ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΑΞΙΝΟΜΗΣΗ ΣΚΙΤΣΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΒΑΘΙΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΚΑΙ ΤΗΣ TensorFlow

Διπλωματική Εργασία

Καφαλή Ευθυμία

Θεσσαλονίκη, Ιούνιος 2019

ΤΑΞΙΝΟΜΗΣΗ ΣΚΙΤΣΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΒΑΘΙΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΚΑΙ ΤΗΣ TensorFlow

Ευθυμία Καφαλή

Πτυχίο Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, 2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής:
Ιωάννης Ρεφανίδης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21/06/2019

Ιωάννης Ρεφανίδης	Ηλίας Σακελλαρίου	Κωνσταντίνος Μαργαρίτης
.....	.....	.....

Ευθυμία Καφαλή

.....

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με τη μελέτη και υλοποίηση μίας αρχιτεκτονικής Βαθιών Νευρωνικών Δικτύων και τη χρήστη αυτής σε ένα πρόβλημα ταξινόμησης εικόνων. Ο στόχος της εργασίας είναι ο σχηματισμός ενός μοντέλου Βαθιάς Μάθησης το οποίο, δοθέντος ενός σκίτσου, θα μπορεί να προβλέψει με ακρίβεια το αντικείμενο το οποίο απεικονίζεται σε αυτό. Ο σκοπός της δημιουργίας ενός τέτοιου μοντέλου είναι η αποθήκευση της γνώσης που απέκτησε κατά την εκπαίδευσή του, προκειμένου να μπορεί μελλοντικά να χρησιμοποιηθεί από παρόμοιες εφαρμογές ταξινόμησης εικόνων για τη βελτίωση των αποτελεσμάτων τους. Το μοντέλο που χρησιμοποιήθηκε εκπαιδεύτηκε πάνω σε περίπου 1.5 εκατομμύρια ασπρόμαυρα σκίτσα του συνόλου δεδομένων "Quick, Draw!", το οποίο προέρχεται από το ομώνυμο διαδικτυακό παιχνίδι που αναπτύχθηκε από τη Google στα πλαίσια των πειραμάτων Τεχνητής Νοημοσύνης "Google AI Experiments". Το πρόβλημα προσεγγίστηκε με τη χρήση μίας συγκεκριμένης αρχιτεκτονικής ενός Βαθιού Συνελικτικού Νευρωνικού Δικτύου, ωστόσο στα αρχικά στάδια έγιναν και πειράματα μικρότερης κλίμακας. Ως αποτέλεσμα, προέκυψε ένα μοντέλο ταξινόμησης εικόνων με ποσοστό επιτυχίας πρώτης πρόβλεψης 72% και ακρίβεια 85% στις πρώτες τρεις προβλέψεις. Ως μελλοντική επέκταση το μοντέλο αυτό μπορεί εύκολα να χρησιμοποιηθεί ως βάση για τη βελτίωση των αποτελεσμάτων άλλων μοντέλων που έχουν ως στόχο την επίλυση παρόμοιων προβλημάτων.

Λέξεις Κλειδιά: Βαθιά Μάθηση, Συνελικτικά Νευρωνικά Δίκτυα, Ταξινόμηση Εικόνων

Abstract

This thesis deals with the study and implementation of a Deep Neural Network architecture and its use in a problem of image classification. The aim of the thesis is to create a Deep Learning model which, given a sketch, will be able to predict the drawn object class with a high prediction accuracy. The purpose of creating such a model is to store the knowledge it acquired during its training, so that it can be used in the future by similar image classification applications with a potential improvement on the results. The model was trained on approximately 1.5 million black and white sketches of the “Quick, Draw!” dataset, which comes from an online game developed by Google as part of "Google AI Experiments". The problem was approached using a specific architecture of a Convolutional Neural Network, however, in the early stages of the implementation, smaller scale experiments were conducted too. As a result, an image classification model with 72% prediction accuracy and a top-3 accuracy of 85% was obtained. As a future extension, this model can be easily used as a basis for improving the results of other models with similar tasks, i.e. to make predictions on similar problems.

Keywords: Deep Learning, Convolutional Neural Networks, Image Classification

Ευχαριστίες

Ευχαριστώ την εταιρεία NVIDIA για την προσφορά της GPU NVIDIA Titan X στο Πανεπιστήμιο Μακεδονίας, καθώς χωρίς αυτή θα ήταν αδύνατο να πραγματοποιηθούν μεγάλης κλίμακας πειράματα στα πλαίσια της διπλωματικής μου εργασίας. Τέλος, ευχαριστώ τον επιβλέποντα καθηγητή μου κ. Ιωάννη Ρεφανίδη για τη συνεχή καθοδήγηση που μου παρείχε κατά το διάστημα εκπόνησης της εργασίας μου και τη διευθέτηση των τεχνικών προβλημάτων που προέκυψαν στα πλαίσια των πειραμάτων.

Acknowledgements

The Titan Xp used for this research was donated by the NVIDIA Corporation.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1 Εισαγωγή

Σκοπός της εργασίας αυτής είναι αφενός η ανάλυση του θεωρητικού υπόβαθρου πίσω από τα Βαθιά Νευρωνικά Δίκτυα και αφετέρου η παρουσίαση μίας αρχιτεκτονικής ικανής να αντιμετωπίσει επαρκώς ένα πρόβλημα ταξινόμησης εικόνων.

Στο πρώτο κεφάλαιο γίνεται μία σύντομη περιγραφή γενικών εννοιών, όπως η Τεχνητή Νοημοσύνη, η Μηχανική και η Βαθιά Μάθηση και γίνεται αναφορά σε γνωστές και ευρέως χρησιμοποιούμενες αρχιτεκτονικές Βαθιών Νευρωνικών Δικτύων. Παρουσιάζεται επιπλέον σύντομα το σύνολο δεδομένων που χρησιμοποιήθηκε για την υποστήριξη της μεθοδολογίας που ακολουθείται στην εργασία αυτή.

Στο δεύτερο κεφάλαιο, παρέχεται μία παρουσίαση του θεωρητικού υπόβαθρου πίσω από τα Νευρωνικά Δίκτυα. Ειδικότερη αναφορά γίνεται στον τύπο Νευρωνικών Δικτύων που χρησιμοποιήθηκε για το πρακτικό κομμάτι της εργασίας αυτής, τα Συνελικτικά Νευρωνικά Δίκτυα, τα επιμέρους χαρακτηριστικά τους και σημαντικές παραμέτρους που πρέπει να ληφθούν υπόψη κατά την υλοποίηση μίας τέτοιας αρχιτεκτονικής.

Στο τρίτο κεφάλαιο γίνεται μία πλήρης περιγραφή της αρχιτεκτονικής που δημιουργήθηκε για την ταξινόμηση των δεδομένων που χρησιμοποιούνται σε αυτή την εργασία, καθώς και σε μία δεύτερη αρχιτεκτονική η οποία δοκιμάστηκε σε αρχικά στάδια των πειραμάτων. Τέλος, αναλύεται και υποστηρίζεται η επιλογή των παραμέτρων που χρησιμοποιήθηκαν με βάση τα δεδομένα και τις απαιτήσεις του προβλήματος.

Στο τέταρτο κεφάλαιο, παρουσιάζονται τα αποτελέσματα που προέκυψαν εφαρμόζοντας την προαναφερόμενη μεθοδολογία, μέσα από αριθμητικά αποτελέσματα, γραφήματα και παραδείγματα προβλέψεων.

Τέλος, στο πέμπτο κεφάλαιο γίνεται μία σύνοψη και προτείνονται μέθοδοι που θα μπορούσαν πιθανώς να βελτιώσουν περαιτέρω το αποτέλεσμα ως μελλοντική επέκταση της διπλωματικής αυτής εργασίας.

1.1 Τεχνητή Νοημοσύνη

Η Τεχνητή Νοημοσύνη (Artificial Intelligence – AI) είναι ένας τομέας της επιστήμης της πληροφορικής, ο οποίος σχετίζεται με τον σχεδιασμό και την υλοποίηση μοντέλων που διαθέτουν, ως ένα βαθμό, ευφυΐα. Η ιδέα της δημιουργίας συστημάτων με ευφυΐα βασίζεται σε μία προσπάθεια μίμησης της ανθρώπινης συμπεριφοράς και συνεπώς ένα τέτοιο σύστημα είναι σε θέση να πραγματοποιεί μαθηματικούς υπολογισμούς και με βάση αυτούς να αποκτά γνώση, να εξάγει συμπεράσματα και να αντιδρά στα ερεθίσματα που δέχεται.

Ο τομέας της Τεχνητής Νοημοσύνης είναι σχετικά πρόσφατος, βρίσκει όμως εφαρμογή ήδη σε πολλούς κλάδους, ενώ τα τελευταία χρόνια είναι στο επίκεντρο της έρευνας και εξελίσσεται συνεχώς. Μερικές από τις πολλές

εφαρμογές της είναι το αυτοματοποιημένο gameplay, η αναγνώριση ομιλίας και φυσικής γλώσσας, η όραση υπολογιστών.

Ανάλογα με το πρόβλημα που επιδιώκει να λύσει ένα ευφυές σύστημα, η Τεχνητή Νοημοσύνη μπορεί να διαχωριστεί σε ένα ευρύτερο σύνολο πεδίων. Στα πλαίσια της εργασίας αυτής γίνεται αναφορά στα πεδία της Μηχανικής και Βαθιάς Μάθησης και αναλύονται ειδικότερα οι εφαρμογές αυτών σε προβλήματα ταξινόμησης εικόνας.

1.2 Μηχανική Μάθηση

Ως Μηχανική Μάθηση (Machine Learning - ML) αναφέρεται ο τομέας εκείνος της Τεχνητής Νοημοσύνης ο οποίος στοχεύει στη μοντελοποίηση συστημάτων που είναι ικανά να αποκτούν γνώση, να την επεξεργάζονται και με βάση αυτή να προσαρμόζονται και να εξάγουν συμπεράσματα. Η βάση των συστημάτων αυτών είναι στην ουσία ένα σύνολο αλγορίθμων, που δέχονται ως είσοδο δεδομένα και μέσα από μία σειρά σύνθετων υπολογισμών, οδηγούνται στο να αναγνωρίζουν ακόμη και τα πιο σύνθετα χαρακτηριστικά τους.

Υπάρχουν διάφορα είδη προβλημάτων που η Μηχανική Μάθηση καλείται να λύσει. Ως βασικές κατηγορίες αναφέρονται τα προβλήματα επιβλεπόμενης, μη – επιβλεπόμενης και ημι – επιβλεπόμενης μάθησης.

Επιβλεπόμενη Μάθηση (Supervised Learning). Σε αυτού του είδους τα προβλήματα είναι διαθέσιμο στον αλγόριθμο ένα σύνολο από δεδομένα (π.χ. εικόνες, βίντεο) καθώς επίσης και ένα σύνολο ετικετών (labels). Κάθε ένα από τα παραδείγματα δεδομένων προσδιορίζεται από μία ετικέτα και συνεπώς, για κάθε ένα από τα δείγματα είναι γνωστή στο σύστημα η επιθυμητή έξοδος. Το ζητούμενο σε αυτά τα προβλήματα είναι αρχικά ο αλγόριθμος να εκπαιδευτεί στην αναγνώριση των ετικετών ενός συνόλου δειγμάτων εκπαίδευσης. Πρέπει στη συνέχεια να προσαρμοστεί με βάση τα λάθη των προβλέψεών του, ώστε να τα ελαχιστοποιήσει, και να καταφέρει εντέλει να προβλέψει με ακρίβεια τις ετικέτες κάποιων δειγμάτων τα οποία δεν έχουν χρησιμοποιηθεί κατά την εκπαίδευσή του. Τα προβλήματα αυτά αποτελούν στην ουσία **προβλήματα ταξινόμησης (classification problems).** Στα επόμενα κεφάλαια της διπλωματικής αυτής εργασίας αναλύονται εκτενέστερα τεχνικές ταξινόμησης εικόνας.

Αν και οι αλγόριθμοι επιβλεπόμενης μάθησης ποικίλουν ως προς το είδος τους και χρησιμοποιούν διαφορετικές τεχνικές, υπάρχουν κάποια βασικά χαρακτηριστικά που αφορούν τη δομή τους και όλοι τους διαθέτουν: Αρχικά, σε κάθε αλγόριθμο επιβλεπόμενης μάθησης είναι απαραίτητο σε αρχικό στάδιο να προσδιορίζεται το σύνολο δεδομένων από το οποίο θα εξαχθεί γνώση. Στη συνέχεια προσδιορίζεται το μοντέλο ως το εργαλείο που θα χρησιμοποιηθεί για την εκμάθηση χαρακτηριστικών από τα δεδομένα. Προκειμένου να προκύψει ορθή γνώση, είναι απαραίτητος ο ορισμός μίας συνάρτησης κόστους, που είναι υπεύθυνη για την τιμωρία των λανθασμένων προβλέψεων και τέλος μία συνάρτηση βελτιστοποίησης χρησιμοποιείται έτσι ώστε να ελαχιστοποιεί τις λάθος προβλέψεις.

Μη – Επιβλεπόμενη Μάθηση (Unsupervised Learning). Το χαρακτηριστικό της μη – επιβλεπόμενης μάθησης είναι το γεγονός ότι τα δείγματα δεδομένων που χρησιμοποιούνται για εκπαίδευση δεν προσδιορίζονται από ετικέτες, δηλαδή δίνονται στο μοντέλο μόνο δείγματα εισόδου που δεν αντιστοιχίζονται με κάποια έξοδο. Ο στόχος σε αυτού του είδους τα προβλήματα είναι ο αλγόριθμος που χρησιμοποιείται να ανακαλύψει μόνος του ομοιότητες/μοτίβα ανάμεσα στα διάφορα δείγματα, προκειμένου να σχηματιστεί ένας αριθμός από συστάδες (clusters) δειγμάτων που παρουσιάζουν με κάποιο τρόπο κοινά χαρακτηριστικά. Η βασικότερη πρόκληση σε πραγματικές εφαρμογές αυτού του είδους προβλημάτων είναι η απόκτηση κατάλληλων δεδομένων, από την πλευρά του να μπορούν με κάποιο τρόπο να ομαδοποιηθούν σε διαφορετικές κατηγορίες. Ο κύριος τύπος προβλημάτων μη – επιβλεπόμενης μάθησης είναι τα **προβλήματα συσταδοποίησης (clustering problems).**

Ημι – Επιβλεπόμενη Μάθηση (Semi – Supervised Learning). Στην ημι – επιβλεπόμενη μάθηση δίνεται ως είσοδος στον αλγόριθμο/μοντέλο ένα σύνολο δεδομένων εκπαίδευσης, το οποίο περιέχει τόσο δείγματα με ετικέτες, όσο και δείγματα χωρίς ετικέτες. Η χρήση αλγορίθμων ημι – επιβλεπόμενης μάθησης είναι ιδανική σε περιπτώσεις όπου, αφενός η εύρεση ενός μεγάλου συνόλου δεδομένων με ετικέτες είναι αδύνατη, αφετέρου η εξαγωγή χαρακτηριστικών από τα δεδομένα είναι δύσκολο να γίνει. Τα δεδομένα που διαθέτουν ετικέτες είναι αυτά που δίνουν μία ώθηση στον αλγόριθμο, ώστε να καταφέρει να εξάγει χρήσιμα συμπεράσματα για τα δείγματα.

1.3 Βαθιά Μάθηση

Η Βαθιά Μάθηση (Deep Learning) είναι ένας πιο ειδικός τύπος Μηχανικής Μάθησης. Κατά βάση ακολουθεί τις αρχές της Μηχανικής Μάθησης και λειτουργεί με έναν παρόμοιο τρόπο, παρόλα αυτά, διαθέτει διαφορετικές ικανότητες και μπορεί να χρησιμοποιηθεί σε πιο σύνθετα προβλήματα, εξάγοντας καλύτερα αποτελέσματα.

Ένα μοντέλο Βαθιάς Μάθησης σχεδιάζεται έτσι ώστε να αναλύει συνεχώς τα δεδομένα που του παρουσιάζονται, με έναν τρόπο παρόμοιο με αυτόν που χρησιμοποιεί ο ανθρώπινος εγκέφαλος, προκειμένου να εξάγει συμπεράσματα. Για να το πετύχει αυτό, χρησιμοποιεί μία δομή από επίπεδα, με κάθε επίπεδο να είναι υπεύθυνο για ένα πλήθος μαθηματικών υπολογισμών. Οι δομές είναι στην ουσία ένα σύνολο από νευρώνες, οι οποίοι συνδέονται μεταξύ τους και σχηματίζουν ένα Νευρωνικό Δίκτυο (Neural Network – NN - ΝΔ). Το πλήθος των επιπέδων του μοντέλου σηματοδοτεί το βάθος ενός ΝΔ. Όσο μεγαλύτερο το βάθος του ΝΔ, τόσο περισσότερους (και πιο σύνθετους) υπολογισμούς είναι σε θέση να πραγματοποιήσει.

Ένα ΝΔ που αποτελείται από μεγάλο πλήθος επιπέδων, έχει δηλαδή μεγάλο βάθος, ονομάζεται Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network). Τα Βαθιά Νευρωνικά Δίκτυα είναι ιδανικά για επίλυση προβλημάτων στα οποία είναι διαθέσιμος ένας πολύ μεγάλος όγκος δεδομένων.

1.4 Ταξινόμηση Εικόνας

Ως ταξινόμηση εικόνας αναφέρεται η διαδικασία κατά την οποία μία εικόνα αντιστοιχίζεται με μία συγκεκριμένη κλάση/ετικέτα, ανάλογα με το αντικείμενο που απεικονίζεται σε αυτή. Για τους ανθρώπους, η διαδικασία αυτή είναι μία από τις πρώτες δεξιότητες που αναπτύσσουμε και εξελίσσουμε με την εμπειρία που αποκτούμε, ώσπου τελικά η γνώση του τι απεικονίζει μία εικόνα είναι μία διαδικασία που προκύπτει φυσιολογικά, χωρίς ιδιαίτερη σκέψη. Η αναγνώριση ενός αντικειμένου σε μία εικόνα, ή στο περιβάλλον γύρω μας, γίνεται τις περισσότερες φορές υποσυνείδητα και είναι αποτέλεσμα της γενίκευσης των γνώσεων που έχουμε αποκτήσει κατά τη διάρκεια της ζωής μας και της προσαρμογής μας στο περιβάλλον γύρω μας.

Με παρόμοιο τρόπο μπορεί να περιγραφεί και η διαδικασία της ταξινόμησης εικόνων από ένα ΝΔ. Στη θέση της υποσυνείδητης ταξινόμησης που κάνει ο ανθρώπινος εγκέφαλος, το ΝΔ μαθαίνει να αναγνωρίζει χαμηλού επιπέδου χαρακτηριστικά της εικόνας, δηλαδή τα μοναδικά χαρακτηριστικά του αντικειμένου που απεικονίζεται σε αυτή. Στη συνέχεια, πρέπει να αναθέσει στην εικόνα μία ετικέτα που προσδιορίζει την κλάση από την οποία περιγράφεται καλύτερα. Πιο συγκεκριμένα, ως έξοδος της διαδικασίας αυτής μπορεί να προκύψει είτε η επικρατέστερη κλάση, είτε μία κατανομή πιθανότητας, με τη μεγαλύτερη πιθανότητα να υποδεικνύει την επικρατέστερη κλάση.

Ο τύπος ΝΔ που χρησιμοποιείται συχνότερα για την αναγνώριση χαρακτηριστικών σε εικόνες και κατ' επέκταση την ταξινόμηση εικόνων είναι τα Συνελικτικά Νευρωνικά Δίκτυα, τα οποία περιγράφονται εκτενώς στο Κεφάλαιο 2. Η έμπνευση τους προκύπτει από τον τρόπο με τον οποίο λειτουργεί ο οπτικός φλοιός, πράγμα που τα καθιστά κατάλληλα για προβλήματα που αφορούν την επεξεργασία εικόνας με οποιονδήποτε τρόπο.

Το 2010 πραγματοποιήθηκε για πρώτη φορά ένας διαγωνισμός ταξινόμησης εικόνων, ο ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), στόχος του οποίου ήταν η ταξινόμηση ενός μεγάλου πλήθους εικόνων, με σκοπό την εξέλιξη της έρευνας στον τομέα της Υπολογιστικής Όρασης. Το σύνολο δεδομένων του ImageNet περιέχει περίπου 20000 κατηγορίες αντικειμένων (όπως για παράδειγμα μπαλόνια, σκύλους, κτλ.) κάθε μία από τις οποίες περιέχει χιλιάδες εικόνες. Οι συμμετέχοντες καλούνται να σχηματίσουν και να εκπαιδεύσουν ένα μοντέλο, το οποίο θα είναι σε θέση να ταξινομεί σωστά μία εικόνα εισόδου στην αντίστοιχη κλάση. Ο διαγωνισμός ImageNet γνώρισε μεγάλη αναγνώριση από ερευνητικές ομάδες και συνεπώς είναι ετήσιος. Όπως γίνεται εύκολα αντιληπτό, αποτελεί πλέον σημείο αναφοράς στα προβλήματα Υπολογιστικής Όρασης. Κατά βάση στον διαγωνισμό ImageNet προτείνονται και χρησιμοποιούνται αρχιτεκτονικές Συνελικτικών Νευρωνικών Δικτύων για τους λόγους που προαναφέρθηκαν. Τα τελευταία χρόνια έχουν προταθεί διάφορες αρχιτεκτονικές Συνελικτικών Νευρωνικών Δικτύων, τόσο στα πλαίσια του διαγωνισμού, όσο και εκτός αυτού, οι σημαντικότερες από τις οποίες αναφέρονται παρακάτω.

LeNet. Το LeNet – 5 [1] ήταν το πρώτο επιτυχημένο Συνελικτικό Νευρωνικό Δίκτυο 7 επιπέδων, το οποίο σχεδιάστηκε προκειμένου να ταξινομεί εικόνες που

απεικόνιζαν χαρακτήρες και ψηφία. Είναι μία ιδιαίτερα αποτελεσματική αρχιτεκτονική ΝΔ όταν πρόκειται για εικόνες χαμηλής ανάλυσης, παρόλα αυτά δεν προτιμάται για προβλήματα ταξινόμησης εικόνων υψηλότερης ανάλυσης, καθώς σε τέτοιες περιπτώσεις απαιτούνται περισσότερα συνελικτικά επίπεδα, δηλαδή μεγαλύτερο βάθος στο δίκτυο.

AlexNet. Η αρχιτεκτονική του AlexNet [2] είναι σχετικά απλή και μοιάζει με αυτή του LeNet – 5, με τη διαφορά ότι το AlexNet είναι μεγαλύτερο, με περισσότερα επίπεδα. Το AlexNet εκπαιδεύτηκε με δεδομένα της βάσης ImageNet, με σκοπό να λάβει μέρος το 2012 στο διαγωνισμό ImageNet (ILSVRC), τον οποίο και κέρδισε. Το AlexNet χρησιμοποιεί τη μη – γραμμική συνάρτηση ενεργοποίησης ReLU, τεχνικές data augmentation και επίπεδα dropout, με αποτέλεσμα το δίκτυο να μην υπερπροσαρμόζεται στα δεδομένα.

VGG16. Το μοντέλο VGG16 [3] μοιάζει με την αρχιτεκτονική AlexNet, με τη διαφορά ότι αποτελείται από 16 συνελικτικά επίπεδα. Το VGG16 προτάθηκε το 2014 για τον διαγωνισμό ImageNet και τότε θεωρούνταν στα πλαίσια της έρευνας ως ένα πολύ Βαθύ Νευρωνικό Δίκτυο. Εκπαιδεύτηκε για περίπου 2 – 3 εβδομάδες σε 4 GPU και αποτελεί μία αρχιτεκτονική που προτιμάται μέχρι και σήμερα για την εξαγωγή χαρακτηριστικών από εικόνες. Το μειονέκτημα του VGG16 είναι το γεγονός ότι αποτελείται από ένα πολύ μεγάλο πλήθος παραμέτρων (περίπου 138 χιλιάδες) και απαιτεί μεγάλη υπολογιστική ισχύ για την εκπαίδευσή του.

ResNet. Το μοντέλο ResNet [4] διαφέρει από τις αρχιτεκτονικές που προαναφέρθηκαν , καθώς δεν είναι ακολουθιακό (sequential), δηλαδή τα επίπεδά του δε στοιβάζονται γραμμικά. Στο ResNet χρησιμοποιούνται μονάδες μικρο - αρχιτεκτονικής, που μπορούν να περιγραφούν ως δίκτυα αποτελούμενα από μικρότερα δίκτυα. Οι μονάδες μικρο - αρχιτεκτονικής του ResNet ονομάζονται building blocks και μία συλλογή από αυτές συγκροτούν στην ευρύτερη αρχιτεκτονική (μακρο – αρχιτεκτονική) του μοντέλου ResNet. Προτάθηκε το 2015 και χρησιμοποιήθηκε για την εκπαίδευση ενός Νευρωνικού Δικτύου 152 επιπέδων, καταφέρνοντας να διατηρήσει την πολυπλοκότητα χαμηλότερη από αυτή του VGG16 και να ξεπεράσει τις δεξιότητες του ανθρώπινου ματιού στην ταξινόμηση εικόνων.

1.5 Σκοπός – Στόχοι

Στη διπλωματική αυτή εργασία χρησιμοποιείται το σύνολο δεδομένων “Quick, Draw!” που αναλύεται παρακάτω. Ο στόχος της εργασίας είναι η εκπαίδευση ενός CNN (Συνελικτικού ΝΔ) πάνω σε ένα σύνολο από ζωγραφισμένα σκίτσα, προκειμένου να προκύψει ένα μοντέλο που θα αναγνωρίζει αποτελεσματικά τι απεικονίζει το καθένα από αυτά, και θα τα ταξινομεί σωστά στην κλάση την οποία ανήκουν.

Το σύνολο δεδομένων “Quick, Draw!”

Το «Quick, Draw!» [5] ξεκίνησε ως ένα παιχνίδι, το οποίο παρουσιάστηκε από τη Google στο συνέδριο Google I/O το 2016. Στην αρχική μορφή του παιχνιδιού μπορούσαν να συμμετέχουν δύο παίκτες, ο ένας ζωγράφιζε ένα σκίτσο και ο άλλος προσπαθούσε να μαντέψει τι απεικονίζει. Το 2017 η ιδέα αυτή προχώρησε ένα βήμα μπροστά, όταν μία ομάδα ερευνητών της Google σκέφτηκε να χρησιμοποιήσει το σύνολο δεδομένων του παιχνιδιού, προκειμένου να εκπαιδεύσει ένα Νευρωνικό Δίκτυο (Sketch – RNN) [6] να αναγνωρίζει τι πρόκειται να ζωγραφίσει ο χρήστης σε πραγματικό χρόνο, αντί για τη συμμετοχή του δεύτερου παίκτη.

Πιο συγκεκριμένα, οι ερευνητές εκτός από την εκπαίδευση του Νευρωνικού Δικτύου, έφτιαξαν ένα διαδικτυακό παιχνίδι, στο οποίο ο χρήστης καλείται να ζωγραφίσει σκίτσα από διάφορες κατηγορίες, όπως π.χ. έναν ελέφαντα, μία χτένα, ή τη Μόνα Λίζα, μέσα σε χρόνο 20 δευτερολέπτων. Όσο ο χρήστης ζωγραφίζει, το εκπαιδευμένο Νευρωνικό Δίκτυο προσπαθεί να μαντέψει τι απεικονίζει το σκίτσο. Στην περίπτωση που το αναγνωρίσει σωστά, ενημερώνει τον χρήστη και το παιχνίδι προχωρά στον επόμενο γύρο. Σε περίπτωση που το ΝΔ δεν καταφέρει να προβλέψει το σκίτσο εντός του χρόνου, στο τέλος του γύρου παρουσιάζονται στον χρήστη οι τρεις πιο πιθανές προβλέψεις του.

Το παιχνίδι είναι διαθέσιμο στη νέα μορφή του στο διαδίκτυο και το σύνολο δεδομένων πλέον αποτελείται από περίπου ένα δισεκατομμύριο ασπρόμαυρα σκίτσα ζωγραφισμένα από χρήστες από όλο τον κόσμο [7] . Από το σύνολο των σκίτσων, η Google έκανε ανοιχτά διαθέσιμο προς το κοινό ένα υποσύνολο αυτών σε διαφορετικές μορφές αρχείων, αποτελούμενο από περίπου 50 εκατομμύρια σκίτσα που ανήκουν σε 345 διαφορετικές κλάσεις, με σκοπό την προώθηση της έρευνας στον τομέα της αναγνώρισης προτύπων, της ταξινόμησης εικόνων, της αναγνώρισης φυσικής γλώσσας και χειρόγραφων χαρακτήρων.

Εικόνα 1-1: Ένα αναγνωρισμένο σκίτσο από το παιχνίδι "Quick, Draw!"

[Πηγή: Experiments With Google]

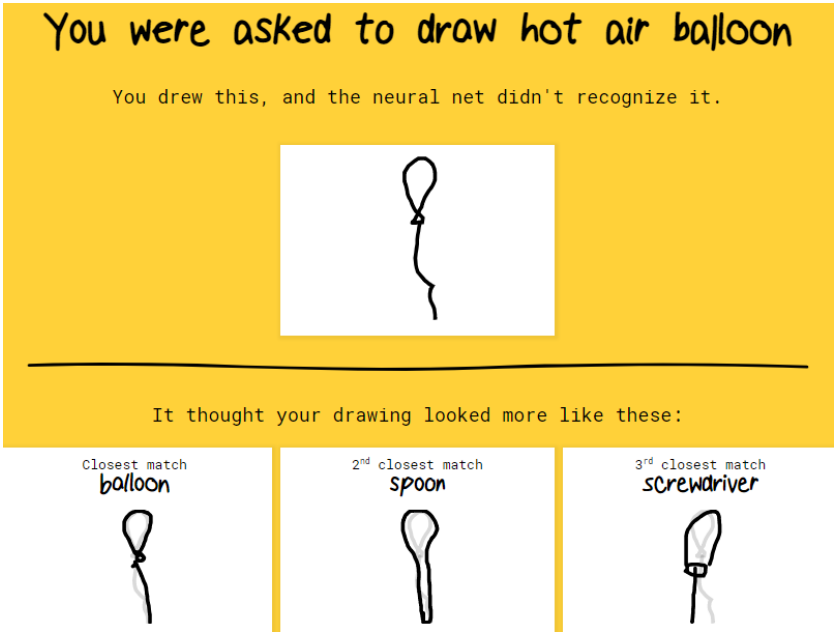
Τον Σεπτέμβριο του 2018 το σύνολο δεδομένων “Quick, Draw!” χρησιμοποιήθηκε ως βάση για τον σχηματισμό του διαγωνισμού “Quick, Draw! Doodle Recognition Challenge” από το Kaggle [8] . Δεδομένου ότι τα σκίτσα προέρχονται από το ίδιο το παιχνίδι, πολλά από αυτά είναι ημιτελή, ή απεικονίζουν κάτι διαφορετικό από την ετικέτα τους, λόγω δυσκολίας κατανόησης της αγγλικής γλώσσας από κάποιους χρήστες, ή και λόγω μιας προσπάθειας των χρηστών να μπερδέψουν το ΝΔ. Το ζητούμενο του διαγωνισμού ήταν, παρά τις διάφορες προκλήσεις που προκύπτουν από μία αλληλεπίδραση χρήστη και υπολογιστή σε επίπεδο ζωγραφικής, οι χρήστες να φτιάξουν ένα αποτελεσματικό μοντέλο αναγνώρισης των αντικειμένων στα σκίτσα, το οποίο αφού εκπαιδευτεί, να είναι σε θέση να ανταπεξέλθει αποτελεσματικά και στην αναγνώριση σκίτσων τα οποία δεν έχει συναντήσει κατά την εκπαίδευσή του. Νικητής του διαγωνισμού ήταν η ομάδα [ods.ai] Pablos, η οποία κατάφερε να πετύχει ακρίβεια τριών πρώτων προβλέψεων 95.48%. Στον διαγωνισμό “Quick, Draw! Doodle Recognition Challenge” του Kaggle συμμετείχε και μία εκδοχή από τα πειράματα που πραγματοποιήθηκαν για την παρούσα διπλωματική εργασία, πετυχαίνοντας ακρίβεια τριών πρώτων προβλέψεων 80.5% (top 79%).

Εικόνα 1-2: Μερικές από τις κλάσεις σκίτσων του συνόλου δεδομένων "Quick, Draw!" [Πηγή: Google Creative Lab on GitHub]

Πριν τη διάθεση του “Quick, Draw!” τα διαθέσιμα σύνολα δεδομένων με σκίτσα περιείχαν έναν πολύ περιορισμένο αριθμό δειγμάτων [9] και έτσι το “Quick, Draw!” από τη διάθεσή του ως και σήμερα έχει γίνει ιδιαίτερα δημοφιλές.

1.6 Ερωτήματα – Υποθέσεις

Με μια ματιά στο σύνολο δεδομένων “Quick, Draw!” γίνεται εύκολα αντιληπτό ότι η επιτυχής αποτύπωση ενός σκίτσου εξαρτάται από πολλούς παράγοντες, όπως οι ικανότητες του κάθε χρήστη στη ζωγραφική, ο βαθμός γνώσης της αγγλικής γλώσσας (κατά πόσο δηλαδή ο χρήστης αντιλαμβάνεται τι του ζητήθηκε να ζωγραφίσει), η εφαρμογή που χρησιμοποιήθηκε για την αποτύπωση του σκίτσου (διαφορετικός βαθμός δυσκολίας με τη χρήση του mouse σε έναν Η/Υ, ή με την αφή στο κινητό) και, ακόμη πιο αφηρημένα, η έμπνευση του χρήστη τη στιγμή του παιχνιδιού.



Εικόνα 1-3: Λάθος πρόβλεψη από το παιχνίδι "Quick, Draw!", λόγω σκίτσου που δεν απεικονίζει αντικείμενο της ζητούμενης κλάσης [Πηγή: Quick Draw With Google]

Με βάση τις παραπάνω παραμέτρους, προκύπτει το ζήτημα του κατά πόσο ένας χρήστης, ή ένα οποιοδήποτε άτομο, μπορεί να αναγνωρίσει με ακρίβεια ένα σκίτσο ζωγραφισμένο από έναν άλλον χρήστη. Σαφώς το ποσοστό σωστής πρόβλεψης των σκίτσων από έναν άνθρωπο είναι πολύ μεγάλο, καθώς η αντίληψή μας έχει προκύψει από την εμπειρία μας και ασυνείδητα μπορούμε να αναγνωρίσουμε άμεσα το αντικείμενο ενός επαρκώς ζωγραφισμένου σκίτσου.

Στα πλαίσια της διπλωματικής αυτής εργασίας, ένα Νευρωνικό Δίκτυο μαθαίνει να αναγνωρίζει τα χαρακτηριστικά σκίτσων που έχουν ζωγραφιστεί από χρήστες. Προκύπτει λοιπόν το ερώτημα: Είναι ικανό ένα Νευρωνικό Δίκτυο που αποκτά εμπειρία κατά την εκπαίδευσή του να αγγίξει, ή να ξεπεράσει την ανθρώπινη αντίληψη για την αποτύπωση ενός σκίτσου;

Τα αποτελέσματα της μεθοδολογίας που χρησιμοποιείται σε αυτή την εργασία έχουν ως στόχο να απαντήσουν κατά προσέγγιση το παραπάνω ερώτημα.

1.7 Συνεισφορά

Η έμπνευση για την εκπόνηση της εργασίας αυτής βασίστηκε στην αλματώδη εξέλιξη της έρευνας των τελευταίων ετών στον τομέα της επεξεργασίας και αναγνώρισης εικόνων. Παράλληλα, η επιλογή του συνόλου δεδομένων που χρησιμοποιείται στην εργασία είναι εμπνευσμένη από ένα ευρέως γνωστό σύνολο δεδομένων, που παρουσιάζει ομοιότητες με το "Quick, Draw!", το MNIST [1] .

Το MNIST είναι ένα σύνολο δεδομένων – σημείο αναφοράς στην ταξινόμηση εικόνων – το οποίο χρησιμοποιήθηκε σε συνδυασμό με το LeNet που προτάθηκε το

1998 και, όπως είναι αναμενόμενο, ειδικά με την ανάπτυξη των Συνελικτικών Νευρωνικών Δικτύων [10], τα state of the art αποτελέσματά του σήμερα βρίσκονται κοντά στο τέλειο.

Οι εικόνες του MNIST είναι ασπρόμαυρες και απεικονίζουν ψηφία από το 0 ως το 9. Η βάση αποτελείται από 60000 δείγματα εκπαίδευσης και 10000 δείγματα για testing και χρησιμοποιείται με στόχο την αντιστοίχιση/ταξινόμηση της κάθε εικόνας με τον μοναδικό αριθμό τον οποίο απεικονίζει. Παρομοίως, οι εικόνες που απαρτίζουν το σύνολο δεδομένων "Quick, Draw!" είναι ασπρόμαυρες και απεικονίζουν όχι ιδιαιτέρως πολύπλοκα σχήματα, πράγμα που το καθιστά όμοιο με το σύνολο δεδομένων MNIST. Η διαφορά των δύο συνόλων δεδομένων έγκειται στο γεγονός ότι οι κλάσεις – στόχοι του MNIST είναι μόνο 10, ενώ του "Quick, Draw!" είναι 345. Επιπλέον, στο MNIST είναι διαθέσιμα συνολικά 70000 δείγματα, ενώ στο "Quick, Draw!" περίπου 50 εκατομμύρια.

Γενικά, με τη χρήση περισσότερων δεδομένων σε οποιοδήποτε πρόβλημα Βαθιάς Μάθησης, τα αποτελέσματα που προκύπτουν καθίστανται περισσότερο ακριβή και μπορούν να γενικευτούν καλύτερα και συνεπώς προκύπτει μία πιο ολοκληρωμένη και σταθερή λύση για το εκάστοτε πρόβλημα. Η ταξινόμηση των εικόνων του MNIST είχε ως αποτέλεσμα την προώθηση της έρευνας στον τομέα τόσο της αναγνώρισης φυσικής γλώσσας, όσο και της αναγνώρισης εικόνας. Έτσι, με τη χρήση του "Quick, Draw!" επιδιώκεται η επέκταση του προβλήματος ταξινόμησης χειρόγραφων ψηφίων, απλών σχημάτων και διάφορων σκίτσων, με στόχο την παρουσίαση ενός μοντέλου που θα διαθέτει γνώση για πολύ περισσότερα και πιο σύνθετα σχήματα από τα 10 ψηφία του MNIST.

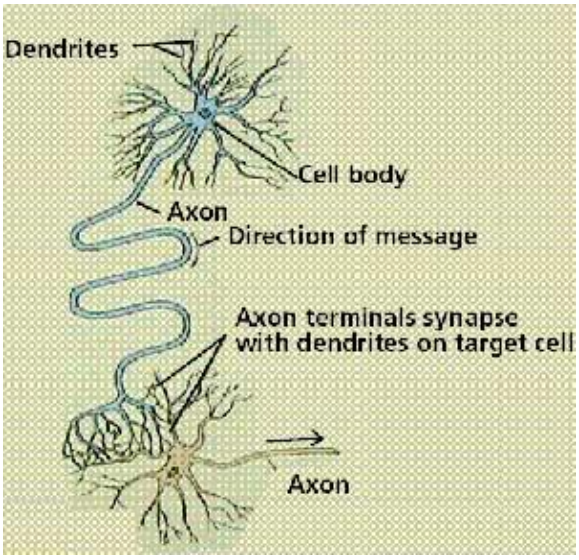
2 Νευρωνικά Δίκτυα

Στο κεφάλαιο αυτό γίνεται μία παρουσίαση των Νευρωνικών Δικτύων. Ξεκινώντας με κάποιες στοιχειώδεις πληροφορίες που σχετίζονται με τη νευροβιολογία, περιγράφονται σύντομα τα Βιολογικά Νευρωνικά Δίκτυα, με στόχο την ομαλότερη εισαγωγή του αναγνώστη σε έννοιες που αφορούν τα Τεχνητά Νευρωνικά Δίκτυα. Αναλύεται, τέλος, η δομή των Τεχνητών Νευρωνικών Δικτύων, προκειμένου ο αναγνώστης να αποκτήσει εξοικείωση με τις σχετικές έννοιες που απαιτούνται για την κατανόηση των επόμενων κεφαλαίων.

2.1 Βιολογικοί Νευρώνες

Για την ευκολότερη κατανόηση των Τεχνητών Νευρωνικών Δικτύων, είναι σημαντικό να αναφερθούν κάποιες βασικές έννοιες από το πεδίο της νευροβιολογίας.

Ένα από τα συστήματα που διαθέτει ο ανθρώπινος οργανισμός είναι το Νευρικό Σύστημα, το οποίο είναι υπεύθυνο για διάφορες λειτουργίες του οργανισμού, όπως η μεταβίβαση πληροφοριών, η μάθηση, η μνήμη και η αλληλεπίδραση με το εξωτερικό περιβάλλον. Η βασική μονάδα του συστήματος αυτού είναι ο εγκέφαλος και αποτελείται από ένα σύνθετο δίκτυο ειδικών κυττάρων, που ονομάζονται νευρώνες. Στον ανθρώπινο εγκέφαλο εκτιμάται ότι υπάρχουν δισεκατομμύρια νευρώνες, οι οποίοι συνδέονται με άλλους νευρώνες προκειμένου να μεταφέρουν και να επεξεργαστούν πληροφορίες.



Εικόνα 2-1: Βιολογικός νευρώνας [Πηγή: Stanford]

Όπως φαίνεται και στην εικόνα 2.1, ο βιολογικός νευρώνας αποτελείται από τον πυρήνα του, που ονομάζεται σώμα (cell body), τους δενδρίτες (dendrites), τον άξονα (axon) και τις συνάψεις (synapses). Οι δενδρίτες είναι η είσοδος του νευρώνα, ενώ ο άξονας είναι η έξοδός του. Οι δενδρίτες ενός νευρώνα λαμβάνουν ηλεκτροχημικές εισόδους από άλλους νευρώνες μέσω των συνάψεων. Αν το σώμα κρίνει ότι το άθροισμα των εισόδων αυτών είναι επαρκές, τότε μεταφέρεται ένα ηλεκτροχημικό σήμα μέσω του άξονα και των συνάψεων του νευρώνα στους δενδρίτες – παραλήπτες άλλων συνδεδεμένων νευρώνων. Συνεπώς, αν το συνολικό φορτίο εισόδου ξεπερνά ένα συγκεκριμένο επίπεδο (κατώφλι), ενεργοποιούνται οι συνδεδεμένοι νευρώνες, διαφορετικά δεν ενεργοποιούνται [11] .

2.2 Τεχνητοί Νευρώνες

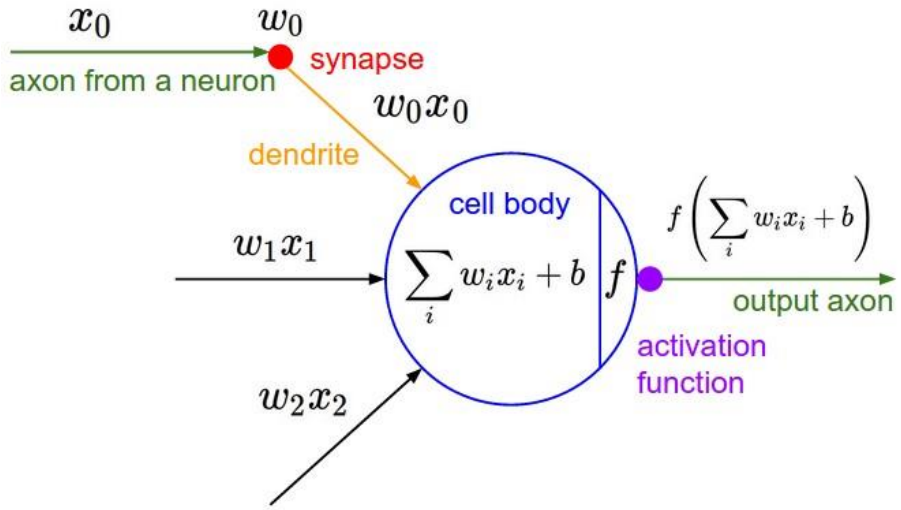
Η έμπνευση για τα Τεχνητά Νευρωνικά Δίκτυα (ή απλώς Νευρωνικά Δίκτυα) προέρχεται από το Κεντρικό Νευρικό Σύστημα του ανθρώπου, καθώς η πρώτη προσέγγισή τους αφορούσε μία μορφή μοντελοποίησης της λειτουργίας του εγκεφάλου (McCulloch,Pitts) [12] .

Το πιο απλό νευρωνικό δίκτυο αποτελείται από μόλις έναν νευρώνα και προτάθηκε το 1958 από τον Rosenblatt [13] . Το νευρωνικό αυτό δίκτυο, γνωστό ως μοντέλο απλού αισθητήρα, ή perceptron, αποτελεί στην ουσία το μαθηματικό μοντέλο ενός βιολογικού νευρώνα και συνεπώς η δομή του μπορεί να αντιστοιχιστεί πλήρως με αυτή ενός βιολογικού νευρώνα. Όμοια με τον βιολογικό νευρώνα, ο μοναδικός νευρώνας του perceptron δέχεται πολλές εισόδους και έχει μία έξοδο.

Ο τεχνητός νευρώνας αποτελείται από έναν αριθμό εισόδων x_1, x_2, …, x_m, κάθε μία από τις οποίες πολλαπλασιάζεται με την τιμή του αντίστοιχου συναπτικού βάρους w_1, w_2, …, w_m. Στη συνέχεια οι σταθμισμένες είσοδοι και ένας εξωτερικός παράγοντας b (bias, πόλωση) αθροίζονται και το αποτέλεσμα αυτής της πράξης αποτελεί την είσοδο για μία συνάρτηση ενεργοποίησης φ, με βάση την οποία προκύπτει η έξοδος $f(x)$ του νευρωνικού δικτύου. Συνεπώς, η έξοδος περιγράφεται από την εξίσωση:

$$f(x) = b + \sum_{i=1}^{n} x_i \cdot w_i$$

Στην εικόνα 2.3 φαίνεται αναλυτικότερα η δομή του τεχνητού νευρώνα, όπου απεικονίζονται οι είσοδοι, τα βάρη, η πόλωση και η συνάρτηση ενεργοποίησης σε παραλληλισμό με τα δομικά στοιχεία του βιολογικού νευρώνα:



Εικόνα 2-2: Μοντέλο Τεχνητού Νευρώνα σε αντιστοίχιση με τον βιολογικό νευρώνα [Πηγή: Stanford on GitHub]

Είσοδοι – Inputs.
Κάθε νευρώνας έχει πολλές εισόδους x. Η κάθε μία από αυτές αναπαρίσταται από μία αριθμητική τιμή και συνδέεται με ένα βάρος w.

Βάρη – Weights.
Σε κάθε είσοδο του νευρώνα προσάπτεται ένα βάρος w, το οποίο επίσης αναπαρίσταται από κάποια αριθμητική τιμή. Οι τιμές των βαρών μεταβάλλονται συνεχώς κατά τη διαδικασία της μάθησης και με αυτόν τον τρόπο καθορίζεται σε γενικές γραμμές ο βαθμός που η κάθε είσοδος επηρεάζει την έξοδο του νευρώνα.

Πόλωση – Bias
Η πόλωση b είναι ένας παράγοντας που μπορεί να περιγραφεί ως μία σταθερή είσοδος που συνδέεται με τα βάρη. Η παράμετρος αυτή είναι σημαντική, αφού επηρεάζει συνολικά την απόδοση του νευρώνα και συνεπώς την επιτυχία της μάθησης.

Συνάρτηση Ενεργοποίησης – Activation Function
Η συνάρτηση ενεργοποίησης φ είναι υπεύθυνη για τον μετασχηματισμό των εισόδων του νευρώνα. Υπάρχουν διαφορετικά είδη συναρτήσεων ενεργοποίησης που αναλύονται στη συνέχεια. Στο μοντέλο perceptron η συνάρτηση ενεργοποίησης είναι βηματική (έξοδος 0 ή 1) και ορίζεται ως:

$$\varphi(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & else \end{cases}$$

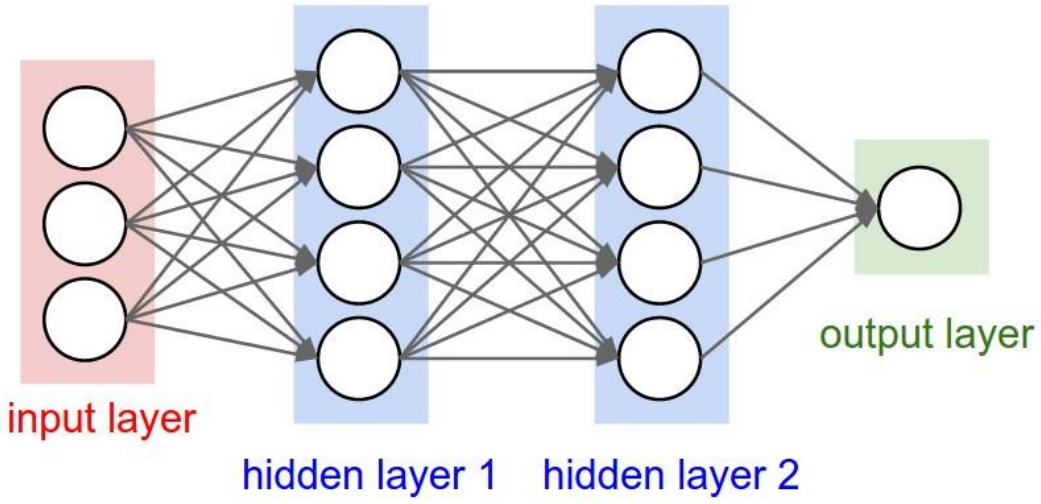
Προκύπτει λοιπόν, ότι το μοντέλο perceptron είναι ένας δυαδικός ταξινομητής (binary classifier), χρησιμοποιείται δηλαδή για την ταξινόμηση του αθροίσματος σε μία κλάση Α, αν είναι μεγαλύτερο ή ίσο του μηδενός, ή σε μία δεύτερη κλάση Β διαφορετικά.

2.3 Νευρωνικά Δίκτυα πολλών επιπέδων

Μπορεί να γίνει αντιληπτό ότι το απλό ΝΔ που παρουσιάστηκε στην προηγούμενη υποενότητα, δε μπορεί να χρησιμοποιηθεί για την ταξινόμηση δεδομένων που δεν είναι γραμμικώς διαχωρίσιμα. Έτσι, χρησιμοποιώντας ως βάση το μοντέλο perceptron αναπτύχθηκαν σταδιακά πιο σύνθετες αρχιτεκτονικές ΝΔ, οι οποίες είναι ικανές να αποθηκεύουν περισσότερη πληροφορία και συνεπώς να επιλύουν πιο σύνθετα προβλήματα.

Εξελίσσοντας το μοντέλο perceptron προέκυψαν τα ΝΔ perceptron πολλαπλών επιπέδων (Multilayer perceptron, MLP), τα οποία αποτελούνται επιπλέον από ένα ή περισσότερα κρυφά επίπεδα (hidden layers). Τα MLP είναι δίκτυα εμπρόσθιας τροφοδότησης (feedforward), δηλαδή οι νευρώνες ενός επιπέδου δε συνδέονται με τους νευρώνες προηγούμενου επιπέδου (η πληροφορία ρέει προς τα μπροστά) και είναι ικανά να επιλύουν μη-γραμμικά προβλήματα. Τα κρυφά επίπεδα των MLP είναι υπεύθυνα για τη διαδικασία της μάθησης, καθώς καθορίζουν ποιες είσοδοι έχουν μεγαλύτερη βαρύτητα για την έξοδο του νευρωνικού δικτύου.

Στην εικόνα 2.4 παρουσιάζεται η δομή ενός ΝΔ τριών επιπέδων, που αποτελείται από τρεις εισόδους, δύο κρυφά επίπεδα και μία έξοδο Όπως φαίνεται υπάρχουν συνδέσεις μεταξύ των επιπέδων, αλλά όχι μεταξύ των νευρώνων του ίδιου επιπέδου.



Εικόνα 2-3: Νευρωνικό Δίκτυο τριών επιπέδων, με τρεις εισόδους, δύο κρυφά επίπεδα και μία έξοδο [Πηγή: Stanford on GitHub]

Σε ένα ΝΔ πολλών επιπέδων τα κρυφά επίπεδα αποτελούνται από μη - γραμμικούς νευρώνες. Δέχονται σήματα από το επίπεδο εισόδου και είναι ικανά να εφαρμόζουν σε αυτά μη – γραμμικούς μετασχηματισμούς, επηρεάζοντας έτσι την έξοδο του νευρωνικού δικτύου. Το επίπεδο εξόδου μπορεί να αποτελείται είτε από γραμμικούς, είτε από μη – γραμμικούς νευρώνες, ή και από συνδυασμό των δύο.

2.3.1 Συναρτήσεις Ενεργοποίησης

Η έξοδος ενός ΝΔ καθορίζεται από τις συναρτήσεις ενεργοποίησης (activation functions), οι οποίες είναι υπεύθυνες για τον μη – γραμμικό μετασχηματισμό των δεδομένων που εισάγονται σε ένα ΝΔ. Παρακάτω αναφέρονται κάποιες από τις πιο διαδεδομένες μη – γραμμικές συναρτήσεις ενεργοποίησης.

Sigmoid (ή Logistic) Activation Function

Η σιγμοειδής συνάρτηση, είναι μία μη – γραμμική συνάρτηση ενεργοποίησης η οποία δέχεται ως είσοδο έναν πραγματικό αριθμό και ως έξοδο επιστρέφει τον αριθμό αυτό εντός του διαστήματος [0,1]. Η συνάρτηση αυτή χρησιμοποιείται κυρίως από μοντέλα όπου το ζητούμενο αποτέλεσμα είναι κάποια πιθανότητα, καθώς όλες οι πιθανότητες εξορισμού βρίσκονται στο διάστημα [0,1]. Η σχέση που περιγράφει τη σιγμοειδή συνάρτηση ενεργοποίησης φαίνεται παρακάτω:

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

Tanh ή Hyperbolic Tangent Activation Function

Η συνάρτηση υπερβολικής εφαπτομένης δέχεται ως είσοδο έναν πραγματικό αριθμό και ως έξοδο δίνει τον αριθμό αυτό εντός του διαστήματος [-1,1]. Η συνάρτηση αυτή χρησιμοποιείται κυρίως σε προβλήματα ταξινόμησης δεδομένων σε δύο κλάσεις. Ο τύπος της φαίνεται παρακάτω:

$$\varphi(x) = \tanh(x)$$

ReLU Activation Function

Η συνάρτηση ReLU (Rectified Linear Unit) είναι μία μη – γραμμική συνάρτηση ενεργοποίησης η οποία είναι ίσως η πιο δημοφιλής στον τομέα της βαθιάς μάθησης και των Συνελικτικών Νευρωνικών Δικτύων τα τελευταία χρόνια. Στην πράξη δέχεται ως είσοδο έναν αριθμό και σε περίπτωση που είναι αρνητικός η έξοδος ισούται με ο, διαφορετικά ισούται με τον ίδιο τον αριθμό. Ο λόγος για τον οποίο χρησιμοποιείται πολύ τα τελευταία χρόνια είναι επειδή επιταχύνει την εύρεση των βέλτιστων τιμών από τον αλγόριθμο gradient descent συγκριτικά με τις συναρτήσεις ενεργοποίησης που αναφέρθηκαν προηγουμένως. Ιδανικά χρησιμοποιείται σε βαθιά νευρωνικά δίκτυα με πολλά επίπεδα, καθώς έχει την ιδιότητα να περιορίζει κατά πολύ τον αριθμό των νευρώνων που ενεργοποιούνται, με αποτέλεσμα το ΝΔ να γίνεται ελαφρύτερο και να επιταχύνονται ιδιαίτερα οι υπολογισμοί. Η σχέση η οποία περιγράφει την ReLU φαίνεται παρακάτω:

$$\varphi(x) = \max(0, x)$$

Softmax Function

Η συνάρτηση softmax χρησιμοποιείται σε προβλήματα ταξινόμησης, καθώς μετατρέπει την έξοδο του ΝΔ σε τιμές ανάμεσα σε 0 και 1 , όπως η σιγμοειδής συνάρτηση, αλλά επιπλέον το συνολικό άθροισμα όλων των εξόδων είναι 1. Η έξοδός της δείχνει δηλαδή την πιθανότητα του κάθε παραδείγματος να ανήκει σε κάθε κλάση, είναι ιδανική επομένως για προβλήματα ταξινόμησης σε πολλές κλάσεις. Η σχέση που εκφράζει τη συνάρτηση ενεργοποίησης softmax καθώς και ένα παράδειγμα εξόδου της φαίνονται παρακάτω:

$$\varphi(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Εικόνα 2-4: Συνάρτηση Ενεργοποίησης Softmax

2.3.2 Συναρτήσεις Κόστους

Στη μηχανική και τη βαθιά μάθηση οι συναρτήσεις κόστους (loss ή cost functions) χρησιμοποιούνται προκειμένου να γίνεται μία εκτίμηση σχετικά με το πόσο καλή επίδοση έχει ένα ΝΔ. Μία συνάρτηση κόστους, δηλαδή, είναι στην ουσία ένα μέτρο, το οποίο εκφράζει το πόσο λάθος μπορεί να είναι το μοντέλο του ΝΔ που χρησιμοποιείται, σχετικά με την ικανότητά του να συσχετίζει τα δεδομένα που δέχεται ως είσοδο, με την έξοδό του. Η συσχέτιση αυτή υπολογίζεται ουσιαστικά από την απόσταση ή διαφορά της τιμής που έχει προβλέψει το ΝΔ από την πραγματική τιμή των δεδομένων.

Οι συναρτήσεις κόστους θα μπορούσαν να ταξινομηθούν σε δύο κατηγορίες, ανάλογα με τη φύση του προβλήματος στο οποίο χρησιμοποιούνται. Η πρώτη κατηγορία αναφέρεται στα προβλήματα regression, δηλαδή στα προβλήματα όπου το μοντέλο επιδιώκει να προβλέψει κάποια συνεχή τιμή. Η κατηγορία αυτή των συναρτήσεων κόστους δεν αναλύεται στην παρούσα διπλωματική, ωστόσο επιγραμματικά αναφέρονται οι συναρτήσεις κόστους Mean Squared Error (MSE/L2 loss), η Mean Absolute Error (MAE/ L1 Loss) και η Mean Bias Error (MBE).

Η δεύτερη κατηγορία αναφέρεται στα προβλήματα ταξινόμησης (classification problems), σε προβλήματα δηλαδή που επιδιώκουν να προβλέψουν την πιθανότητα ένα παράδειγμα x_i να έχει ετικέτα y_i. Η πιο κοινή συνάρτηση

κόστους που χρησιμοποιείται σε προβλήματα ταξινόμησης είναι η συνάρτηση Cross Entropy.

Cross Entropy Cost Function

Η συνάρτηση κόστους cross entropy χρησιμοποιείται για να μετρήσει την απόδοση ενός μοντέλου ταξινόμησης, του οποίου η έξοδος είναι μία κατανομή πιθανότητας, επομένως είναι ιδανική για μοντέλα ΝΔ που στην έξοδό τους έχουν έναν ταξινομητή softmax. Η τιμή της συνάρτησης αυξάνεται όταν η προβλεπόμενη πιθανότητα ένα συγκεκριμένο παράδειγμα να ανήκει σε κάποια συγκεκριμένη κλάση απομακρύνεται από την πραγματική της ετικέτα. Η σχέση που περιγράφει τη συνάρτηση Cross Entropy:

$$CrossEntropyLoss = -\sum_{i}^{C} t_i \log(s_i)$$

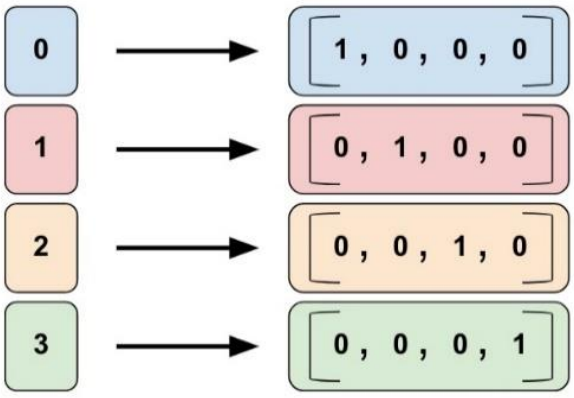
για ένα πρόβλημα ταξινόμησης δεδομένων σε C κλάσεις, όπου t_i και s_i είναι η πραγματική ετικέτα (true label) και το σκορ που έχει προβλεφθεί για κάθε κλάση από i μέχρι C. Σε προβλήματα όπου το ζητούμενο είναι η ταξινόμηση των δεδομένων σε δύο μόνο κλάσεις (binary classification problem) η συνάρτηση γίνεται:

$$BinaryCrossEntropyLoss = -\sum_{i=1}^{C=2} t_i \log(s_i) = -(t_1 \log(s_1) + (1 - t_1) \log(1 - s_1))$$

Τέλος, για τα προβλήματα με ζητούμενο την ταξινόμηση των δεδομένων σε πολλές κλάσεις, όπου το κάθε παράδειγμα ανήκει σε μία μόνο κλάση (multi – label classification problem) η συνάρτηση γίνεται:

$$CategoricalCrossEntropyLoss = -\log \frac{e^{s_p}}{\sum_j^C e^{s_j}}$$

Σε αυτού του τύπου τα προβλήματα συνήθως οι ετικέτες είναι κωδικοποιημένες έτσι ώστε η τιμή εξόδου που αντιστοιχεί σε κάθε παράδειγμα να είναι ένα διάνυσμα μεγέθους C, όπου όλα τα στοιχεία είναι ίσα με 0, εκτός από το στοιχείο που δείχνει τη θέση της κλάσης στην οποία ανήκει το παράδειγμα, το οποίο έχει τιμή ίση με 1. Με τον τρόπο αυτό, διατηρείται μόνο η τιμή της θετικής κλάσης (C_p) στον γενικό τύπο της συνάρτησης Cross Entropy, προκύπτει ο τύπος Categorical Cross Entropy. Η κωδικοποίηση αυτή ονομάζεται one – hot encoding και ένα παράδειγμά της φαίνεται παρακάτω:



Εικόνα 2-5: Παράδειγμα One - Hot Encoding [Πηγή: TensorFlow]

2.3.3 Διαδικασία Μάθησης – Αλγόριθμος Backpropagation

Ο ανθρώπινος εγκέφαλος δέχεται πληροφορίες από το εξωτερικό περιβάλλον και με βάση αυτές σχηματίζει μία συμπεριφορά αντιδράσεων. Η συμπεριφορά αυτή είναι αποτέλεσμα της διαδικασίας μάθησης που συμβαίνει στον ανθρώπινο εγκέφαλο και όπως είναι φυσικό, η συνεχής μάθηση κατά τη διάρκεια της ζωής ενός ανθρώπου, έχει ως αποτέλεσμα την αλλαγή, ή την προσαρμογή των αντιδράσεων του στα ίδια ερεθίσματα. Η προσπάθεια μίμησης της διαδικασίας αυτής από τους υπολογιστές αποτελεί έναν τομέα της Τεχνητής Νοημοσύνης, τη Μηχανική Μάθηση, η οποία τα τελευταία χρόνια παρουσιάζει ιδιαίτερο ενδιαφέρον.

Η διαδικασία της μάθησης στα πλαίσια των ΝΔ είναι στην ουσία ένας αλγόριθμος εκπαίδευσης, σύμφωνα με τον οποίο το ΝΔ μαθαίνει να προσαρμόζεται στις πληροφορίες που δέχεται από το εξωτερικό περιβάλλον, μέσω της ανανέωσης των τιμών των βαρών του. Ο τρόπος με τον οποίο μεταβάλλονται τα βάρη καθορίζει τον αλγόριθμο μάθησης που χρησιμοποιεί το ΝΔ.

Ο πιο διαδεδομένος αλγόριθμος μάθησης είναι ο αλγόριθμος Οπισθοδιάδοσης Σφάλματος (Backpropagation Algorithm). Σύμφωνα με αυτόν δίνονται ως είσοδοι στο ΝΔ κάποια παραδείγματα εκπαίδευσης και η μάθηση προκύπτει μέσω μίας επαναληπτικής διαδικασίας κατά τη διάρκεια της οποίας οι παράμετροι του ΝΔ μεταβάλλονται συνεχώς, με στόχο την καλύτερη επίδοση του δικτύου.

Αρχικά το ΝΔ δεν είναι εκπαιδευμένο και συνεπώς δεν γνωρίζει κάποιον τρόπο για να διαχειριστεί τις εισόδους του, οπότε οι παράμετροί του αρχικοποιούνται τυχαία. Στη συνέχεια, προκειμένου να βελτιωθούν οι τιμές των παραμέτρων αξιολογείται επαναληπτικά η συνάρτηση σφάλματος (loss/cost function) με τη σύγκριση των παραδειγμάτων εκπαίδευσης και των επιθυμητών εξόδων του ΝΔ. Η αποτελεσματικότητα του αλγορίθμου backpropagation οφείλεται στη χρήση μεθόδων βελτιστοποίησης gradient descent, οι οποίες εντοπίζουν τις βέλτιστες τιμές για τα βάρη του ΝΔ. Ο συνδυασμός των βέλτιστων τιμών των βαρών που ελαχιστοποιούν τη συνάρτηση σφάλματος αποτελεί τη λύση στο πρόβλημα βελτιστοποίησης.

Ο αλγόριθμος backpropagation αποτελείται από δύο φάσεις, τη φάση της εμπρόσθιας διάδοσης και τη φάση της οπισθοδιάδοσης. Κατά τη φάση της εμπρόσθιας διάδοσης, τα παραδείγματα εκπαίδευσης παρουσιάζονται στο ΝΔ από επίπεδο σε επίπεδο σε όλους τους νευρώνες, με φορά από την είσοδο προς την έξοδο. Στη φάση της οπισθοδιάδοσης η έξοδος που προέκυψε από την προηγούμενη φάση συγκρίνεται με την είσοδο και με βάση τη διαφορά τους υπολογίζεται το σφάλμα. Στη συνέχεια, διαδίδεται το σήμα του σφάλματος με κατεύθυνση από το επίπεδο εξόδου προς το επίπεδο εισόδου και για όλους τους νευρώνες του ΝΔ υπολογίζεται η συμβολή τους στα σφάλματα των νευρώνων εξόδου, προκειμένου να γίνει η αναπροσαρμογή των βαρών. Η διαδικασία αυτή επαναλαμβάνεται μέχρι τα βάρη να αποκτήσουν τις βέλτιστες τιμές τους.

2.3.4 **Αλγόριθμοι Βελτιστοποίησης**

Οι αλγόριθμοι βελτιστοποίησης στα πλαίσια των ΝΔ χρησιμοποιούνται για την ελαχιστοποίηση της συνάρτησης κόστους. Αυτή εξαρτάται από τις εσωτερικές παραμέτρους του ΝΔ, οι οποίες συμμετέχουν στους υπολογισμούς για την αξιολόγηση των αποτελεσμάτων με βάση την επιθυμητή έξοδο. Όπως προκύπτει, ο ρόλος των εσωτερικών παραμέτρων ενός ΝΔ είναι ιδιαίτερα σημαντικός για τη διαδικασία της μάθησης και για αυτόν τον λόγο έχουν αναπτυχθεί πολλοί και διαφορετικοί αλγόριθμοι βελτιστοποίησης για την ανανέωση και την βελτιστοποίηση των τιμών τους.

Οι πιο ευρέως χρησιμοποιούμενοι αλγόριθμοι βελτιστοποίησης στο πεδίο της Μηχανικής Μάθησης είναι οι αλγόριθμοι της ομάδας gradient descent και βασίζονται στον υπολογισμό της κλίσης (gradient) της συνάρτησης κόστους, σε σχέση με τις παραμέτρους(βάρη, πόλωση). Χρησιμοποιείται κατά τη φάση της οπισθοδιάδοσης του αλγορίθμου backpropagation, ανανεώνοντας σε κάθε επανάληψη τις εσωτερικές παραμέτρους του ΝΔ, με κατεύθυνση αντίθετη της κλίσης της συνάρτησης σφάλματος. Η κλίση υπολογίζεται με τη χρήση μερικών παραγώγων, λόγω της εξάρτησης της συνάρτησης κόστους από πολλές μεταβλητές, και αναπαρίσταται από έναν πίνακα (διανυσματικό πεδίο) αποτελούμενο από μερικές παραγώγους πρώτης τάξης.

Ο ρυθμός με τον οποίο ανανεώνονται τα βάρη σε κάθε επανάληψη, μέχρι να βρεθεί το τοπικό ελάχιστο, ονομάζεται ρυθμός μάθησης α (learning rate) και στην ουσία σηματοδοτεί το μέγεθος του βήματος του αλγορίθμου gradient descent προς την κατεύθυνση του τοπικού ελάχιστου. Η τιμή του ρυθμού μάθησης είναι καθοριστική για το αποτέλεσμα, καθώς όταν είναι πολύ μεγάλη, υπάρχει ο κίνδυνος το τοπικό ελάχιστο να προσπερνάται σε κάθε βήμα, ενώ όταν είναι πολύ μικρός, ο αλγόριθμος θα καθυστερήσει σημαντικά να το εντοπίσει.

Αν και αποτελεσματικός όσον αφορά την εύρεση του τοπικού ελάχιστου, ο κλασικός αλγόριθμος (batch) gradient descent υπολογίζει την κλίση και το κόστος για ολόκληρο το σύνολο δεδομένων σε κάθε επανάληψη, για μία μόνο ανανέωση των παραμέτρων. Αν ϑ είναι οι παράμετροι της συνάρτησης κόστους $J(\vartheta)$, η ανανέωση των παραμέτρων περιγράφεται από την εξίσωση:

$$\vartheta = \vartheta - \alpha \cdot \nabla J(\vartheta)$$

Ως αποτέλεσμα αυτού, για πολύ μεγάλα σύνολα δεδομένων, η διαδικασία της μάθησης μπορεί να καθυστερήσει σημαντικά. Για αυτόν το λόγο αναπτύχθηκαν διάφορες βελτιστοποιημένες παραλλαγές του αλγορίθμου gradient descent, οι κυριότερες από τις οποίες αναφέρονται παρακάτω.

Stochastic Gradient Descent
Ο αλγόριθμος Stochastic Gradient Descent, ή SGD, σε αντίθεση με τον κλασικό gradient descent, υπολογίζει την κλίση των παραμέτρων χρησιμοποιώντας μόνο ένα παράδειγμα εκπαίδευσης κάθε φορά, με αποτέλεσμα να είναι συνήθως γρηγορότερος. Η ανανέωση των παραμέτρων στον SGD περιγράφεται από την εξίσωση:

$$\vartheta = \vartheta - \alpha \cdot \nabla J\big(\vartheta; x(i), y(i)\big), \text{ όπου}$$
$$\{x(i), y(i)\} \text{ ένα ζεύγος από το σύνολο εκπαίδευσης}$$

Η συχνή αυτή ανανέωση των παραμέτρων οδηγεί στην εύρεση ενός καλύτερου τοπικού ελάχιστου, αλλά παράλληλα προκαλεί έντονη διακύμανση στη συνάρτηση κόστους [14] .

Mini Batch Gradient Descent
Αυτή η παραλλαγή του αλγορίθμου gradient descent διαφέρει από τον SGD στο σημείο ότι ανανεώνει τις τιμές των παραμέτρων, χρησιμοποιώντας αντί για ένα παράδειγμα, ένα πακέτο (minibatch) παραδειγμάτων εκπαίδευσης κάθε φορά. Με αυτόν τον τρόπο μειώνεται η διακύμανση της ενημέρωσης παραμέτρων και αυτό οδηγεί σε πιο σταθερή σύγκλιση.

Η επιλογή του μεγέθους του πακέτου παραδειγμάτων εκπαίδευσης (minibatch size) εξαρτάται από τη φύση του προβλήματος και της αρχιτεκτονικής που χρησιμοποιείται. Οι τιμές του μπορεί τυπικά να κυμαίνονται από 64 ως 256, αλλά σε γενικές γραμμές επιλέγεται μετά από δοκιμές μία τιμή που οδηγεί σε καλύτερα αποτελέσματα.

Τόσο ο αλγόριθμος Stochastic Gradient Descent, όσο και ο Mini Batch Gradient Descent, αναφέρονται ως αλγόριθμοι gradient descent. Πάνω στις παραλλαγές του αλγορίθμου gradient descent έχουν προταθεί και κάποιες προσθήκες που τις βελτιστοποιούν περεταίρω, οι σημαντικότερες από τις οποίες αναφέρονται συνοπτικά παρακάτω:

Momentum, όπου χρησιμοποιείται ένας όρος ρ για την αναπαράσταση της ορμής, δηλαδή της σχέσης που έχει η προηγούμενη ανανέωση των παραμέτρων με την επόμενη [15]. Με τη χρήση του Momentum αποφεύγονται οι περιττές ανανεώσεις παραμέτρων και έτσι επιτυγχάνεται συνήθως γρηγορότερη σύγκλιση.

AdaGrad, τεχνική η οποία οδηγεί τον ρυθμό μάθησης *α* στο να προσαρμόζεται με βάση το άθροισμα των τετραγώνων των προηγούμενων κλίσεων που υπολογίστηκαν για κάθε παράμετρο [16]. Χρησιμοποιούνται δηλαδή προσαρμοστικές τιμές ρυθμού μάθησης, πράγμα που εξαλείφει την ανάγκη ορισμού της τιμής *α* αυθαίρετα. Ωστόσο, το πρόβλημα της τεχνικής αυτής είναι ότι ο ρυθμός μάθησης μειώνεται συνεχώς με αποτέλεσμα κάποια στιγμή το σύστημα σταματά να μαθαίνει.

AdaDelta, που αποτελεί μία βελτίωση του AdaGrad, καθώς εξαλείφει το πρόβλημα της συνεχούς μείωσης του ρυθμού μάθησης. Η βελτίωση οφείλεται στο γεγονός ότι ο αλγόριθμος AdaDelta περιορίζει τον αριθμό των συσσωρευμένων κλίσεων που υπολογίστηκαν στο παρελθόν, σε κάποια σταθερή τιμή *w* [17].

RMSProp, άλλη μία παραλλαγή του AdaGrad, σύμφωνα με την οποία και πάλι συσσωρεύεται το ιστορικό των τετραγώνων των κλίσεων για μία παράμετρο, με τη διαφορά ότι ο RMSProp δίνει μεγαλύτερη βαρύτητα στους υπολογισμούς του πρόσφατου παρελθόντος. Αυτό επιτυγχάνεται με τη χρήση ενός όρου (decay rate), ο οποίος χρησιμοποιείται για να εκφράσει το πόσο πρόσφατο ή παλιό είναι το ιστορικό που συσσωρεύτηκε για την παράμετρο [18] .

Adam, ή Adaptive Moment Estimation, που συνδυάζει τα πλεονεκτήματα του AdaGrad και του RMSProp. Ο Adam χρησιμοποιεί επίσης προσαρμοστικούς ρυθμούς μάθησης για κάθε παράμετρο και αποθηκεύει το ιστορικό των τετραγώνων των κλίσεων. Ωστόσο, αποθηκεύει επιπλέον και το ιστορικό των κλίσεων, όπως συμβαίνει στον Momentum. Ως αποτέλεσμα, ο Adam οδηγεί σε πολύ γρήγορη σύγκλιση και συνεπώς σε πιο γρήγορη και αποτελεσματική διαδικασία μάθησης [19].

2.4 Συνελικτικά Νευρωνικά Δίκτυα

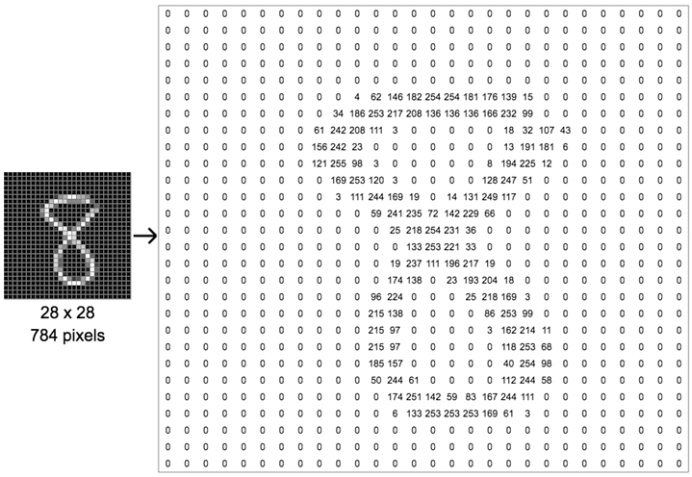
Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks – CNNs - ΣΝΔ) είναι ένας ειδικός τύπος ΝΔ πολλών επιπέδων, τα οποία έχουν σχεδιαστεί προκειμένου να αναγνωρίζουν μοτίβα απευθείας από εικόνες. Τα ΣΝΔ χρησιμοποιούνται σε πολλά είδη εφαρμογών, με ένα τυπικό παράδειγμα να είναι η αναγνώριση αντικειμένων σε εικόνες.

Μία εικόνα είναι για ένα ΝΔ ένας πίνακας, του οποίου κάθε στοιχείο αναπαριστά ένα pixel της εικόνας. Το κάθε pixel περιέχει πληροφορία για το χρώμα της σε εκείνο το σημείο και ανάλογα με το αν η εικόνα είναι έγχρωμη ή ασπρόμαυρη, χαρακτηρίζεται από διαφορετικό αριθμό καναλιών (channels).

Στην περίπτωση της έγχρωμης (RGB) εικόνας υπάρχουν τρία κανάλια, ένα για κάθε χρώμα – κόκκινο (R), πράσινο (G), μπλε (B) – τα οποία είναι στην ουσία τρεις πίνακες δύο διαστάσεων, στοιβαγμένοι ο ένας πάνω στον άλλο, με κάθε έναν να περιέχει στοιχεία με τιμές στο διάστημα [0,255], που περιγράφουν τη φωτεινότητα του pixel. Οι ασπρόμαυρες εικόνες (grayscale images), από την άλλη πλευρά, διαθέτουν ένα κανάλι και άρα περιγράφονται από έναν πίνακα δύο

διαστάσεων, με τιμές στο πεδίο [0,255], όπου το 0 αναπαριστά το μαύρο χρώμα, το 255 το άσπρο και οι ενδιάμεσες τιμές τις διάφορες διαβαθμίσεις του γκρι.

Στα πλαίσια της διπλωματικής αυτής εργασίας εξετάζεται η χρήση των ΣΝΔ σε εφαρμογές αναγνώρισης αντικειμένων σε ασπρόμαυρες εικόνες δύο διαστάσεων.



Εικόνα 2-6: Αναπαράσταση ασπρόμαυρης εικόνας (1 κανάλι) [Πηγή: Medium]

Σε γενικές γραμμές, ένα ΣΝΔ για να πραγματοποιήσει υπολογισμούς και να εξάγει συμπεράσματα για μία εικόνα, τη διαχειρίζεται χρησιμοποιώντας μικρότερες περιοχές της. Κάθε πλέγμα της που επεξεργάζεται αναπαριστά κάποιο διαφορετικό χαρακτηριστικό της εικόνας, όπως για παράδειγμα ακμές, γραμμές ή καμπύλες. Με βάση τα χαρακτηριστικά αυτά δημιουργούνται χάρτες χαρακτηριστικών (feature maps), οι οποίοι στη συνέχεια χρησιμοποιούνται συνδυαστικά, προκειμένου να ταξινομηθεί η εικόνα.

2.4.1 Αρχιτεκτονική των Συνελικτικών Νευρωνικών Δικτύων

Τα κλασικά ΝΔ, όπως προαναφέρθηκε, δέχονται μία είσοδο, στην οποία εφαρμόζουν μη γραμμικούς μετασχηματισμούς, περνώντας τη μέσα από μία σειρά κρυφών επιπέδων. Κάθε επίπεδο αποτελείται από ένα σύνολο νευρώνων και είναι πλήρως συνδεδεμένο με όλους τους νευρώνες του προηγούμενου επιπέδου. Τέλος, υπάρχει το τελευταίο πλήρως συνδεδεμένο επίπεδο, το οποίο είναι το επίπεδο εξόδου και δίνει τις προβλέψεις του ΝΔ.

Σχετικά με τη δομή των ΣΝΔ, τα επίπεδα τους οργανώνονται με έναν διαφορετικό τρόπο. Καθώς η είσοδός τους είναι εικόνα, ακολουθούν μία αρχιτεκτονική κατά την οποία τα επίπεδά τους είναι δομημένα σε τρεις διαστάσεις, το μήκος, το ύψος και το βάθος, το οποίο εδώ αναφέρεται στα κανάλια της εικόνας. Επιπλέον, σε αντίθεση με τα κλασικά ΝΔ, οι νευρώνες του κάθε επιπέδου ενός ΣΝΔ δε συνδέονται με όλους τους νευρώνες του προηγούμενου επιπέδου, αλλά μόνο με ένα μέρος αυτού. Τέλος, η έξοδος ενός ΣΝΔ είναι ένα μοναδικό

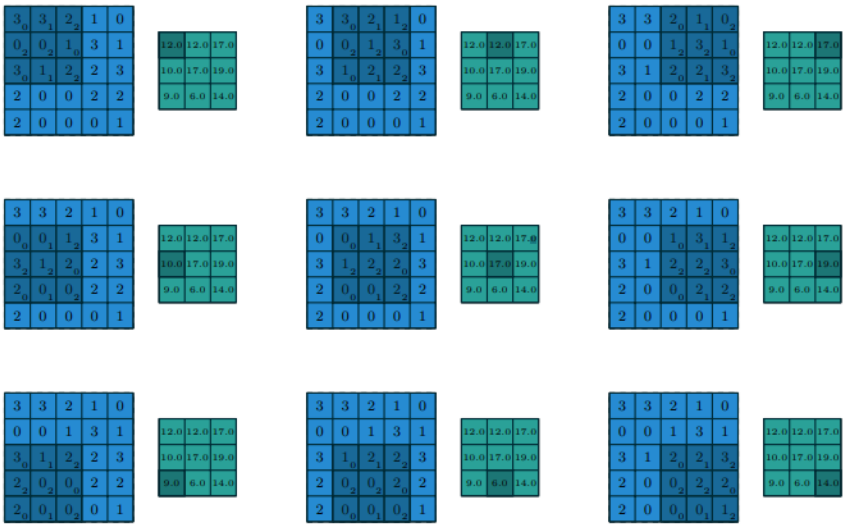
διάνυσμα πιθανοτήτων το οποίο οργανώνεται με βάση το βάθος (κανάλια) της εικόνας.

Τα επίπεδα ενός ΣΝΔ μπορούν να χωριστούν σε δύο κατηγορίες. Μία κατηγορία περιλαμβάνει τα επίπεδα εκείνα που είναι υπεύθυνα για την εξαγωγή χαρακτηριστικών από τις εικόνες (convolutional layers) και τον υπο-δειγματισμό (subsampling) της εικόνας. Η δεύτερη κατηγορία περιλαμβάνει τα επίπεδα που πραγματοποιούν εντέλει την ταξινόμηση της εικόνας σε κλάσεις (classification layers).

2.4.1.1 Συνελικτικά Επίπεδα – Πράξη Συνέλιξης

Το όνομα των ΣΝΔ προκύπτει από την πράξη της συνέλιξης (convolution), της οποίας ο κύριος στόχος όταν χρησιμοποιείται σε ένα ΣΝΔ είναι να εξάγει χαρακτηριστικά (features) από τις εικόνες που δίνονται ως είσοδος, με τη χρήση συνελικτικών φίλτρων. Ένα επίπεδο ΝΔ στο οποίο χρησιμοποιούνται συνελικτικά φίλτρα ονομάζεται συνελικτικό επίπεδο.

Όπως προαναφέρθηκε, κάθε εικόνα είναι ένας πίνακας που περιέχει φωτεινότητες. Στα πλαίσια των ΣΝΔ, η πράξη της συνέλιξης πραγματοποιείται στις εικόνες εισόδου με τη χρήση ενός φίλτρου (convolution filter/kernel), προκειμένου να δημιουργήσει έναν χάρτη χαρακτηριστικών (feature map). Το συνελικτικό φίλτρο είναι και αυτό ένας πίνακας, ο οποίος περιέχει βάρη. Ολισθαίνει σε ολόκληρη την εικόνα και κάθε βάρος του πολλαπλασιάζεται με την αντίστοιχη φωτεινότητα της εικόνας. Το άθροισμα αυτών των (στοιχείο επί στοιχείο) πολλαπλασιασμών αποθηκεύεται σε έναν καινούριο πίνακα, ο οποίος αποτελεί τον χάρτη χαρακτηριστικών. Το μέγεθος ενός χάρτη χαρακτηριστικών προκύπτει από τρεις παραμέτρους που καθορίζονται κατά τη δημιουργία ενός συνελικτικού επιπέδου, το βάθος (depth), το βήμα (stride) και το γέμισμα (padding).



Εικόνα 2-7: Η πράξη της συνέλιξης σε 2 διαστάσεις [Πηγή: [20]]

Depth. Το βάθος ενός feature map είναι ο αριθμός των φίλτρων που χρησιμοποιούνται κατά την πράξη της συνέλιξης και ο καθορισμός του αποτελεί μία από τις σημαντικότερες αποφάσεις που πρέπει να πάρει κανείς, κατά τον σχεδιασμό του μοντέλου. Δεν υπάρχει κάποιος συνιστώμενος αριθμός φίλτρων, καθώς ο ιδανικός αριθμός προκύπτει με πειραματισμό. Ωστόσο, σε γενικές γραμμές, όταν χρησιμοποιούνται πολλά φίλτρα σε χαμηλής ανάλυσης εικόνες, υπάρχει κίνδυνος πολλά χαρακτηριστικά που θα προκύψουν στο χάρτη να είναι διπλά. Αντιθέτως, σε υψηλής ανάλυσης εικόνες, όπου υπάρχουν πιο σύνθετα μοτίβα που πρέπει να ανακαλυφθούν, ένας μικρός αριθμός φίλτρων μπορεί να οδηγήσει σε απώλεια σημαντικών χαρακτηριστικών, με συνέπεια την κακή απόδοση του δικτύου κατά την εκπαίδευση. Κάθε φίλτρο είναι υπεύθυνο για την εξαγωγή κάποιου χαρακτηριστικού (ή συνδυασμό χαρακτηριστικών), με το πιο απλό να είναι αυτό που ανιχνεύει γραμμές στην εικόνα.

Stride. Το βήμα κατά την πράξη της συνέλιξης περιγράφει τον αριθμό των pixels κατά τον οποίο ολισθαίνει το φίλτρο πάνω στην εικόνα κάθε φορά [20] . Όσο μικρότερο το βήμα, τόσο λιγότερη πληροφορία χάνεται. Ωστόσο, δεν υπάρχει κάποια προτεινόμενη τιμή και προσαρμόζεται ανάλογα με το είδος και το μέγεθος των εικόνων εισόδου.

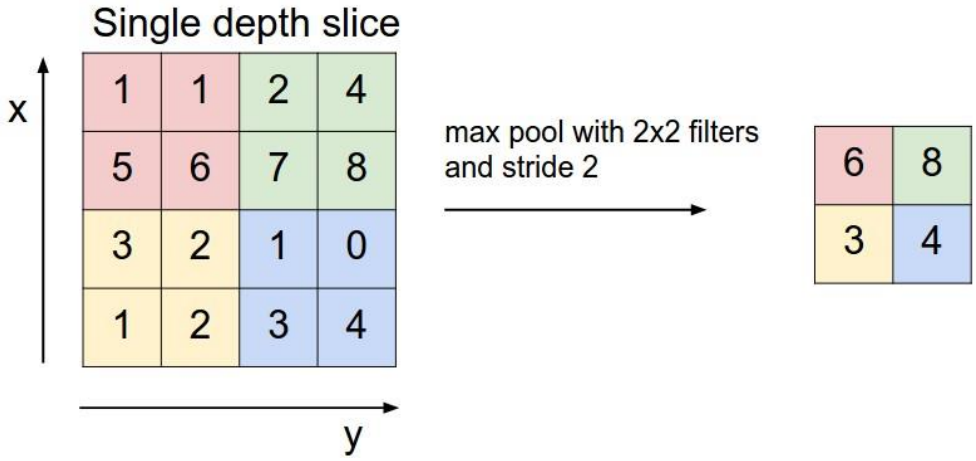
Padding. Το padding είναι ένα πλήθος μηδενικών, που μπορεί να προστεθεί στα περιθώρια της εικόνας [20]. Χρησιμοποιείται σε περιπτώσεις που απαιτείται η έξοδος ενός συνελικτικού επιπέδου να διατηρεί τις διαστάσεις της, με στόχο τη διατήρηση περισσότερης πληροφορίας.

2.4.1.2 Επίπεδα Pooling

Ένα από τα κυριότερα ζητήματα στις σύνθετες αρχιτεκτονικές Βαθιών Νευρωνικών Δικτύων, που αποτελούνται από πολύ μεγάλο αριθμό παραμέτρων, είναι η μείωση των διαστάσεών τους. Τα επίπεδα Pooling, ή αλλιώς επίπεδα υπο – δειγματισμού, είναι υπεύθυνα για τη μείωση των διαστάσεων του κάθε χάρτη χαρακτηριστικών και άρα των παραμέτρων ενός ΣΝΔ. Συνήθως τα επίπεδα Pooling αποτελούν το πρώτο επίπεδο ενός ΣΝΔ, ή ακολουθούν ένα, ή περισσότερα, συνελικτικά επίπεδα.

Υπάρχουν διαφορετικοί τύποι επιπέδων Pooling, ανάλογα με τους υπολογισμούς που πραγματοποιεί το καθένα. Τα πιο ευρέως χρησιμοποιούμενα είναι τα επίπεδα Max και Average Pooling. Στην περίπτωση των επιπέδων Max Pooling ορίζεται μία γειτονιά από pixels στον χάρτη χαρακτηριστικών (π.χ. 2 x 2) και επιλέγεται το μεγαλύτερο από τα στοιχεία που περιέχονται εκεί. Αντίστοιχα, τα επίπεδα Average Pooling υπολογίζουν τον μέσο όρο των στοιχείων της ορισμένης γειτονιάς. Και στις δύο περιπτώσεις το αποτέλεσμα είναι η μείωση των διαστάσεων του χάρτη χαρακτηριστικών.

Με τη χρήση επιπέδων Pooling ενισχύεται η απόδοση του μοντέλου, καθώς η συμπεριφορά του παραμένει αμετάβλητη απέναντι σε μικρές πιθανές ανωμαλίες στα δεδομένα εισόδου και έτσι δεν υπερπροσαρμόζεται σε αυτά. Επιπλέον, είναι ιδιαίτερα σημαντικό το γεγονός της μείωσης των παραμέτρων του ΣΝΔ, με αποτέλεσμα τα δεδομένα να είναι ευκολότερα διαχειρίσιμα.



Εικόνα 2-8: Υπο - δειγματισμός με Max Pooling και stride = 2 [Πηγή: Stanford]

2.4.1.3 Επίπεδα Dropout

Στη Μηχανική Μάθηση η ομαλοποίηση (regularization) είναι μία τεχνική που χρησιμοποιείται έτσι ώστε το μοντέλο που χρησιμοποιείται κάθε φορά να μην υπερπροσαρμόζεται στο σύνολο δεδομένων που του παρουσιάζεται ως είσοδος, δηλαδή να μην εξάγει από τα δεδομένα μόνο σύνολα αλληλεξαρτώμενων χαρακτηριστικών. Η ομαλοποίηση πραγματοποιείται με την επιβολή κάποιας αμοιβής στη συνάρτηση κόστους.

Τα επίπεδα Dropout είναι μία τεχνική ομαλοποίησης που βοηθά στην καλύτερη γενίκευση του μοντέλου. Κατά τη διαδικασία της μάθησης, επιλέγεται ένα τυχαίο σύνολο από νευρώνες οι οποίοι 'αποκόπτονται' από το δίκτυο, δηλαδή μένουν ανενεργοί. Ως συνέπεια αυτού, προσωρινά δε συνεισφέρουν στην ενεργοποίηση άλλων νευρώνων κατά το forward pass και η ανανέωση των βαρών δεν εφαρμόζεται στους νευρώνες κατά το backward pass.

Ακόμη και στην περίπτωση που ένα ποσοστό νευρώνων είναι ανενεργοί, το ΝΔ οφείλει να εξάγει όσο το δυνατόν πιο σωστά συμπεράσματα για κάθε δείγμα. Έτσι, με τη χρήση επιπέδων Dropout το μοντέλο οδηγείται στο να μην προσαρμόζεται υπερβολικά στα δεδομένα.

Τα επίπεδα Dropout χρησιμοποιούνται μόνο κατά τη φάση της εκπαίδευσης. Συνήθως ακολουθούν τα πλήρως συνδεδεμένα επίπεδα του ΝΔ, ωστόσο πολλές φορές χρησιμοποιούνται και μετά από επίπεδα Pooling.

2.4.1.4 Πλήρως Συνδεδεμένα Επίπεδα

Όπως και στα κλασικά ΝΔ, τα πλήρως συνδεδεμένα επίπεδα (Fully Connected Layers) διαθέτουν συνδέσεις με όλες τις ενεργοποιήσεις του προηγούμενου επιπέδου. Η θέση ενός πλήρως συνδεδεμένου επιπέδου είναι στο τέλος ενός ΣΝΔ και είναι υπεύθυνο για την ταξινόμηση των δειγμάτων σε κλάσεις.

Ένα πλήρως συνδεδεμένο επίπεδο δέχεται μία είσοδο από το προηγούμενο επίπεδο (Convolutional, Pooling κτλ.) και τη μετατρέπει σε ένα διάνυσμα Ν διαστάσεων, όπου Ν είναι ο αριθμός των κλάσεων που ορίζονται στο πρόβλημα. Κάθε αριθμός του διανύσματος αυτού είναι μία πιθανότητα που δηλώνει το πόσο πιθανό είναι το κάθε δείγμα να ανήκει στην κλάση που ορίζεται από τη θέση της πιθανότητας αυτής στο διάνυσμα.

2.4.2 Προσαρμογή στα Δεδομένα

Ένα από τα πιο απαιτητικά προβλήματα που προκύπτουν κατά την εκπαίδευση ενός μοντέλου Μηχανικής/Βαθιάς Μάθησης είναι η υπερπροσαρμογή του μοντέλου στα δεδομένα (ovefitting). Το overfitting είναι ένας από τους πιο συχνούς παράγοντες που οδηγούν ένα μοντέλο σε φτωχά αποτελέσματα και επομένως είναι ένα ζήτημα που ο προγραμματιστής οφείλει να διευθετήσει.

Η έννοια overfitting χρησιμοποιείται για να περιγράψει την αδυναμία ενός μοντέλου να γενικεύεται σε νέα δεδομένα. Η γενίκευση αναφέρεται στην ικανότητα του μοντέλου να εφαρμόζει τη γνώση που απέκτησε κατά την εκπαίδευσή του σε δεδομένα που του ήταν άγνωστα κατά τη φάση της εκπαίδευσης, δηλαδή στην ουσία η γενίκευση αναφέρεται στον αντικειμενικό στόχο ενός μοντέλου Μηχανικής/Βαθιάς Μάθησης.

Συγκεκριμένα, στα πλαίσια της ταξινόμησης εικόνων, το overfitting προκύπτει σε περιπτώσεις όπου το μοντέλο μαθαίνει τόσο καλά ακόμη και τις πιο μικρές λεπτομέρειες/χαρακτηριστικά και το θόρυβο στις εικόνες εκπαίδευσης, σε βαθμό που επηρεάζει αρνητικά την απόδοση του μοντέλου σε νέες εικόνες. Αυτό σημαίνει ότι ο θόρυβος, ή οι τυχαίες διακυμάνσεις των δεδομένων εκπαίδευσης προωθούνται και μαθαίνονται εντέλει σαν χαρακτηριστικά των εικόνων, με αποτέλεσμα το μοντέλο να εξάγει λάθος, ή ασήμαντα συμπεράσματα. Το πρόβλημα προκύπτει όταν το μοντέλο καλείται να κάνει μία πρόβλεψη για μία άγνωστη εικόνα, στην οποία δεν υπάρχει ο θόρυβος, ή τα ασήμαντα χαρακτηριστικά που έμαθε να αναγνωρίζει κατά την εκπαίδευση.

Το overfitting είναι πιθανότερο να προκύψει σε μοντέλα στα οποία πραγματοποιούνται μη γραμμικοί μετασχηματισμοί των δεδομένων, λόγω της ευελιξίας που παρουσιάζουν κατά τη διαδικασία της εκπαίδευσης. Προκύπτει λοιπόν, ότι το overfitting είναι ένα πολύ συχνό πρόβλημα και για τα ΣΝΔ και για τον λόγο αυτό χρησιμοποιούνται τύποι επιπέδων των οποίων στόχος είναι η ελαχιστοποίηση του overfitting.

Ένας από τους τρόπους αντιμετώπισης του overfitting στα ΣΝΔ είναι η προσθήκη των επιπέδων Dropout [21] που προαναφέρθηκαν. Τα επίπεδα Dropout εφαρμόζονται μόνο κατά τη φάση της εκπαίδευσης και απενεργοποιούν κάποιους νευρώνες με μία πιθανότητα p. Συνεπώς, κατά την εκπαίδευση το μοντέλο είναι λιγότερο ευαίσθητο στην αλλαγή των βαρών και μειώνεται η πιθανότητα για overfitting. Κατά τη φάση του validation και του test οι νευρώνες αυτοί παραμένουν ενεργοί. Έτσι, κατά κάποιον τρόπο επιτυγχάνεται μία ισορροπία και η σύγκριση των αποτελεσμάτων εκπαίδευσης με αυτά του validation και του test οδηγούν σε ασφαλέστερα συμπεράσματα για τη συνολική απόδοση του μοντέλου.

Το αντίθετο ακριβώς πρόβλημα που μπορεί να προκύψει κατά την εκπαίδευση ονομάζεται υποπροσαρμογή στα δεδομένα (underfitting). Η ύπαρξη underfitting σημαίνει πως το μοντέλο αφενός δεν κατάφερε να μάθει ουσιώδη χαρακτηριστικά από τις εικόνες κατά τη φάση της εκπαίδευσης, και αφετέρου, δεν είναι σε θέση να γενικευτεί επαρκώς σε νέα δεδομένα. Ένα μοντέλο με underfitting δεν αποτελεί λύση σε ένα πρόβλημα Βαθιάς/Μηχανικής Μάθησης, ωστόσο θεωρείται μικρότερο πρόβλημα από το overfitting, καθώς είναι ευκολότερο να εντοπιστεί, λόγω των πολύ φτωχών αποτελεσμάτων στην εκπαίδευση. Το underfitting συνήθως αντιμετωπίζεται με τη χρήση μεγαλύτερων μοντέλων και περισσότερων δεδομένων, προκειμένου να παρουσιαστούν στο μοντέλο περισσότερα δείγματα και από αυτά να εξάγει περισσότερα χαρακτηριστικά [22].

Στην ιδανική περίπτωση, στόχος είναι η επιλογή ενός μοντέλου που προσαρμόζεται επαρκώς σε νέα δεδομένα, πράγμα που είναι αρκετά δύσκολο να γίνει στην πράξη. Συνήθως στην αντιμετώπιση του overfitting και underfitting βοηθά η σχεδίαση γραφικών παραστάσεων που δείχνουν την απόδοση του μοντέλου τόσο κατά τη φάση του training, όσο και του validation στην πάροδο του χρόνου.

Ιδανικά, όσο περνά ο χρόνος και το μοντέλο μαθαίνει, η συνάρτηση κόστους του training set μειώνεται παράλληλα με τη συνάρτηση κόστους του validation set. Αν το training συνεχιστεί για πολλές εποχές, το πιο πιθανό είναι ότι το κόστος του training θα συνεχίσει να μειώνεται. Όταν παράλληλα παρατηρηθεί μείωση, ή στασιμότητα, του κόστους validation είναι η στιγμή που το μοντέλο ξεκινά να μαθαίνει ασήμαντα χαρακτηριστικά. Έτσι μπορούμε να εξάγουμε το συμπέρασμα πως η εκπαίδευση πρέπει να σταματήσει, προκειμένου το μοντέλο μας να μην υπερπροσαρμοστεί στα δεδομένα.

3 Μεθοδολογία

Στο Κεφάλαιο αυτό περιγράφεται λεπτομερώς ο σχεδιασμός που ακολουθήθηκε για την υλοποίηση των πειραμάτων στα πλαίσια της διπλωματικής αυτής εργασίας.

Γίνεται αρχικά αναφορά στα διαθέσιμα εργαλεία και βιβλιοθήκες που χρησιμοποιούνται σε προβλήματα Μηχανικής Μάθησης και τεκμηριώνονται οι λόγοι οι οποίοι οδήγησαν στην επιλογή κάποιων από αυτών.

Στη συνέχεια, περιγράφονται η μορφή, οι ιδιαιτερότητες και οι προκλήσεις του συνόλου δεδομένων "Quick, Draw!" , προκειμένου να δημιουργηθεί μία βάση για την καλύτερη κατανόηση των παραμέτρων που επιλέχθηκαν για τα πειράματα.

Αναλύονται στη συνέχεια οι παράμετροι που έπρεπε να αποφασιστούν για την πραγματοποίηση των πειραμάτων.

3.1 Βιβλιοθήκες και Εργαλεία

Για την διεκπεραίωση των πειραμάτων χρησιμοποιήθηκαν διάφορα εργαλεία ειδικά σχεδιασμένα για την υλοποίηση, χρήση και επέκταση αρχιτεκτονικών Deep Learning. Παρακάτω αναφέρονται μερικές πληροφορίες τόσο για τα εργαλεία που χρησιμοποιήθηκαν (TensorFlow, Keras), όσο και για ένα ακόμη σύγχρονο αντίστοιχο εργαλείο που χρησιμοποιείται ευρέως (PyTorch) και εξηγούνται τα κριτήρια με βάση τα οποία έγινε η επιλογή για τα πειράματα. Υπάρχουν αρκετά ακόμη διαθέσιμα εργαλεία που δεν αναφέρονται στα πλαίσια της εργασίας αυτής.

3.1.1 Python

Η επιλογή της γλώσσας προγραμματισμού για τη συγκεκριμένη εργασία δεν ήταν δύσκολη, καθώς τα τελευταία χρόνια η γλώσσα που χρησιμοποιείται για την υλοποίηση εφαρμογών που βασίζονται σε Νευρωνικά Δίκτυα είναι η Python. Είναι μία γλώσσα που λόγω του απλού συντακτικού της είναι εύκολα αντιληπτή, αλλά ο βασικός λόγος που επιλέγεται σε εφαρμογές Νευρωνικών Δικτύων είναι το γεγονός ότι υπάρχουν διαθέσιμα πάρα πολλά packages σε Python που είναι σχεδιασμένα για αυτόν τον σκοπό. Για τα πειράματα της εργασίας αυτής χρησιμοποιήθηκε η Python 3.

3.1.2 Scikit – learn

Το Scikit – learn (ή Sklearn) είναι μία βιβλιοθήκη για την Python που παρέχει ένα σύνολο από αλγορίθμους classification, regression, clustering (support vector machines, random forests, KMeans, DBSCAN κτλ.), αλλά και πολλές συναρτήσεις που απλοποιούν τις διαδικασίες διαχωρισμού των δεδομένων σε training, test και validation sets, το «ανακάτεμα» (shuffling) των δεδομένων για τη δημιουργία πιο ισορροπημένων datasets, και πολλές ακόμη συναρτήσεις που είναι χρήσιμες κατά την υλοποίηση εφαρμογών Machine Learning.

3.1.3 **Numpy**

Το Numpy είναι μία ακόμη βιβλιοθήκη της Python που παρέχει υποστήριξη για πράξεις μεταξύ πολυδιάστατων δομών (arrays, matrices) και συνοδεύεται από μία πολύ μεγάλη συλλογή υλοποιήσεων μαθηματικών συναρτήσεων. Το γεγονός αυτό, έχει καταστήσει το numpy ένα απαραίτητο εργαλείο για τον υπολογισμό των πράξεων που πραγματοποιούνται κατά την εκπαίδευση ενός ΝΔ και συνεπώς οι υλοποιήσεις πολλών τύπων ΝΔ στα Deep Learning Frameworks έχουν ως προκαθορισμένο τύπο εισαγωγής των δεδομένων στο δίκτυο πίνακες τύπου Numpy.

3.1.4 **TensorFlow**

Η TensorFlow είναι μια open source βιβλιοθήκη για μαθηματικούς υπολογισμούς και μεγάλης κλίμακας προβλήματα Μηχανικής Μάθησης, που σχεδιάστηκε από την ομάδα Google Brain. Διαθέτει ενσωματωμένους πολλούς αλγορίθμους που σχετίζονται με την υλοποίηση Νευρωνικών Δικτύων και των μαθηματικών πράξεων που πραγματοποιούνται για να προκύψει η έξοδός τους. Ένα πλεονέκτημα της TensorFlow είναι το γεγονός ότι χρησιμοποιεί την Python για να παρέχει ένα βολικό API στους χρήστες, ενώ στη βάση της είναι γραμμένη σε C++, με αποτέλεσμα να έχει πολύ ισχυρή απόδοση.

Η TensorFlow [23] είναι σχεδιασμένη έτσι ώστε να παρέχει στους προγραμματιστές γράφους ροής δεδομένων (dataflow graphs), δηλαδή δομές οι οποίες δείχνουν πως τα δεδομένα ρέουν μέσα από μία σειρά κόμβων. Κάθε κόμβος που υπάρχει στον γράφο αναπαριστά κάποιον μαθηματικό υπολογισμό και κάθε σύνδεση μεταξύ των κόμβων είναι ένας πίνακας δεδομένων πολλών διαστάσεων, που ονομάζεται tensor.

Το χαρακτηριστικό της TensorFlow είναι το ότι είναι σχεδιασμένη ως μία υψηλού επιπέδου αφαίρεση, που στοχεύει στο να απαλλάξει τον προγραμματιστή από το να αναλώνεται σε λεπτομέρειες της υλοποίησης των αλγορίθμων, και να επικεντρώνεται στην καθαρή λογική της εκάστοτε εφαρμογής. Η TensorFlow αναλαμβάνει όλες οι λεπτομέρειες που αφορούν τις υλοποιήσεις και φροντίζει να πραγματοποιεί αποτελεσματικά όλους τους υπολογισμούς για να υποστηρίξει τον σχεδιασμό του προγραμματιστή.

3.1.5 **Keras**

Το Keras ([24], [25]) είναι ένα open source API για εφαρμογές Νευρωνικών Δικτύων, που τρέχει πάνω από άλλες βιβλιοθήκες, όπως η TensorFlow. Σχεδιάστηκε από έναν μηχανικό της Google, τον Fransois Chollet, με τρόπο τέτοιο ώστε να είναι φιλικό, εύκολο και γρήγορο ως προς τη χρήση του. Δεν πραγματοποιεί χαμηλού επιπέδου υπολογισμούς, αλλά χρησιμοποιεί μια άλλη βιβλιοθήκη για τον σκοπό αυτό, που ονομάζεται Backend (TensorFlow, Theano, CNTK).

Το Keras είναι δηλαδή στην ουσία είναι ένα υψηλού επιπέδου wrapper για ένα χαμηλότερου επιπέδου API, που διαχειρίζεται τον τρόπο με τον οποίο ορίζονται τα μοντέλα Deep Learning και μπορεί να τα εκπαιδεύει και να τα κάνει compile χρησιμοποιώντας συναρτήσεις κόστους και βελτιστοποίησης. Η διαχείριση

χαμηλότερου επιπέδου υπολογισμών, όπως π.χ. οι πράξεις μεταξύ tensors, η δημιουργία dataflow graphs, ή η δημιουργία μεταβλητών, είναι ενέργειες που διαχειρίζεται η Backend βιβλιοθήκη.

Η διαδικασία του training για ένα μοντέλο που έχει δημιουργηθεί με το Keras μπορεί να γίνει είτε σε μία, είτε παράλληλα σε πολλές GPU, καθώς υποστηρίζεται η παράλληλη επεξεργασία μεγάλου όγκου δεδομένων, με αποτέλεσμα την επιτάχυνση κάθε διαδικασίας που πραγματοποιείται κατά την εκπαίδευση του μοντέλου.

Λόγω της μινιμαλιστικής δομής του το Keras αποτελεί ιδανικό εργαλείο για την κατασκευή μοντέλων Νευρωνικών Δικτύων, ειδικότερα σε περιπτώσεις όπου υπάρχουν ήδη υλοποιημένοι οι τύποι των επιπέδων (π.χ. Convolutional Layer), οι διαδικασίες του forward και του backward pass και οι συναρτήσεις κόστους που χρησιμοποιούνται σε αυτό. Ωστόσο, σε περιπτώσεις όπου απαιτείται η ειδική και εξατομικευμένη παραμετροποίηση επιπέδων, ή διαδικασιών εκπαίδευσης, το Keras δεν προτείνεται ως κατάλληλη επιλογή framework, λόγω της υψηλής αφαίρεσης των διαδικασιών του.

Η κύρια δομή του Keras είναι η δομή Model, που ορίζει τον πλήρη γράφο του μοντέλου. Δημιουργώντας αρχικά ένα στιγμιότυπο του Sequential Model, η διαδικασία του να ορίσει κανείς ένα δικό του μοντέλο περιορίζεται στο να τοποθετήσει όλα τα επίπεδα που θα συμπεριλάβει στο μοντέλο του σειριακά σε αυτό. Δημιουργείται έτσι μία ακολουθία επιπέδων, από την οποία ορίζεται η ροή της πληροφορίας και συνεπώς η τελική έξοδος του μοντέλου.

Εκτός από τους κλασικούς τύπους Νευρωνικών Δικτύων, το Keras υποστηρίζει επιπλέον τη δημιουργία CNNs και RNNs και συνδυασμό των δύο αυτών τύπων, καθώς επίσης και πολλούς τύπους επιπέδων που χρησιμοποιούνται σε αυτά, όπως π.χ. τα επίπεδα Dropout, MaxPooling, Batch Normalization.

3.1.6 **PyTorch**

Το PyTorch [26] είναι ένα ακόμη framework για εφαρμογές Νευρωνικών Δικτύων, το οποίο είναι βασισμένο στο αντίστοιχο framework Torch της Lua και σχεδιάστηκε και χρησιμοποιήθηκε από την ερευνητική ομάδα του Facebook. Το χαρακτηριστικό του PyTorch είναι η ιδιαίτερα ισχυρή υποστήριξη επιτάχυνσής των υπολογισμών μεταξύ tensors σε GPU και η δυναμική δημιουργία γράφων ροής δεδομένων. Ως αποτέλεσμα του δεύτερου, ο προγραμματιστής έχει τον έλεγχο όλων των μεταβλητών προς εκπαίδευση και ανά πάσα στιγμή μπορεί να τις ελέγξει.

3.1.7 **Επιλογή Framework**

Για την υλοποίηση των πειραμάτων στα πλαίσια της εργασίας αυτής επιλέχθηκε η χρήση του Keras σε συνδυασμό με την TensorFlow. Οι βασικότεροι λόγοι για τους οποίους επιλέχθηκαν είναι αρχικά το γεγονός ότι όλοι οι τύποι επιπέδων που χρησιμοποιούνται από το μοντέλο που δημιουργήθηκε υπάρχουν ήδη ορισμένοι στο Keras και επομένως δε χρειάστηκε ο ορισμός κάποιας πιο εξειδικευμένης δομής. Παρομοίως, για το παρόν πρόβλημα αρκούν οι ορισμένες

συναρτήσεις κόστους και βελτιστοποίησης, όπως επίσης και η υποστήριξη της όλης διαδικασίας του forward και του backward pass.

Επιπλέον, δεδομένου ότι το σύνολο δεδομένων που χρησιμοποιείται στην εργασία αποτελείται από μικρές ασπρόμαυρες εικόνες χαμηλής ανάλυσης (28 x 28), αρκεί η ορισμένη αλληλεπίδραση και επιτάχυνση υπολογισμών σε GPU που παρέχεται από το Keras και δεν είναι αναγκαία η περαιτέρω επιτάχυνση που παρέχει το PyTorch.

Συμπερασματικά, στα πλαίσια της εργασίας αυτής επιλέχθηκε ένα πολύ ισχυρό και φιλικό ως προς τη χρήση framework για εφαρμογές Deep Learning, το Keras, που τρέχει πάνω στην πιο χαμηλού επιπέδου Backend βιβλιοθήκη TensorFlow. Λόγω της φύσης του προβλήματος δεν υπήρχε ανάγκη για την υλοποίηση ακόμη πιο χαμηλού επιπέδου operations (συνεπώς ούτε και η ανάγκη συνεχούς παρακολούθησης των γράφων δεδομένων) και επομένως η δημιουργία του μοντέλου ήταν μία εύκολη και ευχάριστη διαδικασία που προέκυψε από την πλήρη εκμετάλλευση των operations που παρέχονται από το Keras.

3.2 Quick, Draw!

Όπως αναφέρθηκε και στην εισαγωγή, στα πλαίσια της εργασίας χρησιμοποιείται το σύνολο δεδομένων “Quick Draw!”. Στην υποενότητα αυτή αναφέρονται μερικές λεπτομέρειες παραπάνω, προκειμένου να γίνουν αργότερα αντιληπτά τα αποτελέσματα, τα οποία σε οποιοδήποτε πρόβλημα είναι άρρηκτα συνδεδεμένα με τον τύπο και το μέγεθος του συνόλου δεδομένων που χρησιμοποιείται κάθε φορά.

Οι εικόνες του “Quick,Draw!” είναι ασπρόμαυρες, διαθέτουν δηλαδή ένα κανάλι, πράγμα που καθιστά τους υπολογισμούς που απαιτούνται κατά την εκπαίδευση πιο γρήγορους από ότι σε RGB εικόνες. Από τις διάφορες μορφές που υπάρχει διαθέσιμο το σύνολο δεδομένων, για το βασικό πείραμα της διπλωματικής αυτής εργασίας χρησιμοποιήθηκε η μορφή στην οποία η κάθε κλάση δίνεται ως ένα αρχείο numpy, που περιέχει πάνω από 10 χιλιάδες ασπρόμαυρα σκίτσα διαστάσεων 28 x 28 για την κάθε κλάση.

Ωστόσο, το πείραμα που συμμετείχε στον αντίστοιχο διαγωνισμό του Kaggle, χρησιμοποιήθηκε μία δεύτερη μορφή των δεδομένων, όπου τα σκίτσα δίνονται ως δομές json, με το κάθε σκίτσο να είναι μία ακολουθία από «μολυβιές». Κάθε «μολυβιά» περιέχει τα σημεία στα οποία ζωγράφισε ο χρήστης συνεχόμενα, χωρίς να σηκώσει το «μολύβι». Η διαφορά είναι ότι για τον δεύτερο τύπο δεδομένων, είναι απαραίτητο κατά τη φόρτωσή τους τα σκίτσα να ζωγραφίζονται στην ουσία από την αρχή, ακολουθώντας το σύνολο των «μολυβιών» (strokes) στο χρόνο, πράγμα που όπως γίνεται αντιληπτό μεγαλώνει την πολυπλοκότητα των υπολογισμών και συνεπώς είναι δυσκολότερο να χρησιμοποιηθεί μεγαλύτερο πλήθος δειγμάτων. Επιπλέον, στον δεύτερο τύπο δεδομένων τα σκίτσα έχουν διαστάσεις 256 x 256.

Τα δεδομένα του Quick, Draw! έχουν υποστεί προεπεξεργασία από τη συλλογή τους ως τη διάθεσή τους στο κοινό, προκειμένου να είναι πιο εύκολη η χρήση και το κατέβασμα του συνόλου δεδομένων. Τα σκίτσα παρέχονται χωρισμένα σε κλάσεις, σε αρχεία .npy, ή .ndjson. Στα πλαίσια της απλοποίησής τους, τα .ndjson δεδομένα έχουν τοποθετηθεί σε μία περιοχή διαστάσεων 256 x 256 και έχουν στοιχιστεί στην πάνω αριστερή γωνία της, ενώ τα .npy είναι στοιχισμένα στο κέντρο μίας 28 x 28 περιοχής. Και στις δύο μορφές έχει γίνει scaling ώστε η μέγιστη τιμή των pixels να είναι η 255. Τέλος, τα σκίτσα έχουν απλοποιηθεί με τη χρήση του αλγορίθμου Ramer – Douglas – Peucker, ο οποίος όταν υπάρχει μία καμπύλη που αποτελείται από τμήματα γραμμών, προσπαθεί να βρει μία παρόμοια καμπύλη με λιγότερα σημεία, ώστε να προκύψει μία πιο εξομαλυμένη μορφή της αρχικής.

Η επιλογή των numpy αρχείων του βασικού πειράματος έγινε με σκοπό ξεκινώντας με έναν μικρό αριθμό δειγμάτων από κάθε κλάση, να είναι δυνατή η επέκταση του αριθμού των δειγμάτων, ώστε το πείραμα να μεγαλώσει σε κλίμακα. Αυτό είναι εύκολο να γίνει με τη χρήση των .npy αρχείων, καθώς ενώ η πολυπλοκότητα αυξάνεται λόγω του μεγαλύτερου όγκου δεδομένων, οι δυαδικοί πίνακες που παρέχονται στα αρχεία (υπάρχει – δεν υπάρχει ακμή) είναι σχετικά εύκολο και γρήγορο να επεξεργαστούν, σε σύγκριση με RGB εικόνες μεγαλύτερων διαστάσεων.

Το σύνολο δεδομένων περιέχει 345 κλάσεις, ανάμεσα στις οποίες περιλαμβάνονται πολλές κλάσεις τετράποδων ζώων (σκύλος, κατσίκα, γάιδαρος, ζέβρα κ.ά.), πτηνών, εργαλείων (τσεκούρι, πένσα, κ.ά.), απλών σχημάτων (τετράγωνο, κύκλος, πινακίδα stop, smiley face), είδη ρουχισμού (μπουφάν, t – shirt κ.ά.), διάφορα είδη αθλητικού εξοπλισμού (μπάλες, μπαστούνι για διαφορετικά αθλήματα κ.ά.), έντομα (μέλισσα, κουνούπι κ.ά.), φρούτα και λαχανικά (μήλο, πορτοκάλι, καρότο, μπρόκολο κ.ά.), αλλά και πιο αφηρημένες έννοιες, όπως π.χ. η μετανάστευση ζώων. Οι παραπάνω κλάσεις αναφέρονται ενδεικτικά, προκειμένου να γίνει αντιληπτό ότι στη μορφή μιας ζωγραφιάς, πολλά σκίτσα από κλάσεις που παρουσιάζουν μεταξύ τους ομοιότητες είναι πολύ πιθανό να μπερδεύουν τον classifier, ενώ αντίστοιχα κάποιες κλάσεις που δεν παρουσιάζουν ομοιότητα με καμία άλλη, όπως π.χ. η Μόνα Λίζα, δίνουν επιπλέον confidence στον classifier. Για παράδειγμα, όπως φαίνεται και από την εικόνα 4.1 δεν είναι ιδιαίτερα εύκολος ο διαχωρισμός δύο σκίτσων που ανήκουν στις κλάσεις smiley face και pig, πιθανώς επειδή τα σκίτσα είναι ημιτελή. Επιπλέον στην εικόνα 4.1 υπάρχει ένα σκίτσο της κλάσης pig που απεικονίζει ολόκληρο το ζώο, ενώ τα υπόλοιπα μόνο το κεφάλι. Αυτή είναι μία συνθήκη στην οποία πρέπει να προσαρμοστεί ο classifier, καθώς η αντίληψη του κάθε ατόμου όταν πρόκειται να ζωγραφίσει κάποιο σκίτσο είναι διαφορετική για το κάθε αντικείμενο. Χαρακτηριστική περίπτωση αποτελούν επίσης σκίτσα που απεικονίζουν το ίδιο αντικείμενο, είναι όμως ζωγραφισμένα από αντίθετες πλευρές.

Στα πειράματα, τόσο σε αυτά που χρησιμοποιήθηκαν δεδομένα μόνο από 10 κλάσεις, όσο και σε αυτά που χρησιμοποιήθηκαν δεδομένα και από τις 345 κλάσεις, δόθηκε ως είσοδος στο μοντέλο ο ίδιος αριθμός σκίτσων για όλες τις κλάσεις. Αυτό είναι πολύ σημαντικό να γίνεται πριν το στάδιο της εκπαίδευσης, καθώς έτσι

παρέχεται ένα ισορροπημένο σύνολο δεδομένων στον classifier και έτσι παρουσιάζει «αμερόληπτη» συμπεριφορά τόσο κατά την εκπαίδευση, όσο και κατά τις προβλέψεις του στα άγνωστα δεδομένα.

Να σημειωθεί, τέλος, ότι το σύνολο δεδομένων Quick, Draw! έχει χρησιμοποιηθεί και σε εφαρμογές που δε σχετίζονται με τη Μηχανική Μάθηση, αλλά αφορούν περισσότερο την ανάλυση των δεδομένων και την εξαγωγή συμπερασμάτων από αυτά. Μία από τις πιο ενδιαφέρουσες είναι η "How do you draw a circle?", όπου αναλύονται 100000 σκίτσα με σκοπό την εξαγωγή συμπερασμάτων σχετικά με το αν οι άνθρωποι ζωγραφίζουν τα κυκλικά σχήματα αριστερόστροφα ή δεξιόστροφα, ανάλογα με την κουλτούρα της χώρας τους [27] . Επίσης, ενδιαφέρον παρουσιάζει η εφαρμογή "Exploring and Visualizing an Open Global Dataset" [28] η οποία μέσα από εντυπωσιακά visualizations των σκίτσων προσπαθεί να εξάγει συμπεράσματα σχετικά με τις ομοιότητες ή διαφορές των διαφόρων αντικειμένων – κλάσεων στα μάτια ανθρώπων από διαφορετικές χώρες. Τέλος, υπάρχουν διαθέσιμες διάφορες εφαρμογές Μηχανικής Μάθησης μικρής κλίμακας, με στόχο την ταξινόμηση των σκίτσων με διαφορετικές τεχνικές ([29], [30], [31]).

3.3 Παραμετροποίηση προβλήματος

Στην ενότητα αυτή αναλύονται οι παράμετροι που έπρεπε να ληφθούν υπόψη κατά τη σχεδίαση της αρχιτεκτονικής που επιλέχθηκε για την ταξινόμηση του συνόλου δεδομένων "Quick, Draw!". Οι παράμετροι αυτές επιλέχθηκαν σε βάθος χρόνου, μετά από μία σειρά πειραμάτων.

Μοντέλο

Ίσως η δυσκολότερη παράμετρος που πρέπει να αποφασιστεί για την υλοποίηση ενός Νευρωνικού Δικτύου, είναι η αρχιτεκτονική του μοντέλου, δηλαδή ο αριθμός και ο τύπος των επιπέδων που θα χρησιμοποιηθούν και ο τρόπος με τον οποίο αυτά συνδέονται μεταξύ τους.

Σχετικά με το πλήθος και τον τύπο των επιπέδων που χρησιμοποιούνται σε μία αρχιτεκτονική ενός Νευρωνικού Δικτύου, δεν υπάρχουν συγκεκριμένοι κανόνες, αλλά κατά βάση μία κατάλληλη αρχιτεκτονική μπορεί να προκύψει λαμβάνοντας υπόψη το πλήθος και τον τύπο των δεδομένων και παρακολουθώντας την απόδοση του μοντέλου στο πέρασμα του χρόνου. Πολλές φορές η παρατήρηση overfitting ή underfitting αποτελεί μία ένδειξη για την αφαίρεση ή την προσθήκη επιπλέον επιπέδων ή/και τη χρήση περισσότερων δειγμάτων στο μοντέλο.

Για την ταξινόμηση των εικόνων του συνόλου δεδομένων "Quick, Draw!" αποφασίστηκε να υλοποιηθεί ένα Συνελικτικό Νευρωνικό Δίκτυο, λόγω της καλής απόδοσης των ΣΝΔ σε προβλήματα όπου τα δεδομένα είναι εικόνες [10]. Η αρχιτεκτονική που επιλέχθηκε μοιάζει κατά βάση με αυτή του LeNet και η επιλογή αυτή έγινε λόγω της ομοιότητας του συνόλου δεδομένων "Quick, Draw!" με το MNIST που χρησιμοποιήθηκε για τη δημοσίευση του LeNet. Συγκεκριμένα, για την υλοποίηση των πειραμάτων δημιουργήθηκαν δύο όμοιες αρχιτεκτονικές, οι οποίες αποτελούνται από δύο μέρη, τα επίπεδα που αφορούν την εξαγωγή

χαρακτηριστικών από τις εικόνες και τα επίπεδα που αφορούν την ταξινόμησή τους.

Εξαγωγή Χαρακτηριστικών. Για την εξαγωγή χαρακτηριστικών από τις εικόνες στο τελικό πείραμα των 345 κλάσεων, χρησιμοποιήθηκαν 4 Συνελικτικά Επίπεδα με το τρίτο από αυτά να ακολουθείται από ένα επίπεδο MaxPooling και το τέταρτο από ένα επίπεδο MaxPooling και ένα επίπεδο Dropout. Στο πείραμα των 10 κλάσεων, χρησιμοποιήθηκαν 5 Συνελικτικά επίπεδα, με το τρίτο και το πέμπτο από αυτά να ακολουθούνται από επίπεδα MaxPooling και Dropout.

Ταξινόμηση Εικόνων. Η ταξινόμηση των εικόνων και στις δύο αρχιτεκτονικές έγινε με τη χρήση ενός επιπέδου Flatten, δύο πλήρως συνδεδεμένων επιπέδων και ενός επιπέδου Dropout να παρεμβάλλεται μεταξύ των δύο τελευταίων.

Περισσότερες λεπτομέρειες για την είσοδο και την έξοδο των επιπέδων που χρησιμοποιήθηκαν αναφέρονται στην επόμενη ενότητα.

Συνάρτηση Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης είναι ιδιαίτερα σημαντικές υπερπαράμετροι για τη διαδικασία της μάθησης, καθώς ο στόχος τους είναι ο μετασχηματισμός της εισόδου ενός επιπέδου σε μία έξοδο, τέτοια ώστε το επόμενο επίπεδο να μπορεί να τη δεχτεί ως είσοδο. Όπως αναφέρθηκε και στο κεφάλαιο 2, υπάρχουν διάφορα είδη συναρτήσεων ενεργοποίησης, με το κάθε ένα από αυτά να ενδείκνυται για κάποιον συγκεκριμένο τύπο προβλήματος.

Στις αρχιτεκτονικές που υλοποιήθηκαν στα πλαίσια της εργασίας αυτής, χρησιμοποιείται η μη – γραμμική συνάρτηση ενεργοποίησης ReLU σε όλα τα επίπεδα, πλην του τελευταίου πλήρως συνδεδεμένου επιπέδου. Το πλεονέκτημά της βρίσκεται στο γεγονός πως όταν η είσοδος της είναι αρνητική, μετατρέπεται σε 0 και έτσι ο αντίστοιχος νευρώνας δεν ενεργοποιείται. Συνεπώς, οι υπολογισμοί γίνονται γρηγορότερα και πιο αποτελεσματικά.

Για το τελικό πλήρως συνδεδεμένο επίπεδο εξόδου, που πραγματοποιεί την ταξινόμηση των δειγμάτων, χρησιμοποιείται η συνάρτηση ενεργοποίησης softmax, καθώς ενδείκνυται για προβλήματα multi – class classification. Η έξοδος της softmax είναι η κατανομή που δείχνει την πιθανότητα για κάθε δείγμα να ανήκει σε μία συγκεκριμένη κλάση.

Batch size

Στη Μηχανική Μάθηση χρησιμοποιείται ο όρος batch size για να περιγράψει τον συνολικό αριθμό των δειγμάτων εκπαίδευσης που «περνάει» από το δίκτυο κάθε φορά (backpropagation). Για παράδειγμα, αν η τιμή του batch size είναι 100, αυτό σημαίνει ότι ο αλγόριθμος μάθησης θα χρησιμοποιήσει τα πρώτα 100 από τα δείγματα εκπαίδευσης για να εκπαιδεύσει το μοντέλο. Θα συνεχίσει με τα επόμενα 100, κ.ο.κ. Στην περίπτωση που περισσέψουν δείγματα λιγότερα από τον ορισμένο αριθμό του batch size, θα χρησιμοποιηθούν μόνο αυτά για το τελικό βήμα εκπαίδευσης του δικτύου.

Οι τιμές του συνήθως κυμαίνονται από 1 ως 256, ωστόσο συχνά χρησιμοποιούνται και μεγαλύτερες τιμές, ανάλογα με τον όγκο των δεδομένων και

τη μνήμη του μηχανήματος στο οποίο εκτελείται η εκπαίδευση (πρέπει τα batches να χωράνε στη μνήμη κάθε φορά). Σε γενικές γραμμές, προτείνεται ο αριθμός του batch size να είναι μικρότερος από τον συνολικό αριθμό των δειγμάτων εκπαίδευσης, καθώς αυτό επιταχύνει την εκπαίδευση. Στην περίπτωση που το batch size είναι μικρότερο από τον συνολικό αριθμό δειγμάτων εκπαίδευσης, το μειονέκτημα είναι ότι όσο μικρότερη είναι η τιμή του batch size, αυξάνεται η πιθανότητα ο υπολογισμός των gradients να είναι ανακριβής, λόγω της έντονης διακύμανσης. Επομένως, καλό είναι να χρησιμοποιείται η μεγαλύτερη δυνατή τιμή batch size για την οποία τα δεδομένα χωράνε στη μνήμη.

Στα πλαίσια της εργασίας, οι πειραματισμοί ξεκίνησαν με το batch size να είναι αρχικά 32, ωστόσο καθώς μεγάλωνε η κλίμακα του προβλήματος, η τιμή ανέβαινε για να επιτευχθεί γρηγορότερη εκπαίδευση. Η τελική τιμή που χρησιμοποιήθηκε ήταν η 1024 για τα πειράματα στα οποία συμμετείχαν όλες οι κλάσεις (για ένα σύνολο δειγμάτων εκπαίδευσης της τάξης του 1.5 εκατομμυρίου), αλλά και η τιμή 512 έδωσε σχεδόν το ίδιο αποτέλεσμα. Για τα πειράματα των 10 κλάσεων, επιλέχθηκε η τιμή 512.

Εποχές Εκπαίδευσης

Μία εποχή εκπαίδευσης (training epoch) είναι στην ουσία ένα forward και ένα backward pass για όλο το σύνολο δειγμάτων εκπαίδευσης και σημαίνει πως κάθε ένα από τα δείγματα εκπαίδευσης είχε την ευκαιρία να ανανεώσει τις εσωτερικές παραμέτρους του μοντέλου. Μία εποχή αποτελείται από βήματα, με κάθε βήμα να χρησιμοποιεί τον αριθμό των δειγμάτων που ορίζεται από το batch size, έως ότου το backpropagation πραγματοποιηθεί για όλα τα δείγματα εκπαίδευσης. Συνεπώς, ο συνολικός αριθμός εποχών σε ένα πρόβλημα Μηχανικής Μάθησης περιγράφει το πόσες φορές το μοντέλο θα επεξεργαστεί το σύνολο των training samples.

Δεν υπάρχει κάποιος κανόνας που να ορίζει το ποιος είναι ο σωστός αριθμός εποχών για ένα πρόβλημα Μηχανικής Μάθησης, καθώς οι εποχές είναι μία υπερπαράμετρος (hyperparameter) του μοντέλου που εξαρτάται από ένα σύνολο παραγόντων, όπως π.χ. ο αριθμός των κρυφών επιπέδων, το σύνολο των δειγμάτων εκπαίδευσης, κ.ά. Όταν ο αριθμός των εποχών είναι πολύ μικρός, είναι πολύ πιθανό το μοντέλο να μην προλάβει να εξάγει πολλά χαρακτηριστικά από τα δεδομένα, ενώ η αντίθετη περίπτωση συνήθως οδηγεί στην εξαγωγή ασήμαντων για το πρόβλημα χαρακτηριστικών. Σε γενικές γραμμές προτείνεται να ορίζεται αρχικά ένας μεγάλος αριθμός εποχών εκπαίδευσης, και να περιορίζεται μετά από παρακολούθηση του πειράματος στον αριθμό των εποχών για τις οποίες το μοντέλο δεν παρουσίαζε overfitting. Για τη διευκόλυνση αυτής της διαδικασίας, το Keras παρέχει ένα callback που ονομάζεται EarlyStopping και στο οποίο ο προγραμματιστής έχει την ευκαιρία να ορίσει τον μέγιστο επιτρεπτό αριθμό εποχών μη βελτίωσης της ακρίβειας ή του κόστους. Σε περίπτωση που ο αριθμός αυτός ξεπεραστεί, η εκπαίδευση σταματά.

Για τα πειράματα της εργασίας σύμφωνα με τους παραπάνω κανόνες, το μοντέλο εκπαιδεύτηκε για 23 εποχές (10 κλάσεις) και για 30 εποχές (345 κλάσεις). Και στις δύο περιπτώσεις χρησιμοποιήθηκε το callback EarlyStopping με patience = 5.

Train – Test Split

Στα προβλήματα Μηχανικής Μάθησης, μία απόφαση που είναι καθοριστική για τα αποτελέσματα του μοντέλου αφορά τον αριθμό των δειγμάτων που χρησιμοποιούνται τόσο για την εκπαίδευση, όσο και την αξιολόγηση του μοντέλου (training, validation και test samples).

Τα training samples είναι αυτά από τα οποία το μοντέλο μαθαίνει χαρακτηριστικά, είναι επομένως ιδιαιτέρως σημαντικό τα δεδομένα αυτά να είναι αρκετά, ώστε να υπάρχει ποικιλία χαρακτηριστικών και να αποφεύγεται το overfitting. Τα validation samples από την άλλη, είναι τα δείγματα τα οποία χρησιμοποιούνται προκειμένου να εφαρμοστεί η γνώση η οποία αποκτήθηκε από τα training samples στο τέλος κάθε εποχής, με στόχο μία αμερόληπτη αξιολόγηση του μοντέλου. Τέλος, τα test samples, είναι κάποια δείγματα δεδομένων τα οποία ήταν άγνωστα στο μοντέλο κατά την εκπαίδευση και χρησιμοποιούνται προκειμένου να αξιολογηθεί το κατά πόσο το μοντέλο μπορεί να εφαρμόσει τη γνώση που απέκτησε και να γενικεύεται σε δεδομένα όμοια με αυτά που χρησιμοποιήθηκαν στο πρόβλημα. Τα test samples χρησιμοποιούνται μόνο μία φορά, αφού το μοντέλο έχει εκπαιδευτεί πλήρως, ενώ τα training και validation samples χρησιμοποιούνται σε κάθε εποχή εκπαίδευσης. Ένα λάθος που γίνεται συχνά, είναι το γεγονός ότι τα validation samples χρησιμοποιούνται και ως test samples. Αυτό που δεν είναι καλή τακτική, καθώς δεν είναι δυνατό να αξιολογηθεί το μοντέλο αντικειμενικά σε δεδομένα που έχει ήδη συναντήσει.

Το scikit-learn παρέχει μία συνάρτηση που ονομάζεται train_test_split για τον διαχωρισμό των δεδομένων στα προαναφερόμενα σύνολα. Στη συνάρτηση αυτή ορίζεται το ποσοστό των δεδομένων που θέλει ο προγραμματιστής να κρατήσει για testing με βάση το συνολικό πλήθος των διαθέσιμων δεδομένων. Κάτι ιδιαίτερα σημαντικό που παρέχει η συνάρτηση, είναι το γεγονός ότι διαχωρίζει τα σύνολα με τέτοιο τρόπο, ώστε να είναι ισορροπημένα, δηλαδή π.χ. να χρησιμοποιείται περίπου ο ίδιος αριθμός samples από κάθε κλάση, τόσο για το training όσο και για το test. Η τεχνική που χρησιμοποιήθηκε σε αυτή τη διπλωματική εργασία προκειμένου η τελική αξιολόγηση του μοντέλου να είναι αντικειμενική, συμπεριλαμβάνει:

1) τον διαχωρισμό των δεδομένων σε train και test sets με τη χρήση της συνάρτησης του scikit-learn κατά την αρχή του ορισμού του προβλήματος

2) την διατήρηση ενός ποσοστού από τα training samples με σκοπό να χρησιμοποιηθούν μόνο ως validation samples. Αυτό ορίστηκε ως παράμετρος (validation_split) στη συνάρτηση fit του Keras, η οποία αναλαμβάνει το backpropagation

Με τον τρόπο αυτό εξασφαλίστηκε πως κατά την αξιολόγηση του μοντέλου θα χρησιμοποιηθούν μόνο δεδομένα άγνωστα στο μοντέλο. Σε γενικές γραμμές, ισχύει ο κανόνας ότι όταν ο αριθμός των training και test/validation samples είναι κοντά, το μοντέλο δεν προλαβαίνει να μάθει σημαντικά χαρακτηριστικά από τα δεδομένα κατά την εκπαίδευση, με αποτέλεσμα να μην αποδίδει επαρκώς ούτε στα training, αλλά ούτε και στα test και validation samples. Αυτό οδηγεί σε underfitting (εκτός ίσως από περιπτώσεις όπου ο συνολικός όγκος των δεδομένων

είναι τεράστιος) και χρειάζεται αναπροσαρμογή. Στην αντίθετη περίπτωση, όταν δηλαδή χρησιμοποιείται πολύ μεγάλο πλήθος training samples σε σχέση με τα validation/test samples, το μοντέλο οδηγείται σε overfitting. Στη διπλωματική αυτή εργασία έγινε κατά βάση ένα train_test_split 80%-20%, και επιπλέον 10% και 20% validation_split επί των training samples, για κάθε πείραμα αντίστοιχα. Δοκιμάστηκε επιπλέον ο διαχωρισμός σε train 70% και test 30%, που οδήγησε όμως σε φτωχότερα αποτελέσματα και συνεπώς απορρίφθηκε.

Συνάρτηση Κόστους

Η επιλογή του cost function κατά βάση εξαρτάται από τον εκάστοτε τύπο του προβλήματος. Το Keras παρέχει ένα πλήθος διαφορετικών cost functions, η κάθε μία από τις οποίες εφαρμόζεται καλύτερα σε κάποιον συγκεκριμένο τύπο προβλήματος (mean_squared_error, categorical_crossentropy, binary_crossentropy, cosine_proximity κ.ά.). Η συνάρτηση κόστους που οδηγεί σε καλύτερα αποτελέσματα είναι και η κατάλληλη επιλογή για το κάθε πρόβλημα.

Στη διπλωματική αυτή εργασία, το ζητούμενο είναι η ταξινόμηση του κάθε σκίτσου σε μία μοναδική κλάση (από τις συνολικά 345 στη δεύτερη σειρά πειραμάτων και από τις συνολικά 10 στην πρώτη σειρά πειραμάτων). Αυτό το πρόβλημα ονομάζεται multi-class classification και ο τύπος cost function που προτείνεται είναι η συνάρτηση κόστους categorical cross entropy που αναλύθηκε στο Κεφάλαιο 2.

Αλγόριθμος Βελτιστοποίησης

Όμοια με την επιλογή συνάρτησης κόστους γίνεται και η επιλογή της συνάρτησης βελτιστοποίησης (optimizer). Στο Keras υπάρχουν διαθέσιμοι όλοι οι optimizers που αναλύθηκαν στο κεφάλαιο 2 (SGD, Adam, Adagrad, Adadelta, RMSprop κ.ά.)

Για τη διπλωματική αυτή εργασία χρησιμοποιήθηκε ο optimizer Adam που προσαρμόζει τον ρυθμό μάθησης σε κάθε παράμετρο. Σε γενικές γραμμές, ο Adam είναι ένας optimizer που προτιμάται σε διάφορα είδη προβλημάτων Μηχανικής Μάθησης, καθώς ελαχιστοποιεί γρηγορότερα το cost function, γι' αυτό και τα τελευταία χρόνια χρησιμοποιείται κατά κόρον. Επιπλέον, δοκιμάστηκε και ο optimizer SGD, αλλά έδωσε φτωχότερα αποτελέσματα και συνεπώς απορρίφθηκε.

Μετρικές

Υπάρχουν διάφοροι τύποι μετρικών που χρησιμοποιούνται για την αξιολόγηση των αποτελεσμάτων ενός προβλήματος ταξινόμησης. Οι κύριοι από αυτούς (που χρησιμοποιούνται και στη διπλωματική εργασία) είναι η ακρίβεια (training, validation, test accuracy), το loss (training, validation, test loss) και το classification report που περιλαμβάνει τα precision, recall και fı score.

Η ακρίβεια δείχνει τον αριθμό των samples που ταξινομήθηκαν σωστά, ως ένα ποσοστό, με βάση όλες τις προβλέψεις που έγιναν για κάθε batch δειγμάτων. Όμοια, το training loss είναι ένα metric που χρησιμοποιείται για την αξιολόγηση της απόδοσης του μοντέλου, σχετικά με την πιθανότητα το κάθε δείγμα να ανήκει σε μία κλάση. Το loss μπορεί να ερμηνευθεί ως ένα μέτρο που ακολουθεί το πόσο confident είναι ένα μοντέλο ταξινόμησης στις προβλέψεις του.

Στη διπλωματική αυτή εργασία, επειδή το πρόβλημα είναι ένα multi class classification 10 και 345 κλάσεων, χρησιμοποιούνται ως μέτρα τόσο η ακρίβεια της πρώτης πρόβλεψης (δηλαδή της κλάσης στην οποία ο classifier έδωσε μεγαλύτερη πιθανότητα να ανήκει το κάθε δείγμα), αλλά και η ακρίβεια στις τρεις πρώτες προβλέψεις (δηλαδή οι τρεις κλάσεις στις οποίες ο classifier έδωσε μεγαλύτερη πιθανότητα να ανήκει το κάθε δείγμα). Στο υπόλοιπο της εργασίας αυτής, τα μέτρα αυτά αναφέρονται ως top − 1 και top − 3 scores, σύμφωνα με τα ονόματα των αντίστοιχων metrics που παρέχονται στο Keras, αντίστοιχα.

Το classification report είναι ένα function της βιβλιοθήκης scikit − learn, που επιστρέφει έναν πίνακα με τις τιμές precision, recall και f1 score για κάθε μία από τις κλάσεις οι οποίες συμμετέχουν στο πρόβλημα της ταξινόμησης. Συνοπτικά:

1) Precision = True Positive / (True Positive + False Positive)
2) Recall = True Positive / (True Positive + False Negative)
3) F score = 2 * (Precision * Recall) / (Precision + Recall)

Όπου:

- True Positive είναι το σύνολο των δειγμάτων που ανήκουν σε μία κλάση X και προβλέφθηκε ότι ανήκουν στην κλάση X
- False Positive είναι το σύνολο των δειγμάτων που ενώ ανήκουν σε μία κλάση Y, προβλέφθηκε ότι ανήκουν σε μία κλάση X
- False Negative είναι το σύνολο των δειγμάτων που ενώ ανήκουν σε μία κλάση X, προβλέφθηκε ότι ανήκουν σε μία κλάση Y.

Συνεπώς, το precision είναι ένα μέτρο που δείχνει το ποσοστό όλων των δειγμάτων που ταξινομήθηκαν σωστά σε κάθε κλάση. Το recall δείχνει το ποσοστό των δειγμάτων που ταξινομήθηκαν σωστά σε μία κλάση, σε σχέση με το σύνολο όλων των δειγμάτων που ανήκαν στην κλάση αυτή στην πραγματικότητα. Τέλος, το F1 score δείχνει έναν σταθμισμένο μέσο όρο των precision και recall.

4 Πειράματα

Πραγματοποιήθηκαν δύο σειρές πειραμάτων διαφορετικής κλίμακας. Η πρώτη σειρά πειραμάτων χρησιμοποιεί σκίτσα από μόνο 10 κλάσεις, ενώ η δεύτερη χρησιμοποιεί σκίτσα από όλες τις διαθέσιμες κλάσεις (345) του "Quick, Draw!". Στην ενότητα αυτή, αναφέρονται λεπτομέρειες σχετικά με τις επιλεγμένες αρχιτεκτονικές και την παραμετροποίησή τους και για τις δύο σειρές πειραμάτων.

Ξεκινώντας τα πειράματα, για αρχή κρίθηκε σκόπιμο να χρησιμοποιηθούν μόνο οι 10 από τις 345 συνολικά κλάσεις του "Quick, Draw!". Ο σκοπός αυτής της σειράς πειραμάτων ήταν η σύγκριση με ένα σημείο αναφοράς, όπως είναι το ποσοστό ακρίβειας ~99% στο MNIST, καθώς και αυτό το σύνολο δεδομένων χρησιμοποιεί παρόμοιες εικόνες από 10 κλάσεις (ψηφία 0-9). Στη συνέχεια έγιναν πειράματα χρησιμοποιώντας σκίτσα και από τις 345 κλάσεις. Στην υποενότητα αυτή δίνονται τα αποτελέσματα και για τις δύο σειρές πειραμάτων.

4.1 Δεδομένα

Ο τύπος δεδομένων που χρησιμοποιήθηκε για τα πειράματα ήταν ασπρόμαυρες εικόνες μεγέθους 28 x 28, με κάθε κλάση να είναι ένα αρχείο numpy που περιέχει όλα τα σκίτσα που ανήκουν στην κλάση αυτή. Στο πείραμα των 10 κλάσεων, χρησιμοποιήθηκαν 10000 σκίτσα από κάθε μία από τις 10 επιλεγμένες κλάσεις του "Quick, Draw!". Οι 10 κλάσεις που επιλέχθηκαν φαίνονται στον παρακάτω πίνακα.

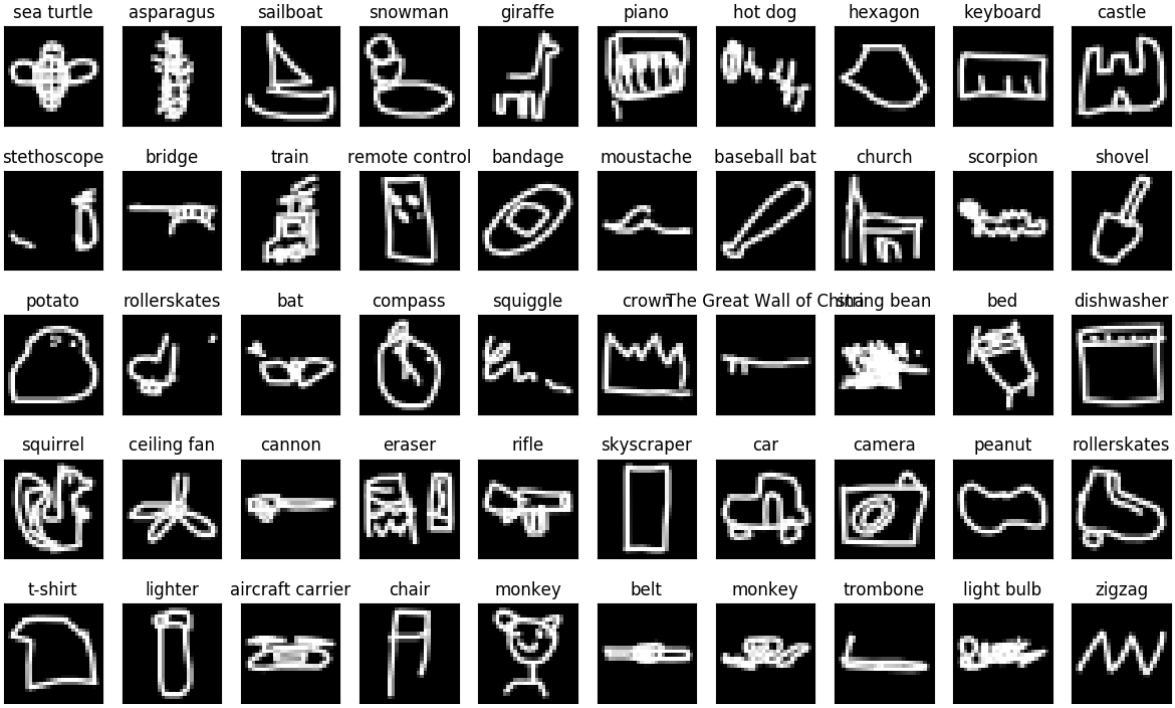
Πίνακας 4-1: Οι 10 κλάσεις σκίτσων για την πρώτη σειρά πειραμάτων

Α/Α	Κλάση
1	calculator
2	ice cream
3	jacket
4	pig
5	pineapple
6	postcard
7	rain
8	smiley face
9	The Eiffel Tower
10	violin

Συνολικά για το πείραμα των 10 κλάσεων χρησιμοποιήθηκαν 100000 σκίτσα. Από το σύνολο αυτό, το 80% των σκίτσων χρησιμοποιήθηκαν ως training samples, ενώ το 20% ως test samples. Τέλος, κατά την εφαρμογή του μοντέλου διατηρήθηκε ένα επιπλέον 20% επί του συνόλου των training samples για τη φάση του validation (64000 training, 16000 validation, 20000 test samples).

Στο τελικό πείραμα χρησιμοποιήθηκαν 6450 σκίτσα για κάθε μία από τις 345 κλάσεις του "Quick, Draw!" (συνολικά 2225250 σκίτσα). Έγινε επίσης train – test split 80% - 20% και επιπλέον validation_split 10% επί των training samples. (1602180 training, 178020 validation, 445050 test samples). Τα ονόματα των 345 κλάσεων είναι διαθέσιμα με αλφαβητική σειρά στο Παράρτημα Α - Dataset.

Εικόνα 4-1: Τυχαία training samples (10 κλάσεις)



Εικόνα 4-2: Τυχαία training samples (345 κλάσεις)

4.2 Εκπαίδευση

Σε αρχικό στάδιο, το μοντέλα εκπαιδεύτηκαν για 40 εποχές, πράγμα που οδήγησε σε overfitting. Έγινε έτσι αναγκαία η χρήση μερικών callbacks του Keras, προκειμένου να ελέγχεται το overfitting και η εκπαίδευση να σταματά όταν το μοντέλο σταματά να μαθαίνει χρήσιμα χαρακτηριστικά από τις εικόνες.

Στην τελική μορφή του πειράματος, έγινε εκπαίδευση για 23 εποχές για το μοντέλο ταξινόμησης των 10 κλάσεων, ενώ για 30 εποχές για το μοντέλο των 345 κλάσεων. Επιπλέον χρησιμοποιήθηκε το callback EarlyStopping του Keras, όπου ορίστηκαν ως μεγαλύτερος ανεκτός χρόνος μη βελτίωσης (patience) της ακρίβειας οι 5 εποχές.

Χρησιμοποιήθηκε επιπλέον το callback ReduceLROnPlateau, όπου ορίστηκε η μείωση του ρυθμού μάθησης στο μισό, στην περίπτωση όπου η ακρίβεια δεν είχε αυξηθεί για 3 συνεχόμενες εποχές εκπαίδευσης. Όπως αναφέρθηκε και στο Κεφάλαιο 2, συχνά η μείωση του ρυθμού μάθησης δίνει μία μικρή ώθηση στο μοντέλο σε περιπτώσεις που έχει σταματήσει να μαθαίνει τα επιθυμητά χαρακτηριστικά, με αποτέλεσμα συνήθως να αυξάνεται η ακρίβεια των προβλέψεων και να μειώνεται το κόστος.

Το batch size που χρησιμοποιήθηκε για το μοντέλο των 10 κλάσεων ήταν 512 δείγματα ανά πέρασμα, και 1024 για το μοντέλο των 345 κλάσεων .

4.3 Μοντέλα

Για τις αρχιτεκτονικές και των δύο μοντέλων χρησιμοποιήθηκε ένας συνδυασμός από Convolutional, MaxPooling και Dropout layers για την εξαγωγή των χαρακτηριστικών από τις εικόνες, σε συνδυασμό με τη συνάρτηση ενεργοποίησης ReLU. Για την ταξινόμηση των εικόνων χρησιμοποιήθηκαν από ένα επίπεδο Flatten, ένα πλήρως συνδεδεμένο ακολουθούμενο από Dropout και ένα πλήρως συνδεδεμένο επίπεδο εξόδου, στο οποίο εφαρμόστηκε η συνάρτηση ενεργοποίησης softmax, αντίστοιχα.

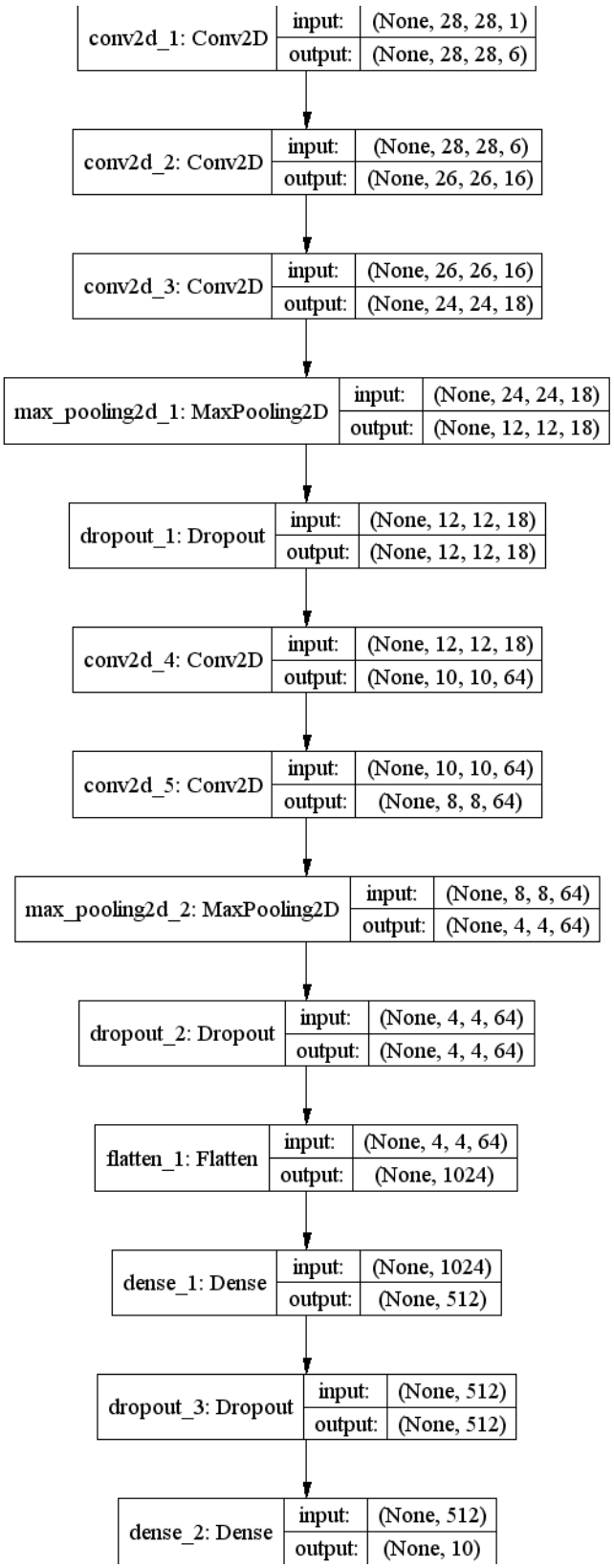
Πίνακας 4-2: Έξοδοι και παράμετροι του μοντέλου ταξινόμησης 10 κλάσεων

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 6)	60
conv2d_2 (Conv2D)	(None, 26, 26, 16)	880
conv2d_3 (Conv2D)	(None, 24, 24, 18)	2610
max_pooling2d_1	(None, 12, 12, 18)	0
dropout_1 (Dropout)	(None, 12, 12, 18)	0
conv2d_4 (Conv2D)	(None, 10, 10, 64)	10432
conv2d_5 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_2	(None, 4, 4, 64)	0
dropout_2 (Dropout)	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130
Total params: 580,840		
Trainable params: 580,840		
Non-trainable params: 0		

Ως παράδειγμα για τον υπολογισμό των παραμέτρων του κάθε επιπέδου αναλύεται το πρώτο Convolutional Layer του μοντέλου 10 κλάσεων. Δέχεται ως είσοδο μία εικόνα διαστάσεων (28, 28) με 1 κανάλι, ενώ το μέγεθος του φίλτρου που χρησιμοποιείται είναι (3,3). Ως έξοδο επιστρέφει 6 activation maps, διαστάσεων (28, 28). Επειδή είναι το πρώτο επίπεδο και ιδανικά δε θέλουμε από την αρχή να χάσουμε πληροφορία από τις εικόνες, το stride ορίζεται σε 1 και επιπλέον χρησιμοποιείται padding, με αποτέλεσμα να μην υπάρχει μείωση των διαστάσεων. Ο συνολικός αριθμός των παραμέτρων για το επίπεδο αυτό υπολογίζεται: (kernel_height * kernel_width +1)*number_of _filters. Ο αριθμός 1 αφορά το bias και για την περίπτωση του συγκεκριμένου layer έχουμε (3*3 + 1) * 6 = 60 παραμέτρους. Φυσικά, στην περίπτωση που η εικόνα ήταν RGB έπρεπε να λάβουμε υπόψη και τα τρία κανάλια της.

Στα επόμενα Convolutional Layers δεν εφαρμόζεται padding και αυτό σε συνδυασμό με τα επίπεδα MaxPooling, οδηγεί σταδιακά σε μείωση των διαστάσεων των activation maps που εξάγονται. Στην πράξη αυτό σημαίνει πως αρχικά το μοντέλο μαθαίνει πιο γενικά χαρακτηριστικά από τις εικόνες, όπως οι ακμές και οι καμπύλες, ενώ στη συνέχεια μαθαίνει να συνδυάζει πιο υψηλού επιπέδου χαρακτηριστικά, όπως για παράδειγμα ο συνδυασμός τριών ακμών, με αποτέλεσμα την αναγνώριση των σκίτσων.

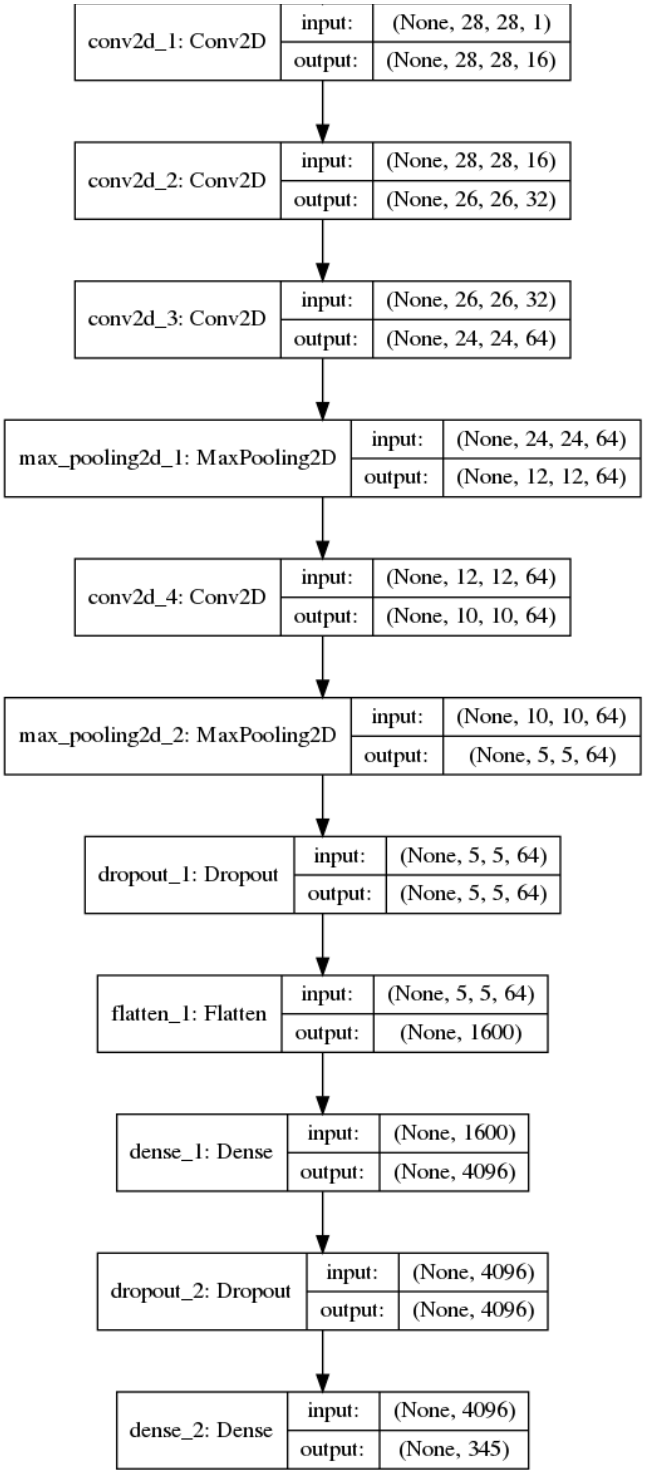
Για τα επίπεδα MaxPooling ορίστηκε η εξαγωγή των μεγαλύτερων τιμών σε μία γειτονιά από pixels διαστάσεων 2 * 2. Στα επίπεδα Dropout επιλέχθηκε να αγνοούνται το 25%, 25% και 50% των νευρώνων για το μοντέλο των 10 κλάσεων και το 25% και 50% για το μοντέλο των 345 κλάσεων, για κάθε επίπεδο Dropout αντίστοιχα.



Εικόνα 4-3: Η αρχιτεκτονική του μοντέλου ταξινόμησης (10 κλάσεις)

Πίνακας 4-3: Έξοδοι και παράμετροι του μοντέλου ταξινόμησης 345 κλάσεων

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 16)	160
conv2d_2 (Conv2D)	(None, 26, 26, 32)	4640
conv2d_3 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 12, 12, 64)	0
conv2d_4 (Conv2D)	(None, 10, 10, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_1 (Dense)	(None, 4096)	6557696
dropout_2 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 345)	1413465
Total params: 8,031,385		
Trainable params: 8,031,385		
Non-trainable params: 0		



Εικόνα 4-4: Η αρχιτεκτονική του μοντέλου ταξινόμησης (345 κλάσεις)

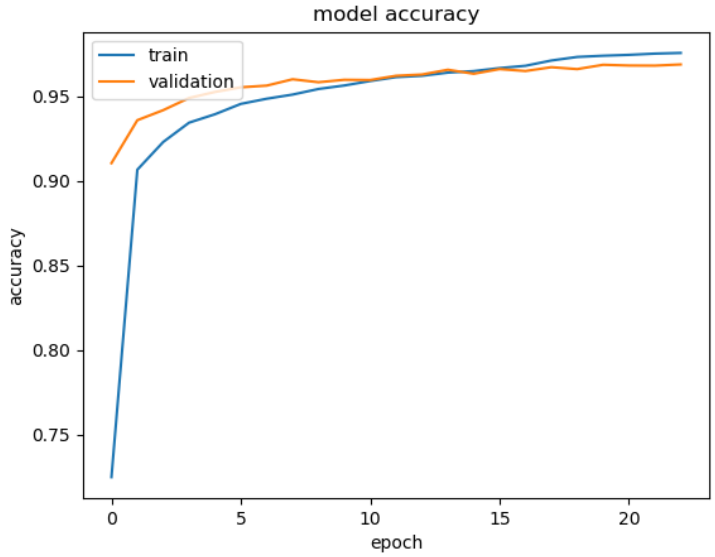
4.3.1.1 Συναρτήσεις κόστους και βελτιστοποίησης, μετρικές

Λόγω της φύσης του προβλήματος (multi – class classification) η συνάρτηση κόστους που επιλέχθηκε είναι η categorical cross entropy, σε συνδυασμό με τον optimizer Adam. Οι τιμές που μετρήθηκαν είναι η συνολική ακρίβεια της πρώτης πρόβλεψης, αλλά και η συνολική ακρίβεια των τριών πρώτων

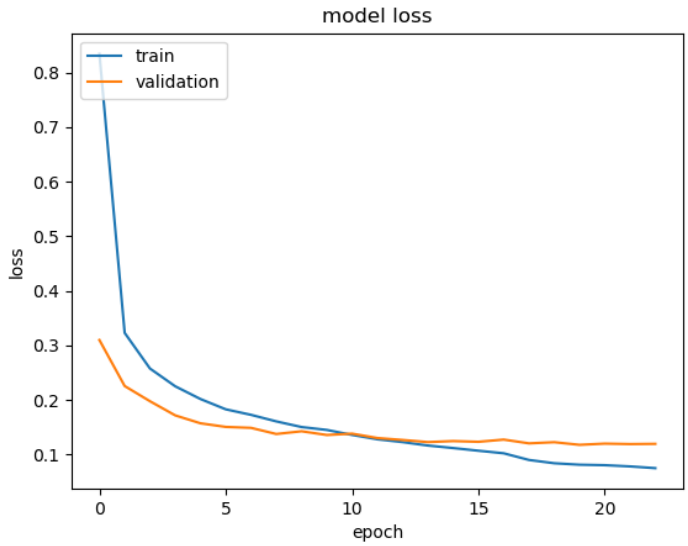
προβλέψεων. Τέλος, για κάθε κλάση μετρήθηκαν οι τιμές precision, recall και f1 score.

4.4 Αποτελέσματα

Στα διαγράμματα παρακάτω φαίνεται η πορεία μάθησης των δύο μοντέλων. Με μπλε χρώμα σημειώνονται οι καμπύλες των training accuracy και loss, ενώ με πορτοκαλί οι καμπύλες των validation accuracy και loss, στο πέρασμα των εποχών.



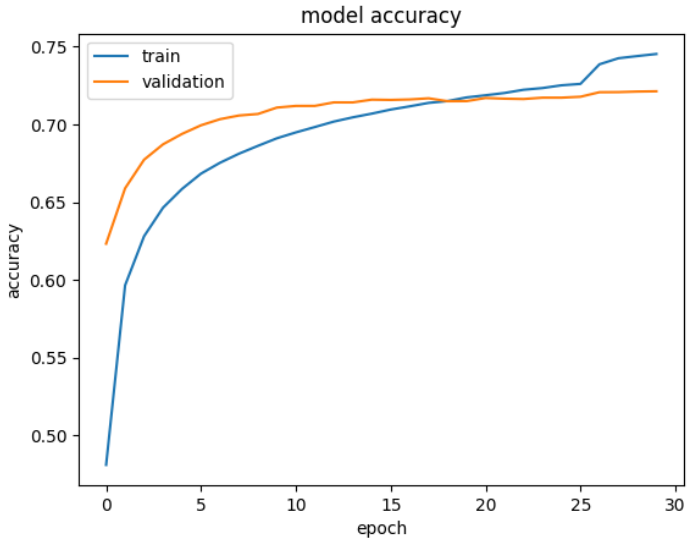
Εικόνα 4-5: Διάγραμμα training - validation accuracy του μοντέλου ταξινόμησης (10 κλάσεις)



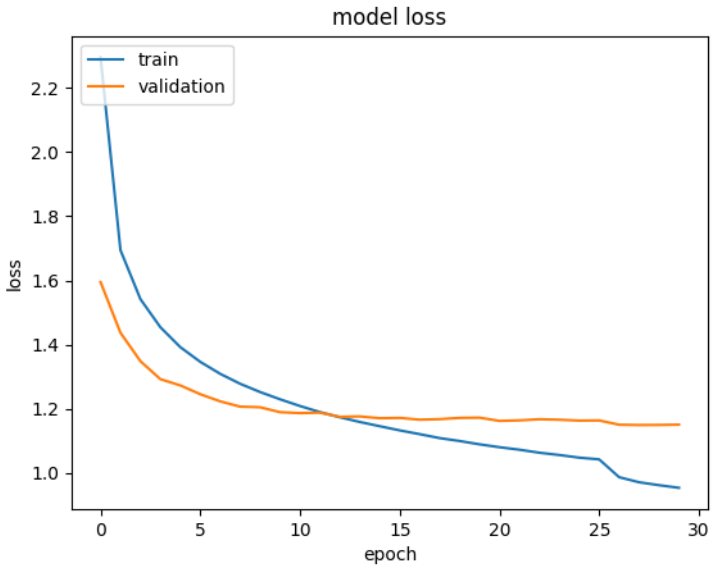
Εικόνα 4-6: Διάγραμμα training - validation loss του μοντέλου ταξινόμησης (10 κλάσεις)

Όπως γίνεται αντιληπτό από τα διαγράμματα, τα μοντέλα προσαρμόζεται καλά στα δεδομένα (δεν υπάρχει σημαντικό overfitting ή underfitting), πράγμα το οποίο αποτελεί έναν δείκτη για την καλή γενίκευσή τους σε άγνωστα δεδομένα. Αυτό φαίνεται και στους πίνακες 4-4 και 4-5, που περιέχουν τα αποτελέσματα από την εφαρμογή των μοντέλων στα άγνωστα δείγματα, ή αλλιώς στα test sets.

Στον περιορισμό του overfitting βοήθησαν τα επίπεδα Dropout, η χρήση περισσότερων δεδομένων, η συνάρτηση ενεργοποίησης ReLU, αλλά και τα callbacks που χρησιμοποιήθηκαν για την παρακολούθηση της απόδοσης του μοντέλου κατά εκπαίδευση. Στην περίπτωση του μοντέλου των 345 κλάσεων, όπως φαίνεται και από τα διαγράμματα, η εκπαίδευση θα μπορούσε να σταματήσει και στην εποχή 25. Ωστόσο, το callback ReduceLROnPlateau έδωσε μία πολύ μικρή ώθηση στο μοντέλο και τελικά η εκπαίδευση σταμάτησε στην εποχή 30.



Εικόνα 4-7: Διάγραμμα training - validation accuracy του μοντέλου ταξινόμησης (345 κλάσεις)



Εικόνα 4-8: Διάγραμμα training - validation loss του μοντέλου ταξινόμησης (345 κλάσεις)

Πίνακας 4-4: Αποτελέσματα του μοντέλου στο test set (10 κλάσεις)

Classification test loss:	0.122
Classification test accuracy	0.966
Classification test top – 3 accuracy	0.988

Πίνακας 4-5: Αποτελέσματα του μοντέλου στο test set (345 κλάσεις)

Classification test loss:	1.156
Classification test accuracy	0.719
Classification test top – 3 accuracy	0.869

Για μία καλύτερη εικόνα της απόδοσης του μοντέλου των 10 κλάσεων, παρακάτω φαίνονται οι τιμές precision, recall και f1 score για κάθε μία από τις 10 κλάσεις. Η στήλη support αναφέρεται στον συνολικό αριθμό δειγμάτων που συμμετείχαν στο test set από κάθε κλάση και όπως φαίνεται υπάρχει ισορροπία στη συνεισφορά της κάθε κλάσης. Λόγω όγκου, τα αντίστοιχα αποτελέσματα για το μοντέλο των 345 κλάσεων αναφέρονται στο Παράρτημα Β – Classification Report.

Πίνακας 4-6: Precision, Recall, F1 - scores (10 κλάσεις)

	Precision	**Recall**	**F1 – score**	**Support**
Calculator	0.98	0.97	0.97	1982
Ice cream	0.98	0.98	0.98	2015
Jacket	0.96	0.96	0.96	2049
Pig	0.94	0.95	0.95	2017
Pineapple	0.97	0.96	0.97	1990
Postcard	0.97	0.96	0.97	1963
Rain	0.97	0.97	0.97	1973
Smiley face	0.98	0.96	0.97	2049
The Eiffel Tower	0.98	0.99	0.98	2026
Violin	0.94	0.95	0.94	1936
Avg / Total	0.97	0.97	0.97	20000

4.5 Visualizations

Στην επόμενη εικόνα, φαίνονται οι προβλέψεις των δύο μοντέλων σε 50 τυχαία δείγματα των test sets. Με μαύρα γράμματα σημειώνεται το όνομα της κάθε κλάσης (ground truth label). Στις περιπτώσεις που τα μοντέλα έκαναν σωστή πρόβλεψη, αναγράφεται με πράσινο χρώμα το όνομα της κάθε κλάσης πάνω από το αντίστοιχο σκίτσο. Για τις λάθος προβλέψεις, σημειώνεται πάνω από το σκίτσο με κόκκινα γράμματα το όνομα της κλάσης που προέβλεψαν τα μοντέλα (predicted

label), καθώς επίσης και η πιθανότητα της πρόβλεψής τους το σκίτσο να ανήκει στην κλάση αυτή.

Εικόνα 4-9: Προβλέψεις του μοντέλου σε σκίτσα του test set (10 κλάσεις)

Εικόνα 4-10: Προβλέψεις του μοντέλου σε σκίτσα του test set (345 κλάσεις)

5 Σύνοψη και συμπεράσματα

Τα Συνελικτικά Νευρωνικά Δίκτυα είναι μία ισχυρή τεχνική που χρησιμοποιήθηκε στην εργασία αυτή, για την υλοποίηση ενός πειράματος αρκετά μεγάλης κλίμακας, με στόχο την ταξινόμηση περίπου 2 εκατομμυρίων σκίτσων που ανήκουν σε 345 διαφορετικές κλάσεις. Τα δεδομένα που χρησιμοποιήθηκαν προέρχονται από ένα σύγχρονο ανοικτό σύνολο δεδομένων, το "Quick, Draw!", το οποίο από το 2018 χρησιμοποιείται σε εφαρμογές Μηχανικής Μάθησης με στόχο την προώθηση της έρευνας στα πεδία της ταξινόμησης εικόνας, της αναγνώρισης προτύπων και φυσικής γλώσσας, και της αναγνώρισης χειρόγραφων χαρακτήρων. Ως, αποτέλεσμα, προέκυψε ένα μοντέλο Βαθιάς Μάθησης, με την ικανότητα να προβλέπει την κλάση των σκίτσων με πιθανότητα 72 %.

Δεδομένης της ποικιλίας των σκίτσων και συνεπώς των ιδιαίτερων σχημάτων και χαρακτηριστικών που παρουσιάζει το κάθε ένα από αυτά, το μοντέλο μπορεί να θεωρηθεί μία πολύ καλή επέκταση του προβλήματος της ταξινόμησης χειρόγραφων ψηφίων από 0 – 9 (MNIST).

Ένα όχι τόσο αναμενόμενο αποτέλεσμα που προέκυψε είναι το γεγονός ότι το μοντέλο είναι ικανό να αναγνωρίζει εκτός από σκίτσα και λέξεις. Αυτό φυσικά προκύπτει από το γεγονός ότι ένα μεγάλο πλήθος των χρηστών που συνεισέφεραν στη δημιουργία του συνόλου δεδομένων, στις περιπτώσεις που δεν ήξεραν πώς να ζωγραφίσουν ένα σκίτσο, ή δεν καταλάβαιναν το όνομα της κλάσης, έγραφαν απλώς το όνομα της κλάσης αυτής.

Λαμβάνοντας υπόψη τις δυσκολίες που παρουσιάζει το συγκεκριμένο σύνολο δεδομένων καθώς και τον όγκο του, η ικανότητα πρόβλεψης του μοντέλου είναι αρκετά καλή και μπορεί εύκολα να συγκριθεί με την απόδοση ενός ατόμου στην ταξινόμηση του ίδιου τύπου σκίτσων.

Το μοντέλο έχει μάθει να αναγνωρίζει με μεγάλο βαθμό επιτυχίας σκίτσα από τις περισσότερες κλάσεις. Φυσικά, οι προβλέψεις του είναι καλύτερες στις κλάσεις που δεν παρουσιάζουν πολύ έντονες ομοιότητες με άλλες κλάσεις, όπως θα συνέβαινε και με το ανθρώπινο μάτι. Παρατηρώντας το classification report στο Παράρτημα Β – Classification Report, φαίνεται η δυσκολία του μοντέλου να διαχωρίσει τα χαρακτηριστικά όμοιων κλάσεων, με τη μικρότερη απόδοση να σχετίζεται με τις κλάσεις σκίτσων που αναπαριστούν τετράποδα ζώα. Ωστόσο, το αποτέλεσμα αυτό ήταν αναμενόμενο, καθώς είναι ιδιαίτερα δύσκολο να ζωγραφίσει κάποιος ένα μικρό σκίτσο με τόση λεπτομέρεια, ώστε να γίνεται αντιληπτό αμέσως από το ανθρώπινο μάτι το ζώο που αναπαρίσταται.

Συμπερασματικά, το μοντέλο που σχεδιάστηκε στα πλαίσια της εργασίας αυτής, αποτελεί ένα καλό εργαλείο αναγνώρισης εικόνας και χειρόγραφων σκίτσων που ως ένα βαθμό συναντά της ίδιες δυσκολίες με το ανθρώπινο μάτι, καθώς η απόδοσή του εξαρτάται από την είσοδο.

5.1 Μελλοντικές Επεκτάσεις

Όπως προαναφέρθηκε, η διάθεση του συνόλου δεδομένων “Quick, Draw!” στο κοινό είναι πολύ πρόσφατη και αυτό αφήνει ανοιχτά επιπλέον περιθώρια έρευνας.

Φυσικά, στην καλύτερη απόδοση ενός μοντέλου ταξινόμησης του “Quick, Draw!” θα βοηθούσε η χρήση και των 50 εκατομμυρίων σκίτσων, πράγμα που απαιτεί την εκπαίδευση του μοντέλου σε περισσότερες από μία GPU. Επίσης, με την επέκταση του συνόλου δεδομένων, είναι αναγκαία και η χρήση ενός πολύ μεγαλύτερου μοντέλου από αυτό που χρησιμοποιήθηκε στη διπλωματική αυτή εργασία. Πιθανώς η χρήση ενός ResNet με πολλά επίπεδα θα έδινε μία μεγάλη ώθηση στα αποτελέσματα της ταξινόμησης.

Σχετικά με τις κλάσεις που το μοντέλο παρουσιάζει αστοχίες, θα μπορούσε αρχικά να γίνει μία ταξινόμηση μόνο αυτών των κλάσεων, ώστε να αποφασιστεί το όριο του αναμενόμενου αποτελέσματος από το τελικό μοντέλο. Επιπλέον, θα μπορούσε να εφαρμοστεί προεπεξεργασία των δεδομένων που θα αφαιρεί τα δείγματα τα οποία δεν είναι ολοκληρωμένα, ή που παρουσιάζουν έντονη απόκλιση από τα υπόλοιπα δείγματα της κλάσης.

Βιβλιογραφία

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, “Gradient-Based Learning Applied to Document Recognition,” p. 46, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *ArXiv14091556 Cs*, Sep. 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *ArXiv151203385 Cs*, Dec. 2015.

[5] “Quick, Draw!” [Online]. Available: https://quickdraw.withgoogle.com/.

[6] D. Ha and D. Eck, “A Neural Representation of Sketch Drawings,” *ArXiv170403477 Cs Stat*, Apr. 2017.

[7] "*Documentation on how to access and use the Quick, Draw!" Dataset.: googlecreativelab/quickdraw-dataset*. Google Creative Lab, 2019.

[8] “Quick, Draw! Doodle Recognition Challenge.” [Online]. Available: https://kaggle.com/c/quickdraw-doodle-recognition.

[9] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales, “Sketch-a-Net that Beats Humans” *ArXiv150107873 Cs*, Jan. 2015.

[10] C. Li, “Comparison of recognition on hand written digits between MLP network and CNNs,” p. 3, 2018.

[11] “Neural Networks - Biology.” [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Biology/index.html.

[12] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[13] F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain” *Psychol. Rev.*, pp. 65–386, 1958.

[14] “Unsupervised Feature Learning and Deep Learning Tutorial.” [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/.

[15] N. Qian, "*On the Momentum Term in Gradient Descent Learning Algorithms*." 1999.

[16] J. C. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization” *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.

[17] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *ArXiv12125701 Cs*, Dec. 2012.

[18] G. Hinton, "rmsprop: A mini-batch version of rprop", [unpublished] [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf .

[19] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization” *ArXiv14126980 Cs*, Dec. 2014.

[20] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning” *ArXiv160307285 Cs Stat*, Mar. 2016.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting” p. 30, 2014.

[22] H. K. Jabbar and R. Z. Khan, “Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study),” in *Computer Science, Communication and Instrumentation Devices*, 2014, pp. 163–172.
[23] “TensorFlow,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/.
[24] “Home - Keras Documentation.” [Online]. Available: https://keras.io/.
[25] "*Deep Learning for humans." Contribute to keras-team/keras development by creating an account on GitHub*. Keras, 2019.
[26] “PyTorch.” [Online]. Available: https://www.pytorch.org.
[27] “Different languages: How cultures around the world draw shapes differently — Quartz.” [Online]. Available: https://qz.com/994486/the-way-you-draw-circles-says-a-lot-about-you/.
[28] “Google AI Blog: Exploring and Visualizing an Open Global Dataset.” [Online]. Available: https://ai.googleblog.com/2017/08/exploring-and-visualizing-open-global.html.
[29] A. Wong, "*Classifying noisy Google QuickDraw images (keras)." Contribute to ankonzoid/Google-QuickDraw development by creating an account on GitHub*. 2019.
[30] S. Pal, "*Identifying hand drawn images from Google Quick draw dataset with Keras using Convolutional Neural networks": subarnop/Kiddo*. 2018.
[31] “Recurrent Neural Networks for Drawing Classification | TensorFlow Core,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/tutorials/sequences/recurrent_quickdraw.

Παράρτημα Α - Dataset

Οι 345 κλάσεις του συνόλου δεδομένων “Quick, Draw!”

```
aircraft carrier
airplane
alarm clock
ambulance
angel
animal migration
ant
anvil
apple
arm
asparagus
axe
backpack
banana
bandage
barn
baseball
baseball bat
basket
basketball
bat
bathtub
beach
bear
beard
bed
bee
belt
bench
bicycle
binoculars
bird
birthday cake
blackberry
blueberry
book
boomerang
bottlecap
bowtie
bracelet
brain
bread
bridge
broccoli
broom
bucket
bulldozer
bus
bush
butterfly
cactus
cake
calculator
calendar
```

camel
camera
camouflage
campfire
candle
cannon
canoe
car
carrot
castle
cat
ceiling fan
cello
cell phone
chair
chandelier
church
circle
clarinet
clock
cloud
coffee cup
compass
computer
cookie
cooler
couch
cow
crab
crayon
crocodile
crown
cruise ship
cup
diamond
dishwasher
diving board
dog
dolphin
donut
door
dragon
dresser
drill
drums
duck
dumbbell
ear
elbow
elephant
envelope
eraser
eye
eyeglasses
face
fan
feather
fence
finger
fire hydrant

fireplace
firetruck
fish
flamingo
flashlight
flip flops
floor lamp
flower
flying saucer
foot
fork
frog
frying pan
garden
garden hose
giraffe
goatee
golf club
grapes
grass
guitar
hamburger
hammer
hand
harp
hat
headphones
hedgehog
helicopter
helmet
hexagon
hockey puck
hockey stick
horse
hospital
hot air balloon
hot dog
hot tub
hourglass
house
house plant
hurricane
ice cream
jacket
jail
kangaroo
key
keyboard
knee
knife
ladder
lantern
laptop
leaf
leg
light bulb
lighter
lighthouse
lightning
line

lion
lipstick
lobster
lollipop
mailbox
map
marker
matches
megaphone
mermaid
microphone
microwave
monkey
moon
mosquito
motorbike
mountain
mouse
moustache
mouth
mug
mushroom
nail
necklace
nose
ocean
octagon
octopus
onion
oven
owl
paintbrush
paint can
palm tree
panda
pants
paper clip
parachute
parrot
passport
peanut
pear
peas
pencil
penguin
piano
pickup truck
picture frame
pig
pillow
pineapple
pizza
pliers
police car
pond
pool
popsicle
postcard
potato
power outlet

purse
rabbit
raccoon
radio
rain
rainbow
rake
remote control
rhinoceros
rifle
river
roller coaster
rollerskates
sailboat
sandwich
saw
saxophone
school bus
scissors
scorpion
screwdriver
sea turtle
see saw
shark
sheep
shoe
shorts
shovel
sink
skateboard
skull
skyscraper
sleeping bag
smiley face
snail
snake
snorkel
snowflake
snowman
soccer ball
sock
speedboat
spider
spoon
spreadsheet
square
squiggle
squirrel
stairs
star
steak
stereo
stethoscope
stitches
stop sign
stove
strawberry
streetlight
string bean
submarine

suitcase
sun
swan
sweater
swing set
sword
syringe
table
teapot
teddy-bear
telephone
television
tennis racquet
tent
The Eiffel Tower
The Great Wall of China
The Mona Lisa
tiger
toaster
toe
toilet
tooth
toothbrush
toothpaste
tornado
tractor
traffic light
train
tree
triangle
trombone
truck
trumpet
t-shirt
umbrella
underwear
van
vase
violin
washing machine
watermelon
waterslide
whale
wheel
windmill
wine bottle
wine glass
wristwatch
yoga
zebra
zigzag

Παράρτημα Β - Classification Report

Τα αποτελέσματα του Classification Report για το μοντέλο των 345 κλάσεων.

	precision	recall	f1-score	support
rifle	0.65	0.70	0.67	1274
hedgehog	0.72	0.79	0.75	1320
axe	0.79	0.78	0.78	1269
floor lamp	0.83	0.72	0.77	1306
t-shirt	0.80	0.85	0.83	1281
bottlecap	0.67	0.54	0.60	1277
sheep	0.80	0.82	0.81	1308
candle	0.77	0.80	0.79	1242
ocean	0.68	0.64	0.66	1258
diamond	0.82	0.85	0.83	1234
mountain	0.84	0.80	0.82	1225
cloud	0.73	0.80	0.76	1285
laptop	0.72	0.82	0.77	1319
steak	0.58	0.41	0.48	1320
wheel	0.75	0.78	0.76	1292
soccer ball	0.62	0.83	0.71	1300
cell phone	0.68	0.75	0.71	1329
sock	0.82	0.78	0.80	1313
calendar	0.53	0.62	0.57	1237
zebra	0.72	0.76	0.74	1268
couch	0.79	0.82	0.80	1287
roller coaster	0.65	0.62	0.63	1317
river	0.61	0.68	0.64	1233
firetruck	0.65	0.56	0.60	1318
chandelier	0.80	0.74	0.77	1293
van	0.56	0.54	0.55	1274
lobster	0.54	0.49	0.52	1287
tennis racquet	0.75	0.82	0.79	1261
tiger	0.51	0.48	0.50	1283
crayon	0.58	0.44	0.50	1316
snail	0.82	0.88	0.85	1313
bucket	0.68	0.69	0.69	1336
blueberry	0.55	0.53	0.54	1252
squiggle	0.52	0.40	0.45	1308
fireplace	0.78	0.76	0.77	1320
ice cream	0.89	0.87	0.88	1291
camel	0.87	0.87	0.87	1279
baseball	0.77	0.74	0.75	1292
triangle	0.87	0.93	0.90	1241
piano	0.62	0.64	0.63	1274
necklace	0.74	0.70	0.72	1287
mushroom	0.80	0.83	0.82	1288
The Eiffel Tower	0.89	0.90	0.90	1293
scissors	0.82	0.84	0.83	1258
lion	0.76	0.76	0.76	1253
toothbrush	0.73	0.79	0.76	1311
pig	0.67	0.70	0.68	1294
suitcase	0.67	0.68	0.68	1297
beach	0.58	0.67	0.62	1326
garden	0.52	0.67	0.58	1273
trumpet	0.66	0.63	0.64	1276

stethoscope	0.86	0.78	0.82	1270
yoga	0.62	0.61	0.62	1294
mouth	0.78	0.80	0.79	1300
hockey stick	0.66	0.60	0.63	1247
blackberry	0.42	0.55	0.48	1282
duck	0.62	0.60	0.61	1297
palm tree	0.79	0.81	0.80	1189
crab	0.77	0.75	0.76	1325
tent	0.87	0.79	0.83	1288
feather	0.62	0.68	0.65	1265
matches	0.61	0.51	0.55	1308
leg	0.58	0.57	0.57	1294
rhinoceros	0.73	0.71	0.72	1309
snorkel	0.85	0.76	0.80	1317
snowman	0.88	0.91	0.89	1363
hot air balloon	0.80	0.80	0.80	1312
cooler	0.47	0.38	0.42	1241
teddy-bear	0.72	0.76	0.74	1272
computer	0.79	0.71	0.75	1306
hand	0.85	0.86	0.85	1284
streetlight	0.76	0.73	0.75	1309
bread	0.73	0.59	0.65	1292
oven	0.62	0.41	0.50	1301
rollerskates	0.87	0.89	0.88	1298
lollipop	0.81	0.85	0.83	1307
cello	0.59	0.57	0.58	1283
violin	0.54	0.45	0.49	1268
eyeglasses	0.87	0.85	0.86	1272
saxophone	0.80	0.81	0.81	1285
drums	0.70	0.76	0.73	1287
hammer	0.76	0.79	0.78	1276
church	0.76	0.73	0.74	1281
jacket	0.82	0.73	0.77	1249
paintbrush	0.65	0.59	0.62	1264
windmill	0.81	0.81	0.81	1303
diving board	0.67	0.49	0.57	1265
moon	0.77	0.59	0.67	1343
car	0.62	0.66	0.64	1305
mermaid	0.74	0.88	0.80	1357
fence	0.73	0.76	0.75	1275
canoe	0.74	0.75	0.74	1303
skyscraper	0.71	0.58	0.64	1286
clarinet	0.59	0.58	0.59	1285
elbow	0.73	0.68	0.70	1253
cactus	0.87	0.83	0.85	1276
tornado	0.57	0.82	0.67	1343
parachute	0.84	0.83	0.83	1260
moustache	0.81	0.73	0.77	1336
cow	0.56	0.70	0.62	1202
knife	0.70	0.64	0.67	1270
megaphone	0.84	0.77	0.80	1326
sun	0.82	0.92	0.87	1310
birthday cake	0.56	0.54	0.55	1261
microwave	0.73	0.86	0.79	1286
sleeping bag	0.66	0.61	0.63	1303
garden hose	0.53	0.23	0.32	1326
ladder	0.92	0.93	0.93	1287
toe	0.72	0.63	0.68	1319
marker	0.43	0.30	0.36	1336
goatee	0.79	0.70	0.74	1347

tooth	0.84	0.80	0.82	1297
flashlight	0.81	0.84	0.82	1296
hot dog	0.68	0.75	0.71	1269
swing set	0.88	0.89	0.88	1209
key	0.81	0.75	0.78	1273
kangaroo	0.70	0.79	0.74	1316
cookie	0.72	0.81	0.76	1273
hurricane	0.34	0.41	0.37	1336
dog	0.53	0.38	0.44	1266
apple	0.83	0.88	0.86	1256
see saw	0.88	0.81	0.84	1330
bandage	0.83	0.78	0.81	1298
cat	0.71	0.68	0.69	1325
airplane	0.73	0.80	0.77	1262
snake	0.64	0.66	0.65	1297
zigzag	0.78	0.82	0.80	1271
popsicle	0.87	0.82	0.85	1347
remote control	0.77	0.80	0.78	1340
spider	0.78	0.76	0.77	1291
line	0.64	0.91	0.75	1273
stitches	0.83	0.78	0.81	1251
fish	0.86	0.85	0.86	1287
spreadsheet	0.64	0.62	0.63	1231
fork	0.86	0.79	0.83	1259
flamingo	0.76	0.78	0.77	1244
truck	0.56	0.42	0.48	1277
bicycle	0.73	0.78	0.76	1280
dumbbell	0.84	0.79	0.81	1242
angel	0.86	0.84	0.85	1308
eraser	0.66	0.61	0.63	1263
grass	0.72	0.76	0.74	1377
paper clip	0.85	0.84	0.85	1303
campfire	0.71	0.82	0.76	1296
knee	0.69	0.68	0.69	1338
lipstick	0.73	0.73	0.73	1252
penguin	0.75	0.80	0.77	1270
castle	0.78	0.81	0.80	1326
lightning	0.80	0.76	0.78	1305
rabbit	0.75	0.78	0.77	1273
frog	0.56	0.49	0.52	1337
radio	0.65	0.70	0.67	1204
pizza	0.78	0.77	0.77	1340
crocodile	0.68	0.74	0.71	1363
boomerang	0.78	0.76	0.77	1324
leaf	0.76	0.72	0.74	1300
lighter	0.73	0.82	0.77	1264
camouflage	0.20	0.45	0.28	1299
teapot	0.82	0.83	0.83	1202
bed	0.74	0.79	0.77	1270
mailbox	0.74	0.80	0.77	1305
barn	0.73	0.68	0.70	1309
mosquito	0.53	0.54	0.54	1294
baseball bat	0.75	0.77	0.76	1219
nose	0.77	0.73	0.75	1276
anvil	0.86	0.80	0.83	1296
cake	0.54	0.57	0.55	1313
nail	0.74	0.60	0.66	1285
wine glass	0.91	0.90	0.90	1362
sink	0.74	0.77	0.76	1322
backpack	0.77	0.79	0.78	1333

ceiling fan	0.72	0.79	0.75	1325
camera	0.83	0.85	0.84	1295
shovel	0.76	0.75	0.75	1353
stop sign	0.91	0.91	0.91	1317
binoculars	0.83	0.78	0.80	1342
fan	0.72	0.64	0.68	1270
bathtub	0.69	0.69	0.69	1276
horse	0.69	0.77	0.73	1305
bench	0.63	0.57	0.60	1294
peas	0.60	0.67	0.64	1259
peanut	0.76	0.74	0.75	1245
screwdriver	0.71	0.70	0.70	1254
bush	0.66	0.56	0.60	1246
house plant	0.83	0.86	0.84	1260
purse	0.73	0.69	0.71	1339
paint can	0.58	0.56	0.57	1316
basketball	0.71	0.67	0.69	1294
map	0.68	0.77	0.72	1271
bee	0.76	0.83	0.79	1299
stereo	0.65	0.67	0.66	1279
tractor	0.73	0.70	0.71	1304
skateboard	0.87	0.89	0.88	1280
pineapple	0.83	0.87	0.85	1331
toilet	0.83	0.83	0.83	1261
golf club	0.71	0.66	0.69	1338
grapes	0.58	0.75	0.65	1296
picture frame	0.78	0.83	0.80	1331
hot tub	0.64	0.51	0.56	1299
speedboat	0.56	0.70	0.62	1235
underwear	0.84	0.75	0.79	1265
school bus	0.46	0.60	0.52	1307
toaster	0.71	0.72	0.72	1294
giraffe	0.86	0.86	0.86	1290
syringe	0.83	0.78	0.80	1264
clock	0.84	0.85	0.84	1242
compass	0.80	0.79	0.80	1270
bear	0.50	0.32	0.39	1276
broom	0.64	0.75	0.69	1314
mug	0.55	0.64	0.59	1256
ear	0.85	0.82	0.84	1326
motorbike	0.62	0.63	0.62	1252
bracelet	0.63	0.65	0.64	1284
house	0.79	0.81	0.80	1334
sword	0.79	0.80	0.80	1246
bird	0.53	0.38	0.44	1337
alarm clock	0.76	0.81	0.79	1267
harp	0.86	0.81	0.83	1286
fire hydrant	0.73	0.67	0.70	1287
trombone	0.59	0.49	0.53	1262
microphone	0.71	0.72	0.71	1344
toothpaste	0.58	0.52	0.54	1292
rake	0.77	0.66	0.71	1270
bus	0.54	0.49	0.51	1286
book	0.80	0.80	0.80	1289
The Great Wall of China	0.47	0.42	0.45	1299
waterslide	0.68	0.64	0.66	1284
chair	0.85	0.86	0.85	1293
dishwasher	0.62	0.55	0.58	1290
sailboat	0.84	0.89	0.87	1254
mouse	0.63	0.50	0.56	1297

frying pan	0.78	0.78	0.78	1291
octopus	0.86	0.86	0.86	1281
belt	0.67	0.72	0.69	1283
butterfly	0.88	0.91	0.89	1269
stove	0.70	0.73	0.71	1369
helicopter	0.78	0.87	0.82	1280
bat	0.81	0.69	0.74	1279
pickup truck	0.61	0.61	0.61	1263
donut	0.82	0.90	0.86	1277
parrot	0.58	0.63	0.60	1314
crown	0.89	0.88	0.89	1340
hat	0.83	0.74	0.78	1262
sweater	0.77	0.71	0.74	1327
hourglass	0.91	0.90	0.90	1318
octagon	0.74	0.69	0.71	1363
shark	0.71	0.75	0.73	1270
eye	0.82	0.87	0.84	1319
pliers	0.77	0.54	0.64	1277
jail	0.79	0.82	0.81	1298
scorpion	0.75	0.74	0.75	1289
hockey puck	0.63	0.64	0.63	1326
watermelon	0.70	0.62	0.66	1264
asparagus	0.57	0.62	0.59	1249
drill	0.82	0.74	0.78	1309
ambulance	0.64	0.72	0.68	1345
headphones	0.93	0.92	0.92	1281
power outlet	0.79	0.72	0.75	1255
table	0.73	0.80	0.77	1275
flower	0.80	0.88	0.84	1279
washing machine	0.78	0.87	0.82	1266
envelope	0.89	0.94	0.91	1256
coffee cup	0.56	0.50	0.53	1222
pear	0.82	0.83	0.83	1278
pillow	0.66	0.66	0.66	1330
string bean	0.58	0.39	0.47	1212
cannon	0.66	0.73	0.69	1250
train	0.59	0.76	0.66	1294
guitar	0.59	0.70	0.64	1302
wristwatch	0.79	0.77	0.78	1290
lighthouse	0.73	0.77	0.75	1297
sea turtle	0.73	0.79	0.76	1285
onion	0.79	0.66	0.72	1243
owl	0.77	0.71	0.74	1321
sandwich	0.66	0.71	0.68	1305
basket	0.70	0.63	0.67	1271
television	0.89	0.86	0.87	1329
vase	0.82	0.77	0.79	1297
light bulb	0.77	0.76	0.77	1303
postcard	0.76	0.66	0.71	1309
rainbow	0.85	0.93	0.89	1273
telephone	0.74	0.53	0.62	1269
beard	0.65	0.71	0.68	1281
arm	0.71	0.69	0.70	1366
panda	0.66	0.67	0.67	1238
aircraft carrier	0.42	0.25	0.31	1339
skull	0.89	0.88	0.88	1338
tree	0.76	0.79	0.77	1291
brain	0.61	0.67	0.64	1307
calculator	0.78	0.88	0.82	1326
wine bottle	0.85	0.87	0.86	1243

swan	0.71	0.68	0.69	1336
elephant	0.73	0.70	0.71	1262
finger	0.75	0.71	0.73	1276
star	0.92	0.92	0.92	1298
face	0.73	0.70	0.72	1349
stairs	0.91	0.90	0.90	1321
cup	0.52	0.41	0.45	1264
square	0.71	0.84	0.77	1287
hamburger	0.79	0.84	0.81	1302
pond	0.51	0.46	0.48	1275
police car	0.73	0.62	0.67	1271
smiley face	0.76	0.81	0.79	1258
squirrel	0.75	0.75	0.75	1301
whale	0.76	0.72	0.74	1322
animal migration	0.49	0.69	0.57	1253
potato	0.61	0.59	0.60	1294
dolphin	0.72	0.68	0.70	1252
hexagon	0.73	0.71	0.72	1271
keyboard	0.58	0.65	0.62	1319
pants	0.84	0.77	0.80	1220
raccoon	0.52	0.39	0.44	1300
umbrella	0.91	0.91	0.91	1289
snowflake	0.84	0.88	0.86	1282
door	0.80	0.84	0.82	1310
shorts	0.74	0.86	0.80	1362
traffic light	0.83	0.87	0.85	1316
ant	0.69	0.82	0.75	1286
shoe	0.74	0.80	0.77	1311
circle	0.67	0.86	0.76	1314
submarine	0.82	0.77	0.80	1276
saw	0.86	0.82	0.84	1264
dragon	0.36	0.55	0.43	1287
The Mona Lisa	0.80	0.87	0.84	1307
carrot	0.82	0.84	0.83	1249
lantern	0.69	0.64	0.67	1298
strawberry	0.87	0.85	0.86	1336
spoon	0.74	0.72	0.73	1332
flying saucer	0.77	0.69	0.73	1328
hospital	0.74	0.76	0.75	1307
bulldozer	0.62	0.77	0.69	1280
pool	0.54	0.32	0.40	1249
bowtie	0.91	0.86	0.88	1293
rain	0.82	0.90	0.86	1226
dresser	0.75	0.70	0.73	1334
cruise ship	0.73	0.73	0.73	1308
helmet	0.73	0.65	0.69	1300
pencil	0.54	0.70	0.61	1254
broccoli	0.66	0.69	0.68	1273
flip flops	0.79	0.79	0.79	1339
foot	0.69	0.67	0.68	1226
bridge	0.76	0.65	0.70	1278
passport	0.61	0.62	0.62	1304
banana	0.68	0.76	0.72	1275
monkey	0.54	0.65	0.59	1262
micro avg	0.72	0.72	0.72	445050
macro avg	0.72	0.72	0.72	445050
weighted avg	0.72	0.72	0.72	445050

Παράρτημα C - Κώδικας

Ο κώδικας των πειραμάτων, καθώς και όλα τα visualizations που παρουσιάστηκαν στο κεφάλαιο 4 είναι διαθέσιμα στο link:
https://drive.google.com/open?id=1YMNYTTydDRom69KlhHsZiGvMtsRwakQ6

Λόγω όγκου (~40GB) τα δεδομένα δεν περιέχονται στα αντίγραφα των πειραμάτων και πρέπει να γίνει download των numpy σκίτσων του "Quick, Draw!" από το link:
https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap/?pli=1

Για την εκτέλεση του κώδικα του τελικού πειράματος, απαιτείται η εγκατάσταση της Python3 και του Anaconda.

Στη συνέχεια:

1. Δημιουργία ενός conda environment

```
conda create --name quickDraw
```

2. Ενεργοποίηση του νέου conda environment

```
conda activate quickDraw
```

3. Εγκατάσταση των απαραίτητων Python packages

```
conda install tensorflow
or
conda install tensorflow-gpu
```

```
conda install keras
conda install matplotlib
conda install scikit – learn
```

4. Τοποθέτηση του φακέλου των .npy αρχείων στο φάκελο Final Eperiment με όνομα numpy_bitmap_sketches.

5. Εκτέλεση του αρχείου `np_quick_draw_classifier.py`

```
python3 np_quick_draw_classifier.py
```