμογή εύκολα και γρήγορα έχουμε φτιάξει ένα αρχείο Makefile. Το Makefile είναι ένα αρχείο που περιέχει ένα set εντολών ή οδηγιών και χρησιμοποιείται για την αυτοματοποίηση της διαδικασίας χτισίματος ενός project.

Όπως βλέπουμε και στην εικόνα 4.22 παρακάτω, στο Makefile έχουμε κάποιες εντολές για να χτίζουμε και να σταματάμε τα containers της εφαρμογής μας με μια μόνο εντολή, “make” και μια από τις παραμέτρους που έχουμε βάλει στο Makefile, για παράδειγμα “make run”.

```
help:
	@echo "Usage: make [option]"
	@echo "OPTIONS:"
	@echo "  run             use to run the project locally (via docker-compose)."
	@echo "  run-detached    use to run the project locally (via docker-compose) in detached mode."


run:
	docker compose down -v
	docker compose up -d

run-clean:
	docker compose down -v
	docker compose up  --build

run-detached:
	docker compose down -v
	docker compose up -d --build

down:
	docker compose down -v
```

Σχήμα 4.22 Makefile

Οι εντολές run, run-clean και run-detached πρώτα σταματάνε τα containers εάν υπάρχουν, με την εντολή “docker compose down -v” και τα ξαναχτίζουν από την αρχή με την εντολή “docker compose up. Οι διαφορές στις 3 run εντολές είναι ότι στην πρώτη απλά ξεκινάει τα containers σε detached mode, detached είναι όταν τα containers τρέχουν στο background και δεν είναι attached στο terminal που τρέξαμε την εντολή, στη δεύτερη χτίζει και ξεκινάει τα containers και στην τρίτη είναι ο συνδυασμός των 2 πρώτων όπου χτίζει και τα ξεκινάει σε detached mode.

Τέλος, έχουμε την εντολή down όπου την χρησιμοποιούμε όταν θέλουμε απλά να σταματήσουμε τα containers που έχουμε ενεργά.

Έτσι, με μια απλή εντολή μπορούμε να σηκώσουμε όλα τα containers της εφαρμογής μας και σε λίγα λεπτά να είναι έτοιμα προς χρήση.

4.5.2 Πως λειτουργεί η εφαρμογή

Αφού έχουμε στήσει την εφαρμογή μπορούμε πλέον να την χρησιμοποιήσουμε. Το πρώτο βήμα είναι να μπούμε στη σελίδα της εφαρμογής από έναν browser, έχουμε δοκιμάσει τον google chrome, χρησιμοποιώντας για url είτε την IP του server που έχουμε στήσει την εφαρμογή είτε το localhost αν το έχουμε στον υπολογιστή μας, και την πόρτα έχουμε ορίσει για το container της εφαρμογής, για παράδειγμα localhost:5000.

Αφού μπούμε στην αρχική σελίδα, μπορούμε να πατήσουμε στη φόρμα που είδαμε και να επιλέξουμε μια εικόνα και να πατήσουμε το κουμπί upload.

Για εικόνες μπορούμε να επιλέξουμε είτε από το internet, κατεβάζοντάς την στον υπολογιστή μας, είτε από εικόνες που έχουμε τραβήξει από το κινητό μας, ανεξάρτητα με την ανάλυσή της μιας και όταν την ανεβάζουμε γίνεται μια προεπεξεργασία πριν την χρησιμοποιήσει το μοντέλο.

5 Συμπεράσματα και Βελτιώσεις

5.1 Συμπεράσματα

Τα συνελικτικά νευρωνικά δίκτυα είναι πολύ ισχυρά εργαλεία για εργασίες ταξινόμησης εικόνων και αναγνώρισης αντικειμένων με πολύ καλές αποδόσεις. Στο μέλλον όμως σίγουρα θα υπάρξουν νέες αρχιτεκτονικές και νέες μεθοδολογίες και διαδικασίες που θα είναι ακόμα καλύτερες για αυτές τις εργασίες.

Η ανάπτυξη ενός τέτοιου μοντέλου είναι μια όπως αυτού που υλοποιήσαμε και είδαμε σε αυτή την πτυχιακή εργασία θέλει αρκετή προσπάθεια και υπομονή μιας και η διαδικασία εκπαίδευσης θέλει αρκετό χρόνο ακόμα και με πολύ καλές κάρτες γραφικών. Παρόλα αυτά καταφέραμε και φτιάξαμε ένα μοντέλο, και το βελτιώσαμε και το χρησιμοποιήσαμε για την εργασία της ταξινόμησης εικόνων με καλά αποτελέσματα ακόμα και με περιορισμένο χρόνο χρήσης κάρτας γραφικών και κατά συνέπεια χρόνο εκπαίδευσης του μοντέλου.

Επίσης αρκετά challenging ήταν και η υλοποίηση μιας web εφαρμογής που κάνει χρήση του εκπαιδευμένου μοντέλου έτσι ώστε ο χρήστης να μπορεί έμμεσα να κάνει χρήση του μοντέλου για τις εικόνες που ανεβάζει.

Παρά τις βελτιώσεις που κάναμε τόσο στο μοντέλο αλλά και στην εφαρμογή που φτιάξαμε υπάρχουν ακόμα βελτιώσεις που μπορούν να γίνουν για να έχουμε ακόμα καλύτερη απόδοση του μοντέλου και περισσότερες λειτουργίες στην εφαρμογή μας.

5.2 Βελτιώσεις

Παρακάτω θα δούμε κάποιες βελτιώσεις που θα μπορούσαν να γίνουν στην εφαρμογή και στο μοντέλο που υλοποιήσαμε σε αυτή την πτυχιακή εργασία.

5.2.1 Χρήστες

Μία βασική βελτίωση είναι να υπάρχουν διαφόρων ειδών χρήστες. Ο απλός χρήστης να μην χρειάζεται λογαριασμό για να χρησιμοποιήσει την εφαρμογή. Ο εγγεγραμμένος χρήστης να χρειάζεται να συνδεθεί με τον λογαριασμό του και να έχει πρόσβαση σε παραπάνω λειτουργίες της εφαρμογής, όπως για παράδειγμα σε αναζήτηση ή ακόμα και σε προφίλ.

5.2.2 Αναζήτηση

Μια άλλη ωραία βελτίωση είναι να υπάρχει αναζήτηση παλαιότερων αποτελεσμάτων του χρήστη. Εδώ θα χρειαζόταν ο χρήστης να έχει λογαριασμό για να μπορεί να γίνει αναζήτηση με βάση κάποιο μοναδικό αναγνωριστικό του ως χρήστης για να μπορεί να δει τα δικά του αποτελέσματα και όχι άλλων χρηστών.

5.2.3 Προφίλ χρήστη

Μια άλλη ωραία προσθήκη θα ήταν ένας χρήστης να έχει δικό του προφίλ και να μπορεί να δίνει πρόσβαση σε άλλον χρήστη.

5.2.4 Ανέβασμα εικόνας από url

Ακόμα μια καλή λειτουργία είναι να μπορεί ο χρήστης να ανεβάσει μια εικόνα κατευθείαν από url, να δίνει δηλαδή το link της εικόνας για παράδειγμα από το google χωρίς να χρειάζεται να κατεβάσει την εικόνα στον υπολογιστή ή το κινητό του και να την ανεβάσει στην εφαρμογή.

5.2.5 Ασφάλεια

Μια από τις πιο σημαντικές βελτιώσεις είναι η ασφάλεια. Αυτό μπορεί να γίνει σε πολλά σημεία της εφαρμογής, για παράδειγμα στους κωδικούς των χρηστών για να είναι κωδικοποιημένα όταν θα αποθηκευτούν στη βάση. Επιπλέον, θα πρέπει να υπάρχει έλεγχος όταν ο χρήστης πάει να μπει στο προφίλ του για να επιβεβαιωθεί ότι έχει το δικαίωμα.

5.2.6 Καλύτερη βελτιστοποίηση παραμέτρων του μοντέλου

Θα μπορούσαμε να έχουμε μεγαλύτερο αριθμό φίλτρων στην αναζήτηση για το βέλτιστο μοντέλο ή/και με μικρότερο βήμα αναζήτησης για να δοκιμάσουμε ακόμα περισσότερα μοντέλα. Επίσης θα μπορούσαμε να δοκιμάσουμε περισσότερους αλγόριθμους βελτιστοποίησης, ρυθμούς μάθησης ακόμα και διαφορετικό είδος hyperparameter tuning.

5.3 Περιορισμοί

Οι περιορισμοί που υπάρχουν αφορούν κυρίως την εκπαίδευση του νευρωνικού δικτύου. Ένα νευρωνικό δίκτυο χρειάζεται υπερβολικά πολλά δεδομένα κατά την εκπαίδευσή του, ακόμα και για πιο απλές εργασίες που στον άνθρωπο θα έπαιρνε λίγη ώρα και λιγότερα δεδομένα. Στην προκειμένη περίπτωση για την ταξινόμηση εικόνων, θα έπρεπε να έχουμε πολλές χιλιάδες εικόνες ανά κατηγορία, για x αριθμό κατηγοριών, που όχι μόνο είναι αρκετά δύσκολο να βρεθούν τόσες πολλές εικόνες αλλά είναι δύσκολο και να αποθηκευτούν, μιας και αυτό θα είχε μεγάλο κόστος. Άλλος ένας περιορισμός είναι η υπολογιστική ισχύ που χρειάζεται ένα νευρωνικό δίκτυο κατά την εκπαίδευσή του. Μια κάρτα γραφικών που χρησιμοποιείται σε συστήματα για την εκπαίδευση νευρωνικών δικτύων μπορεί να κοστίζει μέχρι και αρκετές χιλιάδες ευρώ αλλά ακόμα και λύσεις όπως το cloud μπορεί να κοστίσει αρκετά. Τέλος, υπάρχει και ο χρόνος εκπαίδευσης, όσο πιο πολύπλοκη η αρχιτεκτονική του νευρωνικού δικτύου ή τα δεδομένα που δέχεται, αυξάνουν τον χρόνο εκπαίδευσης αρκετά.

Ο κώδικας της εφαρμογής καθώς και οδηγίες σχετικά με το πως να τρέξουμε την εφαρμογή μπορεί να βρεθεί στον παρακάτω ιστότοπο: https://github.com/OperatingFuture/thesis-image-classification

Κατάλογος Εικόνων

Βιβλιογραφία

A Complete Guide to Image Classification in 2023. (n.d.). Retrieved from viso.ai: https://viso.ai/computer-vision/image-classification/

Basic classification: Classify images of clothing. (n.d.). Retrieved from tensorflow.org: https://www.tensorflow.org/tutorials/keras/classification

DeepLearning.ai. (n.d.). *Convolutional Neural Networks*. Retrieved from coursera.org: https://www.coursera.org/learn/convolutional-neural-networks

Developing RESTful APIs with Python and Flask. (n.d.). Retrieved from auth0.com: https://auth0.com/blog/developing-restful-apis-with-python-and-flask/

dshahid380. (n.d.). *Convolutional Neural Network*. Retrieved from towardsdatascience.com: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

Introduction to the Keras Tuner. (n.d.). Retrieved from tensorflow.org: https://www.tensorflow.org/tutorials/keras/keras_tuner

Karagiannakos, S. (n.d.). *Best practices to write Deep Learning code: Project structure, OOP, Type checking and documentation*. Retrieved from theaisummer.com/: https://theaisummer.com/best-practices-deep-learning-code/

LeCun, Y. (n.d.). *Deep Learning Course*. Retrieved from cds.nyu.edu/: https://cds.nyu.edu/deep-learning/

MongoDB Tutorial. (n.d.). Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/mongodb/index.htm

Object Detection Guide. (n.d.). Retrieved from https://www.fritz.ai/object-detection/

Skalski, P. (n.d.). *Gentle Dive into Math Behind Convolutional Neural Networks*. Retrieved from towardsdatascience.com: https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

Srivastav, P. (n.d.). *Docker for beginners*. Retrieved from https://docker-curriculum.com/: https://docker-curriculum.com/