ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Αυτόνομη Πλοήγηση με Χρήση Μηχανικής Μάθησης και Επαυξημένης Πραγματικότητας

Χρήστος Λευκαδίτης

Επιβλέπων

Σταμούλης Γεώργιος

Κολομβάτσος Κωνσταντίνος

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

2. Δέχομαι ότι η αυτολεξεί ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

Περίληψη

Βρισκόμαστε στην εποχή που ο τομέας της τεχνητής νοημοσύνης και συγκεκριμένα το κομμάτι της μηχανικής μάθησης βρίσκει πολλές και διαφορετικές εφαρμογές. Από προβλέψεις, σχετικά με τις προτιμήσεις των καταναλωτών στον τομέα της διαφήμισης, μέχρι πλήρως αυτόνομα οχήματα που δεν απαιτούν ανθρώπινη παρουσία ως προς τον χειρισμό τους. Η απόδοση των αλγορίθμων τεχνητής νοημοσύνης αυξάνεται συνεχώς και αυτό οφείλεται στο υλικό των υπολογιστών που γίνεται καθημερινά καλύτερο. Συνεπώς δεν είναι λίγες οι φορές που η ικανότητα των αλγορίθμων αυτών να πάρουν συγκεκριμένες αποφάσεις ξεπερνάει αυτή του ανθρώπου, πράγμα που κάνει την τεχνητή νοημοσύνη ένα πολύτιμο εργαλείο για τον άνθρωπο. Σε αυτή την εργασία θα γίνει ανάλυση της αρχιτεκτονικής αλγορίθμων μηχανικής μάθησης καθώς και της εφαρμογής τους για την κατασκευή ενός αυτόνομου οχήματος.

Περιεχόμενα

1. Τεχνητή Νοημοσύνη

1.1 Γενικά

Ο όρος **τεχνητή νοημοσύνη (Artificial Intelligence)** είναι συνδεδεμένος με τον τομέα της πληροφορικής, ο οποίος ασχολείται με την υλοποίηση συστημάτων που μιμούνται την ανθρώπινη συμπεριφορά και παρουσιάζουν στοιχειώδη ευφυΐα. Γνωρίσματα τέτοιων συστημάτων είναι η μάθηση και συνεπώς η προσαρμοστικότητα σε διαφορετικά περιβάλλοντα, η εξαγωγή συμπερασμάτων και η επίλυση προβλημάτων. Η τεχνητή νοημοσύνη έχει αρχίσει και γίνεται όλο και πιο δημοφιλής τα τελευταία χρόνια λόγω της αύξησης του όγκου των δεδομένων, της βελτιστοποίησης αλγορίθμων και φυσικά λόγω της αύξησης στην ισχύ των υπολογιστικών συστημάτων.

1.2 Πρώτα Βήματα

Η γέννηση της Τεχνητής Νοημοσύνης χρονολογείται το 1950, όταν ο Βρετανός **Alan Turing** ξεκίνησε να ερευνά την πιθανότητα της δημιουργίας μηχανών που έχουν την ικανότητα να σκέφτονται. Για να μπορέσει να καταλήξει αν κάτι τέτοιο είναι πιθανό, δημιούργησε το **Turing Test** το οποίο στην συνέχεια δημοσίευσε. Αν μια μηχανή μπορούσε να διεξάγει μια συζήτηση (μέσω τηλέτυπου), η οποία ήταν πανομοιότυπη με αυτή που γίνεται μεταξύ ανθρώπων, τότε το συμπέρασμα του συγκεκριμένου τεστ είναι πως είναι πιθανή η σκέψη σε μια μηχανή. Οι απαντήσεις της μηχανής δεν αξιολογήθηκαν με βάση πόσο σωστές ήταν, αλλά ως προς το είδος της απάντησης, δηλαδή πόσο κοντά βρίσκονται σε μια απάντηση που θα έδινε κανονικά ένας άνθρωπος. Το Turing Test αποτέλεσε ένα από τα πρώτα βήματα που οδήγησαν στη δημιουργία του επιστημονικού πεδίου της Τεχνητής Νοημοσύνης.

1.3 Ορισμός

Η τεχνητή νοημοσύνη ορίζεται απο τον Τζόν Μακάρθι ως «επιστήμη και μεθοδολογία της δημιουργίας νοημόνων μηχανών». Ένας γενικός ορισμός, όμως, αναφέρει ότι: «Τεχνητή Νοημοσύνη είναι ο τομέας της Επιστήμης των Υπολογιστών που ασχολείται με τη σχεδίαση και την υλοποίηση προγραμμάτων τα οποία είναι ικανά να μιμηθούν τις ανθρώπινες γνωστικές ικανότητες, εμφανίζοντας έτσι χαρακτηριστικά που αποδίδουμε συνήθως σε ανθρώπινη συμπεριφορά, όπως η επίλυση προβλημάτων, η αντίληψη μέσω της όρασης, η μάθηση, η εξαγωγή συμπερασμάτων, η κατανόηση φυσικής γλώσσας, κλπ.»

1.4 Είδη Τεχνητής Νοημοσύνης

Διαιρείται στην **κλασσική ή συμβολική τεχνητή νοημοσύνη (symbolic Artificial Intelligence)**, η οποία βασίζεται στην κατανόηση της νοημοσύνης του ανθρώπου και στην εξομοίωση της αλγοριθμικά, χρησιμοποιώντας σύμβολα και λογικούς κανόνες υψηλού επιπέδου, και στην **υποσυμβολική** ή **υπολογιστική νοημοσύνη (computational intelligence)**, η οποία μιμείται τον ανθρώπινο εγκέφαλο, χρησιμοποιώντας αριθμητικά μοντέλα που προσομοιώνουν πραγματικές βιολογικές διαδικασίες, όπως η εξέλιξη των ειδών και η λειτουργία του εγκεφάλου, όπως για παράδειγμα νευρωνικά δίκτυα και γενετικοί αλγόριθμοι.

Η όρος της τεχνητής νοημοσύνης αναφέρεται σε κάτι αρκετά γενικό, καθώς περιλαμβάνει οποιοδήποτε σύστημα παρουσιάζει στοιχειώδη ευφυΐα. Λόγω της ευρύτητας αυτής, στην εποχή μας μπορούμε να βρούμε δείγματα τεχνητής νοημοσύνης, κυριολεκτικά αν κοιτάξουμε γύρω μας. Από τον διαχωρισμό ανεπιθύμητης αλληλογραφίας που γίνεται αυτόματα στο email μας μέχρι την χρήση αναγνώρισης προσώπου σε ένα κινητό τηλέφωνο. Ωστόσο, αν τις κατηγοριοποιήσουμε, οι βασικές εφαρμογές της τεχνητής νοημοσύνης είναι: η **επεξεργασία φυσικής γλώσσας** (Natural Language Processing) όπου γίνεται χειρισμός της φυσικής γλώσσας για δημιουργία συστημάτων text-to-speech ή speech-to-text, η **τεχνητή όραση** που επιτρέπει σε μια μηχανή να έχει όραση και να την μετατρέπει σε πληροφορία, η

επίλυση προβλημάτων, τα **έμπειρα συστήματα** και τα **συστήματα γνώσης**. Επιπρόσθετα πολλές φορές γίνονται εφαρμογές τεχνητής νοημοσύνης ακόμα και σε διαφορετικά επιστημονικά πεδία, όπως η **ρομποτική** στην οποία έχει γίνει μεγάλη πρόοδος με πρότυπα ρομπότ να έχουν την δυνατότητα ομιλίας και διεξαγωγής συζήτησης.

2. Μηχανική μάθηση

2.1 Γενικά

Η μηχανική μάθηση αποτελεί υποκατηγορία της Τεχνητής Νοημοσύνης. Είναι το κομμάτι που δημιουργεί στατιστικά μοντέλα μέσα από μεγάλο όγκο δεδομένων εκπαίδευσης, τα οποία αναγνωρίζονται από υπολογιστικά συστήματα. Τα μοντέλα που παράγονται μετά την εκπαίδευση είναι σε θέση να πάρουν αποφάσεις σχετικά με το ζητούμενο πρόβλημα. Όσα περισσότερα δεδομένα μπορούμε να παρέχουμε στο μοντέλο μας, τόσο μεγαλύτερη ακρίβεια θα μπορέσουμε να πετύχουμε στις αποφάσεις του.

Εικόνα 1.1

2.2 Ορισμός

Η **μηχανική μάθηση (machine learning)** ορίζεται το 1959, από τον Άρθουρ Σάμουελ ως «Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί». Ο επίσημος ορισμός, όμως, που χρησιμοποιείται ευρέως μέχρι και σήμερα είναι: «Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία E ως προς μια κλάση εργασιών T και ένα μέτρο επίδοσης P, αν η επίδοσή του σε εργασίες της κλάσης T, όπως αποτιμάται από το μέτρο P, βελτιώνεται με την εμπειρία E».

2.3 Είδη μηχανικής μάθησης

Τα είδη μηχανικής μάθησης ταξινομούνται σε τρεις κατηγορίες, ανάλογα με την διαδικασία της εκπαίδευσης που χρησιμοποιείται σε κάθε μια. Οι κατηγορίες είναι οι εξής:

Επιβλεπόμενη μάθηση (Supervised learning): Το μοντέλο που εκπαιδεύεται στην συγκεκριμένη κατηγορία «μαθαίνει» από ένα σύνολο δεδομένων, τα οποία του τροφοδοτούμε ως είσοδο και προβλέπει μια ή περισσότερες εξόδους.

Μη επιβλεπόμενη μάθηση (Unsupervised Learning): Το μοντέλο εκπαιδεύεται για ένα σύνολο εισόδων, οι οποίες δίνονται στη μορφή παρατηρήσεων χωρίς καμία γνώση των επιθυμητών εξόδων.

Ενισχυτική μάθηση (Reinforcement learning): Το μοντέλο σε αυτή την κατηγορία προσπαθεί να μάθει μέσα από την άμεση αλληλεπίδραση του με συγκεκριμένο περιβάλλον που του παρέχουμε. Πιο συγκεκριμένα, οι αλγόριθμοι ενισχυτικής μάθησης δουλεύουν με ένα σύστημα ενίσχυσης (ανταμοιβής) για οποιαδήποτε φορά κάνουν κάτι σωστό. Έτσι, ο αλγόριθμος αυτός στην προσπάθεια του να μεγιστοποιήσει το κέρδος του, ανακαλύπτει μόνος του τις ενέργειες που πρέπει να ακολουθήσει.

2.4 Πεδία εφαρμογής

Οι αλγόριθμοι και οι τεχνικές μηχανικής μάθησης μπορούν να χρησιμοποιηθούν σε ένα μεγάλο εύρος εφαρμογών. Κύρια πεδία εφαρμογής είναι:

Οικονομικά: Οι εταιρίες οι οποίες ασχολούνται με τον συγκεκριμένο κλάδο μπορούν να επωφεληθούν αρκετά από τον τομέα της μηχανικής μάθησης. Μοντέλα, που μπορούν να προβλέψουν επενδυτικές ευκαιρίες καθώς και επικείμενες οικονομικές απάτες προς αποφυγή, μπορούν να υλοποιηθούν, ώστε οι εταιρίες να είναι σε θέση να αυξήσουν τα κέρδη τους και να μειώσουν την ζημία τους.

Διαφήμιση: Στον τομέα της διαφήμισης είναι πολύ συχνή η χρήση της μηχανικής μάθησης. Εταιρίες χρησιμοποιούν τα δεδομένα αναζήτησης ως δεδομένα εισόδου, με αποτέλεσμα να μπορούν να προβλέψουν με ευκολία τις προτιμήσεις των καταναλωτών σχετικά με προϊόντα ή υπηρεσίες.

Υγεία: Στον τομέα της υγείας, αισθητήρες μπορούν να μαζέψουν δεδομένα, όπως καρδιακούς παλμούς και πίεση αίματος από τους ασθενείς και στην συνέχεια μοντέλα μηχανικής μάθησης μπορούν να κάνουν προβλέψεις ως προς την κατάσταση του ασθενή. Επιπρόσθετα μπορούν να γίνουν προβλέψεις σχετικά με φαρμακευτικές αγωγές, που πρέπει να ακολουθήσουν οι ασθενείς σύμφωνα με τις παθήσεις τους.

Τεχνολογία: Πολλές λειτουργίες του κινητού τηλεφώνου μας χρησιμοποιούν μηχανική μάθηση. Παραδείγματα αποτελούν οι χάρτες που χρησιμοποιούμε οι οποίοι μας υποδεικνύουν ποια είναι η πιο γρήγορη διαδρομή κάθε φορά σύμφωνα με την κίνηση και οι ηλεκτρονικοί βοηθοί (Google, Alexa, Siri) που μας παρέχουν απαντήσεις σύμφωνα με τις φωνητικές μας αναζητήσεις.

2.5 Περιορισμοί μηχανικής μάθησης

Ο ανθρώπινος νους αφομοιώνει καθημερινά μεγάλο όγκο δεδομένων. Μέσα από αυτά τα δεδομένα που δεχόμαστε καθημερινά μαθαίνουμε συνεχώς καινούρια πράγματα. Εάν οι πληροφορίες που λαμβάνουμε είναι λανθασμένες, τότε και οι ενέργειές μας σχετικά με αυτές τις πληροφορίες θα είναι λανθασμένες. Έτσι ακριβώς λειτουργεί και ένα μοντέλο μηχανικής μάθησης. Τα δεδομένα, που το τροφοδοτούμε, θα πρέπει να ανταποκρίνονται στις προβλέψεις που περιμένουμε να πάρουμε. Εάν, για παράδειγμα, θέλουμε να εκπαιδεύσουμε ένα μοντέλο για ένα όχημα που πηγαίνει μόνο ευθεία και το τροφοδοτήσουμε με δεδομένα στα οποία το όχημα εκτελεί αντίθετη πορεία, δεν θα πάρουμε ποτέ το ζητούμενο αποτέλεσμα.

Όπως αναφέρθηκε σε προηγούμενη ενότητα, τα μοντέλα μηχανικής μάθησης και συγκεκριμένα στην κατηγορία της επιβλεπόμενης μάθησης χρειάζονται δεδομένα, ώστε να «μάθουν» από αυτά και να παράγουν την επιθυμητή έξοδο-πρόβλεψη. Ο αριθμός των δεδομένων αυτών θα πρέπει να είναι αρκετά μεγάλος, ώστε το μοντέλο να είναι σε θέση να γενικεύσει καλύτερα ως προς τις προβλέψεις του και να μας δώσει ικανοποιητικά αποτελέσματα. Ωστόσο πολλές φορές η συλλογή μεγάλου όγκου δεδομένων δεν είναι δυνατή και το μοντέλο δεν είναι σε θέση να δώσει αξιόπιστες προβλέψεις.

Επιπρόσθετα το κομμάτι της εκπαίδευσης είναι ο λόγος που χρειαζόμαστε την αυξημένη υπολογιστική ισχύ, που αναφέρθηκε προηγουμένως. Η εκπαίδευση ενός μοντέλου σε ένα μεγάλο αριθμό δεδομένων μπορεί να διαρκέσει από μερικές μέρες μέχρι και μερικές εβδομάδες, προκειμένου να μπορεί να δώσει τα ζητούμενα αποτέλεσμα. Πολλές φορές μάλιστα μια εκπαίδευση, που διαρκεί μέρες, μπορεί να μην έχει την επιθυμητή απόδοση. Σε αυτήν την περίπτωση θα πρέπει να αλλαχθούν παράμετροι και να ξεκινήσει η διαδικασία από την αρχή. Συνεπώς, υπάρχει μεγάλη ανάγκη για υπολογιστική ισχύ ώστε να μειωθεί ο χρόνος εκπαίδευσης. Έτσι το μεγάλο κόστος που δημιουργεί η αγορά υλικού θέτει περιορισμούς στο συγκεκριμένο επιστημονικό πεδίο.

2.6 Σκοπός της εργασίας

Στην εργασία αυτή θα γίνει ανάλυση της βάσης γνώσης για την αυτόνομη πλοήγηση, την μηχανική μάθηση. Ως προς την απεικόνιση της εργασίας θα γίνει χρήση περιβάλλοντος προσομοίωσης το οποίο περιέχει οχήματα που εκτελούν συνεχή και συγκεκριμένη πορεία καθώς και το όχημα το οποίο θα ελέγχει το μοντέλο που θα δημιουργήσουμε. Επιπρόσθετα θα γίνει ανάλυση συλλογής και επεξεργασίας δεδομένων που χρειαζόμαστε, δημιουργία και εκπαίδευση μοντέλου μηχανικής μάθησης, αναφορά δικτύων και εργαλείων που χρησιμοποιήθηκαν και φυσικά παρουσίαση αποτελεσμάτων της προσομοίωσης και συμπεριφοράς του οχήματος. Παράλληλα θα γίνει ανάλυση των αρχιτεκτονικών αλγορίθμων μηχανικής μάθησης, καθώς και σύγκριση μεταξύ τους σύμφωνα με τα αποτελέσματα.

3. Αρχιτεκτονική αλγορίθμων μηχανικής μάθησης

3.1 Εισαγωγή

Οι τεχνικές μηχανικής μάθησης εφαρμόζονται μέσα από μια σειρά αλγορίθμων, οι οποίες εξετάζονται σε αυτό το κεφάλαιο. Η βασική αρχιτεκτονική, που ακολουθείται για την δημιουργία μοντέλων μηχανικής μάθησης, είναι αυτή των νευρωνικών δικτύων. Θα γίνει ανάλυση διαφορετικών αρχιτεκτονικών νευρωνικών δικτύων καθώς και του είδους που χρησιμοποιούμε κάθε φορά ανάλογα με τις ανάγκες του προβλήματος.

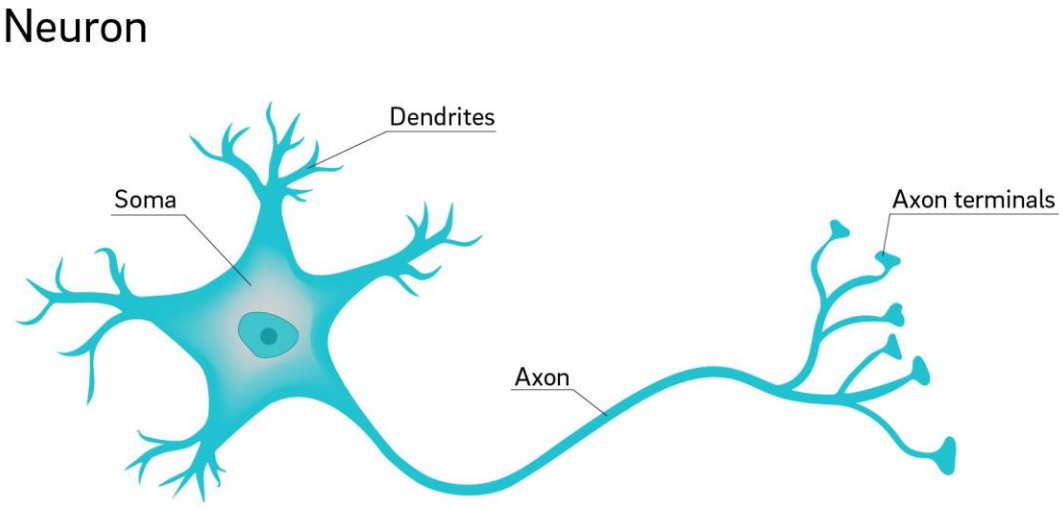
3.2 Νευρωνικά Δίκτυα - Γενικά

Νευρώνας ονομάζεται το δομικό μέρος που μεταφέρει πληροφορίες σε ένα νευρωνικό δίκτυο. Ένα τέτοιο δίκτυο περιλαμβάνει μια ή περισσότερες συνδέσεις μεταξύ νευρώνων. Τα νευρωνικά δίκτυα διακρίνονται σε φυσικά ή βιολογικά νευρωνικά δίκτυα (Biological Neural Networks) και σε τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks). Στην πρώτη περίπτωση το δίκτυο αυτό αποτελεί τον νευρικό ιστό κάθε έμβιου οργανισμού και μέσα σε αυτό μεταφέρονται πληροφορίες μέσω ηλεκτρικών σημάτων. Στην περίπτωση των τεχνητών δικτύων γίνεται μίμηση ως προς τη δομή και τη λειτουργία των βιολογικών δικτύων και το αποτέλεσμα είναι ένα μαθηματικό μοντέλο, το οποίο αποτελεί τον σκελετό για τους αλγόριθμους μηχανικής μάθησης.

3.3 Φυσικά Νευρωνικά Δίκτυα

Υπάρχουν διάφοροι τύποι φυσικών νευρώνων οι οποίοι διαφέρουν ως προς την μορφή τους. Ένας νευρώνας αποτελείται από τους δενδρίτες (dendrites), οι οποίοι είναι τα κανάλια εισόδου του, το κυρίως σώμα (cell body), το οποίο περιλαμβάνει τον πυρήνα του κυττάρου και τον νευράξονα (axon) ή νευρίτη, ο οποίος είναι ο δίαυλος επικοινωνίας με άλλα κύτταρα-

νευρώνες. Η άκρη του νευράξονα καταλήγει στους δενδρίτες διαφορετικού νευρικού κυττάρου όπου η σύνδεση μεταξύ τους γίνεται μέσω συνάψεων. Με αυτή την σύνδεση είναι δυνατή η επικοινωνία μεταξύ νευρώνων. Η ίδια η πληροφορία μεταφέρεται με ηλεκτρικά σήματα.

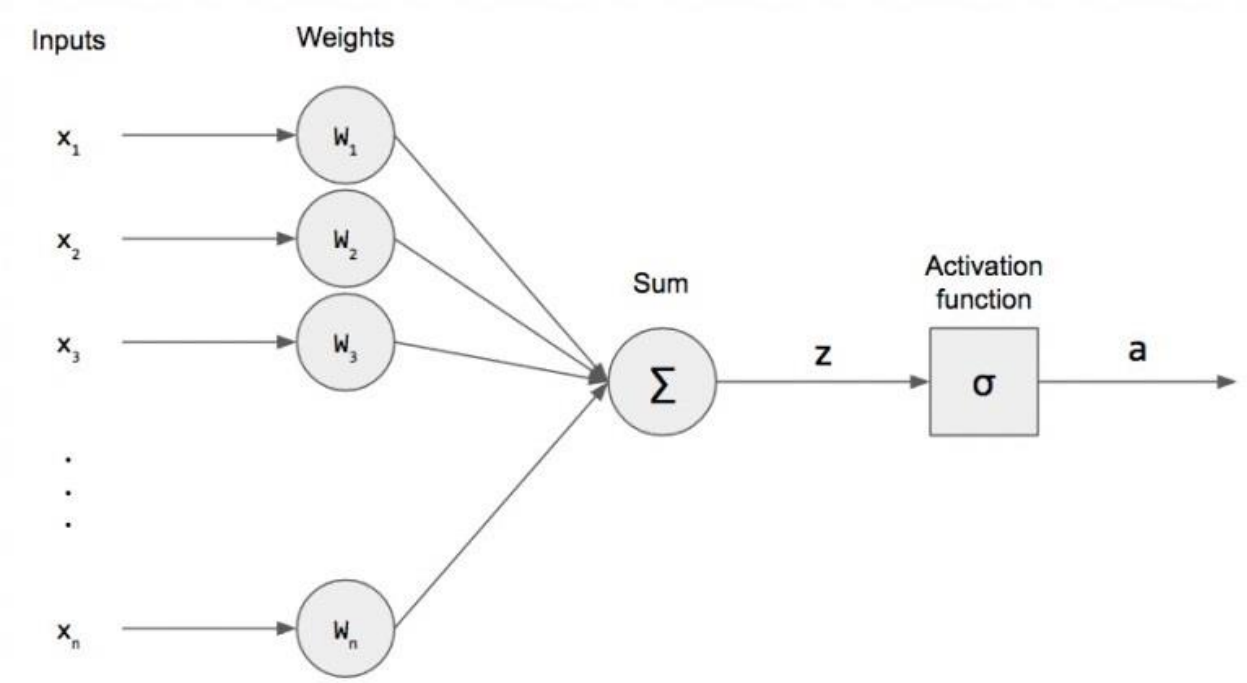


Εικόνα 3.1 – Φυσικός Νευρώνας

3.4 Τεχνητά Νευρωνικά Δίκτυα

3.4.1 Δομή Νευρώνα (Perceptron)

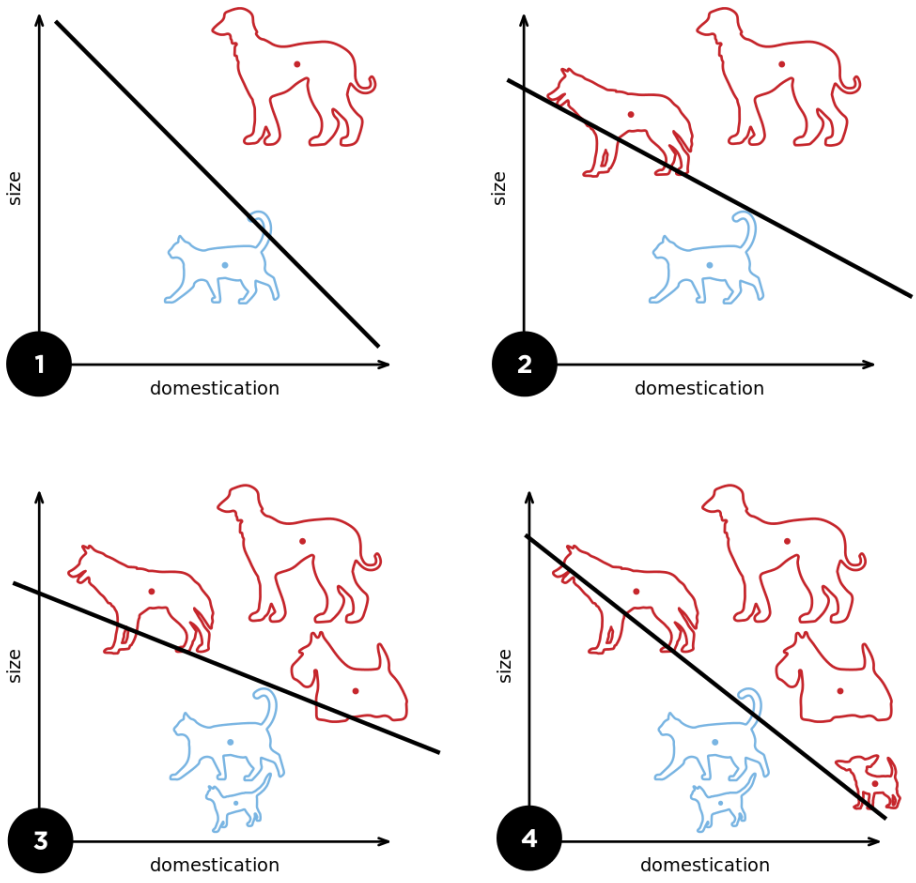
Ένας **τεχνητός νευρώνας** αποτελείται από τις εισόδους (Inputs - X_1 - X_n) και μια έξοδο (Output) πληροφορίας, τα βάρη (Weights – W_1 - W_n) σε κάθε είσοδο, την τιμή κατωφλίου (threshold value), την αθροιστική συνάρτηση (Summation Function) και την συνάρτηση ενεργοποίησης (Activation Function) πριν την έξοδο. Στην εικόνα 3.1 απεικονίζεται ένας τεχνητός νευρώνας ή αλλιώς **στοιχειώδης Perceptron (Basic Perceptron)**.



Εικόνα 3.2 – Τεχνητός Νευρώνας (Perceptron)

3.4.2 Γραμμική Παλινδρόμηση

Για να μιλήσουμε για την λειτουργία του Perceptron, θα πρέπει πρώτα να μιλήσουμε για συναρτήσεις ενεργοποίησης. Μια από τις πιο βασικές είναι η γραμμική παλινδρόμηση (Linear Regression). Το γνώρισμα της γραμμικής παλινδρόμησης είναι η αντιστοίχηση μιας τιμής εξόδου y με βάση κάποια είσοδο x. Ο πιο απλός τύπος συνάρτησης για μια τέτοια αντιστοιχία είναι η ευθεία **y = ax + b.** Αν υπάρχει ένας μεγάλος αριθμός δεδομένων **x** και **y** (π.χ χαρακτηρισμός ζώων ως κατοικίδια σύμφωνα με το μέγεθος τους) μπορούμε να κατασκευάσουμε μια ευθεία που θα κατηγοριοποιεί τα δεδομένα που θα δώσουμε.



Εικόνα 3.3 – Στάδια κατηγοριοποίησης σύμφωνα με τον όγκο δεδομένων

Παρατηρούμε ότι σε κάθε στάδιο η κλίση της ευθείας αλλάζει. Αυτό συμβαίνει, γιατί γίνεται προσθήκη νέων δεδομένων τα οποία αλλάζουν την κλίση της. Η συνάρτηση για την κλίση του μοντέλου γραμμικής παλινδρόμησης δίνεται ως εξής:

$$\widehat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Εικόνα 3.4 – Κλίση γραμμικής παλινδρόμησης

3.4.3 Βηματική Συνάρτηση

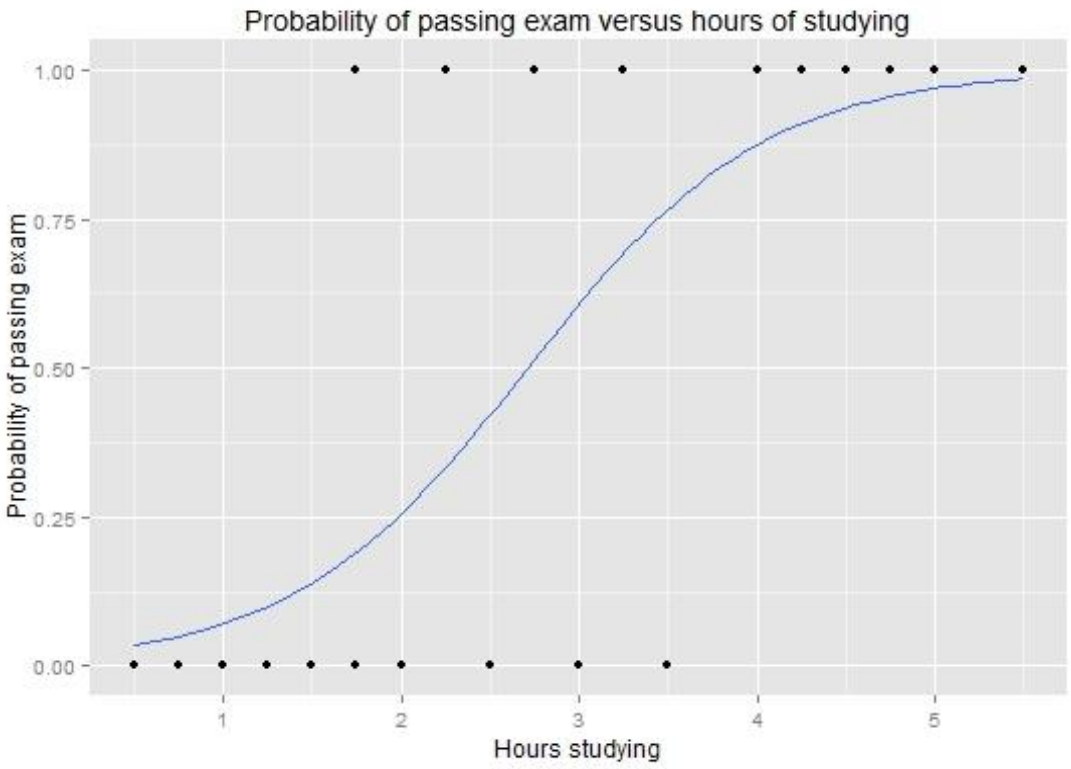
Μια ακόμα συνάρτηση είναι η Βηματική (Hard Limiter) και χρησιμοποιείται, όταν ως έξοδος μπορεί να εξαχθεί μια από τις δύο περιπτώσεις . Στην περίπτωση αυτή η συνάρτηση μας είναι κλαδική (Εικόνα 3.4) και έχει δύο περιπτώσεις. Αν η τιμή που προκύπτει είναι μεγαλύτερη του μηδέν τότε το αποτέλεσμα της είναι ένα. Σε διαφορετική περίπτωση η συνάρτηση θα δώσει την τιμή μηδέν.

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

Εικόνα 3.5 – Συνάρτηση Κατωφλίου

3.4.4 Λογιστική Παλινδρόμηση

Η πιο συνηθισμένη συνάρτηση, η οποία έχει και την μεγαλύτερη χρησιμότητα στην επιβλεπόμενη μάθηση είναι η λογιστική παλινδρόμηση (Logistic Regression) . Τις περισσότερες φορές το πρόβλημα απαιτεί την κατηγοριοποίηση αρκετών ομάδων δεδομένων. Με αυτόν τον τρόπο τα μοντέλα μηχανικής μάθησης παίρνουν αποφάσεις σύμφωνα με προβλέψεις, που περιγράφονται με ποσοστά. Στην συγκεκριμένη κατηγορία χρησιμοποιείται η σιγμοειδής συνάρτηση (Sigmoid) και μας δίνει μια ποσοστιαία τιμή ως έξοδο. Στο παρακάτω παράδειγμα (Εικόνα 3.5) η καμπύλη μας δείχνει την πιθανότητα επιτυχίας σε μια εξέταση σχετικά με τις ώρες μελέτης.



Εικόνα 3.6 – Σιγμοειδής καμπύλη

Σύμφωνα με την παραπάνω εικόνα, ένας μαθητής, που έχει μελετήσει δύο ώρες, έχει πιθανότητες επιτυχίας 25 % στην εξέταση, ενώ για έναν μαθητή ο οποίος έχει μελετήσει πέντε ώρες, οι πιθανότητες επιτυχίας του πλησιάζουν το 100 %.

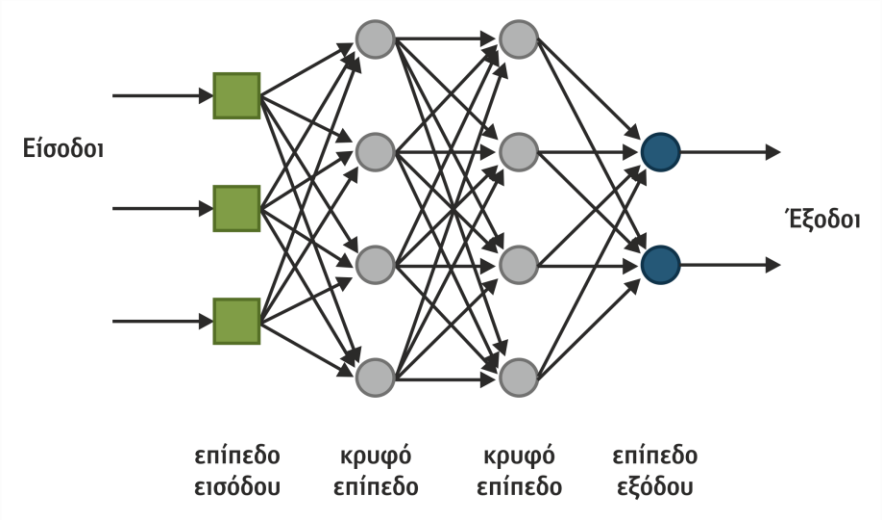
Η σιγμοειδής συνάρτηση δίνεται ως εξής:

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

Εικόνα 3.7 – Σιγμοειδής συνάρτηση

3.4.5 Πολυεπίπεδα ΤΝΔ

Μέχρι τώρα περιγράψαμε την δομή ενός βασικού Perceptron. Παρ' όλα αυτά, τις περισσότερες φορές το δίκτυο που δουλεύουμε θα έχει παραπάνω από έναν νευρώνα καθώς και παραπάνω από ένα επίπεδα (ή στρώματα) νευρώνων. Τα ενδιάμεσα στρώματα εκτός των επιπέδων εισόδου και εξόδου ονομάζονται **κρυφά επίπεδα (hidden layers)**. Ένα δίκτυο με τα παραπάνω χαρακτηριστικά ονομάζεται **πολυεπίπεδο Τεχνητό Νευρωνικό Δίκτυο (Multilayer Neural Network ή Multilayer Perceptron)**.



Εικόνα 3.8 – Πολυεπίπεδο ΤΝΔ

Κάθε επίπεδο περιέχει ένα σύνολο νευρώνων ή κόμβων (nodes), που είναι συνδεδεμένοι με προηγούμενο ή και με επόμενο στρώμα κόμβων. Η σύνδεση ενός στρώματος με το επόμενο γίνεται με την σύνδεση της εξόδου του νευρώνα στο Ν στρώμα με την είσοδο του νευρώνα στο Ν+1 στρώμα (Εικόνα 3.7). Ο κάθε κόμβος αθροίζει την τιμή κατωφλίου και τα γινόμενα κάθε εισόδου με το αντίστοιχο βάρος του και εξάγει την έξοδο σύμφωνα με την συνάρτηση ενεργοποίησης. Η πληροφορία μεταφέρεται στα επόμενα στρώματα με τον ίδιο τρόπο μέχρι να φτάσει στην τελική έξοδο του δικτύου. Αν πάρουμε ως παράδειγμα την εικόνα 3.1 μπορούμε να δούμε ακριβώς τι συμβαίνει σε κάθε νευρώνα. Τα δεδομένα που θα τροφοδοτήσουμε εισάγονται στις εισόδους (X_1 - X_n), στις οποίες θα γίνει πολλαπλασιασμός μεταξύ αυτών και των βαρών (W_1 - W_n) και στην συνέχεια άθροισμα μεταξύ των γινομένων και την τιμή κατωφλίου. Η τιμή κατωφλίου προσδιορίζει την τιμή ενεργοποίησης για την έξοδο.

$$F = b + \sum_{i=1}^{n} x_i w_i$$

Εικόνα 3.9 – Αθροιστική Συνάρτηση (Summation Function)

Για να προκύψει το τελικό αποτέλεσμα (Output), χρησιμοποιούμε μια από τις συναρτήσεις που εξηγήθηκαν στα προηγούμενα κεφάλαια. Επιλέγουμε την κατάλληλη συνάρτηση ενεργοποίησης, σύμφωνα με τις ανάγκες της κατηγοριοποίησης που χρειάζεται το πρόβλημα μας και ως είσοδος δίνεται το αποτέλεσμα της πρόσθεσης που έχει προκύψει από τις εισόδους , τα βάρη και την τιμή κατωφλίου. Αν η συνάρτηση ενεργοποίησης είναι η f, τότε θα έχει την εξής είσοδο:

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

Εικόνα 3.10 – Είσοδος συνάρτησης ενεργοποίησης

Τις περισσότερες φορές ως συνάρτηση ενεργοποίησης χρησιμοποιείται η γραμμική ή η λογιστική ανάλογα με το πρόβλημα. Για προβλήματα ταξινόμησης επιλέγεται η λογιστική, ενώ για προβλήματα συναρτησιακής προσέγγισης η γραμμική.

Ο στόχος ενός νευρωνικού δικτύου είναι να εκπαιδεύσει το μοντέλο με τα δεδομένα που του έχουν δοθεί με όσο το δυνατόν μεγαλύτερη ακρίβεια και συνεπώς όσο το δυνατόν μικρότερο σφάλμα. Έτσι μετά από την συνάρτηση ενεργοποίησης στο επίπεδο εξόδου υπάρχει μια συνάρτηση σφάλματος. Αρκετά βασικές συναρτήσεις σφάλματος είναι η **διασταυρωμένη εντροπία(Categorical Cross Entropy)** και η **Διαφορά Τετραγώνων (mean square error)**. Το αποτέλεσμα της συνάρτησης αυτής χρησιμοποιείται για να γίνει διόρθωση στα βάρη του δικτύου μέσω της **οπισθοδιάδοσης σφάλματος (back-propagation)**. Ο τρόπος λειτουργίας του back propagation αλγόριθμου είναι η μετάδοση σφάλματος στους

προηγούμενους κόμβους και ο επαναπροσδιορισμός στις τιμές των βαρών τους. Με τον επαναπροσδιορισμό των βαρών γίνεται προσπάθεια για μεγαλύτερο ποσοστό ακρίβειας.

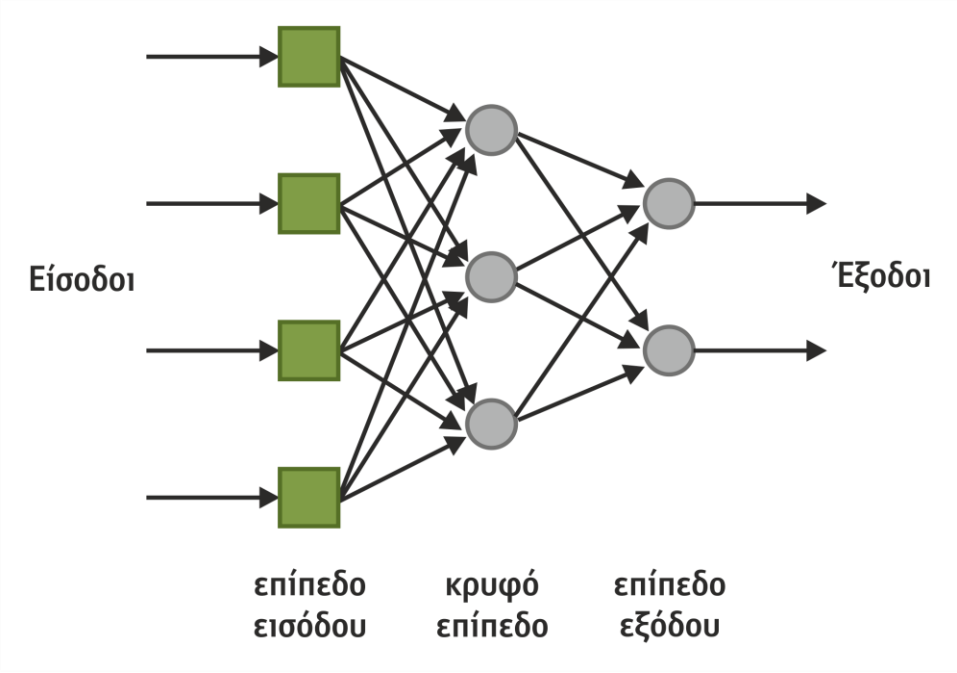
Τέλος το στάδιο της εκπαίδευσης επαναλαμβάνεται μέχρι να ξεπεραστεί ένα συγκεκριμένο όριο ακρίβειας ή μέχρι να μας ικανοποιούν τα αποτελέσματα. Μπορούμε επίσης να το εκπαιδεύσουμε για συγκεκριμένο αριθμό επαναλήψεων (epochs).

Για την κατασκευή ενός νευρωνικού δικτύου, δεν είμαστε σε θέση να γνωρίζουμε πόσους κόμβους ή πόσα κρυμμένα επίπεδα χρειάζεται το δίκτυο που κατασκευάζουμε για οποιοδήποτε πρόβλημα, καθώς δεν υπάρχει κάποια θεωρία πίσω από αυτό. Οι λύσεις που υπάρχουν περιορίζονται στην δοκιμή διαφορετικών δικτύων μέχρι να βρεθεί αυτό που δίνει τα καλύτερα αποτελέσματα. Τέλος, ένας ακόμα τρόπος είναι η χρήση έτοιμων δοκιμασμένων δικτύων.

3.4.6 Μεταφορά Πληροφορίας

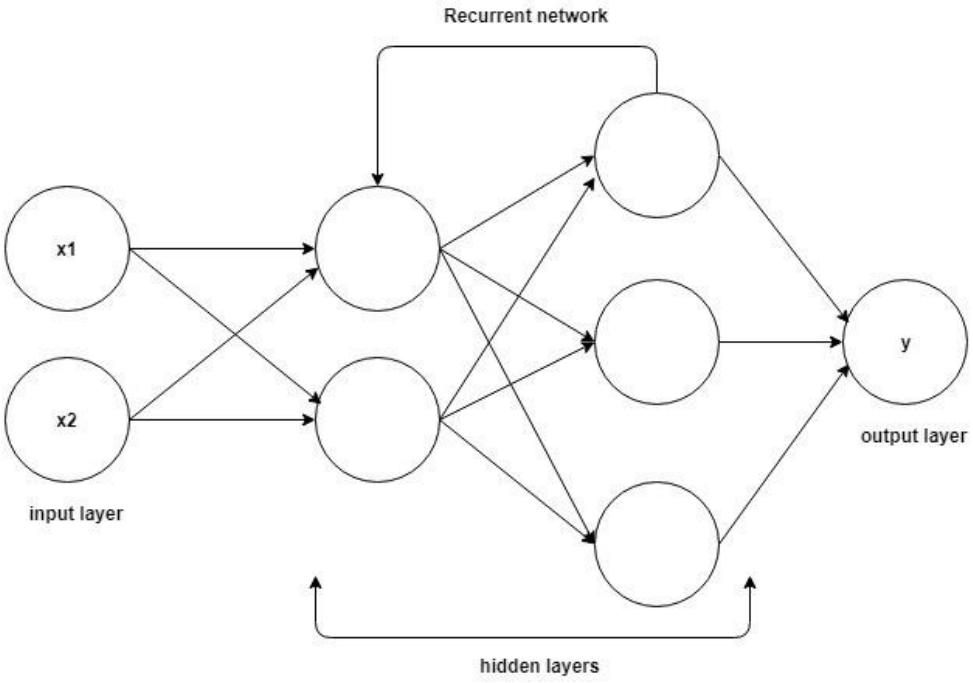
Όσον αφορά τον τρόπο που μεταφέρεται η πληροφορία μεταξύ τον κόμβων σε ένα δίκτυο υπάρχουν δύο κατηγορίες:

Πρόσθιας Τροφοδότησης ΤΝΔ (Feed forward): Οι κόμβοι ενός δικτύου πρόσθιας τροφοδότησης διακρίνονται σε επίπεδα (στρώματα) και οι κόμβοι του ενός επιπέδου τροφοδοτούν τους κόμβους του επόμενου επιπέδου έως το τελικό στρώμα εξόδου του δικτύου. Τα νευρωνικά δίκτυα, που έχουμε δει μέχρι τώρα, λειτουργούν με αυτόν τον τρόπο. Αυτού του τύπου δίκτυα θα εξετάσουμε στην συγκεκριμένη εργασία.



Εικόνα 3.11 – Δίκτυο πρόσθιας τροφοδότησης (feed forward neural network)

Ανατροφοδοτούμενα ΤΝΔ (recurrent neural networks) : Ο τρόπος λειτουργίας των ανατροφοδοτούμενων νευρωνικών δικτύων περιλαμβάνει την διάδοση πληροφορίας σε κόμβους προηγούμενων επιπέδων. Έτσι με αυτό τον τρόπο οι έξοδοι των κόμβων σε ένα επίπεδο θα είναι οι είσοδοι των κόμβων σε προηγούμενο επίπεδο σχηματίζοντας κυκλική πορεία μεταξύ των κόμβων. Συνεπώς, αφού οι είσοδοι των κόμβων εξαρτούνται από την έξοδο των κόμβων επόμενου επιπέδου τα δίκτυα αυτά μπορούμε να πούμε ότι περιέχουν «μνήμη». Τα **Long Short-Term Memory (LSTM)** είναι μια κατηγορία ανατροφοδοτούμενων νευρωνικών δικτύων η οποία χρησιμοποιείται αρκετά στη δημιουργία μοντέλων, που πραγματοποιούν προβλέψεις σχετικά με την φυσική γλώσσα. Ένα παράδειγμα αποτελεί η πρόβλεψη επόμενης λέξης στα κινητά τηλέφωνα κατά τη δημιουργία μηνυμάτων.



Εικόνα 3.12 – Ανατροφοδοτούμενο ΤΝΔ (recurrent neural network)

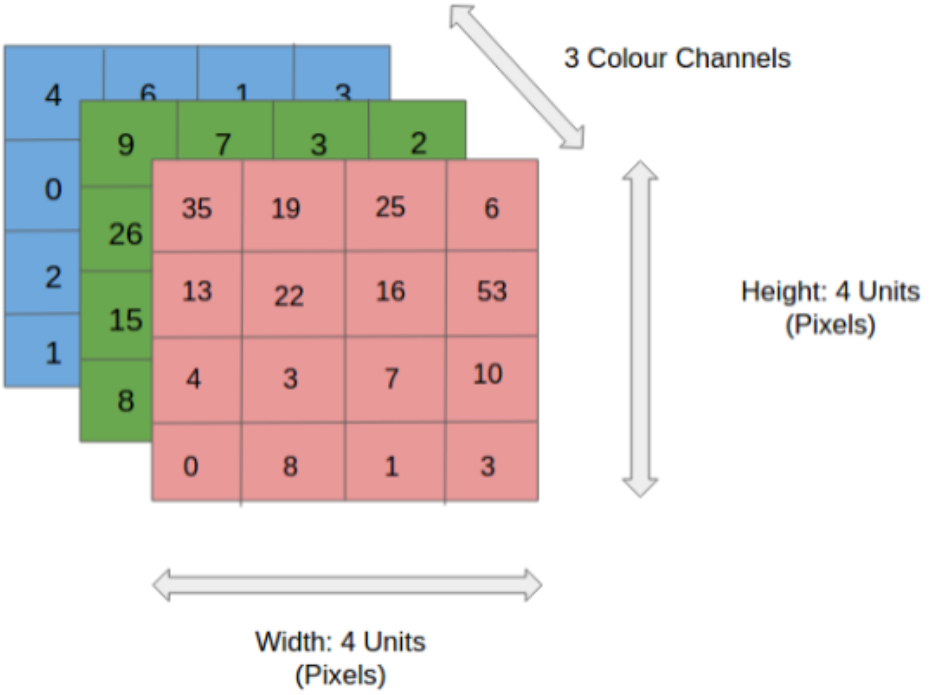
3.5 Συνελικτικά Νευρωνικά Δίκτυα

3.5.1 Γενικά

Τα τεχνητά νευρωνικά δίκτυα στα οποία έγινε αναφορά στα προηγούμενα κεφάλαια, σαφώς δεν είναι κάτι καινούριο. Υπάρχουν ήδη αρκετά χρόνια και η χρήση τους δεν είναι τόσο ευρεία σε πραγματικές εφαρμογές. Παρ' όλα αυτά έθεσαν τη βάση για έναν άλλο τύπο δικτύων που ονομάζουμε **Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNN)** και κατά συνέπεια στην ανάπτυξη της **βαθιάς μάθησης (Deep Learning).** Η βαθιά μάθηση πήρε το όνομα της από τα πολλαπλά κρυφά επίπεδα που μπορεί να έχει ένα τεχνητό νευρωνικό δίκτυο (βάθος). Έτσι, σε συνδυασμό με την αυξημένη υπολογιστική ισχύ που έχουμε στην διάθεση μας και την αρχιτεκτονική ενός CNN, μπορούν να επιτευχθούν εξαιρετικές επιδόσεις σε σχέση με τα κλασσικά νευρωνικά δίκτυα. Τα CNN αρχικά δημιουργήθηκαν για αναγνώριση κειμένου, παρ' όλα αυτά εκεί που διαπρέπουν είναι η κατηγοριοποίηση εικόνας.

3.5.2 Τρόπος Λειτουργίας

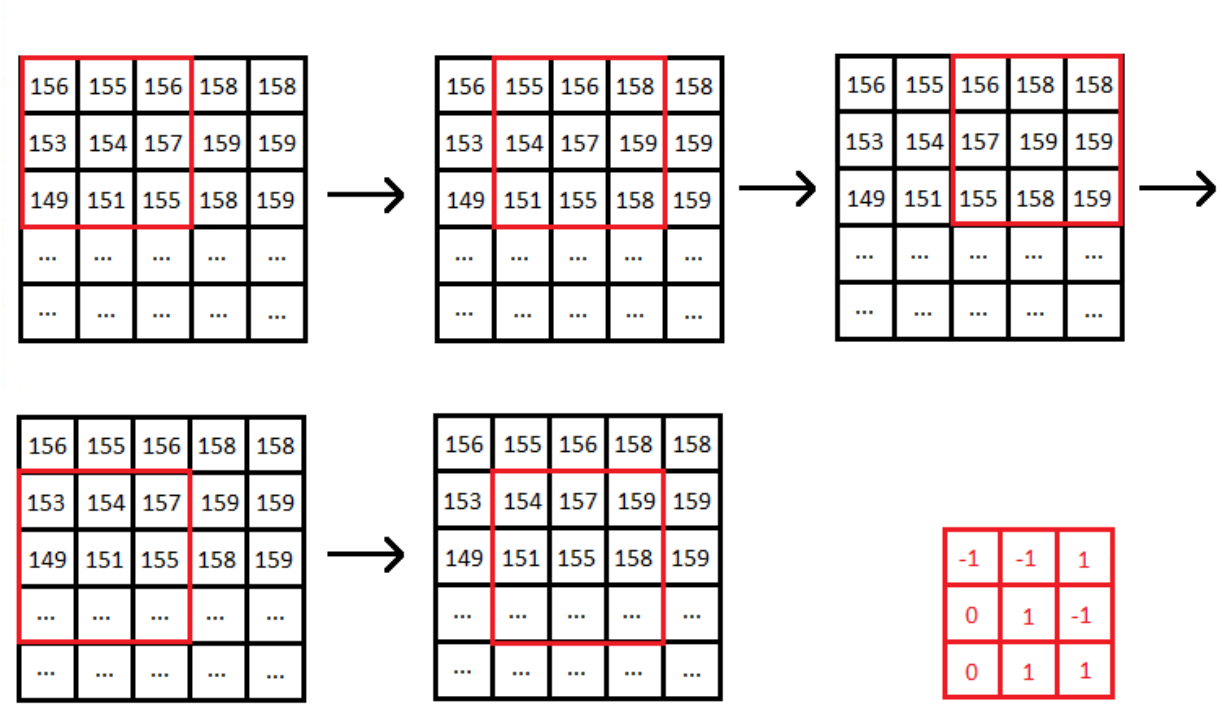
Για να μιλήσουμε για τον τρόπο που δουλεύει ένα CNN, θα πρέπει πρώτα να μιλήσουμε για την δομή της εικόνας. Μια εικόνα χωρίζεται σε τρία κανάλια χρώματος **κόκκινο**, **πράσινο** και **μπλε** (**Red Green Blue - RGB**). Το επίπεδο εισόδου ενός CNN είναι οι 2 διαστάσεις της εικόνας και ο αριθμός των καναλιών της. Τα κρυφά επίπεδα αποτελούνται από έναν συνδυασμό τριών επιπέδων. Αυτά είναι το **επίπεδο συνέλιξης (convolutional layer), το επίπεδο συγκέντρωσης (pooling layer)** και το **πλήρως συνδεδεμένο επίπεδο (fully connected layer).**



Εικόνα 3.13– Κανάλια Εικόνας (4x4x3)

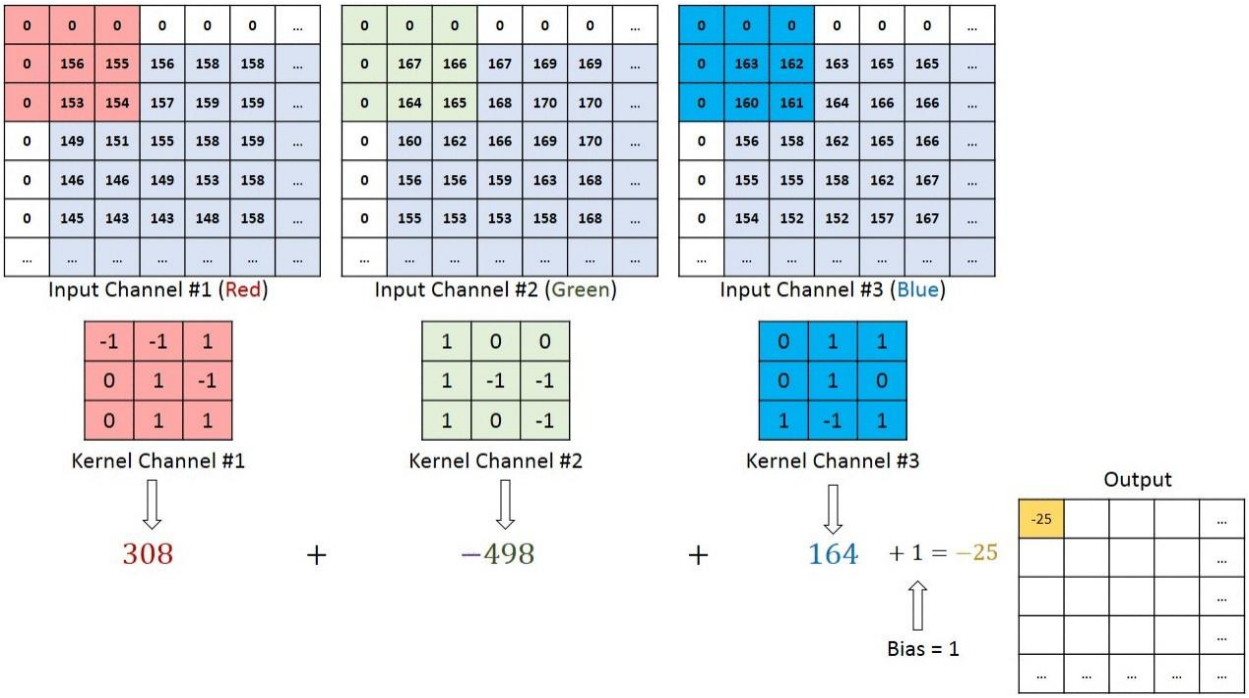
3.5.2.1 Επίπεδο Συνέλιξης

Στο στάδιο της συνέλιξης χρησιμοποιείται ένα **φίλτρο (Kernel/Filter)** που λαμβάνει την εικόνα σταδιακά. Το φίλτρο αυτό είναι ένας πίνακας με αριθμητικές τιμές, που έχουν οριστεί ανάλογα με τον τρόπο που θέλουμε να ληφθεί η εικόνα. Αυτός ο πίνακας θα κάνει συγκεκριμένο αριθμό βημάτων στο κάθε κανάλι της δοσμένης εικόνας (με αριθμητική αναπαράσταση) και κάθε φορά θα το χωρίζει σύμφωνα με το μέγεθος του, μέχρι να λάβει όλη την εικόνα. Με την έννοια λήψης εικόνας εννοείται ότι, σε κάθε πέρασμα (βήμα) του φίλτρου στην εικόνα, γίνεται ένας πολλαπλασιασμός πινάκων μεταξύ του φίλτρου και των αριθμητικών τιμών του καναλιού στο τρέχον βήμα. Το τελικό αποτέλεσμα του πολλαπλασιασμού και η πρόσθεση του με την ορισμένη κλίση (bias) μας δίνει τον πίνακα συνέλιξης. Δίνεται ως παράδειγμα η παρακάτω εικόνα όπου ως φίλτρο δίνεται ο κόκκινος πίνακας (3x3x1) και ως κανάλι της εικόνας ο μαύρος πίνακας (5x5x1):



Εικόνα 3.14 – Συνέλιξη σε κανάλι εικόνας

Το φίλτρο κινείται προς τα δεξιά, μέχρι να φτάσει το τέλος των στηλών. Στην συνέχεια κάνει ένα βήμα κάτω και βρίσκεται πάλι στην αρχή (αριστερά) . Το **βήμα (stride),** που κάνει κάθε φορά το φίλτρο, παρατηρούμε ότι είναι μια στήλη. Αυτό μπορεί να ρυθμιστεί διαφορετικά κάθε φορά σύμφωνα με το δείγμα που θα τροφοδοτήσει το δίκτυο. Αυτή η διαδικασία γίνεται για όσα κανάλια οριστούν στην είσοδο του δικτύου. Εφόσον στο συγκεκριμένο παράδειγμα υπάρχουν τρία κανάλια το αποτέλεσμα θα είναι ένας πίνακας που κάθε κελί θα προκύπτει από την πρόσθεση των τριών τιμών οι οποίες έχουν προκύψει από τον πολλαπλασιασμό πινάκων το φίλτρου με το κάθε κανάλι. Αυτός ο πίνακας ονομάζεται **feature-map**. Ο απεικόνιση της διαδικασίας για την κατασκευή του τελικού πίνακα δίνεται παρακάτω:

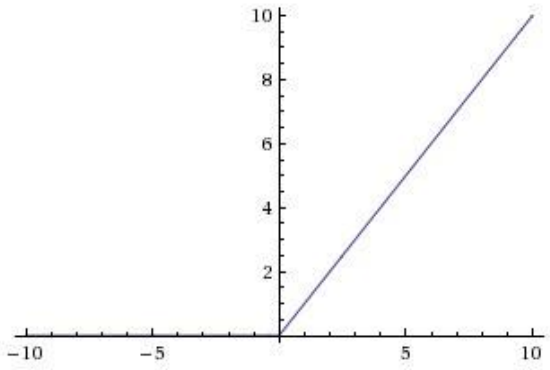


Εικόνα 3.15 – Διαδικασία Συνέλιξης Τριών Καναλιών

Τέλος για την έξοδο χρησιμοποιούμε την συνάρτηση ενεργοποίησης **ReLU (Rectified Linear),** η οποία παρέχει καλύτερα αποτελέσματα στα συνελικτικά δίκτυα από την Σιγμοειδή που είδαμε στα προηγούμενα κεφάλαια. Ο τύπος της συνάρτησης είναι:

$$f(x) = max(0, x)$$

Εικόνα 3.16 – Συνάρτηση ReLU (Rectified Linear)



Εικόνα 3.17 – Αναπαράσταση ReLU (Rectified Linear)

3.5.2.2 Επίπεδο Συγκέντρωσης

Παρόμοιο με το επίπεδο συνέλιξης, το επίπεδο συγκέντρωσης προσπαθεί να απλοποιήσει την πληροφορία από το συνελικτικό επίπεδο. Η διαδικασία είναι πάλι η ίδια, δηλαδή υπάρχει πάλι ένα φίλτρο που «διαβάζει» σταδιακά όλο τον πίνακα, που έχει προκύψει από το επίπεδο συνέλιξης (feature map). Οι τύποι συγκέντρωσης που υπάρχουν είναι δύο. Η **μέγιστη συγκέντρωση (max-pooling)** επιστρέφει την μέγιστη τιμή που θα συναντήσει το φίλτρο στο feature map που έχει προκύψει, και η **μέση συγκέντρωση (average pooling)** επιστρέφει την μέση τιμή του feature map.

3.5.2.3 Πλήρως Συνδεδεμένο Επίπεδο

Το τελευταίο κρυφό στρώμα στα CNN θα είναι πάντα ένα πλήρως συνδεδεμένο επίπεδο. Αυτός ο τύπος δικτύου έχει την κλασσική αρχιτεκτονική πολυεπίπεδου νευρωνικού δικτύου, που αναφέρθηκε σε προηγούμενα κεφάλαια. Οι κόμβοι σε αυτό το σημείο περιέχουν συναρτήσεις ενεργοποίησης ReLU και το δίκτυο λειτουργεί σύμφωνα με αυτά που αναφέρθηκαν στο κεφάλαιο 3.4.

3.5.2.4 Επίπεδο Εξόδου

Το επίπεδο εξόδου είναι το επίπεδο που μετατρέπει τις τιμές εξόδου σε ποσοστά, ώστε να χρησιμοποιηθούν για την αξιολόγηση του μοντέλου. Ο αριθμός νευρώνων εξόδου είναι ίδιος με τις κατηγορίες που θέλουμε να πάρουμε. Το επίπεδο αυτό ονομάζεται softmax και χρησιμοποιείται η τυποποιημένη εκθετική συνάρτηση στην έξοδο του δικτύου. Η συνάρτηση δίνεται ως εξής:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Εικόνα 3.18 – Συνάρτηση Softmax

3.5.3 Υπάρχοντα CNNs

Όπως αναφέρθηκε και πιο νωρίς, είναι αρκετά δύσκολο κάποιος να κατασκευάσει ένα δίκτυο που θα παράγει μοντέλα με ικανοποιητικά αποτελέσματα, καθώς δεν υπάρχουν συγκεκριμένες οδηγίες για κάτι τέτοιο. Παρ' όλα αυτά εταιρίες, όπως η Google και η Microsoft έχουν δημιουργήσει δίκτυα, τα οποία έχουν αρκετά ικανοποιητικές επιδόσεις στο κομμάτι της κατηγοριοποίησης εικόνας (image classification). Κάποια από αυτά είναι το Inception Xception της Google, το ResNet της Microsoft, το AlexNet του Alex Krizhevsky και το VGGNet. Τα μοντέλα, που θα εξεταστούν σε αυτήν την εργασία, είναι το Inception v3 και το Xception.

4. Αυτόνομα Οχήματα

4.1 Γενικά

Ένας τομέας, όπου έχει αρχίσει και εφαρμόζεται η μηχανική μάθηση είναι τα **αυτόνομα οχήματα (autonomous vehicles).** Τα τελευταία χρόνια έχει γίνει μεγάλη πρόοδος ως προς τον βαθμό της αυτονομίας που μπορεί να επιτευχθεί, καθώς τα αυτόνομα οχήματα γίνονται όλο και πιο ασφαλή με το πέρασμα των χρόνων και οι δυνατότητες τους συνεχώς αυξάνονται. Τα πιο σύγχρονα αυτόνομα αυτοκίνητα έχουν εξαιρετικές επιδόσεις στο δρόμο, καθώς δεν έχουν κανένα πρόβλημα να αναγνωρίζουν άλλα αυτοκίνητα και να κρατάνε τις ανάλογες αποστάσεις, να μένουν ή να αλλάζουν λωρίδα όταν χρειάζεται, να προσέχουν τους πεζούς και να σταθμεύουν χωρίς βοήθεια από τον οδηγό. Τέλος η κατασκευή πλήρως αυτόνομων οχημάτων δεν είναι ένα μακρινό σενάριο, καθώς υπάρχουν ήδη οχήματα που δοκιμάζονται καθημερινά έχοντας αποκλειστικά τον έλεγχο. Η συλλογή μεγαλύτερου όγκου δεδομένων, σε συνδυασμό με την συνεχή βελτίωση αισθητήρων και αλγορίθμων μηχανικής μάθησης κάνουν εφικτή τη χρήση πλήρως αυτόνομων αυτοκινήτων στο προσεχές μέλλον.

4.2 Λειτουργία

Ο τρόπος, με τον οποίο λειτουργούν τα αυτόνομα οχήματα, είναι ο συνδυασμός αισθητήρων και λογισμικού. Ο τύπος των αισθητήρων που χρησιμοποιούνται, περιλαμβάνει GPS (Global Positioning System), ραντάρ, κάμερα και LiDAR (Light Detection And Ranging). Το τελευταίο κάνει εκπομπή φωτός στο περιβάλλον και κάνει μετρήσεις της απόστασης των αντικειμένων στο χώρο, σύμφωνα με τον χρόνο επιστροφής των ακτίνων φωτός. Οι αισθητήρες αυτοί είναι τοποθετημένοι με τέτοιο τρόπο, ώστε το αυτοκίνητο να έχει πλήρες οπτικό πεδίο γύρω του. Τροφοδοτούν ουσιαστικά το λογισμικό με πληροφορίες, όπως την απόσταση από τις γραμμές μιας λωρίδας, την κίνηση πεζών ή άλλων οχημάτων ή οποιοδήποτε άλλο αντικείμενο βρίσκεται στον δρόμο. Το λογισμικό αποτελείται από εκπαιδευμένα μοντέλα μηχανικής μάθησης και αλγορίθμους αναγνώρισης αντικειμένων (object detection algorithms),

τα οποία λαμβάνουν αδιάκοπα πληροφορίες από τους αισθητήρες ως εισόδους και δίνουν ως έξοδο την κίνηση του οχήματος σε πραγματικό χρόνο. Παράλληλα τα δεδομένα, που λαμβάνονται από όλα τα αυτόνομα αυτοκίνητα, αποθηκεύονται στο νέφος (cloud) ώστε να γίνει καλύτερη εκπαίδευση των μοντέλων μηχανικής μάθησης.

4.3 Τύποι Αυτόνομων Οχημάτων

Ο βαθμός αυτονομίας ενός οχήματος διακρίνεται σε έξι επίπεδα ανάλογα με τις δυνατότητες που προσφέρει. Πιο αναλυτικά:

Επίπεδο 0: Τα αυτοκίνητα σε αυτό το επίπεδο δεν προσφέρουν κάποιο είδος αυτονομίας. Η οδήγηση εξαρτάται αποκλειστικά από τον ίδιο τον άνθρωπο. Όλα τα συμβατικά αυτοκίνητα περιλαμβάνονται σε αυτή την κατηγορία.

Επίπεδο 1: Τα οχήματα, που ανήκουν σε αυτή την κατηγορία, μπορούν να συλλέξουν κάποιες πληροφορίες σχετικά με το περιβάλλον που βρίσκονται και να βοηθήσουν τον οδηγό με κάποιες βασικές λειτουργίες, όπως το φρενάρισμα για την διατήρηση της απόστασης από άλλα οχήματα.

Επίπεδο 2: Οι ικανότητες του αυτοκινήτου είναι αρκετά αυξημένες στο επίπεδο δύο καθώς το όχημα είναι σε θέση να αναγνωρίζει τα εμπόδια και τις λωρίδες γύρω, να χειρίζεται το τιμόνι καθώς σε κάποιες περιπτώσεις να σταθμεύει αποκλειστικά χωρίς την βοήθεια του οδηγού. Τα αυτοκίνητα επιπέδου δύο, εάν εντοπίσουν κάτι πέρα των δυνατοτήτων τους, ειδοποιούν αμέσως τον οδηγό, που πάντα θα πρέπει να είναι σε ετοιμότητα να πάρει τον έλεγχο. Παραδείγματα αυτής της κατηγορίας αποτελούν τα αυτοκίνητα Tesla Model S και Model X.

Επίπεδο 3: Στο επίπεδο τρία αρχίζουμε και μιλάμε για οχήματα, τα οποία έχουν πλήρη αυτονομία. Σε αυτή την κατηγορία το αυτοκίνητο είναι σε θέση να πάει από το σημείο Α στο σημείο Β αποκλειστικά μόνο του. Ωστόσο, τα αυτοκίνητα του συγκεκριμένου επιπέδου παρουσιάζουν κάποιες δυσκολίες, διότι, ενώ ο οδηγός έχει την ελευθερία να πάρει τα μάτια από τον δρόμο (σύμφωνα με το όχημα), το όχημα δεν είναι σε θέση να αναγνωρίσει αν κάτι είναι πέρα των δυνατοτήτων του. Συνεπώς, ο οδηγός θα πρέπει πάντα να προσέχει τον δρόμο και να είναι έτοιμος να πάρει τον έλεγχο.

Επίπεδο 4: Λόγω των δυσκολιών που παρουσιάζει το προηγούμενο επίπεδο συνήθως οι εταιρίες κάνουν την μετάβαση από το επίπεδο 2 στο 4. Στο επίπεδο 4 αναφερόμαστε σε οχήματα, όπως το Waymo της Google, που έχουν την δυνατότητα κάτω από συγκεκριμένες συνθήκες (καιρικές ή αν βρίσκονται σε συγκεκριμένες τοποθεσίες), να είναι εντελώς αυτόνομα χωρίς η προσοχή του οδηγού να είναι απαραίτητη. Σε αντίθεση με τα αυτοκίνητα επιπέδου 3, τα οχήματα της κατηγορίας αυτής θα ειδοποιήσουν τον οδηγό, εάν πιστεύουν ότι κάτι είναι έξω από τις δυνατότητες τους.

Επίπεδο 5: Στο τελευταίο επίπεδο συναντάμε τα αυτοκίνητα, που είναι πραγματικά αυτόνομα και δεν χρειάζονται την προσοχή του οδηγού. Πιο συγκεκριμένα σε αυτού του τύπου αυτοκίνητα δεν υπάρχει καν τιμόνι. Είναι σε θέση να αντιμετωπίσουν οποιαδήποτε κατάσταση χωρίς κανένα πρόβλημα σε οποιαδήποτε συνθήκη. Το πρότζεκτ Nuro είναι ένα όχημα επιπέδου 5, το οποίο έχει ως τρέχουσα λειτουργία την μεταφορά πακέτων ή αντικειμένων γενικότερα.

4.4 Πλεονεκτήματα Αυτόνομων Οχημάτων

Τα πλεονεκτήματα της ανάπτυξης του συγκεκριμένου τομέα είναι αρκετά. Πιο συγκεκριμένα, ένα αυτόνομο αυτοκίνητο επιπέδου πέντε θα είναι σε θέση να παρέχει:

Μειωμένη κυκλοφοριακή συμφόρηση: Ένας λόγος, που υπάρχει μεγάλο πρόβλημα με την κυκλοφοριακή συμφόρηση, είναι το γεγονός ότι ο άνθρωπος δεν είναι σε θέση να συγχρονιστεί με κάθε οδηγό ως προς την κίνηση του οχήματος του. Ωστόσο, τα αυτόνομα οχήματα, λόγω της ικανότητας τους να διατηρούν συγκεκριμένη απόσταση από τα άλλα οχήματα, μειώνουν αρκετά τη συχνότητα που ένα αμάξι θα πρέπει να σταματήσει και να ξεκινήσει, συνεπώς περιορίζεται το πρόβλημα της κυκλοφοριακής συμφόρησης.

Ασφάλεια: Είναι το κύριο πλεονέκτημα των οχημάτων αυτών, καθώς το 94 % των ατυχημάτων συμβαίνει από ανθρώπινο λάθος. Η συμπεριφορά ενός ατόμου στην οδήγηση εξαρτάται από πολλούς παράγοντες, όπως η διάθεση, ή η ανάγκη να φτάσει στον προορισμό του γρήγορα. Παράγοντες, όπως αυτοί κάνουν τον άνθρωπο απρόβλεπτο στις αποφάσεις του στον δρόμο. Το πρόβλημα μπορεί να περιοριστεί σε μεγάλο βαθμό, όταν το αμάξι είναι σε θέση να πάρει αποφάσεις, βασισμένες μόνο στους κανόνες οδικής κυκλοφορίας και να έχει ως προτεραιότητα την ασφάλεια των επιβατών καθώς και των οδηγών γύρω του.

Μειωμένο Ενεργειακό Κόστος: Το ενεργειακό κόστος των συμβατικών αυτοκινήτων είναι αρκετά μεγάλο. Συνεπώς, υπάρχει σημαντικός αντίκτυπος στο περιβάλλον καθώς και στη διαθεσιμότητα των πόρων. Ένα αυτόνομο όχημα μπορεί να μειώσει αρκετά το κόστος, διότι είναι σε θέση να κάνει τις βέλτιστες επιλογές σχετικά με τον τρόπο οδήγησης.

Εύκολη Πρόσβαση: Τα αυτόνομα αυτοκίνητα και ειδικά αυτά, τα οποία είναι επιπέδου 5, θα είναι σε θέση να διευκολύνουν άτομα με δυσκολίες στην οδήγηση, όπως άτομα με ειδικές ανάγκες και ηλικιωμένους.

4.5 Προκλήσεις

Διατήρηση Χαρτών: Αναφερόμενοι σε οχήματα επιπέδου 4 και άνω, η ανάγκη χαρτογράφησης και συλλογής δεδομένων μιας περιοχής είναι απαραίτητη, πριν δοκιμαστεί κανονικά το όχημα. Για να διατηρηθεί η ακρίβεια ενός αυτόνομου οχήματος και να είναι σε θέση να παίρνει αποφάσεις σε πραγματικό χρόνο, χρειάζονται αρκετά δεδομένα της περιοχής που θα γίνει η πλοήγηση. Αυτή η διαδικασία είναι αρκετά χρονοβόρα, καθώς πραγματικοί οδηγοί πραγματοποιούν την πλοήγηση για την συλλογή δεδομένων. Συνεπώς, οι υποστηριζόμενες περιοχές που το αμάξι θα δοκιμαστεί είναι αρκετά περιορισμένες. Το πρόβλημα δεν σταματάει εδώ καθώς καθημερινά τα δεδομένα μπορούν να αλλάξουν λόγω κατασκευών ή έργων μέσα στην πόλη. Συνεπώς η αρκετά συχνή ανανέωση χαρτών κάνει δύσκολη την υποστήριξη για αρκετές περιοχές.

Καιρικές Συνθήκες: Οι καιρικές συνθήκες κατά την διάρκεια της πλοήγησης ενός αυτόνομου οχήματος είναι αρκετά σημαντικός παράγοντας. Σε καιρικές συνθήκες, όπου οι αισθητήρες του οχήματος επηρεάζονται, το όχημα δυσκολεύεται να πάρει σωστές αποφάσεις σχετικά με την κίνηση.

Νομοθεσία: Ένα πρόβλημα που προκύπτει είναι οι κανονισμοί για χρήση αυτόνομων οχημάτων. Οι εταιρίες βρίσκονται πιο κοντά στην ολοκλήρωση αυτόνομων οχημάτων και ταυτόχρονα πιο μακριά από την κυκλοφορία τους στο ευρύ κοινό. Οι εταιρίες, που κάνουν δοκιμές σε κλειστό περιβάλλον, τελικά θα χρειαστεί να κάνουν δοκιμές σε περιβάλλον με κανονικά οχήματα. Ωστόσο, σε αρκετές πολιτείες της Αμερικής η τεχνολογία αυτή θεωρείται

αρκετά επικίνδυνη και δεν επιτρέπεται σε περιβάλλον με συμβατικά αυτοκίνητα. Συνεπώς οι συνθήκες γίνονται ακόμα πιο δύσκολες για την πλήρη ανάπτυξη αυτόνομων οχημάτων.

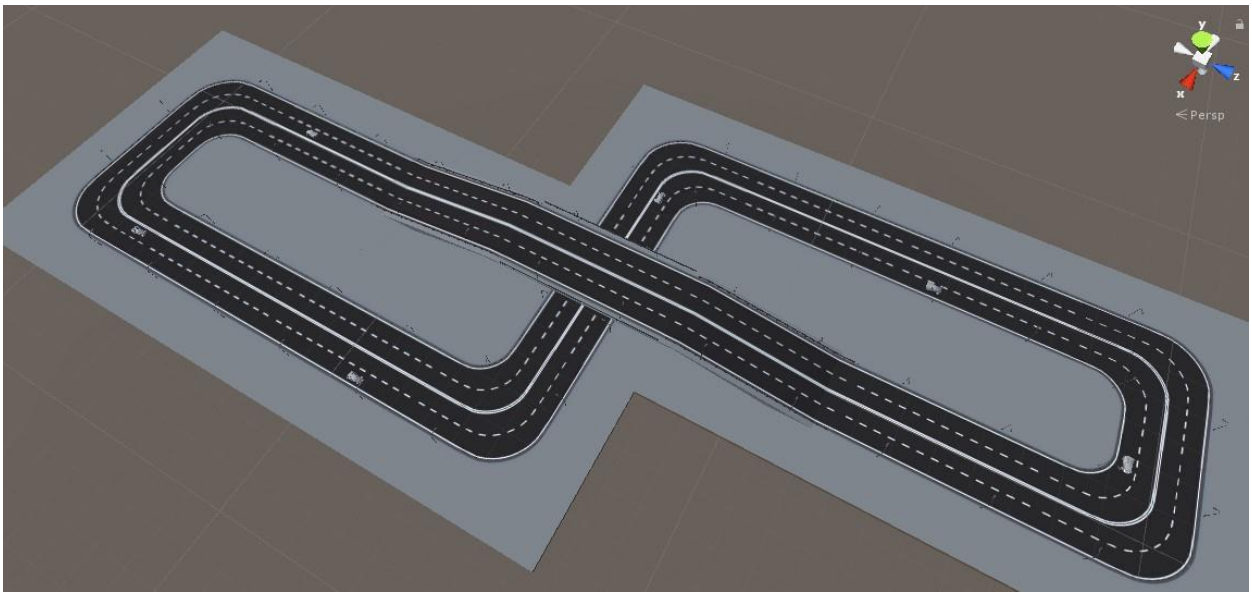
Άνθρωπος: Τα αυτόνομα οχήματα ως κύριο σκοπό έχουν την ασφάλεια της μετακίνησης του ανθρώπου. Ωστόσο, μια μεγάλη πρόκληση που προκύπτει είναι ο ίδιος ο άνθρωπος. Μεγάλο ποσοστό ατόμων δεν θεωρεί τα αυτόνομα οχήματα ασφαλή και δεν θα τα επέλεγαν για την μετακίνηση τους.

5. Ανάπτυξη της εφαρμογής

5.1 Εργαλεία

5.1.1 Unity

Η προσομοίωση του περιβάλλοντος για το αυτόνομο όχημα γίνεται στην μηχανή Unity. Η κύρια χρήση της Unity είναι η ανάπτυξη ηλεκτρονικών παιχνιδιών για πολλαπλές πλατφόρμες. Ωστόσο, οι δυνατότητες της δεν σταματάνε εκεί καθώς παρέχει εργαλεία εικονικής και επαυξημένης πραγματικότητας που κάνουν ρεαλιστική την απεικόνιση της εφαρμογής σε ένα διαφορετικό περιβάλλον. Όλα τα παραπάνω μετατρέπουν την μηχανή Unity σε ένα εύχρηστο περιβάλλον για την ανάπτυξη εφαρμογών που έχουν ως κύριο χαρακτηριστικό την προσομοίωση περιβάλλοντος.



Εικόνα 5.1 – Περιβάλλον προσομοίωσης στην Unity

5.1.2 Tensorflow

Όσον αφορά το κομμάτι της μηχανικής μάθησης που θα ενσωματώσουμε, θα γίνει χρήση του εργαλείου **Tensorflow.** Το Tensorflow είναι μια ανοικτού λογισμικού πλατφόρμα που υποστηρίζει τεχνικές μηχανικής μάθησης. Περιέχει ένα μεγάλο εύρος από βιβλιοθήκες και εργαλεία που χρησιμοποιούνται στους αλγορίθμους μηχανικής μάθησης μετατρέποντας το ένα από τα απαραίτητα εργαλεία για τέτοιου τύπου εφαρμογές.

5.1.3 Keras

Στην κορυφή του εργαλείου Tensorflow θα χρησιμοποιήσουμε το API Keras. Είναι μια βιβλιοθήκη βαθιάς μάθησης, που μας επιτρέπει να χρησιμοποιούμε και να διαχειριζόμαστε τα δίκτυα μηχανικής μάθησης ακόμα πιο εύκολα σε συνδυασμό με το Tensorflow. Περιέχει εύκολη προσαρμογή συνελικτικών επιπέδων και επιπέδων συγκέντρωσης δημιουργώντας ένα εύχρηστο περιβάλλον για δημιουργία δικτύων.

5.1.4 Περιγραφή της εφαρμογής

Σκοπός της εφαρμογής είναι η κατασκευή αυτόνομου οχήματος με χρήση μηχανικής μάθησης. Το ζητούμενο αποτέλεσμα είναι το αυτοκίνητο να αναπτύξει γνώση σχετικά με το περιβάλλον που βρίσκεται και να κάνει ενέργειες πάνω σε αυτή τη γνώση. Παράλληλα θα γίνει σύγκριση αποτελεσμάτων μεταξύ δικτύων για το συγκεκριμένο μοντέλο.

5.2 Συλλογή Δεδομένων

Το πρώτο στάδιο είναι η συλλογή δεδομένων. Η συλλογή δεδομένων πραγματοποιείται όσο γίνεται πλοήγηση του αυτοκίνητου στο περιβάλλον προσομοίωσης. Για την ίδια την συλλογή χρησιμοποιείται το module MSS, με το οποίο μπορούμε να κάνουμε αποκοπή συγκεκριμένου μέρους της οθόνης και να παίρνουμε στιγμιότυπα (screenshots) όσο πιο γρήγορα γίνεται στο κομμάτι αυτό. Το είδος των δεδομένων που δίνουμε στο δίκτυο είναι τα frames μαζί με την είσοδο που έχει δοθεί από το πληκτρολόγιο την συγκεκριμένη χρονική στιγμή κατά τη διάρκεια της πλοήγησης στο χώρο. Σε κάθε δευτερόλεπτο ουσιαστικά λαμβάνονται στιγμιότυπα, με συχνότητα περίπου 30 στιγμιότυπα ανα δευτερόλεπτο (frames per second) καθώς και η ανάλογη είσοδος που αντιστοιχεί σε κάθε frame. Τα στιγμιότυπα λαμβάνονται σε κάμερα πρώτου προσώπου, ώστε να φαίνονται ο δρόμος και τα υπόλοιπα οχήματα.

Εικόνα 5.2 – Κάμερα πρώτου προσώπου (first person view)

Για απλότητα σχετικά με το μοντέλο που θα δημιουργηθεί, η είσοδος που εισάγεται από το πληκτρολόγιο είναι τα πλήκτρα **W, A** και **D.** Αυτά μεταφράζονται σε ευθεία, αριστερά και δεξιά. Τα δεδομένα που λήφθηκαν παρουσιάζουν το όχημα να πηγαίνει ευθεία και ανάμεσα στις λωρίδες και ποτέ στο αντίθετο ρεύμα. Επιπρόσθετα, εάν το όχημα εντοπίσει άλλο όχημα μπροστά και στην ίδια λωρίδα, πηγαίνει στην ελεύθερη δίπλα λωρίδα.

5.3 Επεξεργασία Δεδομένων

Κάθε στιγμιότυπο, που λαμβάνεται, περνάει από μια μικρή επεξεργασία. Αρχικά τα κανάλια χρωμάτων του μετατρέπονται σε αριθμητικές τιμές. Στην συνέχεια μετατρέπουμε την εικόνα σε ασπρόμαυρη. Αυτό γίνεται, καθώς δεν μας ενδιαφέρει το μοντέλο να προσαρμόζεται σε λεπτομέρειες, όπως οι συνθήκες καιρού ή η διαφορά μέρας και νύχτας. Τέλος η εικόνα μετατρέπεται σε 160 x 120 ανάλυση για πιο εύκολη επεξεργασία από το δίκτυο. Κάθε 500 στιγμιότυπα γίνεται αποθήκευση και τα δεδομένα αποθηκεύονται σε ένα numpy αρχείο (ως πίνακας με αριθμητικές τιμές). Συνήθως η πιο συχνή είσοδος στο αρχείο είναι η επιτάχυνση (κουμπί W) περίπου 80% των περιπτώσεων και το υπόλοιπο 20% θα είναι αριστερά και δεξιά. Εάν τροφοδοτήσουμε αρχεία στο δίκτυο, που περιέχουν τόσο μεγάλες διαφορές στα inputs, τότε υπάρχει κίνδυνος **υπερπροσαρμογής (overfitting)** του μοντέλου κατά τη διάρκεια της εκπαίδευσης. Ο σκοπός ενός μοντέλου μηχανικής μάθησης είναι να γενικεύσει όσο καλύτερα γίνεται εισόδους που δεν έχει ξανασυναντήσει. Υπερπροσαρμογή έχουμε, όταν το μοντέλο δεν γενικεύει καλά, σε παραδείγματα που δεν έχει ξαναδεί και, αντιθέτως, προσπαθεί να βρει μοτίβα να προσαρμοστεί, μόνο στα δεδομένα που του τροφοδοτούμε. Συνεπώς, για να αποφύγουμε το πρόβλημα αυτό, θέλουμε τα inputs που παίρνει να είναι ισορροπημένα (balanced) και όσο πιο τυχαία γίνεται. Για αυτό τον λόγο θέλουμε να έχουμε τον ίδιο αριθμό εισόδων για κάθε κατηγορία (αριστερά, ευθεία, δεξιά). Εφαρμόζουμε την παραπάνω τεχνική στα δεδομένα μας καθώς και ανακάτεμα για τον ίδιο

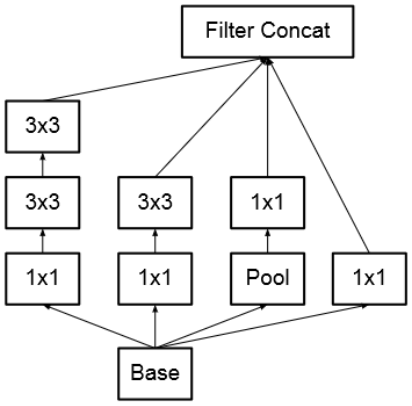
λόγο. Τέλος μαζί με την ισορρόπηση των δεδομένων το συνολικό μέγεθος που θα εκπαιδευτεί το μοντέλο είναι 240 χιλιάδες δεδομένα.

5.4 Εκπαίδευση δεδομένων

Στο κομμάτι της εκπαίδευσης επιλέγουμε από τα δεδομένα, που συλλέξαμε, την εικόνα ως είσοδο για εκπαίδευση και τις εισόδους του πληκτρολογίου ως το αποτέλεσμα που χρειαζόμαστε για τις προβλέψεις του μοντέλου. Επιπρόσθετα το 10% των δεδομένων τα οποία δίνονται στο μοντέλο κατά τη διάρκεια της εκπαίδευσης το χρησιμοποιούμε ως δεδομένα για επαλήθευση. Αυτό το κάνουμε για να δούμε πως αντιδρά το μοντέλο μας σε δεδομένα εκτός εκπαίδευσης, πιο απλά, για να δούμε αν το μοντέλο μας μπορεί να γενικεύσει. Για την εκπαίδευση χρησιμοποιούμε τα δίκτυα Inception v3 και Xception της Google.

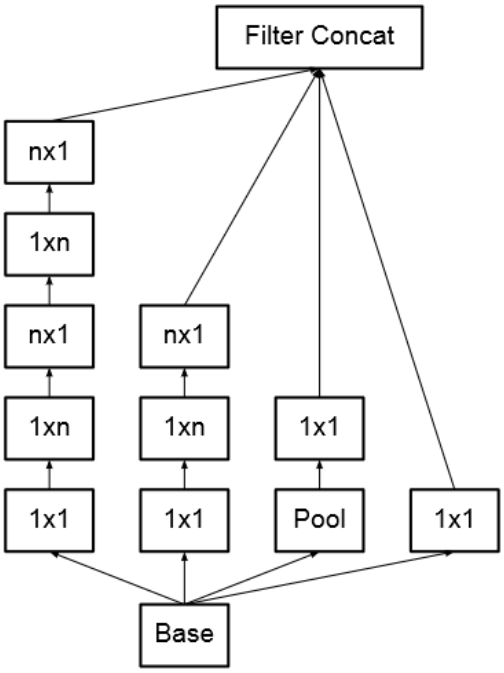
5.4.1 Δίκτυο Inception

Inception-v3: Συνελικτικό δίκτυο κατασκευασμένο από την Google. Αποτελείται από 42 κρυφά επίπεδα. Τα δίκτυα Inception v3 καθώς και οι προηγούμενες εκδόσεις του αποτελούνται από διαδοχικές μονάδες (modules) Inception, όπως αποκαλούνται, οι οποίες περιέχουν τμήματα συνέλιξης καθώς και 1 επίπεδο μέγιστης συγκέντρωσης (max pooling). Η αρχιτεκτονική του δικτύου αυτού περιλαμβάνει τριών τύπων διαφορετικές μονάδες. Η πρώτη μονάδα εμφανίζεται να περιέχει τμήματα συνέλιξης 3x3 και 1x1 καθώς και τμήμα συγκέντρωσης. Στο αρχικό δίκτυο GoogleNet τα τμήματα συνέλιξης, τα οποία ήταν 5x5, αντικαταστάθηκαν με 2 διαδοχικά τμήματα συνέλιξης 3x3 καθώς ήταν 2.78 φορές πιο γρήγορα, όπως φαίνεται στην εικόνα 5.3 στο αριστερό μέρος.



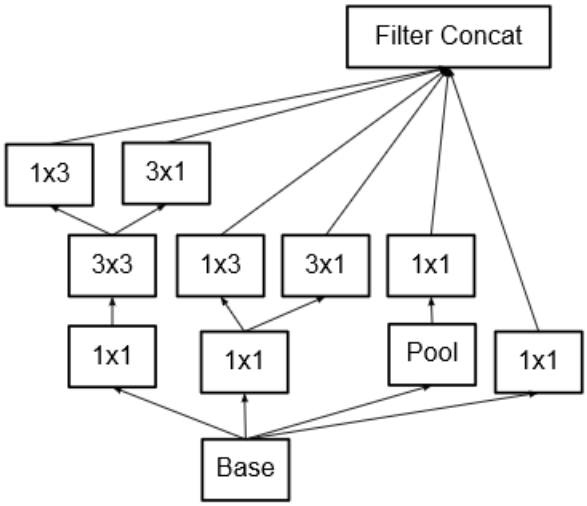
Εικόνα 5.3 – Μονάδα Α

Στην επόμενη μονάδα γίνονται παραγοντοποιήσεις στα φίλτρα. Συνεπώς, εάν έχουμε ένα φίλτρο 3x3 θα μετατραπεί σε 1x3 καθώς θα ακολουθεί φίλτρο 3x1. Με αυτόν τον τρόπο η Google μείωσε ακόμα περισσότερο το κόστος της απόδοσης στα τμήματα συνέλιξης, χωρίς να μειώσει την ακρίβεια.



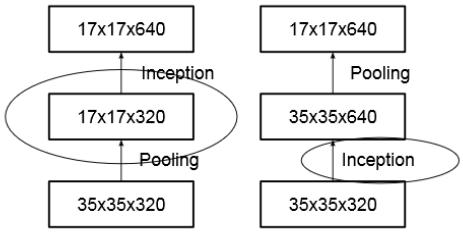
Εικόνα 5.4 – Μονάδα Β

Τα φίλτρα συνέλιξης στην τρίτη μονάδα αυξάνονται ως προς το πλάτος και όχι ως προς το βάθος (expanded filter banks), ώστε να μειώσουν την παραστατική συμφόρηση (representational bottleneck).



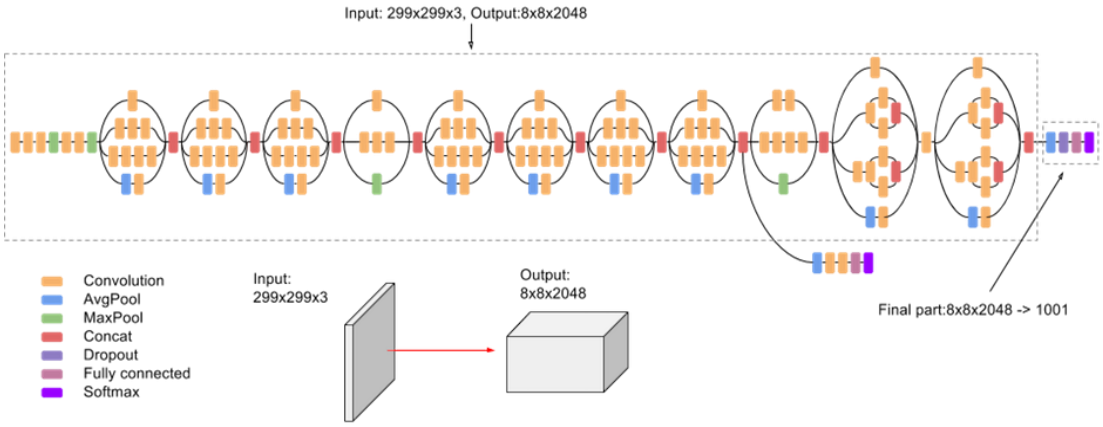
Εικόνα 5.5 – Μονάδα C

Επιπλέον, ενδιάμεσα στα παραπάνω modules υπάρχουν κάποια στάδια τα οποία ονομάζονται Grid Size Reduction. Όπως έχει αναφερθεί πιο νωρίς, μετά από κάθε τμήμα συνέλιξης τα δίκτυα αυτά περιλαμβάνουν ένα τμήμα συγκέντρωσης (pooling), ώστε να μειωθεί το μέγεθος των feature maps που προκύπτουν από την συνέλιξη. Το παραπάνω στάδιο προστίθεται στο δίκτυο Inception ως αντικατάσταση της συγκέντρωσης, η οποία κανονικά θα είχε μεγάλο κόστος στην απόδοση του δικτύου.



Εικόνα 5.6 – Reducing Grid Size

Τέλος, στο επίπεδο εξόδου του δικτύου η τελική έξοδος υπολογίζεται από την συνάρτηση ενεργοποίησης softmax. Το δίκτυο αποτελείται από 3 x Module A, 5 x Module B και 2 x Module C. Το ολοκληρωμένο δίκτυο φαίνεται στην εικόνα 5.7.



Εικόνα 5.7 – Αρχιτεκτονική Inception v3

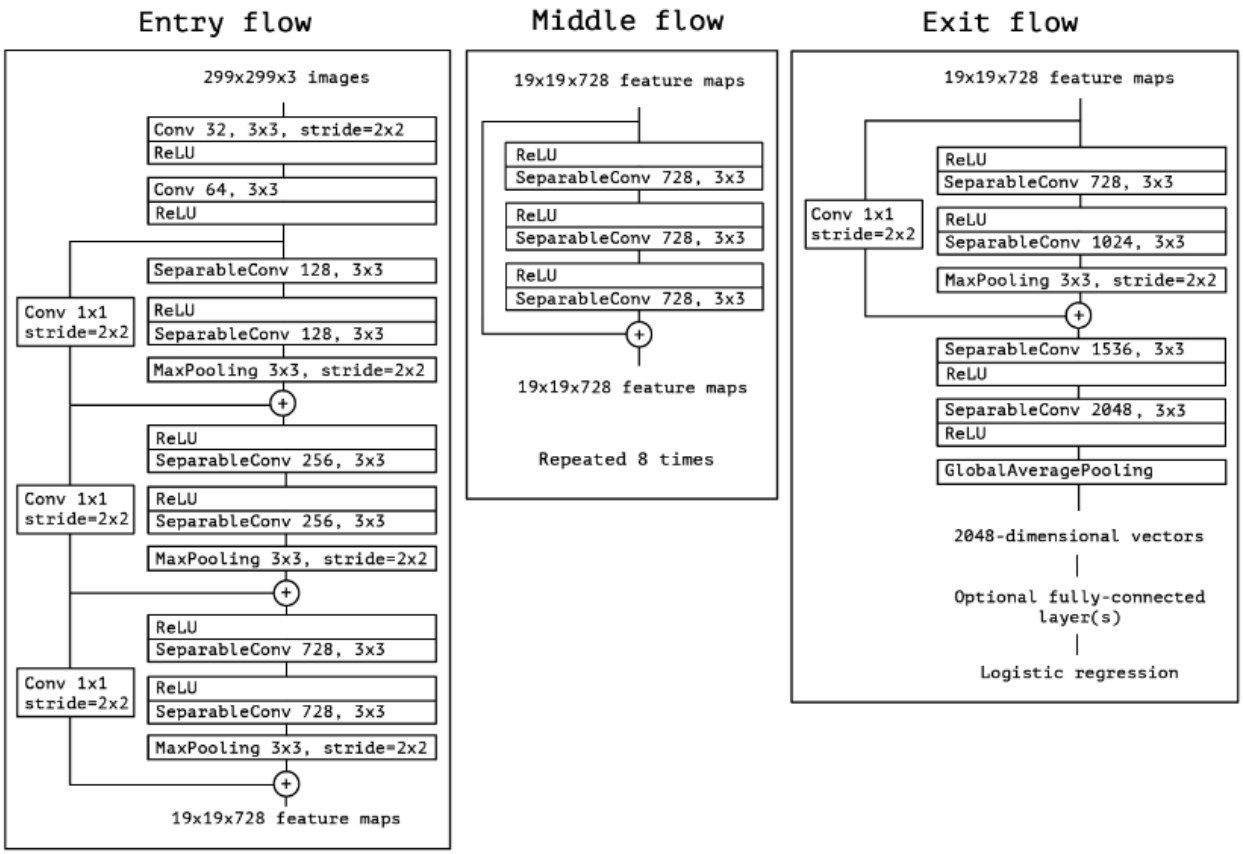
Το Inception-v3 πέτυχε 78 % ακρίβεια στον διαγωνισμό **ILSVRC** (ImageNet Large Scale Visual Recognition Competition) 2015 και αποτελεί βελτίωση ως προς τους προκάτοχους του Inception v1 και v2. Το ImageNet είναι ένα μεγάλο σετ δεδομένων με πάνω από 15 εκατομμύρια εικόνες και περίπου 22 χιλιάδες κατηγορίες, κατασκευασμένο για σκοπούς έρευνας και αποτελεί αρκετά καλό σετ εκπαίδευσης για δίκτυα μηχανικής μάθησης. Το ILSVRC χρησιμοποιεί ένα υποσύνολο του ImageNet με 1.2 εκατομμύρια εικόνες για εκπαίδευση, 50 χιλιάδες εικόνες για επαλήθευση και περίπου 100 χιλιάδες εικόνες για έλεγχο των δικτύων.

5.4.2 Δίκτυο Xception

Xception: Το δίκτυο Xception (extreme inception) είναι μια παραλλαγή του δικτύου Inception-v3. Τα inception modules στο δίκτυο Xception έχουν τροποποιηθεί ως προς τον τρόπο που πραγματοποιείται η συνέλιξη.

Πάρα πολλά δίκτυα χρησιμοποιούν τις τεχνικές συνέλιξης, που αναφέρθηκαν στην ενότητα 3.5. Ωστόσο δίκτυα όπως το Xception κάνουν χρήση μιας άλλης διαδικασίας συνέλιξης η οποία ονομάζεται διαχωριστική ως προς το βάθος συνέλιξη (depthwise separable convolution). Η διαδικασία αυτή περιέχει δύο στάδια. Στο πρώτο στάδιο γίνεται ο διαχωρισμός των καναλιών εισόδου και η μετατροπή τους σε nxn συνελίξεις. Εάν για παράδειγμα υπάρχουν 5 κανάλια τότε αυτά θα χωριστούν σε 5 nxn συνελίξεις. Τα κανάλια που έχουν προκύψει από το προηγούμενο στάδιο συνέλιξης ενώνονται και σχηματίζουν μια εικόνα. Με την χρήση ενός φίλτρου 1x1 θα γίνει συνέλιξη αυτή τη φορά στην εικόνα που δόθηκε ως αποτέλεσμα στο προηγούμενο στάδιο. Αυτή η διαδικασία είναι ιδιαίτερα χρήσιμη, καθώς ελαχιστοποιεί σε μεγάλο ποσοστό τις πράξεις που πρέπει να γίνουν κατά τη διάρκεια της εκπαίδευσης. Ως αποτέλεσμα το δίκτυο γίνεται αρκετά ελαφρύ και γρήγορο κατά τη διάρκεια της εκπαίδευσης.

Ωστόσο η διαδικασία που αναφέρθηκε είναι διαφορετική για το δίκτυο Xception. Τα στάδια ένα και δυο αντιστρέφονται με αποτέλεσμα η 1x1 συνέλιξη (Pointwise Convolution) να γίνεται πρώτα και στην συνέχεια να ακολουθεί η διαχωρισμένη συνέλιξη(depthwise convolution).



Εικόνα 5.8 – Αρχιτεκτονική Xception

Τα modules SeperableConv είναι τα τροποποιημένα κομμάτια του δικτύου και χρησιμοποιούνται στην θέση των Inception modules που αναφέρθηκαν στην ενότητα 5.4.1.

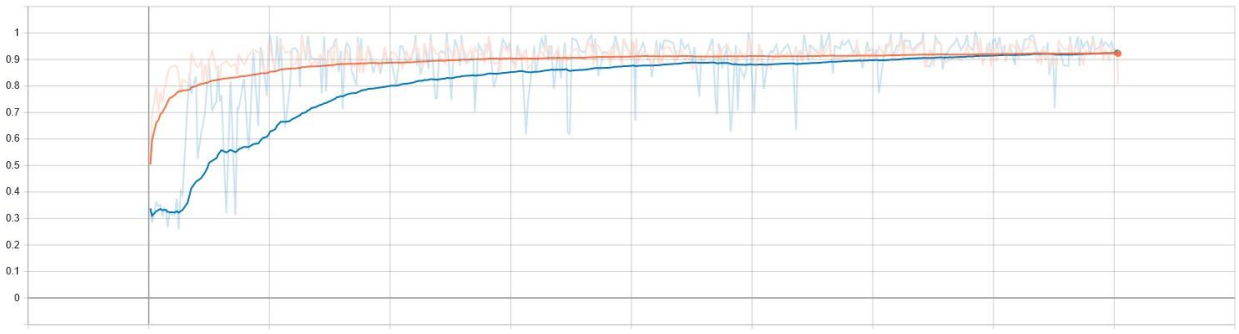
Το δίκτυο Xception κατάφερε να πετύχει μεγαλύτερη απόδοση από τον προκάτοχο του Inception v3 πετυχαίνοντας 79% ακρίβεια στον διαγωνισμό ILSVRC.

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

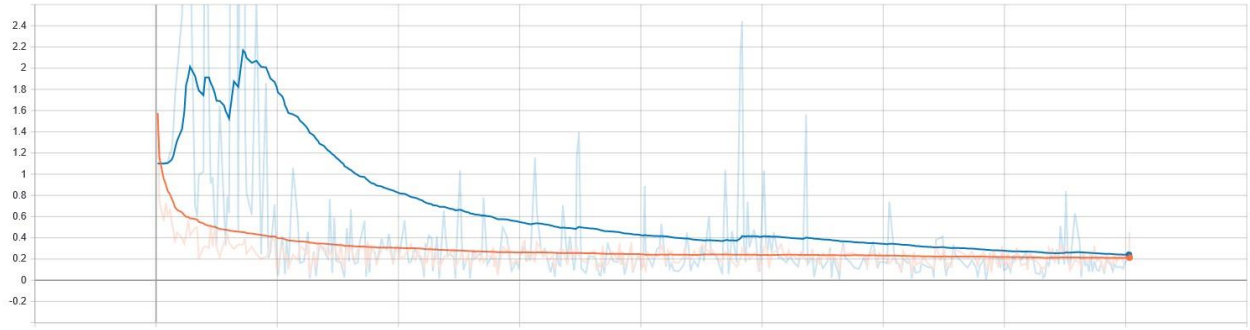
Εικόνα 5.9 – Αποτελέσματα ILSVRC

5.5 Αποτελέσματα

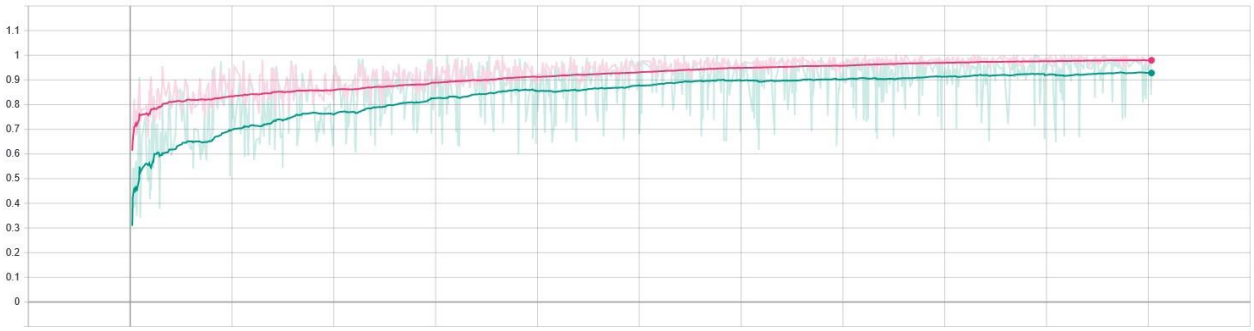
Με την βοήθεια του εργαλείου Tensorboard, που παρέχεται μαζί με το εργαλείο Tensorflow είμαστε σε θέση να απεικονίσουμε τα αποτελέσματα των μοντέλων μας. Παρατηρούμε ότι τα δύο δίκτυα έχουν αρκετά καλά αποτελέσματα με το μοντέλο Xception να πετυχαίνει 92% ακρίβεια και 95% ακρίβεια επαλήθευσης και το Inception v3 96% ακρίβεια και 95% ακρίβεια επαλήθευσης. Όσον αφορά το σφάλμα, το ποσοστό βρίσκεται κατά προσέγγιση στο 10-15%, καθώς το σφάλμα κατά την επαλήθευση βρίσκεται περίπου στο 10%. Οι τιμές που μας ενδιαφέρουν είναι οι τιμές της επαλήθευσης (validation), καθώς είναι τα τυχαία δεδομένα που το μοντέλο μας αντιμετωπίζει. Είναι ουσιαστικά οι πραγματικές τιμές που θα μας εξηγήσουν αν το μοντέλο τα πήγε καλά. Μετά από δοκιμές παρατήρησα πως ένα καλό σημείο, για να σταματήσει η εκπαίδευση του μοντέλου Inception είναι 20 επαναλήψεις (epochs), ενώ χρειάστηκαν μόλις 5 για το Xception. Όλα τα μοντέλα εκπαιδεύτηκαν με ρυθμό εκμάθησης 0,001 και RMSprop optimizer. Μετά τις 5 επαναλήψεις το μοντέλο Xception σταμάτησε να παρουσιάζει σημάδια βελτίωσης ,ενώ μετά τις 7 άρχισε να υπερπροσαρμόζεται.



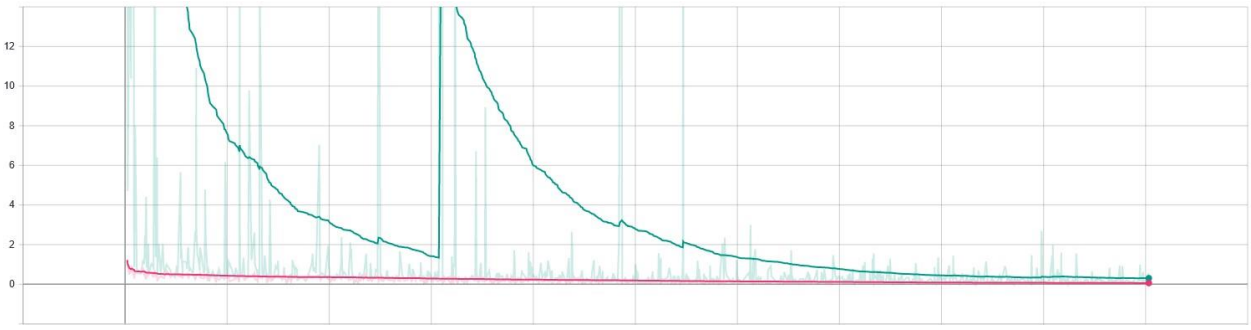
Εικόνα 5.9 – Validation (μπλε χρώμα)/Training (πορτοκαλί χρώμα) Accuracy μοντέλο Xception



Εικόνα 5.10 – Validation (μπλε χρώμα)/Training (πορτοκαλί χρώμα) Loss μοντέλο Xception



Εικόνα 5.11 – Validation (πράσινο χρώμα)/Training (ροζ χρώμα) Accuracy μοντέλο Inception v3



Εικόνα 5.12 – Validation (πράσινο χρώμα)/Training (ροζ χρώμα) Loss μοντέλο Inception v3

Τέλος για το κομμάτι του testing, το μοντέλο εκπαιδεύτηκε να δίνει προβλέψεις ως προς τα input (αριστερά , ευθεία, δεξιά) καθώς βρισκόμαστε στο περιβάλλον προσομοίωσης. Με τον ίδιο τρόπο που έγινε η λήψη δεδομένων στο στάδιο της συλλογής δοκιμάζουμε το μοντέλο που εκπαιδεύσαμε. Το μοντέλο παίρνει είσοδο στιγμιότυπα και αυτή τη φορά προβλέπει την κίνηση του οχήματος και με κάθε πρόβλεψη το όχημα εκτελεί την ανάλογη κίνηση. Ως προς την προσομοίωση το μοντέλο κατάφερε να μείνει αρκετά σταθερό στις γραμμές, ενώ τις περισσότερες φορές είχε επιτυχία στην προσπέραση άλλων οχημάτων.

Συμπεράσματα

Με την χρήση των παραπάνω μοντέλων μηχανικής μάθησης, ανακαλύψαμε ότι με μόλις 240 χιλιάδες δεδομένα είναι δυνατόν να δημιουργήσουμε ένα μοντέλο σχετικά σταθερό, το οποίο έχει γνώση του περιβάλλοντος στο οποίο κινείται. Οι επιδόσεις των δικτύων Inception και Xception ήταν αρκετά καλές και ίσως να μπορούμε να πάρουμε καλύτερα αποτελέσματα με διαφορετικές παραμέτρους. Ωστόσο, ενώ τα αποτελέσματα είναι ικανοποιητικά σε ένα περιβάλλον προσομοίωσης, ο τομέας των αυτόνομων οχημάτων είναι σχετικά νέος και θέλει αρκετό χρόνο ακόμα ώστε να μετατραπεί σε κάτι που θα χρησιμοποιείται ευρέως. Συμπληρωματικά η εργασία μπορεί να βελτιωθεί, με την προσθήκη ανίχνευσης αντικειμένων (object detection), η οποία είναι απαραίτητη για ένα αυτόνομο όχημα καθώς και την συλλογή μεγαλύτερου όγκου δεδομένων.

Παράρτημα Α: Βασικά κομμάτια κώδικα

collect_data.py

```
01 import numpy as np
02 import pandas as pd
03 import cv2
04 import time
05 import os
06 import re
07 import mss
08 from balance_inputs import balance_inputs
09 from getkeys import key_check
10 from directkeys import ReleaseKey, W, A, D
11
12 file = '/Datasets/train_set.npy'
13 data_save_dir = 'Datasets'
14
15 #Μέγεθος αρχείου προς αποθήκευση
16 CHUNK_SIZE = 9000
17 #Όρια της οθόνης προς αποκοπή
18 capture_screen = {"top": 40, "left": 0, "width": 800, "height": 640}
19
20
21 def getFilenameIndex(filename):
22      return re.search(r'\d+', filename).group(0)
23
24 #Βρίσκει το μέγιστο αριθμό αρχείου στον φάκελο Datasets
25 def getCurrentIndex():
26  index_list = []
27  if not os.listdir(data_save_dir):
28      return 1
29  else:
30      for filename in os.listdir(data_save_dir):
31          i = getFilenameIndex(filename)
32          index_list.append(int(i))
33
34  i = max(index_list)
35  i+=1
36  return i
37
38
39 def checkExistingData():
40  if os.path.isfile(file):
41      print("Data already exists, loading existing file")
42      training_data = list(np.load(file))
43  elif os.path.isdir(data_save_dir):
44      training_data = []
45  else:
46      os.mkdir(data_save_dir)
47      training_data = []
48  return training_data
49
50
```

```
51 def main():
52
53  exit = 0
54  sct = mss.mss()
55
56  # Αντίστροφη μέτρηση για την εναλλαγή στο περιβάλλον προσομοίωσης
57  print("Countdown before the collection!")
58  for i in range(1, 4):
59      print(i)
60      time.sleep(1)
61
62  # Αρχικοποίηση λίστας δεδομένων
63  training_data = checkExistingData()
64  filenameIndex = getCurrentIndex()
65  filename = 'train_set{}.npy'.format(filenameIndex)
66
67  while not exit:
68      # Ξεκινάει η διαδικασία συλλογής στιγμιότυπων
69      # Το στιγμιότυπο που τραβήχτηκε το μετατρέπουμε
70      # σε αριθμητικές τιμές RGB και το αποθηκεύουμε
71      # σε numpy array
72      frame = np.array(sct.grab(capture_screen))
73      # Μετατροπή του στιγμιοτύπου σε grayscale
74      frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
75      # Τέλος γίνεται μετατροπή του frame σε 160x120
76      # ανάλυση για το δίκτυο μας
77      frame = cv2.resize(frame, (160, 120))
78
79      keys = key_check()
80
81      # Συλλογή των εισόδων χρήστη
82      user_input = [0, 0, 0]
83      if 'A' in keys:
84          user_input[0] = 1
85      elif 'D' in keys:
86          user_input[2] = 1
87      else:
88          user_input[1] = 1
89
90      # Δημιουργούμε τον τύπο του σετ εκπαιδευσής μας
91      # το οποίο είναι στιγμιότυπο και είσοδος χρήστη
92      training_data.append([frame, user_input])
93
94      # Σώζουμε τα δεδομένα μας κάθε 500 frames
95      if len(training_data) % 500 == 0:
96          print("Data written: "+str(len(training_data)))
97          np.save(os.path.join('Datasets', filename), training_data)
98          # Ολοκληρώνουμε την συλλογή δεδομένων
99          if len(training_data) >= CHUNK_SIZE:
100                 exit = 1
101                 ReleaseKey(W)
102                 ReleaseKey(A)
103                 ReleaseKey(D)
104
105     # Τέλος κάνουμε balance τα δεδομένα μας
106     # για την αποφυγή overfitting
107     balance_inputs(filename)
```

```
108
109
110 if __name__ == "__main__":
111     main()
```

balance_inputs.py

```
01 import numpy as np
02 import pandas as pd
03 from collections import Counter
04 from random import shuffle
05 import os
06
07 def balance_inputs(filename):
08
09  left_inputs = []
10  right_inputs = []
11  accelerate_inputs = []
12
13  #Φορτώνουμε το αρχείο μας
14  trained_data = np.load(os.path.join('Datasets', filename))
15
16  shuffle(trained_data)
17
18  #Λαμβάνουμε τις εισόδους του χρήστη
19  #που έχουν πραγματοποιηθεί την
20  #ώρα της συλλογής
21  for data in trained_data:
22      frame = data[0]
23      user_input = data[1]
24
25      if user_input[0] == 1:
26          left_inputs.append([frame, user_input])
27      elif user_input[1] == 1:
28          accelerate_inputs.append([frame, user_input])
29      elif user_input[2] == 1:
30          right_inputs.append([frame, user_input])
31      else:
32          print("Invalid input")
33
34
35  #Βρίσκουμε τον ελάχιστο αριθμό εισόδων
36  min_len = min(len(right_inputs), len(accelerate_inputs),
len(left_inputs))
37
38  #Και μετατρέπουμε το μέγεθος όλων των εισόδων
39  #στον ελάχιστο αριθμό που βρήκαμε πριν
40  accelerate_inputs = accelerate_inputs[:min_len]
41  right_inputs = right_inputs[:min_len]
42  left_inputs = left_inputs[:min_len]
43
44  #Ενώνουμε τα δεδομένα μας
45  balanced_inputs = right_inputs + accelerate_inputs + left_inputs
```

```
46
47  #και τα ανακατεύουμε
48  shuffle(balanced_inputs)
49
50  #Τέλος αποθηκεύουμε το νέο αρχείο μας
51  np.save(os.path.join('Datasets', filename), balanced_inputs)
52
```

train_model.py

```
01 import os
02 import numpy as np
03 import tensorflow as tf
04 from tensorflow  import keras
05 from tensorflow.keras.models import load_model
06 from random import shuffle
07
08 #Πλάτος στιγμιοτύπου
09 IM_WIDTH = 160
10 #Ύψος στιγμιοτύπου
11 IM_HEIGHT = 120
12
13 #Μέγεθος batch που περνάει από το δίκτυο
14 #κατα τη διάρκεια της εκπαίδευσης
15 B_SIZE = 64
16
17 #Ρυθμός εκμάθησης
18 LR = 0.001
19
20 #Αριθμός περασμάτων εκπαίδευσης
21 EPOCHS = 10
22
23 #Αριθμός αρχείων
24 DATA_BATCHES = 116
25 EXISTING_MODEL = 0
26 NET_NAME = 'Xception'
27 OPTIMIZER = 'RMSprop'
28
29
30 def main():
31  #Επιλογή δικτύου για το μοντέλο που θα χρησιμοποιηθεί
32  model = keras.applications.inception_v3.InceptionV3(include_top=True,
33      weights=None, input_tensor=None, input_shape=(IM_WIDTH, IM_HEIGHT,
1), classes=3)
34
35  # model = keras.applications.xception.Xception(include_top=True,
36  #   weights=None, input_tensor=None, input_shape=(IM_WIDTH, IM_HEIGHT,
1), classes=3)
37
38  model_name = 'trained_model-{}-{}-{}-{}-{}.h5'.format(NET_NAME, LR,
EPOCHS, B_SIZE, OPTIMIZER)
39
40  if EXISTING_MODEL == 1:
```

```
41      model = load_model(model_name)
42  #Δημιουργία καταλόγου για την αποθήκευση logs
43  #για την απεικόνιση του μοντέλου στο Tensorboard
44  logdir = 'keras_log\\trained_model-{}-{}-{}-{}-{}.h5'.format(NET_NAME,
LR, EPOCHS, B_SIZE, OPTIMIZER)
45  #Δημιουργία του callback
46  tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir,
histogram_freq=1,
47                                      update_freq=1000)
48
49  #Επιλογή optimizer
50  rmsprop = keras.optimizers.RMSprop(learning_rate=LR, rho=0.9)
51
52  #Το μοντέλο γίνεται compile με RMSProp optimizer και
53  #συνάρτηση σφάλματος διασταυρωμένη εντροπία
54  model.compile(  optimizer=rmsprop,
55                  loss='categorical_crossentropy',
56                  metrics=['accuracy'],
57                          )
58  #Αριθμός επαναλήψεων της εκπαίδευσης
59  total = DATA_BATCHES * EPOCHS
60  iteration = 0
61
62  for epoch in range(EPOCHS):
63      data = [i for i in range(1, DATA_BATCHES+1)]
64      #Πραγματοποιείται ανακάτεμα σε
65      #κάθε πέρασματων δεδομένων
66      #για την αποφυγή overfitting
67      shuffle(data)
68      for index in enumerate(data):
69          trained_data = np.load(os.path.join('Datasets',
'train_set{}.npy'.format(index[1])))
70
71          #Δημιουργία εισόδου
72          X = np.array([i[0] for i in trained_data]).reshape(-
1,IM_WIDTH,IM_HEIGHT,1)
73          #Δημιουργία εξόδου
74          y = np.array([i[1] for i in trained_data])
75
76          iteration+=1
77          print("{}{}{}{}{}{}{}{}{}".format("Current Epoch: ", epoch+1,
"\nIteration: ", iteration, "/", total, "\n"))
78
79          #Εεκινά η διαδικασία της εκπαίδευσης
80          #Η παράμετρος validation_split μας επιτρέπει
81          #να πάρουμε από τα datasets ένα δείγμα για την
82          #επαλήθευση που θα κάνει το μοντέλο.
83          #Στην περιπτωσή αυτή χρησιμοποιείται 10% του
84          #δείγματος για επαλήθευση
85          model.fit(X, y, batch_size=B_SIZE, epochs=1, shuffle=True,
86              validation_split=0.1, callbacks=[tensorboard_callback])
87
88      model.save(model_name)
89
90
91 if __name__== "__main__":
92     main()
```

run_model.py

```
01 import os
02 import re
03 import cv2
04 import time
05 import mss
06 import random
07 import numpy as np
08 import tensorflow as tf
09 from tensorflow.keras.models import load_model, Model
10 from controls import accelerate, turn_left, turn_right
11
12 #Θέτουμε όριο πρόβλεψης για να
13 #πραγματοποιήσει μια ενέργεια το μοντέλο μας
14 TURN_PREDICTION = 0.85
15 STRAIGHT_PREDICTION = 0.60
16
17 #Διαστάσεις στιγμιοτύπου
18 IM_WIDTH = 160
19 IM_HEIGHT = 120
20
21 capture_screen = {"top": 40, "left": 0, "width": 800, "height": 640}
22
23 #Καθυστέρηση που δίνεται ως όρισμα
24 #στις συναρτήσεις κίνησης
25 DELAY = 0.08
26 #Ρυθμός εκμάθησης
27 LR = 0.001
28
29 #Εύρεση του ονόματος αρχείου
30 trained_model = 'trained_model-Xception-0.001-5-64-RMSprop.h5'
31 #Φόρτωση μοντέλου
32 model = load_model(trained_model)
33
34
35 def main():
36  #Αρχικοποίηση απεικόνισης οθόνης
37     sct = mss.mss()
38
39
40     for i in range(0,3):
41         i+=1
42         print(i)
43         time.sleep(1)
44
45     while(True):
46
47      #Η ίδια διαδικασία πραγματοποιήται
48      #όπως και στην συλλογη δεδομένων
49      #Αυτή τη φορά το δίκτυο θα δεχτεί
50      #ίδιου τύπου στιγμιότυπα και θα
51      #κάνει προβλέψεις
```

```
52
53          screen = np.array(sct.grab(capture_screen))
54
55          screen = cv2.cvtColor(screen, cv2.COLOR_BGR2GRAY)
56
57          screen = cv2.resize(screen, (IM_WIDTH,IM_HEIGHT))
58
59          screen = screen.reshape((-1, 160, 120, 1))
60          screen = tf.cast(screen, tf.float32)
61
62          prediction = model.predict(np.array(screen))[0]
63          print(prediction)
64
65
66       #Η πρόβλεψη που προκύπτει θέλουμε να είναι
67       #πάνω από κάποια όρια προκειμένου να είναι αποδεκτή
68          if prediction[1] > STRAIGHT_PREDICTION:
69              accelerate()
70          elif prediction[0] > TURN_PREDICTION:
71              turn_left(DELAY)
72          elif prediction[2] > TURN_PREDICTION:
73              turn_right(DELAY)
74          else:
75              accelerate()
76
77 if __name__== "__main__":
78      main()
```

https://github.com/keras-team/keras-applications/blob/master/keras_applications/inception_v3.py

keras_applications/inception_v3.py

```
001 """Inception V3 model for Keras.
002 Note that the input image format for this model is different than for
003 the VGG16 and ResNet models (299x299 instead of 224x224),
004 and that the input preprocessing function is also different (same as
Xception).
005 # Reference
006 - [Rethinking the Inception Architecture for Computer Vision](
007     http://arxiv.org/abs/1512.00567) (CVPR 2016)
008 """
009 from __future__ import absolute_import
010 from __future__ import division
011 from __future__ import print_function
012
013 import os
014
015 from . import get_submodules_from_kwargs
016 from . import imagenet_utils
017 from .imagenet_utils import decode_predictions
018 from .imagenet_utils import _obtain_input_shape
019
020
021 WEIGHTS_PATH = (
022     'https://github.com/fchollet/deep-learning-models/'
023     'releases/download/v0.5/'
024     'inception_v3_weights_tf_dim_ordering_tf_kernels.h5')
025 WEIGHTS_PATH_NO_TOP = (
026     'https://github.com/fchollet/deep-learning-models/'
027     'releases/download/v0.5/'
028     'inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5')
029
030 backend = None
031 layers = None
032 models = None
033 keras_utils = None
034
035
036 def conv2d_bn(x,
037               filters,
038               num_row,
039               num_col,
040               padding='same',
041               strides=(1, 1),
042               name=None):
043     """Utility function to apply conv + BN.
044     # Arguments
045         x: input tensor.
046         filters: filters in `Conv2D`.
```

```
047         num_row: height of the convolution kernel.
048         num_col: width of the convolution kernel.
049         padding: padding mode in `Conv2D`.
050         strides: strides in `Conv2D`.
051         name: name of the ops; will become `name + '_conv'`
052             for the convolution and `name + '_bn'` for the
053             batch norm layer.
054     # Returns
055         Output tensor after applying `Conv2D` and `BatchNormalization`.
056     """
057     if name is not None:
058         bn_name = name + '_bn'
059         conv_name = name + '_conv'
060     else:
061         bn_name = None
062         conv_name = None
063     if backend.image_data_format() == 'channels_first':
064         bn_axis = 1
065     else:
066         bn_axis = 3
067     x = layers.Conv2D(
068         filters, (num_row, num_col),
069         strides=strides,
070         padding=padding,
071         use_bias=False,
072         name=conv_name)(x)
073     x = layers.BatchNormalization(axis=bn_axis, scale=False,
name=bn_name)(x)
074     x = layers.Activation('relu', name=name)(x)
075     return x
076
077
078 def InceptionV3(include_top=True,
079                 weights='imagenet',
080                 input_tensor=None,
081                 input_shape=None,
082                 pooling=None,
083                 classes=1000,
084                 **kwargs):
085     """Instantiates the Inception v3 architecture.
086     Optionally loads weights pre-trained on ImageNet.
087     Note that the data format convention used by the model is
088     the one specified in your Keras config at `~/.keras/keras.json`.
089     # Arguments
090         include_top: whether to include the fully-connected
091             layer at the top of the network.
092         weights: one of `None` (random initialization),
093               'imagenet' (pre-training on ImageNet),
094               or the path to the weights file to be loaded.
095         input_tensor: optional Keras tensor (i.e. output of
`layers.Input()`)
096             to use as image input for the model.
097         input_shape: optional shape tuple, only to be specified
098             if `include_top` is False (otherwise the input shape
099             has to be `(299, 299, 3)` (with `channels_last` data format)
100             or `(3, 299, 299)` (with `channels_first` data format).
101             It should have exactly 3 inputs channels,
```

```
            and width and height should be no smaller than 75.
            E.g. `(150, 150, 3)` would be one valid value.
        pooling: Optional pooling mode for feature extraction
            when `include_top` is `False`.
            - `None` means that the output of the model will be
                the 4D tensor output of the
                last convolutional block.
            - `avg` means that global average pooling
                will be applied to the output of the
                last convolutional block, and thus
                the output of the model will be a 2D tensor.
            - `max` means that global max pooling will
                be applied.
        classes: optional number of classes to classify images
            into, only to be specified if `include_top` is True, and
            if no `weights` argument is specified.
    # Returns
        A Keras model instance.
    # Raises
        ValueError: in case of invalid argument for `weights`,
            or invalid input shape.
    """
    global backend, layers, models, keras_utils
    backend, layers, models, keras_utils =
get_submodules_from_kwargs(kwargs)

    if not (weights in {'imagenet', None} or os.path.exists(weights)):
        raise ValueError('The `weights` argument should be either '
                         '`None` (random initialization), `imagenet` '
                         '(pre-training on ImageNet), '
                         'or the path to the weights file to be loaded.')

    if weights == 'imagenet' and include_top and classes != 1000:
        raise ValueError('If using `weights` as `"imagenet"` with
`include_top`'
                         ' as true, `classes` should be 1000')

    # Determine proper input shape
    input_shape = _obtain_input_shape(
        input_shape,
        default_size=299,
        min_size=75,
        data_format=backend.image_data_format(),
        require_flatten=include_top,
        weights=weights)

    if input_tensor is None:
        img_input = layers.Input(shape=input_shape)
    else:
        if not backend.is_keras_tensor(input_tensor):
            img_input = layers.Input(tensor=input_tensor,
shape=input_shape)
        else:
            img_input = input_tensor

    if backend.image_data_format() == 'channels_first':
        channel_axis = 1
```

```
    else:
        channel_axis = 3

    x = conv2d_bn(img_input, 32, 3, 3, strides=(2, 2), padding='valid')
    x = conv2d_bn(x, 32, 3, 3, padding='valid')
    x = conv2d_bn(x, 64, 3, 3)
    x = layers.MaxPooling2D((3, 3), strides=(2, 2))(x)

    x = conv2d_bn(x, 80, 1, 1, padding='valid')
    x = conv2d_bn(x, 192, 3, 3, padding='valid')
    x = layers.MaxPooling2D((3, 3), strides=(2, 2))(x)

    # mixed 0: 35 x 35 x 256
    branch1x1 = conv2d_bn(x, 64, 1, 1)

    branch5x5 = conv2d_bn(x, 48, 1, 1)
    branch5x5 = conv2d_bn(branch5x5, 64, 5, 5)

    branch3x3dbl = conv2d_bn(x, 64, 1, 1)
    branch3x3dbl = conv2d_bn(branch3x3dbl, 96, 3, 3)
    branch3x3dbl = conv2d_bn(branch3x3dbl, 96, 3, 3)

    branch_pool = layers.AveragePooling2D((3, 3),
                                          strides=(1, 1),
                                          padding='same')(x)
    branch_pool = conv2d_bn(branch_pool, 32, 1, 1)
    x = layers.concatenate(
        [branch1x1, branch5x5, branch3x3dbl, branch_pool],
        axis=channel_axis,
        name='mixed0')

    # mixed 1: 35 x 35 x 288
    branch1x1 = conv2d_bn(x, 64, 1, 1)

    branch5x5 = conv2d_bn(x, 48, 1, 1)
    branch5x5 = conv2d_bn(branch5x5, 64, 5, 5)

    branch3x3dbl = conv2d_bn(x, 64, 1, 1)
    branch3x3dbl = conv2d_bn(branch3x3dbl, 96, 3, 3)
    branch3x3dbl = conv2d_bn(branch3x3dbl, 96, 3, 3)

    branch_pool = layers.AveragePooling2D((3, 3),
                                          strides=(1, 1),
                                          padding='same')(x)
    branch_pool = conv2d_bn(branch_pool, 64, 1, 1)
    x = layers.concatenate(
        [branch1x1, branch5x5, branch3x3dbl, branch_pool],
        axis=channel_axis,
        name='mixed1')

    # mixed 2: 35 x 35 x 288
    branch1x1 = conv2d_bn(x, 64, 1, 1)

    branch5x5 = conv2d_bn(x, 48, 1, 1)
    branch5x5 = conv2d_bn(branch5x5, 64, 5, 5)

    branch3x3dbl = conv2d_bn(x, 64, 1, 1)
```

```
    branch3x3dbl = conv2d_bn(branch3x3dbl, 96, 3, 3)
    branch3x3dbl = conv2d_bn(branch3x3dbl, 96, 3, 3)

    branch_pool = layers.AveragePooling2D((3, 3),
                                          strides=(1, 1),
                                          padding='same')(x)
    branch_pool = conv2d_bn(branch_pool, 64, 1, 1)
    x = layers.concatenate(
        [branch1x1, branch5x5, branch3x3dbl, branch_pool],
        axis=channel_axis,
        name='mixed2')

    # mixed 3: 17 x 17 x 768
    branch3x3 = conv2d_bn(x, 384, 3, 3, strides=(2, 2), padding='valid')

    branch3x3dbl = conv2d_bn(x, 64, 1, 1)
    branch3x3dbl = conv2d_bn(branch3x3dbl, 96, 3, 3)
    branch3x3dbl = conv2d_bn(
        branch3x3dbl, 96, 3, 3, strides=(2, 2), padding='valid')

    branch_pool = layers.MaxPooling2D((3, 3), strides=(2, 2))(x)
    x = layers.concatenate(
        [branch3x3, branch3x3dbl, branch_pool],
        axis=channel_axis,
        name='mixed3')

    # mixed 4: 17 x 17 x 768
    branch1x1 = conv2d_bn(x, 192, 1, 1)

    branch7x7 = conv2d_bn(x, 128, 1, 1)
    branch7x7 = conv2d_bn(branch7x7, 128, 1, 7)
    branch7x7 = conv2d_bn(branch7x7, 192, 7, 1)

    branch7x7dbl = conv2d_bn(x, 128, 1, 1)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 128, 7, 1)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 128, 1, 7)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 128, 7, 1)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 192, 1, 7)

    branch_pool = layers.AveragePooling2D((3, 3),
                                          strides=(1, 1),
                                          padding='same')(x)
    branch_pool = conv2d_bn(branch_pool, 192, 1, 1)
    x = layers.concatenate(
        [branch1x1, branch7x7, branch7x7dbl, branch_pool],
        axis=channel_axis,
        name='mixed4')

    # mixed 5, 6: 17 x 17 x 768
    for i in range(2):
        branch1x1 = conv2d_bn(x, 192, 1, 1)

        branch7x7 = conv2d_bn(x, 160, 1, 1)
        branch7x7 = conv2d_bn(branch7x7, 160, 1, 7)
        branch7x7 = conv2d_bn(branch7x7, 192, 7, 1)

        branch7x7dbl = conv2d_bn(x, 160, 1, 1)
```

```
        branch7x7dbl = conv2d_bn(branch7x7dbl, 160, 7, 1)
        branch7x7dbl = conv2d_bn(branch7x7dbl, 160, 1, 7)
        branch7x7dbl = conv2d_bn(branch7x7dbl, 160, 7, 1)
        branch7x7dbl = conv2d_bn(branch7x7dbl, 192, 1, 7)

        branch_pool = layers.AveragePooling2D(
            (3, 3), strides=(1, 1), padding='same')(x)
        branch_pool = conv2d_bn(branch_pool, 192, 1, 1)
        x = layers.concatenate(
            [branch1x1, branch7x7, branch7x7dbl, branch_pool],
            axis=channel_axis,
            name='mixed' + str(5 + i))

    # mixed 7: 17 x 17 x 768
    branch1x1 = conv2d_bn(x, 192, 1, 1)

    branch7x7 = conv2d_bn(x, 192, 1, 1)
    branch7x7 = conv2d_bn(branch7x7, 192, 1, 7)
    branch7x7 = conv2d_bn(branch7x7, 192, 7, 1)

    branch7x7dbl = conv2d_bn(x, 192, 1, 1)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 192, 7, 1)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 192, 1, 7)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 192, 7, 1)
    branch7x7dbl = conv2d_bn(branch7x7dbl, 192, 1, 7)

    branch_pool = layers.AveragePooling2D((3, 3),
                                          strides=(1, 1),
                                          padding='same')(x)
    branch_pool = conv2d_bn(branch_pool, 192, 1, 1)
    x = layers.concatenate(
        [branch1x1, branch7x7, branch7x7dbl, branch_pool],
        axis=channel_axis,
        name='mixed7')

    # mixed 8: 8 x 8 x 1280
    branch3x3 = conv2d_bn(x, 192, 1, 1)
    branch3x3 = conv2d_bn(branch3x3, 320, 3, 3,
                          strides=(2, 2), padding='valid')

    branch7x7x3 = conv2d_bn(x, 192, 1, 1)
    branch7x7x3 = conv2d_bn(branch7x7x3, 192, 1, 7)
    branch7x7x3 = conv2d_bn(branch7x7x3, 192, 7, 1)
    branch7x7x3 = conv2d_bn(
        branch7x7x3, 192, 3, 3, strides=(2, 2), padding='valid')

    branch_pool = layers.MaxPooling2D((3, 3), strides=(2, 2))(x)
    x = layers.concatenate(
        [branch3x3, branch7x7x3, branch_pool],
        axis=channel_axis,
        name='mixed8')

    # mixed 9: 8 x 8 x 2048
    for i in range(2):
        branch1x1 = conv2d_bn(x, 320, 1, 1)

        branch3x3 = conv2d_bn(x, 384, 1, 1)
```

```
        branch3x3_1 = conv2d_bn(branch3x3, 384, 1, 3)
        branch3x3_2 = conv2d_bn(branch3x3, 384, 3, 1)
        branch3x3 = layers.concatenate(
            [branch3x3_1, branch3x3_2],
            axis=channel_axis,
            name='mixed9_' + str(i))

        branch3x3dbl = conv2d_bn(x, 448, 1, 1)
        branch3x3dbl = conv2d_bn(branch3x3dbl, 384, 3, 3)
        branch3x3dbl_1 = conv2d_bn(branch3x3dbl, 384, 1, 3)
        branch3x3dbl_2 = conv2d_bn(branch3x3dbl, 384, 3, 1)
        branch3x3dbl = layers.concatenate(
            [branch3x3dbl_1, branch3x3dbl_2], axis=channel_axis)

        branch_pool = layers.AveragePooling2D(
            (3, 3), strides=(1, 1), padding='same')(x)
        branch_pool = conv2d_bn(branch_pool, 192, 1, 1)
        x = layers.concatenate(
            [branch1x1, branch3x3, branch3x3dbl, branch_pool],
            axis=channel_axis,
            name='mixed' + str(9 + i))
    if include_top:
        # Classification block
        x = layers.GlobalAveragePooling2D(name='avg_pool')(x)
        x = layers.Dense(classes, activation='softmax',
name='predictions')(x)
    else:
        if pooling == 'avg':
            x = layers.GlobalAveragePooling2D()(x)
        elif pooling == 'max':
            x = layers.GlobalMaxPooling2D()(x)

    # Ensure that the model takes into account
    # any potential predecessors of `input_tensor`.
    if input_tensor is not None:
        inputs = keras_utils.get_source_inputs(input_tensor)
    else:
        inputs = img_input
    # Create model.
    model = models.Model(inputs, x, name='inception_v3')

    # Load weights.
    if weights == 'imagenet':
        if include_top:
            weights_path = keras_utils.get_file(
                'inception_v3_weights_tf_dim_ordering_tf_kernels.h5',
                WEIGHTS_PATH,
                cache_subdir='models',
                file_hash='9a0d58056eeedaa3f26cb7ebd46da564')
        else:
            weights_path = keras_utils.get_file(

'inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5',
                WEIGHTS_PATH_NO_TOP,
                cache_subdir='models',
                file_hash='bcbd6486424b2319ff4ef7d526e38f63')
        model.load_weights(weights_path)
```

```
382     elif weights is not None:
383         model.load_weights(weights)
384 
385     return model
386 
387 
388 def preprocess_input(x, **kwargs):
389     """Preprocesses a numpy array encoding a batch of images.
390     # Arguments
391         x: a 4D numpy array consists of RGB values within [0, 255].
392     # Returns
393         Preprocessed array.
394     """
395     return imagenet_utils.preprocess_input(x, mode='tf', **kwargs)
```

https://github.com/keras-team/keras-applications/blob/master/keras_applications/xception.py

keras_applications/xception.py

```
001 """Xception V1 model for Keras.
002 On ImageNet, this model gets to a top-1 validation accuracy of 0.790
003 and a top-5 validation accuracy of 0.945.
004 Do note that the input image format for this model is different than for
005 the VGG16 and ResNet models (299x299 instead of 224x224),
006 and that the input preprocessing function
007 is also different (same as Inception V3).
008 # Reference
009 - [Xception: Deep Learning with Depthwise Separable Convolutions](
010     https://arxiv.org/abs/1610.02357) (CVPR 2017)
011 """
012 from __future__ import absolute_import
013 from __future__ import division
014 from __future__ import print_function
015 
016 import os
017 import warnings
018 
019 from . import get_submodules_from_kwargs
020 from . import imagenet_utils
021 from .imagenet_utils import decode_predictions
022 from .imagenet_utils import _obtain_input_shape
023 
024 
025 TF_WEIGHTS_PATH = (
026     'https://github.com/fchollet/deep-learning-models/'
027     'releases/download/v0.4/'
028     'xception_weights_tf_dim_ordering_tf_kernels.h5')
029 TF_WEIGHTS_PATH_NO_TOP = (
```

```
030     'https://github.com/fchollet/deep-learning-models/'
031     'releases/download/v0.4/'
032     'xception_weights_tf_dim_ordering_tf_kernels_notop.h5')
033
034
035 def Xception(include_top=True,
036              weights='imagenet',
037              input_tensor=None,
038              input_shape=None,
039              pooling=None,
040              classes=1000,
041              **kwargs):
042     """Instantiates the Xception architecture.
043     Optionally loads weights pre-trained on ImageNet.
044     Note that the data format convention used by the model is
045     the one specified in your Keras config at `~/.keras/keras.json`.
046     Note that the default input image size for this model is 299x299.
047     # Arguments
048         include_top: whether to include the fully-connected
049             layer at the top of the network.
050         weights: one of `None` (random initialization),
051               'imagenet' (pre-training on ImageNet),
052               or the path to the weights file to be loaded.
053         input_tensor: optional Keras tensor
054             (i.e. output of `layers.Input()`)
055             to use as image input for the model.
056         input_shape: optional shape tuple, only to be specified
057             if `include_top` is False (otherwise the input shape
058             has to be `(299, 299, 3)`.
059             It should have exactly 3 inputs channels,
060             and width and height should be no smaller than 71.
061             E.g. `(150, 150, 3)` would be one valid value.
062         pooling: Optional pooling mode for feature extraction
063             when `include_top` is `False`.
064             - `None` means that the output of the model will be
065                 the 4D tensor output of the
066                 last convolutional block.
067             - `avg` means that global average pooling
068                 will be applied to the output of the
069                 last convolutional block, and thus
070                 the output of the model will be a 2D tensor.
071             - `max` means that global max pooling will
072                 be applied.
073         classes: optional number of classes to classify images
074             into, only to be specified if `include_top` is True,
075             and if no `weights` argument is specified.
076     # Returns
077         A Keras model instance.
078     # Raises
079         ValueError: in case of invalid argument for `weights`,
080             or invalid input shape.
081         RuntimeError: If attempting to run this model with a
082             backend that does not support separable convolutions.
083     """
084     backend, layers, models, keras_utils =
get_submodules_from_kwargs(kwargs)
085
```

```
086     if not (weights in {'imagenet', None} or os.path.exists(weights)):
087         raise ValueError('The `weights` argument should be either '
088                          '`None` (random initialization), `imagenet` '
089                          '(pre-training on ImageNet), '
090                          'or the path to the weights file to be loaded.')
091 
092     if weights == 'imagenet' and include_top and classes != 1000:
093         raise ValueError('If using `weights` as `"imagenet"` with
`include_top`'
094                          ' as true, `classes` should be 1000')
095 
096     # Determine proper input shape
097     input_shape = _obtain_input_shape(input_shape,
098                                       default_size=299,
099                                       min_size=71,
100 
data_format=backend.image_data_format(),
101                                       require_flatten=include_top,
102                                       weights=weights)
103 
104     if input_tensor is None:
105         img_input = layers.Input(shape=input_shape)
106     else:
107         if not backend.is_keras_tensor(input_tensor):
108             img_input = layers.Input(tensor=input_tensor,
shape=input_shape)
109         else:
110             img_input = input_tensor
111 
112     channel_axis = 1 if backend.image_data_format() == 'channels_first'
else -1
113 
114     x = layers.Conv2D(32, (3, 3),
115                       strides=(2, 2),
116                       use_bias=False,
117                       name='block1_conv1')(img_input)
118     x = layers.BatchNormalization(axis=channel_axis,
name='block1_conv1_bn')(x)
119     x = layers.Activation('relu', name='block1_conv1_act')(x)
120     x = layers.Conv2D(64, (3, 3), use_bias=False, name='block1_conv2')(x)
121     x = layers.BatchNormalization(axis=channel_axis,
name='block1_conv2_bn')(x)
122     x = layers.Activation('relu', name='block1_conv2_act')(x)
123 
124     residual = layers.Conv2D(128, (1, 1),
125                              strides=(2, 2),
126                              padding='same',
127                              use_bias=False)(x)
128     residual = layers.BatchNormalization(axis=channel_axis)(residual)
129 
130     x = layers.SeparableConv2D(128, (3, 3),
131                                padding='same',
132                                use_bias=False,
133                                name='block2_sepconv1')(x)
134     x = layers.BatchNormalization(axis=channel_axis,
name='block2_sepconv1_bn')(x)
135     x = layers.Activation('relu', name='block2_sepconv2_act')(x)
```

```
136     x = layers.SeparableConv2D(128, (3, 3),
137                                padding='same',
138                                use_bias=False,
139                                name='block2_sepconv2')(x)
140     x = layers.BatchNormalization(axis=channel_axis,
name='block2_sepconv2_bn')(x)
141
142     x = layers.MaxPooling2D((3, 3),
143                             strides=(2, 2),
144                             padding='same',
145                             name='block2_pool')(x)
146     x = layers.add([x, residual])
147
148     residual = layers.Conv2D(256, (1, 1), strides=(2, 2),
149                              padding='same', use_bias=False)(x)
150     residual = layers.BatchNormalization(axis=channel_axis)(residual)
151
152     x = layers.Activation('relu', name='block3_sepconv1_act')(x)
153     x = layers.SeparableConv2D(256, (3, 3),
154                                padding='same',
155                                use_bias=False,
156                                name='block3_sepconv1')(x)
157     x = layers.BatchNormalization(axis=channel_axis,
name='block3_sepconv1_bn')(x)
158     x = layers.Activation('relu', name='block3_sepconv2_act')(x)
159     x = layers.SeparableConv2D(256, (3, 3),
160                                padding='same',
161                                use_bias=False,
162                                name='block3_sepconv2')(x)
163     x = layers.BatchNormalization(axis=channel_axis,
name='block3_sepconv2_bn')(x)
164
165     x = layers.MaxPooling2D((3, 3), strides=(2, 2),
166                             padding='same',
167                             name='block3_pool')(x)
168     x = layers.add([x, residual])
169
170     residual = layers.Conv2D(728, (1, 1),
171                              strides=(2, 2),
172                              padding='same',
173                              use_bias=False)(x)
174     residual = layers.BatchNormalization(axis=channel_axis)(residual)
175
176     x = layers.Activation('relu', name='block4_sepconv1_act')(x)
177     x = layers.SeparableConv2D(728, (3, 3),
178                                padding='same',
179                                use_bias=False,
180                                name='block4_sepconv1')(x)
181     x = layers.BatchNormalization(axis=channel_axis,
name='block4_sepconv1_bn')(x)
182     x = layers.Activation('relu', name='block4_sepconv2_act')(x)
183     x = layers.SeparableConv2D(728, (3, 3),
184                                padding='same',
185                                use_bias=False,
186                                name='block4_sepconv2')(x)
187     x = layers.BatchNormalization(axis=channel_axis,
name='block4_sepconv2_bn')(x)
```

```
188
189     x = layers.MaxPooling2D((3, 3), strides=(2, 2),
190                             padding='same',
191                             name='block4_pool')(x)
192     x = layers.add([x, residual])
193
194     for i in range(8):
195         residual = x
196         prefix = 'block' + str(i + 5)
197
198         x = layers.Activation('relu', name=prefix + '_sepconv1_act')(x)
199         x = layers.SeparableConv2D(728, (3, 3),
200                                    padding='same',
201                                    use_bias=False,
202                                    name=prefix + '_sepconv1')(x)
203         x = layers.BatchNormalization(axis=channel_axis,
204                                       name=prefix + '_sepconv1_bn')(x)
205         x = layers.Activation('relu', name=prefix + '_sepconv2_act')(x)
206         x = layers.SeparableConv2D(728, (3, 3),
207                                    padding='same',
208                                    use_bias=False,
209                                    name=prefix + '_sepconv2')(x)
210         x = layers.BatchNormalization(axis=channel_axis,
211                                       name=prefix + '_sepconv2_bn')(x)
212         x = layers.Activation('relu', name=prefix + '_sepconv3_act')(x)
213         x = layers.SeparableConv2D(728, (3, 3),
214                                    padding='same',
215                                    use_bias=False,
216                                    name=prefix + '_sepconv3')(x)
217         x = layers.BatchNormalization(axis=channel_axis,
218                                       name=prefix + '_sepconv3_bn')(x)
219
220         x = layers.add([x, residual])
221
222     residual = layers.Conv2D(1024, (1, 1), strides=(2, 2),
223                              padding='same', use_bias=False)(x)
224     residual = layers.BatchNormalization(axis=channel_axis)(residual)
225
226     x = layers.Activation('relu', name='block13_sepconv1_act')(x)
227     x = layers.SeparableConv2D(728, (3, 3),
228                                padding='same',
229                                use_bias=False,
230                                name='block13_sepconv1')(x)
231     x = layers.BatchNormalization(axis=channel_axis,
name='block13_sepconv1_bn')(x)
232     x = layers.Activation('relu', name='block13_sepconv2_act')(x)
233     x = layers.SeparableConv2D(1024, (3, 3),
234                                padding='same',
235                                use_bias=False,
236                                name='block13_sepconv2')(x)
237     x = layers.BatchNormalization(axis=channel_axis,
name='block13_sepconv2_bn')(x)
238
239     x = layers.MaxPooling2D((3, 3),
240                             strides=(2, 2),
241                             padding='same',
242                             name='block13_pool')(x)
```

```
    x = layers.add([x, residual])

    x = layers.SeparableConv2D(1536, (3, 3),
                               padding='same',
                               use_bias=False,
                               name='block14_sepconv1')(x)
    x = layers.BatchNormalization(axis=channel_axis,
name='block14_sepconv1_bn')(x)
    x = layers.Activation('relu', name='block14_sepconv1_act')(x)

    x = layers.SeparableConv2D(2048, (3, 3),
                               padding='same',
                               use_bias=False,
                               name='block14_sepconv2')(x)
    x = layers.BatchNormalization(axis=channel_axis,
name='block14_sepconv2_bn')(x)
    x = layers.Activation('relu', name='block14_sepconv2_act')(x)

    if include_top:
        x = layers.GlobalAveragePooling2D(name='avg_pool')(x)
        x = layers.Dense(classes, activation='softmax',
name='predictions')(x)
    else:
        if pooling == 'avg':
            x = layers.GlobalAveragePooling2D()(x)
        elif pooling == 'max':
            x = layers.GlobalMaxPooling2D()(x)

    # Ensure that the model takes into account
    # any potential predecessors of `input_tensor`.
    if input_tensor is not None:
        inputs = keras_utils.get_source_inputs(input_tensor)
    else:
        inputs = img_input
    # Create model.
    model = models.Model(inputs, x, name='xception')

    # Load weights.
    if weights == 'imagenet':
        if include_top:
            weights_path = keras_utils.get_file(
                'xception_weights_tf_dim_ordering_tf_kernels.h5',
                TF_WEIGHTS_PATH,
                cache_subdir='models',
                file_hash='0a58e3b7378bc2990ea3b43d5981f1f6')
        else:
            weights_path = keras_utils.get_file(
                'xception_weights_tf_dim_ordering_tf_kernels_notop.h5',
                TF_WEIGHTS_PATH_NO_TOP,
                cache_subdir='models',
                file_hash='b0042744bf5b25fce3cb969f33bebb97')
        model.load_weights(weights_path)
        if backend.backend() == 'theano':
            keras_utils.convert_all_kernels_in_model(model)
    elif weights is not None:
        model.load_weights(weights)

```

```
297     return model
298 
299 
300 def preprocess_input(x, **kwargs):
301     """Preprocesses a numpy array encoding a batch of images.
302     # Arguments
303         x: a 4D numpy array consists of RGB values within [0, 255].
304     # Returns
305         Preprocessed array.
306     """
307     return imagenet_utils.preprocess_input(x, mode='tf', **kwargs)
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://en.wikipedia.org/wiki/Artificial_intelligence

[2]https://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%89%CE%BD%CE%B9%CE%BA%CF%8C_%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

[3] https://el.wikipedia.org/wiki/Μηχανική_μάθηση

[4]https://el.wikipedia.org/wiki/%CE%95%CE%BD%CE%B9%CF%83%CF%87%CF%85%CF%84%CE%B9%CE%BA%CE%AE_%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7

[5] https://el.wikipedia.org/wiki/%CE%9C%CE%B7-%CE%B5%CF%80%CE%B9%CE%B2%CE%BB%CE%B5%CF%80%CF%8C%CE%BC%CE%B5%CE%BD%CE%B7_%CE%9C%CE%AC%CE%B8%CE%B7%CF%83%CE%B7

[6]https://el.wikipedia.org/wiki/%CE%A4%CE%B5%CF%87%CE%BD%CE%B7%CF%84%CE%AE_%CE%BD%CE%BF%CE%B7%CE%BC%CE%BF%CF%83%CF%8D%CE%BD%CE%B7

[7] https://en.wikipedia.org/wiki/Linear_regression

[8] https://en.wikipedia.org/wiki/Perceptron

[9] https://en.wikipedia.org/wiki/Self-driving_car

[10] https://pythonprogramming.net/machine-learning-tutorials/

[11] https://python-mss.readthedocs.io/examples.html

[12] http://aibook.csd.auth.gr/include/slides/Chap01.pdf

[13] http://www.image-net.org/about-overview

[14] https://cloud.google.com/tpu/docs/inception-v3-advanced

[15] https://nuro.ai/about

[16] https://www.tensorflow.org/

[17] https://www.unity3d.com

[18] https://keras.io/

[19] https://arxiv.org/pdf/1512.00567v3.pdf

[20] https://arxiv.org/pdf/1610.02357.pdf

[21] https://en.wikipedia.org/wiki/Recurrent_neural_network#LSTM

[22] https://en.wikipedia.org/wiki/Turing_test

Εικόνες

[1] Εικόνα 1.1 https://www.illuminateed.com/blog/2018/05/will-machine-learning-change-the-path-of-k-12-education/

[2] Εικόνα 3.1 https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html

[3] Εικόνα 3.2 https://pythonmachinelearning.pro/perceptrons-the-first-neural-networks/

[4] Εικόνα 3.3 By Elizabeth Goodspeed - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=40188333

[5] Εικόνα 3.6 By Michaelg2015 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=42442194¨

[6] Εικόνα 3.12 https://hackernoon.com/rnn-or-recurrent-neural-network-for-noobs-a9afbb00e860

[7] Εικόνα 3.13 και 3.15 https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[8] 3.18 Εικόνα Softmax συνάρτησης http://cs231n.github.io/linear-classify/#softmax

[9] Εικόνες 3.4, 3.5, 3.7- 3.11 http://repfiles.kallipos.gr/html_books/93/04a-main.html

[10] Εικόνα 5.3-5.6 https://arxiv.org/pdf/1512.00567v3.pdf

[10] Εικόνα 5.7 https://cloud.google.com/tpu/docs/inception-v3-advanced

[11] Εικόνα 5.8-5.9 https://arxiv.org/pdf/1610.02357.pdf