ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Δημιουργία αλγορίθμου αυτόνομης οδήγησης οχήματος σε περιβάλλον προσομοίωσης με τη χρήση μεθόδων μηχανικής μάθησης»

Του φοιτητή
Ναβροζίδη Δημήτριου
Αρ. Μητρώου: 516095

Επιβλέπων
Γουλιάνας Κωνσταντίνος
Αναπληρωτής Καθηγητής

Ημερομηνία 20/01/2022

Τίτλος Δ.Ε. Δημιουργία αλγορίθμου αυτόνομης οδήγησης οχήματος σε περιβάλλον προσομοίωσης με τη χρήση μεθόδων μηχανικής μάθησης

Κωδικός Δ.Ε. 22202

Ονοματεπώνυμο φοιτητή Ναβροζίδης Δημήτριος

Ονοματεπώνυμο εισηγητή Γουλιάνας Κωνσταντίνος

Ημερομηνία ανάληψης Δ.Ε. 30-03- 2022

Ημερομηνία περάτωσης Δ.Ε. 20/01/2023

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Αφιέρωση»

Πρόλογος

Από όλα τα θέματα που είχα την ευκαιρία να γνωρίσω φοιτώντας στο τμήμα αυτά τα 5 χρόνια, αυτό που μου κέντρισε το ενδιαφέρον περισσότερο ήταν η μηχανική μάθηση. Η ιδέα ότι ένας αλγόριθμος μπορεί να μαθαίνει και στη συνέχεια να λύνει προβλήματα, ανταγωνίζοντας και συχνά κερδίζοντας τον άνθρωπο, μου φάνηκε συναρπαστική. Αυτό που μου κάνει όμως περισσότερο εντύπωση, είναι το πώς αυτός ο αλγόριθμος έχει τη δυνατότητα να μας εκπλήσσει με τα αποτελέσματα του, άλλοτε ξεπερνώντας τις προσδοκίες και άλλοτε αποτυγχάνοντας πλήρως. Αυτή η απροβλεψιμότητα και το μυστήριο που δημιουργεί, είναι που κατά τη γνώμη μου κάνει τη μηχανική μάθηση τόσο ενδιαφέρουσα. Η συγκεκριμένη εργασία δεν έχει κάποιο πραγματικό όφελος και δεν πετυχαίνει κάτι πρακτικά χρήσιμο, είναι απλά ένα πείραμα και μια επίδειξη του τι μπορεί να πετύχει κανείς χρησιμοποιώντας μερικά διαθέσιμα προς όλους εργαλεία και μερικές βασικές γνώσεις προγραμματισμού και μηχανικής μάθησης.

Περίληψη

Η αυτόνομη οδήγηση είναι και θα συνεχίσει να είναι, στο κοντινό μέλλον, ένα καυτό θέμα. Τα περισσότερα συστήματα αυτόματης οδήγησης σήμερα χρησιμοποιούν έναν συνδυασμό διαφορετικών αισθητήρων για να επιτύχουν τον στόχο τους και ένας από αυτούς τους αισθητήρες είναι η κάμερα. Η παρούσα διπλωματική εργασία επιχειρεί να δημιουργήσει έναν απλό αλγόριθμο αυτόνομης οδήγησης σε περιβάλλον προσομοίωσης (βιντεοπαιχνίδι), χρησιμοποιώντας την κάμερα ως μοναδικό αισθητήρα. Ο αλγόριθμος εκμεταλλεύεται 3 διαφορετικούς τύπους βαθιών συνελικτικών νευρωνικών δικτύων για την εξαγωγή χρήσιμων πληροφοριών από τα δεδομένα της κάμερας, τα οποία αξιοποιεί χρησιμοποιώντας απλή προγραμματιστική λογική. Το αποτέλεσμα είναι ένας αλγόριθμος που μπορεί να οδηγήσει με επιτυχία υπό ορισμένες συνθήκες (δρόμοι με μία λωρίδα κυκλοφορίας, γραμμές οδοστρώματος, απουσία άλλων αυτοκινήτων και αντικειμένων), σε διάφορους δημόσιους δρόμους που παρέχει το παιχνίδι. Τέλος, δημιουργείται μια γραφική διεπαφή χρήστη που ομαδοποιεί κάθε βήμα που οδηγεί στη δημιουργία του αλγορίθμου αυτόματης οδήγησης, καθιστώντας την όλη διαδικασία δημιουργίας συνόλων δεδομένων, εκπαίδευσης μοντέλων και δοκιμής αλγορίθμων ταχύτερη και πιο ευχάριστη.

«Creation of an autonomous vehicle driving algorithm in a simulation environment using machine learning methods»

«Dimitrios Navrozidis»

Abstract

Self driving cars are and will continue to be, in the foreseeable future, a hot topic. Most self driving systems now days use a combination of different sensors to achieve their goal, one of these sensors being the camera. This thesis attempts to create a simple self driving algorithm in a simulation environment (videogame), using the camera as its only sensor. The algorithm takes advantage of 3 different types of deep convolutional neural networks to extract useful information from the camera data and makes use of them with simple programming logic. The result is an algorithm that can successfully drive under some specific conditions (single lane roads, road lines, absence of other cars and objects), in various public roads that the game provides. Finally a graphical user interface is created that groups every step that leads to the creation of the self driving algorithm, making the whole process of creating datasets, training models, and testing algortithms faster and more pleasant.

Ευχαριστίες

Ευχαριστώ την οικογένεια μου και τους φίλους μου, που με βοήθησαν καθ’ όλη τη διάρκεια του μαθησιακού μου ταξιδίου όλα αυτά τα χρόνια. Ευχαριστώ τους καθηγητές του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων για τις γνώσεις και την εμπειρία που μου μετέδωσαν κατά τα χρόνια της φοίτησης μου. Τέλος ευχαριστώ τον Youtuber Sentdex, δημιουργό του πρότζεκτ pygta5, από τον οποίο είναι εμπνευσμένο το θέμα της εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Τ.Ν.Δ	Τεχνητά Νευρωνικά Δίκτυα
Ν.Δ.	Νευρωνικά Δίκτυα
Σ.Δ.	Συνελικτικά Δίκτυα
CNN	Convolutional Neural Networks
GD	Gradient Decent
AC	Assetto Corsa

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Το πρόβλημα της αυτόνομης οδήγησης στον πραγματικό κόσμο είναι δύσκολο και περίπλοκο. Το περιβάλλον στους δρόμους είναι απρόβλεπτο και συχνά χαώδες. Οι παράγοντες που πρέπει να λάβει υπόψη ένα τέτοιο σύστημα είναι πολλοί, γραμμές στο οδόστρωμα, κατάσταση οδοστρώματος, πινακίδες, άλλα οχήματα, πεζοί και άλλα αντικείμενα και καταστάσεις που δεν μπορούν να προβλεφτούν. Ένα τόσο πολύπλοκο πρόβλημα προφανώς έχει εξίσου πολύπλοκη λύση, γι' αυτό και τα συστήματα αυτά στον πραγματικό κόσμο χρησιμοποιούν πολλά διαφορετικά εργαλεία και μεθόδους για να πετυχαίνουν το στόχο τους και μάλιστα με αυστηρή ακρίβεια και ασφάλεια, μιας και δεν υπάρχουν περιθώρια για λάθη όταν εμπλέκονται ανθρώπινες ζωές.

Η συγκεκριμένη εργασία πρακτικά δεν έχει σχεδόν καμία σχέση με το πραγματικό πρόβλημα της αυτόνομης οδήγησης. Πρόκειται ίσως για ένα πολύ μικρό υποσύνολο του που έχει να κάνει με τη χρήση καμερών για τη λήψη εικόνας του περίγυρου του οχήματος και στη συνέχεια την εφαρμογή αλγορίθμων μηχανικής μάθησης για την εξαγωγή συμπερασμάτων σχετικά με τη θέση του οχήματος στο χώρο και την αναγνώριση άλλων αντικειμένων, καθώς και για τη συμβολή στη λήψη αποφάσεων. Είναι σημαντικό να τονίσουμε ότι η αποφάσεις για τις ενέργειες του οχήματος δεν βασίζονται ποτέ μονάχα σε εικόνα, όπως επίσης δεν βασίζονται ποτέ αποκλειστικά στα συμπεράσματα αλγορίθμων μηχανικής μάθησης. Ο λόγος είναι πρώτα ότι η εικόνα ανάλογα και με τις συνθήκες, οι οποίες είναι πάντα μεταβαλλόμενες, π.χ. κατάσταση οδοστρώματος και σήμανσης, καιρικές συνθήκες, πιθανές βλάβες εξοπλισμού, μπορεί να μην παρέχει αρκετή και έμπιστη πληροφορία για τη λήψη μιας σωστής και ασφαλούς απόφασης. Όσο αφορά τις μεθόδους μηχανικής μάθησης, ενώ έχουν αδιαμφισβήτητα εκπληκτικές δυνατότητες, ένα από τα ελαττώματα τους είναι ότι συχνά είναι απρόβλεπτες και ασταθείς. Είναι συχνό φαινόμενο τα αποτελέσματα ενός φαινομενικά «καλού» αλγορίθμου να είναι λανθασμένα, χωρίς να μπορούμε να γνωρίζουμε την αιτία. Η διαρκώς αυξανόμενη πολυπλοκότητα και ο αυξανόμενος αριθμός των παραμέτρων των νευρωνικών δικτύων, που όπως θα αναλυθεί παρακάτω αποτελούν το σημαντικότερο συστατικό της μηχανικής μάθησης, έχουν σαν αποτέλεσμα να αντιμετωπίζουμε τους αλγόριθμους αυτούς σαν μαύρα κουτιά, τις εξόδους των οποίων πολλές φορές δεν μπορούμε να εξηγήσουμε. Για τους παραπάνω λόγους η αποφάσεις που λαμβάνει ένα σύστημα αυτόνομης οδήγησης είναι πάντα αποτέλεσμα συνδυασμού και αλληλοεπαλήθευσης πολλών διαφορετικών μεθόδων και πηγών πληροφορίας.[1]

1.2 Στόχος Εργασίας

Ποιος είναι λοιπόν ο στόχος αυτής της εργασίας;

Ο στόχος της εργασίας είναι η δημιουργία ενός αλγορίθμου ο οποίος χρησιμοποιώντας μηχανική μάθηση θα ελέγχει ένα εικονικό όχημα σε έναν εικονικό κόσμο, όπου οι συνθήκες είναι σταθερές και το περιβάλλον πολύ πιο απλό από αυτό του πραγματικού κόσμου. Ο κόσμος αυτός είναι ένα βιντεοπαιχνίδι προσομοίωσης αγώνων μηχανοκίνητου αθλητισμού και γενικότερα προσομοίωσης οδήγησης. Ο αλγόριθμος αυτός θα αντικαθιστά τον άνθρωπο και θα χειρίζεται το όχημα «βλέποντας» ότι θα έβλεπε και ο άνθρωπος και έχοντας τις ίδιες ακριβώς δυνατότητες χειρισμού. Στη γλώσσα της τεχνητής νοημοσύνης πρόκειται για έναν ευφυή πράκτορα. Ο ορισμός του τι είναι ένας ευφυής πράκτορας δεν είναι ξεκάθαρος, μπορεί κανείς να βρει πολλές διαφορετικές παραλλαγές. Στα πλαίσια

της συγκεκριμένης εργασίας, ορίζουμε ως ευφυή πράκτορα οτιδήποτε αντιλαμβάνεται το περιβάλλον μέσω αισθητήρων, κάνει συλλογισμούς, αποφασίζει και δρα σε αυτό μέσα από μηχανισμούς δράσης, για την επίτευξη κάποιον στόχων [2]. Για την εφαρμογή της εργασίας οι αισθητήρες που αντιλαμβάνονται το περιβάλλον και αντικαθιστούν την όραση του ανθρώπου, είναι ενός τύπου υπολογιστική όραση μέσω Νευρωνικών Δικτύων. Οι συλλογισμοί και οι δράσεις γίνονται μέσω απλής προγραμματιστικής λογικής. Ενώ οι μηχανισμοί δράσης είναι εντολές που ελέγχουν τα τρία βασικά όργανα ελέγχου ενός αυτοκινήτου (τιμόνι, γκάζι, φρένο) μέσω ενός εικονικού χειριστηρίου του παιχνιδιού, το οποίο αντικαθιστά το πραγματικό χειριστήριο που χρησιμοποιεί ένας παίχτης για να οδηγήσει το αυτοκίνητο.

Κεφάλαιο 2ο: Θεωρία

2.1 Τεχνητή Νοημοσύνη

Ανά τα χρόνια έχουν δοθεί αρκετοί διαφορετικοί ορισμοί για την έννοια της τεχνικής νοημοσύνης. Ο ορισμός που δίνεται στο βιβλίο [2] επιχειρεί να συμπεριλάβει τα περισσότερα στοιχεία από διαφορετικούς ορισμούς και είναι ο εξής.

«Τεχνική Νοημοσύνη είναι ο τομέας της επιστήμης των υπολογιστών που ασχολείται με τη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων τα οποία είναι ικανά να μιμηθούν τις ανθρώπινες γνωστικές ικανότητες, εμφανίζοντας έτσι χαρακτηριστικά που αποδίδουμε συνήθως σε ανθρώπινη συμπεριφορά, όπως για παράδειγμα η επίλυση προβλημάτων, η αντίληψη και κατανόηση εικόνων, η μάθηση, η εξαγωγή συμπερασμάτων, η κατανόηση φυσικής γλώσσας, κτλ.»

Γενικά η ΤΝ προσπαθεί να πετύχει το στόχο της, να μιμηθεί δηλαδή τον τρόπο που «δουλεύει» ο άνθρωπος, με περισσότερες από μία προσεγγίσεις. Αυτό έχει ως αποτέλεσμα να δημιουργούνται υποσύνολα της επιστήμης αυτής. Ένα παράδειγμα διαχωρισμού και πάλι σύμφωνα με το [2] είναι σε:

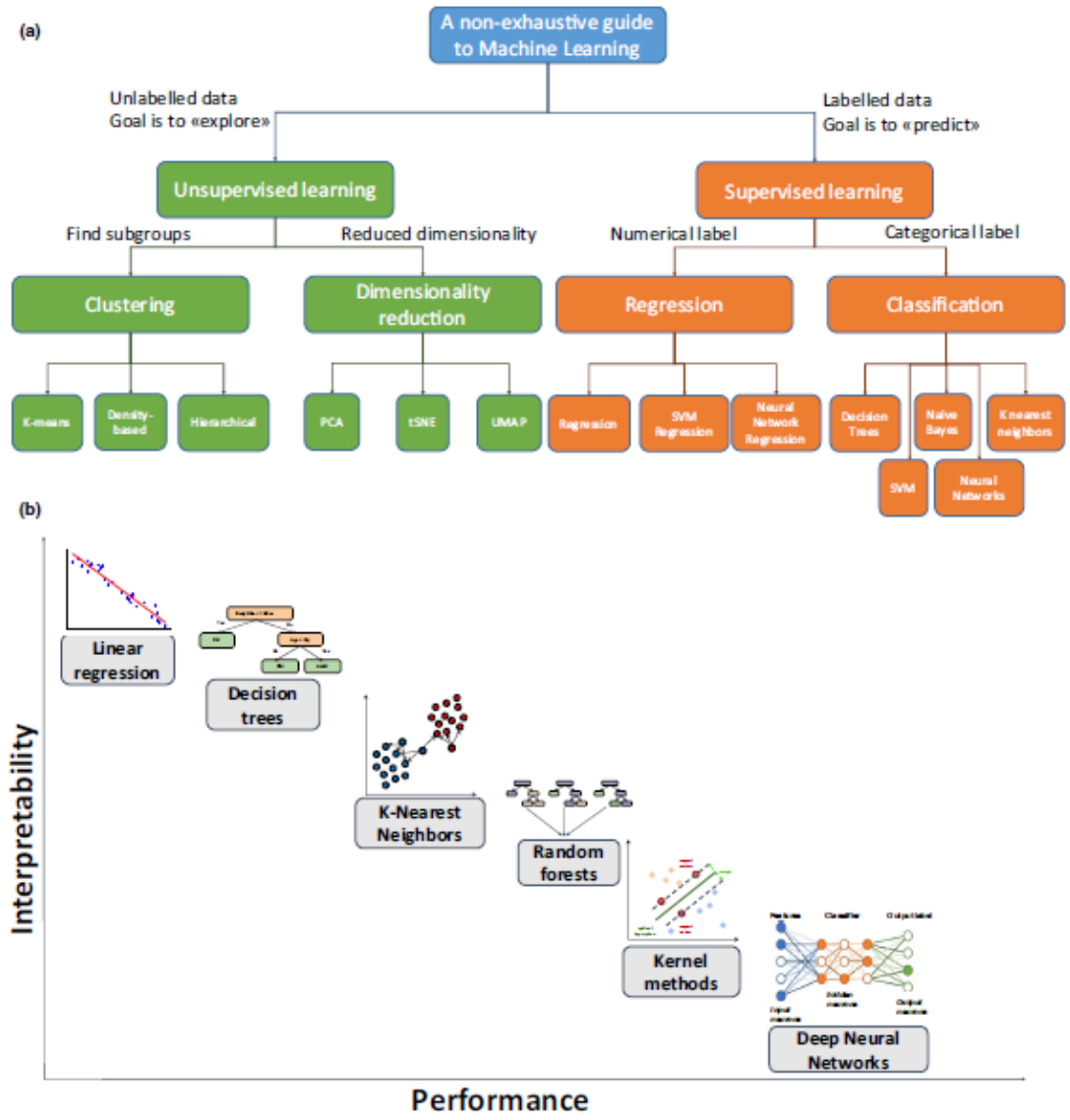
- Κλασική ή συμβολική τεχνητή νοημοσύνη η οποία προσομοιώνει την ανθρώπινη νοημοσύνη προσεγγίζοντας την με αλγόριθμους και συστήματα που βασίζονται στη γνώση, χρησιμοποιώντας σύμβολα ως δομικές μονάδες τα οποία αναπαριστούν έννοιες και σχέσεις ανάμεσα τους. Παραδείγματα αυτής της προσέγγισης είναι εφαρμογές που χρησιμοποιούν λογική, κανόνες, πλαίσια κτλ.
- Υπολογιστική νοημοσύνη η οποία βασίζεται στη μίμηση της διαδικασίας εξέλιξης των ειδών ή της λειτουργίας του εγκεφάλου. Δύο παραδείγματα είναι οι γενετικοί αλγόριθμοι και τα νευρωνικά δίκτυα.

2.2 Μηχανική Μάθηση

Η μηχανική μάθηση είναι ένα πεδίο της επιστήμης των υπολογιστών που μελετά αλγόριθμους και τεχνικές για την αυτοματοποίηση λύσεων σε πολύπλοκα προβλήματα που είναι δύσκολο να προγραμματιστούν χρησιμοποιώντας συμβατικές μεθόδους προγραμματισμού. Η συμβατική μέθοδος προγραμματισμού αποτελείται από δύο διακριτά βήματα. Δεδομένων κάποιων προδιαγραφών για το πρόγραμμα σχετικά με το τι πρέπει να κάνει και όχι το πως, το πρώτο βήμα είναι να δημιουργηθεί ένα συγκεκριμένο σχέδιο για το πρόγραμμα, δηλαδή ένα προκαθορισμένο σύνολο βημάτων ή κανόνων για την επίλυση του προβλήματος. Το δεύτερο βήμα είναι η υλοποίηση του σχεδίου ως πρόγραμμα με τη χρήση κάποιας γλώσσας προγραμματισμού.

Αυτή η προσέγγιση μπορεί να μην είναι αποτελεσματική για πολλά προβλήματα του πραγματικού κόσμου στα οποία η δημιουργία λεπτομερούς σχεδίου μπορεί να είναι αρκετά δύσκολη παρά τις σαφείς προδιαγραφές. Ένα τέτοιο παράδειγμα είναι η ανίχνευση χειρόγραφων χαρακτήρων σε μια εικόνα όπου μας δίνεται ένα σύνολο δεδομένων (dataset) που αποτελείται από μεγάλο αριθμό εικόνων χειρόγραφων χαρακτήρων. Επιπλέον, τα σημεία δεδομένων (datapoints), δηλαδή οι εικόνες, συνοδεύονται από ετικέτες οι οποίες δείχνουν των χαρακτήρα που περιέχουν. Αυτό το σύνολο δεδομένων μαζί με τις ετικέτες είναι ουσιαστικά ένα σύνολο παραδειγμάτων που περιγράφουν πώς πρέπει να συμπεριφέρεται το πρόγραμμα. Ο στόχος είναι να καταλήξουμε σε ένα πρόγραμμα που μπορεί να αναγνωρίσει τους χαρακτήρες σε οποιαδήποτε εικόνα και όχι μόνο αυτούς στο σύνολο δεδομένων. Με μια συμβατική μέθοδο, θα μελετούσαμε πρώτα τα παραδείγματα στο σύνολο των δεδομένων, προσπαθώντας να κατανοήσουμε πώς οι εικόνες αντιστοιχούν σε χαρακτήρες και στη

συνέχεια θα προσπαθούσαμε να βρούμε ένα γενικό σύνολο κανόνων για τον εντοπισμό χαρακτήρων σε οποιαδήποτε εικόνα. Η δημιουργία αυτού του συνόλου κανόνων είναι εξαιρετικά δύσκολη, δεδομένης της μεγάλης ποικιλίας στον τρόπο που μπορούν να γραφούν οι χειρόγραφοι χαρακτήρες.



Σχήμα 2.1: α) Είδη Μηχανικής Μάθησης β) Σχέση Διαλειτουργικότητας και Απόδοσης Μοντέλων Μηχανικής Μάθησης [3]

Στο σχήμα 2.1α βλέπουμε τους διάφορους τύπους αλγορίθμων μηχανικής μάθησης και το πως αυτοί ταξινομούνται ανάλογα με τα χαρακτηριστικά τους. Επιπλέον στο 2.1β μερικοί από τους αλγορίθμους αυτούς συγκρίνονται ως προς τη διαλειτουργικότητα και την απόδοση τους. Όσο οι μέθοδοι γίνονται πιο περίπλοκοι και αποδοτικοί, τόσο το κόστος της διαλειτουργικότητας μεγαλώνει.

2.2.1 Μάθηση Χωρίς Επίβλεψη

Η μάθηση χωρίς επίβλεψη είναι η εκπαίδευση μιας μηχανής που χρησιμοποιεί πληροφορίες μη ταξινομημένες και χωρίς ετικέτες, επιτρέποντας στον αλγόριθμο να ενεργεί σε αυτές τις πληροφορίες χωρίς καθοδήγηση. Σκοπός είναι να ομαδοποιηθούν οι μη ταξινομημένες πληροφορίες σύμφωνα με ομοιότητες, μοτίβα και διαφορές χωρίς προηγούμενη εκπαίδευση δεδομένων. Η μάθηση χωρίς επίβλεψη χωρίζεται σε δύο κατηγορίες:

- **Ομαδοποίηση (clustering):** Η ομαδοποίηση είναι μια τεχνική που ομαδοποιεί δεδομένα χωρίς ετικέτα με βάση τις ομοιότητες ή τις διαφορές τους. Οι αλγόριθμοι ομαδοποίησης χρησιμοποιούνται για την επεξεργασία ακατέργαστων, μη ταξινομημένων δεδομένων σε ομάδες που αντιπροσωπεύονται από δομές ή μοτίβα στις πληροφορίες. Οι αλγόριθμοι ομαδοποίησης μπορούν να κατηγοριοποιηθούν σε τύπους, συγκεκριμένα αποκλειστικούς (specifically exclusive), επικαλυπτόμενους (overlapping), ιεραρχικούς (hierarchical) και πιθανολογικούς (probabilistic). [4]
- **Κανόνες Συσχετίσεων (Association Rules):** Οι κανόνες συσχέτισης χρησιμοποιούνται για την εύρεση συσχετίσεων και συν-εμφανίσεων μεταξύ συνόλων δεδομένων. Οι συσχετίσεις συνήθως αναπαρίστανται με τη μορφή κανόνων ή συχνών συνόλων στοιχείων (frequent itemsets). Ένα παράδειγμα είναι οι αλγόριθμοι μείωσης διαστάσεων. Η μείωση διαστάσεων είναι ο μετασχηματισμός δεδομένων από έναν χώρο υψηλής διάστασης σε έναν χώρο χαμηλής διάστασης, έτσι ώστε η αναπαράσταση χαμηλής διάστασης να διατηρεί κάποιες σημαντικές ιδιότητες των αρχικών δεδομένων, ιδανικά κοντά στην εγγενή τους διάσταση. Η εργασία στις αρχικές διαστάσεις μπορεί να είναι ανεπιθύμητη για πολλούς λόγους, καθώς τα ακατέργαστα δεδομένα είναι συχνά αραιά (sparse) ως συνέπεια της αυξημένης διαστατικότητας και η ανάλυση των δεδομένων είναι συνήθως υπολογιστικά δύσκολη.

2.2.2 Μάθηση με Επίβλεψη

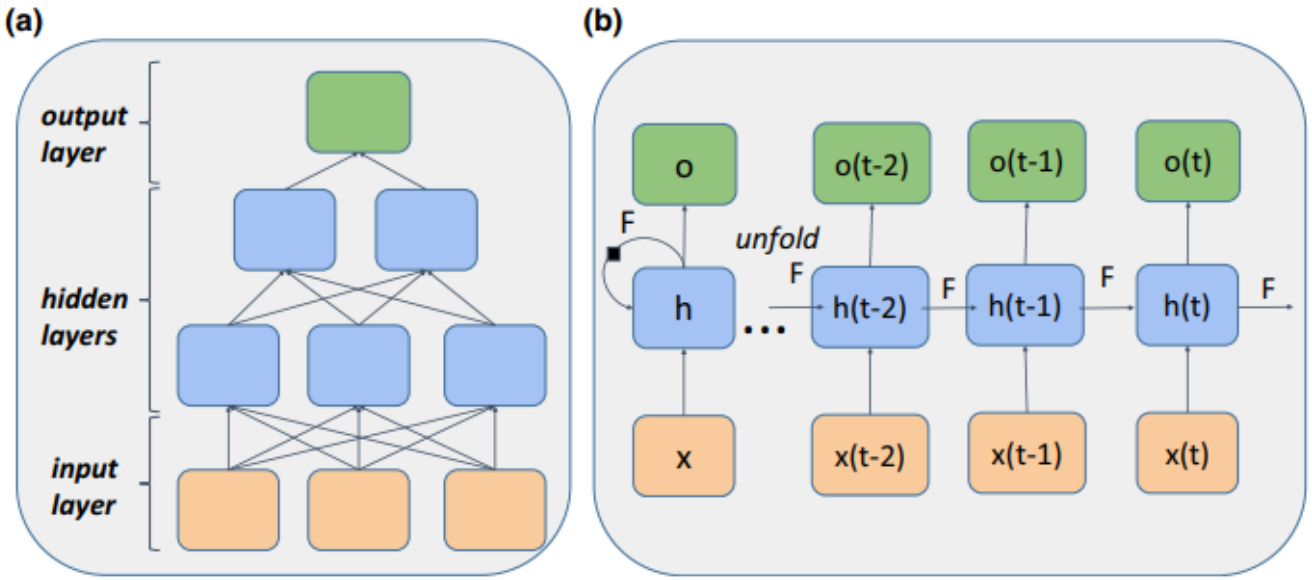
Η μάθηση με επίβλεψη αντίθετα από τη μάθηση χωρίς επίβλεψη χρησιμοποιεί δεδομένα με ετικέτες για την εκπαίδευση αλγορίθμων. Καθώς τα δεδομένα εισόδου τροφοδοτούνται στο μοντέλο, αυτό ανιχνεύει μοτίβα και σχέσεις μεταξύ των δεδομένων εισόδου και των αντίστοιχων ετικετών τους, προσαρμόζοντας τα βάρη του ανάλογα. Το εκπαιδευμένο μοντέλο έπειτα έχει τη δυνατότητα να παράγει σωστά αποτελέσματα, να προβλέπει δηλαδή ετικέτες για καινούρια δεδομένα στα οποία δεν εκπαιδεύτηκε ποτέ. Τα προβλήματα που λύνει η μάθηση με επίβλεψη χωρίζονται σε:

- Ταξινόμηση (classification): Ο σκοπός ενός αλγορίθμου ταξινόμησης είναι η ταξινόμηση των εισερχόμενων δεδομένων σε ένα συγκεκριμένο αριθμό κλάσεων, με βάση τις ετικέτες των δεδομένων που εκπαιδεύτηκε. Μπορούν να χρησιμοποιηθούν για δυαδικές ταξινομήσεις, όπως για παράδειγμα το φιλτράρισμα ηλεκτρονικού ταχυδρομείου σε spam και μη spam ή την κατηγοριοποίηση σχολίων πελατών σε θετικά ή αρνητικά, καθώς και για ταξινομήσεις πολλαπλών κλάσεων, αναγνώριση χειρόγραφων γραμμάτων και αριθμών, αναγνώριση αντικειμένων κλπ.
- Παλινδρόμηση (regression): Οι αλγόριθμοι παλινδρόμησης με βάση κάποιων δεδομένων εισόδου παράγουν (προβλέπουν) μια αριθμητική έξοδο. Πέρα από τη μορφή της εξόδου η "αρχή λειτουργίας" είναι ίδια με αυτήν των αλγορίθμων ταξινόμησης. Μερικά παραδείγματα

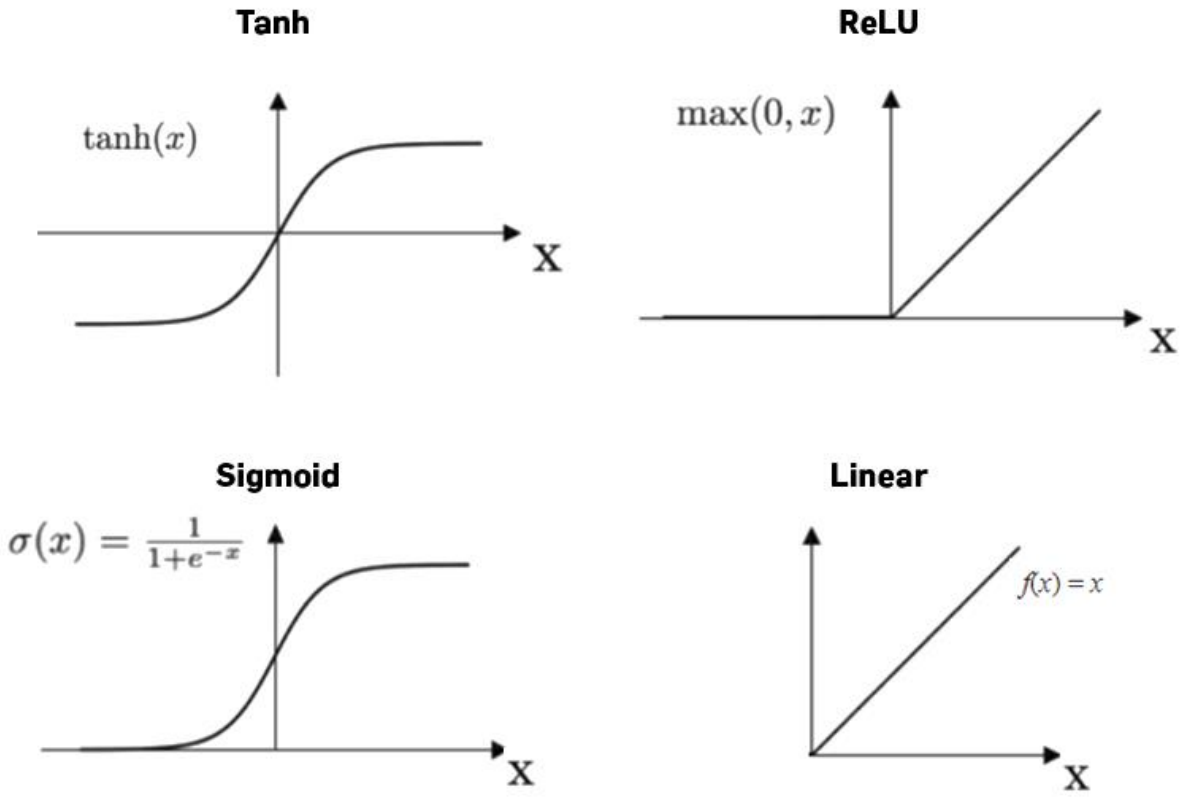
παλινδρόμησης είναι η πρόβλεψη τιμών ακινήτων, πρόβλεψη καιρικών συνθηκών (θερμοκρασία, υγρασία, βροχή, κλπ).

2.2.3 Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα αποτελούν μια συλλογή νευρώνων και ακμών (συνάψεων), σχήμα 2.2α. Σε κάθε ακμή που συνδέει νευρώνες αντιστοιχίζεται μία παράμετρος που ονομάζουμε βάρος. Σε κάθε νευρώνα εφαρμόζεται συνάρτηση ενεργοποίησης η οποία δέχεται σήμα εισόδου επηρεασμένο από τα βάρη και παράγει ένα σήμα εξόδου. Παραδείγματα συχνά χρησιμοποιούμενων συναρτήσεων ενεργοποίησης είναι η υπερβολική εφαπτομένη (tanh), η rectified linear unit (ReLu), η Σιγμοειδής συνάρτηση (Sigmoid) και η γραμμική συνάρτηση (linear). Στο σχήμα 2.3 φαίνονται οι γραφικές τους παραστάσεις. Οι συναρτήσεις ενεργοποίησης επηρεάζουν τον τρόπο με τον οποίο το σήμα μεταδίδεται από το ένα στρώμα στο άλλο.



Σχήμα 2.2: α) Νευρωνικό δίκτυο δύο κρυφών στρωμάτων β) Γενικευμένη αρχιτεκτονική νευρωνικών δικτύων



Σχήμα 2.3: Συναρτήσεις Ενεργοποίησης: Υπερβολική Εφαπτομένη, Rectified Linear Unit, Σιγμοειδής, Γραμμική. [5]

Οι νευρώνες ομαδοποιούνται και χωρίζονται σε στρώμα εισόδου (input layer), κρυφά στρώματα (hidden layers) και στρώμα εξόδου (output layer), σχήμα 2.2α. Τα κρυφά στρώματα εκτελούν το επίπεδο αφαίρεσης που απαιτείται για τη μετάβαση από το επίπεδο εισόδου στο στρώμα εξόδου. Ο αριθμός των κρυφών στρωμάτων καθορίζει αν το σύστημα είναι ρηχής μάθησης , με ένα ή λίγα κρυφά στρώματα, ή βαθιάς μάθησης, με πολλά κρυφά στρώματα. Υπάρχει ένας συμβιβασμός μεταξύ του αριθμού των κρυφών στρωμάτων και του χρόνου που απαιτείται για την εκπαίδευση του μοντέλου. Για αυτόν τον λόγο αν και η ιδέα των νευρωνικών δικτύων δεν είναι καινούργια, η ανάπτυξη τους και η χρήση τους εντοπίζεται κυρίως τα τελευταία χρόνια λόγω των πρόσφατων εξελίξεων στην υπολογιστική ισχύ. Ο πιο συνηθισμένος τύπος είναι τα νευρωνικά τροφοδότησης προς τα εμπρός, όπου η πληροφορία διαδίδεται από το στρώμα εισόδου στα κρυφά στρώματα και εντέλει στο στρώμα εξόδου. Η τρέχουσα κατάσταση του συστήματος δεν καθορίζεται από καμία κατάσταση του παρελθόντος, επομένως πρόκειται για ένα σύστημα χωρίς μνήμη.

2.2.4 Βαθιά Μάθηση

Σύμφωνα με το θεώρημα του καθολικού προσεγγιστή, ένα τεχνητό νευρωνικό δίκτυο με ένα μόνο κρυφό στρώμα και επαρκές αλλά πεπερασμένο πλήθος νευρώνων, μπορεί να προσομοιώσει οποιαδήποτε συνεχής συνάρτηση. Αυτό θεωρητικά σημαίνει ότι με αυτήν την απλή αρχιτεκτονική προσαρμόζοντας των αριθμό των κρυφών νευρώνων ανάλογα με την πολυπλοκότητα του προβλήματος, μπορούμε να λύσουμε όλα τα προβλήματα. Στην πράξη όμως τα πράγματα δεν είναι τόσο απλά. Όσο πιο πολύπλοκο είναι ένα πρόβλημα και η συνάρτηση που πρέπει να προσομοιωθεί

τόσο οι νευρώνες που χρειάζονται και κατά συνέπεια οι παράμετροι του συστήματος αυξάνονται, σε σημείο που οι ανάγκες για υπολογιστική ισχύ ξεπερνούν αυτό που μπορεί να διαθέσει η σημερινή τεχνολογία. Το πρόβλημα αυτό λύνει η βαθιά μάθηση (deep learning).

Η βαθιά μάθηση δεν είναι τίποτα παραπάνω από τεχνητά νευρωνικά δίκτυα τα οποία είναι ιδιαίτερα βαθιά, έχουν δηλαδή μεγάλο αριθμό κρυφών στρωμάτων, σε σχέση με τα «παραδοσιακά» ρηχά νευρωνικά δίκτυα. Έχει αποδειχθεί ότι η αύξηση των κρυφών στρωμάτων, η αύξηση δηλαδή του Τ.Ν.Δ. σε βάθος, καταφέρνει ότι και ένα ρηχό Τ.Ν.Δ. αλλά με πολύ μικρότερο αριθμό νευρώνων. Εντέλει τα βαθιά Ν.Δ. είναι προτιμότερα υπολογιστικά αλλά και ποιοτικά, μιας και ο μικρότερος αριθμός παραμέτρων κάνει το σύστημα λιγότερο επιρρεπές σε υπερπροσαρμογή (over-fitting). Το πρόβλημα της υπερ-προσαρμογής εξηγείται παρακάτω.

2.2.5 Εκπαίδευση Τεχνητών Νευρωνικών Δικτύων

Όπως αναφέρθηκε παραπάνω σε κάθε ακμή, σε κάθε σύνδεση δηλαδή μεταξύ δύο νευρώνων διαδοχικών στρωμάτων, αντιστοιχίζεται μία παράμετρος (βάρος). Έτσι για κάθε στρώμα νευρώνων προκύπτει ένα διάνυσμα με όλα τα βάρη που συνδέουν το τρέχον στρώμα με το προηγούμενο. Κατά την εκπαίδευση των Τ.Ν.Δ. τα διανύσματα των βαρών για κάθε στρώμα, προσαρμόζονται κατάλληλα, έτσι ώστε να εκφράζουν τη σχέση μεταξύ της εισόδου και της εξόδου του δικτύου.

2.2.5.1 Συνάρτηση Σφάλματος

Η συνάρτηση σφάλματος είναι ένα εργαλείο που μας λέει ανά πάσα στιγμή πόσο «καλά» ή «κακά» αποδίδει το νευρωνικό δίκτυο. Αυτό το επιτυγχάνει υπολογίζοντας τη διαφορά μεταξύ των εξόδων που προβλέπει και των πραγματικών εξόδων που γνωρίζουμε από τις ετικέτες των δεδομένων. Ο σκοπός της εκπαίδευσης είναι να ελαχιστοποιήσει αυτήν την συνάρτηση. Μικρή τιμή της συνάρτησης σφάλματος σημαίνει ότι οι τιμές που προβλέπει το Τ.Ν.Δ. είναι κοντά στις πραγματικές τιμές και άρα το Τ.Ν.Δ. κάνει καλά τη δουλειά του. Αντίθετα μεγάλες τιμές σημαίνει ότι οι προβλέψεις είναι μακριά από τις πραγματικές τιμές και το Τ.Ν.Δ. δεν κάνει σωστά τη δουλειά του. Υπάρχουν πολλές διαφορετικές συναρτήσεις που χρησιμοποιούνται ως συναρτήσεις σφάλματος, ενώ η επιλογή εξαρτάται και από το είδος του προβλήματος. Στα πλαίσια της συγκεκριμένης εργασίας χρησιμοποιούνται οι εξής συναρτήσεις:

- Για το πρόβλημα της ταξινόμησης χρησιμοποιείται η συνάρτηση Cross-Entropy η οποία συγκρίνει τις εντροπίες μεταξύ μιας προσέγγισης της κατανομής πιθανότητας που προβλέπει το νευρωνικό δίκτυο και της πραγματικής κατανομής που προσπαθεί να προσεγγίσει.

$$H(P, Q) = -\sum_{x \in X} P(x) \log_2 Q(x)$$

  Όπου H η Cross-Entropy, x οι διακριτές καταστάσεις, P η πραγματική κατανομή πιθανότητας και Q μια προσέγγιση της πραγματικής κατανομής. [9]

- Για το πρόβλημα της παλινδρόμησης χρησιμοποιείται η συνάρτηση Μέσου Τετραγωνικού σφάλματος (Mean Square Error - MSE) η οποία υπολογίζει το μέσο όρο των τετραγώνων των διαφορών μεταξύ των τιμών που προβλέπει το νευρωνικό και των πραγματικών τιμών.

2.2.5.2 Ελαχιστοποίηση Συνάρτησης Σφάλματος

Ξέρουμε ότι για να εκπαιδευτεί ένα Τ.Ν.Δ. πρέπει να ελαχιστοποιηθεί η συνάρτηση σφάλματος. Το ερώτημα τώρα είναι με ποιόν τρόπο μπορεί να γίνει αυτό με ταχύτητα και αποτελεσματικότητα. Η καθιερωμένη μέθοδος βελτιστοποίησης που χρησιμοποιείται είναι η μέθοδος κατάβασης δυναμικού (gradient decent). Η διαδικασία ξεκινά αρχικοποιώντας τυχαία τις παραμέτρους του δικτύου. Στη συνέχεια με κάθε πέρασμα ενός μέρους των δεδομένων εκπαίδευσης, υπολογίζεται η συνάρτηση σφάλματος και το gradient (κλίση) της. Το gradient μας δείχνει προς ποια κατεύθυνση και πόσο μεταβάλλεται η συνάρτηση με τη μεταβολή των παραμέτρων. Το gradient είναι ένα διάνυσμα με όλες τις μερικές παραγώγους της συνάρτησης σφάλματος:

$$\nabla J(W) = (\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \dots, \frac{\partial J}{\partial w_N})$$

Όπου W το διάνυσμα των βαρών και $w_{1\text{-}N}$ τα επιμέρους βάρη.

Γνωρίζοντας πλέον την κλίση της συνάρτησης ως προς όλους τους άξονες, μεταβάλλουμε τις παραμέτρους έτσι ώστε η συνάρτηση σφάλματος να μειωθεί όσο το δυνατών περισσότερο. Ο ρυθμός με τον οποίο μειώνεται η συνάρτηση με κάθε πέρασμα εξαρτάται από μια υπερ-παράμετρο που ονομάζεται ρυθμός εκπαίδευσης. Η επιλογή του ρυθμού εκπαίδευσης α είναι καθοριστική για την απόδοση του δικτύου. Εάν ο ρυθμός εκπαίδευσης είναι πολύ μικρός λογικά θα χρειαστεί υπερβολικά πολύς χρόνος για να βρεθεί το ελάχιστο της συνάρτησης. Ενώ αν είναι πολύ μεγάλος μπορεί να μην καταφέρνει να βρει το ελάχιστο μιας και οι μεταβολές των παραμέτρων δεν θα είναι αρκετά λεπτομερείς.

$$W = W - a\nabla J(W)$$

2.2.5.3 SGD και MB-GD

Επειδή στην πράξη τα Τ.Ν.Δ. είναι τεράστια, με εκατομμύρια παραμέτρους και όσο πάει μεγαλώνουν ακόμα περισσότερο, ο υπολογισμός των παραγώγων για όλες τις παραμέτρους είναι υπερβολικά αργός. Γι’ αυτό και ο απλή μέθοδος gradient decent στην πράξη δεν χρησιμοποιείται. Εναλλακτικά χρησιμοποιούνται άλλες παραλλαγές όπως η Στοχαστική Κατάβαση Δυναμικού (Stochastic Gradient Decent – SGD). Η SGD αφού ανακατεύσει το σύνολο δεδομένων, περνάει από κάθε δείγμα ξεχωριστά και υπολογίζει το gradient για το συγκεκριμένο σημείο, ενημερώνοντας στη συνέχεια τα βάρη. Αν και αρχικά αυτό φαίνεται σαν μια κακή ιδέα επειδή ένα μεμονωμένο παράδειγμα μπορεί να τύχει να είναι ακραία περίπτωση, αποδεικνύεται ότι όταν αυτό γίνεται για κάθε δείγμα του συνόλου δεδομένων με τυχαία σειρά, οι συνολικές διακυμάνσεις εξαλείφονται. Επιπλέον η SGD έχει το πλεονέκτημα ότι μπορεί να ξεφεύγει από τοπικά ελάχιστα, επειδή οι μεταβολές συχνά μπορεί να είναι απότομες. Η MB-GD είναι μια ταχύτερη παραλλαγή της SGD όπου το σύνολο δεδομένων χωρίζεται σε μικρές ομάδες και η διαδικασία εκτελείται ανά ομάδα και όχι ανά μεμονωμένο δείγμα.

2.2.5.4 Ορμή

Η ορμή είναι μία προσθήκη σε μερικές παραλλαγές της GD όπου η ενημέρωση των βαρών έχει αδράνεια. Η ενημέρωση των βαρών δεν είναι πλέον μόνο συνάρτηση της κλίσης της συνάρτησης σφάλματος, αλλά προσαρμόζεται σταδιακά από το ρυθμό της προηγούμενης ενημέρωσης. Στην απλή κατάβαση δυναμικού, χρησιμοποιούμαι τον παρακάτω τύπο:

$$W_t = W_t - a\nabla J(W_t)$$

Το t προστέθηκε για να συμβολίζει τη χρονική στιγμή του κάθε βήματος. Η γενικευμένη σχέση για την κατάβαση δυναμικού με ορμή είναι η εξής:

$$z_t = \beta z_{t-1} + \nabla J(W_{t-1})$$

$$W_t = W_{t-1} - \alpha z_t$$

Η κλίση αντικαταστάθηκε από μια πιο σύνθετη συνάρτηση η οποία λαμβάνει υπόψη την κλίση σε προηγούμενα χρονικά βήματα. Όσο πιο μεγάλο είναι το β, τόσο περισσότερη ορμή έχει η ενημέρωση των παραμέτρων. Αν το β = 0 τότε ουσιαστικά έχουμε απλό gradient decent.

2.2.5.5 Προσαρμοζόμενες Μέθοδοι

Ένας από τους πιο σημαντικούς παράγοντες για την επιτυχή εκπαίδευση ενός Τ.Ν.Δ. είναι ο ρυθμός εκπαίδευσης α. Τυπικά η τιμή του ρυθμού εκπαίδευσης επιλέγεται στην αρχή και στη συνέχεια μειώνεται σταδιακά κάθε κάποια χρονικά βήματα ώστε προς το τέλος της εκπαίδευσης τα βάρη να ενημερώνονται με μικρότερο ρυθμό ψάχνοντας το ελάχιστο με περισσότερη ακρίβεια. Στην πράξη αυτό δεν είναι αρκετό όμως διότι δεν λαμβάνει υπόψη τα ιδιαίτερα χαρακτηριστικά της συνάρτησης σφάλματος σε κάθε στιγμή. Η ορμή βοηθάει λίγο στην προσαρμογή του ρυθμού εκπαίδευσης κατά τη διάρκεια της εκπαίδευσης, αλλά το πρόβλημα της επιλογής του α παραμένει.

Τη λύση στο πρόβλημα δίνουν αλγόριθμοι οι οποίοι προσαρμόζουν το ρυθμό εκπαίδευσης σε κάθε παράμετρο ξεχωριστά, λαμβάνοντας υπόψη την ανομοιογένεια μεταξύ των παραμέτρων. Το πιο απλό παράδειγμα είναι η μέθοδος AdaGrad (Adaptive subGradient), η οποία ενημερώνει την κάθε παράμετρο ξεχωριστά με βάση την κλίση της, αλλά με έναν νέο συντελεστή που προσπαθεί να εξισώσει το συντελεστή μάθησης μεταξύ των παραμέτρων με μεγάλες κλίσης και των παραμέτρων με μικρές. Η AdaGrad ορίζεται από την εξής σχέση:

$$w_i = w_i - \frac{a}{\sqrt{G_i + \varepsilon}} \frac{\partial J}{\partial w_i}$$

Το $\sqrt{G_i + \varepsilon}$ αντιπροσωπεύει το άθροισμα των κλίσεων για τη συγκεκριμένη παράμετρο για κάθε βήμα από τη στιγμή που ξεκίνησε η εκπαίδευση. Το ε είναι ένας πολύ μικρός αριθμός που υπάρχει απλά για την αποφυγή της διαίρεσης με το μηδέν. Διαιρώντας το α με αυτόν τον όρο για κάθε παράμετρο, μειώνουμε το ρυθμό εκπαίδευσης για τις παραμέτρους που είχαν μεγάλες κλίσεις έως τη συγκεκριμένη στιγμή και αντίθετα αυξάνουμε το ρυθμό για τις παραμέτρους που είχαν μικρές κλίσεις.

Η AdaGrad ως επί το πλείστον εξαλείφει την ανάγκη αντιμετώπισης του αρχικού ρυθμού μάθησης α ως υπερπαράμετρο, έχει όμως τις δικές του προκλήσεις. Το πρόβλημα με την AdaGrad είναι ότι η μάθηση μπορεί να σταματήσει πρόωρα καθώς το Gi συσσωρεύεται για κάθε παράμετρο με την πάροδο του χρόνου και μειώνει το μέγεθος των ενημερώσεων. Μια παραλλαγή της AdaGrad, το AdaDelta, αντιμετωπίζει αυτό το πρόβλημα περιορίζοντας το παράθυρο του όρου συσσώρευσης

κλίσης στις πιο πρόσφατες ενημερώσεις. Μια άλλη μέθοδος που μοιάζει πολύ με την AdaDelta είναι η RMSprop. Η RMSprop περιορίζει ομοίως την ενημέρωση αθροίζοντας τα τετράγωνα των προηγούμενων ενημερώσεων, αλλά το κάνει με απλούστερο τρόπο χρησιμοποιώντας μία σχέση με έναν συντελεστή μείωσης, ο οποίος καταλήγει να είναι μια υπερπαράμετρος. Έτσι, τόσο για την AdaDelta όσο και για την RMSprop ο ρυθμός εκπαίδευσης δεν προσαρμόζεται απλά σε σχέση παραμέτρους, αλλά προσαρμόζεται και σε σχέση με το χρόνο, αποφεύγοντας έτσι την εξάλειψη του.

2.2.5.6 ADAM

Η καθιερωμένη μέθοδος βελτιστοποίησης και αυτή που χρησιμοποιείται στη συγκεκριμένη εργασία είναι η μέθοδος ADAM (Adaptive Moment Estimation). Η ADAM συνδυάζει τις μεθόδους που βασίζονται στην ορμή και τις προσαρμοζόμενες μεθόδους. Όπως η AdaDelta και η RMSprop, η ADAM προσαρμόζει το ρυθμό μάθησης για κάθε παράμετρο σύμφωνα με ένα ολισθαίνον παράθυρο των προηγούμενων κλίσεων (gradients), αλλά χρησιμοποιεί και ορμή για την εξομάλυνση της πορείας κατά τη διάρκεια των χρονικών βημάτων.

2.2.6 Συνελικτικά Νευρωνικά Δίκτυα

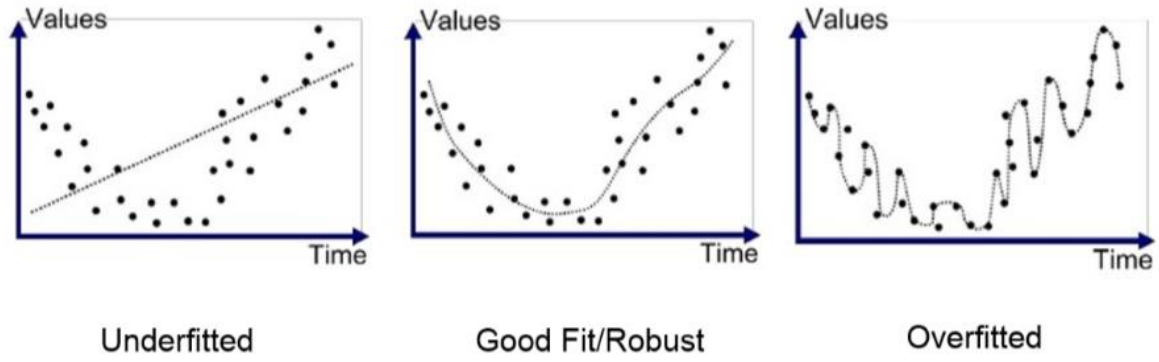
Τα συνελικτικά νευρωνικά δίκτυα, ή convolutional neural networks (CNNs) είναι παραλλαγή των παραδοσιακών Τ.Ν.Δ. Ότι ισχύει στα συνηθισμένα Τ.Ν.Δ. σχετικά με την αρχή λειτουργίας και τη βασική αρχικεκτονική, ισχύει και για τα CNNs. Τα CNNs χρησιμοποιούνται κυρίως στον τομέα της αναγνώρισης προτύπων σε εικόνες. Για αυτό το λόγω στην αρχιτεκτονική τους κωδικοποιούνται χαρακτηριστικά ειδικά για την εικόνα, τα οποία καθιστούν το δίκτυο κατάλληλο για το συγκεκριμένο σκοπό, ενώ ταυτόχρονα όπως θα φανεί παρακάτω μειώνεται σημαντικά ο αριθμός των παραμέτρων που απαιτείται για τη δημιουργία του μοντέλου.

Ένας από τους μεγαλύτερους περιορισμούς των παραδοσιακών μορφών Τ.Ν.Δ. είναι ότι έχουν την τάση να δυσκολεύονται με την υπολογιστική πολυπλοκότητα που απαιτεί ο υπολογισμός δεδομένων εικόνας. Συνηθισμένα σύνολα δεδομένων αναφοράς όπως η βάση δεδομένων MNIST που αποτελείται από εικόνες που περιέχουν χειρόγραφα ψηφία, είναι κατάλληλα για τα περισσότερα Τ.Ν.Δ. λόγω της μικρής διαστατικότητας της εικόνας (28x28). Με το συγκεκριμένο σύνολο δεδομένων, οι νευρώνες του πρώτου κρυφού στρώματος έχουν 784 βάρη, 28x28x1 μιας και πρόκειται για ασπρόμαυρες εικόνες. Αυτός ο αριθμός βαρών είναι διαχειρίσιμος για τις περισσότερες μορφές Τ.Ν.Δ.

Αν αντικαταστήσουμε το προηγούμενο σύνολο δεδομένων με ένα σύνολο δεδομένων που αποτελείται από έγχρωμες εικόνες διαστάσεων 64x64, ο αριθμός των βαρών σε κάθε νευρώνα του πρώτου κρυφού στρώματος αυξάνεται σε 12.288. Για να αντιμετωπιστεί αυτή η κλίμακα εισόδου, το δίκτυο πρέπει να γίνει πολύ μεγαλύτερο σε σχέση με αυτό που χρησιμοποιείται για την ταξινόμηση του MNIST. Από αυτό καταλαβαίνει κανείς ότι όσο οι διαστάσεις των εικόνων αυξάνονται, ο αριθμός των παραμέτρων γίνεται όλο και πιο εξωφρενικός και κατά συνέπεια το δίκτυο μεγαλώνει.

Γιατί όμως μας απασχολεί αυτό; Γιατί δεν αρκεί απλά να αυξήσουμε των αριθμό των στρωμάτων και των αριθμό των νευρώνων μέσα σε αυτά; Υπάρχουν δύο βασικοί λόγοι για τους οποίους αυτή η τακτική είναι καταστροφική. Ο πρώτος είναι το υπολογιστικό κόστος και ο χρόνος που χρειάζεται για την εκπαίδευση αυτών των τεράστιων δικτύων. Ο δεύτερος λόγος είναι το φαινόμενο της υπερμοντελοποίησης (overfitting). Η υπερμοντελοποίηση συμβαίνει όταν ένα μοντέλο προσαρμόζεται

με υπερβολική ακρίβεια στα δεδομένα εκπαίδευσης, χάνοντας έτσι τη δυνατότητα να εντοπίζει και να μαθαίνει τα μοτίβα που κρύβει ένα σύνολο δεδομένων. Αυτό έχει ως αποτέλεσμα το μοντέλο να αποδίδει πολύ χειρότερα σε νέα δεδομένα. Στο σχήμα 2.4 φαίνεται ξεκάθαρα η διαφορά μεταξύ ενός μοντέλου το οποίο έχει συλλάβει το «νόημα» των δεδομένων και ενός μοντέλου που έχει απλά απομνημονεύσει τα δεδομένα στα οποία έχει εκπαιδευτεί. Μία πολύ εύστοχη αναλογία από την πραγματική ζωή, ενός θέματος που λογικά έχει απασχολήσει κάθε μαθητή και φοιτητή κατά τη διάρκεια της μαθητικής και φοιτητικής ζωής, είναι η διαφορά μεταξύ της καλής κατανόησης μιας ιδέας ή ενός κειμένου και της αποστήθισης της. Εκτός από αυτές τις δύο περιπτώσεις βέβαια, υπάρχει και η περίπτωση της υπομοντελοποίησης (underfitting), όπου το μοντέλο δεν έχει καταφέρει ούτε να κατανοήσει τα δεδομένα αλλά ούτε και να τα απομνημονεύσει. Αυτό συνήθως είναι αποτέλεσμα της αντίθετης περίπτωσης όπου το μοντέλο δεν είναι αρκετά μεγάλο, δεν έχει δηλαδή αρκετές παραμέτρους.

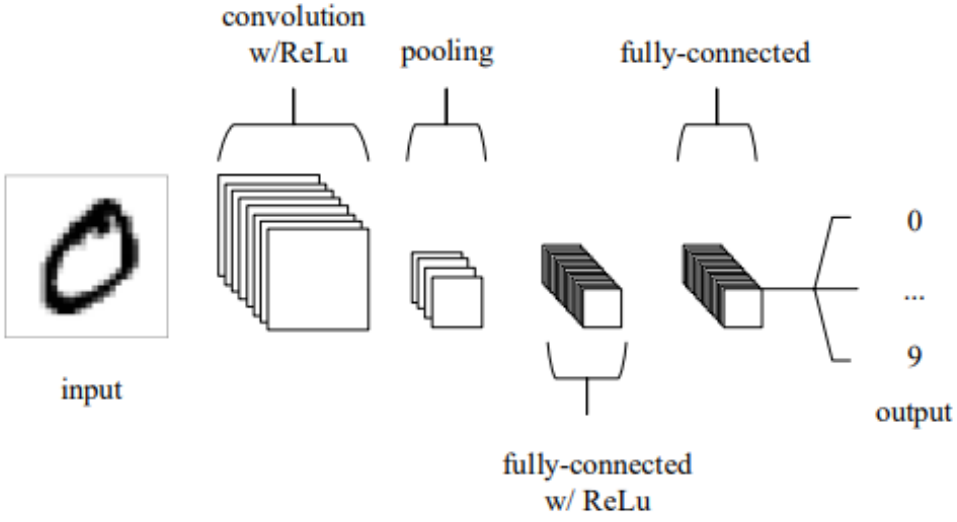


Σχήμα 2.4: Γραφική αναπαράσταση των φαινομένων υπομοντελοποίησης (underfitting), καλής γενικευμένης μοντελοποίησης και υπερμοντελοποίησης (overfitting).

2.2.7 Αρχιτεκτονική Συνελικτικών Νευρωνικών Δικτύων

Κατά βάση η αρχιτεκτονική των CNN χωρίζεται σε τρία στάδια επεξεργασίας, στην πράξη δηλαδή, όπως φαίνεται και στο σχήμα 2.5, χωρίζεται στα εξής τρία είδη στρωμάτων:

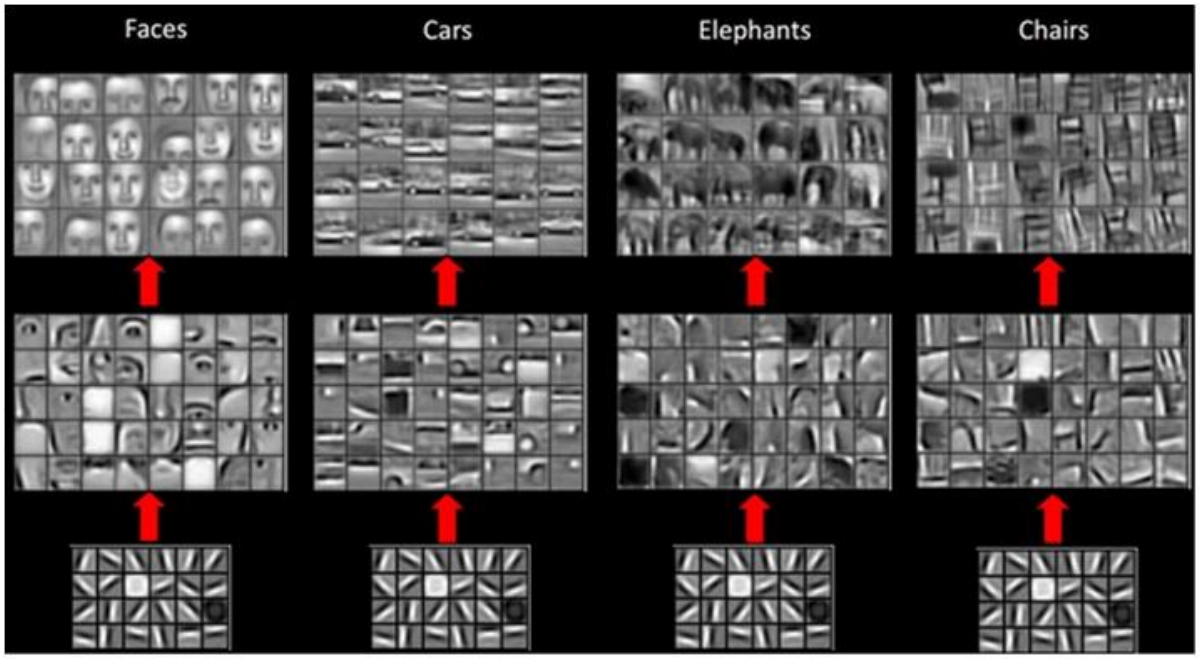
- Συνελικτικό στρώμα (convolution layer)
- Στρώμα υπο-δειγματοληψίας (pooling layer)
- Πλήρως συνδεδεμένο στρώμα (fully-connected layer)



Σχήμα 2.5: Βασική Αρχιτεκτονική Συνελικτικών Νευρωνικών Δικτύων. [7]

2.2.7.1 Συνελικτικό Στρώμα

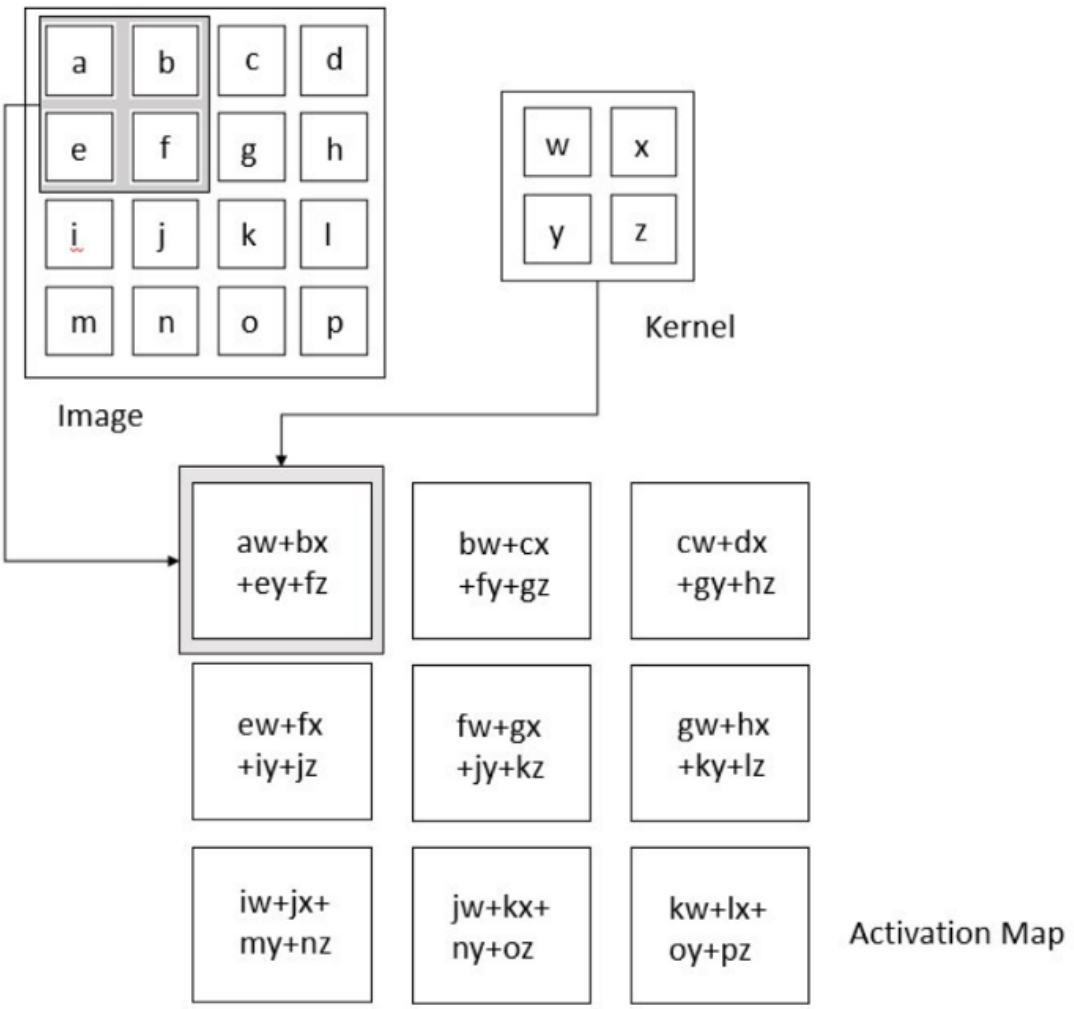
Αυτό το στρώμα είναι υπεύθυνο για τον εντοπισμό χαρακτηριστικών στην εικόνα. Το δίκτυο ξεκινά εντοπίζοντας πολύ βασικά χαρακτηριστικά όπως ακμές, χρωματικές ζώνες, υφές και σταδιακά συνδυάζοντας τα «ευρήματα» από τα προηγούμενα στρώματα, εντοπίζει όλο και πιο πολύπλοκα χαρακτηριστικά, καταλήγοντας να αναγνωρίζει τις πολύπλοκες μορφές που επιθυμούμαι, σχήμα 2.6.



Σχήμα 2.6: Οπτική αναπαράσταση σταδιακής αναγνώρισης μορφών από CNN. [8]

Το επίπεδο συνέλιξης είναι το βασικό δομικό στοιχείο του CNN και φέρει το κύριο μέρος του υπολογιστικού φορτίου στο δίκτυο. Αυτό το στρώμα υπολογίζει το γινόμενο μεταξύ δύο πινάκων, όπου ο ένας πίνακας είναι το σύνολο των παραμέτρων που καλείται να μάθει το μοντέλο, γνωστό και ως πυρήνας (Kernel). Ο άλλος πίνακας είναι ένα περιορισμένο τμήμα, ένα υποσύνολο της εικόνας. Ο πυρήνας είναι χωρικά μικρότερος από την εικόνα εισόδου, αλλά είναι μεγαλύτερος σε βάθος. Αν πρόκειται για εικόνα τριών καναλιών (RGB), τότε το ύψος και το πλάτος θα είναι μικρότερα, αλλά το βάθος θα εκτείνεται και στα τρία κανάλια.

Κατά τη διάρκεια του εμπρόσθιου περάσματος, ο πυρήνας ολισθαίνει κατά μήκος του ύψους και του πλάτους της εικόνας, παράγοντας μια τιμή που αναπαριστά το συγκεκριμένο τμήμα της εικόνας. Αφού ο πυρήνας «σκανάρει» όλη την εικόνα, παράγεται μια δισδιάστατη αναπαράσταση της εικόνας, γνωστή ως χάρτης ενεργοποίησης (activation map), που εκφράζει την απόκριση του πυρήνα σε κάθε χωρική θέση της εικόνας. Το μέγεθος της ολίσθησης του πυρήνα ονομάζεται βήμα. Όσο μικρότερο είναι το βήμα, τόσο πιο μεγάλης διάστασης, και άρα πιο λεπτομερής είναι ο χάρτης ενεργοποίησης. Στο σχήμα 2.7 φαίνεται το πώς παράγεται ένας χάρτης ενεργοποίησης 3x3, από μια εικόνα 4x4, με βήμα 1.



Σχήμα 2.7: Διαδικασία συνέλιξης και δημιουργία χάρτη ενεργοποίησης (activation map). [9]

Υπάρχουν τρεις βασικοί λόγοι για τους οποίους τα CNNs καθιερώθηκαν ως η πιο δημοφιλής αρχιτεκτονική για υπολογιστική όραση:

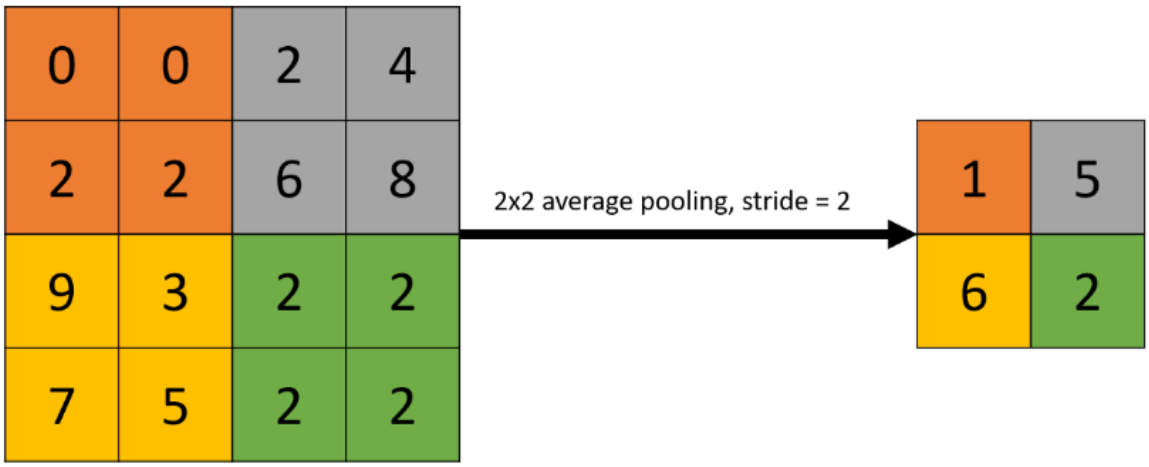
- Αραιή αλληλεπίδραση (sparse interaction): Στα «παραδοσιακά» στρώματα νευρωνικών δικτύων κάθε μονάδα εξόδου αλληλεπιδρά με κάθε μονάδα εισόδου. Αυτό σημαίνει ότι το σύνολο των βαρών δημιουργούν ένα πίνακα παραμέτρων που περιγράφει την αλληλεπίδραση μεταξύ της μονάδας εισόδου και της μονάδας εξόδου. Αντιθέτως τα CNNs έχουν αραιή αλληλεπίδραση. Αυτό συμβαίνει γιατί ο πυρήνας είναι μικρότερος από την είσοδο, π.χ. μια εικόνα μπορεί να έχει χιλιάδες ή εκατομμύρια εικονοστοιχεία, αλλά κατά την επεξεργασία με τη χρήση του πυρήνα μπορεί να ανιχνευτούν σημαντικές πληροφορίες από δεκάδες ή εκατοντάδες εικονοστοιχεία. Αυτό σημαίνει ότι τελικά χρειάζεται να αποθηκεύσουμε λιγότερες παραμέτρους. Έτσι μειώνονται οι απαιτήσεις μνήμης του μοντέλου, αυξάνεται η ταχύτητα εκπαίδευσης και βελτιώνεται η στατιστική του αποδοτικότητα.

- Επαναχρησιμοποίηση παραμέτρων (parameter sharing): Σε ένα «παραδοσιακό» νευρωνικό δίκτυο κάθε στοιχείο του πίνακα βαρών χρησιμοποιείται μόνο μία φορά και στη συνέχεια δεν επανεξετάζεται. Αντίθετα σε ένα CNN τα βάρη ενός πυρήνα χρησιμοποιούνται πολλές φορές για την επεξεργασία πολλών περιοχών μια εικόνας, άρα πολλών διαφορετικών εισόδων.

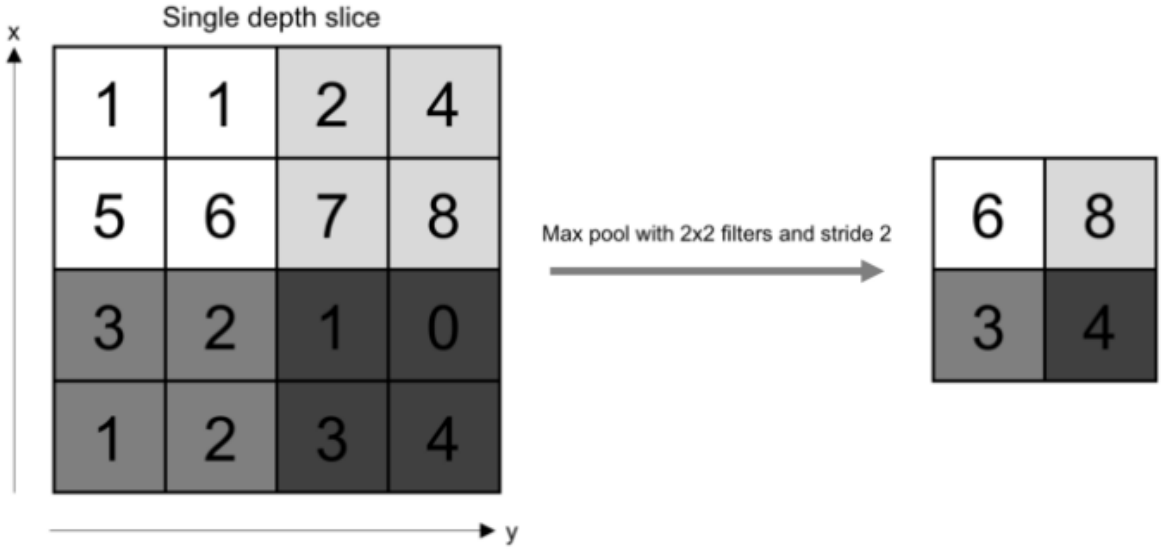
- Ισοδύναμη αναπαράσταση (equivariant representation): Λόγω της επαναχρησιμοποίησης των παραμέτρων, τα στρώματα του CNN έχουν την ιδιότητα της ισοδύναμης μετάφρασης. Αν αλλάξουμε την είσοδο με έναν τρόπο, τότε και η έξοδος θα αλλάξει με τον ίδιο τρόπο.

2.2.7.2 Στρώμα Υπό-δειγματοληψίας (Pooling Layer)

Το στρώμα υπο-δειγματοληψίας αντικαθιστά την έξοδο του δικτύου σε ορισμένα σημεία κάνοντας μια σύνοψη μερικών γειτονικών στοιχείων. Αυτό έχει ως αποτέλεσμα τη μείωση των διαστάσεων της εισόδου στο επόμενο στρώμα και άρα τη μείωση των παραμέτρων και του υπολογιστικού φόρτου. Οι πιο συχνά χρησιμοποιούμενες μέθοδοι υπο-δειγματοληψίας είναι το average pooling, σχήμα 2.8, όπου υπολογίζεται ο μέσος όρος των τιμών των στοιχείων ενός ορισμένου υποσυνόλου της προηγούμενης εξόδου και το max pooling, όπου υπολογίζεται η μέγιστη τιμή με αντίστοιχο τρόπο. Στο σχήμα 2.9 φαίνεται σχηματικά πως από ένα πίνακα 4x4 με max pooling 2x2 και βήμα 2, καταλήγουμε σε έναν πίνακα 2x2.



Σχήμα 2.8: Οπτική αναπαράσταση average pooling. [10]



Σχήμα 2.9: Οπτική αναπαράσταση max pooling. [9]

Εκτός από την προφανή οικονομία μνήμης και επεξεργαστικής ισχύς, το max pooling προσφέρει ανεξαρτησία όσο αφορά τη θέση, την περιστροφή και τη κλίμακα των χαρακτηριστικών που εντοπίζονται σε μια εικόνα. Ουσιαστικά οι παραπάνω πληροφορίες είναι αυτές που χάνονται κατά τη συμπύκνωση, ενώ παραμένει η ουσιαστική πληροφορία που εκφράζει κατά πόσο ένα χαρακτηριστικό υπάρχει ή όχι σε μια εικόνα. Αυτό ανάλογα με το πρόβλημα που προσπαθούμε να λύσουμε μπορεί να είναι ή και να μην είναι επιθυμητό. Σε κάθε περίπτωση αυτή η απλοποίηση μπορεί να έχει σημαντικό όφελος στην απόδοση του δικτύου.

2.2.7.3 Πλήρως Συνδεδεμένο Στρώμα (Fully Connected Layer)

Στο πλήρως συνδεδεμένο στρώμα όλοι οι νευρώνες του προηγούμενου στρώματος συνδέονται με όλους τους νευρώνες του επόμενου στρώματος, επομένως ο πίνακας βαρών εκφράζει την πλήρη αλληλεπίδραση μεταξύ εισόδου και εξόδου. Αυτό το στρώμα βρίσκεται πάντα στο τέλος και αυτό που κάνει είναι να λαμβάνει τα χαρακτηριστικά που εξάγουν τα προηγούμενα στρώματα και με βάση αυτά να αντιστοιχεί την είσοδο σε μια κλάση αν πρόκειται για πρόβλημα ταξινόμησης, ή να εξάγει μια αριθμητική τιμή που εκφράζει την τρέχουσα είσοδο αν πρόκειται για πρόβλημα παλινδρόμησης.

2.2.7.4 Έξοδος Νευρωνικού Δικύου (Ταξινόμηση / Παλινδρόμηση)

Ανάλογα με το πρόβλημα που προσπαθούμε να λύσουμε, χρησιμοποιούμε ένα τεχνητό νευρωνικό δίκτυο, στην περίπτωση που μας ενδιαφέρει ένα CNN, είτε για ταξινόμηση, είτε για παλινδρόμηση. Στην πράξη το μόνο που αλλάζει σε αυτές τις δύο περιπτώσεις είναι η έξοδος του δικτύου.

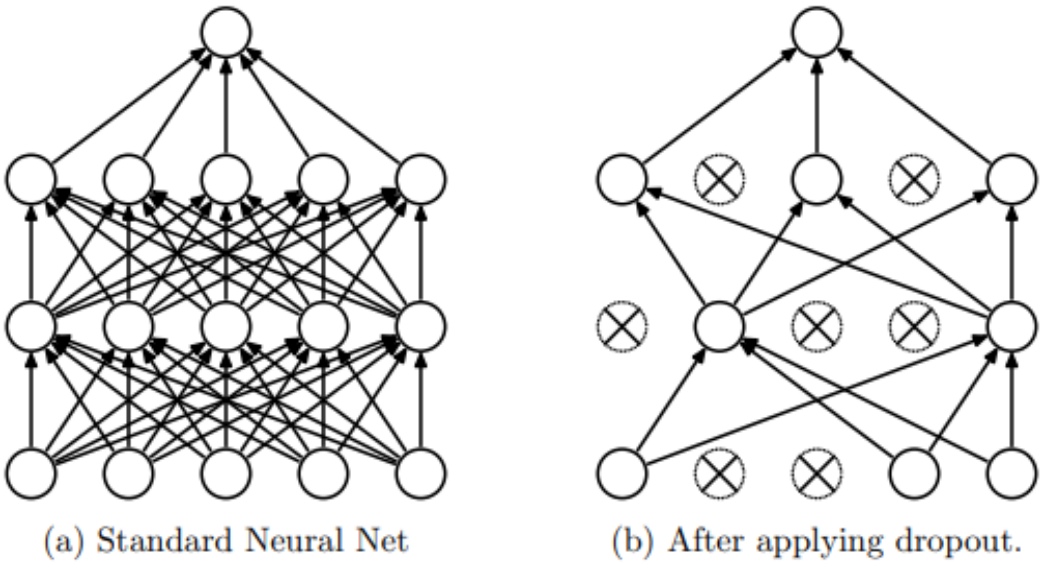
- Ταξινόμηση (classification): Η καθιερωμένη μέθοδος ταξινόμησης στην έξοδο ενός CNN είναι η χρήση της συνάρτησης ενεργοποίησης Softmax. Η softmax έχει την ιδιότητα να μετατρέπει ένα διάνυσμα αριθμών σε διάνυσμα πιθανοτήτων. Η έξοδος αποτελείται από τόσους νευρώνες όσες και οι κλάσεις του προβλήματος και κάθε νευρώνας μετά το πέρας

ενός κύκλου επεξεργασίας, λαμβάνει μία τιμή πιθανότητας μεταξύ του 0 και του 1. Ο νευρώνας με τη μεγαλύτερη τιμή εκφράζει την κλάση στην οποία προβλέπει το δίκτυο πως αντιστοιχεί η τρέχουσα είσοδος.

- Παλινδρόμηση (regression): Στην περίπτωση της παλινδρόμησης τα πράγματα πολύ πιο απλά. Η συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η γραμμική συνάρτηση μιας και το δίκτυο ζητείται να προβλέψει μια αριθμητική τιμή.

2.2.8 Dropout

Το dropout είναι μια τεχνική κανονικοποίησης για τη μείωση της υπερ-προσαρμογής στα νευρωνικά δίκτυα. Η υπερ-προσαρμογή, όπως αναφέρθηκε και παραπάνω, συμβαίνει όταν ένα μοντέλο είναι υπερβολικά πολύπλοκο και είναι σε θέση να μάθει τις λεπτομέρειες και το θόρυβο στα δεδομένα εκπαίδευσης σε σημείο που να επηρεάζει αρνητικά την ικανότητα του μοντέλου να γενικεύει σε νέα δεδομένα. Το dropout είναι ένας τρόπος αντιμετώπισης αυτού του προβλήματος μηδενίζοντας τυχαία ορισμένες από τις εξόδους ενός επιπέδου κατά τη διάρκεια της εκπαίδευσης, σχήμα 2.10.. Αυτό έχει ως αποτέλεσμα την "αποβολή" αυτών των νευρώνων από το δίκτυο, εμποδίζοντας την ενημέρωση των βαρών τους κατά τη διάρκεια της εκπαίδευσης.



Σχήμα 2.10. a) Πλήρως συνδεδεμένο νευρωνικό δίκτυο χωρίς Dropout. B) Πλήρως συνδεδεμένο νευρωνικό δίκτυο με Dropout. [11]

Η ιδέα πίσω από το dropout είναι ότι, με την τυχαία απόρριψη νευρώνων κατά τη διάρκεια της εκπαίδευσης, το μοντέλο αναγκάζεται να μάθει πολλαπλές ανεξάρτητες αναπαραστάσεις των ίδιων δεδομένων. Αυτό αυξάνει την ικανότητα του μοντέλου να γενικεύει, επειδή το μοντέλο δεν μπορεί να βασίζεται σε κάποιον μεμονωμένο νευρώνα ή ομάδα νευρώνων για να κάνει όλες τις προβλέψεις του.

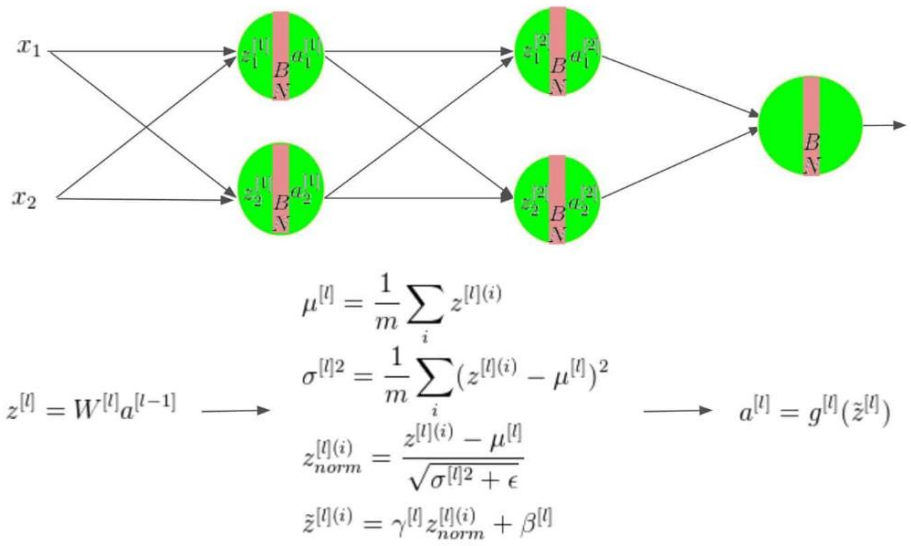
Αντ' αυτού, πρέπει να μάθει να χρησιμοποιεί πολλαπλούς διαφορετικούς συνδυασμούς νευρώνων για να κάνει ακριβείς προβλέψεις.

Το dropout εφαρμόζεται συνήθως σε πλήρως συνδεδεμένα (dense) στρώματα σε ένα νευρωνικό δίκτυο, αν και μπορεί να εφαρμοστεί και σε συνελικτικά στρώματα. Όταν χρησιμοποιείται, είναι σημαντικό να χρησιμοποιείται ένα ποσοστό dropout που δεν είναι πολύ υψηλό, διότι ένα υψηλό ποσοστό μπορεί να προκαλέσει υποπροσαρμογή του μοντέλου. Μια συνήθης επιλογή για το ποσοστό εγκατάλειψης είναι μεταξύ 0,2 και 0,5. Εξίσου σημαντικό είναι να χρησιμοποιείται μόνο κατά τη διάρκεια της εκπαίδευσης και όχι κατά τη διάρκεια της ανάκλησης, διότι ο τυχαίος μηδενισμός ορισμένων νευρώνων θα επηρεάσει σημαντικά την ικανότητα του μοντέλου να κάνει σωστές προβλέψεις.

2.2.9 Batch Normalization (Κανονικοποίηση Παρτίδας)

Το batch normalization είναι μια τεχνική που χρησιμοποιείται στη βαθιά μάθηση για την κανονικοποίηση των εξόδων των συναρτήσεων ενεργοποίησης στα επίπεδα ενός δικτύου (activation layers). Παρουσιάστηκε από τους Sergey Ioffe και Christian Szegedy στην εργασία τους "Batch Normalization" το 2015: Accelerating Deep Network Training by Reducing Internal Covariate Shift". [12]

Ο στόχος του Batch Normalization είναι η βελτίωση της απόδοσης και της σταθερότητας των νευρωνικών δικτύων, ιδίως όταν εκπαιδεύονται βαθιά δίκτυα με πολλά επίπεδα. Κατά την εκπαίδευση ενός βαθιού νευρωνικού δικτύου, οι ενεργοποιήσεις κάθε στρώματος μπορεί να διαφέρουν σημαντικά από στρώμα σε στρώμα, γεγονός που μπορεί να δυσχεράνει τη μάθηση του δικτύου. Το batch normalization βοηθά στην αντιμετώπιση αυτού του προβλήματος με την κανονικοποίηση των ενεργοποιήσεων κάθε στρώματος , χρησιμοποιώντας τη μέση τιμή και την τυπική απόκλιση της παρτίδας δεδομένων που επεξεργάζεται κάθε φορά. Στη συνέχεια, οι κανονικοποιημένες ενεργοποιήσεις κλιμακώνονται και μετατοπίζονται έτσι ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1, σχήμα 2.11.



Σχήμα 2.11, Γραφική αναπαράσταση του Batch Normalization και υπολογισμοί. [13]

Οι τέσσερεις εξισώσεις του σχήματος 2.11 κάνουν υπολογίζουν τα παρακάτω:

- Το μέσω όρο μ μιας μικρής παρτίδας δεδομένων.
- Τη διακύμανση της παρτίδας αυτής.
- Το z_{norm} αφαιρώντας το μέσω όρο μ από τις εξόδους των προηγούμενων στρωμάτων z και διαιρώντας με την τυπική απόκλιση σ. Το ε είναι ένας μικρός αριθμός που προστίθεται για να αποφευχθεί η διαίρεση με το 0.
- Τα $\hat{z}_{norm}$ πολλαπλασιάζοντας z_{norm} με ένα συντελεστή γ και μία μετατόπιση β. Οι δύο παράμετροι γ και β προσαρμόζονται κατά τη διάρκεια της εκπαίδευσης μαζί με τα βάρη W.

Υπάρχουν διάφορα οφέλη από τη χρήση της κανονικοποίησης παρτίδας στη βαθιά μάθηση:

- Μπορεί να επιταχύνει τη διαδικασία εκπαίδευσης επιτρέποντας στο δίκτυο να χρησιμοποιεί υψηλότερους ρυθμούς μάθησης.
- Μπορεί να βελτιώσει τη γενίκευση του δικτύου σε νέα δεδομένα μειώνοντας το internal covariate shift, το οποίο συμβαίνει ακριβώς όταν η στατιστική κατανομή της εισόδου στα δίκτυα διαφέρει δραστικά από τις εισόδους του παρελθόντος.
- Μπορεί επομένως να κάνει το δίκτυο πιο ανθεκτικό στις αλλαγές στην κατανομή των δεδομένων εισόδου.
- Μειώνει την σημαντικότητα της «σωστής» αρχικοποίησης των βαρών.
- Μειώνει την ανάγκη για Dropout

Συνολικά, η ομαλοποίηση παρτίδας είναι μια χρήσιμη τεχνική για τη βελτίωση της απόδοσης και της σταθερότητας των δικτύων βαθιάς μάθησης. Έχει υιοθετηθεί ευρέως σε διάφορες εφαρμογές, όπως η ταξινόμηση εικόνων, η επεξεργασία φυσικής γλώσσας και η αναγνώριση ομιλίας. [14]

2.2.10 Data Augmentation (Επαύξηση Δεδομένων)

Το data augmentation είναι μια τεχνική που χρησιμοποιείται για την τεχνητή αύξηση του μεγέθους ενός συνόλου δεδομένων με τη δημιουργία τροποποιημένων εκδόσεων των υφιστάμενων δεδομένων. Αυτό γίνεται συχνά προκειμένου να βελτιωθεί η απόδοση των μοντέλων μηχανικής μάθησης, ιδίως όταν ο όγκος των διαθέσιμων δεδομένων είναι μικρός. Η επαύξηση δεδομένων μπορεί να είναι ιδιαίτερα χρήσιμη για εργασίες ταξινόμησης εικόνων, όπου χρησιμοποιείται συχνά για τη δημιουργία πρόσθετων παραδειγμάτων εκπαίδευσης με την εφαρμογή διαφόρων μετασχηματισμών σε υπάρχουσες εικόνες. Αυτοί οι μετασχηματισμοί μπορεί να περιλαμβάνουν πράγματα όπως η περιστροφή ή η περικοπή της εικόνας, η προσαρμογή της φωτεινότητας ή της αντίθεσης ή η προσθήκη θορύβου.

Υπάρχουν πολλές διαφορετικές τεχνικές που μπορούν να χρησιμοποιηθούν για την επαύξηση δεδομένων και οι συγκεκριμένες τεχνικές εξαρτώνται από τον τύπο των δεδομένων. Έτσι για παράδειγμα αν αντί για εικόνες έχουμε δεδομένα κειμένου, τότε μπορούν να χρησιμοποιηθούν τεχνικές όπως η αντικατάσταση συνωνύμων ή η επανεγγραφή κειμένου. Είναι επομένως σημαντικό οι τεχνικές data augmentation που χρησιμοποιούνται να είναι κατάλληλες για τον τύπο δεδομένων στα οποία θα εφαρμοστούν.

Επιπλέον όταν χρησιμοποιείται data augmentation πρέπει να γίνεται με τέτοιο τρόπο ώστε να μην εισάγονται biases. Bias σημαίνει να έχει το μοντέλο την τάση να τα πηγαίνει καλύτερα σε ένα συγκεκριμένο τύπο δεδομένων. Για παράδειγμα αν έχουμε ένα σύνολο δεδομένων με εικόνες προσώπων και κάνουμε μόνο περιστροφή, τότε μπορεί να καταλήξουμε με ένα μοντέλο που τείνει να

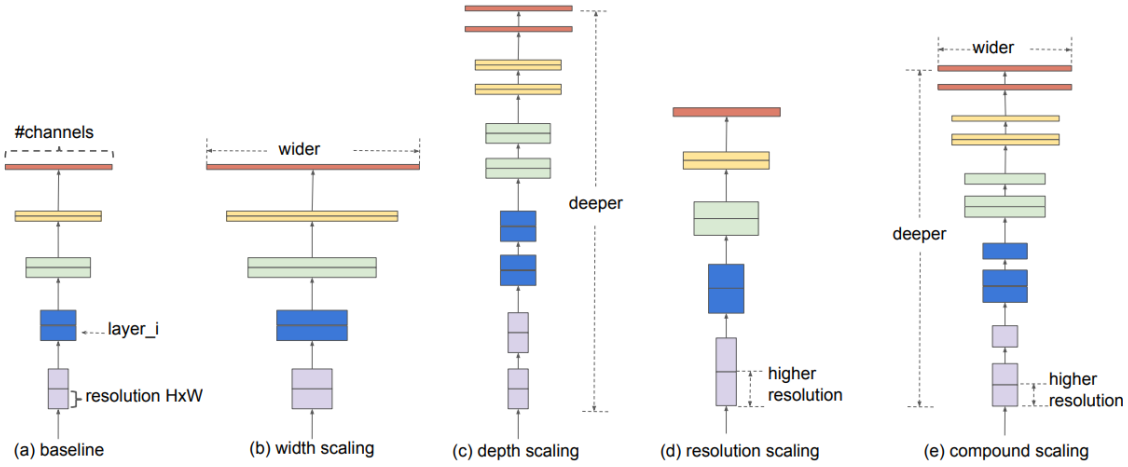
δίνει καλά αποτελέσματα μόνο στην αναγνώριση περιστρεφόμενων προσώπων. Για να αποφευχθεί αυτό είναι σημαντικό να χρησιμοποιούμε μια ποικιλία τεχνικών data augmentation και επίσης να διασφαλίζουμε ότι τα επαυξημένα δεδομένα είναι αντιπροσωπευτικά των δεδομένων του πραγματικού κόσμου στα οποία θα χρησιμοποιηθεί το μοντέλο.

Υπάρχουν διάφορα οφέλη από τη χρήση του data augmentation στη μηχανική μάθηση. Πρώτον, μπορεί να συμβάλει στη μείωση της υπερπροσαρμογής, η οποία αποτελεί κοινό πρόβλημα κατά την εκπαίδευση μοντέλων μηχανικής μάθησης. Με τη δημιουργία πρόσθετων παραδειγμάτων εκπαίδευσης, η επαύξηση δεδομένων επιτρέπει στο μοντέλο να δει πιο διαφορετικές εκδοχές των δεδομένων, γεγονός που μπορεί να το βοηθήσει να γενικεύσει καλύτερα σε νέα, αθέατα δεδομένα. Δεύτερον, μπορεί να χρησιμοποιηθεί για τη βελτίωση της στιβαρώτητας ενός μοντέλου με τη δημιουργία παραδειγμάτων που διαφέρουν ελαφρώς από τα αρχικά δεδομένα. Αυτό μπορεί να βοηθήσει το μοντέλο να είναι πιο ανθεκτικό σε μικρές αλλαγές στα δεδομένα εισόδου, κάτι που μπορεί να είναι σημαντικό σε εφαρμογές του πραγματικού κόσμου όπου τα δεδομένα μπορεί να μην είναι πάντα απόλυτα καθαρά και ομοιόμορφα.

Σχήμα 2.12. Παράδειγμα data augmentation σε δεδομένα εικόνας. [15]

2.2.11 EfficientNet

Τα EfficientNets είναι μια οικογένεια μοντέλων συνελικτικών νευρωνικών δικτύων (CNN) που αναπτύχθηκαν από τους ερευνητές της Google Mingxing Tan και Quoc V. Le το 2019. Έχουν σχεδιαστεί για να επιτυγχάνουν ακρίβεια τελευταίας τεχνολογίας, ενώ παράλληλα είναι αποδοτικά ως προς τους πόρους, καθιστώντας τα κατάλληλα για ανάπτυξη σε συσκευές με περιορισμένη υπολογιστική ισχύ. Ένας βασικός παράγοντας στο σχεδιασμό τους είναι η χρήση μιας μεθόδου σύνθετης κλιμάκωσης για τη συστηματική αύξηση των διαστάσεων του δικτύου. Η βασική ιδέα είναι η κλιμάκωση των διαστάσεων του δικτύου με συνεπή τρόπο και όχι η τυχαία αύξηση του αριθμού των στρωμάτων ή του αριθμού των φίλτρων σε κάθε στρώμα. Αυτό επιτυγχάνεται με την ταυτόχρονη κλιμάκωση του βάθους, του πλάτους και της ανάλυσης του δικτύου, σχήμα 2.12. Μια από τις βασικές καινοτομίες των EfficientNets είναι η χρήση μιας διαχωρίσιμης κατά βάθος λειτουργία συνέλιξης, η οποία επιτρέπει στο μοντέλο να εκτελεί τον ίδιο υπολογισμό με μια τυπική λειτουργία συνέλιξης χρησιμοποιώντας λιγότερες παραμέτρους και υπολογισμούς. Αυτό έχει ως αποτέλεσμα ένα πιο αποδοτικό μοντέλο που μπορεί να εκπαιδευτεί ταχύτερα και με λιγότερα δεδομένα.

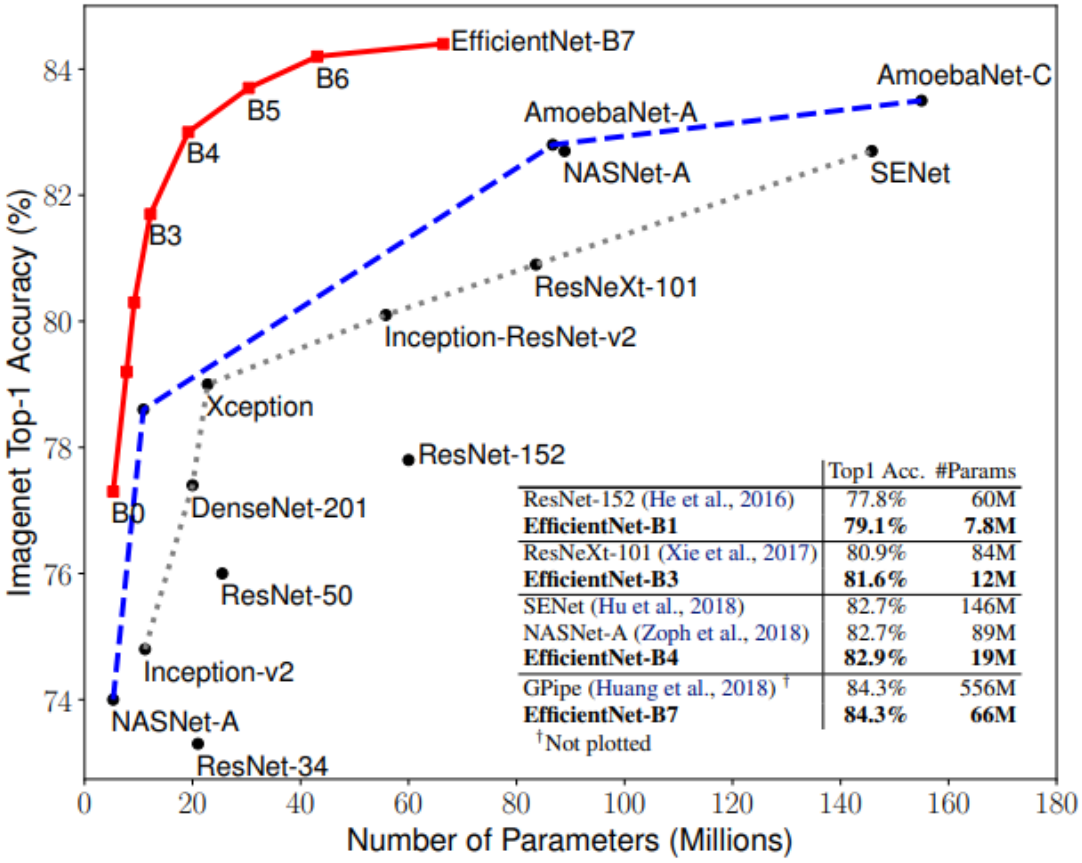


Σχήμα 2.13. a) Αναπαράσταση ενός υποθετικού βαθιού νευρωνικού δικτύου. b) Το ίδιο δίκτυο με αύξηση κατά πλάτος. c) Με αύξηση κατά βάθος d) Με αύξηση την ανάλυσης. e) Ταυτόχρονη αύξηση – κλιμάκωση σε όλες τις «διαστάσεις». [16]

Εκτός από τη χρήση των διαχωρίσιμων κατά βάθος συνελίξεων, τα EfficientNets χρησιμοποιούν επίσης μια σειρά από άλλες τεχνικές για τη βελτίωση της αποδοτικότητας, όπως:

- Μηχανισμοί προσοχής: Επιτρέπουν στο μοντέλο να εστιάζει στα πιο σημαντικά μέρη της εισόδου, αντί να επεξεργάζεται εξίσου ολόκληρη την είσοδο.
- Επίπεδα συμφόρησης: Μειώνουν τον αριθμό των καναλιών στην έξοδο ενός στρώματος συνελίξεων, με αποτέλεσμα να μειώνεται ο αριθμός των παραμέτρων και των υπολογισμών.
- Συνδέσεις παράλειψης: Επιτρέπουν στο μοντέλο να παρακάμπτει τα στρώματα, επιτρέποντας στις κλίσεις (gradients) να ρέουν πιο άμεσα από τα στρώματα εξόδου στα προηγούμενα στρώματα του δικτύου. Αυτό μπορεί να βελτιώσει την αποτελεσματικότητα της εκπαίδευσης και να μειώσει τον κίνδυνο υπερπροσαρμογής.

Τα δίκτυα EfficientNets έχουν επιτύχει κορυφαίες επιδόσεις σε μια σειρά από benchmarks (σημείων αναφοράς), συμπεριλαμβανομένων των ImageNet, COCO και άλλων. Έχουν επίσης χρησιμοποιηθεί για την επίτευξη ισχυρών επιδόσεων σε διάφορες πρακτικές εφαρμογές, όπως η ανίχνευση αντικειμένων και η γλωσσική μετάφραση. Ένα από τα βασικά πλεονεκτήματα των EfficientNets είναι η αποδοτικότητα τους. Είναι σε θέση να επιτυγχάνουν υψηλή ακρίβεια με σημαντικά λιγότερες παραμέτρους και υπολογισμούς από άλλα μοντέλα τελευταίας τεχνολογίας, όπως φαίνεται και στο σχήμα 2.13, γεγονός που τα καθιστά κατάλληλα για ανάπτυξη σε συσκευές με περιορισμένους υπολογιστικούς πόρους, όπως για παράδειγμα τα smartphones.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Σχήμα 2.14. Γραφική αναπαράσταση της σύγκρισης της σχέσεις απόδοσης και πολυπλοκότητας (αριθμός παραμέτρων), κορυφαίων αρχιτεκτονικών βαθιών νευρωνικών δικτύων. [16]

2.2.12 Προ-εκπαιδευμένα Μοντέλα (Pre-trained Models) – Μάθηση Μεταφοράς (Transfer Learning)

Τα προ-εκπαιδευμένα μοντέλα βαθιάς μάθησης είναι μοντέλα νευρωνικών δικτύων που έχουν ήδη εκπαιδευτεί σε ένα μεγάλο σύνολο δεδομένων και είναι διαθέσιμα προς χρήση. Αυτά τα μοντέλα έχουν ήδη μάθει να αναγνωρίζουν μοτίβα και χαρακτηριστικά στα δεδομένα στα οποία εκπαιδεύτηκαν και μπορούν να χρησιμοποιηθούν για να κάνουν προβλέψεις ή να ταξινομήσουν νέα δεδομένα. Τα προ-εκπαιδευμένα μοντέλα είναι ιδιαίτερα χρήσιμα όταν θέλουμε να χρησιμοποιήσουμε τεχνικές βαθιάς μάθησης, αλλά δεν έχουμε στη διάθεσή μας μεγάλο όγκο δεδομένων για να εκπαιδεύσουμε ένα μοντέλο από την αρχή. Μπορούν επίσης να είναι χρήσιμα όταν θέλουμε να χρησιμοποιήσουμε

ένα μοντέλο τελευταίας τεχνολογίας, αλλά δεν έχουμε τους πόρους ή την τεχνογνωσία για να το εκπαιδεύσουμε μόνοι μας. Υπάρχουν πολλά διαθέσιμα προ-εκπαιδευμένα μοντέλα, συμπεριλαμβανομένων μοντέλων που έχουν εκπαιδευτεί σε δημοφιλή σύνολα δεδομένων εικόνας, όπως το ImageNet και το COCO, και μοντέλων που έχουν εκπαιδευτεί σε μεγάλα σύνολα δεδομένων κειμένου, όπως η Wikipedia. Αυτά τα μοντέλα μπορούν να χρησιμοποιηθούν για εργασίες όπως η ταξινόμηση εικόνων, η ανίχνευση αντικειμένων και η γλωσσική μετάφραση, μεταξύ άλλων.

Για να χρησιμοποιήσουμε ένα προ-εκπαιδευμένο μοντέλο, συνήθως κατεβάζουμε τα βάρη και την αρχιτεκτονική του μοντέλου από ένα αποθετήριο ή έναν ιστότοπο και στη συνέχεια μπορούμε να το χρησιμοποιήσουμε για να κάνουμε προβλέψεις. Ανάλογα με το συγκεκριμένο μοντέλο και την εφαρμογή, μπορεί επίσης να χρειαστεί να τελειοποιήσουμε το μοντέλο (fine-tuning) εκπαιδεύοντάς το σε ένα μικρότερο σύνολο δεδομένων ειδικά για την συγκεκριμένη εφαρμογή. Γενικά αυτή η διαδικασία είναι γνωστή ως transfer learning (μάθηση μεταφοράς).

Υπάρχουν δύο κύριοι τύποι μάθησης μεταφοράς, εξαγωγή χαρακτηριστικών και λεπτομερής ρύθμιση.

- Η εξαγωγή χαρακτηριστικών (future extraction) περιλαμβάνει τη χρήση του προ-εκπαιδευμένου μοντέλου ως σταθερού εξαγωγέα χαρακτηριστικών, όπου η έξοδος των επιπέδων του προ-εκπαιδευμένου μοντέλου τροφοδοτείται ως είσοδος σε ένα νέο μοντέλο που εκπαιδεύεται για να εκτελέσει τη νέα εργασία.
- Η λεπτομερής ρύθμιση (fine-tuning) περιλαμβάνει το «ξεπάγωμα» ορισμένων από τα στρώματα του προ-εκπαιδευμένου μοντέλου και την εκπαίδευσή τους στο νέο σύνολο δεδομένων μαζί με τα νεοπροστιθέμενα στρώματα.

Ένα από τα βασικά πλεονεκτήματα της χρήσης προ-εκπαιδευμένων μοντέλων είναι ότι μπορούν να ρυθμιστούν λεπτομερώς ώστε να επιτύχουν καλή απόδοση σε μια συγκεκριμένη εργασία με σχετικά μικρό όγκο δεδομένων με ετικέτες (labels). Αυτό οφείλεται στο γεγονός ότι το μοντέλο έχει ήδη μάθει πολλά από τα χαρακτηριστικά και τα μοτίβα που είναι σχετικά με τη νέα εργασία. Επιπλέον έχει αποδηχθεί ότι τα προ-εκπαιδευμένα μοντέλα μπορούν να γενικευτούν καλύτερα σε νέες εφαρμογές

Κεφάλαιο 3ο: Εργαλεία

3.1 Εισαγωγή

Τα δύο βασικά εργαλεία που χρησιμοποιήθηκαν στην εργασία είναι το περιβάλλον προσομοίωσης (βιντεοπαιχνίδι αγώνων αυτοκινήτων) Assetto Corsa και η γλώσσα προγραμματισμού Python.

Το assetto corsa είναι ένας από τους δημοφιλέστερους προσομοιωτές αγώνων (racing simulator). Ο λόγος που επιλέχθηκε ένα παιχνίδι σαν περιβάλλον προσομοίωσης είναι λόγω της υπεροχής τους όσο αφορά την προσομοίωση της συμπεριφοράς του αυτοκινήτου στον εικονικό κόσμο και της αλληλεπίδρασης του με το οδόστρωμα. Βιντεοπαιχνίδια όπως το Assetto Corsa, κάνουν πολύ καλύτερη δουλειά σε αυτό το κομμάτι σε σύγκριση με τους περισσότερους δωρεάν προσομοιωτές οδήγησης, όπως για παράδειγμα το CARLA[17] ή το PGDRIVE[18], που είναι φτιαγμένοι ειδικά για δοκιμές αλγορίθμων αυτόνομης οδήγησης και άλλων παρόμοιων εφαρμογών. Το πλεονέκτημα βέβαια αυτών των πλατφόρμων, μιας και είναι σχεδιασμένες ειδικά για τέτοιους σκοπούς, είναι ότι προσφέρουν διάφορα εργαλεία και πληροφορίες, που είναι χρήσιμα ή ακόμα και απαραίτητα για κάποιες εφαρμογές, π.χ. αισθητήρες ταχύτητας, αισθητήρες υπερύθρων, αισθητήρες κλίσης, προσομοίωση κυκλοφορίας αυτοκινήτων σε αστικό δίκτυο κλπ. Στην περίπτωση της συγκεκριμένης εργασίας το μόνο αισθητήριο που χρειάζεται είναι η κάμερα, ενώ δεν μας ενδιαφέρει η αλληλεπίδραση με άλλα αυτοκίνητα, σήματα κλπ.

Η python είναι μία από τις δημοφιλέστερες γλώσσες προγραμματισμού γενικού σκοπού και με διαφορά η πιο δημοφιλής για Μηχανική Μάθηση, με μια τεράστια συλλογή από βιβλιοθήκες η οποία διαρκώς αυξάνεται και βελτιώνεται. Είναι μια γλώσσα υψηλού επιπέδου, σε σύγκριση με άλλες δημοφιλείς γλώσσες όπως, C++, Java, C# κλπ, με αποτέλεσμα να είναι πολύ εύκολη στη γραφή και στην ανάγνωση. Αυτό βοηθάει το χρήστη να επικεντρώνεται στο πρόβλημα που θέλει να λύσει και όχι στην ίδια τη γλώσσα, στο πως δηλαδή να γράψει αυτό που θέλει.

3.2 Assetto Corsa

Τι ακριβώς είναι το Assetto Corsa?

Σύμφωνα με τη Wikipedia [19] «Το Assetto Corsa είναι ένας προσομοιωτής αγώνων που προσπαθεί να προσφέρει μια ρεαλιστική εμπειρία οδήγησης με μια ποικιλία αυτοκινήτων δρόμου και αγώνων μέσω λεπτομερούς φυσικής και προσομοίωσης ελαστικών σε πίστες αγώνων που αναδημιουργήθηκαν με τεχνολογία σάρωσης λέιζερ. Υποστηρίζει μια σειρά περιφερειακών συσκευών όπως ποντίκι, πληκτρολόγιο, τροχούς, τηλεχειριστήρια, κ.α. Το λογισμικό μπορεί να επεκταθεί μέσω τροποποιημένου περιεχομένου τρίτων κατασκευαστών.»

Γιατί από όλα τα βιντεοπαιχνίδια επιλέχθηκε το Assetto Corsa;

Υπάρχουν αμέτρητα βιντεοπαιχνίδια με οδήγηση αυτοκινήτων, κάποια είναι φτιαγμένα καθαρά γι' αυτό, όπως είναι τα παιχνίδια αγώνων, άλλα περιέχουν τη δυνατότητα οδήγησης αυτοκινήτων μεταξύ άλλων. Η εργασία αυτή πρακτικά θα μπορούσε να γίνει χρησιμοποιώντας οποιοδήποτε παιχνίδι, αλλά το AC διαθέτει κάποια χαρακτηριστικά τα οποία το ξεχωρίζουν.

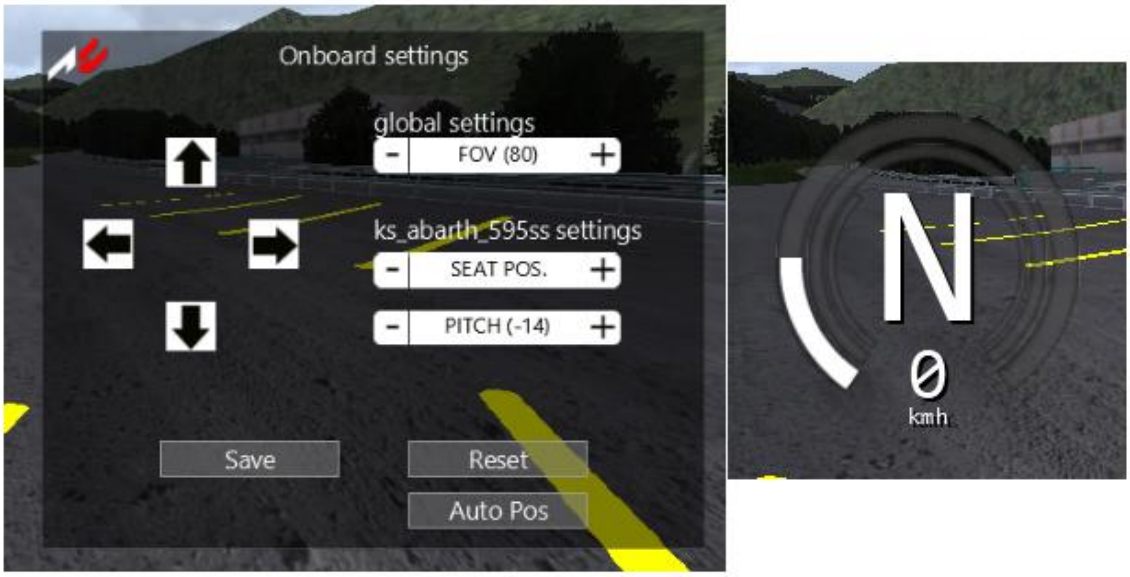
- Ρεαλιστική εμπειρία οδήγησης, λεπτομερής προσομοίωση των κανόνων της φυσικής. Το AC είναι μεταξύ των πιο επιτυχημένων προσομοιωτών οδήγησης σε αυτό το κομμάτι, και εφόσον η εργασία επικεντρώνεται καθαρά στο κομμάτι του χειρισμού του αυτοκινήτου, με όσο το δυνατόν αν όχι γρήγορες, τουλάχιστον ρεαλιστικές ταχύτητες.
- Ποικιλία αυτοκινήτων δρόμου και πίστες αγώνων, ή κομμάτια οδικού δικτύου, που αναδημιουργήθηκαν με τεχνολογία σάρωσης λέιζερ. Το AC περιλαμβάνει μία τεράστια συλλογή, η οποία διαρκώς αναπτύσσεται, από ψηφιοποιημένους πραγματικούς δρόμους. Αυτό μας δίνει τη δυνατότητα να συλλέξουμε δεδομένα και να δοκιμάσουμε τον αλγόριθμο σε πολλούς διαφορετικούς δρόμους με διαφορετικά τοπία, τα οποία επιπλέον είναι πολύ ρεαλιστικά αναδημιουργημένα, Εικόνες 3.1 – 3.4.
- Προσομοίωση συστημάτων προηγμένης υποβοήθησης οδηγού (ADAS), όπως σύστημα αντιμπλοκαρίσματος τροχών, σύστημα ελέγχου πρόσφυσης (Traction Control) και ηλεκτρονικό σύστημα ευστάθειας (ESC).
- Το AC διαθέτει αρκετά εργαλεία, για ρυθμίσεις και εξατομίκευση του περιβάλλοντος. Δύο παραδείγματα σημαντικά που χρησιμοποιούνται στα πλαίσια της εργασίας, είναι η δυνατότητα μετακίνησης της κάμερας (view model), χειροκίνητα και με μεγάλη ελευθερία και το ταμπλό με πληροφορίες για το αυτοκίνητο κατά την οδήγηση, όπως είναι η ένδειξη της ταχύτητας, εικόνα 3.5.

Εικόνα 3.1. Αναδημιουργημένος δρόμος με τεχνολογία σάρωσης λέιζερ.

Εικόνα 3.2. Αναδημιουργημένος δρόμος με τεχνολογία σάρωσης λέιζερ.

Εικόνα 3.3. Αναδημιουργημένος δρόμος με τεχνολογία σάρωσης λέιζερ.

Εικόνα 3.4. Αναδημιουργημένος δρόμος με τεχνολογία σάρωσης λέιζερ.



Εικόνα 3.5, ρυθμίσης κάμερας και ένδειξη ταχύτητας.

3.3 Python

Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού γενικού σκοπού που χρησιμοποιείται ευρέως στη βιομηχανία λογισμικού. Δημιουργήθηκε στα τέλη της δεκαετίας του 1980 από τον Guido van Rossum, έναν Ολλανδό προγραμματιστή, ο οποίος προσπαθούσε να δημιουργήσει μια πιο ευανάγνωστη και φιλική προς το χρήστη εναλλακτική λύση σε σχέση με άλλες γλώσσες προγραμματισμού της εποχής. Από την ίδρυσή της, έχει γίνει μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο, με εκατομμύρια χρήστες και χιλιάδες βιβλιοθήκες και πλαίσια.

Ένας από τους κύριους λόγους για τους οποίους η Python έχει γίνει τόσο δημοφιλής είναι η απλότητα και η αναγνωσιμότητά της. Σε αντίθεση με πολλές άλλες γλώσσες προγραμματισμού, χρησιμοποιεί ένα συντακτικό που είναι πολύ πιο κοντά στην ανθρώπινη γλώσσα, γεγονός που διευκολύνει τους αρχάριους να μάθουν και τους έμπειρους προγραμματιστές να διαβάζουν και να κατανοούν κώδικα που έχουν γράψει άλλοι. Επίσης διαθέτει μια μεγάλη ενσωματωμένη βιβλιοθήκη που περιλαμβάνει ένα ευρύ φάσμα συναρτήσεων και modules, γεγονός που καθιστά εύκολη την εκτέλεση κοινών εργασιών χωρίς να χρειάζεται να γράψει πολύ κώδικα ο χρήστης.

Ένας άλλος λόγος για τον οποίο η Python είναι τόσο δημοφιλής, είναι η ευελιξία της. Χρησιμοποιείται σε ένα ευρύ φάσμα εφαρμογών, συμπεριλαμβανομένης της ανάπτυξης ιστοσελίδων, της επιστημονικής πληροφορικής, της ανάλυσης δεδομένων, της τεχνητής νοημοσύνης και πολλών άλλων. Διαθέτει επίσης μια μεγάλη και ενεργή κοινότητα προγραμματιστών που συνεισφέρουν στη γλώσσα και δημιουργούν βιβλιοθήκες και πλαίσια για διάφορες εργασίες, γεγονός που διευκολύνει τους προγραμματιστές να δημιουργούν πολύπλοκες εφαρμογές με ελάχιστη προσπάθεια.

Ένα από τα βασικά της χαρακτηριστικά είναι η υποστήριξή της για αντικειμενοστραφή προγραμματισμό (Object Oriented Programming / OOP). Ο OOP είναι ένα παράδειγμα προγραμματισμού που οργανώνει τον κώδικα σε "αντικείμενα" που αναπαριστούν οντότητες του πραγματικού κόσμου και τα χαρακτηριστικά και τις συμπεριφορές τους. Αυτό διευκολύνει τη συγγραφή και τη συντήρηση του κώδικα, καθώς τα αντικείμενα μπορούν να επαναχρησιμοποιηθούν και να τροποποιηθούν εύκολα. Επιπλέον όμως υποστηρίζει και άλλα παραδείγματα προγραμματισμού, όπως ο διαδικαστικός προγραμματισμός και ο λειτουργικός προγραμματισμός, γεγονός που παρέχει στους προγραμματιστές ευελιξία στον τρόπο που επιλέγουν να δομούν τον κώδικά τους.

Η Python διαθέτει επίσης ορισμένα προηγμένα χαρακτηριστικά που την καθιστούν μια ισχυρή γλώσσα για την ανάλυση δεδομένων και τους επιστημονικούς υπολογισμούς. Έχει ενσωματωμένη υποστήριξη για αριθμητικές πράξεις και διαθέτει μια σειρά από βιβλιοθήκες, όπως η NumPy και η Pandas, που διευκολύνουν την εκτέλεση πολύπλοκων υπολογισμών και τη διαχείριση δεδομένων. Διαθέτει επίσης μεγάλο αριθμό βιβλιοθηκών για την οπτικοποίηση δεδομένων, όπως οι Matplotlib και Seaborn, οι οποίες διευκολύνουν τη δημιουργία διαγραμμάτων και γραφικών παραστάσεων επαγγελματικής ποιότητας.

Φυσικά είναι και δημοφιλής στον τομέα της τεχνητής νοημοσύνης και της μηχανικής μάθησης. Καθώς διαθέτει πολλές βιβλιοθήκες, όπως η TensorFlow και η scikit-learn, που διευκολύνουν τη δημιουργία και την εκπαίδευση μοντέλων μηχανικής μάθησης. Χρησιμοποιείται επίσης στην επεξεργασία φυσικής γλώσσας, δηλαδή στην ικανότητα των υπολογιστών να κατανοούν και να αναλύουν την ανθρώπινη γλώσσα, πράγμα που έχει ένα ευρύ φάσμα εφαρμογών, όπως η γλωσσική μετάφραση, η ανάλυση συναισθήματος και τα chatbots.

Εκτός από τις τεχνικές δυνατότητές της, η Python διαθέτει και μια σειρά άλλων πλεονεκτημάτων που την καθιστούν καλή επιλογή για πολλές εφαρμογές. Διαθέτει μια μεγάλη και ενεργή κοινότητα προγραμματιστών που συνεισφέρουν στη γλώσσα και παρέχουν υποστήριξη στους χρήστες. Είναι επίσης ανοικτού κώδικα, πράγμα που σημαίνει ότι ο πηγαίος κώδικας είναι διαθέσιμος σε οποιονδήποτε και μπορεί να τροποποιηθεί και να διανεμηθεί ελεύθερα. Αυτό διευκολύνει τους προγραμματιστές να βασιστούν στο έργο άλλων και συμβάλλει στη διασφάλιση ότι η γλώσσα θα συνεχίσει να εξελίσσεται και να βελτιώνεται.

Παρά τα πολλά πλεονεκτήματά της όμως, δεν είναι τέλεια. Μία από τις κύριες επικρίσεις της γλώσσας είναι ότι συνήθως είναι πιο αργή από άλλες γλώσσες, όπως η C ή η C++, οι οποίες έχουν σχεδιαστεί για εργασίες χαμηλού επιπέδου και είναι πιο κοντά στο υλικό (hardware). Αυτό μπορεί να κάνει την Python λιγότερο κατάλληλη για ορισμένους τύπους εφαρμογών, όπως συστήματα πραγματικού χρόνου ή εφαρμογές που απαιτούν μεγάλη υπολογιστική ισχύ. Πράγματι αυτός ο περιορισμός στην ταχύτητα είναι κάτι που μας επηρεάζει στην εφαρμογή της συγκεκριμένης εργασίας.

Συνολικά, η Python είναι μια ισχυρή και ευέλικτη γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως σε μια ποικιλία εφαρμογών. Η απλότητα και η αναγνωσιμότητά της την καθιστούν καλή επιλογή για αρχάριους, ενώ τα προηγμένα χαρακτηριστικά και οι βιβλιοθήκες της την καθιστούν κατάλληλη για πιο σύνθετες εργασίες.

3.4 Βιβλιοθήκες της Python

Η python όπως αναφέρθηκε και παραπάνω έχει μια τεράστια συλλογή βιβλιοθηκών, για μια πολύ μεγάλη ποικιλία εφαρμογών. Στα πλαίσια της εργασίας για το κομμάτι της μηχανικής μάθησης αλλά και για το κομμάτι της δημιουργίας του User Interface, χρησιμοποιήθηκαν οι εξής βιβλιοθήκες:

- Os
- Time
- Cv2
- Numpy
- Tensorflow
- Keras
- Threading
- Win32api
- Win32gui
- Win32ui
- Ctypes
- Pandas
- Struct
- Pyqt5
- Re
- Subprocess
- Sys
- Multiprocessing?
- Shitil?
- Webbrowser
- Pynput

Στη συνέχεια θα αναλυθεί το τι κάνει η κάθε μια ξεχωριστά.

3.4.1 Os

Η βιβλιοθήκη os είναι μια ενότητα της τυπικής βιβλιοθήκης της Python που παρέχει λειτουργίες για την αλληλεπίδραση με το λειτουργικό σύστημα. Μας επιτρέπει να κάνουμε πράγματα όπως η δημιουργία και η διαγραφή καταλόγων, η μετακίνηση και η αντιγραφή αρχείων, η λήψη πληροφοριών σχετικά με το σύστημα, όπως ο τρέχων κατάλογος εργασίας και ο διαθέσιμος χώρος στο δίσκο και η γέννηση νέων διεργασιών. [20]

Ακολουθούν μερικά παραδείγματα συναρτήσεων της βιβλιοθήκης os:

- os.getcwd(): Επιστρέφει τον τρέχοντα κατάλογο εργασίας.
- os.listdir(path): Επιστρέφει μια λίστα αρχείων και καταλόγων στην καθορισμένη διαδρομή.
- os.mkdir(path): Δημιουργεί έναν νέο κατάλογο στην καθορισμένη διαδρομή.
- os.rename(src, dst): Μετονομάζει το αρχείο ή τον κατάλογο στη διαδρομή src στη διαδρομή dst.
- os.rmdir(path): Διαγράφει τον κατάλογο στην καθορισμένη διαδρομή, η οποία πρέπει να είναι κενή.

3.4.2 Time

Η βιβλιοθήκη time είναι μια ενότητα της τυπικής βιβλιοθήκης της Python που παρέχει συναρτήσεις για την εργασία με το χρόνο. Μας επιτρέπει να κάνουμε πράγματα όπως η μέτρηση του χρόνου που έχει παρέλθει, η μετατροπή μεταξύ διαφορετικών μορφών χρόνου και ο «ύπνος», η παύση πρακτικά του προγράμματος, για ένα καθορισμένο χρονικό διάστημα. [21]

Ακολουθούν τα δύο παραδείγματα συναρτήσεων που χρησιμοποιήθηκαν στη συγκεκριμένη εργασία:

- time.time(): Επιστρέφει την τρέχουσα ώρα σε δευτερόλεπτα από την εποχή (η εποχή είναι ένα προκαθορισμένο χρονικό σημείο, συνήθως η αρχή του έτους 1970).
- time.sleep(seconds): Διακόπτει το πρόγραμμα για τον καθορισμένο αριθμό δευτερολέπτων.

3.4.3 Sys

Η βιβλιοθήκη sys είναι μια ενότητα της τυπικής βιβλιοθήκης της Python που παρέχει πρόσβαση σε μεταβλητές και συναρτήσεις που σχετίζονται με τον διερμηνέα της Python. Μας επιτρέπει να κάνουμε πράγματα όπως η πρόσβαση σε ορίσματα της γραμμής εντολών, να «εργαζόμαστε» με τη διαδρομή και το πρόθεμα (prefix) του διερμηνευτή και να τροποποιούμε τη συμπεριφορά του.[22]

Ακολουθούν μερικά παραδείγματα συναρτήσεων και μεταβλητών της βιβλιοθήκης sys:

- sys.argv: Μια λίστα από συμβολοσειρές που αντιπροσωπεύουν τα ορίσματα της γραμμής εντολών που περνούν στο script (σενάριο). Το πρώτο στοιχείο της λίστας είναι το όνομα της ίδιας της δέσμης ενεργειών.
- sys.path: Μια λίστα συμβολοσειρών που αντιπροσωπεύει τη διαδρομή αναζήτησης για τα modules (ενότητες). Χρησιμοποιείται για την εύρεση module που εισάγονται σε ένα πρόγραμμα.
- sys.prefix: Μια συμβολοσειρά που αντιπροσωπεύει το πρόθεμα που χρησιμοποιείται για την εύρεση της τυπικής βιβλιοθήκης της Python.
- sys.exit(status): Προκαλεί την έξοδο του διερμηνευτή Python με τον καθορισμένο κωδικό κατάστασης.
- sys.stdout: Ένα αντικείμενο που μοιάζει με αρχείο και αναπαριστά την τυπική ροή εξόδου. Μπορούμε να το χρησιμοποιήσουμε για να γράψουμε έξοδο στην κονσόλα.

3.4.4 Re

Η βιβλιοθήκη re είναι μια ενότητα της τυπικής βιβλιοθήκης της Python που παρέχει συναρτήσεις για τη χρήση κανονικών εκφράσεων. Οι κανονικές εκφράσεις είναι ένας ισχυρός και συνοπτικός τρόπος για την αναπαράσταση μοτίβων σε συμβολοσειρές και η βιβλιοθήκη re μας επιτρέπει να αναζητούμε και να επεξεργαζόμαστε συμβολοσειρές που ταιριάζουν με ένα δεδομένο μοτίβο. [23]

Οι δύο πιο συνηθισμένες συναρτήσεις της βιβλιοθήκης re είναι οι εξής:

- re.search(pattern, string): Αναζητάει την πρώτη εμφάνιση ενός μοτίβου (pattern) σε μία συμβολοσειρά (string). Επιστρέφει ένα αντικείμενο Match αν βρεθεί μια αντιστοιχία, ή None αν δεν βρεθεί καμία αντιστοιχία.
- re.match(pattern, string): Προσπαθεί να ταιριάξει το μοτίβο με την αρχή της συμβολοσειράς. Επιστρέφει ένα αντικείμενο Match αν βρεθεί αντιστοίχιση, ή None αν δεν βρεθεί αντιστοίχιση.

3.4.5 Webbrowser

Η βιβλιοθήκη webbrowser στην Python είναι μια ενότητα που μας επιτρέπει να ανοίγουμε ένα πρόγραμμα περιήγησης στο διαδίκτυο από τον κώδικα της Python. Μπορούμε να τη χρησιμοποιήσουμε για να ανοίξουμε μια διεύθυνση URL στο προεπιλεγμένο πρόγραμμα περιήγησης ιστού (browser) του χρήστη ή να ανοίξουμε ένα συγκεκριμένο browser, διευκρινίζοντας το με την εντολή, webbrowser.get(‘chromium’).open(‘url_name’). [24]

3.4.6 Subprocess

Η βιβλιοθήκη subprocess είναι μια ενσωματωμένη βιβλιοθήκη της Python που μας επιτρέπει να ξεκινάμε νέες διεργασίες μέσα από τον κώδικα, καθώς και να ελέγχουμε και να αλληλεπιδράμε με την είσοδο και την έξοδό τους. Μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη subprocess για να εκτελούμε εξωτερικά προγράμματα από τον κώδικα Python και να καταγράφουμε την έξοδό τους. Αυτό είναι χρήσιμο για την αυτοματοποίηση εργασιών που περιλαμβάνουν την εκτέλεση βοηθητικών προγραμμάτων γραμμής εντολών ή άλλων εξωτερικών προγραμμάτων. [25]

3.4.7 Threading

Η βιβλιοθήκη threading είναι μια ενσωματωμένη βιβλιοθήκη της Python που παρέχει υποστήριξη για τη δημιουργία και τη διαχείριση threads. Ένα thread είναι μια ξεχωριστή ροή εκτέλεσης μέσα σε ένα πρόγραμμα, και η βιβλιοθήκη αυτή μας επιτρέπει να δημιουργούμε πολλαπλά threads για ταυτόχρονη εκτέλεση μέσα στο πρόγραμμα. Αυτό μπορεί να είναι χρήσιμο για τη βελτίωση της απόδοσης ορισμένων τύπων προγραμμάτων, επιτρέποντάς τους να εκτελούν εργασίες παράλληλα. Για παράδειγμα, αν έχουμε ένα πρόγραμμα που πρέπει να εκτελέσει μια βαριά εργασία που χρησιμοποιεί πολλούς πόρους ή παίρνει πολύ χρόνο, όπως η λήψη ενός μεγάλου αρχείου από το Διαδίκτυο, μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη threading για να δημιουργήσουμε ένα ξεχωριστό thread για την εκτέλεση αυτής της εργασίας, ενώ το κύριο νήμα του προγράμματος συνεχίζει να εκτελείται, έτσι το πρόγραμμα δεν χρειάζεται να περιμένει να τελειώσει αυτή η εργασία. Επιπλέον παρέχει άλλες δυνατότητες για τη διαχείριση νημάτων, όπως τη δυνατότητα ορισμού της

προτεραιότητας ενός thread ή την αναμονή για την ολοκλήρωση ενός thread πριν συνεχίσει το πρόγραμμα. [26]

3.4.8 Pywin32

Η pywin32 είναι μια βιβλιοθήκη που παρέχει τη δυνατότητα πρόσβασης και χρήσης του Windows Application Programming Interface (API, Διεπαφή Προγραμματισμού Εφαρμογών) μέσα από την Python. Έτσι ο χρήστης έχει στη διάθεση του μία μεγάλη ποικιλία από συναρτήσεις μέσω των οποίων μπορεί να κάνει πράγματα όπως: [27]

- Να διαβάσει και να γράψει στο registry.
- Να διαχειριστεί διεργασίες και threads.
- Να αλληλεπιδράσει με το σύστημα αρχείων.
- Να δημιουργήσει και να τροποποιήσει παράθυρα διαλόγου των Windows.
- Να εμφανίσει παράθυρα μηνυμάτων και άλλου τύπου ειδοποιήσεις.
- Να διαχειριστεί γεγονότα (events) όπως εισόδους από το ποντίκι και το πληκτρολόγιο.
- Να αλληλεπιδράσει με το clipboard. Το clipboard είναι το μέρος στον υπολογιστή όπου προσωρινά αποθηκεύονται αντιγραμμένα ή αποκομμένα δεδομένα.

3.4.9 Ctypes

Σύμφωνα με το επίσημο documentation της Python, το ctypes είναι μια ξένη βιβλιοθήκη συναρτήσεων που παρέχει τύπους δεδομένων συμβατούς με τη C και επιτρέπει την κλήση συναρτήσεων σε DLL ή κοινόχρηστες βιβλιοθήκες. Μια ξένη βιβλιοθήκη συναρτήσεων σημαίνει ότι ο κώδικας της Python μπορεί να καλεί συναρτήσεις της C χρησιμοποιώντας μόνο Python, χωρίς να απαιτούνται ειδικές ή προσαρμοσμένες επεκτάσεις. [28]

Μια βιβλιοθήκη δυναμικής σύνδεσης (Dynamic Link Library - DLL) είναι ένας τύπος αρχείου που περιέχει κώδικα και δεδομένα που μπορούν να χρησιμοποιηθούν από πολλά προγράμματα ταυτόχρονα. Τα DLL χρησιμοποιούνται συχνά για την παροχή κοινής λειτουργικότητας που μπορεί να διαμοιραστεί από πολλές εφαρμογές, όπως λειτουργίες δικτύωσης, στοιχεία γραφικής διεπαφής χρήστη ή μαθηματικούς αλγορίθμους. Αποθηκεύονται συνήθως σε μια κεντρική θέση στον υπολογιστή και φορτώνονται στη μνήμη ανάλογα με τις ανάγκες των προγραμμάτων που τα χρησιμοποιούν. Αυτό επιτρέπει σε πολλαπλά προγράμματα να μοιράζονται τον ίδιο κώδικα και τα ίδια δεδομένα, γεγονός που μπορεί να συμβάλει στη μείωση της απαιτούμενης μνήμης και του χώρου στο δίσκο του συστήματος. Τα DLL χρησιμοποιούνται στο λειτουργικό σύστημα Windows και έχουν την επέκταση αρχείου ".dll".

3.4.10 Pynput

Η βιβλιοθήκη pynput είναι μια third party βιβλιοθήκη της python για την παρακολούθηση και τον έλεγχο της εισόδου του χρήστη. Παρέχει ένα σύνολο συναρτήσεων για την πρόσβαση στην κατάσταση του πληκτρολογίου, του ποντικιού και άλλων συσκευών εισόδου, καθώς και για την προσομοίωση της εισόδου χρήστη. Μπορεί να χρησιμοποιηθεί για την αυτοματοποίηση εργασιών, τη δημιουργία προσαρμοσμένων διεπαφών χρήστη ή την ανάπτυξη εφαρμογών που πρέπει να ανταποκρίνονται σε εισόδους του χρήστη σε πραγματικό χρόνο. [29]

3.4.11 Shutil

Η βιβλιοθήκη shutil είναι μια ενσωματωμένη βιβλιοθήκη της Python για λειτουργίες υψηλού επιπέδου σε αρχεία. Παρέχει ένα σύνολο συναρτήσεων για την αντιγραφή, μετακίνηση και διαγραφή αρχείων, καθώς και για τη δημιουργία και διαγραφή καταλόγων. Έχει σχεδιαστεί για να είναι μια εναλλακτική, πιο φιλική προς το χρήστη, συγκριτικά με τη βιβλιοθήκη χαμηλότερου επιπέδου os, η οποία παρέχει παρόμοια λειτουργικότητα αλλά απαιτεί περισσότερο κώδικα για να εκτελέσει τις ίδιες εργασίες. Η βιβλιοθήκη shutil είναι ένα βολικό εργαλείο για την εκτέλεση κοινών λειτουργιών αρχείων στην Python και χρησιμοποιείται συχνά για εργασίες όπως η δημιουργία αντιγράφων ασφαλείας αρχείων, ο συγχρονισμός καταλόγων ή η αυτοματοποίηση εργασιών διαχείρισης αρχείων. [30]

3.4.12 Struct

Η βιβλιοθήκη struct είναι μια ενσωματωμένη βιβλιοθήκη της Python για τη μετατροπή μεταξύ τιμών της Python και δομών δεδομένων τύπου C. Παρέχει ένα σύνολο συναρτήσεων για το πακετάρισμα και την αποσυμπίεση δεδομένων σε συγκεκριμένη μορφή, οι οποίες μπορούν να χρησιμοποιηθούν για τη μετατροπή μεταξύ τιμών Python και δεδομένων αποθηκευμένων σε δυαδική μορφή, όπως αρχεία ή ροές δικτύου. Η βιβλιοθήκη struct χρησιμοποιείται συχνά για εργασίες όπως η ανάγνωση και εγγραφή δυαδικών αρχείων, η επικοινωνία με servers ή η αλληλεπίδραση με άλλα συστήματα που χρησιμοποιούν δυαδική αναπαράσταση για δεδομένα. [31]

3.4.13 Pyqt5

Η βιβλιοθήκη pyqt5 είναι ένα σύνολο συνδέσεων (bindings) της Python με το cross-platform πλαίσιο (framework) εφαρμογών Qt. Το Qt είναι ένα ολοκληρωμένο C++ framework ανάπτυξης εφαρμογών που περιλαμβάνει ένα ευρύ φάσμα εργαλείων και βιβλιοθηκών για τη δημιουργία γραφικών διεπαφών χρήστη, εφαρμογών βάσεων δεδομένων και πολλών άλλων. Η pyqt5 μας επιτρέπει να χρησιμοποιούμε το πλαίσιο Qt από την Python, χρησιμοποιώντας ένα σύνολο από bindings που παρέχουν πρόσβαση στο API της Qt. Υλοποιείται ως ένα σύνολο από Python modules που περιβάλλουν (wrapping) τις υποκείμενες βιβλιοθήκες Qt C++ και είναι διαθέσιμη σε όλες τις πλατφόρμες που υποστηρίζονται από την Qt. [32]

3.4.14 Cv2

Η cv2 είναι μια βιβλιοθήκη της Python με bindings για τη χρήση της βιβλιοθήκης OpenCV, μιας ανοιχτού κώδικα βιβλιοθήκης γραμμένης σε C++. Η OpenCV παρέχει μια διεπαφή στην python για πρόσβαση και χρήση των αλγορίθμων και των utilities της. Χρησιμοποιείται για εφαρμογές όπως η επεξεργασία εικόνων και βίντεο, η ανίχνευση αντικειμένων και η αναγνώριση εικόνων. [33]

3.4.15 Numpy

Το NumPy είναι το θεμελιώδες πακέτο για επιστημονικούς υπολογισμούς στην Python. Πρόκειται για μια βιβλιοθήκη που παρέχει ένα αντικείμενο πολυδιάστατων πινάκων, διάφορα παράγωγα αντικείμενα (όπως masked πίνακες και πίνακες) και μια σειρά από ρουτίνες για γρήγορες λειτουργίες σε πίνακες, συμπεριλαμβανομένων μαθηματικών, λογικών, χειρισμού σχημάτων, ταξινόμησης, επιλογής, εισόδου/εξόδου, διακριτών μετασχηματισμών Fourier, βασικής γραμμικής άλγεβρας, βασικών στατιστικών πράξεων, τυχαίας προσομοίωσης και πολλών άλλων.

Στον πυρήνα του πακέτου NumPy, βρίσκεται το αντικείμενο ndarray. Αυτό ενθυλακώνει n-διάστατους πίνακες ομοιογενών τύπων δεδομένων, με πολλές λειτουργίες να εκτελούνται σε μεταγλωττισμένο κώδικα για λόγους απόδοσης. Υπάρχουν αρκετές σημαντικές διαφορές μεταξύ των πινάκων NumPy και των τυπικών ακολουθιών της Python [34]:

- Οι πίνακες NumPy έχουν σταθερό μέγεθος κατά τη δημιουργία τους, σε αντίθεση με τις λίστες της Python (οι οποίες μπορούν να αυξάνονται δυναμικά). Η αλλαγή του μεγέθους ενός πίνακα ndarray θα δημιουργήσει έναν νέο πίνακα και θα διαγράψει τον αρχικό.
- Τα στοιχεία σε έναν πίνακα NumPy απαιτείται να είναι όλα του ίδιου τύπου δεδομένων και επομένως θα έχουν το ίδιο μέγεθος στη μνήμη. Η εξαίρεση είναι ότι μπορεί κανείς να έχει πίνακες από αντικείμενα (της Python, συμπεριλαμβανομένης της NumPy), επιτρέποντας έτσι πίνακες με στοιχεία διαφορετικού μεγέθους.
- Οι πίνακες NumPy διευκολύνουν προηγμένες μαθηματικές και άλλου τύπου πράξεις σε μεγάλους αριθμούς δεδομένων. Συνήθως, τέτοιες πράξεις εκτελούνται πιο αποτελεσματικά και με λιγότερο κώδικα από ό,τι είναι δυνατό με τη χρήση των ενσωματωμένων ακολουθιών της Python.
- Μια αυξανόμενη πληθώρα επιστημονικών και μαθηματικών πακέτων που βασίζονται στην Python χρησιμοποιούν πίνακες NumPy αν και αυτά συνήθως υποστηρίζουν είσοδο ακολουθιών Python, μετατρέπουν την είσοδο αυτή σε πίνακες NumPy πριν από την επεξεργασία και συχνά εξάγουν πίνακες NumPy. Με άλλα λόγια, για να χρησιμοποιήσουμε αποτελεσματικά ένα μεγάλο μέρος (ίσως ακόμη και το μεγαλύτερο μέρος) του σημερινού επιστημονικού/μαθηματικού λογισμικού που βασίζεται στην Python, δεν αρκεί να γνωρίζουμε μόνο πώς να χρησιμοποιήσουμε τους ενσωματωμένους τύπους ακολουθιών της Python πρέπει επίσης να γνωρίζουμε πώς να χρησιμοποιήσουμε πίνακες NumPy.
- Το TensorFlow που χρησιμοποιήθηκε στην εργασία για τη δημιουργία των νευρωνικών δικτύων, την εκπαίδευση, την ανάκληση και γενικά για ότι έχει να κάνει με το κομμάτι της μηχανικής μάθησης, όπως θα δούμε παρακάτω, είναι μια από τις περιπτώσεις που αναφέρθηκαν παραπάνω και βασίζεται στη NumPy για τη λειτουργία του.

3.4.16 TensorFlow

Το TensorFlow [35] είναι ένα framework ανοικτού κώδικα που χρησιμοποιείται για αριθμητικούς υπολογισμούς και μηχανική μάθηση. Αναπτύχθηκε από την Google και χρησιμοποιείται ευρέως στον τομέα της τεχνητής νοημοσύνης και της μηχανικής μάθησης. Το TensorFlow επιτρέπει στους προγραμματιστές να δημιουργούν και να εκπαιδεύουν μοντέλα μηχανικής μάθησης χρησιμοποιώντας μια ποικιλία διαφορετικών γλωσσών προγραμματισμού, συμπεριλαμβανομένης της Python, παρόλο που είναι γραμμένο σε C++ με σκοπό τη μέγιστη απόδοση. Έχει σχεδιαστεί για να είναι ευέλικτο, αποδοτικό και επεκτάσιμο, καθιστώντας το μια δημοφιλή επιλογή για ένα ευρύ φάσμα εργασιών μηχανικής μάθησης. Βασίζεται στην ιδέα των tensors (τανυστών), οι οποίοι είναι πολυδιάστατοι πίνακες που χρησιμοποιούνται για την αναπαράσταση δεδομένων. Αυτοί οι tensors μπορούν να

επεξεργαστούν και να μετασχηματιστούν χρησιμοποιώντας μια σειρά από "ops" (operations - λειτουργίες), οι οποίες εκτελούν μαθηματικές πράξεις.

Μπορεί να χρησιμοποιηθεί για ένα ευρύ φάσμα εργασιών μηχανικής μάθησης, όπως ταξινόμηση (classification), παλινδρόμηση, (regression), ομαδοποίηση (clustering) και μείωση διαστάσεων. Παρέχει μια ποικιλία διαφορετικών στρωμάτων (layer), συναρτήσεων και optimizer (βελτιστοποιητών) που μπορούν να χρησιμοποιηθούν για τη δημιουργία και την εκπαίδευση μοντέλων μηχανικής μάθησης, καθώς και εργαλεία για την αξιολόγηση και τη σύγκριση της απόδοσης διαφορετικών μοντέλων.

Το TensorFlow παρέχει τόσο υψηλού όσο και χαμηλού επιπέδου API για την κατασκευή μοντέλων μηχανικής μάθησης. Τα API υψηλού επιπέδου, όπως τα Keras και TensorFlow Estimators, καθιστούν εύκολη τη δημιουργία και την εκπαίδευση μοντέλων με τη χρήση προκαθορισμένων layer και συναρτήσεων. Τα API χαμηλού επιπέδου, όπως το TensorFlow Core API, παρέχουν στους προγραμματιστές μεγαλύτερο έλεγχο στη διαδικασία κατασκευής μοντέλων και τους επιτρέπουν να υλοποιούν προσαρμοσμένα layer και συναρτήσεις.

Μπορεί να χρησιμοποιηθεί τόσο για εκπαίδευση όσο και ανάκληση (δηλαδή, χρήση ενός εκπαιδευμένου μοντέλου για να κάνει προβλέψεις σε νέα δεδομένα), ενώ παρέχει εργαλεία για την εξαγωγή εκπαιδευμένων μοντέλων σε περιβάλλοντα παραγωγής και για την ανάπτυξη αυτών των μοντέλων με μια ποικιλία διαφορετικών ρυθμίσεων, συμπεριλαμβανομένων των server, των κινητών συσκευών και του cloud.

3.4.17 TensorBoard

Το TensorBoard [36] είναι ένα διαδικτυακό εργαλείο που παρέχεται με το TensorFlow και μας επιτρέπει να απεικονίζουμε και να αναλύουμε την απόδοση των μοντέλων μηχανικής μάθησης. Παρέχει μια σειρά διαδραστικών αναπαραστάσεων και εργαλείων που μας βοηθούν να κατανοήσουμε πώς συμπεριφέρονται τα μοντέλα μας και πώς να βελτιώσουμε την απόδοσή τους. Μπορούμε να προβάλουμε μια ποικιλία διαφορετικών μετρήσεων και γραφημάτων που σχετίζονται με το μοντέλο, όπως η απώλεια εκπαίδευσης (training loss) και επαλήθευσης (validation loss), η ακρίβεια του σε διαφορετικά σύνολα δεδομένων και η κατανομή των βαρών και των biases. Μπορούμε επίσης να απεικονίσουμε τη δομή του μοντέλου και να δούμε πώς μεταβάλλεται με την πάροδο του χρόνου, καθώς και να προβάλουμε και να συγκρίνουμε την απόδοση διαφορετικών μοντέλων.

Όλες αυτές οι πληροφορίες και οι λειτουργίες είναι στη διάθεση μας μέσω ενός πολύ φιλικού και εύχρηστου γραφικού περιβάλλοντος, το οποίο μάλιστα ανοίγει σε browser κάνοντας το ακόμα πιο εύχρηστο. Στην εικόνα 3.5, βλέπουμε ένα παράδειγμα αναπαράστασης των training loss και validation loss, από ένα μοντέλο που εκπαιδεύσαμε. Στην αριστερή στήλη μπορούμε να επιλέξουμε οποιαδήποτε αποθηκευμένα μοντέλα που έχουμε εκπαιδεύσει στο παρελθόν και έχουν καταγραφεί από το TensorBoard και να τα συγκρίνουμε μεταξύ τους, ενώ μπορούμε και να κάνουμε μερικές ρυθμίσεις για τον τρόπο με τον οποίο απεικονίζεται το γράφημα. Αυτή είναι η πιο απλή, αλλά ίσως και η πιο χρήσιμη λειτουργία του TensorBoard.

Εικόνα 3.6. TensorBoard

Κεφάλαιο 4ο: Υλοποίηση

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα δούμε πως η παραπάνω θεωρία και τα εργαλεία που αναλύθηκαν θα χρησιμοποιηθούν για να πετύχουμε το στόχο, ο οποίος είναι να οδηγήσουμε ένα εικονικό αυτοκίνητο σε ένα εικονικό περιβάλλον χρησιμοποιώντας συνελικτικά νευρωνικά δίκτυα μαζί με απλή προγραμματιστική λογική.

Σε πρώτη φάση θα δείξω ποιο ακριβώς είναι το εικονικό περιβάλλον, ποιο είναι το αυτοκίνητο που επιλέχτηκε και γιατί επιλέχθηκε, ποιες «πίστες» χρησιμοποιήθηκαν για να εκπαιδευτούν και να δοκιμαστούν τα μοντέλα και ποια εργαλεία και ποιες πληροφορίες, που μας παρέχει το παιχνίδι χρησιμοποιήθηκαν. Επίσης θα δείξω με ποιον τρόπο χειρίζεται το πρόγραμμα το παιχνίδι, πως δηλαδή επικοινωνεί με το παιχνίδι, ποιες είναι οι έξοδοι του προγράμματος και πως φτάνουν στο παιχνδίδι, αλλά και τι και πως βλέπει το πρόγραμμα, ποιες είναι δηλαδή οι πληροφορίες εισόδου.

Στη συνέχεια θα δείξω όλη τη μεθοδολογία, όλα τα βήματα και όλη τη λογική που ακολούθησα για να πετύχω το στόχο της εργασίας, η οποία πρακτικά χωρίζεται σε τρία βασικά μέρη:

- Αλγόριθμοι συλλογής δεδομένων και εκπαίδευσης μοντέλων.
- Αλγόριθμος αυτόνομης οδήγησης.
- Γραφική διεπαφή χρήστη.

4.2 Περιβάλλον – Assetto Corsa

Όταν ξεκίνησα την εργασία, αλλά και καθ’ όλη τη διάρκεια εκπόνησης έπρεπε να κάνω κάποιες επιλογές. Στην πραγματικότητα αυτές οι επιλογές ήταν αποτέλεσμα πολλών δοκιμών, στα πλαίσια αυτού του κειμένου όμως θα αναφερθούν κυρίως οι τελικές επιλογές μαζί με την αιτιολόγηση τους.

4.2.1 Επιλογή αυτοκινήτου

Η πρώτη απόφαση λοιπόν ήταν τι αυτοκίνητο να επιλέξω. Το παιχνίδι το ίδιο προσφέρει μια αρκετά μεγάλη ποικιλία και επιπρόσθετα δίνει τη δυνατότητα σε τρίτους να δημιουργήσουν αυτοκίνητα από το 0 και να τα εισάγουν στο παιχνίδι. Μετά από πολλές δοκιμές αποφάσισα ότι το πιο σημαντικό για τις ανάγκες της εργασίας, είναι το αυτοκίνητο να είναι όσο πιο «αργό» γίνεται, να έχει δηλαδή όσο το δυνατόν λιγότερη ιπποδύναμη, έτσι ώστε να είναι πιο εύκολο στον έλεγχο. Ιδίως στις αρχές της ανάπτυξης του αλγορίθμου, όταν δοκίμαζα πράγματα, τα πιο γρήγορα αυτοκίνητα έβγαιναν εύκολα εκτός ελέγχου. Κατέληξα λοιπόν στο Abarth 595 SS της εικόνας 4.1, το οποίο είναι ένα από τα επίσημα αυτοκίνητα που παρέχει το παιχνίδι και είναι αργό, ελαφρύ και ευκίνητο, με μόνο 32 ίππους, 470 κιλά και χαμηλή τελική ταχύτητα 130 km/h. Στην πράξη βέβαια πλέον, στην παρούσα φάση του αλγορίθμου μπορεί να χρησιμοποιηθεί οποιοδήποτε αυτοκίνητο χωρίς κανένα πρόβλημα.

Εικόνα 4.1. Τα στατιστικά του Abarth 595 SS.

4.2.2 Επιλογή «δρόμων»

Η δεύτερη επιλογή ήταν σε ποιους δρόμους, ποιες «πίστες», θέλουμε να οδηγάει το αυτοκίνητο. Και πάλι το παιχνίδι προσφέρει μια αρκετά μεγάλη ποικιλία από δρόμους, οι οποίοι κατά πλειοψηφία είναι πίστες αγώνων όπως για παράδειγμα η πίστα της εικόνας 4.2. Το «πρόβλημα» των πιστών αγώνων, είναι ότι είναι πολύ πλατιοί και χωρίς γραμμές, παρά στα δύο άκρα του δρόμου. Όπως θα δούμε αργότερα ο αλγόριθμος βασίζεται πολύ στης γραμμές, γι' αυτό και εντέλει η οδήγηση σε μία πίστα σαν αυτήν είναι δυνατή αλλά δύσκολη λόγω της υπερβολικής ελευθερίας κίνησης εντός των γραμμών.

Η επόμενη κατηγορία δρόμων είναι οι «κανονικοί» δρόμοι του οδικού δικτύου. Υπάρχουν πάρα πολύ δρόμοι δημιουργημένοι από τρίτους που προσομοιώνουν πραγματικά κομμάτια δρόμων, όπως το παράδειγμα της εικόνας 4.3. Στο παράδειγμα αυτό βρισκόμαστε σε έναν αυτοκινητόδρομο της Ιαπωνίας, όπου συναντάμε μια πολύ μεγάλη ποικιλία δρόμων, είτε μίας λωρίδας, είτε πολλαπλών λωρίδων. Ο λόγος που δεν χρησιμοποιήθηκαν τέτοιοι δρόμοι είναι λόγω της πολυπλοκότητας που εισάγουν. Ο αλγόριθμος στην παρούσα φάση του αν και είναι σε θέση να οδηγήσει σε τέτοιους δρόμους, μιας και βασίζεται κατά βάση στις λωρίδες, οι οποίες υπάρχουν όπως φαίνεται στη φωτογραφία, δεν θα ήταν σε θέση να πάρει αποφάσεις όπως σε ποια λωρίδα να παραμένει. Η μετάβαση σε τέτοιους δρόμους και η δυνατότητα τέτοιον αποφάσεων είναι μία από τις προτάσεις για βελτίωση, οι οποίες θα αναλυθούν λεπτομερώς στο τέλος του κειμένου.

Εικόνα 4.2. Παράδειγμα πίστας αγώνων «Silverstone Circuit, Αγγλία».

Εικόνα 4.3. Παράδειγμα αυτοκινητοδρόμου, «Shutoko Expressway, Ιαπωνία»

Φτάνουμε λοιπόν στην κατηγορία δρόμων που χρησιμοποιήθηκαν τελικά για την εκπαίδευση και τη δοκιμή των μοντέλων. Όπως βλέπουμε στην εικόνα 4.4, πρόκειται για δρόμους που σε όλη τους την έκταση είναι μίας λωρίδας και οι δύο αντίθετες λωρίδες χωρίζονται είτε με συνεχόμενες, είτε με διακεκομμένες γραμμές. Πρόκειται λοιπόν για ένα πολύ συγκεκριμένο περιβάλλον στο οποίο ο αλγόριθμος χρειάζεται να πάρει συγκεκριμένες ξεκάθαρες αποφάσεις. Ο σκοπός είναι το αυτοκίνητο να ξεκινήσει να κινείται παραμένοντας εντός της λωρίδας του από την αρχή της διαδρομής μέχρι το τέλος. Για να το καταφέρει αυτό, πρέπει να παίρνει διαρκώς, σωστές αποφάσεις για το πόσο να στρίψει το τιμόνι, πόσο γκάζι πρέπει να πατήσει και αν πρέπει να φρενάρει και πόσο.

Εικόνα 4.4. Coste Odolo προς Caino, Ιταλία.

4.2.3 Επιλογή γωνίας κάμερας και ένδειξη ταχύτητας

Η τελευταία και μάλλον καθοριστικότερη επιλογή είναι η γωνία της κάμερας. Όπως αναφέρθηκε και παραπάνω, το Asseto Corsa μας δίνει τη δυνατότητα να το τοποθετήσουμε την κάμερα χειροκίνητα πρακτικά όπου θέλουμε. Στις προηγούμενες φωτογραφίες η κάμερα βρίσκεται πίσω από το αυτοκίνητο, αυτή όμως δεν είναι η θέση της κάμερας που χρησιμοποιούμε, είναι απλά για να φανεί καλύτερα στις φωτογραφίες το μέγεθος του δρόμου και του περιβάλλοντος γενικά σε σχέση με το αυτοκίνητο. Από εδώ και πέρα για οτιδήποτε ακολουθήσει, θεωρούμε ότι η θέση της κάμερας είναι αυτή της εικόνας 4.5. Η κάμερα τοποθετείται σαν να είναι στο μπροστά μέρος της οροφής του αυτοκινήτου με μια κλίση προς τα κάτω, με λίγη μεγέθυνση και ευρύτερο φακό. Με αυτόν τον τρόπο έχουμε μια καλή εικόνα του δρόμου και τον γραμμών, για αρκετή απόσταση.

Εικόνα 4.5. Η εικόνα που βλέπει ο αλγόριθμος, Mount Akina, Ιαπωνία.

Η καθοριστικότητα της σωστής θέσης της κάμερα φαίνεται συγκρίνοντας τις εικόνες 4.6 και 4.7, όπου η πρώτη δείχνει το αυτοκίνητο πάνω σε μία πολύ απότομη στροφή 180 μοιρών με την τυπική κάμερα του παιχνιδιού χωρίς τις ρυθμίσεις, ενώ η δεύτερη δείχνει το αυτοκίνητο ακριβώς στην ίδια θέση, αλλά με την κάμερα μετά τη ρύθμιση. Προφανώς στην πρώτη περίπτωση είναι δύσκολο να καταλάβει ακόμα και άνθρωπος την κατεύθυνση του δρόμου και πράγματι οι πρώτες προσπάθειες που έγιναν με αυτήν την κάμερα, είχαν προβλήματα σε απότομες στροφές.

Στις Εικόνες 4.5, 4.6 και 4.7, στο αριστερό μέρος της οθόνης βλέπουμε και μια ένδειξη της ταχύτητας με την οποία κινείται το αυτοκίνητο και της ταχύτητας στην οποία βρίσκεται το κιβώτιο των ταχυτήτων. Η ένδειξη της ταχύτητας, η οποία είναι η μόνη που μας ενδιαφέρει, όπως θα δούμε στη συνέχεια θα χρησιμοποιηθεί ως πληροφορία για τη λειτουργία του αλγορίθμου.

Εικόνα 4.6. Τυπική κάμερα, Monte Erice, Ιταλία.

Εικόνα 4.7. Κάμερα μετά τη ρύθμιση, Monte Erice, Ιταλία.

4.3 Επικοινωνία του προγράμματος με το παιχνίδι, Είσοδοι – Έξοδοι

Ο τρόπος με τον οποίο επικοινωνεί ένα παίχτης με το παιχνίδι και ο τρόπος που επικοινωνεί το πρόγραμμα αυτόνομης οδήγησης είναι πρακτικά ακριβώς ο ίδιος. Όπως ο άνθρωπος βλέπει την οθόνη (πληροφορίες εισόδου) και επεμβαίνει είτε με το πληκτρολόγιο, είτε με ειδικό χειριστήριο για παιχνίδια (πληροφορίες εξόδου – ενέργειες παίχτη), έτσι ακριβώς και το πρόγραμμα «βλέπει» την οθόνη τραβώντας επαναλαμβανόμενα screenshots (στιγμιότυπα) και επεμβαίνει στέλνοντας τις εντολές του, αυτή τη φορά μέσω ενός προσομοιωτή χειριστηρίου παιχνιδιών.

Για τη λήψη των στιγμιότυπων της οθόνης χρησιμοποιούνται μερικές από τις βιβλιοθήκες της python, οι οποίες παρουσιάστηκαν νωρίτερα, παρακάτω θα αναλυθεί ο κώδικας λεπτομερώς. Για τον προσομοιωτή χειριστηρίου παιχνιδιών χρησιμοποιείται ειδικό third-party πρόγραμμα, ενώ για την επικοινωνία με το παιχνίδι χρησιμοποιούνται πάλι κάποιες από τις βιβλιοθήκες που αναφέρθηκαν, λεπτομέρειες παρακάτω.

4.3.1 Στιγμιότυπα οθόνης (Είσοδοι)

Ο κώδικας για τη λήψη στιγμιότυπων βρίσκεται στο αρχείο grapscreen.py, Παράρτημα Α. Για τη λήψη των πληροφοριών της οθόνης από τα windows χρησιμοποιείται η βιβλιοθήκη pywin32, η οποία όπως αναφέρθηκε και παραπάνω, επικοινωνεί μέσω του API των windows και παίρνει τις πληροφορίες της οθόνης που χρειαζόμαστε. Όλη η δουλειά γίνεται από τη συνάρτηση **grab_screen**() η οποία δέχεται από εμάς τις συντεταγμένες του μέρους της οθόνης που θέλουμε να πάρουμε screenshot και επιστρέφει ένα numpy array που περιέχει την RGB (red-green-blue, κόκκινο-πράσινο-μπλέ) εικόνα. Σε περίπτωση που αφήσουμε κενή την παράμετρο των συντεταγμένων, επιστρέφει ολόκληρη την οθόνη.

Για την προβολή των screenshot που παίρνει η grab_screen χρησιμοποιούμε τη συνάρτηση imshow() της βιβλιοθήκης cv2. Εκτός από τη συνάρτηση grab_screen(), υπάρχει ένα κομμάτι κώδικα σε σχόλιο, το οποίο χρησιμοποιήθηκε για λόγους δοκιμών. Ο κώδικας αυτός κάνει επαναλαμβανόμενα screenshots, τα προβάλει συνεχόμενα και μετράει το χρόνο μεταξύ κάθε screenshot. Αυτό γίνεται με τη χρήση της βιβλιοθήκης time, μέσω της διαφοράς της ώρα μεταξύ επαναλήψεων του while loop. Αυτό ήταν χρήσιμο για να αποκτήσουμε μια ιδέα για το πόσο συνεισφέρει η διαδικασία της λήψης του screenshot στη συνολική καθυστέρηση του αλγορίθμου. Οι υπόλοιπο παράγοντες καθυστέρησης θα αναλυθούν αργότερα.

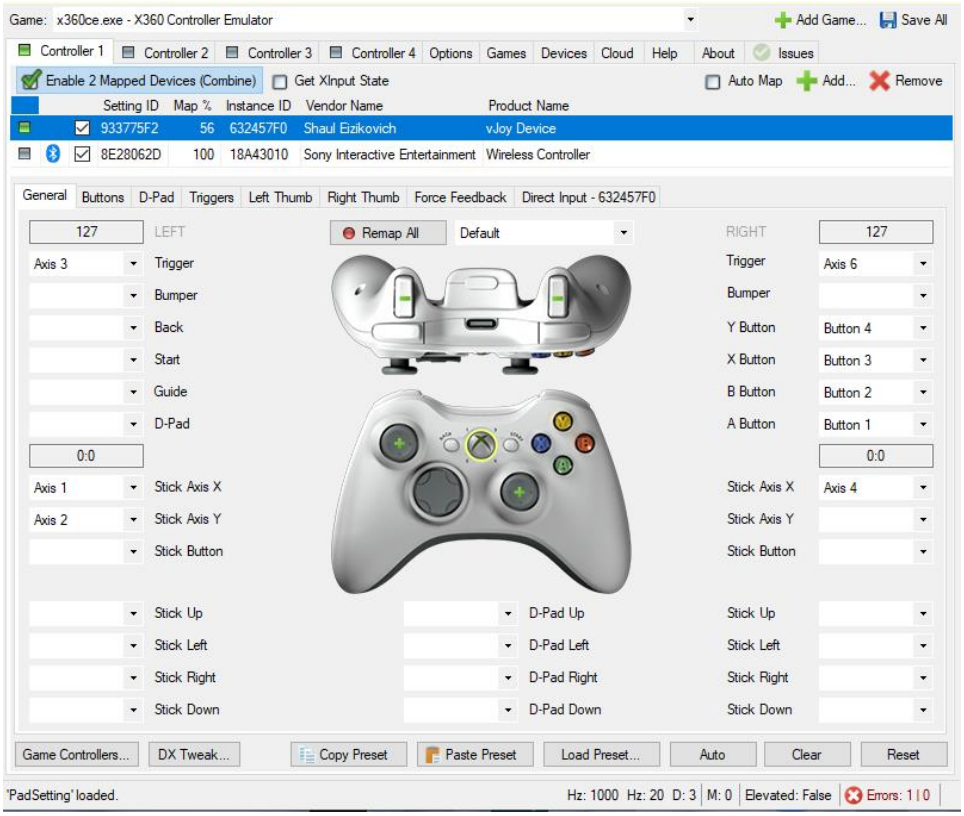
Ο κώδικας του grabscreen.py προέρχεται από το πρότζεκτ του Sentdex “pygta5” και είναι γραμμένος από κάποιον με το όνομα Franneckl p. [37]

4.3.2 Χειριστήριο (Έξοδοι)

Για να επιτευχθεί η αναπαραγωγή των εντολών από το πρόγραμμα χρειάστηκαν τα εξής 2 third-party προγράμματα:

- X360ce (Xbox 360 Controller Emulator) [38]: Το πρόγραμμα αυτό είναι ένας προσομοιωτής Xbox controller (χειριστηρίου), (το Xbox είναι κονσόλα παιχνιδιών της Microsoft, τα Windows όντας επίσης της Microsoft, αντιμετωπίζουν διαφορετικά τα Xbox controllers από οποιαδήποτε άλλα controllers της αγοράς). Πρακτικά αυτό που κάνει είναι να δέχεται εντολές από ένα οποιοδήποτε controller, το οποίο δεν είναι φτιαγμένο για το Xbox και να τις

μετατρέπει στη μορφή που αναγνωρίζουν τα Windows. Ο λόγος που χρειάζεται αυτή η μετατροπή είναι γιατί ορισμένα παιχνίδια δεν υποστηρίζουν χειριστήρια που δεν είναι της Microsoft. Στην εικόνα 4.8, βλέπουμε τη διεπαφή χρήστη του x360ce, η σειρά που είναι επισημασμένη με μπλε χρώμα δείχνει ότι η εφαρμογή επικοινωνεί με τη vJoy εικονική συσκευή.



Εικόνα 4.8. Γραφική διεπαφή χρήστη του x360ce.

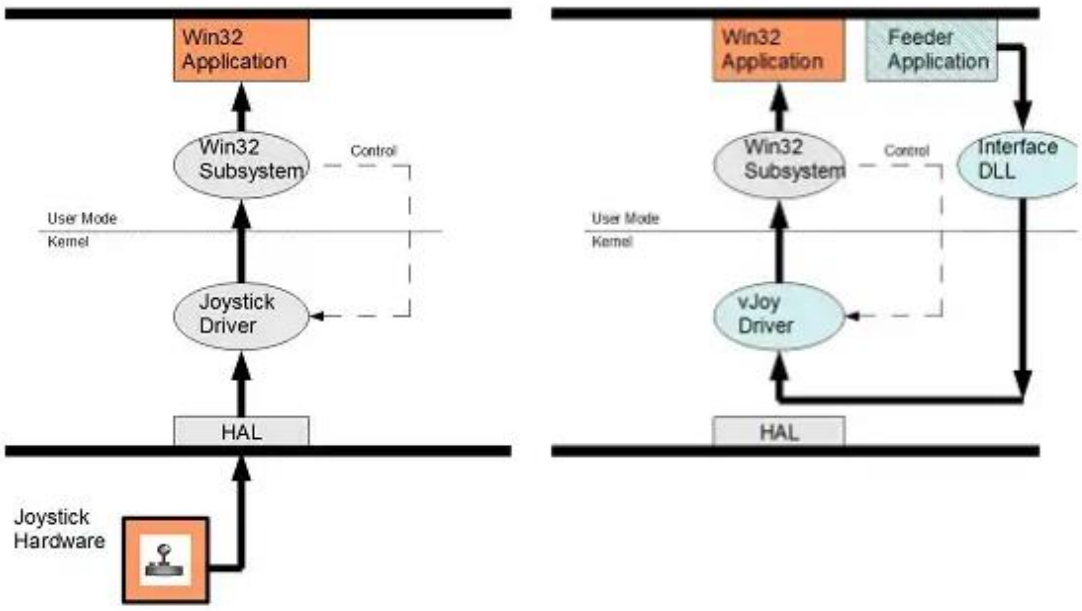
- VJoy (Virtual Joystick): Το vJoy είναι ένα πρόγραμμα που προσομοιώνει ένα τυπικό, μη Xbox, χειριστήριο. Επιπλέον μας παρέχει ένα API (vjoyInterface.dll) μέσω του οποίο μπορούμε να επικοινωνήσουμε με το vJoy από την Python. Με αυτόν τον τρόπο πρακτικά χειριζόμαστε αυτό το εικονικό χειριστήριο μέσα από τον κώδικα της Python. Το vJoy είναι γραμμένο από τον Shaul Eizikovich [39].

Χάρη στο συνδυασμό των δύο προγραμμάτων, μπορούμε να στείλουμε εντολές Xbox χειριστηρίου από την Python. Η διαδικασία συνοψισμένη έχει ως εξής:

Ο κώδικας της Python «λέει», ποια «κουμπιά» να πατηθούν, καλώντας συναρτήσεις από το API του vJoy που παρέχεται μέσω του αρχείου «vJoyInterface.dll», στο εικονικό χειριστήριο vJoy, το οποίο με τη σειρά του στέλνει την εντολή στο x360ce, το οποίο μετατρέπει την εντολή στη μορφή που το λειτουργικό σύστημα των Windows αναγνωρίζει ως Xbox εντολές.

Στο σχήμα 4.1 φαίνεται η διαφορά της επικοινωνίας ενός πραγματικού χειριστηρίου και του εικονικού χειριστηρίου vJoy. Στην πρώτη περίπτωση, το πραγματικό χειριστήριο στέλνει κάποια εντολή, μέσω του Driver που είναι υπεύθυνο για την επικοινωνία με τα Windows. Από εκεί η πληροφορία καταλήγει στην εφαρμογή μας. Στην περίπτωση του vJoy, έχουμε το Feeder Application, το οποίο στην περίπτωση μας είναι το Python πρόγραμμα, το οποίο χρησιμοποιεί ένα API, μέσω του αρχείου vjoyInterface.dll, για να στείλει τις εντολές στο vJoy Driver, το οποίο με τη σειρά του θα περάσει την

πληροφορία στα Windows και τελικά στην εφαρμογή. Το μόνο που λείπει από το σχήμα 4.8, είναι η παρεμβολή του x360ce, μεταξύ του vJoy Driver και των Windows, μεταφράζοντας τις εντολές του vJoy στη «γλώσσα» των Xbox χειριστηρίων.



Σχήμα 4.1. Στα αριστερά βλέπουμε το ταξίδι της πληροφορίας από το πραγματικό χειριστήριο στην εφαρμογή, ενώ στα δεξιά βλέπουμε την ίδια διαδικασία με το εικονικό vjoy χειριστήριο. Το Feeder Application για εμάς είναι ο κώδικας της Python. [40]

4.3.2.1 VJoy.py

Ο κώδικας του αρχείου VJoy.py είναι υπεύθυνος για την επικοινωνία του Python προγράμματος μας και του VJoy Driver. Χρησιμοποιώντας συναρτήσεις του vjoyInterface.dll στέλνει της εντολές που θέλουμε από το εικονικό χειριστήριο.

Οι σημαντικότερες συναρτήσεις στο vJoy.py είναι οι εξής:

- **open**(): «Ανοίγει», μια επικοινωνία με τον vJoy Driver, πριν ανανεώσουμε την κατάσταση του χειριστηρίου.
- **close**(): «Κλείνει», την επικοινωνία με τον vJoy Driver, αφού ανανεώσουμε την κατάσταση του vJoy χειριστηρίου.
- **generateJoystickPostition**(key_values): Δημιουργεί μία λίστα, με την επιθυμητή κατάσταση όλων των μεταβλητών του χειριστηρίου, δηλαδή πρακτικά όλων των «κουμπιών», και την μετατρέπει σε ένα C struct, τα οποίο και επιστρέφει. Αυτό το βήμα είναι απαραίτητο ώστε να σταλθεί η κατάσταση του controller στη σωστή μορφή.
- **update**(generateJoystickPostition()): Δέχεται ως παράμετρο την έξοδο της *generateJoystickPostition* μέσω του vJoyInterface.dll, ανανεώνει την κατάσταση του χειριστηρίου, ανάλογα με της τιμές που δώσαμε στην *generateJoystickPostition.*
- Εντέλει χρησιμοποιώντας τις παραπάνω συναρτήσεις, δημιουργείται η ***mytest_phase4****(throttle_perc, brake_perc, steering_perc)*, εικόνα 4.9, η οποία κάνει όλες τις παραπάνω ενέργειες, ανανεώνοντας την κατάσταση του χειριστηρίου με βάση της 3 παραμέτρους που της δίνουμε:
- throttle_perc (ποσοστό γκαζιού), το οποίο είναι μία ακέραια τιμή με εύρος 0 – 255.

- brake_perc (ποστοστό φρένου), το οποίο είναι μια ακέραια τιμή με εύρος 0 – 255.
- Steering_perc (θέση του τιμονιού), το οποίο είναι μια ακέραια τιμή με εύρος 0 – 32786.*

*Σημείωση: Και οι 3 τιμές «στέλνονται» με εύρος 0 – 32786, ο λόγος της ανομοιομορφίας είναι ότι η μετατροπή από 0 – 255 σε 0 – 32786, γίνεται σε διαφορετικά σημεία του κώδικα.

```
def mytest_phase4(throttle_perc, brake_perc, steering_perc):
    vj.open()
    throttle_percentage = int(throttle_perc*(32786/100))
    brake_percentage = int(brake_perc*(32786/100))
    steering_percentage = round(steering_perc)
    joystickPosition = vj.generateJoystickPosition(wAxisX = steering_percentage,wAxisZRot = throttle_percentage,wAxisZ = brake_percentage)
    vj.update(joystickPosition)
    vj.close()
```

Εικόνα 4.9. Συνάρτηση mytest_phase4.

4.3.2.2 Read_xbox_controller_state.py

Άλλο ένα αρχείο κώδικα που όπως θα δούμε στη συνέχεια είναι απαραίτητο για τη δημιουργία των Dataset, είναι το read_xbox_controller_state.py. Η λειτουργία του είναι να καταγράφει την κατάσταση του πραγματικού χειριστηρίου οποιαδήποτε στιγμή, όσο ο χρήστης παίζει το παιχνίδι (οδηγάει). Ο κώδικας αυτός, χρησιμοποιώντας τη βιβλιοθήκη ctypes, επικοινωνεί με το λειτουργικό σύστημα των Windows και αντλεί τις πληροφορίες που χρειαζόμαστε για την κατάσταση του χειριστηρίου. Ο συγγραφές αυτού του κώδικα είναι ο Ryan Barnes. [41]

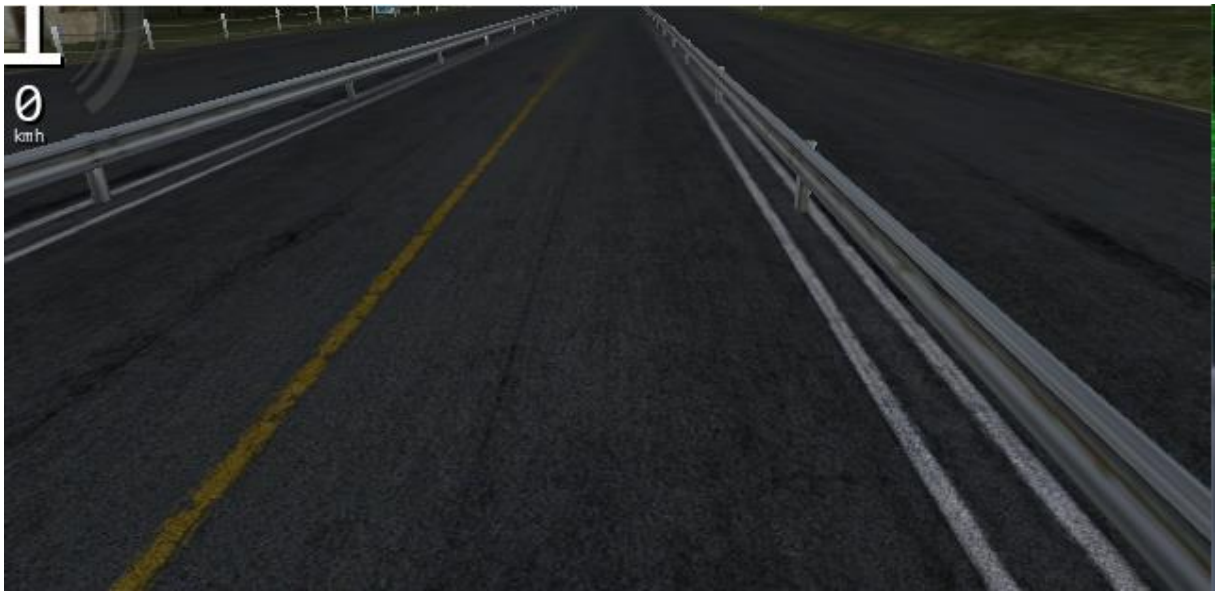
4.4 Συλλογή Δεδομένων

Για να εκπαιδεύσουμε τα CNNs, χρειαζόμαστε datasets, τα οποία να αποτελούνται από δεδομένα με ετικέτες. Το ερώτημα λοιπόν είναι, τι δεδομένα θέλουμε να συλλέξουμε και πως θα τα συλλέξουμε. Ο τελικός αλγόριθμος αυτόνομης οδήγησης χρησιμοποιεί τρία είδη CNNs, επομένως πρέπει να δημιουργηθούν τριών τύπων datasets, το καθένα με τα δεδομένα, με τα οποία θέλουμε να εκπαιδεύσουμε το κάθε CNN. Τα τρία είδη CNN είναι υπεύθυνα για τις τρεις εξής λειτουργίες:

- CNN 1 (Ταξινόμηση ψηφίων): Το CNN 1 λαμβάνει screenshots 22x17 pixels από το κομμάτι της οθόνης που επισημάνεται με το κόκκινο πλαίσιο και στο οποίο εμφανίζεται η ένδειξη της ταχύτητας, εικόνα 4.10. Το CNN 1 χρησιμοποιείται δύο φορές, μία για κάθε ψηφίο.

Εικόνα 4.10. Περιοχή screenshot ψηφίου ένδειξης ταχύτητας.

- CNN 2 (Πρόβλεψη τιμής steering_input): Το CNN 2, λαμβάνει ως είσοδο μία εικόνα, όπως η εικόνα 4.11 και δίνει ως έξοδο μια πρόβλεψη για το ποια πρέπει να είναι η τιμή του steering_input, πόσο δηλαδή πρέπει να είναι στριμμένο το τιμόνι με βάση την εικόνα εισόδου. Όπως βλέπουμε από την εικόνα, δεν καταλαμβάνει όλο το οπτικό πεδίο που βλέπει ο πραγματικός χρήστης στην οθόνη, όπως π.χ. είδαμε στην εικόνα 4.5 νωρίτερα. Ο λόγος που κόβουμε αυτό που βλέπει το πρόγραμμα, είναι για να γλυτώσουμε μερικά «άχρηστα» pixels, τα οποία το μόνο που θα έκαναν είναι να μας επιβαρύνουν με παραπάνω μνήμη στο δίσκο, παραπάνω χρόνο εκπαίδευσης και ανάκλησης και μεγαλύτερη πολυπλοκότητα εισόδου, περισσότερες ευκαιρίες δηλαδή να κάνει λάθος το μοντέλο μας.
- CNN 3 (Πρόβλεψη «απόλυτης τιμής» steering input): Το CNN 3, λαμβάνει την ίδια ακριβώς είσοδο με το CNN 1, αλλά δίνει έξοδο την «απόλυτη τιμή» της θέσης του τιμονιού. Στην πράξη αυτό σημαίνει ότι μας δίνει την πληροφορία του πόσο πρέπει να στρίψει το αυτοκίνητο, ανεξάρτητα από την κατεύθυνση.



Εικόνα 4.11. Τα screenshot που απαρτίζουν τα dataset για το CNN 2 και η εικόνα που πρακτικά βλέπει το πρόγραμμα όταν «οδηγάει».

Τα 4 αρχεία με κώδικα υπεύθυνο για τη δημιουργία όλων των Datasets της εργασίας είναι τα εξής:

- Digit_capturing.py
- Data_collecting.py
- Data_collecting_turn_anticipation.py
- Data_collecting_complete.py

4.4.1 Digit_capturing.py

Ο κώδικας του αρχείου digit_capturing.py μας βοηθάει να δημιουργήσουμε Datasets, αποτελούμενα από εικόνες ψηφίων με ετικέτες. Ο τρόπος που δουλεύει είναι ο εξής, όταν τρέχουμε το αρχείο, ο κώδικας μπαίνει σε ένα while loop, περιμένοντας να τον διακόψουμε. Με το πάτημα ενός κουμπιού από το χειριστήριο, συγκεκριμένα το κουμπί «B», καλείται η συνάρτηση *grabscreen()*, η οποία παίρνει ένα screenshot από ένα εκ των δύο ψηφίων, ανάλογα με το όρισμα που της έχουμε δώσει. Στη συνέχεια μετατρέπει την εικόνα σε ασπρόμαυρη μορφή, μιας και η πληροφορία του χρώματος δεν μας

χρειάζεται. Τέλος μας προβάλει αυτό το screenshot, με τη συνάρτησης *imshow()* της βιβλιοθήκης cv2 και μας ζητάει να εισάγουμε από το πληκτρολόγιο το ψηφίο του screenshot, μας ζητάει δηλαδή την ετικέτα. Όταν δώσουμε τιμή μέσω της κονσόλας, δημιουργείται μία λίστα με δύο στοιχεία, ένα numpy array με τις τιμές των pixel της εικόνας και ένας ακέραιος αριθμός για την ετικέτα. Η λίστα αυτή γίνεται append, προστίθεται δηλαδή σε μία λίστα «*training_data*» μαζί με όλες τις προηγούμενες καταγραφές. Αυτή η διαδικασία επαναλαμβάνεται διαρκώς, μέχρι να πατήσουμε το κουμπί «Y», όπου τότε η λίστα «training_data» σώζεται στη μνήμη του υπολογιστή, σε μορφή «.npy file», ως ένα numpy array δηλαδή.

Εδώ υπάρχει μια σημαντική ιδιαιτερότητα. Οι περιοχές που πάντα καταγράφουμε είναι τα pixels στα οποία εμφανίζονται τα δύο ψηφία ενός διψήφιου αριθμού. Υπάρχει όμως μια «ειδική» περίπτωση, η περίπτωση του μονοψήφιου αριθμού, όταν δηλαδή το αυτοκίνητο κινείται με ταχύτητες μικρότερες των 10 km/h όπου εμφανίζεται ένα μόνο ψηφίο ακριβώς ανάμεσα στις δύο περιοχές, εικόνα 4.12. Αυτό έχει ως αποτέλεσμα τα δύο screenshot να πιάνουν μισό αριθμό αριστερά και μισό δεξιά.

Η λύση που δίνουμε σε αυτό το πρόβλημα, είναι να αντιμετωπίζουμε οποιονδήποτε «κομμένο» αριθμό ως ένα 11ο ψηφίο, το οποίο εκφράζει την ύπαρξη μονοψήφιου αριθμού. Επειδή στην πράξη δεν υπάρχει στιγμή κατά την οδήγηση που θέλουμε το αυτοκίνητο να κινείται με λιγότερα από 10 χιλιόμετρα, όπως θα δούμε αργότερα, το πρόγραμμα αντιμετωπίζει όλες τις ταχύτητες κάτω από 10 km/h με τον ίδιο τρόπο.

Εικόνα 4.12. Αριστερά βλέπουμε πως οι μονοψήφιοι αριθμοί εμφανίζονται ανάμεσα στις δύο περιοχές που καταγράφομε. Δεξιά βλέπουμε την περίπτωση που το ψηφίο εμφανίζεται σωστά, ολόκληρο εντός της περιοχής που καταγράφουμε.

Είναι λογικό να σκεφτεί κανείς ότι ίσως να είναι υπερβολή η χρήση CNN για την αναγνώριση 10 (11) σταθερά ίδιων χαρακτήρων. Υπάρχει όμως μια λεπτομέρεια που κάνει το πρόβλημα αυτό λίγο πιο περίπλοκο. Η λεπτομέρεια είναι ότι το φόντο του ψηφίου αλλάζει συνεχώς όσο το αυτοκίνητο κινείται, και μάλιστα υπάρχουν περιπτώσεις στις οποίες το φόνο γίνεται σχεδόν λευκό, όταν το φόντο είναι ο ουρανός, κάνοντας την αναγνώριση του ψηφίου οριακά δύσκολη και για έναν άνθρωπο.

4.4.2 Data_collecting.py

Στο data_collecting.py γίνεται πρακτικά κάτι αντίστοιχο με το προηγούμενο, digit_capturing.py. Μόνο που αυτή τη φορά η περιοχή που «στοχεύουμε» με τη συνάρτηση grapscreen(), είναι αυτή που βλέπουμε στην εικόνα 4.11, ενώ οι ετικέτα των εικόνων, είναι η κατάσταση του μοχλού του χειριστηρίου μας, ο μοχλός, εικόνα 4.14, δίνει τη δυνατότητα στο χρήστη να επικοινωνεί με το παιχνίδι και να στρίβει το τιμόνι του αυτοκινήτου με έναν φαινομενικά συνεχή τρόπο. Στην πραγματικότητα ο μοχλός μπορεί να πάρει 256 διακριτές θέσεις στον X άξωνα και 256 στον Y, εμάς μας ενδιαφέρει μόνο ο άξωνας X. Οι τιμές του μοχλού κυμαίνονται από -128 έως +127, εμείς όμως για να αποφύγουμε τους αρνητικούς αριθμούς, τις μετατρέπουμε σε uint8 μορφή, δηλαδή σε θετικούς 8-bit ακέραιους αριθμούς (0 - 255).

Όσο λοιπόν ο κώδικας «τρέχει», κάνει διαδοχικά screenshots της περιοχής της οθόνης που μας ενδιαφέρει, η οποία είναι (640x310 pixel) και ταυτόχρονα (στην πραγματικότητα υπάρχει αναγκαστικά μια μικρή καθυστέρηση της τάξης των ms που δεν παίζει ρόλο), καταγράφει την τρέχουσα κατάσταση του χειριστηρίου, χρησιμοποιώντας τη συνάρτηση gamepad της κλάσης rController η οποία έχει γίνει import από το αρχείο read_xbox_controller_state.py. Κάθε 0,05 δευτερόλεπτα, δηλαδή 20 φορές το λεπτό, ένα ζευγάρι του numpy array (screenshot) και της uint8 μεταβλητής (μοχλός) αποθηκεύονται στη λίστα training_data.

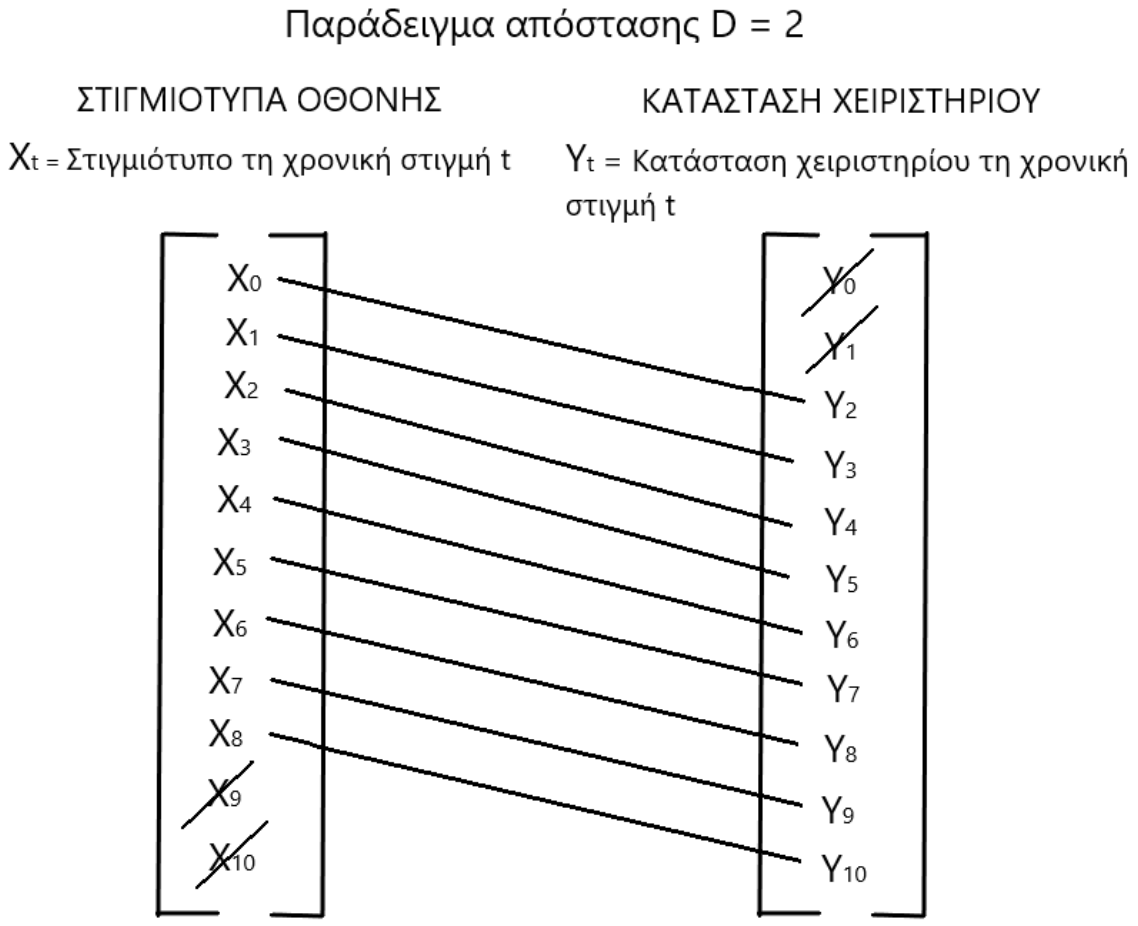
Κάθε 500 καταγραφές ζευγαριού, εικόνας και τιμής μοχλού, αποθηκεύεται η λίστα training_data στο δίσκο σαν numpy αρχείο. Ακόμα πατώντας το κουμπί «B» από το χειριστήριο, η το «P» από το πληκτρολόγιο, μπορούμε να διακόψουμε τη διαδικασία καταγραφής και πατώντας το ξανά να τη συνεχίσουμε. Έτσι με εύκολο και ελεγχόμενο τρόπο μπορούμε να δημιουργήσουμε Dataset, τα οποία στη συνέχεια όπως θα δούμε θα χρησιμοποιήσουμε για να εκπαιδεύσουμε τα CNNs.

Εικόνα 4.13. Μοχλός χειριστηρίου, χειρισμός τιμονιού.

4.4.3 Data_collecting_turn_anticipation.py

Το *data_clollecting_turn_anticipation.py* κάνει ότι κάνει και το *data_collecting.py*, καταγράφει το ίδιο ακριβώς κομμάτι της οθόνης και το ίδιο ακριβώς input, την κατάσταση του μοχλού του χειριστηρίου. Οι μοναδικές τέσσερεις διαφορές είναι οι εξής:

- Οι τιμές του χειριστηρίου (-128 έως 127), αντί να μεταφέρονται στο εύρος (0 – 255), μετατρέπονται σε απόλυτες τιμές (0 - 128), αφαιρώντας έτσι πρακτικά την πληροφορία της κατεύθυνσης που στρίβει το τιμόνι.
- Τα στιγμιότυπα της οθόνης και οι καταστάσεις του μοχλού του χειριστηρίου αποθηκεύονται σε ξεχωριστούς πίνακες.
- Τη στιγμή που είτε πατηθεί το κουμπί τερματισμού, είτε μαζευτούν 500 καταγραφές, αντί να αποθηκευτεί ένας πίνακας που συνδυάζει - αντιστοιχίζει τα στοιχεία των δύο πινάκων, γίνεται μια παραλλαγή αυτού. Ο πίνακας των καταστάσεων του χειριστηρίου, ολισθαίνει πάνω στον πίνακα των στιγμιότυπων κατά d στοιχεία. Το d (distance - απόσταση) είναι μια παράμετρος που ορίζεται από τον χρήστη. Στη συνέχεια αντιστοιχίζονται τα στοιχεία των δύο πινάκων, αλλά λόγω της ολίσθησης που προηγήθηκε, κάθε στιγμιότυπο μια χρονικής στιγμής t, αντιστοιχίζεται με μία κατάσταση μίας χρονικής στιγμής t + d. Στο σχήμα 4.2, βλέπουμε ένα παράδειγμα για 11 χρονικές στιγμές, t = (0 - 10) και d = 2. Τα δύο ακραία στοιχεία που περισσεύουν αφαιρούνται.
- Δεν υπάρχει πλέον κουμπί παύσης με την έννοια που υπήρχε στο *data_collecting*, με κάθε παύση, τα δεδομένα που έχουνε συλλεχθεί ως τη στιγμή που πατιέται το κουμπί, αποθηκεύονται κατευθείαν. Αυτό συμβαίνει γιατί αν υπάρχουν παύσεις μεταξύ της καταγραφής των δεδομένων, τότε τα δεδομένα αυτά δεν θα έχουν νόημα. Όλη η λογική αυτού του Dataset είναι η αντιστοίχιση παροντικών ενεργειών με παρελθοντικές εικόνες, αυτή η λογική όμως βασίζεται στο ότι δεν υπάρχουν κενά μεταξύ των καταγεγραμμένων καταστάσεων.



Σχήμα 4.2. Αναπαράσταση ολίσθησης πινάκων για ένα παράδειγμα 11 χρονικών στιγμών και απόσταση μεταξύ στιγμιότυπου και ενέργειας d = 2.

Ο λόγος που γίνεται αυτό όπως θα εξηγήσω βαθύτερα και στη συνέχεια, είναι ώστε με βάση τη θέση που βρίσκεται το αυτοκίνητο μια χρονική στιγμή t, να μπορεί να προβλέψει την ενέργεια που θα πρέπει να κάνει τη χρονική στιγμή t + d. Με αυτόν τον τρόπο μπορεί να οδηγάει «κοιτώντας» πιο μακριά και να προετοιμάζεται για στροφές που μπορεί να ακολουθήσουν.

4.4.4 Data_collecting_complete.py

Το αρχείο αυτό δεν είναι τίποτα παραπάνω από το συνδυασμό των δύο προηγούμενων. Αυτό που κάνει είναι να μας επιτρέπει, την ίδια στιγμή, να δημιουργούμε και τα δύο Datasets. Αυτό που αντιστοιχεί εικόνα με ενέργεια την ίδια χρονική στιγμή και αυτό που αντιστοιχεί εικόνα με ενέργεια μερικών χρονικών στιγμών αργότερα. Ο λόγος ύπαρξης των δύο ξεχωριστών αρχείων και του συνδυασμού τους, είναι γιατί η κάθε προσέγγιση έχει θετικά και αρνητικά.

Το πλεονέκτημα της ταυτόχρονης καταγραφής των δύο dataset είναι ο χρόνος και η μνήμη που γλυτώνουμε, αλλά αυτό μας περιορίζει στο ότι τα δύο dataset είναι πρακτικά όμοια. Επίσης αναγκαστικά το data_collecting_complete.py κληρονομεί την απαίτηση να μην υπάρχουν κενά μεταξύ των καταγραφών των καταστάσεων, πράγμα που κάνει λιγότερο ομαλή την όλη διαδικασία.

Από την άλλη διαχωρίζοντας τις δύο διαδικασίες, σπαταλάμε περισσότερο χρόνο και μνήμη, γιατί αποθηκεύουμε το διπλάσιο όγκο εικόνας, αλλά μπορούμε να φτιάξουμε το κάθε dataset όπως ακριβώς θέλουμε, ανάλογα με τις ανάγκες του κάθε μοντέλου και του τρόπου που σκοπεύουμε να το χρησιμοποιήσουμε.

4.5 Εκπαίδευση Νευρωνικών Δικτύων

Το επόμενο βήμα ύστερα από τη συλλογή δεδομένων, είναι η εκπαίδευση των CNN χρησιμοποιώντας αυτά τα δεδομένα. Για την επίτευξη του τελικού σκοπού τα αυτόνομης οδήγησης, χρησιμοποιούμε ταυτόχρονα τρία διαφορετικά CNN. Κάθε ένα από τα CNN χρησιμοποιεί ένα από τα τρία είδη dataset που έχουμε δημιουργήσει. Τα τρία αρχεία που περιέχουν τον κώδικα για την εκπαίδευση των ΝΔ είναι τα εξής:

- **train_speed_model.py** – CNN 1: Πολύ απλό CNN, με πολύ λίγες παραμέτρους. Σκοπός του είναι να αναγνωρίζει τα ψηφία της ένδειξης της ταχύτητας στην οθόνη.

- **train_model_steering.py** – CNN 2: Τελευταίας τεχνολογίας CNN (EfficientB0), πολύπλοκο με πολλές μεν παραμέτρους (5.3M), αλλά ταυτόχρονα λίγες σχετικά με την απόδοση του, συγκριτικά με άλλα αντίστοιχα ανταγωνιστικά CNN.

- **train_model_turn_anticipation.py** – CNN 3: Λίγο παλαιότερο, αλλά και πάλι τελευταίας τεχνολογίας CNN (MobileNetV2), αντίστοιχου σκελους με το CNN 2, αλλά με λιγότερες παραμέτρους (3.5), αλλά και κατώτερη απόδοση.

Και στα 3 αρχεία ορίζονται οι συναρτήσεις, create*_model(), concat_files() load_dataset, callback()* και *main()* (τα ονόματα των αρχείων μπορεί να διαφέρουν λίγο μεταξύ των αρχείων, αλλά η ουσία είναι η ίδια).

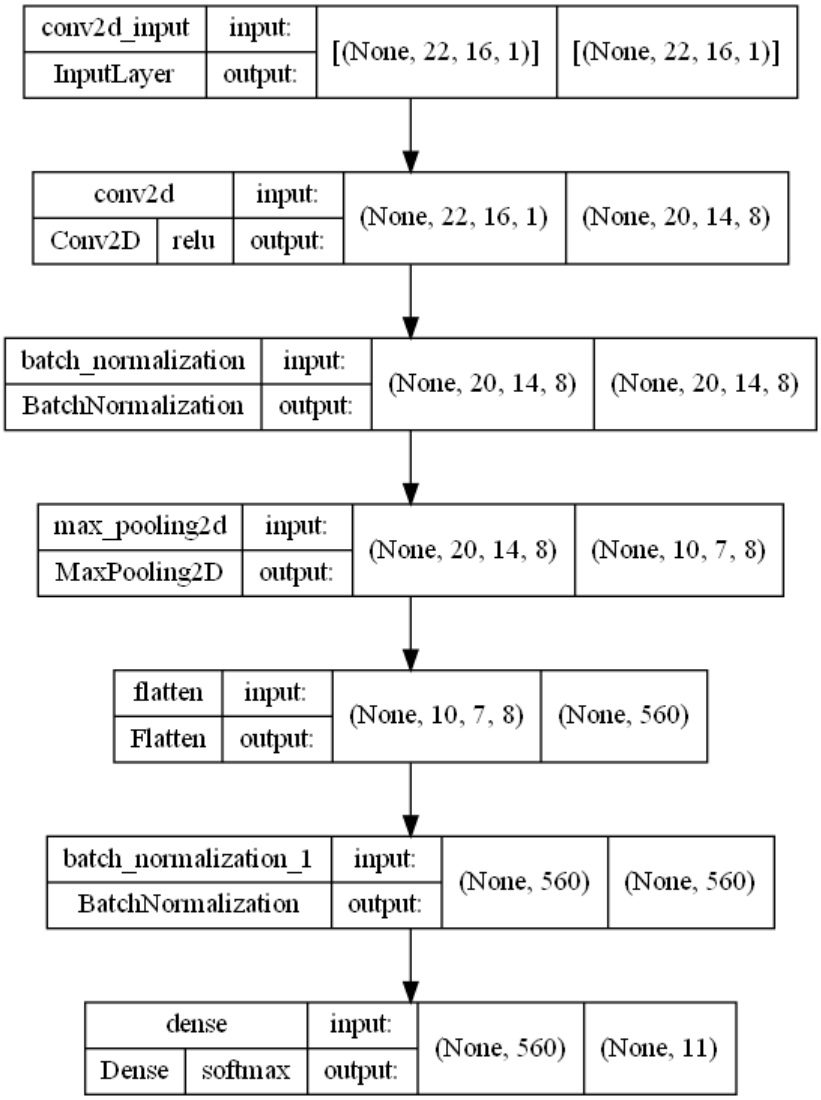
- Η συνάρτηση **create_model()** δημιουργεί το μοντέλο, ορίζοντας την αρχιτεκτονική του, από το σχήμα (shape) εισόδου και τα ενδιάμεσα στρώματα, μέχρι την έξοδο, τη συνάρτηση βελτιστοποίησης και τις διάφορες υπερπαραμέτρους.
- Η συνάρτηση **concat_files()** διαβάζει όλα τα αρχεία (numpy πίνακες) τα οποία δημιουργήθηκαν κατά τη διαδικασία δημιουργίας του dataset, όπως περιγράφηκε παραπάνω και τα συνδυάζει για να δημιουργήσει ένα numpy array που περιέχει όλα τα δεδομένα.
- Η συνάρτηση **load_dataset()** προετοιμάζει τα δεδομένα για να τα τροφοδοτήσει ύστερα στο μοντέλο. Συγκεκριμένα, αφού τα ανακατέψει, τα χωρίζει σε δεδομένα εκπαίδευσης (train data) και δεδομένα ανάκλησης (validation data), με αναλογία 80 – 20. Τα validation data χρησιμοποιούνται για να παρακολουθείται η απόδοση του μοντέλου ανά εποχή. Στη συνέχεια μέσω της συνάρτησης *ImageDataGenerator()* του tensorflow, τα δεδομένα οργανώνονται σε ομάδες και εφαρμόζεται σε αυτά data augmentation.
- Η συνάρτηση **callback()** ουσιαστικά ορίζει συναρτήσεις ή ειδικές λειτουργίες, οι οποίες εκτελούνται κατά τη διάρκεια της εκπαίδευσης του μοντέλου. Συγκεκριμένα εμείς χρησιμοποιούμε τη συνάρτηση *ModelCheckpoint()*, η οποία είναι υπεύθυνη για την αποθήκευση του εκπαιδευμένου μοντέλου στον υπολογιστή μας. Αυτό το βήμα είναι απαραίτητο ώστε να μπορούμε στη συνέχεια να χρησιμοποιήσουμε τα μοντέλα που εκπαιδεύουμε, ώστε να κάνουμε προβλέψεις.
- Η συνάρτηση **main()** αφού χρησιμοποιήσει τις παραπάνω συναρτήσεις για να ορίσει το φορτώσει τα δεδομένα, να δημιουργήσει το μοντέλο και να ορίσει τα callbacks, καλεί τη συνάρτηση *fit()*, η οποία είναι υπεύθυνη για την εκπαίδευση του μοντέλου, για τον αριθμό των εποχών που επιλέγουμε.

4.5.1 train_speed_model.py – CNN 1

Το πρόβλημα της αναγνώρισης των 10+1 ψηφίων, είναι ένα πολύ εύκολο πρόβλημα, εφόσον τα ψηφία εμφανίζονται πάντα στην ίδια θέση και η μορφή τους δεν αλλάζει ποτέ. Η μόνη «δυσκολία» είναι ότι αλλάζει το φόντο της εικόνας, όσο το αυτοκίνητο κινείται. Όπως και να έχει, υπάρχουν πάρα πολλά νευρωνικά δίκτυα που θα μπορούσαμε να χρησιμοποιήσουμε, CNN ή μη. Εμείς χρησιμοποιούμε το CNN του σχήματος 4.3, το οποίο αποτελείται από ένα συνελικτικό στρώμα με 8 φίλτρα και, πυρήνα (3, 3), με συνάρτηση ενεργοποίησης relu, ένα batch normalisation στρώμα, ένα max pooling στρώμα (2x2), ένα flatten στρώμα, ένα δεύτερο batch normalization και τέλος ένα softmax στην έξοδο με 11 νευρώνες, έναν για κάθε ψηφίο. Σαν μέθοδος βελτιστοποίησης χρησιμοποιείται η SGD με ρυθμό εκπαίδευσης 0.01 και ορμή 0.9, ενώ ορίζεται η συνάρτηση σφάλματος Categorical Crossentropy. Τέλος σαν μέτρο αξιολόγησης του μοντέλου χρησιμοποιείται η ακρίβεια του, το ποσοστό δηλαδή των επιτυχημένων προβλέψεων σε κάθε batch. Συνολικά το μοντέλο έχει 8523 παραμέτρους, πολύ λιγότερους σε σχέση με τα επόμενα μοντέλα που θα δούμε.

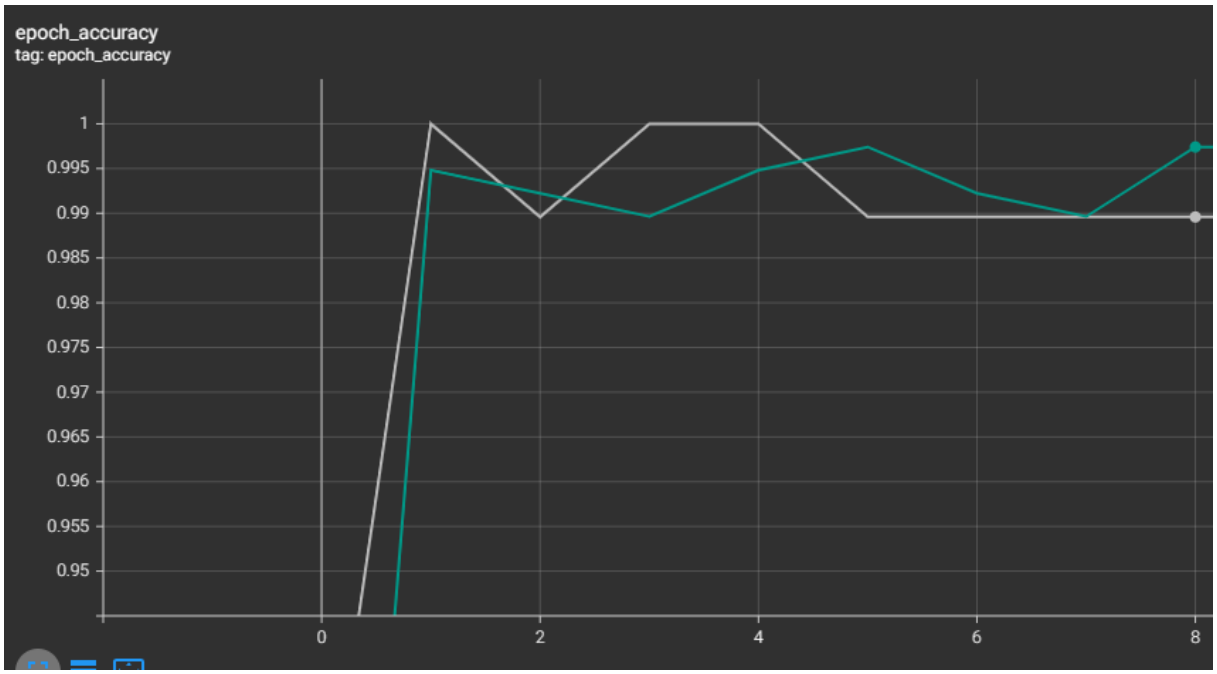
Σε αυτήν την περίπτωση το *ImageDataGenerator()*, δημιουργεί ομάδες δεδομένων ανά 8 ζευγάρια εικόνας και ετικέτας, ενώ εφαρμόζει και ένα πολύ ελαφρύ data augmentation, μετακίνησης της εικόνας κατά μήκος, σε πιθανό εύρος -0.05 έως 0.05. Επίσης από τη στιγμή που το μοντέλο χρησιμοποιεί softmax στην έξοδο, οφείλουμε να τροποποιήσουμε τις ετικέτες των δεδομένων σε one hot encoded μορφή.

Τέλος το *ModelCheckpoint* που ορίζουμε θα αποθηκεύσει στη μνήμη το μοντέλο με τα εκπαιδευμένα βάρη, της εποχής που πέτυχε το μέγιστο validation accuracy.



Σχήμα 4.3. Αναπαράσταση στρωμάτων CNN 1 σε γράφημα.

Με τη βοήθεια του TensorBoard, βλέπουμε τη γραφική απεικόνιση του train accuracy και του validation accuracy. Παρατηρούμε ότι από την πρώτη εποχή ακόμα το μοντέλο πιάνει πρακτικά 100 % accuracy και στο train και στο test. Ιδανικά βλέποντας το αυτό θα έπρεπε να αντικαταστήσουμε το μοντέλο με κάποιο μικρότερο, ώστε να γλυτώσουμε χρόνο επεξεργασίας, αλλά στην πράξη η διαφορά που θα πετυχαίναμε θα ήταν απειροελάχιστη συγκριτικά με τους υπόλοιπους παράγοντες καθυστέρησης.



Σχήμα 4.4. Γραφική αναπαράσταση των train (πράσινο) και validation(γκρί) accuracy.

4.5.2 train_model_steering.py – CNN 2

Το CNN 2 είναι το βασικότερο νευρωνικό δίκτυο που χρησιμοποιούμε στην εργασία και γενικά το σημαντικότερο κομμάτι του συνολικού αλγορίθμου. Η απόδοση του είναι καθοριστική για το τελικό αποτέλεσμα. Η λειτουργία του είναι να δέχεται εικόνα από το δρόμο, αυτό που θα έβλεπε ουσιαστικά και ο παίχτης του παιχνιδιού, με εξαίρεση μια περιοχή στο πάνω μέρος της εικόνας, όπως εξηγήθηκε παραπάνω και όπως είδαμε στην εικόνα 4.11. Λαμβάνοντας λοιπόν σαν είσοδο εικόνα, μας δίνει μια πρόβλεψη, για την τιμή της μεταβλητής που ελέγχει το τιμόνι, πρακτικά πρόκειται για μια τιμή στο εύρος 0 – 255 όπου:

- 0 σημαίνει μέγιστη στροφή αριστερά
- 255 σημαίνει μέγιστη στροφή προς τα δεξιά
- 127 ή 128 σημαίνει ότι προχωράμε ευθεία, στην πράξη δεν υπάρχει ακριβές κέντρο εφόσον έχουμε 256 τιμές, και το 256 είναι ζυγός αριθμός.
- και φυσικά αντίστοιχα όλες οι ενδιάμεσες τιμές.

Ακριβώς λόγω της καθοριστικότητας του για την τελική ποιότητα του αποτελέσματος, δοκιμάστηκαν αρκετά νευρωνικά δίκτυα. Όλες οι δοκιμές όμως έγιναν με CNN, μιας και είναι με διαφορά η πιο αποτελεσματική λύση όταν έχουμε να κάνουμε με εικόνες και επίσης πάντα με προ-εκπαιδευμένα μοντέλα, μιας και είναι πρακτικά αδύνατο ένα άτομο να δημιουργήσει κάτι καλύτερο από το μηδέν. Στην πραγματικότητα ακόμα και περιορίζοντας τις επιλογές μας σε προ-εκπαιδευμένα συνελικτικά νευρωνικά δίκτυα, πάλι έπρεπε να περιοριστούμε σε πολύ λίγες και συγκεκριμένες επιλογές.

Ο μεγαλύτερος περιορισμός, που όπως θα εξηγήσω και αργότερα δυσκολεύει και την επέκταση και βελτίωση του αλγορίθμου, είναι η επεξεργαστική ισχύς. Η εκπαίδευση αλλά και η ανάκληση βαθιών νευρωνικών δικτύων, είναι μια τρομερά απαιτητική υπολογιστική εργασία. Όταν έχουμε να κάνουμε με νευρωνικά δίκτυα του σκέλους των πιο εξελιγμένων CNN που χρησιμοποιούμε, είναι απαραίτητη η

χρήση κάρτας γραφικών (GPU). Μάλιστα η κάρτα πρέπει υποχρεωτικά να είναι της Nvidia και επιπλέον το πόσο πρόσφατη και γενικά το πόσο ικανή είναι, επηρεάζει καθοριστικά το αποτέλεσμα. Η εργασία αυτή δεν θα ήταν δυνατή χωρίς τη χρήση μιας καλής Nvidia κάρτας γραφικών, συγκεκριμένα χρησιμοποιήθηκε το μοντέλο Nvidia RTX 3060 TI, μια πρόσφατη, αλλά μέσης δυναμικότητας κάρτας, συγκριτικά με τα υπόλοιπα εμπορικά μοντέλα της Nvidia. Οι επιλογές των νευρωνικών δικτύων λοιπόν γι' αυτόν τον λόγο έπρεπε να γίνουν με μεγάλη προσοχή.

Τα κριτήρια, από άποψη επεξεργαστικής ανάγκης, με βάσει τα οποία επιλέχθηκαν τελικά τα CNN, είναι ο αριθμός των παραμέτρων, το βάθος και η απόδοση, που θεωρητικά μπορεί να προσφέρει το μοντέλο. Ο αριθμός των παραμέτρων και το βάθος, γιατί όσο μικρότερα, τόσο λιγότερο χρόνο παίρνει η εκπαίδευση και κυρίως τόσο λιγότερο χρόνο (της τάξης των ms), παίρνει η ανάκληση. Στην πράξη η ταχύτητα της ανάκλησης καθορίζει το πόσο «συχνά» μπορεί να κάνει προβλέψεις ο αλγόριθμος για την επόμενη ενέργεια του αυτοκινήτου, καθορίζει δηλαδή τις εντολές ανά δευτερόλεπτο, αλλά και την ορθότητα της εντολής, μικραίνοντας τη χρονική απόσταση μεταξύ της παρατήρησης και της ενέργειας. Η απόδοση του μοντέλου, προφανώς γιατί θέλουμε όσο γίνεται ακριβείς προβλέψεις, δεν έχει αξία ένα μοντέλο που είναι γρήγορο μεν, αλλά μία στις δέκα φορές στρίβει στη λάθος κατεύθυνση.

Στον πίνακα 4.1, βλέπουμε ένα μέρος από τα προ-εκπαιδευμένα μοντέλα που προσφέρει το keras. Τα μοντέλα με τα οποία έγιναν οι περισσότερες δοκιμές ήταν τα εξής, με κατάταξη ανάλογα με την επιτυχία των αποτελεσμάτων:

- EfficientNetB0
- MobileNetV2
- DenseNet121
- ResNet50

Η αξιολόγηση της αποτελεσματικότητας των μοντέλων στη συγκεκριμένη εφαρμογή δεν είναι εύκολο να μετρηθεί, με διαγράμματα και στατιστικά. Ο λόγος είναι ότι το σημαντικότερο χαρακτηριστικό ενός εκπαιδευμένου μοντέλου για εμάς, είναι να μην κάνει «μεγάλα» σφάλματα, τα οποία να οδηγούν το αυτοκίνητο εκτός πορείας με άτακτο τρόπο. Το πόσο ακριβή θα είναι τα αποτελέσματα, αν δηλαδή προβλέψει τιμή για το τιμόνι 50, αντί για 40 που θα ήταν το ιδανικό, δεν επηρεάζει τόσο πολύ το αποτέλεσμα, μιας και αυτά τα σφάλματα αυτό-εξουδετερώνονται, όταν «αθροίζουμε» μια σειρά από προβλέψεις. Αν δηλαδή σε ένα καρέ (frame), γίνει λάθος πρόβλεψη κατά 10 μονάδες προς τα αριστερά, θα διορθωθεί λογικά στο επόμενο και στα επόμενα καρέ. Άλλωστε ακόμα και όταν οδηγάμε στην πραγματική ζωή, δεν τοποθετούμε πάντα το τιμόνι με 100% ακρίβεια στη σωστή θέση ανάλογα με την καμπύλη του δρόμου, αλλά κάνουμε διαρκώς μικροδιορθώσεις οι οποίες στην πράξη δεν φαίνονται.

Το πιο σημαντικό είναι το μοντέλο να πετύχει όσο καλύτερη γενίκευση γίνεται με βάση τα δεδομένα που το τροφοδοτούμε και να αποφεύγει τις ακραίες λανθασμένες προβλέψεις που θα το στείλουν εκτός δρόμου. Επομένως η ποιότητα του μοντέλου τελικά φαίνεται στην πράξη, βλέποντας πως συμπεριφέρεται στο δρόμο και πόσο συχνά κάνει τέτοιου είδους λάθη.

Μετά από πολλά πειράματα λοιπόν, εκτιμήθηκε ότι το καταλληλότερο μοντέλο γι΄αυτόν τον σκοπό, είναι το EfficientNetB0, μιας όπως και στα «χαρτιά», έδειξε να προσφέρει την καλύτερη σχέση ακρίβειας, γενίκευσης και ταχύτητας. Στη συνέχεια θα αναλυθεί η αρχιτεκτονική του, καθώς και οι όποιες τροποποιήσεις έγιναν στα πλαίσια της εφαρμογής μας.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8
EfficientNetB4	75	82.9%	96.4%	19.5M	258	308.3	15.1
EfficientNetB5	118	83.6%	96.7%	30.6M	312	579.2	25.3
EfficientNetB6	166	84.0%	96.8%	43.3M	360	958.1	40.4
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9	61.6

Πίνακας 4.1. Προ-εκπαιδευμένα CNN του Keras, με πληροφορίες σχετικά ακρίβεια, αριθμό παραμέτρων, βάθος και ταχύτητα.

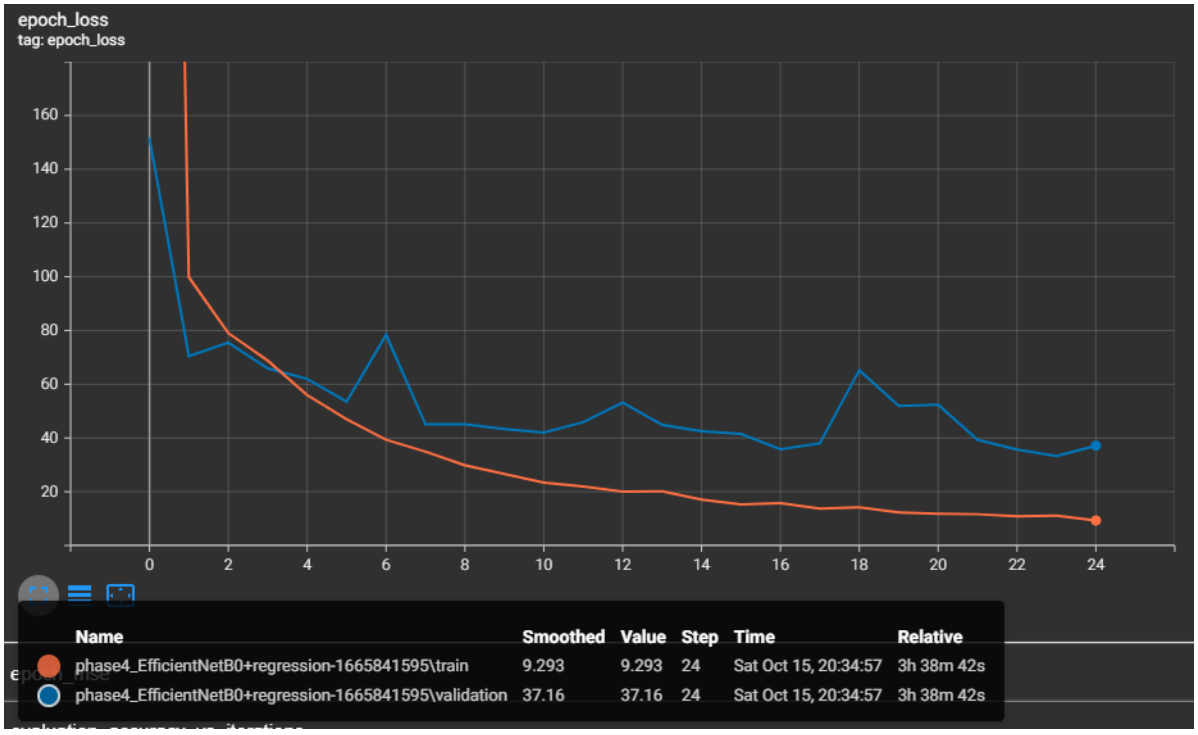
Το ταξίδι για την επίτευξη των καλύτερων προβλέψεων για την εφαρμογή μας δεν σταματάει στην επιλογή του προ-εκπαιδευμένου μοντέλου. Στη συνέχεια πρέπει προσαρμόσουμε αυτό το πολύ δυνατό εργαλείο που έχουμε στη διάθεση μας, στο συγκεκριμένο πρόβλημα που προσπαθούμε να λύσουμε. Πρέπει να αποφασίσουμε τι μορφή θα έχει η έξοδος του δικτύου, τι optimizer θα χρησιμοποιήσουμε, τι συνάρτηση σφάλματος και άλλες τέτοιου τύπου υπερ-παραμέτρους. Για άλλη μια φορά οι επιλογές έγιναν με βάση τη λογική και την εμπειρία.

Παρακάτω θα δούμε μερικά παραδείγματα, του πως η επιλογή των υπερπαραμέτρων που αναφέρθηκαν επιρεάζουν την απόδοση του μοντέλου, μεταξύ των υπερ-παραμέτρων θεωρούμε και το Data Augmentation, με όλες τις παραλλαγές του. Πριν από αυτό όμως, θα αναφέρω μερικά πράγματα που είναι κοινά για όλα τα παραδείγματα που θα ακολουθήσουν.

- Σε όλα τα πειράματα τα βάρη αρχικοποιήθηκαν στα ImageNet βάρη. Αυτό πρακτικά σημαίνει ότι ξεκινάμε να εκπαιδεύουμε το μοντέλο από εκεί που σταμάτησε στην εκπαίδευση του με το ImageNet dataset.
- Το Input shape προφανώς είναι πάντα 310x640, εφόσον τα δεδομένα μας δεν αλλάζουν.
- Στην έξοδο του EfficientNetB0 τοποθετούμε ένα Global Average Pooling 2D layer.
- Η έξοδος του συνολικού μοντέλου είναι πάντα ένας νευρώνας με γραμμική ενεργοποίηση, είναι δηλαδή ένας αριθμός.
- Σαν Optimizer χρησιμοποιούμε πάντα τον Adam, εφόσον αυτή τη στιγμή είναι η πιο δημοφιλής επιλογή.
- Συνάρτηση σφάλματος είναι η Mean Squared Error (MSE). Ο λόγος αυτής της επιλογής είναι ότι η MSE έχει το χαρακτηριστικό να τιμωρεί τα μεγάλα σφάλματα, επειδή η διαφορά μεταξύ της πρόβλεψης και της πραγματικής τιμής για κάθε είσοδο, υψώνεται στο τετράγωνο. Έτσι το σφάλμα αυξάνεται εκθετικά. Όπως εξηγήθηκε παραπάνω, η αποφυγή τον μεγάλων σφαλμάτων είναι πολύ σημαντική για τη συγκεκριμένη εφαρμογή.
- Ως μέθοδος μέτρησης (metric) της απόδοσης του μοντέλου χρησιμοποιείται το σφάλμα, δηλαδή το MSE πάλι.

Έχοντας ξεκαθαρίσει τα κοινά σημεία μεταξύ των διαφορετικών εκδοχών των μοντέλων που θα δούμε, μπορούμε να προχωρήσουμε στην πρώτη εκδοχή, η οποία βέβαια είναι το EfficientNetB0, χωρίς καμία προσθήκη και το σημαντικότερο ίσως, χωρίς την χρήση Data Augmentation. Η αξιολόγηση των μοντέλων σε αυτή τη φάση θα γίνει με βάση τα training και validation loss.

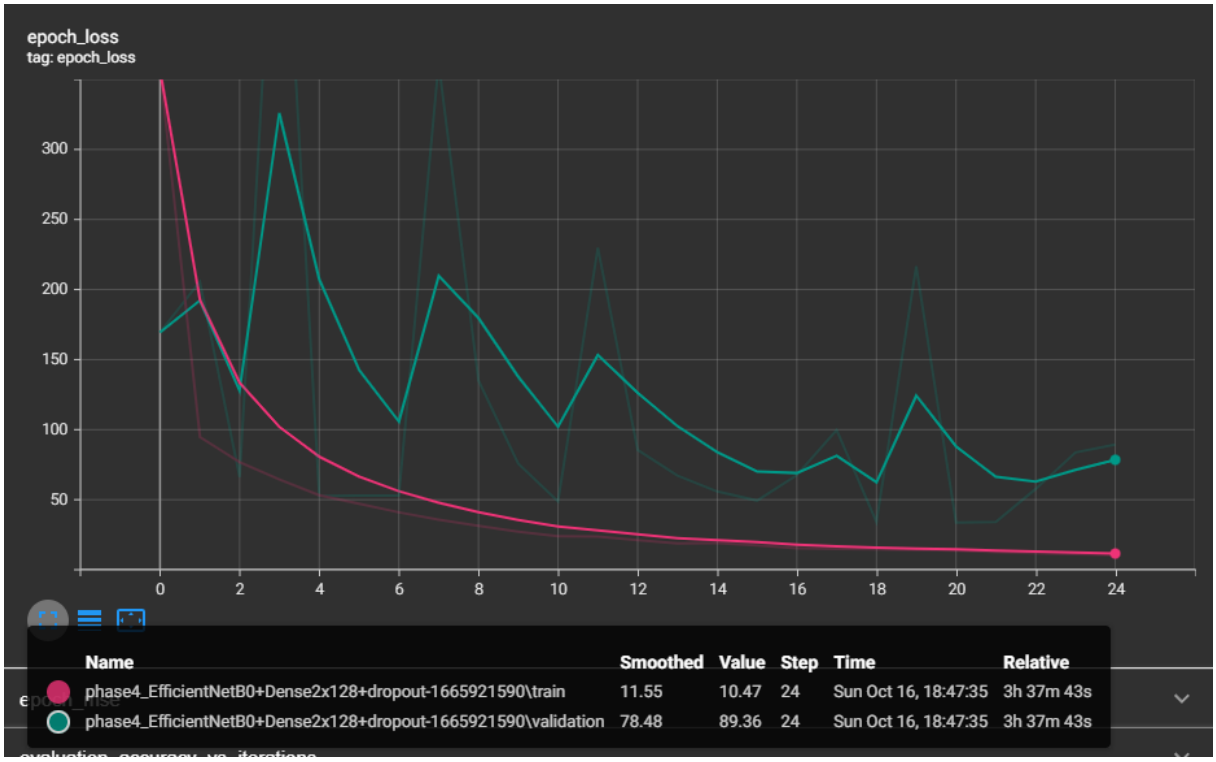
Στο σχήμα 4.5 βλέπουμε την πορεία των training και validation loss για 24 εποχές. Βλέπουμε πως σταδιακά από την 3η εποχή και έπειτα το μοντέλο κάνει πολύ έντονο overfitting.



Σχήμα 4.5. Training και validation loss, για το EfficientNetB0 στην τυπική του μορφή, χωρίς προσθήκες και χωρίς Data Augmentaion.

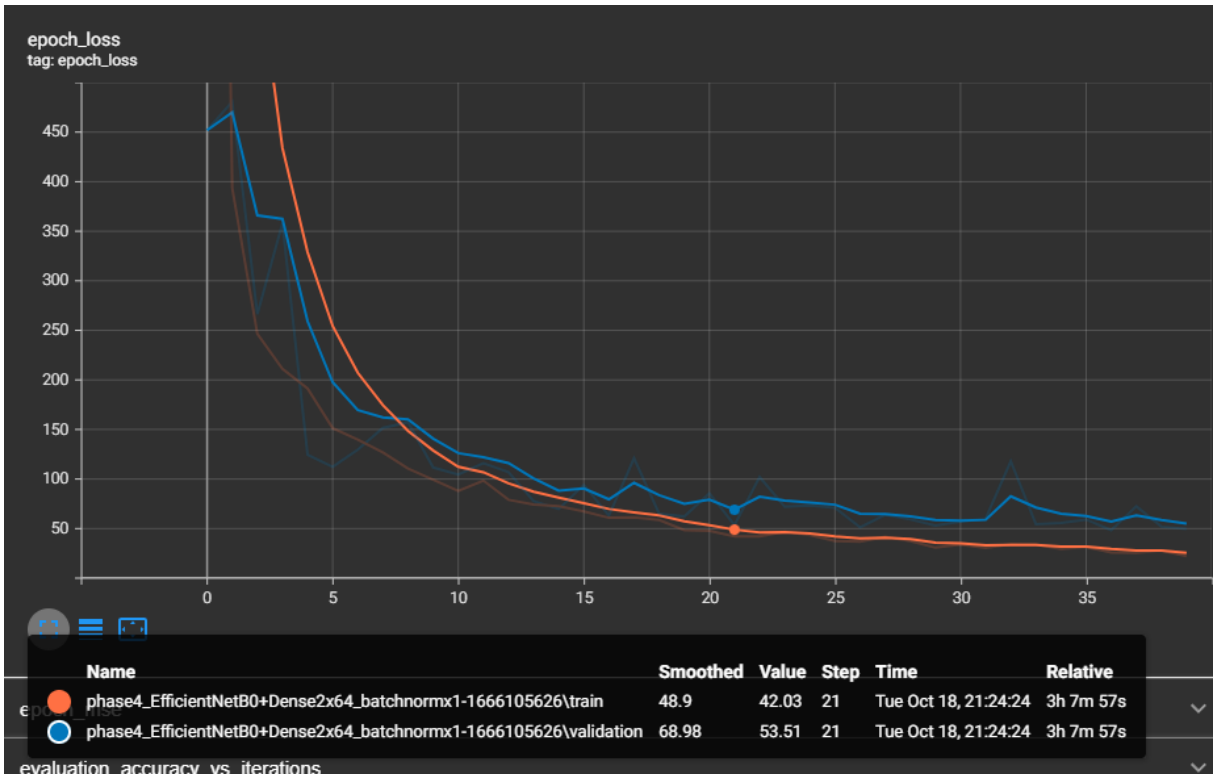
Η επόμενη απόπειρα που αξίζει να αναφερθεί, είναι με την προσθήκη 2 dense στρωμάτων, από 64 ή 128 νευρώνες το καθένα. Και στις δύο περιπτώσεις, 64 και 128, είδαμε μια μικρή βελτίωση στο overfitting. Για παράδειγμα με τους 128 νευρώνες ανά στρώμα, καταφέραμε μια σχέση training – validation loss, 16 – 32, το οποίο αν και ακόμα δείχνει πολύ έντονο overfititting, είναι μια σημαντική βελτίωση από το 9 – 37, χωρίς το dense κομμάτι.

Το επόμενο πείραμα έγινε με την προσθήκη μερικών συνδυασμών dropout layer μεταξύ των δύο dense στρωμάτων της εξόδου. Το αποτέλεσμα όπως φαίνεται και στο παράδειγμα του σχήματος 4.7 σε ένα από τα πειράματα, ήταν καταστροφικό, με 10 – 89 training – validation loss , οπότε και απορρίφτηκε αυτή η μέθοδος.



Σχήμα 4.6. Training και validation loss, για το EfficientNetB0, με την προσθήκη 2x128 dense layer στην έξοδο και 2 dropout layers μεταξύ των dense στρωμάτων.

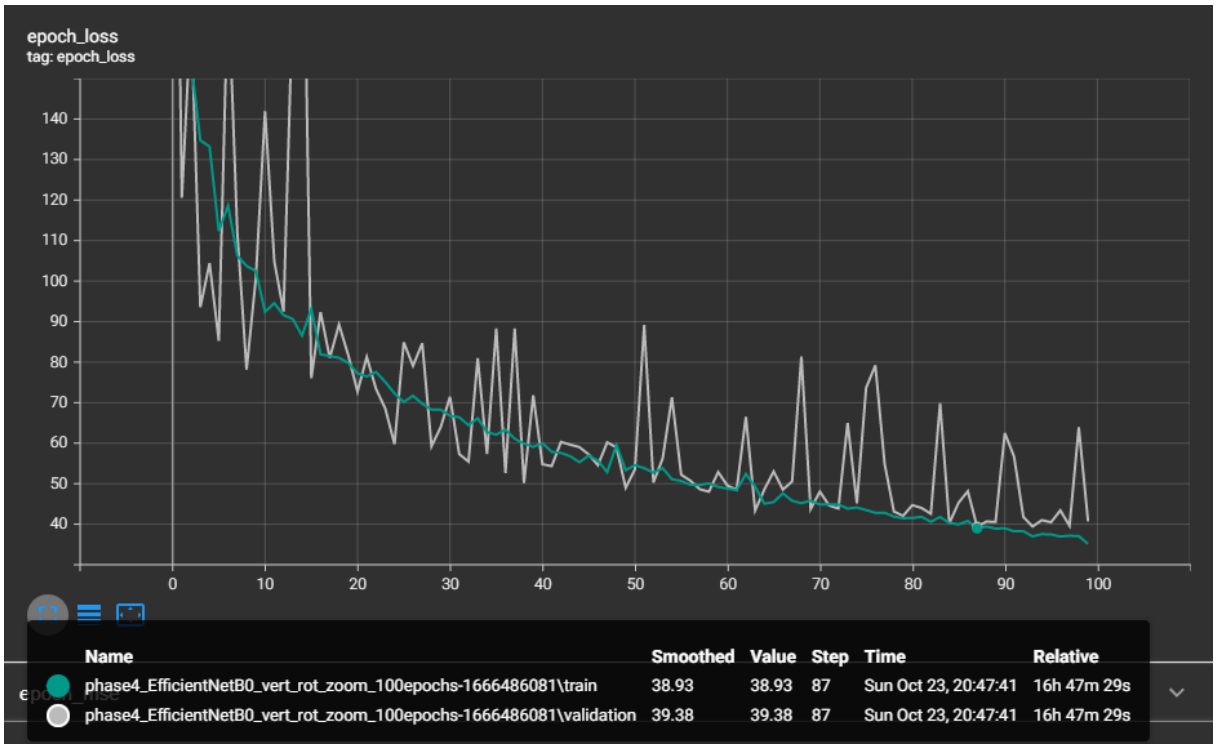
Στη συνέχεια το dropout αντικαταστάθηκε με ένα batch normalization μεταξύ των στρωμάτων. Στο σχήμα 4.8 βλέπουμε ένα παράδειγμα, για 2x64 dense layer και batch normalization μεταξύ τους. Πάλι βλέπουμε πως όσο προχωράμε στις εποχές το overfitting χειροτερεύει, αλλά για παράδειγμα αν παρατηρήσουμε στην εποχή 21, τα train και validation loss, είναι 42 και 53 αντίστοιχα. Αν και συγκρίνοντας με το πρώτο παράδειγμα χωρίς καμία προσθήκη, για τον ίδιο αριθμό εποχών, το σφάλμα είναι μεγαλύτερο, βλέπουμε μια βελτίωση στο overfitting.



Σχήμα 4.7. Training και validation loss, για το EfficientNetB0, με την προσθήκη 2x64 dense layer στην έξοδο και 1 batch normalization layer μεταξύ των dense στρωμάτων.

Και φτάνουμε στο τελευταίο και καλύτερο μοντέλο, τουλάχιστον με βάση τα διαγράμματα, το οποίο πρακτικά είναι το ίδιο με το προηγούμενο, με 2x64 dense layer, batch normalization αλλά πλέον με εφαρμογή Data Augmentation στα δεδομένα. Οι δύο μέθοδοι augmentation που χρησιμοποιούμε είναι, vertical flip (κατακόρυφη περιστροφή) και μεγέθυνση στο εύρος 0 – 20 %. Στο σχήμα 4.9 βλέπουμε τα αποτελέσματα σε βάθος 100 εποχών. Ο αριθμός των εποχών αυξήθηκε τόσο πολύ, διότι με το data augmentation η εκπαίδευση έγινε πιο δύσκολη και χρονοβόρα, πράγμα που φυσικά είναι επιθυμητό για την αντιμετώπιση του overfitting.

Παρατηρούμε ότι στην εποχή 87, πετυχαίνουμε train και validation loss, 39 – 39, πράγμα που σημαίνει ότι το πρόβλημα του overfitting έχει αντιμετωπιστεί επιτυχώς. Αυτό φυσικά φαίνεται και στην πράξη, όπου οι ακραίες, ανεξήγητες ενέργειες έχουν σχεδόν εξαλειφθεί.



Σχήμα 4.8. Training και validation loss για το EfficientNetB0, με την προσθήκη 2x64 dense layer στην έξοδο, batch normalization layer μεταξύ των dense στρωμάτων και data augmentation στα δεδομένα (vertical flip, zoom 0 – 20 %).

4.5.3 train_model_turn_anticipation.py – CNN 3

Η λειτουργία του CNN 3 είναι παρόμοια με του CNN 2. Δέχεται τις ίδιες εικόνες και μας δίνει έξοδο παρόμοιας μορφής (0 - 128), μόνο που επειδή εκπαιδεύεται με διαφορετικό dataset, το οποίο αποτελείται από μετατοπισμένα δεδομένα, όπως είδαμε στο σχήμα 4.2, αυτή η έξοδος μας δίνει μια διαφορετική πληροφορία. Αυτήν την πληροφορία, όπως θα δούμε παρακάτω, την αξιοποιούμε με τελείως διαφορετικό τρόπο και γι’ αυτόν τον λόγο οι απαιτήσεις μας από άποψη απόδοσης και ταχύτητας είναι διαφορετικές. Ο σκοπός του μοντέλου αυτού θα εξηγηθεί αναλυτικά αργότερα, για την ώρα αυτό που χρειάζεται να ξέρουμε είναι ότι σε αυτό το μοντέλο, μας ενδιαφέρει περισσότερο η ταχύτητα παραγωγής προβλέψεων, παρά η ακρίβεια των προβλέψεων. Με βάση αυτό, το προ-εκπαιδευμένο μοντέλο που επιλέξαμε είναι το MobileNetV2, το οποίο όπως βλέπουμε και στον πίνακα 4.1, έχει time (ms) per inference step 3.8, που σημαίνει ότι ο χρόνος που χρειάζεται για να παράγει μια πρόβλεψη είναι 3.8 ms, αισθητά μικρότερος σε σχέση με τα 4.9 ms του EfficientNetB0.

4.6 Αλγόριθμος Αυτόνομης Οδήγησης, test_model.py

Έχοντας αναλύσει πλέον όλα τα εργαλεία, τώρα θα δούμε πως θα τα συνδυάσουμε για να πετύχουμε τον επιθυμητό στόχο, που είναι η αυτόνομη οδήγηση του αυτοκινήτου. Χάρη στα τρία CNN που αναλύσαμε παραπάνω, το πρόγραμμα κατά τη διάρκεια της οδήγησης, έχει στη διάθεση του τρείς διαφορετικές πληροφορίες:

- Με ποια ταχύτητα κινείται το αυτοκίνητο.
- Ποια είναι η σωστή θέση του τιμονιού ώστε το αυτοκίνητο να συνεχίσει να κινείται μεταξύ των γραμμών.
- Ποια θα είναι, μερικά frames αργότερα, η σωστή «απόλυτη τιμή» του τιμονιού ώστε το αυτοκίνητο να συνεχίζει να κινείται μεταξύ των γραμμών. Ουσιαστικά η πληροφορία αυτή λέει στο πρόγραμμα πόσο θα χρειαστεί να στρίψει μελλοντικά, δίχως να περιλαμβάνει πληροφορία για την κατεύθυνση της στροφής.

Με βάση αυτές τις 3 πληροφορίες το πρόγραμμα πρέπει να αποφασίσει σε ποια θέση να «τοποθετήσει» το τιμόνι, πόσο να «πιέσει» το γκάζι και πόσο να «πιέσει» το φρένο.

- Η θέση του τιμονιού ταυτίζεται πάντα με την έξοδο του CNN 2, επομένως δεν χρειάζεται περαιτέρω εξήγηση.
- Η θέση του γκαζιού και του φρένου, καθορίζεται λαμβάνοντας υπόψη την τρέχουσα ταχύτητα του αυτοκινήτου (έξοδος CNN 1) και τη «απόλυτη» θέση του τιμονιού που προβλέπει το CNN 3 για μερικά frames μπροστά.

Αναλυτικά ο αλγόριθμος που καθορίζει τις τιμές του γκαζιού και του φρένου λειτουργεί ως εξής:

- Το CNN 1 ενεργοποιείται δύο φορές διαδοχικά, με είσοδο κάθε φορά την περιοχή της οθόνης στην οποία εμφανίζονται τα ψηφία της ένδειξης της ταχύτητας και μας επιστρέφει δύο ακέραιους μονοψήφιους αριθμούς.
- Οι δύο αριθμοί συνδυάζονται και μας δίνουν την τρέχουσα ταχύτητα (current_speed) του αυτοκινήτου.
- Το CNN 3 ενεργοποιείται, με είσοδο την οθόνη την τρέχουσα στιγμή και μας επιστρέφει μια πρόβλεψη για τη θέση του τιμονιού μερικά frames μπροστά. Ο αριθμός αυτός των frames έχει καθοριστεί προηγουμένως από εμάς.
- Με βάση την πρόβλεψη της μελλοντικής τιμής του τιμονιού, υπολογίζεται η επιθυμητή ταχύτητα του αυτοκινήτου για την τρέχουσα στιγμή (desired_speed), από την εξίσωση:

$$desired\_speed = max\_speed - speed\_factor * future\_steering\_angle$$

όπου το max_speed, είναι επίσης μια παράμετρος που ορίζουμε εμείς και καθορίζει τη μέγιστη ταχύτητα με την οποία μπορεί το αυτοκίνητο να κινείται σε οποιαδήποτε στιγμή και το speed_factor είναι ένας συντελεστής, η τιμή του οποίου βρέθηκε εμπειρικά και καθορίζει το ρυθμό μείωσης της επιθυμητής ταχύτητας με βάση την τιμή του future_steering_angle, με βάση δηλαδή το πόσο θα χρειαστεί στα επόμενα frames να στρίψει.

- Το desired_speed συγκρίνεται με το current_speed. Αν ισχύει current_speed > desired_speed, σημαίνει ότι το αυτοκίνητο κινείται πιο γρήγορα από την επιθυμητή ταχύτητα και θέλουμε να επιβραδύνουμε. Ενώ αν ισχύει current_speed < desired_speed, σημαίνει ότι το αυτοκίνητο κινείται πιο αργά από την επιθυμητή ταχύτητα και θέλουμε να επιταχύνουμε. Και στις δύο περιπτώσεις η αλλαγή της ταχύτητας επιτυγχάνεται μεταβάλλοντας τις τιμές του γκαζιού και του φρένου ως εξής:
- Επιβράδυνση: Στην περίπτωση της επιβράδυνσης, υπολογίζεται η νέα θέση του γκαζιού, σαν ποσοστό επί τοις 100, με την εξίσωση:

$$new\_throttle\_perc = throttle\_perc - 0.6 * (current\ speed - desired\ speed)$$

Αν υπολογιστεί το new_throttle < 0, τότε ορίζεται ίσο με 1, καθώς δεν μπορεί να πάρει αρνητικές τιμές.

Στη συνέχεια ορίζεται ένας μετρητής break_counter ο οποίος θα καθορίσει για πόσα frames θα πατηθεί το φρένο, ως εξής:

*brake_counter = 2 * (current_speed – desired_speed)*

Τέλος υπολογίζεται η θέση του φρένου, σαν ποσοστό επί τοις 100:

*new_brake_percentage = 5 * (current_speed – desired_speed)*

Αν υπολογιστεί το new_brake_percentage > 100, τότε ορίζεται ίσο με 100, καθώς δεν μπορεί να πάρει τιμές μεγαλύτερες από 100.

- Επιτάχυνση: Στην περίπτωση της επιτάχυνσης το φρένο μηδενίζεται και το γκάζι ορίζεται από την εξίσωση:

  *new_throttle_perc = throttle_perc + 0.3 * (desired_speed – final_number)*

  Αν υπολογιστεί το new_throttle_perc > 100, τότε ορίζεται ίσο με 100.

- Κατ' εξαίρεση αν η ένδειξη της ταχύτητας είναι μονοψήφιος αριθμός, τότε το φρένο μηδενίζεται και το γκάζι αυξάνεται σταδιακά ανά μια μονάδα.

Επειδή όλη αυτή η διαδικασία προσθέτει μια σημαντική καθυστέρηση, το μεγαλύτερο μέρος της οποίας προέρχεται από την πρόβλεψη του CNN 3 και επειδή στην πράξη δεν είναι απολύτως απαραίτητο να ελέγχεται και να μεταβάλλεται η ταχύτητα του αυτοκινήτου σε κάθε frame, ορίζουμε έναν μετρητή ο οποίος καθορίζει πόσο συχνά θα γίνεται αυτή η διαδικασία. Έτσι η συχνότητα ελέγχου και μεταβολής της ταχύτητας του αυτοκινήτου γίνεται μια παράμετρος που θέτουμε εμείς πριν ενεργοποιήσουμε την αυτόνομη οδήγηση. Οι τρεις πιο συνήθεις τιμές που δοκιμάστηκαν για αυτήν την παράμετρο είναι τα 2, 5 και 10 frames.

Αυτή η παραχώρηση γίνεται για να διατηρήσουμε ένα σχετικά καλό frame rate, αν και πάλι υποβαθμίζει τις δυνατότητες του αλγορίθμου. Το πρόβλημα εμφανίζεται στην περίπτωση μεγάλης ευθείας που ακολουθείται από απότομη στροφή. Σε αυτήν την περίπτωση είναι πολύ σημαντικό να ανιχνευτεί σύντομα αυτή η αλλαγή της καμπυλότητας του δρόμου, ώστε το αυτοκίνητο να προσαρμόσει την ταχύτητα του γρήγορα. Με τη μείωση της συχνότητας του ελέγχου της ταχύτητας, υπάρχει ο κίνδυνος να παρουσιαστεί μια τέτοια περίπτωση στροφής και το αυτοκίνητο να αντιδράσει μετά από έως και 10 frames, αν υποθέσουμε ότι έχουμε ορίσει την παράμετρο ίση με 10. Γι' αυτόν τον λόγο προτείνεται να ορίζουμε την παράμετρο ανάλογα με τα χαρακτηριστικά της εκάστοτε διαδρομής.

4.7 Γραφική Διεπαφή Χρήστη, interface_v4.py

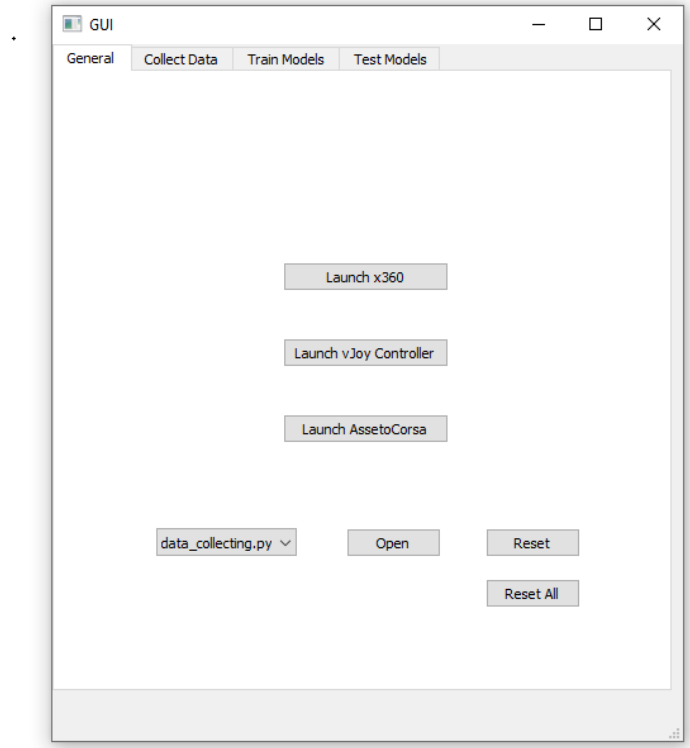
Η ανάγκη δημιουργίας της διεπαφής γεννήθηκε όταν κατά τη διάρκεια εκπόνησης της εργασίας, αντιλήφθηκα πόσο χρονοβόρα και επαναλαμβανόμενη ήταν όλη η διαδικασία, από τη δημιουργία ενός νέου dataset, μέχρι την εκπαίδευση του και την δοκιμή του στην πράξη. Κάθε φορά έπρεπε να αλλάζω με το χέρι, παραμέτρους και ονόματα αρχείων, έπρεπε να ανοίγω χειροκίνητα ένα, ένα αρχείο και να το τρέχω και πάλι χειροκίνητα να ενεργοποιώ το TensorBoard για να δω τα αποτελέσματα της εκπαίδευσης των μοντέλων. Όταν αυτή η ίδια ρουτίνα γίνεται δεκάδες και εκατοντάδες φορές, αυτές οι καθυστερήσεις και ο εκνευρισμός αθροίζονται.

Η γραφική διεπαφή χρήστη οργανώνει σε ένα μέρος όλα τα στάδια από την συλλογή των δεδομένων και την εκπαίδευση των μοντέλων, μέχρι την ενεργοποίηση και τη χρήση στην πράξη, του τελικού

αλγορίθμου αυτόνομης οδήγησης. Αυτοματοποιεί μερικές διαδικασίες και κρύβει τα πάντα πίσω από κουμπιά (Buttons), κελιά εισαγωγής κειμένου (TextBoxes) και αναπτυσσόμενες λίστες (ComboBoxes). Έτσι όλη η εμπειρία είναι πιο ομαλή και ευχάριστη, ενώ με μερικές συμβουλές και οδηγίες, μπορεί να πειραματιστεί και κάποιος χωρίς να ξέρει πως ακριβώς δουλεύει ο κώδικας από πίσω.

Η διεπαφή χωρίζεται σε 4 καρτέλες:

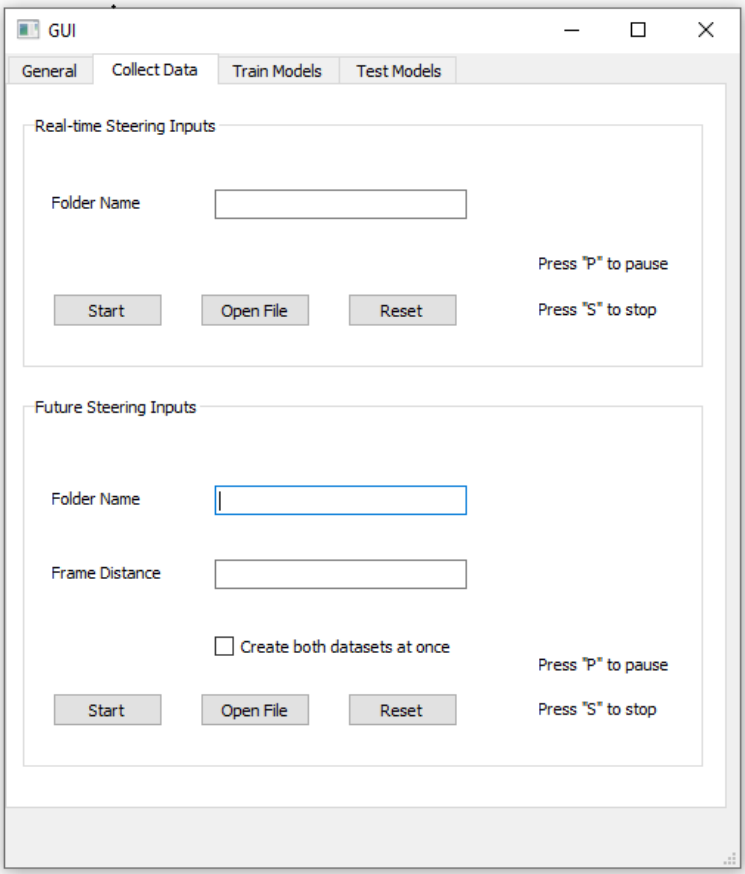
- Η πρώτη καρτέλα «**General**», εικόνα 4.15, μας επιτρέπει με το πάτημα ενός κουμπιού να ανοίξουμε κάθε ένα από τα εξωτερικά προγράμματα που χρησιμοποιούμε, όπως είναι το x360, το vJoy Controller και το παιχνίδι το Assetto Corsa. Επίσης μπορούμε από ένα *combobox* να επιλέξουμε οποιοδήποτε από τα αρχεία της εργασίας, να το ανοίξουμε και αν θέλουμε να το τροποποιήσουμε. Στη συνέχεια βέβαια με το κουμπί «*Reset*», μπορούμε να επιστρέψουμε το αρχείο στην αρχική του μορφή, έτσι μπορεί οποιοσδήποτε να κάνει αλλαγές σε ένα αρχείο, χωρίς να «σπάσει» το πρόγραμμα. Τέλος με το «*Reset All*» μπορούμε να επιστρέψουμε όλα τα αρχεία στην αρχική τους μορφή.



Εικόνα 4.14. Γραφική διεπαφή χρήστη, καρτέλα «General».

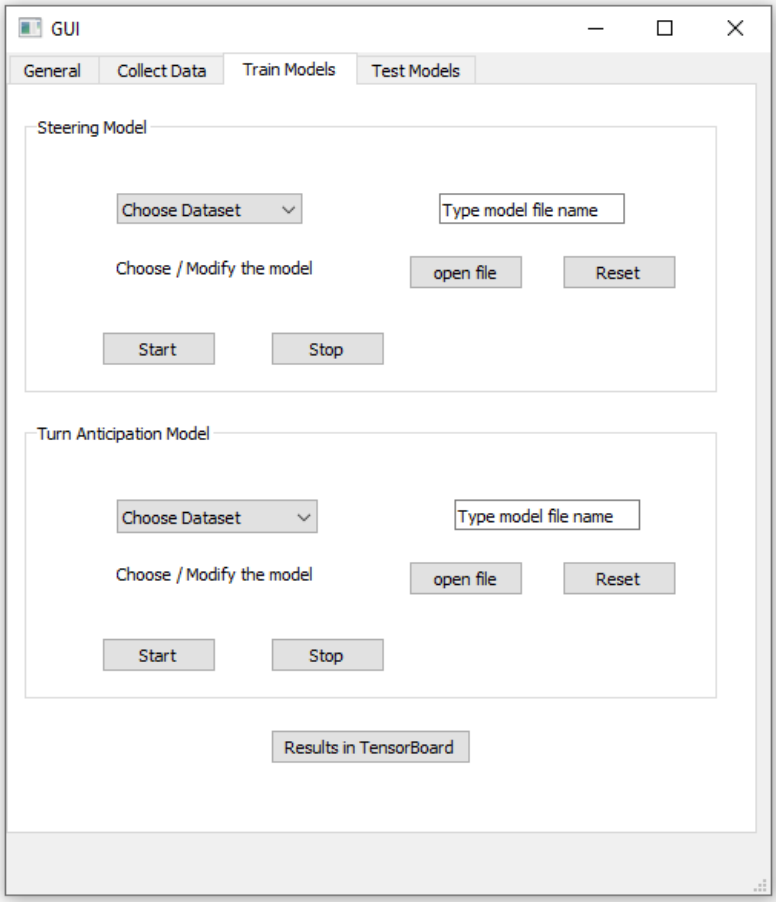
- Στη δεύτερη καρτέλα «**Collect Data**», εικόνα 4.16, μπορούμε να δημιουργήσουμε Datasets. Χωρίζεται σε δύο μέρη, «Real-time Steering Inputs» και «Future Steering Inputs». Τα δύο αυτά μέρη αφορούν τα Datasets για το CNN 2 και το CNN 3 αντίστοιχα. Επιπλέων με το *CheckBox* «*Create both datasets at once*», μπορούμε να δημιουργούμε και τα δύο ταυτόχρονα. Μέσω των κελιών «Folder Name» ορίζουμε το όνομα του φακέλου μέσα στον οποίο θα αποθηκευτούν όλα τα αρχεία του Dataset. Στο «Future Steering Inputs» υπάρχει ένα επιπλέον κελί στο οποίο ορίζουμε την απόσταση σε frames, μεταξύ της εικόνας και της ενέργειας της κάθε καταγραφής στο Dataset. Ή αλλιώς την ολίσθηση που θα εφαρμοστεί πριν την αποθήκευση των δεδομένων, όπως εξηγήθηκε στην ενότητα 4.4.3. Το κουμπί «*Start*» εκκινεί τη συλλογή των δεδομένων. Τέλος όπως φαίνεται και στην εικόνα, υποδεικνύονται τα δύο κουμπιά του πληκτρολογίου, με τα οποία μπορούμε να κάνουμε παύση (Pause) και λήξη (Stop) της συλλογής των δεδομένων. Τα κουμπιά «*Open File*» και

«*Reset*» έχουν την ίδια λειτουργία με την προηγούμενη καρτέλα, αλλά περιορίζονται κάθε φορά στο συγκεκριμένο αρχείο που είναι υπεύθυνο για τη συγκεκριμένη λειτουργία.



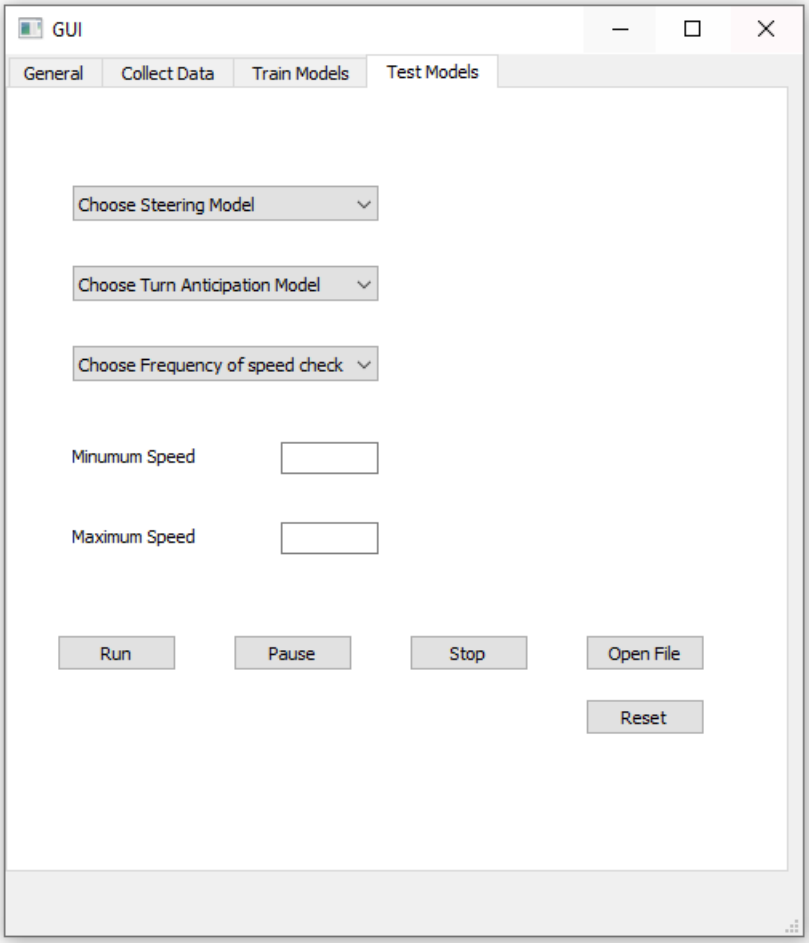
Εικόνα 4.15. Γραφική διεπαφή χρήστη, καρτέλα «Collect Data».

- Η τρίτη καρτέλα «**Train Models**», εικόνα 4.17, είναι για την εκπαίδευση των μοντέλων. Η διεπαφή μας δίνει τη δυνατότητα να εκπαιδεύσουμε μόνο τα CNN 2 και CNN 3, στις περιοχές Steering Model και Turn Anticipation αντίστοιχα. Δεν υπάρχει δυνατότητα εκπαίδευσης του CNN 1, καθώς δεν υπάρχει λόγος αλλαγής του μοντέλου και εκπαίδευσης από την αρχή, εφόσον το ήδη εκπαιδευμένο από εμένα μοντέλο, συμπεριλαμβάνεται στα αρχεία και δουλεύει σε κάθε περίπτωση. Στις δύο περιοχές για το CNN 2 και το CNN 3, μπορούμε να επιλέξουμε από το αντίστοιχο ComboBox το Dataset με το οποίο θέλουμε αν εκπαιδεύσουμε το μοντέλο. Τα ComboBox γεμίζουν αυτόματα με τα Dataset που έχουμε ήδη δημιουργήσει και έχουμε αποθηκεύσει στον υπολογιστή. Στα TextBoxes ορίζουμε το όνομα με το οποίο θα αποθηκευτεί το τρέχον μοντέλο που εκπαιδεύουμε. Το κουμπί «*Start*» εκκινεί την εκπαίδευση και το κουμπί «*Stop*» την τερματίζει πρόωρα. Τα κουμπιά «*open file*» και «*Reset*», έχουν την ίδια λειτουργία όπως και στην προηγούμενη καρτέλα. Το κουμπί «*Results in TensorBoard*» ενεργοποιεί τον TensorBoard server στην καθορισμένη πόρτα και στη συνέχεια ανοίγει μια σελίδα στον Browser στην καθορισμένη διεύθυνση.



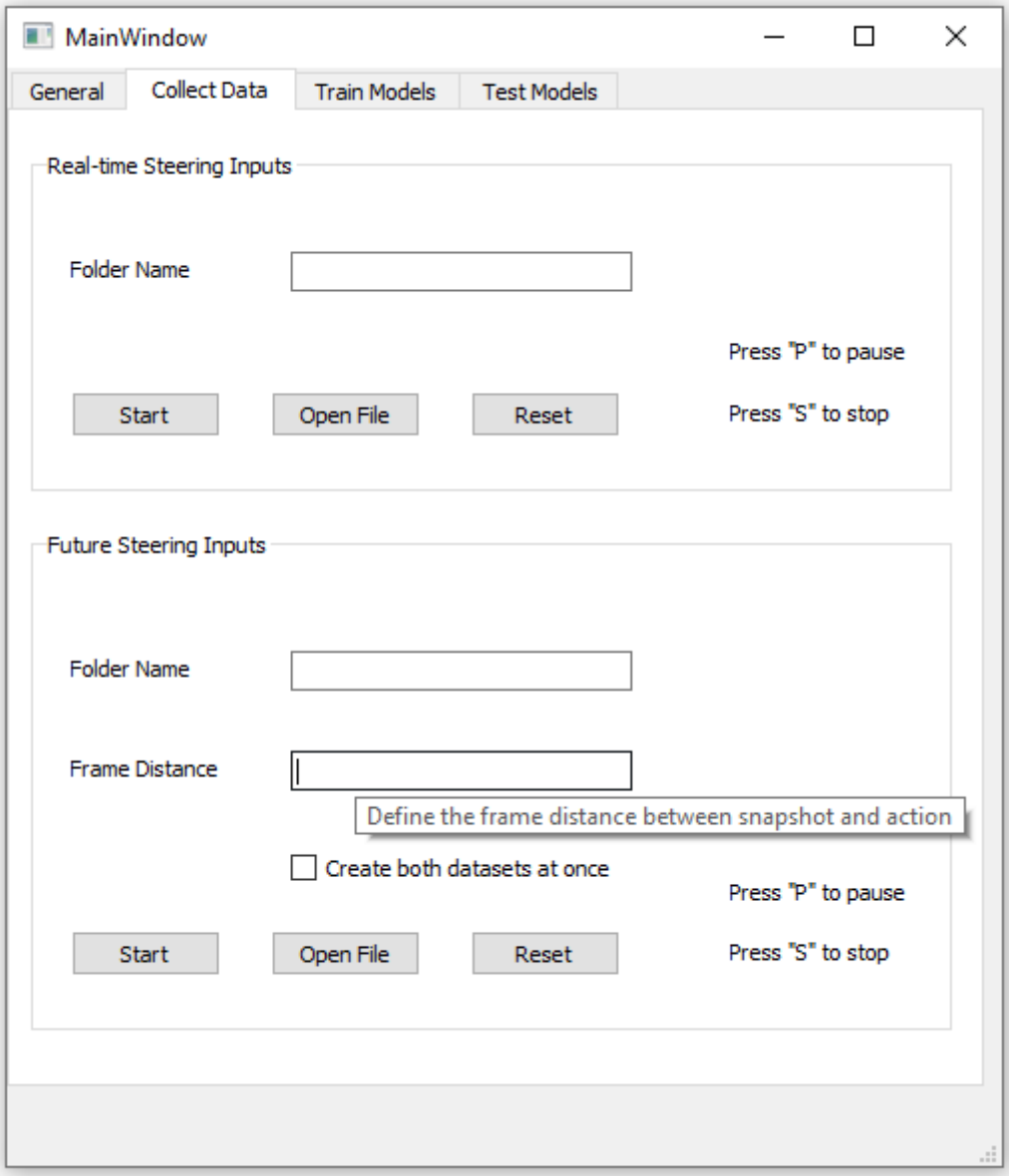
Εικόνα 4.16. Γραφική διεπαφή χρήστη, καρτέλα «Train Models».

- Μέσω της καρτέλας «**Test Models**», εικόνα 4.18, μπορούμε να δοκιμάσουμε τον αλγόριθμο στην πράξη, επιλέγοντας από τρία *ComboBoxes*, τα δύο μοντέλα που θα χρησιμοποιήσουμε, «*Steering Model*» και «*Turn Anticipation Model*» και τη συχνότητα του ελέγχου της ταχύτητας με την οποία κινείται το αυτοκίνητο «*Frequency of speed check*» (2, 5, 10). Στα κελιά «Minimum Speed» και «Maximum Speed» ορίζουμε την ελάχιστη και τη μέγιστη ταχύτητα με την οποία μπορεί να κινείται το αυτοκίνητο. Τα κουμπιά «*Run*», «*Pause*» και «*Stop*», εκκινούν, διακόπτουν και τερματίζουν αντίστοιχα την αυτόνομη οδήγηση. Τα κουμπιά «Open File» και «Reset», έχουν την ίδια ακριβώς λειτουργία με τις προηγούμενες καρτέλες.



Εικόνα 4.17. Γραφική διεπαφή χρήστη, καρτέλα «Test Models».

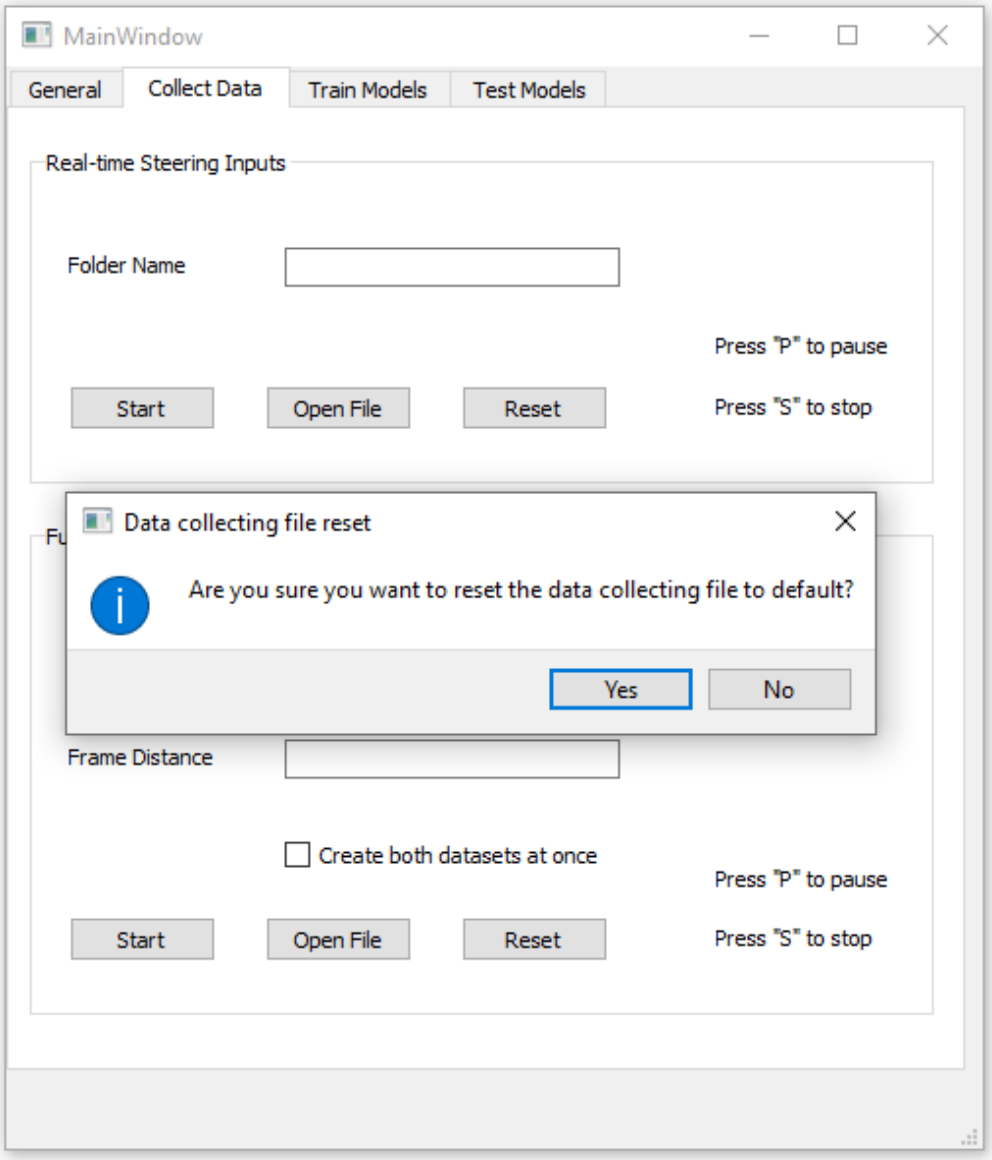
Η πλειοψηφία των Button, ComboBox και TextBox, έχουν εξήγηση της λειτουργίας τους η οποία εμφανίζεται σαν ετικέτα, όταν περνάμε από πάνω τους με το ποντίκι, όπως στο παράδειγμα της εικόνας 4.18, με την εξήγηση του TextBox «*Frame Distance*»



Εικόνα 4.18. Ετικέτα του κελιού Frame Distance.

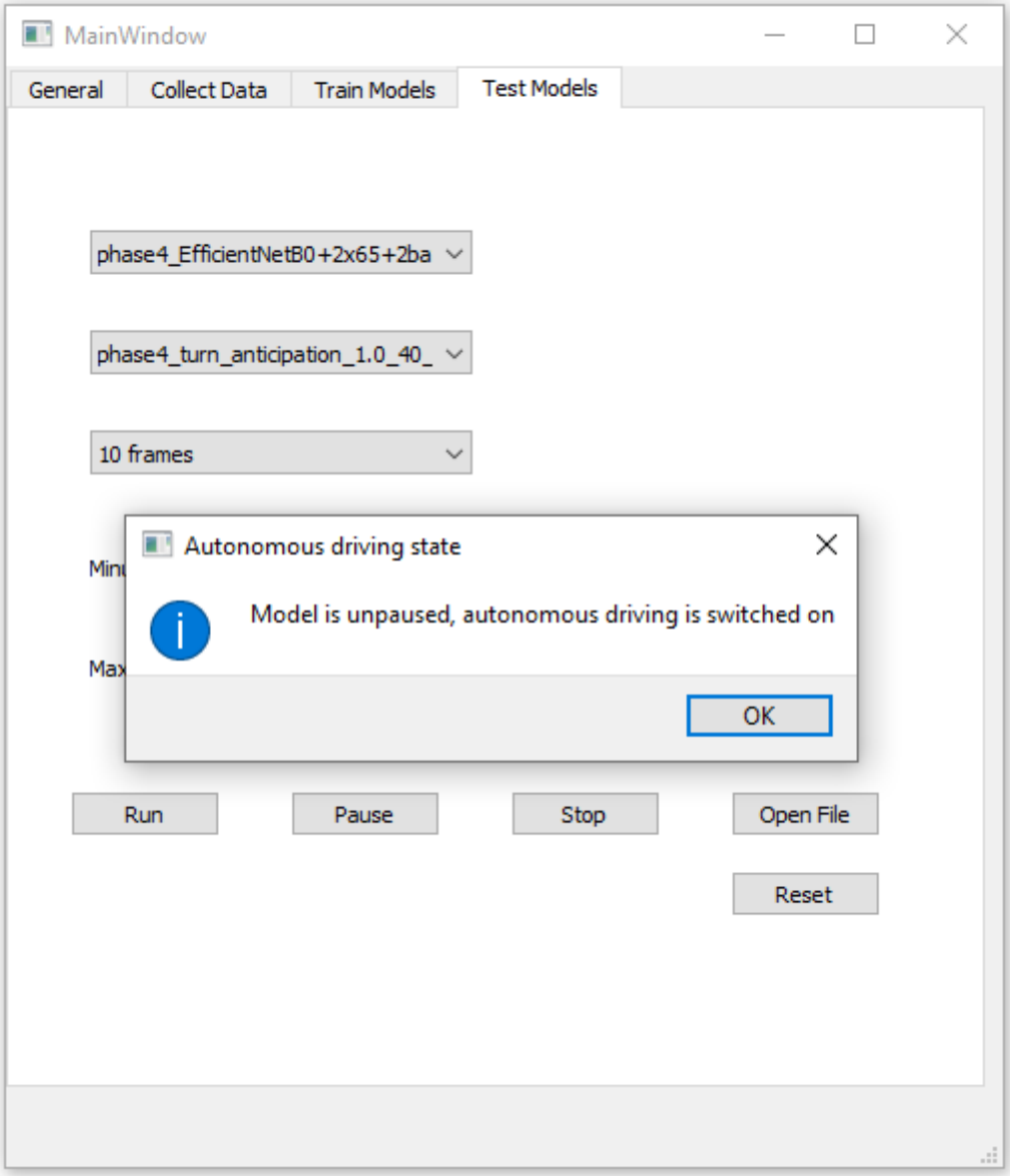
Επιπλέον σε μερικές ενέργειες, υπάρχουν αναδυόμενα παράθυρα τα οποία είτε ζητούν επιβεβαίωση για μια συγκεκριμένη ενέργεια, είτε ενημερώνουν το χρήστη για την αλλαγή που προκάλεσε μια συγκεκριμένη ενέργεια, είτε ενημερώνουν τον χρήστη ότι η συγκεκριμένη ενέργεια δεν μπορεί να πραγματοποιηθεί. Για παράδειγμα:

- Όταν πατηθεί οποιοδήποτε κουμπί «*Reset*», το αναδυόμενο παράθυρο ζητά την επιβεβαίωση του χρήστη, πριν προχωρήσει στο reset του αρχείου, εικόνα 4.19.



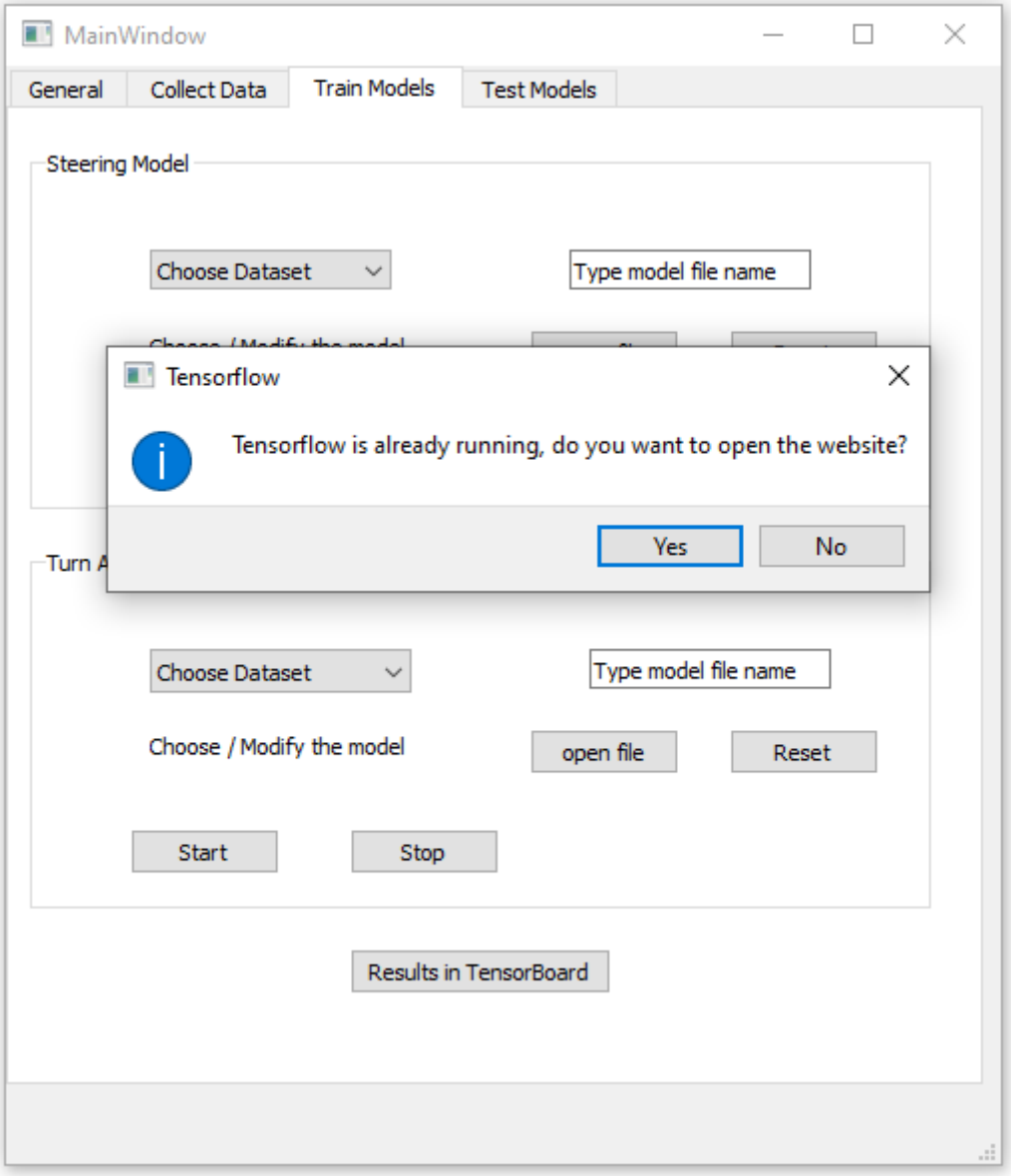
Εικόνα 4.19. Αναδυόμενο παράθυρο μετά από το πάτημα του button «Reset».

- Όταν πατηθεί το κουμπί «*Pause*», για παύση ή εκκίνηση της αυτόνομης οδήγησης, το αναδυόμενο παράθυρο μας ενημερώνει ότι η αυτόνομη ενεργοποιήθηκε ή απενεργοποιήθηκε αντίστοιχα, εικόνα 4.20.



Εικόνα 4.20. Αναδυόμενο παράθυρο μετά το πάτημα του button «Pause».

- Όταν πατηθεί για δεύτερη φορά το κουμπί «*Results in TensorBoard*», το αναδυόμενο παράθυρο ενημερώνει τον χρήστη ότι το TensorBoard είναι ήδη σε λειτουργία και τον ρωτάει αν θέλει να το ανοίξει στον Browser, εικόνα 4.21.



Εικόνα 4.21. Αναδυόμενο παράθυρο μετά το πάτημα του button «Results in TensorBoard».

Κεφάλαιο 5ο: Αποτέλεσμα - Βελτιώσεις

5.1 Αποτέλεσμα

Το αποτέλεσμα αυτής της εργασίας, είναι να έχουμε δημιουργήσει έναν αλγόριθμο, ο οποίος χρησιμοποιώντας τρία συνελικτικά νευρωνικά δίκτυα και λίγη απλή προγραμματιστική λογική, οδηγάει επιτυχώς ένα αυτοκίνητο σε πολλούς διαφορετικούς δρόμους, οι οποίοι όμως έχουν μερικά κοινά χαρακτηριστικά, μέσα σε έναν προσομοιωτή οδήγησης.

Ο αλγόριθμος αποδίδει στο μέγιστο, σχεδόν αλάνθαστα, σε δρόμους ενός ρεύματος, με παρουσία γραμμών οδοστρώματος σε όλο το μήκος τους και χωρίς την παρουσία άλλων οχημάτων ή εμποδίων. Παρά ταύτα ανάλογα την περίπτωση και μερικές φορές προς έκπληξη μας, αποδίδει ικανοποιητικά και σε διαφορετικά περιβάλλοντα, καταφέρνοντας ως επί το πλείστον να ολοκληρώσει διαδρομές χωρίς πολλά και μεγάλα λάθη. Χαρακτηριστικό παράδειγμα είναι πίστες αγώνων, στις οποίες υπάρχει από λίγο έως καθόλου παρουσία γραμμών στο οδόστρωμα. Αυτό μας δείχνει ότι τα μοντέλα πετυχαίνουν ικανοποιητική γενίκευση στα δεδομένα με τα οποία τροφοδοτούνται.

Όσο ισχύουν όμως οι παραπάνω προϋποθέσεις, η απόδοση του είναι ιδιαίτερα αξιόπιστη ακόμα και όταν οι δρόμοι στους οποίους δοκιμάζεται διαφοροποιούνται από αυτούς στους οποίους εκπαιδεύτηκε, σε θέματα όπως, χρώμα οδοστρώματος, χρώμα γραμμών, κατάσταση γραμμών (ευκρίνεια, συνεχόμενες ή διακεκομμένες γραμμές), κλίση και καμπυλότητα δρόμου και χαρακτηριστικά του περιβάλλοντος γύρω από τον δρόμο. Και πάλι αυτό είναι δείγμα της καλής γενίκευσης των μοντέλων στα δεδομένα.

Επιπλέων έχουμε δημιουργήσει, μία μικρή εφαρμογή με γραφική διεπαφή, μέσω της οποίας ένας χρήστης, ο οποίος ιδανικά έχει μερικές βασικές γνώσεις προγραμματισμού και μηχανικής μάθησης, μπορεί με ευκολία να δημιουργήσει από μόνος του συλλογές δεδομένων, ενδεχομένως με μερικές τροποποιήσεις σε οποιοδήποτε παρόμοιο παιχνίδι ή άλλο περιβάλλον προσομοίωσης επιθυμεί, να εκπαιδεύσει νευρωνικά δίκτυα και να δοκιμάσει τον συνολικό αλγόριθμο αυτόνομης οδήγησης στην πράξη. Επιπλέον έχει τη δυνατότητα και την ελευθερία να πειραματιστεί, μεταβάλλοντας παραμέτρους, ή και ολόκληρα κομμάτια του κώδικα, ξεκινώντας όμως από κάτι που ήδη δουλεύει επαρκώς και έχοντας στη διάθεση του έτοιμα όλα τα απαραίτητα εργαλεία.

5.2 Βελτιώσεις

Οι δυνατότητες για βελτιώσεις ή/και προσθήκες στην παρούσα εργασία είναι πάρα πολλές. Είναι πολλές οι ιδέες για προσθήκη λειτουργιών ή επέκταση των υπαρχόντων και γενικά για περαιτέρω πειραματισμό. Ο λόγος που η εργασία «περιορίστηκε» στην παρούσα μορφή, είναι είτε λόγω περιορισμένου χρόνου, ή λόγω περιορισμών από άποψη hardware. Μάλιστα τα δύο συνδέονται μεταξύ τους, μιας και όσο ισχυρότερο είναι το υπολογιστικό σύστημα που διαθέτει κάποιος, τόσο περισσότερα μπορεί να πετύχει και σε λιγότερο χρόνο.

Με βάση αυτό θα χωρίσω τις προτάσεις βελτίωσης, σε προτάσεις που απαιτούν μόνο επένδυση σε χρόνο και σε προτάσεις που επιπλέον απαιτούν ικανότερο hardware.

5.2.1 Βελτιώσεις που απαιτούν μόνο χρόνο

Παρακάτω περιγράφονται συνοπτικά ιδέες για βελτιώσεις οι οποίες απαιτούν μόνο επένδυση χρόνου:

- Αντικατάσταση των CNN 2 και CNN 3 με ένα CNN το οποίο προβλέπει την ιδανική θέση του τιμονιού για το τρέχον frame και ταυτόχρονα για έναν πεπερασμένο αριθμό μελλοντικών frame. Η ιδέα είναι ότι έτσι το πρόγραμμα θα διαθέτει μια πιο αναλυτική ιδέα της καμπυλότητας του δρόμου, γλυτώνοντας ταυτόχρονα την καθυστέρηση που προσθέτει το δεύτερο CNN.
- Προσαρμογή του πόσο μακριά «βλέπει» το πρόγραμμα, ανάλογα με την ταχύτητα που κινείται. Στην πράξη αυτό σημαίνει ότι όσο μεγαλύτερη είναι η τρέχουσα ταχύτητα του αυτοκινήτου, τόσο πιο μελλοντική πρόβλεψη καμπυλότητας του δρόμου θα χρησιμοποιεί για να προσαρμόζει την ταχύτητα του. Σκοπός είναι οι πιο αποδοτικές αυξομειώσεις της ταχύτητας, με σκοπό τη διατήρηση της μέγιστης δυνατής ταχύτητας κατά την οδήγηση χωρίς την έξοδο από το δρόμο.
- Πειραματισμός με περισσότερα ήδη δρόμων, π.χ. πίστες αγώνων.
- Δοκιμή περισσότερων προ-εκπαιδευμένων CNN Και περισσότερων εκδοχών των ήδη δοκιμασμένων. Ειδικά πειραματισμός με μικρότερα νευρωνικά δίκτυα, βασιζόμενοι στην ιδέα ότι η γενίκευση είναι πιο σημαντική από την απόλυτη ακρίβεια των προβλέψεων και το τέλειο train / validation loss. Ενδεικτικά η εκπαίδευση ενός CNN του σκέλους EfficientNetB0 με το hardware που χρησιμοποιήθηκε για την εργασία χρειάζεται περίπου 12 ώρες για 60 εποχές.
- Βελτίωση της γραφικής διεπαφής χρήστη. Διόρθωση bugs, προσθήκη παραπάνω λειτουργιών, αισθητική βελτίωση.
- Βελτίωση γενικά της εμπειρίας χρήσης της εφαρμογής, αυτοματοποιώντας μερικά πράγματα, που έχουν κυρίως να κάνουν με το «στήσιμο» των διάφορων components (συστατικών), που συμμετέχουν στο συνολικό σύστημα, όπως είναι οι third party εφαρμογές. Για παράδειγμα, αλλαγή των συντεταγμένων των screenshot, ανάλογα με την ανάλυση της οθόνης του χρήστη, ώστε να μη χρειάζεται μετακίνηση χειροκίνητα του παραθύρου του παιχνιδιού στο σωστό σημείο που καταγράφει το πρόγραμμα. Κάτι αντίστοιχο και με την ένδειξη της ταχύτητας.
- Απόπειρα χρήσης διαφορετικών αυτοκινήτων και οδήγησης με μεγαλύτερες ταχύτητες.
- Προσθήκη κάποιου είδους χρονομετρησμού, ως τρόπο αξιολόγησης των μοντέλων, αλλά και για λόγους ψυχαγωγίας.
- Δημιουργία leaderboard (πίνακας κατάταξης), όπου διάφοροι συνδυασμοί αλγορίθμου και αυτοκινήτων θα «ανταγωνίζονται» μεταξύ τους.

5.2.2 Βελτιώσεις που απαιτούν ισχυρότερο hardware

Παρακάτω περιγράφονται συνοπτικά ιδέες για βελτιώσεις οι οποίες εκτός από επιπλέον χρόνο, απαιτούν και ισχυρότερο hardware:

- Δημιουργία μεγαλύτερων dataset σε αριθμό καταγραφών και/ή ανάλυσης εικόνας. Ο περιορισμός εδώ είναι το μέγεθος της ram του υπολογιστή. Ο υπολογιστής που χρησιμοποιήθηκε για την εργασία διαθέτει 16 GB RAM.
- Χρήση μεγαλύτερων, ικανότερων νευρωνικών δικτύων σε συνδυασμό με τα μεγαλύτερα datasets.
- Αύξηση του batch size, δηλαδή του μεγέθους της παρτίδας που τροφοδοτείται στο μοντέλο σε κάθε κύκλο εκπαίδευσης. Αυτήν την στιγμή το μέγεθος δεν μπορεί να ξεπεράσει το 8 λόγω μη επάρκειας RAM και VRAM.
- Έλεγχος της ταχύτητας και της καμπυλότητας του δρόμου και προσαρμογή της ταχύτητας του αυτοκινήτου σε κάθε frame.
- Επίτευξη μεγαλύτερου frame rate λόγω καλύτερου hardware.
- Αύξηση της ανώτατης ταχύτητας του αυτοκινήτου, βασιζόμενοι στο καλύτερο frame rate.

- Προσθήκη επιπλέον νευρωνικών δικτύων που χρησιμοποιούνται παράλληλα και προσθέτουν λειτουργίες όπως η αναγνώριση αντικειμένων και εμποδίων.
- Προσθήκη λειτουργίας αλλαγής λωρίδας και/ή προσπέρασης, χρησιμοποιώντας πληροφορίες από τα προαναφερόμενα νέα νευρωνικά δίκτυα.
- Αναγνώριση σημάτων και προσθήκη λειτουργιών όπως αυτόματη προσαρμογή ταχύτητας ανάλογα με τα όρια που ορίζουν τα σήματα, πρόβλεψη στροφών μέσω σημάτων και άλλα.

5.2.3 Επιπλέον πιο αισιόδοξη βελτίωση.

Χρήση ενισχυτικής μάθησης η οποία με κάποιον τρόπο παρακολουθεί και αξιολογεί την «απόδοση» του μοντέλου και ανάλογα επιβραβεύει ή τιμωρεί. Αποτέλεσμα το μοντέλο να βελτιώνεται μόνο του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Yu Huang, Yue Chen, “Survey of the State-of-Art Autonomous Driving Technologies with Deep Learning”, 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Macau, China, 11-14 Δεκεμβρίου 2020.

[2] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακκελαρίου, Τεχνητή Νοημοσύνη. Εκδόσεις Πανεπιστημίου Μακεδονίας, 2020

[3] Solveig Badillo, Balazs Benfai, Fabian Birzele, Iakov I. Davydov, Lucy Hutchinson, Tony Kam-Thong, Juliane Siebourg-Polster, Bernhard Steiert, Jitao David Zhang, An Introduction to Machine Learning. Wiley Periodicals, inc, 03 Μαρτίου 2020.

[4] Unsupervised Learning | IBM. https://www.ibm.com/cloud/learn/unsupervised-learning, 21 Σεπτεμβρίου 2020

[5] Supervised Learning | IBM. https://www.ibm.com/cloud/learn/supervised-learning, 19 Αυγούστου 2020.

[5] Geoffrey, Deep Learning – Activation Function. https://towardsaws.com/activation-function-f76bfc03e215, 31 Αυγούστου 2022.

[6] Jason Brownlee, “A Gentle Introduction to Cross-Entropy for Machine Learning”, https://machinelearningmastery.com/cross-entropy-for-machine-learning/, 22 December 2020.

[7] Keiron O’Shea, Ryan Nash, “An Introduction to Convolutional Neural Networks”, 26 Νοεμβρίου 2015.

[8] Charles Siegel, Jeff Daily, Abhinav Vishnu, “Adaptive Neuron Apoptosis for Accelerating Deep Learning on Large Scale Systems”. Pacific Northwest National Laboratory Richland, WA 99352, Οκτοβρίου 2016.

[9] Mayank Mishara, “Convolutional Neural Networks, Explained”, Towards Data Science, https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939, 26 Αγούστου 2020.

[9] Jason Brownlee, “A Gentle Introduction to Cross-Entropy for Machine Learning”, https://machinelearningmastery.com/cross-entropy-for-machine-learning/, 22 December 2020.

[10] Pavan Sanagapati, “What is Pooling in Deep Learning”, https://www.kaggle.com/questions-and-answers/59502.

[11] Nitish Srivastana, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting», Journal of Machine Learning Research 15, 14 Ιουνίου 2014.

[12] Sergey loffe, Christian Szegedy, «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift», 2 Μαρτίου 2015.

[13] Saunita Nayak, Batch Normalization in Deep Networks, https://learnopencv.com/batch-normalization-in-deep-networks/, 5 Ιουλίου 2018.

[14] Saurav Singla, «Why is Batch Normalization useful in deep neural Networks», https://towardsdatascience.com/batch-normalisation-in-deep-neural-network-ce65dd9e8dbf.

[15] Abhishek Singh, «What is Image Data Augmentation», https://www.kaggle.com/getting-started/190280.

[16] Mingxing Tan, Quoc V. Le, «EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks», International Conference on Machine Learning, 28 Μαίου 2019.

[17] CARLA, https://carla.org/.

[18] PGDRIVE, https://github.com/decisionforce/pgdrive.

[19] Assetto Corsa Wikipedia Page, https://en.wikipedia.org/wiki/Assetto_Corsa.

[20] Python os library official documentation, https://docs.python.org/3/library/os.html.

[21] Python time library official documentation, https://docs.python.org/3/library/time.html

[22] Python sys library official documentation, https://docs.python.org/3/library/sys.html.

[23] Python re library official documentation, https://docs.python.org/3/library/re.html.

[24] Python webbrowser official documentation, https://docs.python.org/3/library/webbrowser.html.

[25] Python subprocess official documentation, https://docs.python.org/3/library/subprocess.html.

[26] Python threading official documentation, https://docs.python.org/3/library/threading.html.

[27] Python pywin32 library documentation, http://timgolden.me.uk/pywin32-docs/contents.html.

[28] Python ctypes library official documentation, https://docs.python.org/3/library/ctypes.html.

[29] Python pymput library documentation, https://pypi.org/project/pynput/.

[30] Python shutil library official documentation, https://docs.python.org/3/library/shutil.html.

[31] Python struct library official documentation, https://docs.python.org/3/library/struct.html.

[32] Python pyqt5 library documentation, https://www.riverbankcomputing.com/static/Docs/PyQt5/.

[33] Python cv2 library documentation, https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.

[34] Python numpy library documentation, https://numpy.org/doc/stable/user/whatisnumpy.html.

[35] TensorFlow official website, https://www.tensorflow.org/.

[36] TensorBoard official website, https://www.tensorflow.org/tensorboard.

[37] Sentdex GitHub repository, pygta5, grabscreen.py https://github.com/Sentdex/pygta5/blob/master/original_project/grabscreen.py

[38] Προσομοιωτής Xbox Controller x360ce, https://www.x360ce.com/.

[39] Shaul Eizikovich GitHub, https://github.com/shauleiz.

[40] VJoy implementation using data from Wireless IMU, https://carpathianbyte.ro/home/index.php/portfolio/vjoy-implementation-using-data-from-wireless-imu/.

[41] Ryan Barnes GitHub , https://github.com/bayangan1991/PYXInput/blob/master/pyxinput/read_state.py

Christian Janiesch, Patrick Zschech, Kai Heinrich, Machine learning and deep learning, Exelctronic Markets 31, 8 Απριλίου 2021.

Ruoyu Sun, Optimization for deep learning: theory and algorithms, 21 Δεκεμβρίου 2019.

Connor Shorten, Taghi M.Khoshgoftaar, A survey on Image Data Augmentation for Deep Learning, Journal of Big Data 6, 6 Ιουλίου 2019.

Saad Albawi, Tareq Abed Mohammed, Saad Al-Zawi, Understanding of a convolutional neural network, 2017 International Conference on Engineering and Technology (ICET), 23 Αυγούστου 2017.

Nadia Jmour, Sehla Zayen, Afef Abdelkrim, Convolutional neural networks for image classification, 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET), 25 Μαρτίου 2018.

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας Εργασίας

- **Grabscreen.py**

```python
# Done by Franneckl
import cv2
import numpy as np
import win32gui, win32ui, win32con, win32api
import time

def grab_screen(region=None):
    hwin = win32gui.GetDesktopWindow()
    if region:
            left,top,x2,y2 = region
            width = x2 - left
            height = y2 - top
    else:
        width = win32api.GetSystemMetrics(win32con.SM_CXVIRTUALSCREEN)
        height = win32api.GetSystemMetrics(win32con.SM_CYVIRTUALSCREEN)
        left = win32api.GetSystemMetrics(win32con.SM_XVIRTUALSCREEN)
        top = win32api.GetSystemMetrics(win32con.SM_YVIRTUALSCREEN)

    hwindc = win32gui.GetWindowDC(hwin)
    srcdc = win32ui.CreateDCFromHandle(hwindc)
    memdc = srcdc.CreateCompatibleDC()
    bmp = win32ui.CreateBitmap()
    bmp.CreateCompatibleBitmap(srcdc, width, height)
    memdc.SelectObject(bmp)
    memdc.BitBlt((0, 0), (width, height), srcdc, (left, top), win32con.SRCCOPY)

    signedIntsArray = bmp.GetBitmapBits(True)
    img = np.fromstring(signedIntsArray, dtype='uint8')
    img.shape = (height,width,4)
    img =  np.delete(img,-1 , 2)  #extradaki

    srcdc.DeleteDC()
    memdc.DeleteDC()
    win32gui.ReleaseDC(hwin, hwindc)
    win32gui.DeleteObject(bmp.GetHandle())

    #return cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  #output is already rgb
    return img

def main():
    last_time = time.time()
    while True:
        screen = grab_screen(region = (0, 200, 640, 510))
        #number1 = grab_screen(region = (0, 240, 17, 262))
        #number2 = grab_screen(region = (16, 240, 33, 262))
        #print(screen[54][54])
        print(screen.shape)
    #    print('loop took {} seconds'.format(time.time()-last_time))
    #    last_time = time.time()
    #    cv2.imshow('window', screen)
        print('loop took {} seconds'.format(time.time()-last_time))
        last_time = time.time()
        #cv2.imshow('window',cv2.cvtColor(screen, cv2.COLOR_BGR2RGB))
        cv2.imshow('window',screen)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            cv2.destroyAllWindows()
            break

if __name__ == '__main__':
    main()
```

- **Get_keys.py**

```python
# Citation: Box Of Hats (https://github.com/Box-Of-Hats )

import win32api as wapi

keyList = ["\b"]
for char in "ABCDEFGHIJKLMNOPQRSTUVWXYZ 123456789,.'£$/\\":
    keyList.append(char)

def key_check():
    keys = []
    for key in keyList:
        if wapi.GetAsyncKeyState(ord(key)):
            keys.append(key)
    return keys

```

- **Vjoy.py**

```python
# step 1: https://sourceforge.net/projects/vjoystick/files/latest/download
# step 2: SDK: http://vjoystick.sourceforge.net/site/index.php/component/weblinks/weblink/13-uncategorised/11-redirect-vjoy2sdk?task=weblink.go
# step 3: CONST_DLL_VJOY = "vJoyInterface.dll" ...KEEP .DLL local?
# step 4: http://www.x360ce.com/, 64 bit download
# step 5: extract, copy to gtav directory
# step 6: run, should auto-detect vjoy, test with example make sure it works.
# step 7: CLOSE the app, run game. Test with example to see if works.

# SOURCE: https://gist.github.com/Flandan/fdadd7046afee83822fcff003ab47087#file-vjoy-py

import ctypes
import struct

CONST_DLL_VJOY = "vJoyInterface.dll"

class vJoy(object):
    def __init__(self, reference = 1):
        self.handle = None
        self.dll = ctypes.CDLL( CONST_DLL_VJOY )
        self.reference = reference
        self.acquired = False

    def open(self):
        if self.dll.AcquireVJD( self.reference ):
            self.acquired = True
            return True
        return False

    def close(self):
        if self.dll.RelinquishVJD( self.reference ):
            self.acquired = False
            return True
        return False

    def generateJoystickPosition(self,
        wThrottle = 0, wRudder = 0, wAileron = 0,

        # left thb x        left thb y     left trigger
        wAxisX = 16393,   wAxisY = 16393,   wAxisZ = 0,

        # right thb x       right thb y        right trigger
        wAxisXRot = 16393, wAxisYRot = 16393, wAxisZRot = 0,

        # ???         ???        ???
        wSlider = 0, wDial = 0, wWheel = 0,
        # ???         ???        ???
        wAxisVX = 0, wAxisVY = 0, wAxisVZ = 0,
        # ???         ???               ???
        wAxisVBRX = 0, wAxisVBRY = 0, wAxisVBRZ = 0,
        # 1 = a
        # 2 = b  3 = a+b ??
        # 4 = x  5 = x+a ?? 6 = x+b
        # 8 = y
        lButtons = 0, bHats = 0, bHatsEx1 = 0, bHatsEx2 = 0, bHatsEx3 = 0):
```

Vjoy.py (συνέχεια)

```
        """
        typedef struct _JOYSTICK_POSITION
        {
            BYTE    bDevice; // Index of device. 1-based
            LONG    wThrottle;
            LONG    wRudder;
            LONG    wAileron;
            LONG    wAxisX;
            LONG    wAxisY;
            LONG    wAxisZ;
            LONG    wAxisXRot;
            LONG    wAxisYRot;
            LONG    wAxisZRot;
            LONG    wSlider;
            LONG    wDial;
            LONG    wWheel;
            LONG    wAxisVX;
            LONG    wAxisVY;
            LONG    wAxisVZ;
            LONG    wAxisVBRX;
            LONG    wAxisVBRY;
            LONG    wAxisVBRZ;
            LONG    lButtons;   // 32 buttons: 0x00000001 means button1 is pressed, 0x80000000 -> button32 is pressed
            DWORD   bHats;      // Lower 4 bits: HAT switch or 16-bit of continuous HAT switch
                        DWORD   bHatsEx1;   // 16-bit of continuous HAT switch
                        DWORD   bHatsEx2;   // 16-bit of continuous HAT switch
                        DWORD   bHatsEx3;   // 16-bit of continuous HAT switch
        } JOYSTICK_POSITION, *PJOYSTICK_POSITION;
        """
        joyPosFormat = "BlllllllllllllllllIIII"
        pos = struct.pack( joyPosFormat, self.reference, wThrottle, wRudder,
                                    wAileron, wAxisX, wAxisY, wAxisZ, wAxisXRot, wAxisYRot,
                                    wAxisZRot, wSlider, wDial, wWheel, wAxisVX, wAxisVY, wAxisVZ,
                                    wAxisVBRX, wAxisVBRY, wAxisVBRZ, lButtons, bHats, bHatsEx1, bHatsEx2, bHatsEx3 )
        return pos

    def update(self, joystickPosition):
        if self.dll.UpdateVJD( self.reference, joystickPosition ):
            return True
        return False

vj = vJoy()

def mytest_phase4(throttle_perc, brake_perc, steering_perc):
    vj.open()
    throttle_percentage = int(throttle_perc*(32786/100))
    brake_percentage = int(brake_perc*(32786/100))
    steering_percentage = round(steering_perc)
    joystickPosition = vj.generateJoystickPosition(wAxisX = steering_percentage,wAxisZRot = throttle_percentage,wAxisZ = brake_percentage)
    vj.update(joystickPosition)
    vj.close()

'''
if __name__ == '__main__':
'''
```

- **Read_xbox_controller_state.py**

```python
"""Read the current state of Xbox Controllers"""
from ctypes import *

# Xinput DLL
try:
    _xinput = windll.xinput1_4
except OSError as err:
    _xinput = windll.xinput1_3


class _xinput_gamepad(Structure):
    """CType XInput Gamepad Object"""
    _fields_ = [("wButtons", c_ushort), ("left_trigger", c_ubyte),
                ("right_trigger", c_ubyte), ("thumb_lx", c_short),
                ("thumb_ly", c_short), ("thumb_rx", c_short),
                ("thumb_ry", c_short)]

    fields = [f[0] for f in _fields_]

    def __dict__(self):
        return {field: self.__getattribute__(field) for field in self.fields}

    def __str__(self):
        return str(self.__dict__())

    def __getitem__(self, string):
        return self.__dict__()[string]


class _xinput_state(Structure):
    """CType XInput State Object"""
    _fields_ = [("dwPacketNumber", c_uint),
                ("XINPUT_GAMEPAD", _xinput_gamepad)]

    fields = fields = [f[0] for f in _fields_]

    def __dict__(self):
        return {field: self.__getattribute__(field) for field in self.fields}

    def __str__(self):
        return str(self.__dict__())

    def __getitem__(self, string):
        return self.__dict__()[string]


class rController(object):
    """XInput Controller State reading object"""

    # All possible button values
    _buttons = {
        'DPAD_UP': 0x0001,
        'DPAD_DOWN': 0x0002,
        'DPAD_LEFT': 0x0004,
        'DPAD_RIGHT': 0x0008,
        'START': 0x0010,
        'BACK': 0x0020,
        'LEFT_THUMB': 0x0040,
        'RIGHT_THUMB': 0x0080,
        'LEFT_SHOULDER': 0x0100,
        'RIGHT_SHOULDER': 0x0200,
        'A': 0x1000,
        'B': 0x2000,
        'X': 0x4000,
        'Y': 0x8000
    }

```

Read_xbox_controller_state.py (συνέχεια)

```
68      def __init__(self, ControllerID):
69          """Initialise Controller object
70          ControllerID    Int     Position number of desired controller (order of connection)
71          """
72          self.ControllerID = ControllerID
73          self.dwPacketNumber = c_uint()
74
75      @property
76      def gamepad(self):
77          """Returns the current gamepad state. Buttons pressed is shown as a raw integer value.
78          Use rController.buttons for a list of buttons pressed.
79          """
80          state = _xinput_state()
81
82          _xinput.XInputGetState(self.ControllerID - 1, pointer(state))
83          self.dwPacketNumber = state.dwPacketNumber
84          return state.XINPUT_GAMEPAD
85
86      @property
87      def buttons(self):
88          """Returns a list of buttons currently pressed"""
89          return [name for name, value in rController._buttons.items()
90                  if self.gamepad.wButtons & value == value]
91
92
93  def main():
94      """Test the functionality of the rController object"""
95      import time
96
97      print('Testing controller in position 1:')
98      print('Running 3 x 3 seconds tests')
99
100     # Initialise Controller
101     con = rController(1)
102
103     # Loop printing controller state and buttons held
104     for i in range(3):
105         print('Waiting...')
106         time.sleep(2.5)
107         print('State: ', con.gamepad)
108         print('Buttons: ', con.buttons)
109         time.sleep(0.5)
110
111     print('Done!')
112
113
114 if __name__ == '__main__':
115     main()
```

- **Data_collecting.py**

```python
import os, time
import numpy as np
from read_xbox_controller_state import rController
from grabscreen import grab_screen
from getkeys import key_check
from threading import Event


def data_collecting_main(folder_name, event):
    #elegxos uparkshs onomatos dataset / dhmiourgia apo to 0 an den uparxei
    starting_value = 0
    print(folder_name)
    if os.path.exists('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder/'+folder_name):
        print('folder exists, we are proceeding')
    else:
        os.mkdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder/'+folder_name)
        print('folder created')
    while True:
        data_folder_path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder/'
        files_name = '/training_data-{0}.npy'
        file_name_temp = data_folder_path+folder_name+files_name
        file_name = file_name_temp.format(starting_value)
        print(file_name)
        if os.path.isfile(file_name):
            print('File exists, moving along', starting_value)
            starting_value += 1
        else:
            print('File does not exist, starting fresh!', starting_value)
            break
    start_collecting_data(file_name, starting_value, folder_name, event)


def start_collecting_data(file_name, starting_value, folder_name, event):
    print('Starting test in 3 seconds')
    time.sleep(1)
    print('3')
    time.sleep(1)
    print('2')
    time.sleep(1)
    print('1')
    con = rController(1)
    training_data = []
    last_time = time.time()
    paused = False

    while True:
        if not paused:
            screen = grab_screen(region = (0, 200, 640, 510))  #screenshot
            screen = screen.astype("uint8")
            #print(screen.shape)
            gamepad_state = con.gamepad
            left_thumbx_temp = gamepad_state['thumb_lx']/256
            left_thumbx = round(left_thumbx_temp) + 128  # -127 me 127
            if left_thumbx==256:
                left_thumbx = 255
            right_trigger = gamepad_state['right_trigger']
            left_trigger = gamepad_state['left_trigger']

            #training_data.append([screen, left_thumbx, right_trigger, left_trigger])   #steering, throttle, brake
            training_data.append([screen, left_thumbx])                                 #only steering
            time.sleep(0.05)

```

Data_collecting.py (συνέχεια)

```
            if len(training_data) % 100 == 0:
                print(len(training_data))
                if len(training_data) == 500:
                    np.save(file_name, training_data)
                    print('SAVED')
                    training_data = []
                    starting_value += 1
                    file_name_temp = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder/'+folder_name+'/training_data-{0}.npy'
                    file_name = file_name_temp.format(starting_value)

        pause_key_check = con.buttons
        keyboard_keys = key_check()
        if 'B' in pause_key_check:
            if paused:
                paused = False
                print('Unpaused!')
                time.sleep(1)
            else:
                print('Pausing!')
                paused = True
                time.sleep(1)

        elif 'P' in keyboard_keys:
            if paused:
                paused = False
                print('Unpaused!')
                time.sleep(1)
            else:
                print('Pausing!')
                paused = True
                time.sleep(1)

        if event.is_set():
            print('Data collection has been terminated')
            break

if __name__ == '__main__':
    folder_name = 'test'
    event = Event()
    data_collecting_main(folder_name, event)
```

- **Data_collecting_turn_anticipation.py**

```python
#Turn anticipation
import os, time
import numpy as np
from read_xbox_controller_state import rController
from grabscreen import grab_screen
from getkeys import key_check
from threading import Event

def future_data_collecting_main(folder_name, frame_distance, event):
    #elegxos uparkshs onomatos dataset / dhmiourgia apo to 0 an den uparxei
    starting_value = 0
    if os.path.exists('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'+folder_name):
        print('folder exists, we are proceeding')
    else:
        os.mkdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'+folder_name)
        print('folder created')
    while True:
        data_folder_path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'
        files_name = '/training_data-{0}.npy'
        file_name_temp = data_folder_path+folder_name+files_name
        file_name = file_name_temp.format(starting_value)
        if os.path.isfile(file_name):
            print('File exists, moving along', starting_value)
            starting_value += 1
        else:
            print('File does not exist, starting fresh!', starting_value)
            break
    start_collecting_future_data(file_name, starting_value, folder_name, frame_distance, event)


def start_collecting_future_data(file_name, starting_value, folder_name, frame_distance, event):
    print('Starting test in 3 seconds')
    time.sleep(1)
    print('3')
    time.sleep(1)
    print('2')
    time.sleep(1)
    print('1')

    frame_distance = int(frame_distance)
    con = rController(1)
    training_data = []
    snapshot_list = []
    action_list = []

    last_time = time.time()
    paused = False

    while True:
        if not paused:
            screen = grab_screen(region = (0, 200, 640, 510))           #screenshot
            screen = screen.astype("uint8")
            gamepad_state = con.gamepad                                 #katastash xeiristiriou
            left_thumbx_temp = gamepad_state['thumb_lx']/256            #katastash moxlou xeiristhriou
            left_thumbx = abs(round(left_thumbx_temp))  # 0 - 127
            snapshot_list.append(screen)
            action_list.append(left_thumbx)

            time.sleep(0.05)

```

Data_collecting_turn_anticipation.py (συνέχεια)

```
            if len(action_list) % 100 == 0:
                print(len(action_list))
                #apothikeush teleutaiwn 500 katagrafwn
                if len(action_list) == 500:
                    del snapshot_list[(500-frame_distance):500]
                    del action_list[0:frame_distance]
                    for i in range(0,len(snapshot_list),1):
                        training_data.append([snapshot_list[i], action_list[i]])
                    np.save(file_name, training_data)
                    print('SAVED')
                    training_data = []
                    snapshot_list = []
                    action_list = []
                    starting_value += 1
                    file_name_temp = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'+folder_name+'/training_data-{0}.npy'
                    file_name = file_name_temp.format(starting_value)

        #leitourgia pause
        pause_key_check = con.buttons
        keyboard_keys = key_check()
        if 'B' in pause_key_check:
            if paused:
                paused = False
                print('Unpaused!')
                time.sleep(1)
            else:
                print('Pausing!')
                if len(action_list) > frame_distance:
                    del snapshot_list[(len(snapshot_list)-frame_distance):len(snapshot_list)]
                    del action_list[0:frame_distance]
                    for i in range(0,len(snapshot_list),1):
                        training_data.append([snapshot_list[i], action_list[i]])
                    np.save(file_name, training_data)
                    print('SAVED')
                    training_data = []
                    snapshot_list = []
                    action_list = []
                    starting_value += 1
                    file_name_temp = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'+folder_name+'/training_data-{0}.npy'
                    file_name = file_name_temp.format(starting_value)
                else:
                    training_data = []
                    snapshot_list = []
                    action_list = []
                    print("RESET")
                paused = True
                time.sleep(1)

        elif 'P' in keyboard_keys:
            if paused:
                paused = False
                print('Unpaused!')
                time.sleep(1)
            else:
                print('Pausing!')
                if len(action_list) > frame_distance:
                    del snapshot_list[(len(snapshot_list)-frame_distance):len(snapshot_list)]
                    del action_list[0:frame_distance]
                    for i in range(0,len(snapshot_list),1):
                        training_data.append([snapshot_list[i], action_list[i]])
                    np.save(file_name, training_data)
                    print('SAVED')
                    training_data = []
                    snapshot_list = []
                    action_list = []
                    starting_value += 1
                    file_name_temp = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'+folder_name+'/training_data-{0}.npy'
                    file_name = file_name_temp.format(starting_value)
                else:
                    training_data = []
                    snapshot_list = []
                    action_list = []
                    print("RESET")
                paused = True
                time.sleep(1)

        if event.is_set():
            print('Data collection, for future inputs, has been terminated')
            break

if __name__ == '__main__':
    folder_name = 'test'
    event = Event()
    future_data_collecting_main(folder_name, 20, event)
```

- **Data_collecting_complete.py**

```python
#data collecting real-time and future

import os, time
import numpy as np
from read_xbox_controller_state import rController
from grabscreen import grab_screen
from getkeys import key_check
from threading import Event


def data_collecting_complete_main(folder_name, frame_distance, event):
    #elegxos uparkshs onomatos dataset / dhmiourgia apo to 0 an den uparxei
    starting_value = 0
    if os.path.exists('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_complete/'+folder_name):
        print('folder exists, we are proceeding')
    else:
        os.mkdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_complete/'+folder_name)
        print('folder created')
    while True:
        data_folder_path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_complete/'
        files_name = '/training_data-{0}.npy'
        file_name_temp = data_folder_path+folder_name+files_name
        file_name = file_name_temp.format(starting_value)
        if os.path.isfile(file_name):
            print('File exists, moving along', starting_value)
            starting_value += 1
        else:
            print('File does not exist, starting fresh!', starting_value)
            break
    start_collecting_complete_data(file_name, starting_value, folder_name, frame_distance, event)


def start_collecting_complete_data(file_name, starting_value, folder_name, frame_distance, event):
    print('Starting test in 3 seconds')
    time.sleep(1)
    print('3')
    time.sleep(1)
    print('2')
    time.sleep(1)
    print('1')

    frame_distance = int(frame_distance)
    con = rController(1)
    training_data = []
    snapshot_list = []
    action_list = []

    last_time = time.time()
    paused = False

    while True:
        if not paused:

            screen = grab_screen(region = (0, 200, 640, 510))  #screenshot
            screen = screen.astype("uint8")
            gamepad_state = con.gamepad     #katastash xeiristhriou
            left_thumbx_temp = gamepad_state['thumb_lx']/256    #katastash moxlou xeiristhriou
            left_thumbx = abs(round(left_thumbx_temp))  # 0 - 127,
                                                        #eksagwgh mono plhroforias kampulothtas xwris plhroforia kateu8unshs
            snapshot_list.append(screen)
            action_list.append(left_thumbx)

            time.sleep(0.05)    #kathorismos frame rate

```

Data_collecting_complete.py (συνέχεια)

```
            if len(action_list) % 100 == 0:
                print(len(action_list))
                #apothhkeush 500 teleutaiwn katagrafwn
                if len(action_list) == 500:
                    for i in range(0,len(snapshot_list)-frame_distance,1):
                        training_data.append([snapshot_list[i], action_list[i], action_list[i+frame_distance]])
                    np.save(file_name, training_data)
                    print('SAVED')
                    training_data = []
                    snapshot_list = []
                    action_list = []
                    starting_value += 1
                    file_name_temp = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_complete/'+folder_name+'/training_data-{0}.npy'
                    file_name = file_name_temp.format(starting_value)

            #leitourgia pause-apothhkeush
            pause_key_check = con.buttons
            keyboard_keys = key_check()
            if 'B' in pause_key_check:
                if paused:
                    paused = False
                    print('Unpaused!')
                    time.sleep(1)
                else:
                    print('Pausing!')
                    if len(action_list) > frame_distance:
                        for i in range(0,len(snapshot_list)-frame_distance,1):
                            training_data.append([snapshot_list[i], action_list[i], action_list[i+frame_distance]])
                        np.save(file_name, training_data)
                        print('SAVED')
                        training_data = []
                        snapshot_list = []
                        action_list = []
                        starting_value += 1
                        file_name_temp = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_complete/'+folder_name+'/training_data-{0}.npy'
                        file_name = file_name_temp.format(starting_value)
                    else:
                        training_data = []
                        snapshot_list = []
                        action_list = []
                        print("RESET")
                    paused = True
                    time.sleep(1)
```

```

            elif 'P' in keyboard_keys:
                if paused:
                    paused = False
                    print('Unpaused!')
                    time.sleep(1)
                else:
                    print('Pausing!')
                    if len(action_list) > frame_distance:
                        for i in range(0,len(snapshot_list)-frame_distance,1):
                            training_data.append([snapshot_list[i], action_list[i], action_list[i+frame_distance]])
                        np.save(file_name, training_data)
                        print('SAVED')
                        training_data = []
                        snapshot_list = []
                        action_list = []
                        starting_value += 1
                        file_name_temp = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_complete/'+folder_name+'/training_data-{0}.npy'
                        file_name = file_name_temp.format(starting_value)
                    else:
                        training_data = []
                        snapshot_list = []
                        action_list = []
                        print("RESET")
                    paused = True
                    time.sleep(1)

            if event.is_set():
                print('Data collection, for future inputs, has been terminated')
                break

    if __name__ == '__main__':
        folder_name = 'test'
        event = Event()
        data_collecting_complete_main(folder_name, 20, event)
```

- **Digit_capturing.py**

```python
#digit capturing

import os, time, cv2
import numpy as np
from read_xbox_controller_state import rController
from grabscreen import grab_screen

#elegxos uparkshs onomatos dataset / dhmiourgia apo to 0 an den uparxei
starting_value = 0
while True:
    file_name = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder/incoplete_digits/digit_dataset-{0}.npy'.format(starting_value)
    if os.path.isfile(file_name):
        print('File exists, moving along', starting_value)
        starting_value += 1
    else:
        print('File does not exist, starting fresh!', starting_value)
        break

def main(file_name, starting_value):
    print('Starting test')
    rgb_weights = [0.2989, 0.5870, 0.1140]
    con = rController(1)
    training_data = []
    finished = False

    while True:
        if not finished:
            capture_key_check = con.buttons
            if 'B' in capture_key_check:
                screen = grab_screen(region = (16, 240, 33, 262))
                #screen = grab_screen(region = (0, 240, 17, 262))
                screen_grayscale_float = np.dot(screen[...,:3], rgb_weights)
                screen_grayscale = screen_grayscale_float.astype('uint8')
                print(screen.shape)
                cv2.imshow('window',screen_grayscale)
                if cv2.waitKey(25) & 0xFF == ord('q'):
                    cv2.destroyAllWindows()
                    break
                digit_input = input("Enter the digit you see: ")
                digit = int(digit_input)
                print(digit)
                training_data.append([screen_grayscale, digit])
            finish_key_check = con.buttons
            if 'Y' in finish_key_check:
                np.save(file_name, training_data)
                training_data = []
                starting_value += 1
                file_name = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder/incoplete_digits/digit_dataset-{0}.npy'.format(starting_value)
                time.sleep(0.2)

main(file_name, 0)

```

- **Train_model-steering.py**

```python
import os
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Activation, Dense, Flatten, GlobalAveragePooling2D, BatchNormalization, Dropout
from tensorflow.keras.optimizers import  Adam
from tensorflow.keras.applications import EfficientNetB0, MobileNetV2, DenseNet121
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.callbacks import TensorBoard
import time

width = 640
height = 310
epochs = 10

def create_model():
    model = Sequential()
    EfNet = EfficientNetB0(include_top=False, weights="imagenet", input_shape=(height, width, 3))
    #EfNet = EfficientNetB0(include_top=False, weights=None, input_shape=(height, width, 3))
    #plot_model(EfNet, to_file="C:/Users/jimpa/Desktop/steering_model_EfNetB0.png",
                #show_shapes=True)
    #MobNet = MobileNetV2(include_top=False, weights="imagenet", input_shape=(height, width, 3))
    #DNet121 = DenseNet121(include_top=False, weights="imagenet", input_shape=(height, width, 3))
   # EfNet.summary()
    model.add(EfNet)
    #model.add(MobNet)
    #model.add(DNet121)
    model.add(GlobalAveragePooling2D(name="avg_pool"))
    #model.add(BatchNormalization())
    #model.add(Dropout(0.5))
    #model.add(Flatten())
    model.add(Dense(64, activation='relu', name='dense1'))
    #model.add(BatchNormalization())
    #model.add(Dropout(rate=0.2))
    model.add(Dense(64, activation='relu', name='dense2'))
    #model.add(BatchNormalization())
    #model.add(Dropout(rate=0.2))
    #model.add(Dense(17, activation='softmax'))
    model.add(Dense(1, activation="linear"))     #regressio

    model.summary()

    optimizer = Adam()
    model.compile(optimizer=optimizer,
                  #loss=[keras.losses.CategoricalCrossentropy(from_logits=False)],           #onehot outputs
                  #loss=[keras.losses.SparseCategoricalCrossentropy(from_logits=True)],   #integer outputs regression
                  loss=[keras.losses.MeanSquaredError()],          #paizei na thelei auto gia regression
                  #loss=[keras.losses.SparseCategoricalCrossentropy(from_logits=False)],   #integer outputs με Softmax
                  metrics = ["mse"]
                  #metrics = ["loss"]
                 )
    #plot_model(model, to_file="C:/Users/jimpa/Desktop/steering_model_EfNetB0.png",
                #show_shapes=True)
    return model

def concat_files(number_of_files, file_names):
    single_array_dataset = np.empty((0,4),dtype=np.uint8)
    for j in range(0,number_of_files,1):
        dataset = np.load(file=file_names.format(j), allow_pickle=True)
        single_array_dataset = np.concatenate((single_array_dataset, dataset), axis=0)
    print(single_array_dataset.shape)
    return single_array_dataset

```

Train_model-steering.py (συνέχεια)

```python
def load_dataset(number_of_files, file_names):
    dataset = concat_files(number_of_files, file_names)
    np.random.shuffle(dataset)
    dataset_length = len(dataset)
    dataset_split_point = int(dataset_length*0.2)
    print('split point',dataset_split_point)
    print(dataset_split_point)
    train = dataset[:-dataset_split_point]
    val = dataset[-dataset_split_point:]

    train_x = np.array([i[0] for i in train])
    train_x = train_x.reshape((len(train_x),310,640,3))       #training images
    train_x = train_x.astype("uint8")
    print("Memory size of numpy array in bytes:", train_x.size * train_x.itemsize)
    print(train_x.nbytes)
    print(train_x.shape)
    train_labels = [i[1] for i in train]               #training image labels   only steering
    train_labels = (np.array(train_labels)).astype("uint8")     # 0 - 255
    valid_x = np.array([i[0] for i in val])
    valid_x = valid_x.reshape((len(valid_x),310,640,3))   #validation images
    valid_x = valid_x.astype("uint8")
    valid_labels = [i[1] for i in val]             #validation labels   only steering
    valid_labels = (np.array(valid_labels)).astype("uint8")     # 0 - 255

    #onehot_train_labels = to_categorical(train_labels, num_classes=11, dtype='uint8')
    #onehot_valid_labels = to_categorical(valid_labels, num_classes=11, dtype='uint8')
    #train_labels_int = [np.argmax(y, axis=None, out=None) for y in train_labels]
    #valid_labels_int = [np.argmax(y, axis=None, out=None) for y in valid_labels]

    batch_size = 8

    train_datagen = ImageDataGenerator(
        #rescale = 1/.255
        #width_shift_range=0.05
        #rotation_range=30,
        zoom_range=0.2,
        vertical_flip=True,
        dtype = "uint8"
        )
    val_datagen = ImageDataGenerator(
        #rescale = 1/.255
        #width_shift_range=0.05
        #vertical_flip=True,
        dtype = "uint8"
        )
    x_train = train_datagen.flow(
        train_x,
        train_labels,
        batch_size=batch_size,
        shuffle=True
        )
    x_val = val_datagen.flow(
        valid_x,
        valid_labels,
        batch_size=1,
        shuffle=False
        )

    return x_train, x_val

```

- **Train_model_turn_anticipation.py**

```python
#train turn anticipation

import os
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Activation, Dense, Flatten, GlobalAveragePooling2D, BatchNormalization, Dropout
from tensorflow.keras.optimizers import  Adam
from tensorflow.keras.applications import EfficientNetB0, MobileNetV2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import TensorBoard
import time

width = 640
height = 310
epochs = 10

def create_model():
    model = Sequential()
    #EfNet = EfficientNetB0(include_top=False, weights="imagenet", input_shape=(height, width, 3))
    MobNet = MobileNetV2(include_top=False, weights="imagenet", input_shape=(height, width, 3))
    #EfNet.summary()
    #model.add(EfNet)
    model.add(MobNet)
    model.add(GlobalAveragePooling2D(name="avg_pool"))
    #model.add(BatchNormalization())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(64, activation='relu'))
    #model.add(Dropout(0.5))
    #model.add(Flatten())
    #model.add(Dense(17, activation='softmax'))
    model.add(Dense(1, activation="linear"))     #regression

    optimizer = Adam()
    model.compile(optimizer=optimizer,
                  #loss=[keras.losses.CategoricalCrossentropy(from_logits=False)],           #onehot outputs
                  #loss=[keras.losses.SparseCategoricalCrossentropy(from_logits=True)],   #integer outputs regression
                  loss=[keras.losses.MeanSquaredError()],          #paizei na thelei auto gia regression
                  #loss=[keras.losses.SparseCategoricalCrossentropy(from_logits=False)],   #integer outputs με Softmax
                  metrics = ["mse"]
                  #metrics = ["loss"]
                 )
    return model

def concat_files(number_of_files, file_names):
    single_array_dataset = np.empty((0,2),dtype=np.uint8)
    for j in range(0,number_of_files,1):
        dataset = np.load(file=file_names.format(j), allow_pickle=True)
        single_array_dataset = np.concatenate((single_array_dataset, dataset), axis=0)
    print(single_array_dataset.shape)
    return single_array_dataset

```

Train_model_turn_anticipation.py (συνέχεια)

```
def load_dataset(number_of_files, file_names):
    dataset = concat_files(number_of_files, file_names)
    np.random.shuffle(dataset)
    dataset_length = len(dataset)
    dataset_split_point = int(dataset_length*0.2)
    print('split point',dataset_split_point)
    print(dataset_split_point)
    train = dataset[:-dataset_split_point]
    val = dataset[-dataset_split_point:]

    train_x = np.array([i[0] for i in train])
    train_x = train_x.reshape((len(train_x),310,640,3))       #training images
    train_x = train_x.astype("uint8")
    print("Memory size of numpy array in bytes:", train_x.size * train_x.itemsize)
    print(train_x.nbytes)
    print(train_x.shape)
    #print(type(train_x[0][0][0][0]))
    train_labels = [i[1] for i in train]               #training image labels   only steering
    train_labels = (np.array(train_labels)).astype("uint8")     # 0 - 255

    valid_x = np.array([i[0] for i in val])
    valid_x = valid_x.reshape((len(valid_x),310,640,3))   #validation images
    valid_x = valid_x.astype("uint8")
    valid_labels = [i[1] for i in val]             #validation labels   only steering
    valid_labels = (np.array(valid_labels)).astype("uint8")     # 0 - 255

    #onehot_train_labels = to_categorical(train_labels, num_classes=11, dtype='uint8')
    #onehot_valid_labels = to_categorical(valid_labels, num_classes=11, dtype='uint8')
    #train_labels_int = [np.argmax(y, axis=None, out=None) for y in train_labels]
    #valid_labels_int = [np.argmax(y, axis=None, out=None) for y in valid_labels]

    batch_size = 8

    train_datagen = ImageDataGenerator(
        #rescale = 1/.255
        #width_shift_range=0.05
        zoom_range=0.2,
        vertical_flip=True,
        dtype = "uint8"
        )
    val_datagen = ImageDataGenerator(
        #rescale = 1/.255
        #width_shift_range=0.05
        dtype = "uint8"
        )
    x_train = train_datagen.flow(
        train_x,
        train_labels,
        batch_size=batch_size,
        shuffle=True
        )
    x_val = val_datagen.flow(
        valid_x,
        valid_labels,
        batch_size=1,
        shuffle=False
        )
    return x_train, x_val

```

Train_model_turn_anticipation.py (συνέχεια)

```python
def callback(model_name):
    checkpoint_path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/saved_models_turn_anticipation/'+model_name
    checkpoint_dir = os.path.dirname(checkpoint_path)
    cp_callback = tf.keras.callbacks.ModelCheckpoint(
        filepath = checkpoint_path,
        save_weights_only = False,
        monitor='val_loss',
        save_best_only=True,
        verbose = 2
    )
    return cp_callback

#FITTING THE DATA INTO THE MODEL
def train_model_turn_anticipation_main(folder_name, model_name):
    epochs = 60
    model = create_model()
    print('model created')
    cp_callback = callback(model_name)
    file_names = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'+folder_name+'/training_data-{0}.npy'
    number_of_files_plus1 = len(os.listdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation/'+folder_name))
    train_data, val_data = load_dataset(number_of_files_plus1, file_names)
    #TensorBoard setup
    name = model_name.format(int(time.time()))
    tensorboard = TensorBoard(log_dir='logs/{}'.format(name))

    history = model.fit(
        x = train_data,
        validation_data = val_data,
        epochs = epochs,
        callbacks = [cp_callback, tensorboard],
        verbose = 2,
        )

if __name__ == '__main__':
    folder_name = 'phase4_turn_anticipation_1.0'
    model_name = 'MobNetV2+2x64+VertFlip+zoom02'
    train_model_turn_anticipation_main(folder_name, model_name)

```

- **Test_model.py**

```python
#algorithmos autonomhs odhghshs
from getkeys import key_check
import numpy as np
from grabscreen import grab_screen
from vjoy import mytest_phase4
import tensorflow as tf
from tensorflow import keras

#provlepsieis
def test_model_main(steering_model, turn_anticipation_model, check_frequency, min_speed, max_speed):
    WIDTH = 640
    HEIGHT = 310

    #loading the model
    checkpoint_path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/saved_models/'+steering_model
    model = keras.models.load_model(checkpoint_path)
    model.summary()

    #loading speed model
    checkpoint_path_speed = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/saved_models_speed_recogn/speed_recognition_with_incompletes'
    model_speed = keras.models.load_model(checkpoint_path_speed)
    model_speed.summary()

    #loading turn anticipation model
    checkpoint_path_future = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/saved_models_turn_anticipation/'+turn_anticipation_model
    model_future = keras.models.load_model(checkpoint_path_future)
    model_future.summary()

    print(check_frequency)
    print("Starting")
    paused = False
    #arxikopoihsh timwn
    desired_speed = 20
    steering_angle = 16392
    future_steering_angle = 16392
    throttle_percentage = 50
    brake_percentage = 0
    new_brake_percentage = 0
    counter = 0
    break_counter = 0
    check_frequency = 2
    speed_factor = (max_speed - min_speed)/128

    while(True):
        if not paused:
            #provlepsi steering angle
            screen = grab_screen(region = (0, 200, 640, 510))
            screen_reshaped = screen.reshape(1,HEIGHT,WIDTH, 3)
            prediction = model.predict(screen_reshaped)
            #steering_angle_pred = round((prediction[0][0]))
            steering_angle = round((prediction[0][0]))*128
            #diavasma eikonwn psifion
            number1 = grab_screen(region = (0, 240, 16, 262))
            number2 = grab_screen(region = (16, 240, 32, 262))
            #rgb to grayscale
            rgb_weights = [0.2989, 0.5870, 0.1140]
            number1_grayscale_float = np.dot(number1[...,:3], rgb_weights)
            number2_grayscale_float = np.dot(number2[...,:3], rgb_weights)
            #metatroph se integers
            number1_grayscale = number1_grayscale_float.astype('uint8')
            number2_grayscale = number2_grayscale_float.astype('uint8')

            #elegxos taxuthtas kai mellontikhs kampulothtas
            if counter==check_frequency:
                #provlepsh pshfiwn endeikshs taxuthtas
                number1_prediction = model_speed.predict(number1_grayscale.reshape(1,22,16,1))
                number2_prediction = model_speed.predict(number2_grayscale.reshape(1,22,16,1))
```

Test_model.py (συνέχεια)

```
68
69                  if int(np.argmax(number2_prediction)) == 10:
70                      if throttle_percentage <= 99:
71                          throttle_percentage = throttle_percentage + 1
72                      brake_percentage = 0
73                  else:
74                      string1 = str(np.argmax(number1_prediction))
75                      string2 = str(np.argmax(number2_prediction))
76                      final_string = string1+string2
77                      final_number = int(final_string)
78                      future_steering_angle = model_future.predict(screen_reshaped)
79                      print(future_steering_angle)
80                      desired_speed = max_speed - speed_factor*future_steering_angle[0][0]
81                      desired_speed = round(desired_speed)
82
83                      #upologismos throttle percentage
84                      #epivradinsi
85                      if final_number > desired_speed:
86                          new_throttle_percentage = round(throttle_percentage - 0.6*(final_number-desired_speed))
87                          if new_throttle_percentage >= 1:
88                              throttle_percentage = new_throttle_percentage
89                          else:
90                              throttle_percentage = 1
91
92                          break_counter = 2*(final_number - desired_speed)
93                          if (final_number - desired_speed) < 21:     #auto htan 50
94                              new_brake_percentage = 5*(final_number - desired_speed)
95                          else:
96                              new_brake_percentage = 100      #kai auto htan 50
97                                                              #twra vgazei pio polu nohma alla den to exw testarei akoma
98                      #epitaxunsh
99                      elif final_number < desired_speed:
100                         brake_counter = 0
101                         new_brake_percentage = 0
102                         brake_percentage = 0
103                         new_throttle_percentage = round(throttle_percentage + 0.3*(desired_speed-final_number))
104                         if new_throttle_percentage <= 99:
105                             throttle_percentage = new_throttle_percentage
106                         else:
107                             throttle_percentage = 100
108
109                 counter = 0
110             else:
111                 counter = counter + 1
112             #elegxos diarkias epivradunshs
113             if break_counter > 0:
114                 brake_percentage = new_brake_percentage
115                 break_counter = break_counter - 1
116             elif break_counter == 0:
117                 brake_percentage = 0
118
119             #apostolh entolwn
120             steering_input = steering_angle
121             throttle_input = throttle_percentage
122             brake_input = brake_percentage
123             mytest_phase4(throttle_input, brake_input, steering_input)
124
125         #pause / stop
126         keyboard_keys = key_check()
127         if 'P' in keyboard_keys:
128             if paused == False:
129                 paused = True
130                 print('pausing')
131             else:
132                 paused = False
133                 print('unpausing')
134         elif 'S' in keyboard_keys:
135             print('stopping')
136             break
137
138
139 if __name__ == '__main__':
140     test_model_main('phase4_EfficientNetB0+Dense2x64_BatchNorm_225-40', 'phase4_turn_anticipation_1.0_40_epochs', 5, 10, 50)
141
```

- **Interface_v4.py**

```python
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'Interface_Design_1.0.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets
#added imports to the auto-generated pyqt desinger code
from PyQt5.QtWidgets import QMessageBox
import re
import subprocess
import os
from os import listdir
from os.path import isfile, join
from data_collecting import data_collecting_main
from data_collecting_turn_anticipation import future_data_collecting_main
from data_collecting_complete import data_collecting_complete_main
from train_model_steering import train_model_main
from train_model_turn_anticipation import train_model_turn_anticipation_main
from test_model import test_model_main
from threading import Thread
from threading import Event
import sys
from getkeys import key_check
import shutil
import webbrowser
from pynput.keyboard import Key, Controller
#added imports ending

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(498, 550)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.tab_widget = QtWidgets.QTabWidget(self.centralwidget)
        self.tab_widget.setGeometry(QtCore.QRect(0, 0, 491, 511))
        self.tab_widget.setObjectName("tab_widget")
        self.tab_0 = QtWidgets.QWidget()
        self.tab_0.setObjectName("tab_0")
        #self.pushButton_ds4_launch = QtWidgets.QPushButton(self.tab_0)
        #self.pushButton_ds4_launch.setGeometry(QtCore.QRect(180, 90, 131, 23))
        #self.pushButton_ds4_launch.setObjectName("pushButton_ds4_launch")
        self.pushButton_x360_launch = QtWidgets.QPushButton(self.tab_0)
        self.pushButton_x360_launch.setGeometry(QtCore.QRect(180, 150, 131, 23))
        self.pushButton_x360_launch.setObjectName("pushButton_x360_launch")
        self.pushButton_vjoy_controller_launch = QtWidgets.QPushButton(self.tab_0)
        self.pushButton_vjoy_controller_launch.setGeometry(QtCore.QRect(180, 210, 131, 23))
        self.pushButton_vjoy_controller_launch.setObjectName("pushButton_vjoy_controller_launch")
        self.pushButton_ac_launch = QtWidgets.QPushButton(self.tab_0)
        self.pushButton_ac_launch.setGeometry(QtCore.QRect(180, 270, 131, 23))
        self.pushButton_ac_launch.setToolTip("")
        self.pushButton_ac_launch.setObjectName("pushButton_ac_launch")
        self.file_list = QtWidgets.QComboBox(self.tab_0)
        self.file_list.setGeometry(QtCore.QRect(80, 360, 111, 22))
        self.file_list.setObjectName("file_list")
        self.file_list.addItem("")
        #added stuff
        view = self.file_list.view()
        view.setFixedWidth(300)
        #end of added stuff
```

Interface_v4.py (συνέχεια)

```
        self.pushButton_open_file_main = QtWidgets.QPushButton(self.tab_0)
        self.pushButton_open_file_main.setGeometry(QtCore.QRect(230, 360, 75, 23))
        self.pushButton_open_file_main.setObjectName("pushButton_open_file_main")
        self.pushButton_reset_selected_file = QtWidgets.QPushButton(self.tab_0)
        self.pushButton_reset_selected_file.setGeometry(QtCore.QRect(340, 360, 75, 23))
        self.pushButton_reset_selected_file.setObjectName("pushButton_reset_selected_file")
        self.pushButton_reset_all_files = QtWidgets.QPushButton(self.tab_0)
        self.pushButton_reset_all_files.setGeometry(QtCore.QRect(340, 400, 75, 23))
        self.pushButton_reset_all_files.setObjectName("pushButton_reset_all_files")
        self.tab_widget.addTab(self.tab_0, "")
        self.tab_1 = QtWidgets.QWidget()
        self.tab_1.setObjectName("tab_1")
        self.future_steering_inputs = QtWidgets.QGroupBox(self.tab_1)
        self.future_steering_inputs.setGeometry(QtCore.QRect(10, 210, 461, 251))
        self.future_steering_inputs.setObjectName("future_steering_inputs")
        self.label_future_steering_folder_name = QtWidgets.QLabel(self.future_steering_inputs)
        self.label_future_steering_folder_name.setGeometry(QtCore.QRect(20, 60, 71, 16))
        self.label_future_steering_folder_name.setObjectName("label_future_steering_folder_name")
        self.lineEdit_future_steering_folder_name = QtWidgets.QLineEdit(self.future_steering_inputs)
        self.lineEdit_future_steering_folder_name.setGeometry(QtCore.QRect(130, 60, 171, 20))
        self.lineEdit_future_steering_folder_name.setObjectName("lineEdit_future_steering_folder_name")
        self.label_frame_distance = QtWidgets.QLabel(self.future_steering_inputs)
        self.label_frame_distance.setGeometry(QtCore.QRect(20, 110, 81, 16))
        self.label_frame_distance.setObjectName("label_frame_distance")
        self.lineEdit_frame_distance_data_collect = QtWidgets.QLineEdit(self.future_steering_inputs)
        self.lineEdit_frame_distance_data_collect.setGeometry(QtCore.QRect(130, 110, 171, 20))
        self.lineEdit_frame_distance_data_collect.setObjectName("lineEdit_frame_distance_data_collect")
        self.pushButton_start_future_steering_inputs = QtWidgets.QPushButton(self.future_steering_inputs)
        self.pushButton_start_future_steering_inputs.setGeometry(QtCore.QRect(20, 200, 75, 23))
        self.pushButton_start_future_steering_inputs.setObjectName("pushButton_start_future_steering_inputs")
        self.pushButton_open_file_future_steering_inputs = QtWidgets.QPushButton(self.future_steering_inputs)
        self.pushButton_open_file_future_steering_inputs.setGeometry(QtCore.QRect(120, 200, 75, 23))
        self.pushButton_open_file_future_steering_inputs.setObjectName("pushButton_open_file_future_steering_inputs")
        self.checkBox_both_datasets_at_once = QtWidgets.QCheckBox(self.future_steering_inputs)
        self.checkBox_both_datasets_at_once.setGeometry(QtCore.QRect(130, 160, 171, 17))
        self.checkBox_both_datasets_at_once.setObjectName("checkBox_both_datasets_at_once")
        self.label_pause_future_steering_inputs = QtWidgets.QLabel(self.future_steering_inputs)
        self.label_pause_future_steering_inputs.setGeometry(QtCore.QRect(350, 170, 91, 21))
        self.label_pause_future_steering_inputs.setObjectName("label_pause_future_steering_inputs")
        self.label_stop_future_steering_inputs = QtWidgets.QLabel(self.future_steering_inputs)
        self.label_stop_future_steering_inputs.setGeometry(QtCore.QRect(350, 200, 91, 20))
        self.label_stop_future_steering_inputs.setObjectName("label_stop_future_steering_inputs")
        self.pushButton_reset_file_data_collect_future_inputs = QtWidgets.QPushButton(self.future_steering_inputs)
        self.pushButton_reset_file_data_collect_future_inputs.setGeometry(QtCore.QRect(220, 200, 75, 23))
        self.pushButton_reset_file_data_collect_future_inputs.setObjectName("pushButton_reset_file_data_collect_future_inputs")
        self.realtime_steering_inputs = QtWidgets.QGroupBox(self.tab_1)
        self.realtime_steering_inputs.setGeometry(QtCore.QRect(10, 20, 461, 171))
        self.realtime_steering_inputs.setObjectName("realtime_steering_inputs")
        self.label_realtime_steering_folder_name = QtWidgets.QLabel(self.realtime_steering_inputs)
        self.label_realtime_steering_folder_name.setGeometry(QtCore.QRect(20, 50, 71, 16))
        self.label_realtime_steering_folder_name.setObjectName("label_realtime_steering_folder_name")
        self.lineEdit_real_time_steering_dataset_folder_name = QtWidgets.QLineEdit(self.realtime_steering_inputs)
        self.lineEdit_real_time_steering_dataset_folder_name.setGeometry(QtCore.QRect(130, 50, 171, 20))
        self.lineEdit_real_time_steering_dataset_folder_name.setObjectName("lineEdit_real_time_steering_dataset_folder_name")
        self.pushButton_start_data_collect_realtime = QtWidgets.QPushButton(self.realtime_steering_inputs)
        self.pushButton_start_data_collect_realtime.setGeometry(QtCore.QRect(20, 120, 75, 23))
        self.pushButton_start_data_collect_realtime.setObjectName("pushButton_start_data_collect_realtime")
        self.pushButton_open_file_data_collect_realtime = QtWidgets.QPushButton(self.realtime_steering_inputs)
        self.pushButton_open_file_data_collect_realtime.setGeometry(QtCore.QRect(120, 120, 75, 23))
        self.pushButton_open_file_data_collect_realtime.setAcceptDrops(False)
        self.pushButton_open_file_data_collect_realtime.setObjectName("pushButton_open_file_data_collect_realtime")
        self.label_pause_real_time_steering_inputs = QtWidgets.QLabel(self.realtime_steering_inputs)
        self.label_pause_real_time_steering_inputs.setGeometry(QtCore.QRect(350, 89, 91, 21))
        self.label_pause_real_time_steering_inputs.setObjectName("label_pause_real_time_steering_inputs")
        self.label_stop_real_time_steering_inputs = QtWidgets.QLabel(self.realtime_steering_inputs)
```

Interface.py (συνέχεια)

```
        self.label_stop_real_time_steering_inputs.setGeometry(QtCore.QRect(350, 120, 91, 20))
        self.label_stop_real_time_steering_inputs.setObjectName("label_stop_real_time_steering_inputs")
        self.pushButton_reset_file_data_collect_realtime = QtWidgets.QPushButton(self.realtime_steering_inputs)
        self.pushButton_reset_file_data_collect_realtime.setGeometry(QtCore.QRect(220, 120, 75, 23))
        self.pushButton_reset_file_data_collect_realtime.setObjectName("pushButton_reset_file_data_collect_realtime")
        self.tab_widget.addTab(self.tab_1, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.pushButton_open_tensorboard = QtWidgets.QPushButton(self.tab_2)
        self.pushButton_open_tensorboard.setGeometry(QtCore.QRect(170, 420, 131, 23))
        self.pushButton_open_tensorboard.setObjectName("pushButton_open_tensorboard")
        self.groupBox_3 = QtWidgets.QGroupBox(self.tab_2)
        self.groupBox_3.setGeometry(QtCore.QRect(10, 20, 451, 181))
        self.groupBox_3.setObjectName("groupBox_3")
        self.comboBox_steering_dataset = QtWidgets.QComboBox(self.groupBox_3)
        self.comboBox_steering_dataset.setGeometry(QtCore.QRect(60, 50, 121, 20))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.comboBox_steering_dataset.sizePolicy().hasHeightForWidth())
        self.comboBox_steering_dataset.setSizePolicy(sizePolicy)
        self.comboBox_steering_dataset.setSizeAdjustPolicy(QtWidgets.QComboBox.AdjustToContentsOnFirstShow)
        #added stuff
        view = self.comboBox_steering_dataset.view()
        view.setFixedWidth(300)
        #end of added stuff
        self.comboBox_steering_dataset.setObjectName("comboBox_steering_dataset")
        self.comboBox_steering_dataset.addItem("")
        self.lineEdit_steering_model_file_name = QtWidgets.QLineEdit(self.groupBox_3)
        self.lineEdit_steering_model_file_name.setGeometry(QtCore.QRect(270, 50, 121, 20))
        self.lineEdit_steering_model_file_name.setObjectName("lineEdit_steering_model_file_name")
        self.pushButton_start_steering_model_train = QtWidgets.QPushButton(self.groupBox_3)
        self.pushButton_start_steering_model_train.setGeometry(QtCore.QRect(50, 140, 75, 23))
        self.pushButton_start_steering_model_train.setObjectName("pushButton_start_steering_model_train")
        self.pushButton_stop_steering_model_train = QtWidgets.QPushButton(self.groupBox_3)
        self.pushButton_stop_steering_model_train.setGeometry(QtCore.QRect(160, 140, 75, 23))
        self.pushButton_stop_steering_model_train.setObjectName("pushButton_stop_steering_model_train")
        self.pushButton_open_file_steering_model = QtWidgets.QPushButton(self.groupBox_3)
        self.pushButton_open_file_steering_model.setGeometry(QtCore.QRect(250, 90, 75, 23))
        self.pushButton_open_file_steering_model.setObjectName("pushButton_open_file_steering_model")
        self.label_choose_modify_model = QtWidgets.QLabel(self.groupBox_3)
        self.label_choose_modify_model.setGeometry(QtCore.QRect(60, 90, 161, 16))
        self.label_choose_modify_model.setObjectName("label_choose_modify_model")
        self.pushButton_reset_model_at_default = QtWidgets.QPushButton(self.groupBox_3)
        self.pushButton_reset_model_at_default.setGeometry(QtCore.QRect(350, 90, 75, 23))
        self.pushButton_reset_model_at_default.setObjectName("pushButton_reset_model_at_default")
        self.groupBox_4 = QtWidgets.QGroupBox(self.tab_2)
        self.groupBox_4.setGeometry(QtCore.QRect(10, 220, 451, 181))
        self.groupBox_4.setObjectName("groupBox_4")
        self.comboBox_turn_anticipation_dataset = QtWidgets.QComboBox(self.groupBox_4)
        self.comboBox_turn_anticipation_dataset.setGeometry(QtCore.QRect(60, 50, 131, 22))
        self.comboBox_turn_anticipation_dataset.setObjectName("comboBox_turn_anticipation_dataset")
        #added stuff
        view = self.comboBox_turn_anticipation_dataset.view()
        view.setFixedWidth(300)
        #end of added stuff
        self.comboBox_turn_anticipation_dataset.addItem("")
        self.lineEdit_turn_anticipation_model_file_name = QtWidgets.QLineEdit(self.groupBox_4)
        self.lineEdit_turn_anticipation_model_file_name.setGeometry(QtCore.QRect(280, 50, 121, 20))
        self.lineEdit_turn_anticipation_model_file_name.setObjectName("lineEdit_turn_anticipation_model_file_name")
        self.pushButton_start_turn_anticipation_model_train = QtWidgets.QPushButton(self.groupBox_4)
        self.pushButton_start_turn_anticipation_model_train.setGeometry(QtCore.QRect(50, 140, 75, 23))
        self.pushButton_start_turn_anticipation_model_train.setObjectName("pushButton_start_turn_anticipation_model_train")
        self.pushButton_stop_turn_anticipation_model_train = QtWidgets.QPushButton(self.groupBox_4)
        self.pushButton_stop_turn_anticipation_model_train.setGeometry(QtCore.QRect(160, 140, 75, 23))
        self.pushButton_stop_turn_anticipation_model_train.setObjectName("pushButton_stop_turn_anticipation_model_train")
```

Interface.py (συνέχεια)

```
        self.pushButton_open_file_turn_anticipation_model = QtWidgets.QPushButton(self.groupBox_4)
        self.pushButton_open_file_turn_anticipation_model.setGeometry(QtCore.QRect(250, 90, 75, 23))
        self.pushButton_open_file_turn_anticipation_model.setObjectName("pushButton_open_file_turn_anticipation_model")
        self.pushButton_reset_turn_anticipation_model_at_default = QtWidgets.QPushButton(self.groupBox_4)
        self.pushButton_reset_turn_anticipation_model_at_default.setGeometry(QtCore.QRect(350, 90, 75, 23))
        self.pushButton_reset_turn_anticipation_model_at_default.setObjectName("pushButton_reset_turn_anticipation_model_at_default")
        self.label_choose_modify_model_2 = QtWidgets.QLabel(self.groupBox_4)
        self.label_choose_modify_model_2.setGeometry(QtCore.QRect(60, 90, 161, 16))
        self.label_choose_modify_model_2.setObjectName("label_choose_modify_model_2")
        self.tab_widget.addTab(self.tab_2, "")
        self.tab_3 = QtWidgets.QWidget()
        self.tab_3.setObjectName("tab_3")
        self.comboBox_steering_model = QtWidgets.QComboBox(self.tab_3)
        self.comboBox_steering_model.setGeometry(QtCore.QRect(40, 60, 191, 22))
        self.comboBox_steering_model.setObjectName("comboBox_steering_model")
        self.comboBox_steering_model.addItem("")
        #added stuff
        view = self.comboBox_steering_model.view()
        view.setFixedWidth(450)
        #end of added stuff
        self.pushButton_run_test = QtWidgets.QPushButton(self.tab_3)
        self.pushButton_run_test.setGeometry(QtCore.QRect(30, 340, 75, 23))
        self.pushButton_run_test.setObjectName("pushButton_run_test")
        self.pushButton_stop_test = QtWidgets.QPushButton(self.tab_3)
        self.pushButton_stop_test.setGeometry(QtCore.QRect(250, 340, 75, 23))
        self.pushButton_stop_test.setObjectName("pushButton_stop_test")
        self.pushButton_pause_test = QtWidgets.QPushButton(self.tab_3)
        self.pushButton_pause_test.setGeometry(QtCore.QRect(140, 340, 75, 23))
        self.pushButton_pause_test.setObjectName("pushButton_pause_test")
        self.comboBox_turn_anticipation_model = QtWidgets.QComboBox(self.tab_3)
        self.comboBox_turn_anticipation_model.setGeometry(QtCore.QRect(40, 110, 191, 22))
        self.comboBox_turn_anticipation_model.setObjectName("comboBox_turn_anticipation_model")
        self.comboBox_turn_anticipation_model.addItem("")
        #added stuff
        view = self.comboBox_turn_anticipation_model.view()
        view.setFixedWidth(400)
        #end of added stuff
        self.comboBox_frequency_of_speed_check = QtWidgets.QComboBox(self.tab_3)
        self.comboBox_frequency_of_speed_check.setGeometry(QtCore.QRect(40, 160, 191, 22))
        self.comboBox_frequency_of_speed_check.setObjectName("comboBox_frequency_of_speed_check")
        self.comboBox_frequency_of_speed_check.addItem("")
        self.comboBox_frequency_of_speed_check.addItem("")
        self.comboBox_frequency_of_speed_check.addItem("")
        self.comboBox_frequency_of_speed_check.addItem("")
        self.lineEdit_minimum_speed = QtWidgets.QLineEdit(self.tab_3)
        self.lineEdit_minimum_speed.setGeometry(QtCore.QRect(170, 220, 61, 20))
        self.lineEdit_minimum_speed.setObjectName("lineEdit_minimum_speed")
        self.lineEdit_maximum_speed = QtWidgets.QLineEdit(self.tab_3)
        self.lineEdit_maximum_speed.setGeometry(QtCore.QRect(170, 270, 61, 20))
        self.lineEdit_maximum_speed.setObjectName("lineEdit_maximum_speed")
        self.label_minimum_speed = QtWidgets.QLabel(self.tab_3)
        self.label_minimum_speed.setGeometry(QtCore.QRect(40, 220, 81, 16))
        self.label_minimum_speed.setObjectName("label_minimum_speed")
        self.label_maximum_speed = QtWidgets.QLabel(self.tab_3)
        self.label_maximum_speed.setGeometry(QtCore.QRect(40, 270, 81, 16))
        self.label_maximum_speed.setObjectName("label_maximum_speed")
        self.pushButton_open_file_test = QtWidgets.QPushButton(self.tab_3)
        self.pushButton_open_file_test.setGeometry(QtCore.QRect(360, 340, 75, 23))
        self.pushButton_open_file_test.setObjectName("pushButton_open_file_test")
        self.pushButton_reset_file_test = QtWidgets.QPushButton(self.tab_3)
        self.pushButton_reset_file_test.setGeometry(QtCore.QRect(360, 380, 75, 23))
        self.pushButton_reset_file_test.setObjectName("pushButton_reset_file_test")
        self.tab_widget.addTab(self.tab_3, "")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 498, 21))
```

Interface.py (συνέχεια)

```
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.tab_widget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

#added code to the auto generated pyqt designer

        #def launch_ds4_clicked():
        #    subprocess.call('start C:/Users/jimpa/Desktop/DS4Windows/DS4Windows.exe', shell=True)
        def launch_x360_clicked():
            subprocess.call('start C:/Users/jimpa/Desktop/Joystick_emulator/x360ce/x360ce.exe', shell=True)

        def launch_vjoy_controller_clicked():
            subprocess.call('start C:/Program\" "Files/vJoy/x64/vjoyFeeder.exe', shell=True)

        def launch_ac_clicked():
            subprocess.call('start G:/steam/steamapps/common/assettocorsa/AssettoCorsa', shell=True)

        def pushButton_open_file_main_clicked():
            content = self.file_list.currentText()
            print(content)
            path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/'
            command = 'python -m idlelib '+path+content
            subprocess.call(command)

        def start_real_time_data_collection():
            folder_name = self.lineEdit_real_time_steering_dataset_folder_name.text()
            event = Event()
            event.clear()
            thread = Thread(target=data_collecting_main, args=(folder_name, event))
            thread.start()
            while True:
                keyboard_keys = key_check()
                if 'S' in keyboard_keys:
                    event.set()
                    break

        def reset_file_data_collecting(self):
            msgBox = QMessageBox()
            msgBox.setIcon(QMessageBox.Information)
            msgBox.setText("Are you sure you want to reset the data collecting file to default?")
            msgBox.setWindowTitle("Data collecting file reset")
            msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
            returnValue = msgBox.exec()
            if returnValue == QMessageBox.Yes:
                os.remove('C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/data_collecting.py')
                src = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/default_files/data_collecting_default.py'
                dst = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/data_collecting.py'
                shutil.copy(src,dst)

        def open_file_data_collect_realtime_clicked():
            path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/data_collecting.py'
            command = 'python -m idlelib '+path
            subprocess.call(command)

```

Interface.py (συνέχεια)

```
323         def start_future_steering_data_collection():
324             folder_name = self.lineEdit_future_steering_folder_name.text()
325             frame_distance = self.lineEdit_frame_distance_data_collect.text()
326             event = Event()
327             event.clear()
328             if self.checkBox_both_datasets_at_once.checkState()==2:
329                 thread = Thread(target=data_collecting_complete_main, args=(folder_name, frame_distance, event))
330                 thread.start()
331             else:
332                 thread = Thread(target=future_data_collecting_main, args=(folder_name, frame_distance, event))
333                 thread.start()
334                 while True:
335                     keyboard_keys = key_check()
336                     if 'S' in keyboard_keys:
337                         event.set()
338                         break
339 
340         def reset_file_data_collecting_future():
341             msgBox = QMessageBox()
342             msgBox.setIcon(QMessageBox.Information)
343             msgBox.setText("Are you sure you want to reset the future data collecting file to default?")
344             msgBox.setWindowTitle("Future data collecting file reset")
345             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
346             returnValue = msgBox.exec()
347             if returnValue == QMessageBox.Yes:
348                 os.remove('C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/data_collecting_turn_anticipation.py')
349                 src = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/default_files/data_collecting_turn_anticipation_default.py'
350                 dst = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/data_collecting_turn_anticipation.py'
351                 shutil.copy(src,dst)
352 
353         def open_file_data_collect_future_clicked():
354             path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/data_collecting_turn_anticipation.py'
355             command = 'python -m idlelib '+path
356             subprocess.call(command)
357 
358         def start_steering_model_training():
359             model_name = self.lineEdit_steering_model_file_name.text()
360             print(model_name)
361             dataset_folder = self.comboBox_steering_dataset.currentText()
362             print(dataset_folder)
363             thread = Thread(target=train_model_main, args=(dataset_folder, model_name))
364             thread.start()
365 
366         def stop_steering_model_training():
367             msgBox = QMessageBox()
368             msgBox.setIcon(QMessageBox.Information)
369             msgBox.setText("Are you sure you want to stop the training and shut down the program?")
370             msgBox.setWindowTitle("Shuting down")
371             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
372             returnValue = msgBox.exec()
373             if returnValue == QMessageBox.Yes:
374                 exit()
375 
376         def reset_steering_model_file():
377             msgBox = QMessageBox()
378             msgBox.setIcon(QMessageBox.Information)
379             msgBox.setText("Are you sure you want to reset the model file to default?")
380             msgBox.setWindowTitle("Model file reset")
381             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
382             returnValue = msgBox.exec()
383             if returnValue == QMessageBox.Yes:
384                 print('we are in')
385                 os.remove('C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/train_model_steering.py')
386                 src = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/default_files/train_model_steering_default.py'
387                 dst = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/train_model_steering.py'
388                 shutil.copy(src,dst)
389 
```

Interface.py (συνέχεια)

```
390         def open_train_model_steering_file():
391             path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/train_model_steering.py'
392             command = 'python -m idlelib '+path
393             subprocess.call(command)
394 
395         def start_turn_anticipation_model_training():
396             model_name = self.lineEdit_turn_anticipation_model_file_name.text()
397             dataset_folder = self.comboBox_turn_anticipation_dataset.currentText()
398             thread = Thread(target=train_model_turn_anticipation_main, args=(dataset_folder, model_name))
399             thread.start()
400 
401         def stop_turn_anticipation_model_training():
402             msgBox = QMessageBox()
403             msgBox.setIcon(QMessageBox.Information)
404             msgBox.setText("Are you sure you want to stop the training and shut down the program?")
405             msgBox.setWindowTitle("Shuting down")
406             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
407             returnValue = msgBox.exec()
408             if returnValue == QMessageBox.Yes:
409                 exit()
410 
411         def reset_turn_anticipation_model_file():
412             msgBox = QMessageBox()
413             msgBox.setIcon(QMessageBox.Information)
414             msgBox.setText("Are you sure you want to reset the model file to default?")
415             msgBox.setWindowTitle("Model file reset")
416             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
417             returnValue = msgBox.exec()
418             if returnValue == QMessageBox.Yes:
419                 os.remove('C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/train_model_turn_anticipation.py')
420                 src = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/default_files/train_model_turn_anticipation_default.py'
421                 dst = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/train_model_turn_anticipation.py'
422                 shutil.copy(src,dst)
423 
424         def open_train_model_turn_anticipation_file():
425             path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/train_model_turn_anticipation.py'
426             command = 'python -m idlelib '+path
427             subprocess.call(command)
428         self.tensorboard_counter = 0
429 
430         def open_tensorboard():
431             command = "tensorboard --logdir=C:/Users\jimpa/Desktop/ergasia_self_driving_car/phase4_interface/logs --port=6009"
432             subprocess.call(command)
433             webbrowser.open('http://localhost:6009/')
434 
435         def launch_tensorboard():
436             tensorboard_counter = 0
437             if self.tensorboard_counter == 0:
438                 thread = Thread(target=open_tensorboard)
439                 thread.start()
440                 self.tensorboard_counter = self.tensorboard_counter + 1
441                 webbrowser.open('http://localhost:6009/')
442             else:
443                 print('already running')
444                 msgBox = QMessageBox()
445                 msgBox.setIcon(QMessageBox.Information)
446                 msgBox.setText("Tensorflow is already running, do you want to open the website?")
447                 msgBox.setWindowTitle("Tensorflow")
448                 msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
449                 returnValue = msgBox.exec()
450                 if returnValue == QMessageBox.Yes:
451                     webbrowser.open('http://localhost:6009/')
452 
```

Interface.py (συνέχεια)

```
453         def run_model():
454             #gathering data
455             steering_model = dataset_folder = self.comboBox_steering_model.currentText()
456             turn_anticipation_model = self.comboBox_turn_anticipation_model.currentText()
457             speed_test_freq = self.comboBox_frequency_of_speed_check.currentText()
458             min_speed = int(self.lineEdit_minimum_speed.text())
459             max_speed = int(self.lineEdit_maximum_speed.text())
460             #running test function
461             thread = Thread(target=test_model_main, args=(steering_model, turn_anticipation_model, speed_test_freq, min_speed, max_speed))
462             thread.start()
463         self.model_paused = False
464 
465         def pause_model():
466             keyboard = Controller()
467             keyboard.press('p')
468             keyboard.release('p')
469             if self.model_paused==False:
470                 self.model_paused = True
471                 msgBox = QMessageBox()
472                 msgBox.setIcon(QMessageBox.Information)
473                 msgBox.setText("Model is paused, autonomous driving is switched off")
474                 msgBox.setWindowTitle("Autonomous driving state")
475                 msgBox.setStandardButtons(QMessageBox.Ok)
476                 returnValue = msgBox.exec()
477             else:
478                 self.model_paused = False
479                 msgBox = QMessageBox()
480                 msgBox.setIcon(QMessageBox.Information)
481                 msgBox.setText("Model is unpaused, autonomous driving is switched on")
482                 msgBox.setWindowTitle("Autonomous driving state")
483                 msgBox.setStandardButtons(QMessageBox.Ok)
484                 returnValue = msgBox.exec()
485 
486         def stop_model():
487             keyboard = Controller()
488             msgBox = QMessageBox()
489             msgBox.setIcon(QMessageBox.Information)
490             msgBox.setText("Model is about to be terminated, are you sure you want to continue?")
491             msgBox.setWindowTitle("Autonomous driving state")
492             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
493             returnValue = msgBox.exec()
494             if returnValue==QMessageBox.Yes:
495                 keyboard.press('s')
496                 keyboard.release('s')
497 
498         def open_file_test_model():
499             path = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/test_model.py'
500             command = 'python -m idlelib '+path
501             subprocess.call(command)
502 
503         def reset_file_test_model(self):
504             msgBox = QMessageBox()
505             msgBox.setIcon(QMessageBox.Information)
506             msgBox.setText("Are you sure you want to reset the test model file to default?")
507             msgBox.setWindowTitle("Test model file reset")
508             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
509             returnValue = msgBox.exec()
510             if returnValue == QMessageBox.Yes:
511                 os.remove('C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/test_model.py')
512                 src = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/default_files/test_model_default.py'
513                 dst = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/test_model.py'
514                 shutil.copy(src,dst)
```

Interface.py (συνέχεια)

```
515
516         def reset_selected_file():
517             msgBox = QMessageBox()
518             msgBox.setIcon(QMessageBox.Information)
519             msgBox.setText("Are you sure you want to reset the test model file to default?")
520             msgBox.setWindowTitle("Test model file reset")
521             msgBox.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
522             returnValue = msgBox.exec()
523             if returnValue == QMessageBox.Yes:
524                 selected_file = self.file_list.currentText()
525                 before_dot,after_dot = selected_file.split('.', 1)
526                 selected_file_default = before_dot+'_default.'+after_dot
527                 os.remove('C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/'+selected_file)
528                 src = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/default_files/'+selected_file_default
529                 dst = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface/'+selected_file
530                 shutil.copy(src,dst)
531
532
533         ##self.pushButton_ds4_launch.clicked.connect(launch_ds4_clicked)
534         self.pushButton_x360_launch.clicked.connect(launch_x360_clicked)
535         self.pushButton_vjoy_controller_launch.clicked.connect(launch_vjoy_controller_clicked)
536         self.pushButton_ac_launch.clicked.connect(launch_ac_clicked)
537         self.pushButton_open_file_main.clicked.connect(pushButton_open_file_main_clicked)
538         self.pushButton_start_data_collect_realtime.clicked.connect(start_real_time_data_collection)
539         self.pushButton_reset_file_data_collect_realtime.clicked.connect(reset_file_data_collecting)
540         self.pushButton_open_file_data_collect_realtime.clicked.connect(open_file_data_collect_realtime_clicked)
541         self.pushButton_start_future_steering_inputs.clicked.connect(start_future_steering_data_collection)
542         self.pushButton_reset_file_data_collect_future_inputs.clicked.connect(reset_file_data_collecting_future)
543         self.pushButton_open_file_future_steering_inputs.clicked.connect(open_file_data_collect_future_clicked)
544         self.pushButton_start_steering_model_train.clicked.connect(start_steering_model_training)
545         self.pushButton_reset_model_at_default.clicked.connect(reset_steering_model_file)
546         self.pushButton_stop_steering_model_train.clicked.connect(stop_steering_model_training)
547         self.pushButton_open_file_steering_model.clicked.connect(open_train_model_steering_file)
548         self.pushButton_start_turn_anticipation_model_train.clicked.connect(start_turn_anticipation_model_training)
549         self.pushButton_stop_turn_anticipation_model_train.clicked.connect(stop_turn_anticipation_model_training)
550         self.pushButton_reset_turn_anticipation_model_at_default.clicked.connect(reset_turn_anticipation_model_file)
551         self.pushButton_open_file_turn_anticipation_model.clicked.connect(open_train_model_turn_anticipation_file)
552         self.pushButton_open_tensorboard.clicked.connect(launch_tensorboard)
553         self.pushButton_run_test.clicked.connect(run_model)
554         self.pushButton_pause_test.clicked.connect(pause_model)
555         self.pushButton_stop_test.clicked.connect(stop_model)
556         self.pushButton_open_file_test.clicked.connect(open_file_test_model)
557         self.pushButton_reset_file_test.clicked.connect(reset_file_test_model)
558         self.pushButton_reset_selected_file.clicked.connect(reset_selected_file)
559
560
561     def init(self):
562             mypath = 'C:/Users/jimpa/Desktop/ergasia_self_driving_car/phase4_interface'
563             general_file_list = [f for f in listdir(mypath) if isfile(join(mypath, f)) and re.search( ".py$",f)]
564             self.file_list.clear()
565             self.file_list.addItems(general_file_list)
566             data_folders_steering = os.listdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder')
567             self.comboBox_steering_dataset.addItems(data_folders_steering)
568             data_folders_turn_anticipation = os.listdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/data_folder_turn_anticipation')
569             self.comboBox_turn_anticipation_dataset.addItems(data_folders_turn_anticipation)
570             saved_steering_models = os.listdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/saved_models')
571             self.comboBox_steering_model.addItems(saved_steering_models)
572             saved_turn_anticipation_models = os.listdir('C:/Users/jimpa/Desktop/ergasia_self_driving_car/saved_models_turn_anticipation')
573             self.comboBox_turn_anticipation_model.addItems(saved_turn_anticipation_models)
574
575 #added code ending
576
```

Interface.py (συνέχεια)

```
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "GUI"))
        #self.pushButton_ds4_launch.setToolTip(_translate("MainWindow", "Dualshock 4 (Ps4 controller) simulator "))
        #self.pushButton_ds4_launch.setText(_translate("MainWindow", "Launch DS4"))
        self.pushButton_x360_launch.setToolTip(_translate("MainWindow", "XBox controller simumlator"))
        self.pushButton_x360_launch.setText(_translate("MainWindow", "Launch x360"))
        self.pushButton_vjoy_controller_launch.setToolTip(_translate("MainWindow", "Virtual controller interface"))
        self.pushButton_vjoy_controller_launch.setText(_translate("MainWindow", "Launch vJoy Controller"))
        self.pushButton_ac_launch.setText(_translate("MainWindow", "Launch AssetoCorsa"))
        self.file_list.setToolTip(_translate("MainWindow", "List of all the application\'s files"))
        self.file_list.setItemText(0, _translate("MainWindow", "list of files"))
        self.pushButton_open_file_main.setToolTip(_translate("MainWindow", "Open file for manual permanent changes, reset to revert back"))
        self.pushButton_open_file_main.setText(_translate("MainWindow", "Open"))
        self.pushButton_reset_selected_file.setToolTip(_translate("MainWindow", "Reset chosen file from list to default version"))
        self.pushButton_reset_selected_file.setText(_translate("MainWindow", "Reset"))
        self.pushButton_reset_all_files.setToolTip(_translate("MainWindow", "Reset all the files in the list to default version"))
        self.pushButton_reset_all_files.setText(_translate("MainWindow", "Reset All"))
        self.pushButton_reset_file_data_collect_realtime.setToolTip(_translate("MainWindow", "Reset chosen file from list to default version"))
        self.pushButton_reset_file_data_collect_realtime.setText(_translate("MainWindow", "Reset"))
        self.pushButton_reset_file_data_collect_future_inputs.setToolTip(_translate("MainWindow", "Reset all the files in the list to default version"))
        self.pushButton_reset_file_data_collect_future_inputs.setText(_translate("MainWindow", "Reset All"))
        self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_0), _translate("MainWindow", "General"))
        self.future_steering_inputs.setTitle(_translate("MainWindow", "Future Steering Inputs"))
        self.label_future_steering_folder_name.setText(_translate("MainWindow", "Folder Name"))
        self.lineEdit_future_steering_folder_name.setToolTip(_translate("MainWindow", "Name the Dataset"))
        self.label_frame_distance.setText(_translate("MainWindow", "Frame Distance"))
        self.lineEdit_frame_distance_data_collect.setToolTip(_translate("MainWindow", "Define the frame distance between snapshot and action"))
        self.pushButton_start_future_steering_inputs.setToolTip(_translate("MainWindow", "Start the process of collecting data for future steering inputs, X frames ahead of
        self.pushButton_start_future_steering_inputs.setText(_translate("MainWindow", "Start"))
        self.pushButton_open_file_future_steering_inputs.setToolTip(_translate("MainWindow", "Open file for manual permanent changes, reset to revert back"))
        self.pushButton_open_file_future_steering_inputs.setText(_translate("MainWindow", "Open File"))
        self.checkBox_both_datasets_at_once.setToolTip(_translate("MainWindow", "Enabling this will create a dataset that combines the data of the real-time steering inputs
        self.checkBox_both_datasets_at_once.setText(_translate("MainWindow", "Create both datasets at once"))
        self.label_pause_future_steering_inputs.setText(_translate("MainWindow", "Press \"P\" to pause"))
        self.label_stop_future_steering_inputs.setText(_translate("MainWindow", "Press \"S\" to stop"))
        self.pushButton_reset_file_data_collect_future_inputs.setToolTip(_translate("MainWindow", "Reset to the default version of the file"))
        self.pushButton_reset_file_data_collect_future_inputs.setText(_translate("MainWindow", "Reset"))
        self.realtime_steering_inputs.setTitle(_translate("MainWindow", "Real-time Steering Inputs"))
        self.label_realtime_steering_folder_name.setText(_translate("MainWindow", "Folder Name"))
        self.lineEdit_real_time_steering_dataset_folder_name.setToolTip(_translate("MainWindow", "Name the Dataset"))
        self.pushButton_start_data_collect_realtime.setToolTip(_translate("MainWindow", "Start process of collecting data of real-time steering inputs, matching the time th
        self.pushButton_start_data_collect_realtime.setText(_translate("MainWindow", "Start"))
        self.pushButton_open_file_data_collect_realtime.setToolTip(_translate("MainWindow", "Open file for manual permanent changes, reset to revert back"))
        self.pushButton_open_file_data_collect_realtime.setText(_translate("MainWindow", "Open File"))
        self.label_pause_real_time_steering_inputs.setText(_translate("MainWindow", "Press \"P\" to pause"))
        self.label_stop_real_time_steering_inputs.setText(_translate("MainWindow", "Press \"S\" to stop"))
        self.pushButton_reset_file_data_collect_realtime.setToolTip(_translate("MainWindow", "Reset to the default version of the file"))
        self.pushButton_reset_file_data_collect_realtime.setText(_translate("MainWindow", "Reset"))
        self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_1), _translate("MainWindow", "Collect Data"))
        self.pushButton_open_tensorboard.setToolTip(_translate("MainWindow", "Activate TensorBoard and launch TensorBoard webpage"))
        self.pushButton_open_tensorboard.setText(_translate("MainWindow", "Results in TensorBoard"))
        self.groupBox_3.setTitle(_translate("MainWindow", "Steering Model"))
        self.comboBox_steering_dataset.setToolTip(_translate("MainWindow", "Choose steering dataset to train on"))
        self.comboBox_steering_dataset.setItemText(0, _translate("MainWindow", "Choose Dataset"))
        self.lineEdit_steering_model_file_name.setText(_translate("MainWindow", "Type model file name"))
        self.pushButton_start_steering_model_train.setToolTip(_translate("MainWindow", "Start the training process"))
        self.pushButton_start_steering_model_train.setText(_translate("MainWindow", "Start"))
        self.pushButton_stop_steering_model_train.setToolTip(_translate("MainWindow", "Stop the training process and shut down the program"))
        self.pushButton_stop_steering_model_train.setText(_translate("MainWindow", "Stop"))
        self.pushButton_open_file_steering_model.setToolTip(_translate("MainWindow", "Open train-steering-model file for manual permanent changes, reset to revert back"))
        self.pushButton_open_file_steering_model.setText(_translate("MainWindow", "open file"))
        self.label_choose_modify_model.setText(_translate("MainWindow", "Choose / Modify the model"))
        self.pushButton_reset_model_at_default.setToolTip(_translate("MainWindow", "Reset to the default version of the file"))
        self.pushButton_reset_model_at_default.setText(_translate("MainWindow", "Reset"))
```

Interface.py (συνέχεια)

```
        self.groupBox_4.setTitle(_translate("MainWindow", "Turn Anticipation Model"))
        self.comboBox_turn_anticipation_dataset.setToolTip(_translate("MainWindow", "Choose turn anticipation dataset to train on"))
        self.comboBox_turn_anticipation_dataset.setItemText(0, _translate("MainWindow", "Choose Dataset"))
        self.lineEdit_turn_anticipation_model_file_name.setText(_translate("MainWindow", "Type model file name"))
        self.pushButton_start_turn_anticipation_model_train.setToolTip(_translate("MainWindow", "Start the training process"))
        self.pushButton_start_turn_anticipation_model_train.setText(_translate("MainWindow", "Start"))
        self.pushButton_stop_turn_anticipation_model_train.setToolTip(_translate("MainWindow", "Stop the training process and shut down the program"))
        self.pushButton_stop_turn_anticipation_model_train.setText(_translate("MainWindow", "Stop"))
        self.pushButton_open_file_turn_anticipation_model.setToolTip(_translate("MainWindow", "Open train-turn-anticipation-model file for manual permanent changes, reset to revert back"))
        self.pushButton_open_file_turn_anticipation_model.setText(_translate("MainWindow", "open file"))
        self.pushButton_reset_turn_anticipation_model_at_default.setToolTip(_translate("MainWindow", "Reset to the default version of the file"))
        self.pushButton_reset_turn_anticipation_model_at_default.setText(_translate("MainWindow", "Reset"))
        self.label_choose_modify_model_2.setText(_translate("MainWindow", "Choose / Modify the model"))
        self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_2), _translate("MainWindow", "Train Models"))
        self.comboBox_steering_model.setToolTip(_translate("MainWindow", "This is the model that controls the steering angle of the car"))
        self.comboBox_steering_model.setCurrentText(_translate("MainWindow", "Choose Steering Model"))
        self.comboBox_steering_model.setItemText(0, _translate("MainWindow", "Choose Steering Model"))
        self.pushButton_run_test.setToolTip(_translate("MainWindow", "Start the autonomous driving"))
        self.pushButton_run_test.setText(_translate("MainWindow", "Run"))
        self.pushButton_stop_test.setToolTip(_translate("MainWindow", "Terminate the autonomous driving"))
        self.pushButton_stop_test.setText(_translate("MainWindow", "Stop"))
        self.pushButton_pause_test.setToolTip(_translate("MainWindow", "Turn on/off the autonomous driving"))
        self.pushButton_pause_test.setText(_translate("MainWindow", "Pause"))
        self.comboBox_turn_anticipation_model.setToolTip(_translate("MainWindow", "This is the model that predicts the curviture of the road ahead"))
        self.comboBox_turn_anticipation_model.setCurrentText(_translate("MainWindow", "Choose Turn Anticipation Model"))
        self.comboBox_turn_anticipation_model.setItemText(0, _translate("MainWindow", "Choose Turn Anticipation Model"))
        self.comboBox_frequency_of_speed_check.setToolTip(_translate("MainWindow", "This defines how often (every how many frames) the car will adjust its speed based on the curviture of the road "))
        self.comboBox_frequency_of_speed_check.setItemText(0, _translate("MainWindow", "Choose Frequency of speed check"))
        self.comboBox_frequency_of_speed_check.setItemText(1, _translate("MainWindow", "2 frames"))
        self.comboBox_frequency_of_speed_check.setItemText(2, _translate("MainWindow", "5 frames"))
        self.comboBox_frequency_of_speed_check.setItemText(3, _translate("MainWindow", "10 frames"))
        self.lineEdit_minimum_speed.setToolTip(_translate("MainWindow", "Minimum speed that the car can travel with, or speed during maximum steering angle"))
        self.lineEdit_maximum_speed.setToolTip(_translate("MainWindow", "Maximum speed that the car can travel with, or speed during 0 steering angle"))
        self.label_minimum_speed.setText(_translate("MainWindow", "Minumum Speed"))
        self.label_maximum_speed.setText(_translate("MainWindow", "Maximum Speed"))
        self.pushButton_open_file_test.setToolTip(_translate("MainWindow", "Open file for manual permanent changes, reset to revert back"))
        self.pushButton_open_file_test.setText(_translate("MainWindow", "Open File"))
        self.pushButton_reset_file_test.setToolTip(_translate("MainWindow", "Reset file to its default version"))
        self.pushButton_reset_file_test.setText(_translate("MainWindow", "Reset"))
        self.tab_widget.setTabText(self.tab_widget.indexOf(self.tab_3), _translate("MainWindow", "Test Models"))


#main changed from the default auto-generated pyqt designer code
def main():
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    ui.init()
    sys.exit(app.exec_())


if __name__ == "__main__":
    main()
```