



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Διπλωματική εργασία

Υλοποίηση αυτοοδηγούμενου οχήματος σε
περιβάλλον προσομοίωσης με χρήση τεχνικών deep
learning

Σπύρος Καφτάνης
kaftanis@ceid.upatras.gr
AM: 5542

Οκτώβριος 2018

Επιβλέπων καθηγητής:

Σπυρίδων Λυκοθανάσης

Σπύρος Δ. Καφτάνης

Υλοποίηση αυτοοδηγούμενου οχήματος σε περιβάλλον προσομοίωσης
με χρήση τεχνικών deep learning

Διπλωματική εργασία. Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών, Οκτώβριος 2018

ACKNOWLEDGMENT

.....

ΠΕΡΙΛΗΨΗ

Τα αυτοοδηγούμενα (self-driving) οχήματα ήδη έχουν αρχίσει να κάνουν την εμφάνισή τους στην αγορά και προβλέπεται ότι μέσα στα επόμενα χρόνια θα έχουν κατακλίσει τους δρόμους. Τα οχήματα αυτά με την βοήθεια αλγορίθμων μηχανικής μάθησης (machine learning) και δεδομένων από πραγματικούς οδηγούς μπορούν να μάθουν να οδηγούν και να κινούνται ακόμα και σε διαδρομές που δεν έχουν δει ξανά. Τα δεδομένα εκπαίδευσης των αλγορίθμων είναι συνήθως βίντεο από κάμερες που τοποθετούνται πάνω στα οχήματα καθώς και σήματα από διάφορους αισθητήρες ή και από gps.

Για την υλοποίηση των αλγορίθμων αυτών χρησιμοποιούνται συνήθως προσομοιωτές μιας και η απευθείας εφαρμογή τους στα πραγματικά οχήματα είναι ιδιαίτερα δαπανηρή. Ένα είδος προσομοιωτή είναι και ένα video game το οποίο έχει σκοπό να προσφέρει μία όσο το δυνατόν πιο ρεαλιστική εμπειρία στους παίκτες. Μάλιστα αυτού του είδους προσομοίωση χρησιμοποιείται και από μεγάλες εταιρίες κατασκευής αυτοοδηγούμενων οχημάτων. [1][[2]

Σε αυτή τη διπλωματική εργασία σκοπός είναι η υλοποίηση ενός intelligent agent (IA) ο οποίος θα μπορεί να οδηγήσει ένα μονοθέσιο τύπου formula με τον καλύτερο δυνατό τρόπο στο εικονικό περιβάλλον προσομοίωσης του επίσημου παιχνιδιού της formula 1. Η εκπαίδευση του αλγορίθμου μάθησης γίνεται με δεδομένα που παρήχθησαν από την αλληλεπίδραση του γράφοντα με το παιχνίδι, είναι δηλαδή supervised. Για τη λύση του συγκεκριμένου προβλήματος, χρειάστηκε να μελετηθούν διάφορα θέματα όπως η προεπεξεργασία δεδομένων διαφορετικού τύπου, τα νευρωνικά δίκτυα και κυρίως τα convolutional neural networks, η βελτιστοποίηση των παραμέτρων τους κ.α. Επίσης αντιμετωπίστηκαν αρκετές προγραμματιστικές προκλήσεις που αφορούσαν κυρίως την επικοινωνία με το περιβάλλον προσομοίωσης.

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή.....	1
1.1 Η τεχνητή νοημοσύνη και η μηχανική μάθηση στον σύγχρονο κόσμο.....	1
1.2 Αυτοοδηγούμενα οχήματα.....	2
1.3 Προκλήσεις αυτοοδήγησης σε αγωνιστικά οχήματα.....	3
1.4 Αντικείμενο διπλωματικής.....	3
1.4.1 Συνεισφορά.....	4
1.5 Οργάνωση κειμένου.....	4
2. Σχετικές εργασίες.....	5
2.1 Roborace.....	5
2.2 Udacity's Self-Driving Car Simulator.....	6
3. Θεωρητικό υποβάθρο.....	9
3.1 Στατιστική.....	9
3.1.1 Κανονικοποίηση αλλαγής διαστήματος (Feature scaling).....	10
3.1.2 Κανονικοποίηση με χρήση τυπικής απόκλισης (Standardization).....	10
3.2 Επεξεργασία εικόνας.....	11
3.2.1 RGB κωδικοποίηση χρωμάτων.....	11
3.2.2 YUV κωδικοποίηση χρωμάτων.....	12
3.3 Μηχανική μάθηση και νευρωνικά δίκτυα.....	13
3.3.1 Το απλό perceptron.....	14
3.3.2 Deep Learning.....	16
3.3.3 Convolutional Neural Networks (CNN).....	16
4. Συλλογή και επεξεργασία δεδομένων.....	17
4.1 Δεδομένα ροής βίντεο.....	17
4.2 Δεδομένα τηλεμετρίας.....	18
4.3 Δεδομένα εισόδου.....	20
4.4 Τρόπος αποθήκευσης δεδομένων.....	20
4.5 Προεπεξεργασία δεδομένων.....	21
4.5.1 Επεξεργασία εικόνας.....	25
4.5.1.A Περιοχή ενδιαφέροντος.....	25
4.5.1.B Κωδικοποίηση χρωμάτων.....	25
4.5.1.C Κανονικοποίηση.....	27
4.5.2 Επεξεργασία δεδομένων τηλεμετρίας.....	28
4.5.3 Συνδυασμός εικόνας και τηλεμετρίας.....	29
5. Μοντέλο deep learning.....	31
5.1.1 Πρώτες προσπάθειες.....	31
5.1.2 Περιγραφή τελικού μοντέλου.....	33
5.1.3 Ανάλυση μοντέλου.....	34
5.1.3.A Multimodal learning.....	34

5.1.3.B CNN και fully connected.....	35
5.1.3.C Συνάρτηση ενεργοποίησης.....	37
5.1.3.D Συνάρτηση σφάλματος.....	38
5.1.3.E Το τελικό μοντέλο.....	41
5.1.4 Εκπαίδευση μοντέλου.....	43
6. Αξιολόγηση.....	47
6.1.1 Benchmarking.....	47
6.1.2 Σύγκριση με άλλα μοντέλα.....	48
6.1.3 Βελτίωση του Medium Fusion.....	48
6.1.4 Παρουσίαση εργαλείου αξιολόγησης.....	49
7. Τεχνικές λεπτομερείες.....	53
7.1.1 Τεχνικές προδιαγραφές υπολογιστικών συστημάτων.....	53
7.1.2 Πρακτικές συλλογής δεδομένων.....	54
7.1.3 Πρακτικές κανονικοποίησης δεδομένων.....	54
7.1.4 Εργαλείο αξιολόγησης.....	55
7.1.5 Βιβλιοθήκες.....	56
8. Ιδέες για μελλοντικές εργασίες.....	57
A. βιβλιογραφία.....	59
B. κωδικας.....	60
B.1 read_all_data.py.....	60
B.2 read_steering.py.....	64
B.3 Normalization.py.....	65
B.4 create_data_batches.py.....	66
B.5 train_model.py.....	69
B.6 model.py.....	70
B.7 evaluation_screen.py (evaluation tool).....	71

1.

ΕΙΣΑΓΩΓΗ

Από τη στιγμή που οι άνθρωποι ξεκίνησαν να κατασκευάζουν μηχανές με σκοπό να αυτοματοποιήσουν τις εργασίες τους, άρχισε να ξεπηδά η ιδέα της “σκεπτόμενης μηχανής”. Οι άνθρωποι φαντάστηκαν ένα μέλλον όπου οι μηχανές θα μπορούν να πραγματοποιήσουν εργασίες που απαιτούν σύνθετη σκέψη, πέρα από την αυτοματοποίηση των γραμμών παραγωγής. Σήμερα και καθώς σιγά σιγά φτάνουμε στο σημείο να πραγματοποιήσουμε αυτή τη φαντασίωση, σκεφτόμαστε πλέον το επόμενο στάδιο, δηλαδή το τι θα συμβεί όταν οι μηχανές καταφέρουν να μας ξεπεράσουν σχεδόν στα πάντα.

1.1 Η τεχνητή νοημοσύνη και η μηχανική μάθηση στον σύγχρονο κόσμο

Η τεχνητή νοημοσύνη, κυρίως με τη εφαρμογή αλγορίθμων machine learning και ιδιαίτερα αλγορίθμων deep learning, φαίνεται να είναι ικανή να αλλάξει τον κόσμο σε τέτοιο επίπεδο ώστε να μπορούμε πλέον να μιλάμε για μία τέταρτη βιομηχανική επανάσταση. Όπως φαίνεται, η δημοσιότητα των τελευταίων τριών ετών δεν είναι καθόλου τυχαία και αναμένεται να έχει διάρκεια. Η αύξηση της έρευνας στον τομέα αυτό είναι ραγδαία και τα εμπορικά προϊόντα αυξάνονται διαρκώς σε σημείο όπου πολλές κοινωνικές και πολιτικές επιπτώσεις έχουν ήδη αρχίσει να τρομάζουν τους λαούς. Σε αντίθεση όμως με τους λουδίτες¹, οι περισσότεροι μπορούμε να δούμε καθαρά πως οι τεχνολογίες τέτοιου είδους είναι απλά το αποτέλεσμα της αύξησης του βιοτικού επιπέδου και της ποιότητας ζωής και εργασίας των ανθρώπων. Είναι ίσως ο πιο γρήγορος τρόπος να δημιουργήσουμε κοινωνίες στις οποίες η πλειοψηφία των ανθρώπων θα ασχολείται με πράγματα που απαιτούν σκέψη, εξειδικευμένες ικανότητες, επιχειρηματικό όραμα και κοινωνικές δεξιότητες.

1. Ο λουδιτισμός είναι το κίνημα που ξεκίνησαν οι εργάτες της κλωστοϋφαντουργίας τον 19ο αιώνα στην Αγγλία. Οι λουδίτες διαμαρτύρονταν καταστρέφοντας τις νέες μηχανές των εργοστασίων τους, προϊόντα της Βιομηχανικής Επανάστασης. Οι δικές τους κινητοποιήσεις είχαν και άλλα κίνητρα, αλλά το όνομά τους έχει καθιερωθεί ως χαρακτηρισμός για αυτούς που τάσσονται κατά της τεχνολογίας.

1.2 Αυτοοδηγούμενα οχήματα

Ένας τομέας ο οποίος αναμένεται να είναι από τους πρώτους που θα αντικατασταθεί από τις μηχανές είναι ο τομέας των επαγγελματιών που σχετίζονται με την οδήγηση. Η οδήγηση μπορεί να καταλήγει μία εντελώς αυτοματοποιημένη εργασία για τους ανθρώπους μετά από κάποια εμπειρία, παρ' όλα αυτά η αυτοματοποίησή της είναι ένα εξαιρετικά δύσκολο πρόβλημα. Υπάρχουν εκατομμύρια διαφορετικές καταστάσεις στις οποίες μπορεί να βρεθεί ένας οδηγός, πράγμα που σημαίνει ότι πρέπει να συνδυάσει ικανότητες γρήγορης κατανόησης της τωρινής κατάστασης, γρήγορου εντοπισμού των σημάτων και αίσθηση της μελλοντικής του κατάστασης και όλα αυτά χωρίς να έχει στη διάθεσή του όλα τα νούμερα ή τον γρηγορότερο επεξεργαστή στον κόσμο.

Αναμένεται ότι τα πλήρως αυτοοδηγούμενα οχήματα θα είναι διαθέσιμα μέσα στα επόμενα τέσσερα χρόνια, με διάφορες εταιρίες να κάνουν διαφορετικές προβλέψεις για το πότε αυτό θα επιτευχθεί. Τα σημερινά αυτοοδηγούμενα οχήματα δεν είναι πλήρως αυτοοδηγούμενα, μιας και απαιτούν την παρουσία οδηγού καθώς δεν είναι ικανά να αντεπεξέλθουν σε όλα τα σενάρια. Η τεχνολογία αυτή, όποτε και αν φτάσει στο τελικό στάδιο της αυτοματοποίησης αναμένεται να μειώσει στο ελάχιστο τα ατυχήματα στον δρόμο και η κίνηση, ειδικά όταν περάσουμε το μεταβατικό στάδιο και πάψουμε να έχουμε ανθρώπους οδηγούς, όπου τα οχήματα θα μπορούν να επικοινωνούν μεταξύ τους και να παίρνουν γρήγορα τις βέλτιστες αποφάσεις.



Εικόνα 1.1. Concept αυτοοδηγούμενου οχήματος

1.3 Προκλήσεις αυτοοδήγησης σε αγωνιστικά οχήματα

Κάτι εξίσου ενδιαφέρον είναι η οδήγηση σε πίστα αγώνων και η αυτοματοποίηση αυτού. Οι οδηγοί αγώνων, μιας και κινούνται με ιδιαίτερα υψηλές ταχύτητες και συνήθως χωρίς ηλεκτρονικά βοηθήματα έχουν ένα χαοτικά δυσκολότερο έργο από αυτό των απλών οδηγών στον δρόμο. Συνήθως μάλιστα χαρακτηρίζονται υπεραθλητές. Σήμερα τα αυτοοδηγούμενα οχήματα κινούνται συνήθως σε αρκετά χαμηλές ταχύτητες, πέρα ίσως από τις περιπτώσεις όπου βρίσκονται σε μεγάλες εθνικές οδούς με ευδιάκριτες διακεκομμένες γραμμές όπου το πρόβλημα είναι αρκετά ευκολότερο. Επιπλέον, η οδήγηση στη πίστα δεν ορίζεται ρητά από κάποιες γραμμές, μιας και η λεγόμενη αγωνιστική γραμμή, η οποία στην ουσία ελαχιστοποιεί το στρίψιμο του τιμονιού έτσι ώστε να ελαχιστοποιήσει τον τελικό χρόνο, είναι νοητή.

1.4 Αντικείμενο διπλωματικής

Το αντικείμενο αυτής της διπλωματικής είναι η υλοποίηση ενός οχήματος το οποίο θα μπορεί να κινηθεί με γρήγορη ταχύτητα σε μία εικονική πίστα έχοντας διαθέσιμα λίγα δεδομένα, όπως η μπροστινή εικόνα (και όχι οι πλαϊνές), η θέση του στον χάρτη και η ταχύτητα. Το πρόβλημα θα ήταν σχετικά ευκολότερο αν γινόταν χρήση δεδομένων από αισθητήρες και άλλα όργανα τα οποία όμως δεν είναι διαθέσιμα.

Το πρόβλημα αντιμετωπίζεται ως ένα supervised πρόβλημα, κάτι που σημαίνει ότι για την εκπαίδευση χρησιμοποιήθηκαν δεδομένα πραγματικού gameplay για τα οποία γνωρίζαμε τις “πραγματικές” τιμές εξόδου. Σκοπός είναι ο αλγόριθμος (ο οποίος είναι ένα convolutional νευρωνικό δίκτυο) να μάθει την έννοια της οδήγησης βρίσκοντας μοτίβα στα δεδομένα που του δίνονται σαν είσοδο. Με αυτό το τρόπο θα μπορεί να προβλέψει το ποσοστό γκαζιού και φρένου καθώς και τη γωνία τιμονιού που πρέπει να εφαρμοστούν για κάθε χρονική στιγμή.

Στην ουσία η οδήγηση αντιμετωπίζεται σαν μία συνάρτηση, όπου είσοδος είναι τα pixels της εικόνας, η ταχύτητα και η θέση στη πίστα και έξοδος είναι το ποσοστό του γκαζιού, του φρένου και η γωνία του τιμονιού. Ένα νευρικό δίκτυο ως εκτιμητής συναρτήσεων θα πρέπει να μπορεί να προσεγγίσει τη συνάρτηση αυτή παράγοντας realtime κάποιες επιθυμητές εξόδους.

1.4.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Υλοποίηση συστήματος συλλογής και αποθήκευσης και οργάνωσης βίντεο από την οθόνη, καθώς και δεδομένων τηλεμετρίας από το περιβάλλον προσομοίωσης.
2. Επεξεργασία δεδομένων που περιλαμβάνει αλλαγές στο μοντέλο χρωμάτων, κανονικοποίηση κ.α.
3. Συνδυασμός δεδομένων εικόνας και δεδομένων τηλεμετρίας ως είσοδος σε ένα νευρωνικό δίκτυο.
4. Υλοποίηση CNN που περιλαμβάνει μελέτη αρχιτεκτονικής και επιλογής υπερπαραμέτρων.
5. Υλοποίηση συστήματος αποστολής δεδομένων στο περιβάλλον προσομοίωσης.

1.5 Οργάνωση κειμένου

Στο επόμενο κεφάλαιο θα παρουσιαστούν κάποιες σχετικές εργασίες και το πως διαφέρουν από τη παρούσα ενώ στο κεφάλαιο 3 θα παρουσιαστεί συνοπτικά το θεωρητικό υπόβαθρο που χρειάστηκε για την υλοποίηση αυτής της εργασίας. Στη συνέχεια θα αναλυθούν τα βήματα που αναφέρθηκαν στο 1.4.1 ξεκινώντας από την συλλογή και την επεξεργασία των δεδομένων στο κεφάλαιο 4 και φτάνοντας στην ανάλυση του μοντέλου στο κεφάλαιο 5. Τέλος, παρουσιάζεται μία αξιολόγηση των αποτελεσμάτων (κεφάλαιο 6), κάποιες τεχνικές λεπτομέρειες (κεφάλαιο 7) οι οποίες αποφεύγονται σκοπίμως στα προηγούμενα κεφάλαια και τέλος στο κεφάλαιο 8 κάποιες προοπτικές συνέχισης της εργασίας.

2.

ΣΧΕΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Όπως έγινε σαφές από την εισαγωγή, η έρευνα γύρω από τον τομέα των αυτοοδηγούμενων οχημάτων βρίσκεται σε μεγάλη άνοδο μιας και το πρόβλημα στη πραγματικότητα δεν έχει ακόμα λυθεί. Είτε μιλάμε για δρόμο είτε μιλάμε για πίστα απέχουμε ακόμα μερικά χρόνια από την πλήρη αυτόματη οδήγηση κάτω από όλες τις συνθήκες. Στο κεφάλαιο αυτό θα αναφερθούν συνοπτικά κάποια projects τα οποία είναι κοντά στο θέμα αυτής της διπλωματικής εργασίας, δηλαδή projects που έχουν να κάνουν με αυτόνομη οδήγηση σε πίστα, δηλαδή σε περιβάλλον υψηλών ταχυτήτων χωρίς ιδιαίτερους περιοριστικούς κανόνες (σήμανση κτλ).

2.1 Roborace

Πρόκειται για ένα πραγματικό όχημα τύπου formula, το οποίο σχεδιάστηκε μόνο για αυτόματη οδήγηση. Παρουσιάστηκε για πρώτη φορά στη Βαρκελώνη από τον Denis Sverdlov, τον CEO της Roborace and Charge και τον σχεδιαστή, Daniel Simon. Σκοπός αυτού του εγχειρήματος είναι να γίνει η αρχή σε μία νέα γενιά αγώνων όπου θα ανταγωνίζονται τα λογισμικά και η τεχνογνωσία ακόμα και στην οδήγηση του οχήματος, πράγμα που φαίνεται ιδιαίτερα ενδιαφέρον. Το συγκεκριμένο όχημα βασίζεται στην πλατφόρμα Nvidia Drive PX 2, η οποία συνδυάζει deep learning, έλεγχο αισθητήρων και περιμετρική όραση. Είναι ικανό να καταλάβει σε πραγματικό χρόνο τι συμβαίνει γύρω από το όχημα και να τοποθετήσει με ακρίβεια το όχημα σε έναν εικονικό χάρτη χαράζοντας παράλληλα τη πορεία του.

Το ισχυρό υπολογιστικό hardware σε συνδυασμό με τους δεκάδες αισθητήρες και τις κάμερες τριακοσίων εξήντα μοιρών δίνουν στους ερευνητές πολλά δεδομένα τα οποία μπορούν να αξιοποιήσουν για την επίτευξη του στόχου. Στο project αυτής της διπλωματικής τα δεδομένα είναι πολύ πιο περιορισμένα. Στη πραγματικότητα περιορίζονται στα δεδομένα που θα είχε και ένας πραγματικός οδηγός.



Εικόνα 2.. Το Robocar στο Goodwood hillclimb

2.2 Udacity's Self-Driving Car Simulator

Το Udacity με αφορμή το nanodegree πρόγραμμα “Become a Self Driving Car Engineer” δημιούργησε ένα παιχνίδι με σκοπό να διδάξει στους μαθητές του το πως θα εκπαιδεύσουν ένα όχημα να κινείται στο δρόμο. Το παιχνίδι αυτό περιλαμβάνει ενσωματωμένους τρόπους συλλογής δεδομένων αλλά και δοκιμής των μοντέλων. Περιλαμβάνει επίσης δύο διαφορετικές διαδρομές. Το πρόβλημα αντιμετωπίζεται ως supervised, όπως δηλαδή και το πρόβλημα της παρούσας διπλωματικής. Οι διαφορές παρατηρούνται στο μοντέλο φυσικής του περιβάλλοντος προσομοίωσης, μιας και το μοντέλο φυσικής του περιβάλλοντος του Udacity είναι ιδιαίτερα απλοϊκό, καθώς και στις μικρές ταχύτητες και τις πολλές κάμερες που έχει διαθέσιμες ο προγραμματιστής.



Εικόνα 2.2. Ο εκπαιδευτικός προσομοιωτής του udacity

3.

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Για την εκπόνηση αυτής της διπλωματικής εργασίας χρησιμοποιήθηκαν θεωρητικές γνώσεις από διάφορους τομείς της επιστήμης των υπολογιστών και των μαθηματικών. Πιο συγκεκριμένα για την ανάλυση και την επεξεργασία των δεδομένων, χρήσιμες ήταν κάποιες γνώσεις από τον κλάδο της στατιστικής και την επεξεργασία εικόνας, ενώ για την υλοποίηση του κύριου αλγορίθμου της εργασίας γνώσεις από τον τομέα της μηχανικής μάθησης (machine learning). Στο κεφάλαιο αυτό θα παρουσιαστούν συνοπτικά κάποια θεωρητικά στοιχεία τα οποία κρίνονται σημαντικά για τη κατανόηση του παρόντος κειμένου και της διαδικασίας ακόμα και από άτομα που δεν έχουν εντρυφήσει στον συγκεκριμένο τομέα.

3.1 Στατιστική

Η Στατιστική είναι μία μεθοδική μαθηματική, παλαιότερα τεχνική και σήμερα επιστήμη που επιχειρεί να εξαγάγει έγκυρη γνώση χρησιμοποιώντας εμπειρικά δεδομένα παρατήρησης ή και πειράματος. Κύριο αντικείμενο έρευνας και μελέτης της Στατιστικής είναι η συλλογή, ταξινόμηση, επεξεργασία, παρουσίαση, ανάλυση και ερμηνεία διαφόρων δεδομένων με απώτερο στόχο την εξαγωγή ασφαλών συμπερασμάτων για λήψη ορθών αποφάσεων.

Ένα σημαντικό μέρος της στατιστικής αφορά της επεξεργασία των δεδομένων ώστε να έρθουν σε μία καλύτερη και πιο εύκολα διαχειρίσιμη μορφή. Τα δεδομένα μπορεί να είναι χαοτικά, κάποιες κλάσεις τους μπορεί να βρίσκονται σε πολύ διαφορετικά διαστήματα μεταξύ τους ή να ακολουθούν πολύ ακανόνιστες κατανομές, πράγμα που κάνει την εξαγωγή συμπερασμάτων δύσκολη έως ανέφικτη. Για το σκοπό αυτό χρησιμοποιούνται διάφορες τεχνικές οι οποίες βοηθάνε στην βελτίωση της ποιότητας των δεδομένων για την εξαγωγή συμπερασμάτων.

3.1.1 Κανονικοποίηση αλλαγής διαστήματος (Feature scaling)

Πολλοί αλγόριθμοι machine learning όπως και τα νευρωνικά δίκτυα έχουν σημαντικά προβλήματα όταν τα χαρακτηριστικά που τους δίνονται σαν είσοδο (features) βρίσκονται σε διαφορετικές κλίμακες. Αν για παράδειγμα κάποιο feature παίρνει τιμές στο διάστημα [50,100] και ένα δεύτερο στο διάστημα [0,1] τότε ένας ταξινομητής (classifier) θα έδινε πολύ μεγάλη αξία το πρώτο feature και σχεδόν θα αγνοούσε το δεύτερο. Για να λυθεί αυτό το πρόβλημα συνηθίζεται η μεταφορά όλων των δεδομένων σε ένα κοινό διάστημα το οποίο συνήθως είναι το [0,1] ή το [-1,1].

Ένας τρόπος για τη μεταφορά κάποιων δεδομένων X σε ένα διάστημα [0,1] είναι ο παρακάτω μετασχηματισμός:

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}$$

όπου x i-οστό στοιχείο της λίστας X , $\min(X)$ η ελάχιστη τιμή της και $\max(X)$ η μέγιστη.

Με παρόμοιο τρόπο, χωρίς όμως να ορίζουμε ένα συγκεκριμένο διάστημα $[x,y]$ στο οποίο θέλουμε να οριοθετήσουμε τα δεδομένα κανονικοποιούμε τα δεδομένα χρησιμοποιώντας τη μέση τιμή ως εξής:

$$x' = \frac{x - \text{average}(X)}{\max(X) - \min(X)}$$

Είναι, τέλος, αρκετά προφανής η διαδικασία μεταφοράς των δεδομένων σε οποιοδήποτε άλλο διάστημα επιθυμούμε, αλλά η μεταφορά του στο διάστημα [0,1] συνήθως είναι αρκετή

3.1.2 Κανονικοποίηση με χρήση τυπικής απόκλισης (Standardization)

Η παραπάνω τρόποι μετασχηματισμού των δεδομένων μπορεί να είναι αρκετοί σε πολλά προβλήματα, παρ' όλα αυτά δεν είναι ιδανικοί από άποψη αξιοποίησης του διαστήματος τιμών. Αυτό πρακτικά σημαίνει ότι αν έχουμε πολλές τιμές μαζεμένες κοντά στη τιμή x_1 και πολύ λίγες κοντά στη τιμή x_2 , όπου το x_1 είναι αρκετά μακριά από το x_2 , τότε η παραπάνω κανονικοποίηση θα κρατήσει πολλές τιμές κοντά στη νέα τιμή x_1' και λίγες κοντά στη x_2' έτσι όπως αυτές προέκυψαν από τον μετασχηματισμό. Αυτό δεν είναι ιδανικό μιας και το μεγαλύτερο μέρος και του νέου διαστήματος θα παραμείνει κενό.

Μια πιο αποτελεσματική λύση στο πρόβλημα είναι η μεταφορά της κατανομής των δεδομένων σε μία νέα κατανομή με μέσο μηδέν και τυπική απόκλιση ένα, δηλαδή σε μία κανονική κατανομή $N(0,1)$.

$$x' = \frac{x - \text{mean}(X)}{\text{stddev}(X)}$$

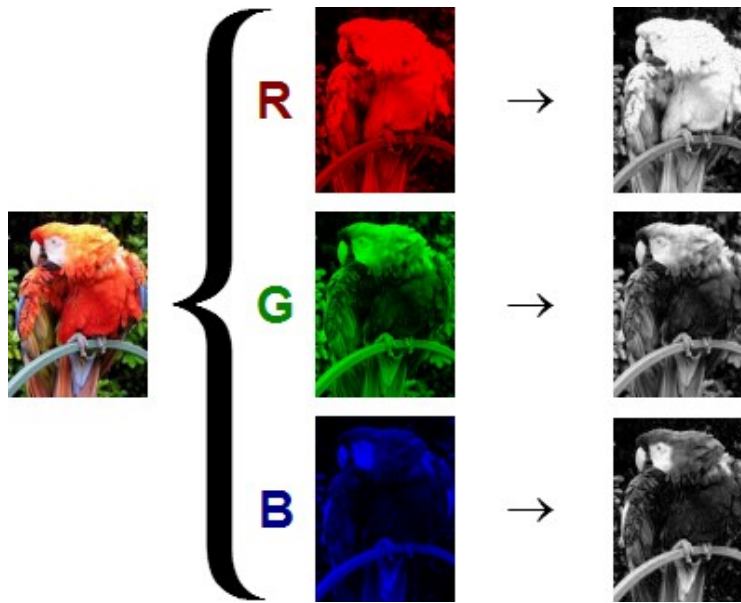
Η μέθοδος αυτή δεν εγγυάται τη μεταφορά σε κάποιο συγκεκριμένο διάστημα, αλλά σε μία συγκεκριμένη κατανομή, πράγμα που συνήθως είναι ακόμα σημαντικότερο.

3.2 Επεξεργασία εικόνας

Επεξεργασία εικόνας ονομάζεται κάθε μορφή αλγοριθμικής επεξεργασίας, ανάλυσης και χειρισμού ψηφιακών δεδομένων εικόνας ή βίντεο, όπως και το σχετικό επιστημονικό πεδίο της πληροφορικής. Στην επεξεργασία εικόνας, τόσο η είσοδος όσο και η έξοδος των υπολογισμών είναι δεδομένα εικόνας / βίντεο (έγχρωμα, ασπρόμαυρα ή σε αποχρώσεις του γκριζου). Από την επεξεργασία εικόνας εκπορεύονται επίσης και αλγόριθμοι ανάλυσης / κατανόησης εικόνας, αλλά εκεί υφίσταται επικάλυψη με το συγγενές γνωστικό πεδίο της τεχνητής νοημοσύνης ονόματι μηχανική όραση.

3.2.1 RGB κωδικοποίηση χρωμάτων

Η αναπαράσταση των εικόνων στη μνήμη ενός υπολογιστή γίνεται με τη χρήση n-διάστατων αριθμητικών πινάκων. Όταν έχουμε ασπρόμαυρες (greyscale) εικόνες ο πίνακας είναι δύο διαστάσεων και κάθε τιμή μέσα σε αυτόν αναπαριστά την απόχρωση του γκριζου. Όταν έχουμε εικόνες στις οποίες θέλουμε να έχουμε χρώμα τότε χρησιμοποιούμε έναν πίνακα τριών διαστάσεων. Κάθε διάσταση του πίνακα ονομάζεται κανάλι (channel) με μία συνηθισμένη αναπαράσταση να είναι η αναπαράσταση RGB στη οποία το κάθε κανάλι αναφέρεται σε κάθε ένα από τα τρία βασικά χρώματα (κόκκινο, πράσινο, μπλε).



Εικόνα 3.1. Διαχωρισμός των RGB καναλιών μίας εικόνας

3.2.2 YUV κωδικοποίηση χρωμάτων

Μία πιο ασυνήθιστη κωδικοποίηση χρωμάτων είναι η κωδικοποίηση YUV η οποία μπορεί να παραχθεί από έναν γραμμικό μετασχηματισμό από τη κωδικοποίηση RGB. Η YUV χρησιμοποιεί επίσης τρεις διαστάσεις. Το Y αναφέρεται στη φωτεινότητα (luminance) και οι συνιστώσες U και V αφορούν στη χρωματική πληροφορία (chrominance). Το μοντέλο YUV χρησιμοποιείται στα συστήματα μετάδοσης PAL και NTSC. Χρησιμοποιήθηκε ως εναλλακτικό του RGB, προκειμένου το έγχρωμο σήμα να είναι συμβατό με τις παλαιότερες ασπρόμαυρες τηλεοράσεις. Οι τηλεοράσεις αυτές απλά αγνοούν τις U και V χρωματικές πληροφορίες. Επειδή γίνεται διαχωρισμός της φωτεινότητας από το χρώμα, υποστηρίζεται η αποδοτική συμπίεση του χρώματος. Για τη παραγωγή μιας YUV κωδικοποίησης έχοντας μία RGB με στοιχεία R, G και B αρκεί να εφαρμόσουμε τον παρακάτω μετασχηματισμό:

$$Y = R * .299000 + G * .587000 + B * .114000$$

$$U = R * -.168736 + G * -.331264 + B * .500000 + 128$$

$$V = R * .500000 + G * -.418688 + B * -.081312 + 128$$

3.3 Μηχανική μάθηση και νευρωνικά δίκτυα

Η μηχανική μάθηση (machine learning) είναι υποπεδίο της επιστήμης των υπολογιστών που αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Το 1959, ο Άρθουρ Σάμουελ ορίζει τη μηχανική μάθηση ως "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί". Η μηχανική μάθηση διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα.

Το πεδίο αυτό έχει παλιές ρίζες, παρ' όλα αυτά η ραγδαία αύξηση στις εφαρμογές, στις επιχειρηματικές δραστηριότητες καθώς και στην έρευνα έγινε φανερή τα τελευταία 6-10 χρόνια λόγω της εύκολης πρόσβασης σε μεγάλο όγκο δεδομένων και σε ισχυρά υπολογιστικά συστήματα.

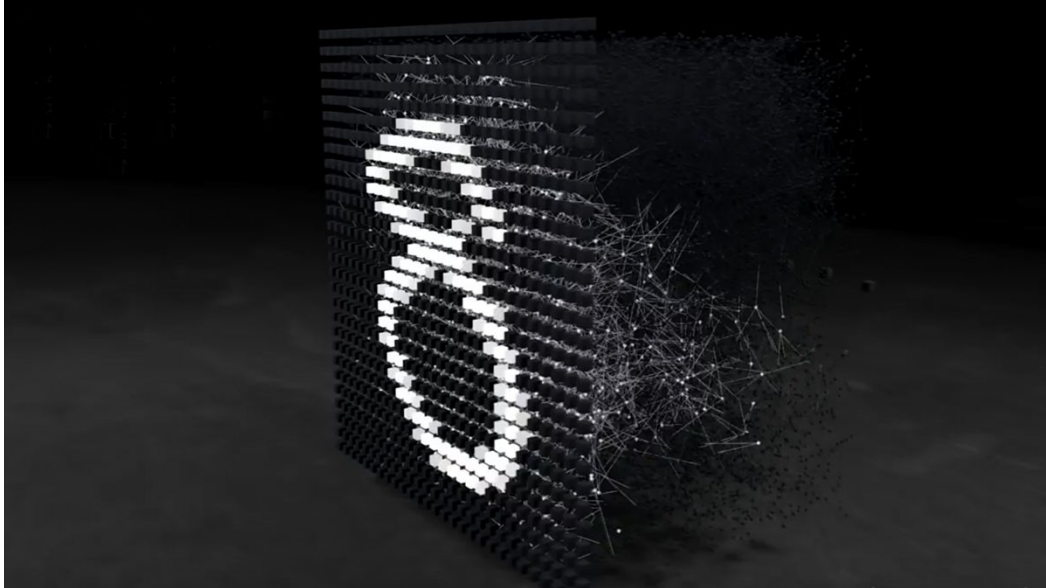
Εάν θελήσουμε να δούμε τη μηχανική μάθηση ως ένα απλουστευμένο μαθηματικό μοντέλο μπορούμε να υποθέσουμε ότι έχουμε ένα σύστημα:

$$AX=B$$

Εάν το A και το B είναι γνωστά τότε η εύρεση του X είναι στη πραγματικότητα ένα πρόβλημα μηχανικής μάθησης. Στη μηχανική μάθηση έχοντας στη διάθεσή μας δεδομένα εισόδου (A) καθώς και το αποτέλεσμα (B), καλούμαστε να υπολογίσουμε τη συνάρτηση που τα συνδέει. Το συγκεκριμένο είδος μηχανικής μάθησης ονομάζεται εποπτευόμενη (supervised) μηχανική μάθηση και με αυτή θα ασχοληθούμε σε αυτή την εργασία.

Τα **νευρωνικά δίκτυα** (artificial neural networks) αρχικά προτάθηκαν ως ένα μαθηματικό μοντέλο προσομοίωσης της πολύπλοκης λειτουργίας του ανθρώπινου εγκεφάλου. Η δομή του εγκεφάλου είναι τέτοια ώστε να επιτρέπει την παράλληλη επεξεργασία δεδομένων και τη δυνατότητα συνεχούς μάθησης μέσω της αλληλεπίδρασης με το περιβάλλον. Τα δύο αυτά βασικά χαρακτηριστικά συμβάλλουν στην ικανότητα, αφενός, να εκτελεί δύσκολα καθήκοντα, όπως ταχύτατη αναγνώριση μορφών, ταξινόμηση κ.ά., αφετέρου, να εξελίσσεται συνεχώς, μαθαίνοντας από το περιβάλλον του κατά την αλληλεπίδρασή του με αυτό. Η δομή του τεχνητού νευρωνικού δικτύου μιμείται κατά το δυνατό εκείνη του βιολογικού νευρωνικού δικτύου, ώστε να εμφανίζει παρόμοιες ιδιότητες. Κατ' αναλογία επομένως με ένα δίκτυο νευρώνων εγκεφάλου, ένα τεχνητό δίκτυο αποτελείται από ένα σύνολο τεχνητών νευρώνων που αλληλεπιδρούν, συνδεδεμένοι μεταξύ τους με τις λεγόμενες συνάψεις (synapses). Ο βαθμός αλληλεπίδρασης είναι διαφορετικός για κάθε ζεύγος νευρώνων και καθορίζεται από τα λεγόμενα συνοπτικά βάρη (synaptic weights).

Αποδεικνύεται [3] ότι ένα νευρωνικό δίκτυο με πεπερασμένο αριθμό νευρώνων μπορούν να προσεγγίσουν οποιαδήποτε συνάρτηση!



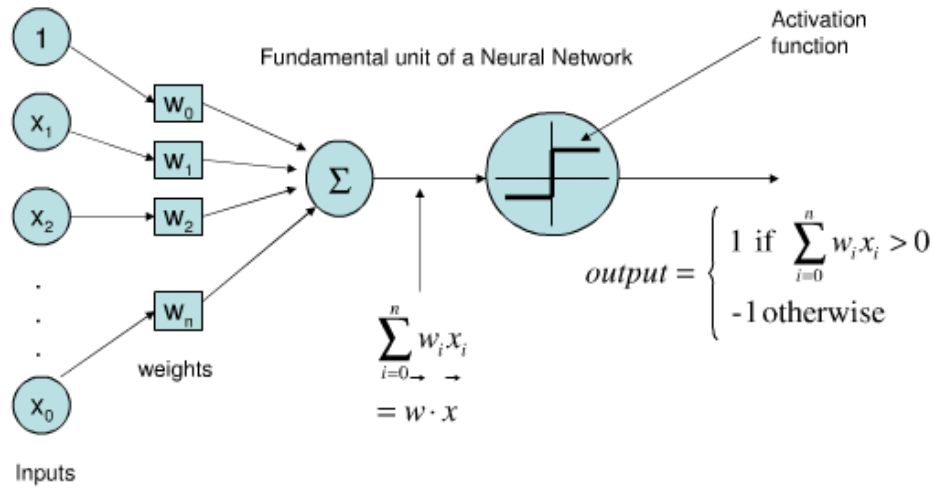
Εικόνα 3.2. Τρισδιάστατη αναπαράσταση ενός νευρωνικού δικτύου ικανό να αναγνωρίσει χειρόγραφα ψηφία

3.3.1 Το απλό perceptron

Το perceptron είναι η πιο απλή μορφή νευρωνικού δικτύου ενός επιπέδου. Ο αλγόριθμος αυτός δέχεται ως είσοδο κάποιες τιμές x_i και έχοντας γνωστές τις εξόδους y_i προσπαθεί μέσω της διαδικασίας της εκπαίδευσης να υπολογίσει τα βάρη (weights) τα οποία θα οδηγήσουν νέα δεδομένα x_i τα οποία ο αλγόριθμος δεν έχει δει ξανά, σε σωστές προβλέψεις.

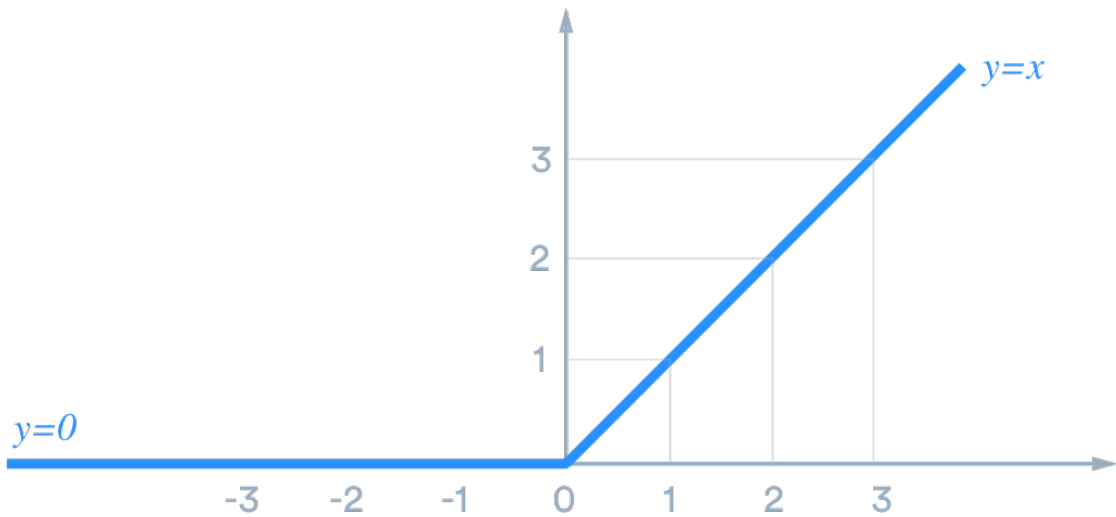
Τα βάρη αρχικοποιούνται τυχαία (συνήθως με βάση κάποια κατανομή) και με τη χρήση του αλγόριθμου backpropagation ανανεώνονται προς τη κατεύθυνση όπου ελαχιστοποιείται η απόσταση της πραγματική λύσης από τη λύση που το δίκτυο παράγει τη δεδομένη χρονική στιγμή.

Η έξοδος του δικτύου υπολογίζεται μέσω της διαδικασίας που φαίνεται στην εικόνα 3.3. Οι τιμές εισόδου πολλαπλασιάζονται με τα βάρη με το άθροισμα αυτών των γινομένων να οδηγείται σε μία συνάρτηση ενεργοποίησης (activation function). Η χρησιμότητα της συνάρτησης αυτής είναι να δώσει τη δυνατότητα στον αλγόριθμο να μπορεί να προσεγγίσει και μη γραμμικές συναρτήσεις.



Εικόνα 3.3. Αναπαράσταση του perceptron

Οι συναρτήσεις ενεργοποίησης έχουν συνήθως τη μορφή σιγμοειδούς συνάρτησης αν και τον τελευταίο καιρό η συνάρτηση όπου δίνει τα καλύτερα αποτελέσματα είναι η ReLu, η οποία παρουσιάζεται γραφικά στην εικόνα 3.4.



Εικόνα 3.4. Η συνάρτηση ενεργοποίησης ReLu.

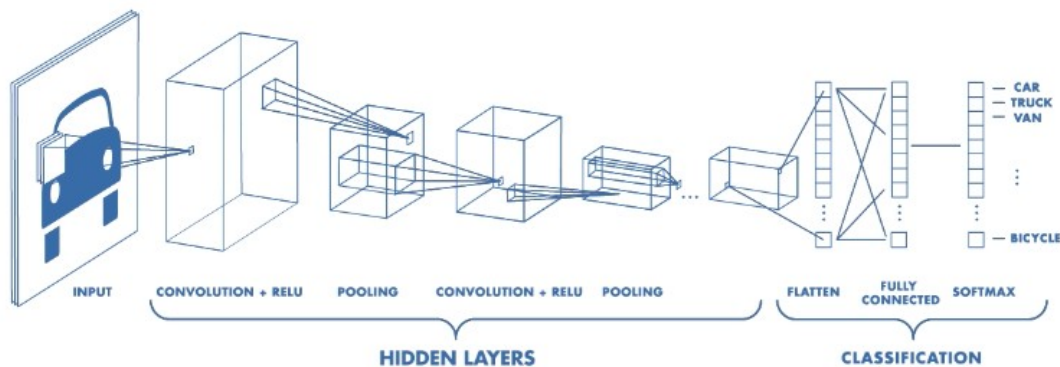
Κατά τη διαδικασία της εκπαίδευσης ο τρόπος με τον οποίο θα ανανεώνονται τα βάρη εξαρτάται από τη συνάρτηση loss που έχει επιλεγεί, δηλαδή τη συνάρτηση που δείχνει το πόσο η λύση που έχουμε τώρα απέχει από τη πραγματική.

3.3.2 Deep Learning

Όταν ενωθούν πολλά perceptron τέτοια ώστε να δημιουργούν μία αρχιτεκτονική, τότε δημιουργείται ένα νευρωνικό δίκτυο νευρώνων και όταν αυτό το δίκτυο είναι μεγάλο από την άποψη των επιπέδων (layers) ονομάζεται deep neural network. Όσο αυξάνονται οι νευρώνες και τα επίπεδα τόσο αυξάνονται οι και παράμετροι (βάρη) που πρέπει να υπολογιστούν πράγμα που σημαίνει ότι τόσο πιο πολύπλοκες συναρτήσεις μπορούν να προσεγγιστούν. Αυτό βέβαια δε σημαίνει ότι μπορούμε αν αυξήσουμε τη πολυπλοκότητα να έχουμε βελτίωση στο αποτέλεσμα, μιας και αυτό έχει άμεση σχέση με το πρόβλημα.

3.3.3 Convolutional Neural Networks (CNN)

Τα CNN είναι ένα είδος νευρωνικών δικτύων τα οποία χρησιμοποιούνται κυρίως για οπτικά δεδομένα όπως εικόνες. Το δίκτυο αυτό αποτελείται από ένα απλό fully connected νευρωνικό το οποίο όμως έπεται από το convolution κομμάτι το οποίο είναι περίπου ένα είδος προεπεξεργασίας των δεδομένων. Αυτό σημαίνει ότι το δίκτυο μαθαίνει φίλτρα τα οποία είναι στη πραγματικότητα πολύ μικρά νευρωνικά δίκτυα. Σε κάθε πέρασμα το κάθε φίλτρο διασχίζει την εικόνα υπολογίζοντας το γινόμενο ανάμεσα στα στοιχεία του φίλτρου και της εισόδου. Σαν αποτέλεσμα, το δίκτυο μαθαίνει φίλτρα που ενεργοποιούνται (παίρνουν μεγάλες τιμές) όταν εντοπίζουν κάτι που μοιάζει με αυτό που έχουν μάθει τη δεδομένη χρονική στιγμή όπου κι αν βρίσκεται αυτό.



Εικόνα 3.5. Αναπαράσταση ενός CNN

4.

ΣΥΛΛΟΓΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Το πρώτο στάδιο για την επίλυση του προβλήματος είναι η συλλογή και η επεξεργασία των δεδομένων. Μιας και αναφερόμαστε σε ένα πρόβλημα εποπτευόμενης μάθησης είναι προφανές ότι χρειαζόμαστε κάποια “πραγματικά” δεδομένα τα οποία ο αλγόριθμος μηχανικής μάθησης θα χρησιμοποιήσει στη συνέχεια για να μπορέσει να προσεγγίσει τη συνάρτηση που ερευνάται, δηλαδή τη συνάρτηση της οδήγησης στο συγκεκριμένο περιβάλλον προσομοίωσης. Τα δεδομένα αυτά προέρχονται από την αλληλεπίδραση του γράφοντα με το παιχνίδι και εκφράζουν τις πραγματικές ή τις σωστές κινήσεις που χρειάζεται να γίνουν για να μπορέσει το όχημα να περιηγηθεί στην εικονική πίστα. Τα δεδομένα συλλέχθηκαν από μία συγκεκριμένη πίστα για λόγους ευκολίας χωρίς βλάβη της γενικότητας. Το περιβάλλον που χρησιμοποιήθηκε ήταν το F12017 της εταιρίας codemasters.

Τα δεδομένα που υπάρχουν στη διάθεση του προγραμματιστή είναι ελάχιστα μιας και πρόκειται για έναν εμπορικό εξομοιωτή που έχει σκοπό να προσφέρει μία όσο το δυνατό πιο ρεαλιστική εμπειρία στους παίκτες. Τα δεδομένα που κρίθηκαν εν τέλει χρήσιμα για την εκπαίδευση του αλγορίθμου ήταν η εικόνα του παιχνιδιού καθώς και κάποια στοιχεία τηλεμετρίας που περιγράφουν τη γεωγραφική θέση του οχήματος και την ταχύτητά του.

4.1 Δεδομένα ροής βίντεο

Η εικόνα του παιχνιδιού αποτελεί το κύριο μέρος των δεδομένων των οποίων συλλέχθηκαν. Στην ουσία αποθηκεύονται τα αριθμητικά pixels με τιμές από 0 έως 255 στη κωδικοποίηση χρώματος RGB και διαστάσεις 180x250x3. Προφανώς, για να γίνει εφικτή η σύλληψη ενός μεγάλου μέρους της οθόνης του παιχνιδιού σε αυτές τις διαστάσεις, το παιχνίδι εκτελείται σε μικρό παράθυρο και όχι σε ολόκληρη την οθόνη όπως συνηθίζεται.

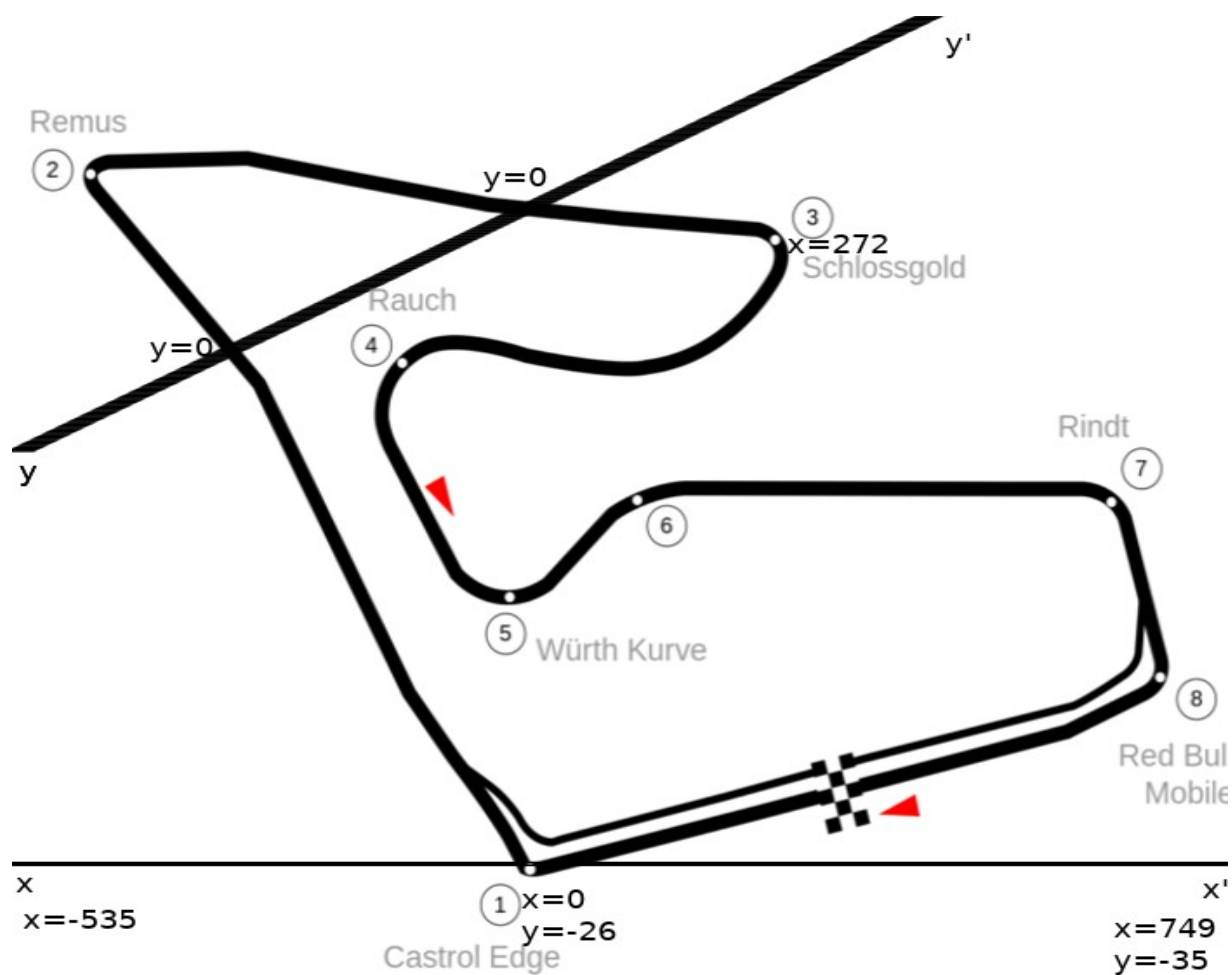


Εικόνα 4.1. Μία τυχαία εικόνα του dataset

4.2 Δεδομένα τηλεμετρίας

Το περιβάλλον προσομοίωσης διαθέτει τη λειτουργία εξαγωγής κάποιων δεδομένων-μεταβλητών τα οποία φαίνονται χρήσιμα για την επίλυση του προβλήματος. Η ζωντανή λήψη αυτών των δεδομένων γίνεται με τη χρήση UDP πακέτων μέσω sockets, πράγμα το οποίο υλοποιείται σχετικά εύκολα στη python. Τα διαθέσιμα δεδομένα τηλεμετρίας είναι πολλά και περιλαμβάνουν μεταξύ άλλων τη ταχύτητα του οχήματος, τη θέση του κάθε τροχού, τη φθορά των ελαστικών, τη γωνιακή ταχύτητα προς κάθε άξονα, τις δυνάμεις G που δέχεται ο οδηγός κ.α. Μετά από δοκιμές τα δεδομένα που επιλέχθηκαν ήταν η σχετική θέση του οχήματος η οποία αναπαρίσταται επαρκώς από τον x και από τον y άξονα καθώς και η ταχύτητα. Όπως φαίνεται και στην εικόνα 4.2, δεν είναι αναγκαία η τρίτη διάσταση (z) για την ακριβή τοποθέτηση του οχήματος μέσα στον χώρο.

Όπως φαίνεται και στο script [A1] τα δεδομένα τηλεμετρίας έρχονται κάθε χρονική στιγμή μαζί με τα δεδομένα τις εικόνες. Προφανώς λόγω υπερφόρτωσης του συστήματος, μαζί με τη διαδικασία συλλογής δεδομένων τρέχει και το περιβάλλον προσομοίωσης, ο αριθμός των frames ανά δευτερόλεπτο είναι μειωμένος, όχι όμως σε σημείο που να γίνεται ύποπτος για τυχόν προβλήματα.



Εικόνα 4.2. Χάρτης της εξεταζόμενης διαδρομής. Στο χάρτη εμφανίζεται η τοποθέτηση των αξόνων x και y οι οποίοι όπως φαίνεται, δεν είναι ορθοκανονικοί. Η συγκεκριμένη διαδρομή βρίσκεται στην Αυστρία.

4.3 Δεδομένα εισόδου

Η συλλογή των δεδομένων ολοκληρώνεται με τα δεδομένα εισόδου του παίχτη. Τα δεδομένα αυτά είναι τα δεδομένα τα οποία ο αλγόριθμος μάθησης θα πρέπει να μπορεί να προβλέψει, δηλαδή η ποσότητα πίεσης για τα πετάλ του γκαζιού (Γ) και της πέδησης (Σ) καθώς και η ποσότητα στριψίματος του τιμονιού (Σ). Και οι τρεις αυτές τιμές βρίσκονται στο διάστημα $[-1,1]$.

Η αποθήκευση αυτών των δεδομένων γίνεται με τη χρήση της βιβλιοθήκης pygame. Επειδή και οι τρεις αυτές τιμές πρέπει να διαβάζονται από το script [A1] όσο το δυνατό πιο κοντά χρονικά γίνεται, η διαδικασία παραλληλοποιείται με τη βοήθεια τριών scripts – ένα για κάθε δεδομένο (παράδειγμα για τη ποσότητα στριψίματος στο [A2]) – τα οποία τρέχουν ταυτόχρονα και αποθηκεύουν την είσοδο του παίχτη σε ένα προσωρινό αρχείο από το οποίο το script [A1] διαβάζει τη τιμή.

Η συλλογή των δεδομένων Γ και Σ έγιναν με τη χρήση του αντίστοιχου συστήματος της τιμονιέρας Logitech Driving Force gt, ενώ η συλλογή των δεδομένων Σ με την εξομοίωση ενός χειριστηρίου μέσω ποντικιού μιας και προέκυψαν πολλά θέματα με την αντιστοίχιση των τιμών εισόδου του παίχτη και των τιμών εξόδου του αλγορίθμου. Ο τρόπος αυτός, μέσω της χρήσης της εφαρμογής njoy εξασφάλισε μία ένα προς ένα αντιστοιχία ανάμεσα στις τιμές αυτές.

4.4 Τρόπος αποθήκευσης δεδομένων

Τα δεδομένα που τελικά συλλέχθηκαν για κάθε χρονική στιγμή της διαδικασίας συλλογής δεδομένων ήταν: μία εικόνα $180 \times 250 \times 3$, η τιμές $x, y, speed$ της τηλεμετρίας και οι τιμές Γ, Π, Σ , δηλαδή 90006 τιμές που μάλιστα ανήκουν σε διαφορετικά διαστήματα.

Αποθηκεύτηκαν συνολικά 38000 χρονικές στιγμές, τα δεδομένα των οποίων χωρίστηκαν σε 76 αρχεία όπου το καθένα περιείχε 500 τέτοιες χρονικές στιγμές.

Κάθε χρονική στιγμή αποτελείται από τους πίνακες. Ο πρώτος πίνακας περιέχει την εικόνα μεγέθους $180 \times 250 \times 3$, ο δεύτερος περιέχει τα δεδομένα τηλεμετρίας ενώ ο τρίτος περιέχει τα πραγματικά $x, y, speed$.

Με τη χρήση αυτής της δομής επιτυγχάνεται εύκολη διαχείριση και επεξεργασία των δεδομένων κάτι που είναι ιδιαίτερα σημαντικό για την εκπαίδευση του αλγορίθμου.

Ο κώδικας του python script το οποίο διαβάζει τα δεδομένα και τα αποθηκεύει με τον τρόπο που αναλύθηκε βρίσκεται στο Παράθεμα [A1].

4.5 Προεπεξεργασία δεδομένων

Μετά από τη παραπάνω ανάλυση έχουμε καταλήξει στο ότι το μοντέλο θα δέχεται σαν είσοδο την εικόνα του παιχνιδιού και τις τιμές $x, y, speed$ και θα πρέπει να δίνει σαν έξοδο τις τιμές Γ, Π και Σ , όπως αυτά ορίστηκαν στις προηγούμενες παραγράφους. Τα δεδομένα αυτά και ιδιαίτερα τα δεδομένα εισόδου δε γίνεται να δοθούν στον αλγόριθμο με τη παρούσα μορφή τους μιας και τα δεδομένα αυτά (τα οποία μπορούν να θεωρηθούν τυχαίες μεταβλητές) παίρνουν τιμές σε αρκετά διαφορετικά διαστήματα. Συγκεκριμένα τα $pixels$ της εικόνας παίρνουν τιμές στο διάστημα $[0-255]$ (μόνο ακέραιες τιμές), τα x παίρνουν δεκαδικές τιμές περίπου στο διάστημα $[-535, 749]$, τα y περίπου στο $[-42, 20]$ (επίσης δεκαδικές τιμές) και τέλος το $speed$ περίπου στο $[19-80]$, όπως φαίνεται και στην εικόνα 4.4.

Στη πράξη είναι εφικτή η χρήση αυτούσιων αυτών των δεδομένων σαν είσοδο στο νευρωνικό δίκτυο, παρ' όλα αυτά κάτι τέτοιο θα οδηγούσε σε σοβαρά προβλήματα κατά τη διαδικασία της εκπαίδευσης. Ας υποθέσουμε ότι χρησιμοποιούμε σαν συνάρτηση ενεργοποίησης την sigmoid η οποία δίνεται από τον τύπο:

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

Η συνάρτηση αυτή εφαρμόζεται σε κάθε νευρώνα και αυτό που πετυχαίνει είναι να μεταφέρει τις τιμές ενεργοποίησης στο διάστημα $[0,1]$. Κατά τη διαδικασία της εκπαίδευσης υπολογίζονται κάποιες μερικοί παράγωγοι της συνάρτησης ενεργοποίησης. Στο παρακάτω κομμάτι κώδικα φαίνεται ένα παράδειγμα για το πως εκπαιδεύεται ένα μοντέλο.

```
1.def sigmoid(x,deriv=False):
2.    if (deriv==True):
3.        return x*(1-x)
4.
5.    return 1/(1+np.exp(-x))
```

Κώδικας 4.1. Η συνάρτηση sigmoid και η παράγωγός της

```
1. #training step
2. for j in range(60000):
3.
4.     l0 = X
5.
6.     l1 = nonlin(np.dot(l0, syn0))
7.     l2 = nonlin(np.dot(l1, syn1))
8.
9.     l2_error = y - l2
```

```

10.
11.     if (j % 10000) == 0:
12.         print ("Error:" + str(np.mean(np.abs(l2_error)))) )
13.
14.     l2_delta = l2_error*sigmoid(l2,deriv=True)
15.
16.     l1_error = np.dot(l2_delta,syn1.T) # np.dot(l2_delta*syn1.T)
17.     l1_delta = l1_error * sigmoid(l1,deriv=True)
18.
19.     #update weights (gradient descent)
20.     syn1 += np.dot(l1.T,l2_delta)
21.
22.     syn0 += np.dot(l0.T,l1_delta)

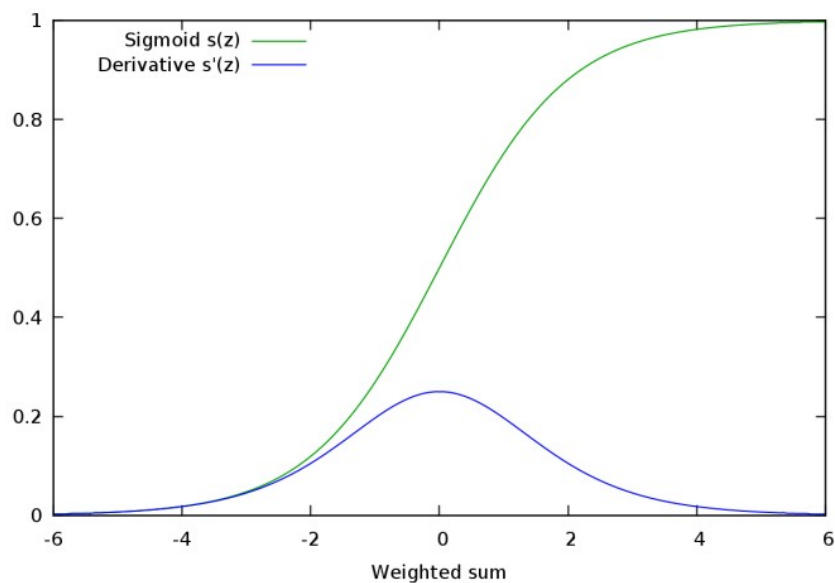
```

Κώδικας 4.2. Παράδειγμα εκπαίδευσης ενός μικρού νευρωνικού δικτύου

Το δίκτυο που φαίνεται εδώ είναι ένα απλό feed forward νευρωνικό δίκτυο τριών επιπέδων. Από τη συνάρτηση sigmoid που έχει οριστεί, επιστρέφεται η τιμή της συνάρτησης για κάποιο x καθώς και η τιμή της παραγώγου. Στις γραμμές 4 με 7 του training εφαρμόζεται το forward pass όπου υπολογίζεται το εσωτερικό γινόμενο του προηγούμενου επιπέδου με τις συνάψεις με τη τιμή αυτή να περνάει στη συνέχεια από τη συνάρτηση ενεργοποίησης, όπου στη συγκεκριμένη περίπτωση είναι η sigmoid.

Στη συνέχεια (γραμμή 9 υπολογίζεται η διαφορά ανάμεσα στην έξοδο που βγάζει το δίκτυο και τη πραγματική έξοδο (σφάλμα). Το σφάλμα αυτό πρέπει να διαδοθεί προς τα πίσω έτσι ώστε να συναπτικά βάρη να αλλάξουν τιμές έτσι ώστε να προσεγγίσουν καλύτερα την έξοδο y . Η διαδικασία αυτή φαίνεται στις γραμμές 14-22. Εκεί το σφάλμα του τελευταίου επιπέδου, το οποίο έχει υπολογιστεί, πολλαπλασιάζεται με την παράγωγο της συνάρτησης ενεργοποίησης κάτι που δίνει ως αποτέλεσμα delta, δηλαδή το κατά πόσο πρέπει να αλλάξουν τα συναπτικά βάρη και προς ποιά κατεύθυνση. Η διαδικασία αυτή προχωράει από δεξιά προς τα αριστερά για κάθε επίπεδο του δικτύου και στο τέλος οι τιμές των συναπτικών βαρών ανανεώνονται με βάση τα deltas που υπολογίστηκαν.

Από τη παραπάνω ανάλυση γίνεται εμφανής η διαδικασία της εκπαίδευσης, καθορίζεται άμεσα από τη συνάρτηση ενεργοποίησης που έχει επιλεγεί, αλλά και από τη παράγωγό της. Στο παρακάτω σχήμα φαίνεται η συνάρτηση sigmoid και η παράγωγός της.

Εικόνα
4.3. Η

συνάρτηση ενεργοποίησης sigmoid και η παράγωγός της σχηματικά.

Είναι φανερό ότι η παράγωγος μηδενίζει για μεγάλες τιμές της συνάρτησης sigmoid δηλαδή για τιμές κοντά στο 1 και μεγαλύτερες και τιμές κοντά στο 0 και μικρότερες.

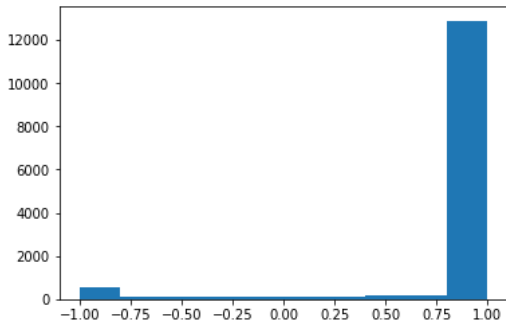
Επιστρέφοντας στο πρόβλημα της κανονικοποίησης, εάν οι τιμές εισόδου είναι μεγάλες (όπως για παράδειγμα τα pixels που παίρνουν τιμές στο $[0, 255]$), τότε κατά τη διαδικασία του forward pass, η τιμή της sigmoid θα είναι μηδέν. Αυτό θα κάνει το δίκτυο να μη μπορεί να μάθει τίποτα. Από την άλλη εάν μεταφέρουμε τα τις τιμές σε ένα μικρότερο διάστημα τότε η εκπαίδευση θα είναι δυνατή. Το πρόβλημα αυτό ονομάζεται Vanishing gradient problem

Ένα ακόμα πρόβλημα με τα συγκεκριμένα δεδομένα είναι ότι έχουν μεταξύ τους μεγάλη διαφορά στις τιμές. Αυτό σημαίνει ότι το δίκτυο θα δίνει πιο πολύ “αξία” στα δεδομένα με μεγαλύτερες τιμές και σχεδόν θα αγνοεί αυτά με τις μικρότερες, κάτι το οποίο δεν είναι επιθυμητό.

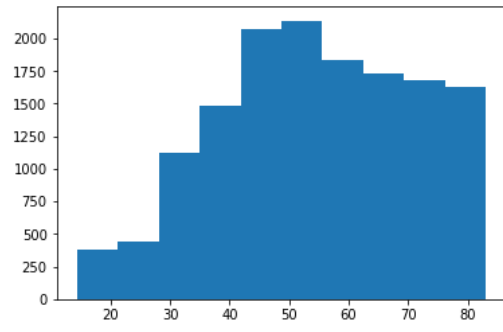
Μία άλλη συνάρτηση ενεργοποίησης είναι η ReLu, η οποία χρησιμοποιείται στις περισσότερες εφαρμογές τον τελευταίο καιρό. Η συνάρτηση έχει τη παρακάτω απλή μορφή:

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

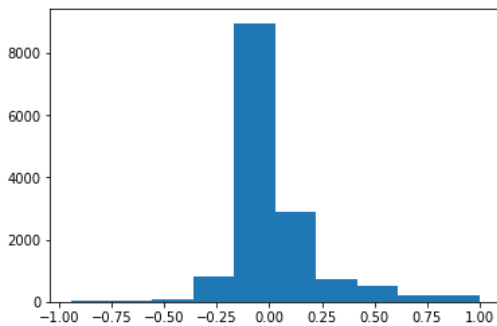
Σε αυτή τη συνάρτηση ενεργοποίησης φαίνεται να μην εμφανίζεται το παραπάνω πρόβλημα μιας και η συνάρτηση αυτή δεν μειώνει το εύρος των θετικών τιμών. Το πρόβλημα εδώ όμως θα ήταν ότι τα συναπτικά βάρη θα έπαιρναν μεγάλες τιμές (μιας και η ReLu το επιτρέπει). Αυτό μπορεί να δημιουργήσει αρκετά προβλήματα μιας και όταν τα βάρη είναι μεγάλα τότε αυτά είναι πολύ ευαίσθητα σε μικρούς θορύβους που πιθανόν προέρχονται από τα αρχικά δεδομένα είτε από τα σφάλματα που δημιουργούνται από τους υπολογισμούς. Για το πρόβλημα αυτό χρησιμοποιούνται και πιο εξειδικευμένες τεχνικές, όπως το L2 regularization και το dropout, αλλά η κανονικοποίηση των δεδομένων παραμένει αναγκαία.



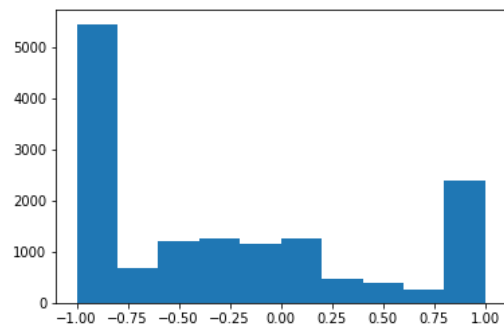
4.4Α Πέδηση



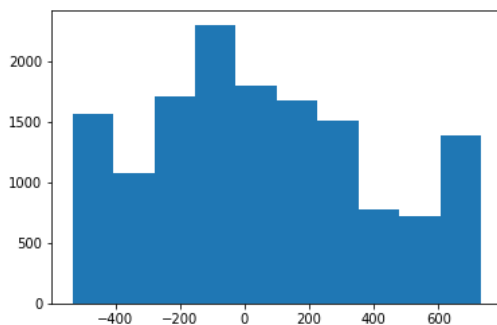
4.4Β Ταχύτητα



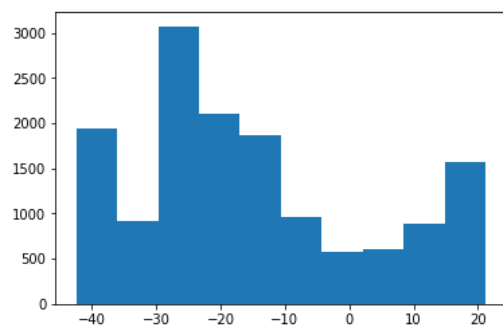
4.4Γ Στρίψιμο



4.4Δ Γκάζι



4.4Ε Άξονας X



4.4Ζ Άξονας Y

Εικόνα 4.4. Ιστογράμματα κατανομής δεδομένων

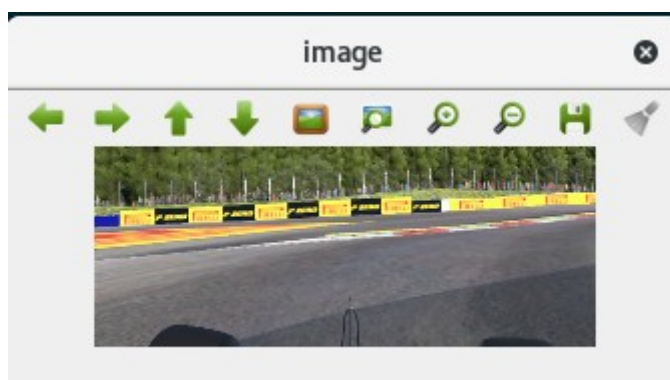
Στα ιστογράμματα της εικόνας 4.4 παρουσιάζονται ποιοτικά δεδομένα ενός αντιπροσωπευτικού μέρους του dataset. Παρατηρείται η διαφορετική διασπορά ανάμεσα στα δεδομένα, καθώς και κάποιες αναμενόμενες ιδιαιτερότητες όπως ότι το τιμόνι τις περισσότερες φορές βρίσκεται στο μέσο (δηλαδή υπάρχει κίνηση προς τα μπροστά) και τις περισσότερες φορές το γκάζι είναι τέρμα πατημένο, ενώ στη πέδηση συμβαίνει το αντίστροφο.

Σημειώνεται ότι στο γκάζι και στη πέδηση η τιμή -1 δηλώνει ότι ο αντίστοιχος μοχλός είναι τέρμα πατημένος και το 1 το αντίθετο.

4.5.1 Επεξεργασία εικόνας

4.5.1.A Περιοχή ενδιαφέροντος

Στην εικόνα 4.1 φαίνεται η αρχική εικόνα η οποία λαμβάνεται από το περιβάλλον προσομοίωσης. Η εικόνα αυτή μοιάζει να έχει περισσότερη πληροφορία απ' όση φαίνεται να χρειάζεται για να μπορέσει το μοντέλο να προβλέψει τις τιμές Γ,Π,Σ. Για το λόγο αυτό κάνοντας έναν απλό μετασχηματισμό χρησιμοποιώντας το εργαλείο `opencv`, γίνεται επιλογή της περιοχής ενδιαφέροντος η οποία φαίνεται στην εικόνα 4.5



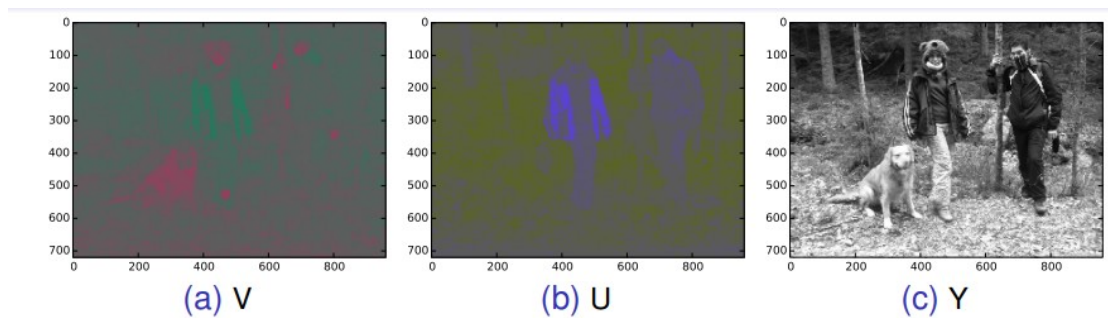
Εικόνα 4.5 Περιοχή ενδιαφέροντος

4.5.1.B Κωδικοποίηση χρωμάτων

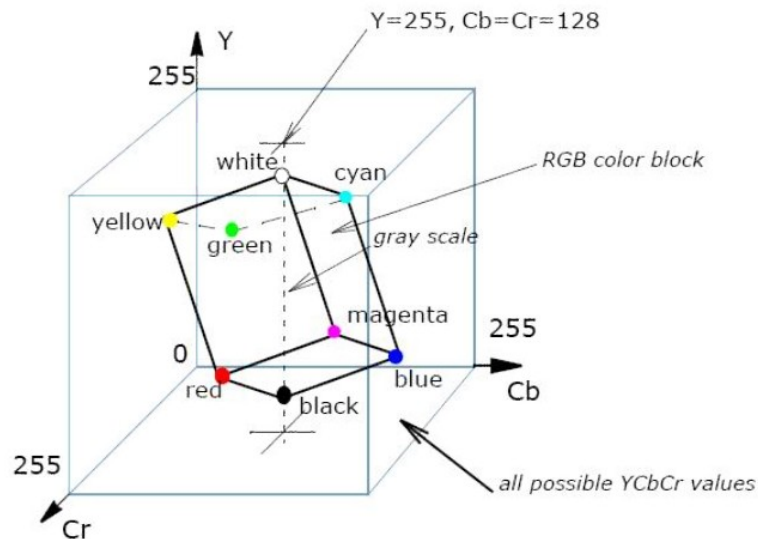
Με τη χρήση εμπειρικής παρατήρησης, φάνηκε ότι η κωδικοποίηση χρωμάτων YUV μπορεί να δώσει οπτικά μία καλύτερη διάκριση του δρόμου με τις περιοχές που είναι εκτός δρόμου (γρασίδι

κτλ). Πέρα από την εμπειρική παρατήρηση μέσω πειραμάτων έχει προκύψει ότι ο χωρισμός του καναλιού φωτεινότητας από του καναλιού του χρώματος μπορούν να βελτιώσουν τη διαδικασία της μάθησης. [4]

Η αρχική ιδέα γύρω από τη κωδικοποίηση αυτή (όπως περιγράφεται και στην αναφορά [4]) είναι ο χωρισμός των καναλιών Y και UV, η εξαγωγή χαρακτηριστικών από αυτά και έπειτα η ένωσή τους με το υπόλοιπο νευρωνικό δίκτυο. Στη πράξη όμως υλοποιήθηκε κάτι παρόμοιο όπως αναφέρεται στο κεφάλαιο 5, παρ' όλα αυτά η κωδικοποίηση αυτή διατηρήθηκε μιας και διαισθητικά και οπτικά φαίνεται ότι διαχωρίζεται καλύτερα ο δρόμος από τις περιοχές εκτός δρόμου, πράγμα που πιθανόν να είναι σημαντικό για την αποτελεσματικότητα του αλγορίθμου.



Εικόνα 4.6 Παράδειγμα καναλιών YUV



Εικόνα 4.7 Χώρος χρωμάτων YUV

4.5.1.C Κανονικοποίηση

Μία εικόνα, η οποία αποτελεί μία χρονική στιγμή του βίντεο, περιέχει 100x250x3 δηλαδή 75K τιμές. Για να έρθουν τα δεδομένα αυτά σε μία βέλτιστη μορφή για το νευρωνικό δίκτυο πρέπει να κανονικοποιηθούν. Η διαδικασία κανονικοποίησης που επιλέγεται είναι η κανονικοποίηση κατά την οποία αφαιρείται η μέση τιμή και έπειτα η ποσότητα αυτή διαιρείται με τη τυπική απόκλιση του δείγματος, όπως φαίνεται στη παρακάτω σχέση.

$$x' = \frac{x - \text{mean}(X)}{\text{stddev}(X)}$$

Η σχέση αυτή όμως δε δίνει όλη τη πληροφορία για το πως πρέπει να γίνει η κανονικοποίηση μιας και μία εικόνα αποτελείται από 3 κανάλια, συνεπώς υπάρχουν διάφοροι χώροι οι οποίοι μπορούν να χρησιμοποιηθούν σαν περιοχές ενδιαφέροντος στις οποίες θα υπολογιστεί η μέση τιμή και η τυπική απόκλιση. Ενδεικτικά μία θεώρηση θα ήταν ο υπολογισμός των μετρικών αυτών για κάθε pixel, δηλαδή η εύρεση του μέσου για κάθε θέση όπου εμφανίζονται pixels. Δηλαδή θα υπολογιζόταν η μέση τιμή και η τυπική απόκλιση για το pixel που βρίσκεται στη θέση (i,j,k) και αυτή η τιμή θα χρησιμοποιούνταν για τη κανονικοποίηση ενός στοιχείου το οποίο θα βρισκόταν στην ίδια θέση. Ο υπολογισμός αυτός φαίνεται στη παρακάτω σχέση για τη μέση τιμή.

$$\forall i, j, k \rightarrow \mu_{i,j,k} = \frac{\sum_N \text{img}(i, j, k)}{kN} \quad (1)$$

με τη τυπική απόκλιση να προκύπτει στην συνέχεια από τον ορισμό της.

$$s = \sqrt{\sum_{i=1}^N \frac{x_i - \mu^2}{Nk - 1}} \quad (1.1)$$

όπου N το πλήθος των εικόνων του dataset και k το πλήθος των καναλιών.

Μία πιο απλή εναλλακτική προσέγγιση είναι ο υπολογισμός των ζητούμενων στατιστικών τιμών με τη χρήση όλων των pixel της εικόνας, δηλαδή στην ουσία υπολογίζοντας τη μέση τιμή και τη τυπική απόκλιση όλου του dataset.

$$\mu = \frac{\sum_N \text{img}(i, j, k)}{N} \quad (2)$$

Ένας τρίτος τρόπος, ο οποίος προτιμήθηκε στη συγκεκριμένη υλοποίηση, είναι ο υπολογισμός των μέσων τιμών και τυπικών αποκλίσεων όλου του dataset για κάθε κανάλι

ξεχωριστά.

$$\forall k \rightarrow \mu_k = \frac{\sum_N img(i, j, k)}{kN} \quad (3)$$

Με τη μέθοδο αυτή καταλήγουμε σε τρεις μέσες τιμές (μία για κάθε κανάλι) και τρεις τυπικές αποκλίσεις.

Σημειώνεται ότι η κανονικοποίηση αυτή εφαρμόζεται σε δεδομένα μετά από τον μετασχηματισμό τους σε YUV.

Η κανονικοποίηση αυτή μεταφέρει τα δεδομένα κάθε καναλιού σε μία προσέγγιση της κανονικής κατανομής, δηλαδή μίας κατανομής με μέση τιμή μηδέν και τυπική απόκλιση ένα. Το πλεονέκτημα αυτής της μεθόδου είναι ότι εφόσον το κάθε κανάλι θα ακολουθεί μία κανονική κατανομή, τότε τα συναπτικά βάρη που αφορούν ένα από αυτά τα κανάλια δε θα μπορεί να πάρει άδικα πολύ μεγάλες τιμές, άρα και πολύ μεγάλη αξία. Αν για παράδειγμα χρησιμοποιούταν ο τρόπος (2) τότε στο δίκτυο οι τιμές ενός καναλιού θα μπορούσαν να έχουν πολύ μεγαλύτερη διασπορά από τις τιμές ενός άλλου, πράγμα που θα μπορούσε να οδηγούσε σε μεγάλη αύξηση των βαρών που το αφορούν.

	Κανάλι 1	Κανάλι 2	Κανάλι 3
Μέση τιμή	96.27572741493113	129.70090178693755	123.5361440900808
Τυπική απόκλιση	44.434308166340124	10.913365277202386	11.608120967956518

Πίνακας 1. Υπολογισθέντες μέσες τιμές και τυπικές αποκλίσεις κάθε καναλιού για το dataset το οποίο εξετάστηκε.

4.5.2 Επεξεργασία δεδομένων τηλεμετρίας

Τα δεδομένα τηλεμετρίας τα οποία πέρασαν από το στάδιο της κανονικοποίησης ήταν τα τηλεμετρίας εισόδου, δηλαδή τα: x,y,speed. Τα δεδομένα εξόδου (Γ,Π,Σ) βρίσκονται και τα τρία εξ' αρχής στο διάστημα [-1,1], συνεπώς δεν υπάρχει λόγος να κανονικοποιηθούν.

Η διαδικασία με την οποία γίνεται η κανονικοποίηση για τα δεδομένα αυτά είναι όμοια με τη διαδικασία όπου ακολουθήθηκε στη κανονικοποίηση των εικόνων, δηλαδή για κάθε νέα τιμή αφαιρείται η μέση τιμή του dataset και η διαφορά αυτή διαιρείται με τη τυπική απόκλιση, εξασφαλίζοντας έτσι δεδομένα τα οποία προσεγγίζουν τα κανονικά.

$$\mu = \frac{\sum_{i=0}^N x_i}{N} \quad (4)$$

Η διαδικασία αυτή είναι όμοια για κάθε στοιχείο $x, y, speed$. Οι τιμές που υπολογίστηκαν φαίνονται στον παρακάτω πίνακα.

	x	y	$speed$
Μέση τιμή	53.431771465267005	-15.4245473103691	54.935332703891554
Τυπική απόκλιση	339.6735376759244	18.144510216494556	16.24000406834198

Πίνακας2. Υπολογισθέντες μέσες τιμές και τυπικές αποκλίσεις κάθε στοιχείου τηλεμετρίας για το dataset το οποίο εξετάστηκε.

4.5.3 Συνδυασμός εικόνας και τηλεμετρίας

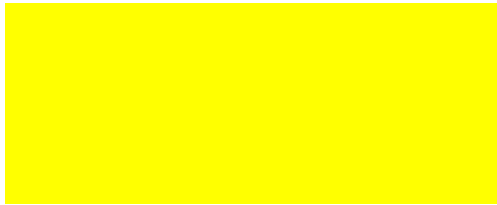
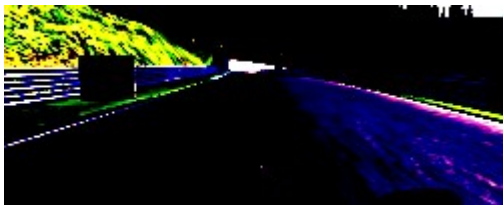
Η συνύπαρξη δομικά και ποσοτικά διαφορετικών δεδομένων στο επίπεδο εισόδου ενός νευρωνικού δικτύου δεν είναι κάτι τετριμμένο, μιας και το μέγεθος ή και η σειρά των δεδομένων μπορεί να επηρεάσουν δραστικά το αποτέλεσμα και τη διαδικασία της μάθησης

Στο συγκεκριμένο πρόβλημα πρέπει να συνδυαστούν δεδομένα εικόνας με κάποιες επιπλέον λίγες τιμές. Συγκεκριμένα τα δεδομένα της μίας εικόνας η οποία δίνεται σαν είσοδο στον αλγόριθμο μία δεδομένη χρονική στιγμή είναι $100 \times 250 \times 3 = 75K$ στοιχεία (float) ενώ τα στοιχεία της τηλεμετρίας είναι 3. Φαίνεται ότι εάν προστεθούν με κάποιο τρόπο αυτές οι τρεις τιμές μαζί με τις υπόλοιπες 75K τότε δε θα επηρεάσουν καθόλου την έξοδο του αλγορίθμου. Υπάρχουν δύο τρόποι για να αντιμετωπιστεί αυτό το πρόβλημα. Ο πρώτος είναι η εισαγωγή των τριών αυτών τιμών όχι στο αρχικό επίπεδο, αλλά σε κάποιο από τα τελευταία επίπεδα όπου όλη η πληροφορία από τα 75K στοιχεία της εικόνας θα έχει συμπιεστεί σε λίγους νευρώνες. Ο δεύτερος τρόπος, αυτός ο οποίος εφαρμόστηκε, είναι η δημιουργία ενός διαφορετικού καναλιού για κάθε ένα από τα τρία δεδομένα της τηλεμετρίας. Κάθε κανάλι θα έχει μέγεθος όσο και η εικόνα (δηλαδή 100×250) και θα επαναλαμβάνει τη τιμή του στοιχείου για το οποίο αναφέρεται σε όλο του το εύρος. Η μέθοδος αυτή προφανώς δεν είναι οικονομική μιας και η είσοδος θα έχει τώρα $100 \times 250 \times 6 = 150K$ στοιχεία (δηλαδή θα είναι διπλάσια), αλλά προσφέρει τη δυνατότητα οπτικοποίησης η οποία μπορεί να φανεί ιδιαίτερα χρήσιμη για τη κατανόηση της συμπεριφοράς του μοντέλου και του λόγου για το οποίο παίρνει αποφάσεις. Επιπλέον από τον ορισμό αυτό γίνεται σαφές ότι τα στοιχεία (εικόνα) και $(x, y, speed)$ θα έχουν την ίδια αξία, δε θα υπάρχει δηλαδή κάποια προκατάληψη προερχόμενη από την επεξεργασία.

Η οπτικοποίηση γίνεται εύκολα μιας και στην ουσία κάθε πλειάδα δεδομένων τηλεμετρίας $(x, y, speed)$ εκφράζεται από μία έγχρωμη RGB εικόνα η οποία μπορεί να βοηθήσει στη κατανόηση των στοιχείων για τα οποία δείχνει ενδιαφέρον ο αλγόριθμος ιδιαίτερα σε περιπτώσεις overfitting.

Ακολουθούν δύο παραδείγματα από το πως μοιάζει η τελική είσοδος του αλγορίθμου μετά το

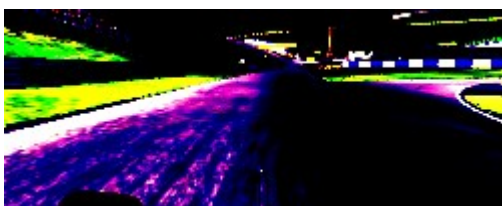
στάδιο της προεπεξεργασίας.



4.8.A Η κανονικοποιημένη εικόνα

4.8.B Η αναπαράσταση των x,y,speed

Εικόνα 4.8. Οι δύο εικόνες που δίνονται τελικά σαν είσοδος στο νευρωνικό δίκτυο (παράδειγμα 1)



4.9.A Η κανονικοποιημένη εικόνα

4.9.B Η αναπαράσταση των x,y,speed

Εικόνα 4.9. Οι δύο εικόνες που δίνονται τελικά σαν είσοδος στο νευρωνικό δίκτυο (παράδειγμα 2)

5.

ΜΟΝΤΕΛΟ DEEP LEARNING

Η καρδιά της επίλυσης του προβλήματος το οποίο αφορά αυτή την εργασία βρίσκεται στο μοντέλο το οποίο θα κάνει τις προβλέψεις για το ποιά θα είναι η κίνηση που θα πρέπει να γίνει κάθε χρονική στιγμή. Το μοντέλο είναι στην ουσία μία πάρα πολύ περίπλοκη συνάρτηση πολλών χιλιάδων μεταβλητών η οποία πρέπει να προσεγγιστεί όσο πιο αποτελεσματικά γίνεται έτσι ώστε οι προβλέψεις να είναι ικανοποιητικές σε δεδομένα τα οποία ήταν άγνωστα κατά τη διαδικασία της εκπαίδευσης. Το μοντέλο στη συγκεκριμένη περίπτωση είναι ένα νευρωνικό δίκτυο και μάλιστα με αρκετά μεγάλο αριθμό νευρώνων και επιπέδων. Το μεγάλο πλήθος αυτό δίνει το όνομα *deep learning*. Στο κεφάλαιο αυτό θα παρουσιαστούν κάποιες πρώτες προσεγγίσεις καθώς και το τελικό μοντέλο το οποίο έδωσε τα καλύτερα αποτελέσματα καθώς και λεπτομέρειες για το πως έγινε η επιλογή διάφορων συστατικών στοιχείων του αλγορίθμου καθώς και αρχιτεκτονικών.

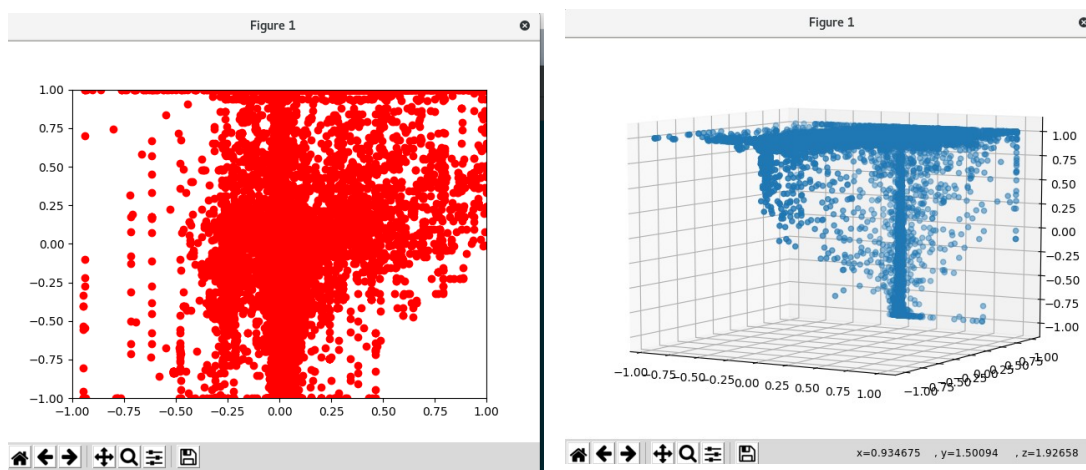
5.1.1 Πρώτες προσπάθειες

Το πρόβλημα προσεγγίστηκε με μία πληθώρα διαφορετικών τρόπων. Αρχικά το πρόβλημα αντιμετωπίστηκε μόνο με τη χρήση εικόνας ενώ τα μοντέλα ήταν απλά CNNs, τα οποία είναι ενδεδειγμένα για προβλήματα που αφορούν εικόνες και βίντεο. Τα μοντέλα περιλάμβαναν κάποια convolutional layers στην αρχή και στη συνέχεια κάποια fully connected. Οι μεγάλες διαφορές σε σχέση με το τελικό μοντέλο αφορούσαν κυρίως τη δομή του μοντέλου και το είδος των δεδομένων. Για παράδειγμα, σε κάποια από τις δοκιμές τα δεδομένα του στριψίματος δίνονταν σαν τριαδικά δεδομένα (-1 ή 0 ή 1) δηλαδή το πρόβλημα (όσον αφορά το στρίψιμο) είχε μετατραπεί σε πρόβλημα κατηγοριοποίησης (classification). Η προσέγγιση αυτή δεν ήταν ιδανική.

Σε μία άλλη προσπάθεια δοκιμάστηκε μία παραλλαγμένη εκδοχή του inception της google [5] το οποίο όμως αποδείχτηκε πολύ αργό τότε και στην εκπαίδευση (training) όσο και στις δοκιμές (testing).

Επιπλέον υλοποιήθηκαν αρκετοί διαφορετικοί τρόποι για τη προσέγγιση των τιμών Σ,Γ,Π. Μία από αυτές περιλάμβανε ένα μοντέλο για κάθε μία από αυτές τις προβλέψεις, ενώ μία

άλλη ένα μοντέλο το οποίο θα μπορεί να προβλέψει όλες τις τιμές ταυτόχρονα. Μιας και οι τιμές Γ (γκάζι), Π (πέδηση) φαίνονται εύκολα υπολογίσιμες όταν γνωρίζουμε το Σ (στρίψιμο), αφού είναι πολύ πιθανό να έχουμε μεγάλη τιμή στο Γ όταν έχουμε Σ κοντά στο μηδέν και το αντίστροφο και μεγάλη τιμή στο Π όταν έχουμε Σ μακριά από το μηδέν και αντίστροφα. Με βάση το σκεπτικό αυτό ένα μοντέλο deep learning το οποίο θα έδινε μία καλή προσέγγιση για το Σ και ένα απλό μοντέλο logistic regression ή SVR το οποίο με βάση το Σ που προβλέφθηκε θα υπολόγιζε το Γ ή τη Π μοιάζει μία καλή λύση. Στη πράξη όμως τα δεδομένα είναι πολύ πιο απρόβλεπτα μιας και κατά τη διαδικασία της οδήγησης γίνονται διορθώσεις, το σημείο φρεναρίσματος δεν είναι σταθερό σε κάθε γύρο κτλ. Ο παραπάνω ισχυρισμός φαίνεται και από την οπτικοποίηση ενός δείγματος των δεδομένων.



Εικόνα 5.1 Αριστερά: Οπτικοποίηση δεδομένων Σ και Γ . Δεξιά: Οπτικοποίηση δεδομένων Σ, Γ, Π . Όπως φαίνεται, δε μπορεί να υπάρξει κάποια καμπύλη ή κάποιο επίπεδο το οποίο μπορεί να κάνει fit στα δεδομένα αυτά. Εάν μπορούσε να βρεθεί τέτοιο επίπεδο θα μπορούσε να γίνει πρόβλεψη της ποσότητας του γκαζιού και της πέδησης μόνο με τη χρήση της ποσότητας στρίψιματος. Το ότι κάτι τέτοιο δε φαίνεται να συμβαίνει δείχνει ότι χρειάζονται περισσότερα χαρακτηριστικά (features) για τη λύση του προβλήματος όπως για παράδειγμα η εικόνα από το περιβάλλον προσομοίωσης.

Στο Κεφάλαιο 6, όπου γίνεται αξιολόγηση του μοντέλου, υπάρχει και μία σύγκριση του τελικού μοντέλου με κάποια από τα μοντέλα τα οποία απορρίφθηκαν καθώς και σύγκριση μοντέλων που έλυναν μόνο το πρόβλημα της πρόβλεψης του Σ .

5.1.2 Περιγραφή τελικού μοντέλου

Το τελικό μοντέλο είναι ένα CNN το οποίο δέχεται σαν είσοδο τα κανονικοποιημένα δεδομένα με τη διαδικασία η οποία αναλύθηκε στο κεφάλαιο 4 και δίνει σαν έξοδο και τις τρεις ζητούμενες τιμές, δηλαδή τη τιμή του γκαζιού του φρένου και της πέδησης. Το μοντέλο αποτελείται από 5 convolutional layers και από 4 fully connected. Η βελτιστοποίηση γίνεται με τον adam optimizer και το learning rate είναι σταθερό στο 0.0001, το οποίο έδειξε να οδηγεί γρήγορα σε μείωση του σφάλματος.

Οι συνολικές παράμετροι (ή βάρη) που πρέπει να υπολογιστούν είναι 910465. Στον παρακάτω πίνακα φαίνεται το πλήθος των παραμέτρων για κάθε επίπεδο του δικτύου.

Επίπεδο	Είδος	Σχήμα εξόδου	Πλήθος παραμέτρων
1	Είσοδος	(None, 100, 250, 6)	0
2A	Conv2D	(None, 48, 123, 24)	3624
2B	Conv2D	(None, 48, 123, 24)	3624
3	Conv2D	(None, 22, 60, 36)	21636
4	Conv2D	(None, 9, 28, 48)	43248
5	Conv2D	(None, 7, 26, 64)	27712
6	Conv2D	(None, 5, 24, 64)	36928
7	Fully Connected	(None, 100)	768100
8	Fully Connected	(None, 50)	5050
9	Fully Connected	(None, 10)	510
10	Fully Connected	(None, 3)	33

Πίνακας 3: Πλήθος παραμέτρων κάθε επιπέδου

Η κωδική ονομασία Conv2D αναφέρεται σε ένα CNN δύο διαστάσεων ενώ η ονομασία fully connected στο απλό multilayer perceptron, δηλαδή σε ένα σύστημα απλών νευρώνων όπου ο κάθε νευρώνας του επιπέδου i συνδέεται με κάθε νευρώνα του επιπέδου $i+1$. Στο σχήμα εξόδου φαίνεται ο ταυστής (tensor), δηλαδή ο πολυδιάστατος πίνακας ο οποίος προκύπτει σε κάθε επίπεδο. Το None δηλώνει ότι το πλήθος των ταυστών αυτών δεν είναι προκαθορισμένο. Αυτό δίνει ελευθερία στην επιλογή του πλήθους των batches που μπορεί κάποιος να δώσει στο δίκτυο κατά τη διαδικασία της εκπαίδευσης. Στο πρώτο επίπεδο το σχήμα εξόδου είναι το: (None, 100, 250, 6). Αυτό σημαίνει ότι μπορούμε να έχουμε σαν είσοδο όσα δεδομένα σχήματος 100x250x6 επιθυμούμε. Ένα δεδομένα σχήματος 100x250x6 σε αυτή τη περίπτωση είναι τα δεδομένα εισόδου όπως αυτά προέκυψαν έπειτα από την επεξεργασία που αναλύθηκε στο κεφάλαιο 4.

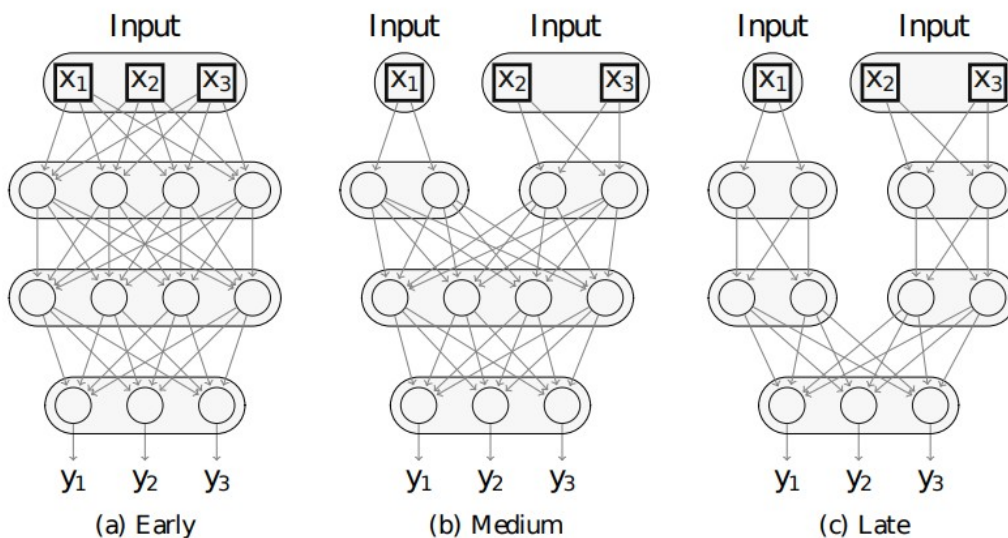
Η αφαιρετική αυτή άποψη του μοντέλου δίνει μία πρώτη εικόνα για τη δομή του. Παρ' όλα αυτά ένα μοντέλο περιέχει πάρα πολλά πράγματα τα οποία πρέπει να αναλυθούν για να υπάρξουν ικανοποιητικά αποτελέσματα. Η ανάλυση αυτή πραγματοποιείται στην επόμενη παράγραφο.

5.1.3 Ανάλυση μοντέλου

Στη παράγραφο αυτή θα αναλυθούν όλα τα συστατικά μέρη του μοντέλου καθώς και η αρχιτεκτονική του, η οποία διαφέρει σε κάποια σημεία από το αφαιρετικό σχήμα του πίνακα 3.

5.1.3.A Multimodal learning

Το multimodal learning είναι ένα σκεπτικό στρατηγικών το οποίο βοηθά στη μάθηση από δεδομένα που προέρχονται από διαφορετικές πηγές, δηλαδή στη μάθηση από δεδομένα διαφορετικού τύπου. Μερικά παραδείγματα εφαρμογής αυτών των τεχνικών είναι η ταξινόμηση βίντεο τα οποία περιέχουν ήχο και εικόνα. Καθώς το πλήθος των χαρακτηριστικών (features) μεγαλώνει το ίδιο συμβαίνει και με τη πολυπλοκότητα της λύσης και αυτό οδηγεί σε διαφορετικές προσεγγίσεις. Για παράδειγμα μία προσέγγιση θα ήταν η εκπαίδευση σε διαφορετικό μοντέλο για τον κάθε τύπο δεδομένων και στη συνέχεια μία μέση εκτίμηση των προβλέψεων αυτών των μοντέλων. Κάτι τέτοιο όμως αδυνατεί να εντοπίσει μοτίβα τα οποία βρίσκονται κρυμμένα στον συνδυασμό των διαφορετικών δεδομένων. Για καλύτερα αποτελέσματα χρησιμοποιούνται κάποιες ενδιαφέρουσες τεχνικές όπως το early fusion, το medium fusion και το late fusion όπως αυτά φαίνονται στην εικόνα 5.2.



Εικόνα 5.2. Οι διάφορες τεχνικές multimodal μοντέλων.

Το early fusion είναι η τετριμμένη περίπτωση όπου γίνεται εισαγωγή όλων των δεδομένων μαζί στο επίπεδο εισόδου. Το medium fusion (αυτό που χρησιμοποιείται και στη παρούσα εργασία), επεξεργάζεται στην ουσία κάθε διαφορετικό είδος δεδομένων σε ένα ξεχωριστό νευρωνικό δίκτυο ενός επιπέδου. Μέσω αυτής της διαδικασίας εξαγονται κάποια αρχικά features από αυτά τα δεδομένα. Στη συνέχεια τα δύο αυτά μικρά νευρωνικά ενώνονται με αποτέλεσμα το δίκτυο να χρησιμοποιήσει κάποια χρήσιμα features από κάθε είδους δεδομένων επεξεργασμένα σε ένα επίπεδο. Τέλος στο late fusion η ένωση αυτή γίνεται στο προτελευταίο επίπεδο που σημαίνει ότι το δίκτυο μπορεί να μάθει μοτίβα τα οποία αφαιρούν ταυτόχρονα και τα δύο είδη δεδομένων σε ένα αρκετά πιο υψηλό επίπεδο κατανόησης (εξαγωγής χαρακτηριστικών).

Στη προσέγγιση αυτής της εργασίας, στο στάδιο της προεπεξεργασίας, τα δεδομένα $x, y, speed$ μετατράπηκαν σε εικόνες, οπότε φαίνεται ότι έχουμε μόνο δεδομένα εικόνες σαν είσοδο. Παρ' όλα αυτά η εύρεση features -ξεχωριστά για την εικόνα από το περιβάλλον προσομοίωσης και ξεχωριστά για την εικόνα που αναφέρεται στα $x, y, speed$ - βελτίωσε την ακρίβεια του μοντέλου μιας και τα μητρώα αυτά έχουν εντελώς διαφορετική δομή. Η ένωση των δύο μικρών νευρωνικών γίνεται στο δεύτερο επίπεδο (medium fusion). Η διαίσθηση πίσω από αυτή την επιλογή έχει να κάνει με την ισχυρή σχέση που φαίνεται να έχουν τα δεδομένα της εικόνας από το περιβάλλον προσομοίωσης με τα δεδομένα $x, y, speed$.

Εναλλακτικά ή ακόμα και συνδυαστικά, θα μπορούσε να υπάρξει και διαχωρισμός των YUV καναλιών και συγκεκριμένα του καναλιού Y από τα κανάλια U και V, αλλά η παραπάνω προσέγγιση, δηλαδή το medium fusion με δεδομένα εικόνες από το περιβάλλον προσομοίωσης από τη μία και των $x, y, speed$ από την άλλη, δίνει ικανοποιητικά αποτελέσματα.

5.1.3.B CNN και fully connected

Το μοντέλο, πέρα από το διπλό medium fusion layer, αποτελείται από άλλα 4 convolutional layers και από άλλα 4 fully connected. Η είσοδος και η έξοδος κάθε convolutional layer είναι ένας 4D ταυστής (tensor) όπου η πρώτη διάσταση είναι μεταβλητή (None στη python) και αναπαριστά το πλήθος batches δεδομένων που επεξεργάζονται και τα άλλα 3 είναι φυσικά οι εικόνες που αποτελούνται από 6 κανάλια.

Για κάθε convolutional layer ορίζεται το πλήθος των φίλτρων, καθώς και το μέγεθος τους. Τα φίλτρα είναι από 24 έως 64 και το μέγεθός τους από 3 έως 5. Οι τιμές κρατήθηκαν σε λογικά πλαίσια μιας και η αύξηση των φίλτρων ή του μεγέθους σημαίνει μεγάλη αύξηση στο πλήθος των μεταβλητών που πρέπει να βελτιστοποιηθούν. Το stride, το οποίο δείχνει τον αριθμό των pixels

τον οποίο το φίλτρο περνάει όταν διασχίζει την εικόνα παίρνει τιμές από 3 έως 5. Οι επιλογές αυτές έγιναν εμπειρικά και μέσω έρευνας που προήλθε από διάφορες εργασίες της βιβλιογραφίας.

Η αρχικοποίηση των συναπτικών βαρών των φίλτρων έγινε με τυχαίες τιμές οι οποίες ακολουθούν scaling κανονική κατανομή. Τα τυχαία δεδομένα που ακολουθούν κατανομή βοηθούν στην αντιμετώπιση του προβλήματος των πολύ μεγάλων ή πολύ μικρών παραγώγων. Το scaling κομμάτι έχει να κάνει με την επιλογή του διαστήματος στο οποίο θα απλώνεται η κανονική κατανομή. Το διάστημα αυτό εξαρτάται από τη διάσταση της εισόδου (έστω dim) και είναι το:

$$\left[\frac{-\sqrt{3}}{\sqrt{dim}}, \frac{\sqrt{3}}{\sqrt{dim}} \right]$$

Ο τρόπος αυτός, ο οποίος περιγράφεται στο paper [6] Random Walk Initialization for training very deep feedforward networks ακριβώς επειδή το διάστημα είναι μεταβλητό ανάλογα με το μέγεθος της εισόδου το διάστημα της κανονικής κατανομής μικραίνει όταν τα δεδομένα εισόδου είναι πολλά και το μεγαλώνει όταν είναι λίγα. Η επιλογή αυτή μοιάζει λογική μιας και αν η είσοδος ήταν μεγάλη και το διάστημα τιμών της κανονικής κατανομής επίσης μεγάλο θα ήταν πιο πιθανή μία μεγάλη αύξηση των τιμών των βαρών που θα μπορούσαν να οδηγούσαν τη διαδικασία της εκπαίδευσης σε δυσκολίες.

Μία άλλη δικλείδα ασφαλείας του μοντέλου είναι το weight decay το οποίο παίρνει εδώ τη τιμή 0.001. Το weight decay είναι ένας όρος κανονικοποίησης ο οποίος τιμωρεί τα πολύ μεγάλα βάρη. Η τιμή 0.001 εκφράζει μία σχετικά μικρή “τιμωρία”. Ας υποθέσουμε ότι έχουμε τη συνάρτηση σφάλματος $E(w)$ την οποία επιθυμούμε να ελαχιστοποιήσουμε. Ο αλγόριθμος gradient descent θα ανανεώσει τα βάρη του δικτύου ως εξής:

$$w_i = w_i - \eta \frac{\partial E}{\partial w_i}$$

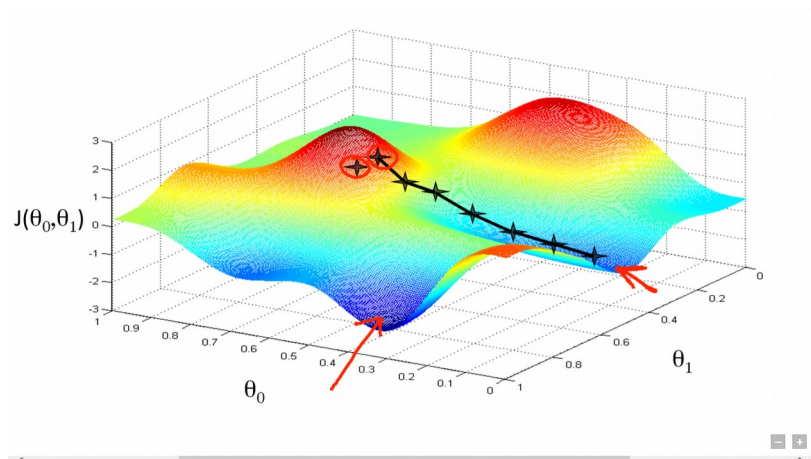
όπου η είναι το learning rate. Σημειώνεται ότι για τον αλγόριθμο αυτό χρησιμοποιήθηκε σταθερό learning rate με τη τιμή 0.0001, το οποίο αν και μοιάζει αρκετά μικρό βοήθησε στη σωστή σύγκλιση του μοντέλου χωρίς αυτή να είναι ιδιαίτερα αργή και γενικά έδειξε να αποδίδει το ίδιο ικανοποιητικά με μεθόδους μειούμενου learning rate κατά τη διάρκεια της εκπαίδευσης που δοκιμάστηκαν.

Γενικά όσο μεγαλύτερο είναι το learning rate τόσο μεγαλύτερες αλλαγές γίνονται στα βάρη, όπως φαίνεται και από τον παραπάνω τύπο, αλλά αν η τιμή είναι πολύ μεγάλη τότε είναι πιθανό ο gradient descend να “προσπεράσει” το ολικό ελάχιστο και άρα να δώσει μία κακή λύση.

Στο σημείο αυτό έρχεται η κανονικοποίηση των βαρών, έτσι ώστε να μειωθεί η ελευθερία τους και να αποφευχθούν προβλήματα μεγάλων βαρών και overfitting. Ο τρόπος για να γίνει αυτό είναι να μετασχηματίσουμε τη συνάρτηση κόστους σε:

$$E'(w) = E(w) + \frac{\lambda}{2} w^2$$

όπου η παράμετρος λ είναι το weight decay.



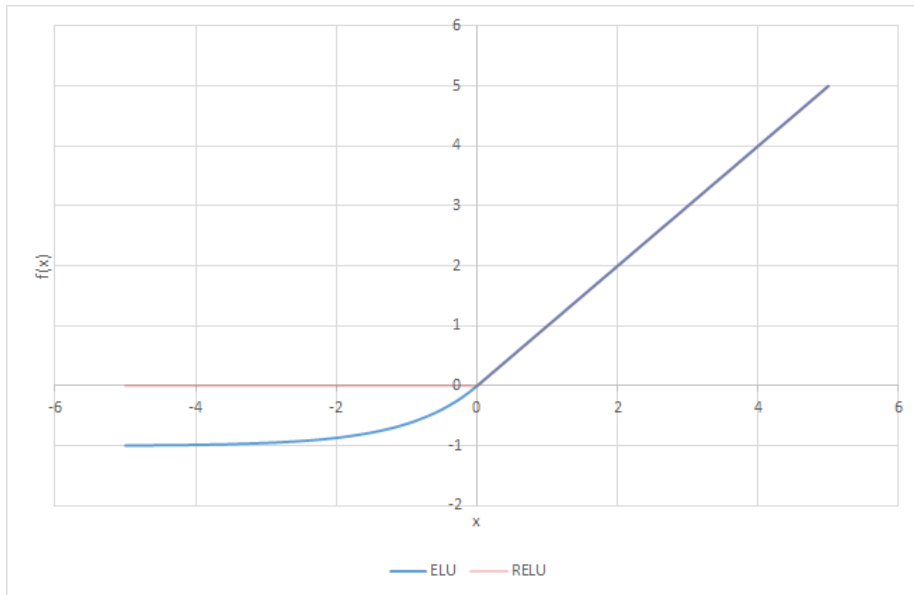
Εικόνα 5.3. Οπτική αναπαράσταση της βελτιστοποίησης μέσω gradient descent

Το τελευταίο convolutional layer δίνει ως έξοδο ένα 4D tensor μεγέθους (None,34,84,64), όπου None είναι το πλήθος των δεδομένων που βρίσκονται στο επεξεργαζόμενο batch. Σε αυτό το tensor βρίσκεται η γνώση που έχει προκύψει από τα φίλτρα τα οποία διέσχισαν τις εικόνες. Η γνώση αυτή περνάει στη συνέχεια μέσα από τα fully connected επίπεδα (δηλαδή επίπεδα πλήρως διασυνδεδεμένων νευρώνων) απ' όπου θα προκύψει και η τελική πρόβλεψη. Χρησιμοποιήθηκαν τέσσερα τέτοια επίπεδα μεγέθους 100,50 και 10, ενώ το τελευταίο επίπεδο είχε μέγεθος όσο και το πλήθος των εξόδων των οποίων θέλουμε να προβλέψουμε, δηλαδή τρία.

Τα βάρη σε αυτά τα επίπεδα είναι αρχικοποιημένα μέσω της κατανομής truncated normal και το weight decay έχει τη τιμή 0.001. Η truncated κανονική κατανομή είναι σαν την κανονική με τη διαφορά ότι κόβονται οι τιμές όπου το μέτρο τους απέχει περισσότερο από 2 τυπικές αποκλίσεις από τον μέσο. Στη πράξη αφαιρούνται οι ουρές της κατανομής, πράγμα το οποίο πετυχαίνει την αποκοπή των πιο ακραίων τιμών της κατανομής.

5.1.3.C Συνάρτηση ενεργοποίησης

Η συνάρτηση ενεργοποίησης που χρησιμοποιείται σε όλο το μοντέλο είναι μία παραλλαγή της ReLu, η Elu [7]. Η διαφορά από τη κλασική ReLu είναι ότι για αρνητικές τιμές η τιμή της συνάρτησης δεν είναι μηδέν αλλά μία μικρή τιμή η οποία τείνει στο 1.



Εικόνα 5.4 Η συνάρτηση ενεργοποίησης Elu

Οι διαφορές στην απόδοση των δύο συναρτήσεων ενεργοποίησης είναι αμελητέες στο συγκεκριμένο πρόβλημα.

5.1.3.D Συνάρτηση σφάλματος

Η συνάρτηση σφάλματος (loss function) $E(w)$ είναι η συνάρτηση η οποία περιγράφει την απόσταση των προβλέψεων από τις πραγματικές τιμές και σκοπός ενός νευρωνικού δικτύου είναι η εύρεση των τιμών w που την ελαχιστοποιούν. Οι συναρτήσεις σε προβλήματα ταξινόμησης σε κλάσεις διαφέρουν από τις συναρτήσεις σε regression προβλήματα μιας και στα τελευταία πρέπει να βρεθεί η αριθμητική απόσταση μεταξύ συνεχών τιμών.

Η πιο συνηθισμένη συνάρτηση σφάλματος είναι η συνάρτηση του μέσου τετραγωνικού σφάλματος η οποία δίνεται από τον παρακάτω τύπο.

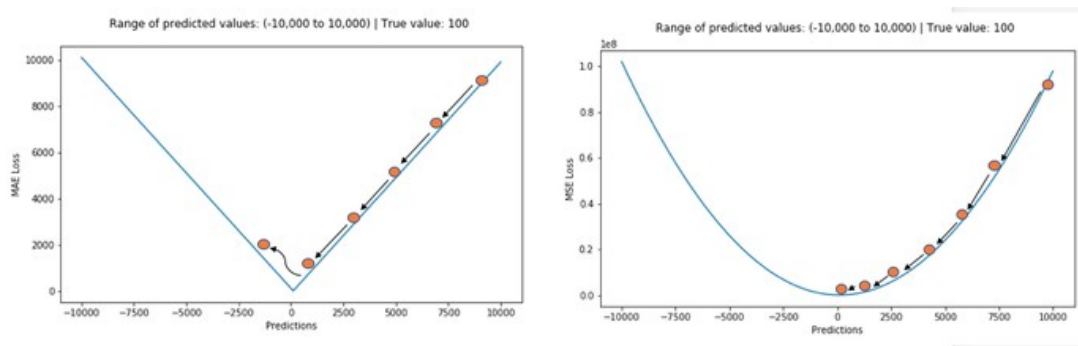
$$MSE = \frac{\sum_{i=1}^n (y_i - y_i^p)^2}{n}$$

Μία ακόμα συνήθης συνάρτηση ενεργοποίησης είναι η συνάρτηση του μέσου απόλυτου σφάλματος.

$$MAE = \frac{\sum_{i=1}^n |y_i - y_i^p|}{n}$$

Οι δύο αυτές προσεγγίσεις έχουν κάποια πλεονεκτήματα και κάποια μειονεκτήματα. Για παράδειγμα η MAE είναι αρκετά χρήσιμη όταν τα δεδομένα έχουν ακραίες τιμές (outliers),

αλλά το αρνητικό της είναι ότι η παράγωγός της είναι σταθερή πράγμα που σημαίνει ότι θα είναι μεγάλη ακόμα και για μικρές τιμές της συνάρτησης, πράγμα το οποίο δεν είναι καλό για τη διαδικασία της μάθησης. Αντίθετα η MSE ενώ για μεγάλες τιμές έχει μεγάλη παράγωγο, όσο πλησιάζουμε στο ελάχιστο η παράγωγος μικραίνει, δίνοντας μεγαλύτερη ακρίβεια στην διαδικασία της ελαχιστοποίησης.



Εικόνα 5.5 Οι συναρτήσεις ενεργοποίησης MAE και MSE

Παρ' όλα αυτά καμία από τις δύο αυτές συναρτήσεις ενεργοποίησης δεν είναι βέλτιστη για το πρόβλημα αυτής της εργασίας. Το κοινό πρόβλημα και των δύο αυτών συναρτήσεων είναι ότι όταν έχουμε ένα unbalanced dataset οι προβλέψεις δεν είναι οι επιθυμητές. Για παράδειγμα όταν το 90% των παρατηρήσεων έχουν τη τιμή 150 και το υπόλοιπο 10% τιμές στο διάστημα 0-30 τότε η συνάρτηση MAE θα προβλέπει πάντα τη τιμή 150 αγνοώντας τις υπόλοιπες σαν outliers μιας οι προβλέψεις θα τείνουν προς τη διάμεσο των δεδομένων. Αντίστοιχα με τη χρήση της MSE θα έχουμε πολλές προβλέψεις στο διάστημα 0-30 μιας και η συνάρτηση θα τείνει προς τα outliers.

Στο συγκεκριμένο πρόβλημα λοιπόν, μία σύντομη μέτρηση ενός μέρους των δεδομένων έδειξε ότι το 75.2% των δεδομένων δίνει έξοδο στο τιμόνι τιμές στο $[-0.1, 0.1]$ ενώ ολόκληρο το διάστημα είναι το $[-1, 1]$. Αυτό είναι λογικό μιας και τη περισσότερη ώρα το τιμόνι βρίσκεται στην θέση 0, το γκάζι είναι τέρμα πατημένο και το φρένο δε χρησιμοποιείται.

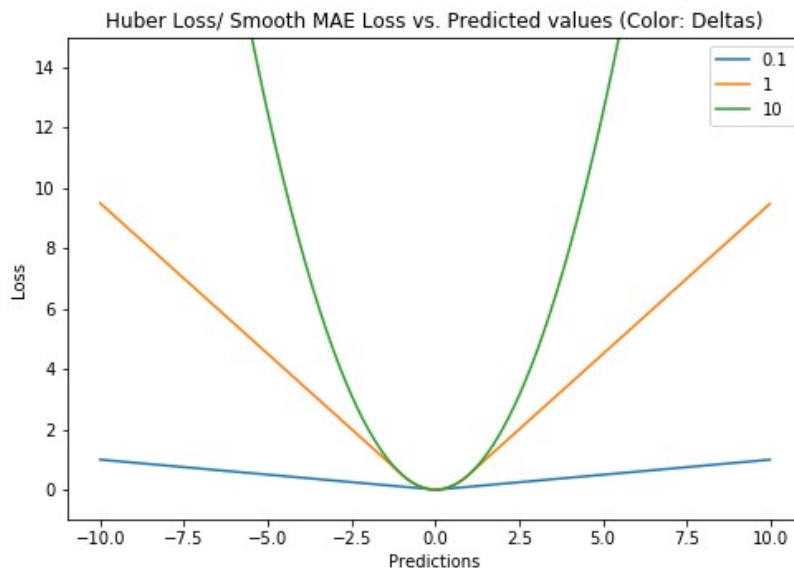
Διάστημα	Τιμόνι	Πέδηση	Γκάζι
$[-1, -0.3)$	1.17%	5,87%	55.76
$[-0.3, -0.1)$	7.13%	0.90%	8.94%
$[-0.1, 0.1)$	74.77%	0.88%	9.37%
$[0.1, 0.3)$	7.57%	0.86%	4.70%
$[0.3, 1]$	8.73%	91.46%	21.21%

Πίνακας 4: Προσέγγιση ποσοστού κάθε διαστήματος των δεδομένων εκπαίδευσης. Σημείωση: Η μέγιστη εφαρμογή των πετάλ “πέδηση” και “γκάζι” δίνουν τη τιμή -1.

Η παραπάνω ανάλυση των δεδομένων ταιριάζει με περιγραφή των προβλημάτων των συναρτήσεων σφάλματος MSE και MAE, οπότε πρέπει να βρεθεί μία διαφορετική προσέγγιση. Σημειώνεται ότι δοκιμάστηκε το balance των δεδομένων, αλλά προέκυψαν διαφορετικά προβλήματα, οπότε η μέθοδος εγκαταλείφθηκε. Επιπλέον η μέθοδος αυτή συνηθίζεται περισσότερο σε προβλήματα ταξινόμησης σε κλάσεις.

Μία λύση στο πρόβλημα είναι η χρήση της συνάρτησης Huber σαν συνάρτηση σφάλματος. Η συνάρτηση αυτή είναι αρχικά λιγότερο ευαίσθητη στα outliers απ' ό,τι η MSE. Είναι επίσης παραγωγίσιμη στο μηδέν. Η συνάρτηση αποτελείται από δύο κλάδους όπου στην ουσία ταυτίζεται με το MAE για μεγάλες τιμές σφάλματος, ενώ για μικρές ταυτίζεται με το τετραγωνικό. Για να οριστεί το “μεγάλο” και το “μικρό” υπάρχει η υπερπαράμετρος δ η οποία μπορεί να οριστεί από τον χρήστη.

$$L_{\delta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$



Εικόνα 5.6 Η συνάρτηση Huber για διάφορες τιμές του δ

Η Huber loss συνδυάζει τα θετικά στοιχεία των MSE και MAS επιτρέποντας τον ευκολότερο εντοπισμό του ελάχιστου καθώς και την σωστή αντιμετώπιση των outliers.

5.1.3.E Το τελικό μοντέλο

Το μοντέλο υλοποιήθηκε με τη βοήθεια της βιβλιοθήκης tflearn η οποία τρέχει πάνω στο tensorflow. Ακολουθεί ο κώδικας του μοντέλου, έτσι ώστε να γίνουν φανερές όλες οι λεπτομέρειες της σχεδίασής του.

```
network = input_data(shape=[None, 100, 250, 6])

part_one = network[...,:3]
part_two = network[...,:3]

network1 = conv_2d(part_one, 24, 5, strides=5, activation='elu')
network2 = conv_2d(part_two, 24, 5, strides=5, activation='elu')
network = tflearn.merge([network1, network2], 'concat')

network = conv_2d(network, 36, 5, strides=5, activation='elu')
network = conv_2d(network, 48, 5, strides=5, activation='elu')
network = conv_2d(network, 64, 3, strides=3, activation='elu')
network = conv_2d(network, 64, 3, strides=3, activation='elu')

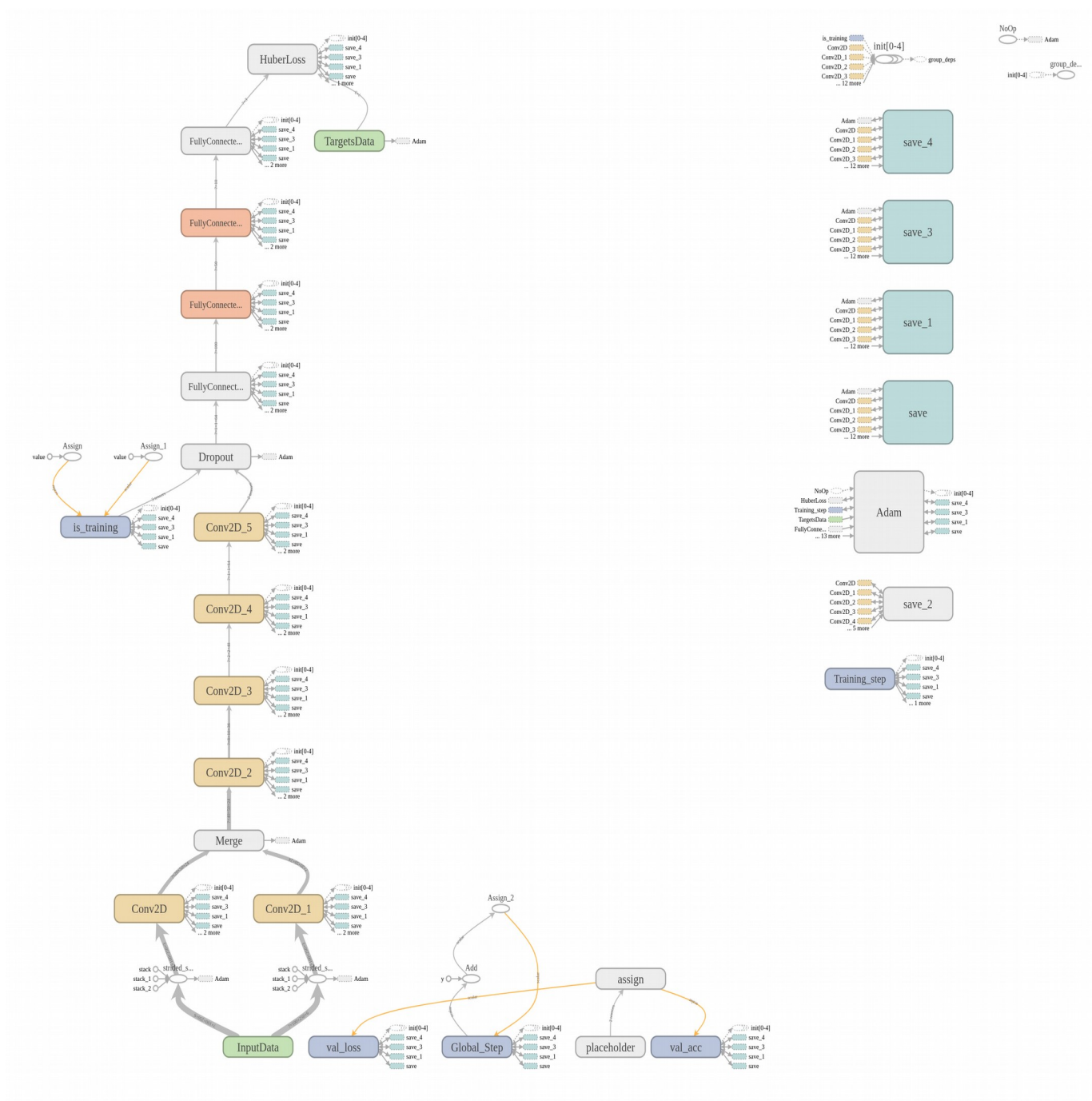
network = dropout(network, 0.5)

network = fully_connected(network, 100, activation='elu')
network = fully_connected(network, 50, activation='elu')
network = fully_connected(network, 10, activation='elu')
network = fully_connected(network, output, activation='linear')

network = tflearn.regression(network, optimizer='adam',
                             loss='huber_loss', metric=None, learning_rate = 0.0001)

model = tflearn.DNN(network, checkpoint_path='model_spiroset',
                    max_checkpoints=1, tensorboard_verbose=2, tensorboard_dir='log')
```

Σχηματικά το μοντέλο φαίνεται στην επόμενη εικόνα.



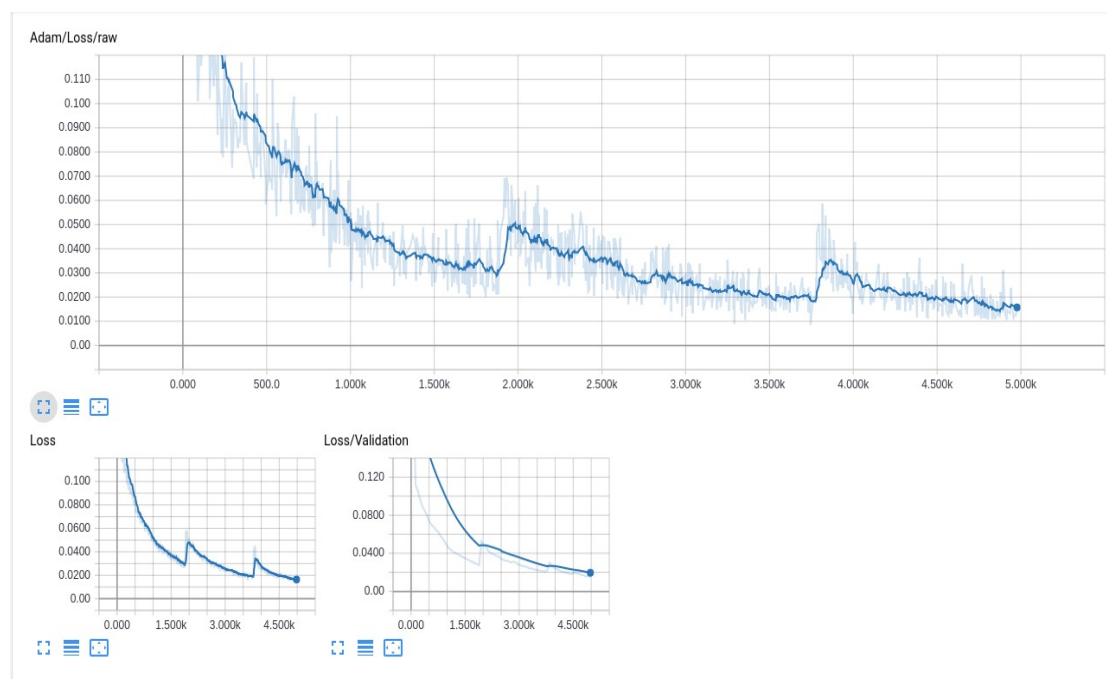
Εικόνα 5.7 Σχηματική αναπαράσταση του μοντέλου με τη βοήθεια του tensorboard

5.1.4 Εκπαίδευση μοντέλου

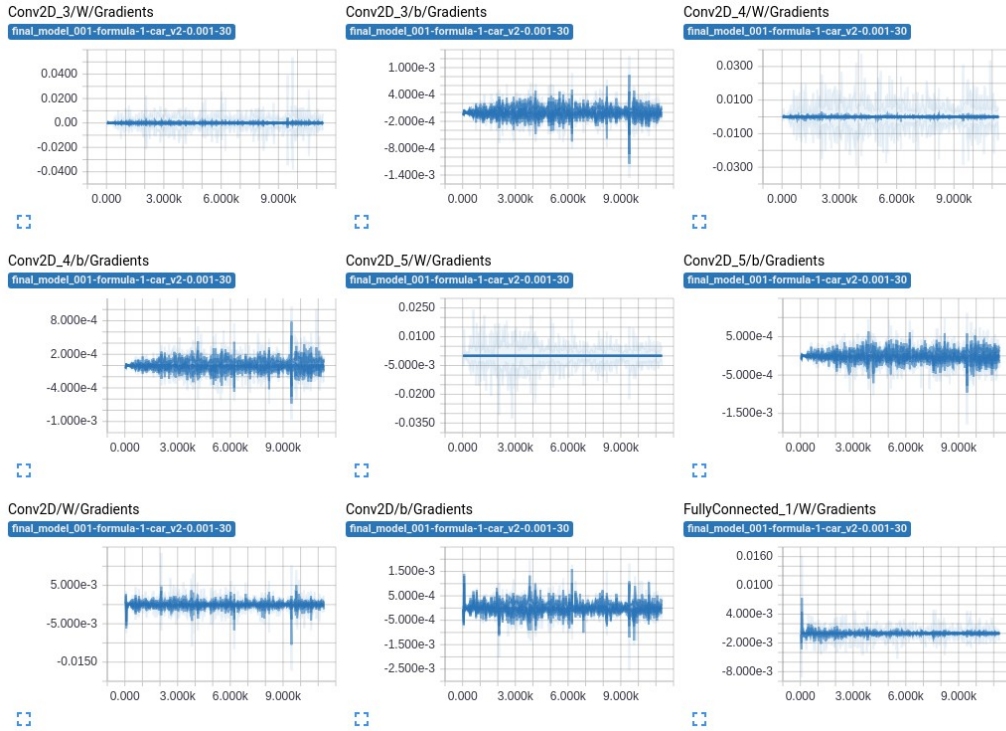
Όπως φαίνεται και στον παραπάνω κώδικα, η εκπαίδευση του μοντέλου έγινε με σταθερό learning rate στο 0.0001 και ο αλγόριθμος βελτιστοποίησης που χρησιμοποιήθηκε ήταν ο Adaptive Moment Estimation (Adam) [8]. Ο αλγόριθμος αυτός είναι μία επέκταση του Stochastic gradient descent και συνδυάζει χαρακτηριστικά από τον AdaGrad και τον RMSProp. Συγκεκριμένα, η μέθοδος αυτή διατηρεί ένα learning rate ανά παράμετρο το οποίο βελτιώνει την απόδοση σε προβλήματα αραιών παραγώγων (πχ επεξεργασία φυσικής γλώσσας και προβλήματα με εικόνες). Τα rates αυτά προσαρμόζονται ανάλογα με τη μέση τιμή των μέτρων των παραγώγων ως προς τα βάρη, δηλαδή ως προς της ταχύτητα αλλαγής των τιμών τους.

Για την εκπαίδευση χρησιμοποιήθηκε ένα μέρος από τα δεδομένα τα οποία έχουν συλλεχθεί. Από αυτά τα δεδομένα το 80% χρησιμοποιήθηκε για την εκπαίδευση και το υπόλοιπο 20% για την αξιολόγηση (validation). Τα δεδομένα εισήχθησαν στον αλγόριθμο με τυχαία σειρά και ολοκληρώθηκαν 30 εποχές (epochs) εκπαίδευσης.

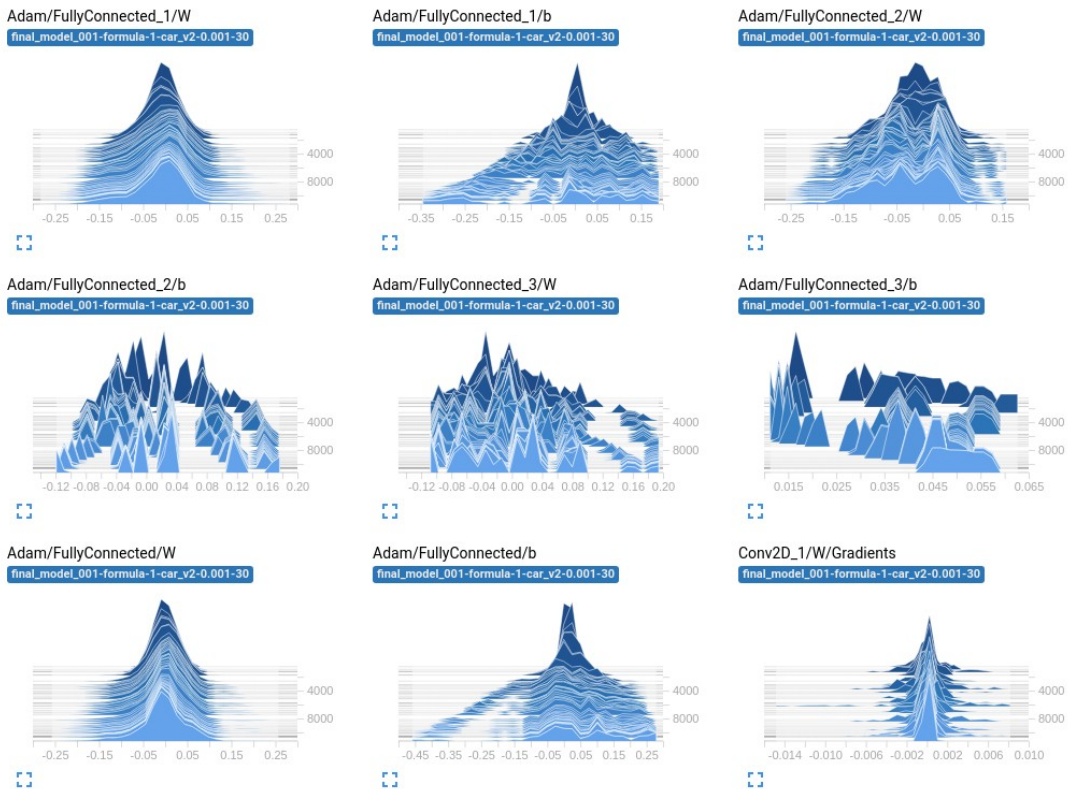
Με τη βοήθεια του εργαλείου tensorboard έγινε εξαγωγή πολύ χρήσιμων γραφημάτων για τη κατανόηση της διαδικασίας της εκπαίδευσης του μοντέλου. Τα γραφήματα αυτά παρουσιάζονται στη συνέχεια.



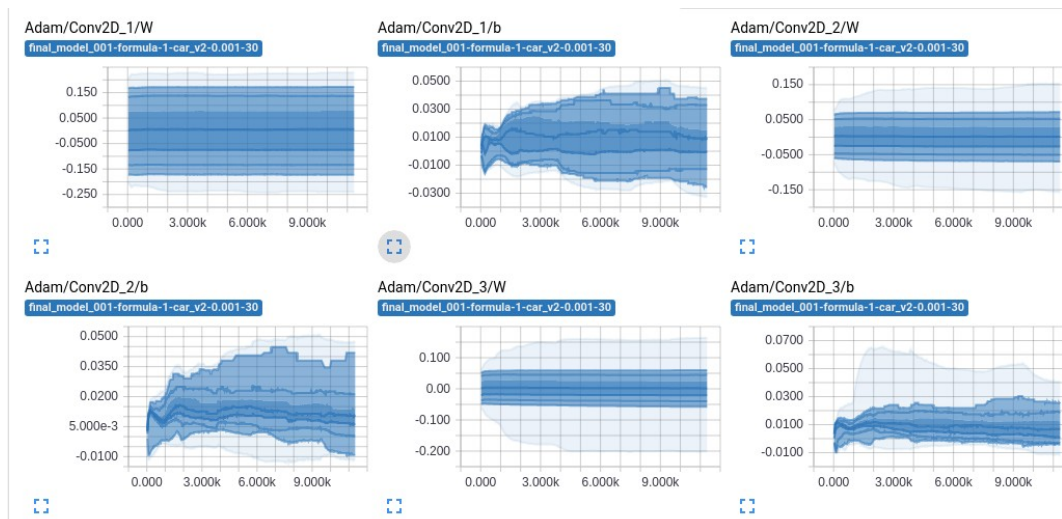
Εικόνα 5.8 Η γραφική παράσταση της συνάρτησης σφάλματος μετά από 30 epochs



Εικόνα 5.9 Οι τιμές των παραγώγων σε διάφορα επίπεδα του δικτύου. Παρατηρείται ότι οι τιμές αυτές παραμένουν σε ένα μικρό διάστημα, λόγω των μέτρων που λήφθηκαν.



Εικόνα 5.10 Οι τιμές των βαρών και των biases σε διάφορα επίπεδα του δικτύου σε τρισδιάστατη απεικόνιση.



Εικόνα 5.11 Οι τιμές των βαρών και των biases σε διάφορα επίπεδα του δικτύου

6.

ΑΞΙΟΛΟΓΗΣΗ

Η αξιολόγηση του αλγορίθμου έγινε σε δύο φάσεις. Η πρώτη αποτελείται από ένα απλό benchmarking στο οποίο μετρήθηκε η μέση διαφορά των “πραγματικών” τιμών από τις τιμές που το μοντέλο προέβλεψε και η δεύτερη φάση αποτελείται από ένα οπτικό εργαλείο το οποίο κάνει προβλέψεις πάνω σε πραγματικά δεδομένα τα οποία ο χρήστης μπορεί να δει. Η κύρια μετρική που χρησιμοποιήθηκε για την αξιολόγηση του αλγορίθμου ήταν η MAE δηλαδή η μέση απόσταση απόλυτου σφάλματος. Επειδή οι τιμές τις εξόδου βρίσκονται στο $[-1,1]$ το μέγιστο σφάλμα που μπορεί να υπάρξει είναι το 2, όταν το μοντέλο προβλέπει ακριβώς την αντίθετη τιμή.

Η αξιολόγηση πραγματοποιήθηκε σε δεδομένα που δεν έχουν χρησιμοποιηθεί για την εκπαίδευση του μοντέλου.

6.1.1 Benchmarking

Για το αρχικό benchmarking του μοντέλου υλοποιήθηκε ένα απλό script το οποίο υπολόγισε το μέσο απόλυτο σφάλμα και το μέσο τετραγωνικό σφάλμα για κάθε ένα από τα τρία δεδομένα εξόδου του δικτύου. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα.

Μετρική	Στρίψιμο	Πέδηση	Γκάζι
MAE	0.068	0.07	0.26

Πίνακας 5: Απόδοση μοντέλου

6.1.2 Σύγκριση με άλλα μοντέλα

Ένα λίγο διαφορετικό μοντέλο που δοκιμάστηκε ήταν ένα μοντέλο στο οποίο εφαρμόστηκαν δύο *lstm* layers. Τα επίπεδα αυτού του είδους χρησιμοποιούν γνώση και από προηγούμενα δεδομένα έτσι ώστε να παράξουν τη πρόβλεψή τους. Προφανώς τα δεδομένα σε αυτή τη περίπτωση δεν ανακατεύτηκαν, αλλά δόθηκαν στη σειρά στον αλγόριθμο. Παρ' όλα αυτά, η προσέγγιση αυτή δεν έδειξε να βοηθά στη βελτίωση της απόδοσης όπως φαίνεται και στον παρακάτω πίνακα. Τα δεδομένα δείχνουν το μέσο απόλυτο σφάλμα και το μέσο τετραγωνικό σφάλμα για τα δεδομένα του στριψίματος.

Μετρική	Βασικό Μοντέλο	Με LSTM layers
MAE	0.043	0.03
MSE	0.0059	0.0050

Πίνακας 6: Σύγκριση *lstm* μοντέλου με το βασικό μοντέλο του κεφαλαίου 5

6.1.3 Βελτίωση του Medium Fusion

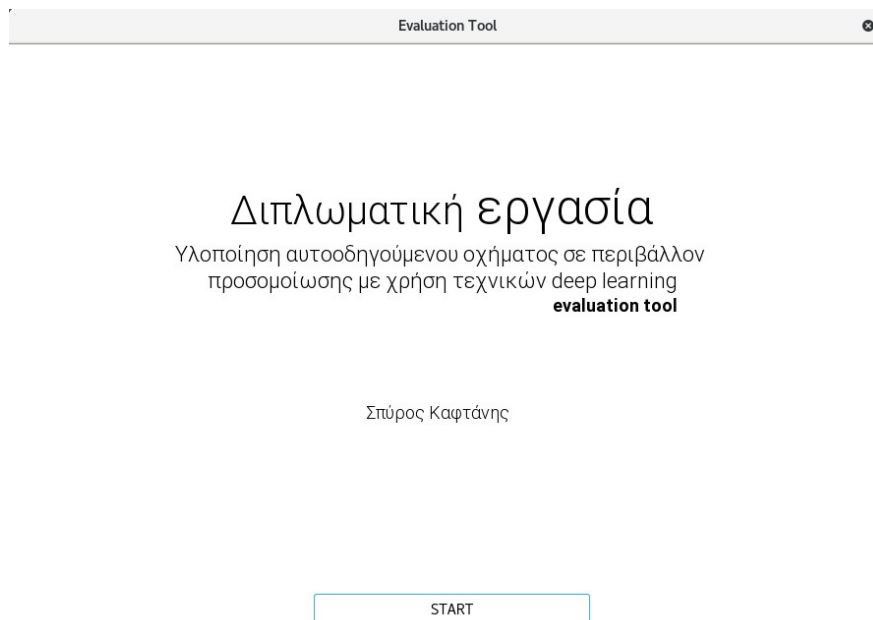
Όπως αναφέρθηκε στο κεφάλαιο 5.1.3 το μοντέλο υλοποιήθηκε με τη τακτική *medium fusion* όπου η εικόνα από το περιβάλλον προσομοίωσης και η εικόνα που προέκυψε από το συνδυασμό των χαρακτηριστικών *x,y,speed* πέρασαν από ένα μικρό *convolutional layer* και στη συνέχεια ενώθηκαν μαζί σε ένα κοινό νευρωνικό δίκτυο. Η τεχνική αυτή βελτίωσε το μοντέλο σε έναν καθόλου αμελητέο βαθμό. Η δοκιμή της τεχνικής αυτής έγινε σε ένα δίκτυο όμοιο με αυτό που αναλύθηκε στο κεφάλαιο 5 και οι μετρήσεις έγιναν πάνω στη πρόβλεψη της τιμής του στριψίματος. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα.

Μετρική	Χωρίς Medium Fusion	Με Medium Fusion
MAE	0.043	0.03
MSE	0.0059	0.0050

Πίνακας 7: Σύγκριση *medium fusion* με την απλή αρχιτεκτονική

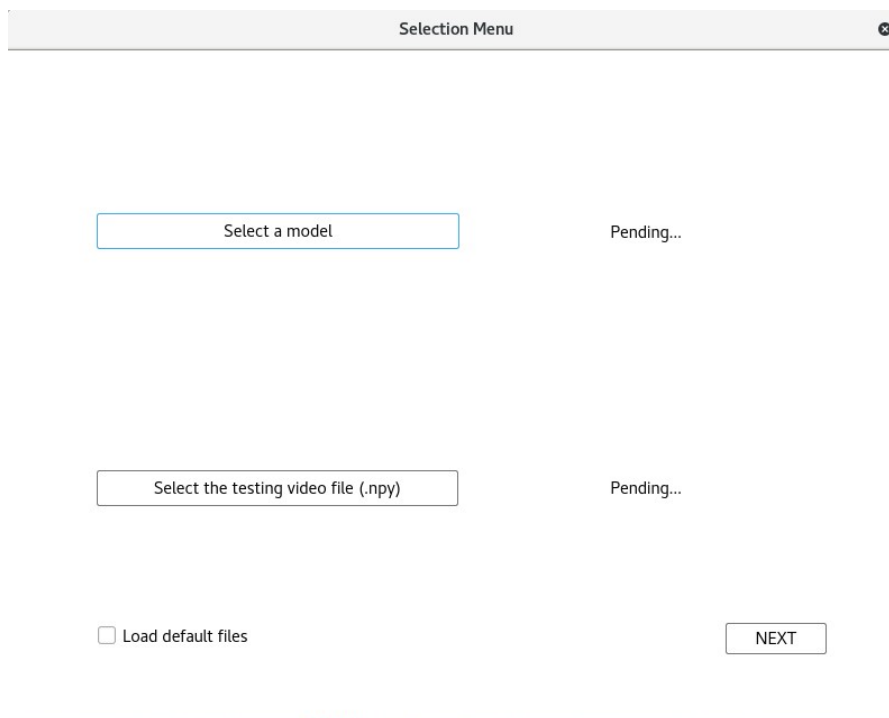
6.1.4 Παρουσίαση εργαλείου αξιολόγησης

Για την οπτική αξιολόγηση σε πραγματικά δεδομένα που δεν έχουν χρησιμοποιηθεί για τη διαδικασία της εκπαίδευσης αναπτύχθηκε ένα εργαλείο γραμμένο σε python με τη βοήθεια της βιβλιοθήκης γραφικών qt.



Εικόνα 6.1 Η αρχική εικόνα του εργαλείου αξιολόγησης

Στο εργαλείο αυτό υπάρχει η δυνατότητα επιλογής μοντέλου και των δεδομένων δοκιμής από τον χρήστη μέσω του κατάλληλου μενού. Αυτό επιτρέπει την εύκολη εναλλαγή των δοκιμών χωρίς να χρειάζεται παρέμβαση στον κώδικα της εφαρμογής. Στις εικόνες που ακολουθούν χρησιμοποιείται το προεπιλεγμένο μοντέλο, που είναι το μοντέλο που περιγράφηκε στο κεφάλαιο 5.



Selection Menu

Select a model Pending...

Select the testing video file (.npy) Pending...

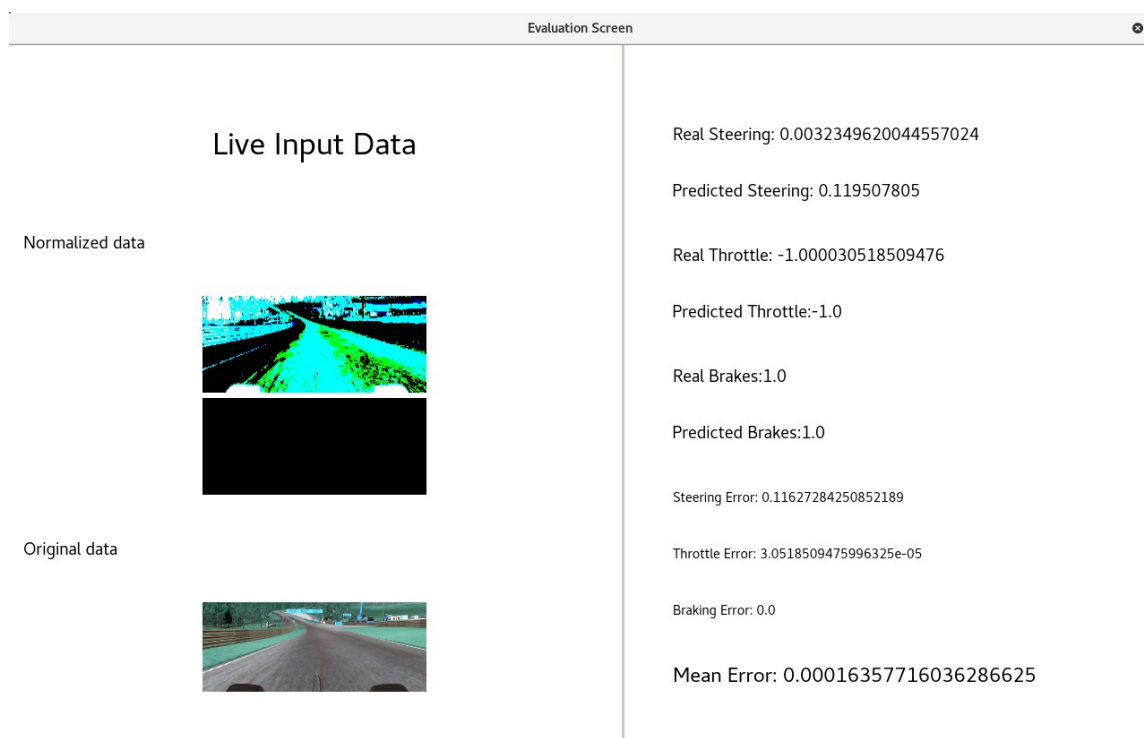
☐ Load default files

NEXT

Εικόνα 6.2 Μενού επιλογής μοντέλου και δεδομένων δοκιμής

Στο εργαλείο αυτό παρουσιάζεται η αρχική εικόνα από το περιβάλλον προσομοίωσης (κομμένη στο μέγεθος 100x250) μαζί με την ακριβή είσοδο του δικτύου η οποία είναι η κανονικοποιημένη εικόνα από το περιβάλλον προσομοίωσης μαζί με την μονοχρωματική εικόνα που συνθέτουν τα χαρακτηριστικά $x, y, speed$.

Στο δεξί μέρος φαίνονται σε πραγματικό χρόνο οι πραγματικές τιμές των Σ, Γ, Π καθώς και οι τιμές που ο αλγόριθμος προβλέπει. Φαίνεται επίσης το σφάλμα για κάθε έξοδο καθώς και το μέσο σφάλμα. Το μέσο σφάλμα αφορά τη μέση τιμή σφάλματος για όλες τις εξόδους (Σ, Γ, Π) καθ' όλη όμως τη διάρκεια λειτουργίας του προγράμματος και όχι μόνο κατά τη συγκεκριμένη χρονική στιγμή. Η τιμή αυτή βοηθάει στο να δούμε πόσο καλά πηγαίνει το μοντέλο κατά τη πάροδο του χρόνου.



Εικόνα 6.3 Στιγμιότυπο οθόνης από το εργαλείο αξιολόγησης

7.

ΤΕΧΝΙΚΕΣ ΛΕΠΤΟΜΕΡΕΙΕΣ

Στο κεφάλαιο αυτό παρουσιάζονται συνοπτικά κάποιες λεπτομέρειες της υλοποίησης οι οποίες κρίνονται άξιες αναφοράς. Η εργασία υλοποιήθηκε στη γλώσσα python με τη βοήθεια δύο διαφορετικών υπολογιστικών συστημάτων (βλ. 7.1.1) και με τη χρήση διάφορων βιβλιοθηκών (βλ. 7.1.5)

7.1.1 Τεχνικές προδιαγραφές υπολογιστικών συστημάτων

Το πρώτο σύστημα που χρησιμοποιήθηκε ήταν ο προσωπικός υπολογιστής του γράφοντα. Χρησιμοποιήθηκε για την εκτέλεση του περιβάλλοντος προσομοίωσης και για τη συλλογή των δεδομένων. Οι τεχνικές προδιαγραφές του φαίνονται στον παρακάτω πίνακα:

Τύπος και συχνότητα επεξεργαστή	Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
Πυρήνες	4
Threads ανά πυρήνα	2
Μνήμη RAM	8GB
Κάρτα Γραφικών	NVIDIA GTX 650, 1GB

Πίνακας 8: Τεχνικές προδιαγραφές προσωπικού υπολογιστή

Η διαδικασία της εκπαίδευσης είναι μια ιδιαίτερα απαιτητική εργασία η οποία απαιτεί αρκετούς παράλληλους υπολογισμούς. Για το λόγο αυτό η εκπαίδευση γίνεται συνήθως σε κάρτες γραφικών οι οποίες ολοκληρώνουν τη διαδικασία σε πολύ μικρότερο χρονικό διάστημα. Η κάρτα γραφικών του προσωπικού υπολογιστή δεν ήταν επαρκής για να επιτύχει ένα αξιοπρεπή χρονικό διάστημα εκπαίδευσης και γι αυτό το λόγο χρειάστηκε και ένα δεύτερο μηχάνημα το οποίο ανήκει στο Πανεπιστήμιο Πατρών. Το σύστημα αυτό είναι ιδανικό για εκπαίδευση μοντέλων deep learning κυρίως λόγω της ισχυρής κάρτας γραφικών που διαθέτει.

Τύπος και συχνότητα επεξεργαστή	Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz
Πυρήνες	4
Threads ανά πυρήνα	2
Μνήμη RAM	16GB
Κάρτα Γραφικών	NVIDIA Tesla K40c, 12GB

Πίνακας 8: Τεχνικές προδιαγραφές υπολογιστή Πανεπιστημίου Πατρών

7.1.2 Πρακτικές συλλογής δεδομένων

Η συλλογή δεδομένων από το περιβάλλον προσομοίωσης είναι μία παράλληλη διαδικασία, μιας και πρέπει ταυτόχρονα να διαβάζονται δεδομένα από πέντε διαφορετικές πηγές, οι οποίες είναι οι συσκευές εισόδου για το στρίψιμο, το γκάζι και τη πέδηση, τα δεδομένα από την οθόνη του παιχνιδιού και τα δεδομένα από τη τηλεμετρία.

Η συλλογή δεδομένων από την οθόνη γίνεται με τη χρήση της βιβλιοθήκης `opencv`, ενώ η συλλογή των δεδομένων από τις συσκευές εισόδου με τη βοήθεια της `pygame`. Για τη λήψη των δεδομένων τηλεμετρίας τώρα, χρησιμοποιείται η δυνατότητα του παιχνιδιού να εξαγάγει τα δεδομένα αυτά σε ένα socket σε κάποια συγκεκριμένη τοπική διεύθυνση.

Για κάθε ένα από τα δεδομένα τα οποία διαβάζονται από τις συσκευές εισόδου υπάρχει ένα script το οποίο διαβάζει τα δεδομένα αυτά και τα αποθηκεύει σε κατάλληλα αρχεία. Έπειτα το script `read_all_data.py` [A1] αφού διαβάσει τα δεδομένα από την οθόνη και από τη τηλεμετρία, διαβάζει και τα δεδομένα που είναι αποθηκευμένα στα αρχεία και τα αποθηκεύει όλα μαζί χρονισμένα σε νέα αρχεία με τη μορφή `numpy array` (`npz`).

Με τη τεχνική αυτή επιτυγχάνεται παραλληλισμός μέσω διαφορετικών διεργασιών.

Τα δεδομένα που συλλέχθηκαν και χρησιμοποιήθηκαν για την εκπαίδευση και αξιολόγηση των μοντέλων είχαν μέγεθος 6.1GB και αποτελούνταν από 38000 frames.

7.1.3 Πρακτικές κανονικοποίησης δεδομένων

Μιας και το μέγεθος των δεδομένων είναι σχετικά μεγάλο, η διαδικασία της κανονικοποίησης χωρίζεται σε κάποια στάδια έτσι ώστε να αποφευχθούν προβλήματα μνήμης στο σύστημα. Αν υπολογίσουμε μάλιστα ότι το μέγεθος των `numpy arrays` τα οποία χρησιμοποιούνται σχεδόν παντού σε αυτή την εργασία καταλαμβάνουν περισσότερο χώρο στη μνήμη, τότε είναι σχεδόν σίγουρο ότι θα υπάρχει πρόβλημα.

Το πρώτο στάδιο της διαδικασίας είναι η εύρεση των μέσων τιμών και των τυπικών

αποκλίσεων των δεδομένων με τη διαδικασία που περιγράφηκε στο Κεφάλαιο 4 [A3]. Οι τιμές αυτές αποθηκεύονται και χρησιμοποιούνται έπειτα στο αρχείο [A4] το οποίο παράγει νέα αρχεία δεδομένων τα οποία περιέχουν πλέον τα κανονικοποιημένα δεδομένα σε μορφή έτοιμη για να διαβαστεί από το νευρωνικό δίκτυο. Αυτή η εκ νέου αποθήκευση των δεδομένων σε κανονικοποιημένη μορφή γίνεται για να αποφευχθεί η κανονικοποίηση κατά τη διαδικασία της εκπαίδευσης, κάτι που θα ήταν ιδιαίτερο δαπανηρό. Ο κύριος λόγος για το επιπλέον κόστος που θα είχε μία τέτοια προσέγγιση είναι το γεγονός ότι τα δεδομένα πρέπει να μετατραπούν από μία εικόνα 100x250x3 συν άλλα τρία στοιχεία (x,y,speed) που είναι αρχικά, σε ένα τανυστή μεγέθους 100x250x6.

7.1.4 Εργαλείο αξιολόγησης

Το εργαλείο αξιολόγησης, το οποίο παρουσιάστηκε στη παράγραφο 6.1.4, αναπτύχθηκε με τη χρήση της βιβλιοθήκης qt, και συγκεκριμένα στην έκδοση για python που ονομάζεται PyQt.

Μετά από την επιλογή του μοντέλου και των δεδομένων δοκιμής πρέπει στο ίδιο παράθυρο να εμφανίζονται ταυτόχρονα τρεις διαφορετικές εικόνες, μία με τα αρχικά δεδομένα και οι δύο με τα κανονικοποιημένα. Για να επιτευχθεί αυτό δημιουργείται ένα thread το οποίο είναι υπεύθυνο για να φορτώνει ένα batch δεδομένων, να τα στέλνει προς κανονικοποίηση και στη συνέχεια να εμφανίζει τη κάθε εικόνα στη κατάλληλη θέση. Επειδή το widget του qt το οποίο εμφανίζει εικόνες μπορεί να διαβάζει εικόνες μόνο από αρχεία, μέσω της βιβλιοθήκης PIL η κάθε φωτογραφία αποθηκεύεται κάθε στιγμή στο δίσκο και έπειτα διαβάζεται από το widget της qt. Το overhead που δημιουργείται είναι αμελητέο.

7.1.5 Βιβλιοθήκες

Στον παρακάτω πίνακα αναφέρονται κάποιες από τις python βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας.

Όνομα βιβλιοθήκης	Χρησιμότητα
tensorflow	Framework της google για machine learning
tflearn	Βιβλιοθήκη πάνω από tensorflow η οποία αυτοματοποιεί κάποιες διαδικασίες
numpy	Βιβλιοθήκη επιστημονικού υπολογισμού
Cv2 (opencv)	Βιβλιοθήκη για υπολογιστική όραση
PyQt5	Βιβλιοθήκη για γραφικό περιβάλλον
Pygame	Για διάβασμα τιμών από συσκευές εισόδου

Πίνακας 9: Βιβλιοθήκες που χρησιμοποιήθηκαν

8.

ΙΔΕΕΣ ΓΙΑ ΜΕΛΛΟΝΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Η εργασία αυτή αποτελεί το πρώτο βήμα προς τη λύση του μεγάλου προβλήματος της ολοκληρωτικής αυτόνομης οδήγησης με ένα τέτοιου είδους δυσκολίας περιβάλλον προσομοίωσης. Η δυσκολία έγκειται στο ότι το περιβάλλον είναι απρόβλεπτο και η οδήγηση γρήγορα και χωρίς ηλεκτρονικά βοηθήματα και στο ότι η λήψη αποφάσεων γίνεται κατά κύριο λόγο με τα pixels της οθόνης.

Το επόμενο βήμα, που μόνο για τεχνικούς λόγους δεν ολοκληρώθηκε εδώ είναι η εφαρμογή των τιμών που προβλέπει το μοντέλο στο πραγματικό παιχνίδι. Αυτή η λειτουργία θα έδινε μία ακόμα καλύτερη οπτικοποίηση των αποτελεσμάτων.

Ένα άλλο σημαντικό βήμα το οποίο θα μπορούσε να συντελεστεί είναι η χρήση αλγορίθμων reinforcement learning. Αλγόριθμοι τέτοιου τύπου θα μπορούσαν να εξερευνήσουν μόνοι τους το περιβάλλον χωρίς απαραίτητα να χρειάζονται δεδομένα εκπαίδευσης. Κάτι τέτοιο πιθανόν να έδινε πολύ γρήγορους και ανταγωνιστικούς agents. Οι συγκεκριμένοι αλγόριθμοι ήταν αδύνατον να υλοποιηθούν στη συγκεκριμένη εργασία μιας και το συγκριμένο περιβάλλον προσομοίωσης δε μπορεί να προσομοιώσει τη διαδικασία του παιχνιδιού σε λειτουργία fast forward, πράγμα που σημαίνει ότι κάτι τέτοιοι θα χρειαζόταν πάρα πολύ χρόνο να εκπαιδευτεί ιδιαίτερα να λάβουμε υπ' όψιν το hardware το οποίο ήταν διαθέσιμο.

Σαν συνέχεια των παραπάνω θα ήταν αρκετά εντυπωσιακό ένα πρωτάθλημα στο οποίο θα ανταγωνίζονται λάτρεις των αγώνων και της τεχνητής νοημοσύνης σε συνθήκες πραγματικού αγώνα σε online lobbies, έχοντας στη διάθεσή τους μόνο βασικά δεδομένα όπως αυτά που χρησιμοποιήθηκαν και σε αυτή την εργασία.



ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Self-driving cars are powered by video game technology
<https://thenextweb.com/cars/2017/07/05/self-driving-cars-powered-video-game-technology/>
- [2] How video games helped give us the self-driving car
https://www.washingtonpost.com/news/the-switch/wp/2017/07/05/how-video-games-helped-give-us-the-self-driving-car/?noredirect=on&utm_term=.a883ab099de6
- [3] Universal approximation theorem
https://en.wikipedia.org/wiki/Universal_approximation_theorem
- [4] Merging chrominance and luminance in early, medium, and late fusion using Convolutional Neural Networks
<http://users.ics.aalto.fi/perellm1/thesis/pdf/mscthesis.pdf>
- [5] Google inception model
<https://github.com/google/inception>
- [6] Random walk initialization for training very deep feed forward networks
<https://arxiv.org/pdf/1412.6558.pdf>
- [7] Self-Normalizing Neural Networks
<https://arxiv.org/pdf/1706.02515.pdf>
- [8] Adam: A method for stochastic optimization
<https://arxiv.org/pdf/1412.6980.pdf>
- [9] Analysis of dropouts
<https://pgaleone.eu/deep-learning/regularization/2017/01/10/anaysis-of-dropout/>
- [10] A systematic study of the class imbalance problem in convolutional neural networks
<https://arxiv.org/pdf/1710.05381.pdf>

B. ■ ΚΩΔΙΚΑΣ

Στο παράρτημα αυτό υπάρχουν κάποιοι κώδικες της εργασίας που κρίνονται σημαντικοί.

B.1 read_all_data.py

```
from __future__ import print_function
#telemetry staff
import socket
from structs import UDPPacket
import ctypes

import numpy as np
from PIL import ImageGrab
import cv2
import time
from grabscreen import grab_screen
import os

def set_telemetry(packet):

    telemetry_values = {}
    telemetry_values['x'] = packet.x
    telemetry_values['y'] = packet.y
    telemetry_values['z'] = packet.z
    telemetry_values['speed'] = packet.speed
    telemetry_values['xv'] = packet.xv
    telemetry_values['yv'] = packet.yv
    telemetry_values['zv'] = packet.zv
    telemetry_values['xr'] = packet.xr
    telemetry_values['yr'] = packet.yr
    telemetry_values['zr'] = packet.zr
    telemetry_values['xd'] = packet.xd
    telemetry_values['yd'] = packet.yd
    telemetry_values['zd'] = packet.zd
```

```

telemetry_values['sRL'] = packet.susp_pos[0]
telemetry_values['sRR'] = packet.susp_pos[1]
telemetry_values['sFL'] = packet.susp_pos[2]
telemetry_values['sFR'] = packet.susp_pos[3]
telemetry_values['svRL'] = packet.susp_vel[0]
telemetry_values['svRR'] = packet.susp_vel[1]
telemetry_values['svFL'] = packet.susp_vel[2]
telemetry_values['svFR'] = packet.susp_vel[3]
telemetry_values['wsRL'] = packet.wheel_speed[0]
telemetry_values['wsRR'] = packet.wheel_speed[1]
telemetry_values['wsFL'] = packet.wheel_speed[2]
telemetry_values['wsFR'] = packet.wheel_speed[3]
telemetry_values['yaw'] = packet.yaw
telemetry_values['pitch'] = packet.pitch
telemetry_values['roll'] = packet.roll
telemetry_values['xlv'] = packet.x_local_velocity
telemetry_values['ylv'] = packet.y_local_velocity
telemetry_values['zlv'] = packet.z_local_velocity
telemetry_values['aax'] = packet.ang_acc_x
telemetry_values['aay'] = packet.ang_acc_y
telemetry_values['aaz'] = packet.ang_acc_z
telemetry_values['throttle'] = packet.throttle
telemetry_values['steer'] = packet.steer
telemetry_values['brake'] = packet.brake

return telemetry_values

def get_packet(address, port):
    """
    Recieve a single UDP telemetry packet from the specified port and ip address

    :param address: IP address for the socket
    :param port: Port for the socket
    :return: A UDPPacket
    """
    # create a socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    # bind the socket to the specified ip address and port
    sock.bind((address, port))
    # recieve data
    data, addr = sock.recvfrom(ctypes.sizeof(UDPPacket))
    # convert from raw bytes to UDPPacket structure
    return UDPPacket.from_buffer_copy(data)

def get_telemetry(address, port):
    """
    Generator function which yields UDPPackets from the specified ip address and port

```

```

:param address: IP address for receiving packets
:param port: Port on which to receive packets
:yield: a UDPPacket for each udp packet received
"""

last_packet = None
while True:
    packet = get_packet(address, port)
    if last_packet is None or packet.time > last_packet.time:
        yield packet
        last_packet = packet

# countdown
for i in list(range(4))[:-1]:
    print(i + 1)
    time.sleep(1)

training_data = []
last_time = time.time()

starting_value = 1

SAVED_FILE_NAME = '6990_telemetry_training_data-{}.npy'

while True:
    file_name = SAVED_FILE_NAME.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

while (True):

    screen = grab_screen(region=(0, 100, 1000, 740))

    screen = cv2.cvtColor(screen, cv2.COLOR_BGR2RGB)

    # resize frame
    screen = cv2.resize(screen, (250,250))

    # region of interest
    #screen = screen [78:150,0:250]

    #new region of interest
    screen = screen [70:290, 0:250]

    loop_time = time.time() - last_time

```



```
for packet in get_telemetry("20777):

    telemetry_values = set_telemetry(packet)

    break

# read steering
f = open("ReadData/current_steering.txt", "r")
for line in f:
    steering = line

# f = open("ReadData/current_throttle.txt", "r")
# for line in f:
#     throttle = line

# f = open("ReadData/current_brakes.txt", "r")
# for line in f:
#     brakes = line

current_controls = {}
current_controls['steering'] = steering
# current_controls['throttle'] = throttle
# current_controls['brakes'] = brakes

training_data.append([screen,telemetry_values,current_controls])

last_time = time.time()

#save the training data 500 at a time
if len(training_data) % 500 == 0:

    if len(training_data) == 500:
        np.save(file_name,training_data)
        print('SAVED')
        training_data = []
        starting_value += 1
        file_name = SAVED_FILE_NAME.format(starting_value)

#quit with 'q'
if cv2.waitKey(25) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    break
```

B.2 read_steering.py

```
import pygame
pygame.init()

def main():

    joysticks = []
    clock = pygame.time.Clock()
    keepPlaying = True
    χαχα
    # for all the connected joysticks
    for i in range(0, pygame.joystick.get_count()):
        # create an Joystick object in our list
        joysticks.append(pygame.joystick.Joystick(i))
        # initialize them all (-1 means loop forever)
        joysticks[-1].init()
        # print a statement telling what the name of the controller is
        print ("Detected joystick " + joysticks[-1].get_name(), "")

    while keepPlaying:
        clock.tick(60)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                print ("Received event 'Quit', exiting.")
                keepPlaying = False

            elif event.type == pygame.JOYAXISMOTION:
                if (event.axis == 0):
                    with open('current_steering.txt', 'w') as file:
                        file.write(str(event.value))

main()
pygame.quit()
```

B.3 Normalization.py

```
import pygame
pygame.init()

def main():

    joysticks = []
    clock = pygame.time.Clock()
    keepPlaying = True
    χαχα
    # for all the connected joysticks
    for i in range(0, pygame.joystick.get_count()):
        # create an Joystick object in our list
        joysticks.append(pygame.joystick.Joystick(i))
        # initialize them all (-1 means loop forever)
        joysticks[-1].init()
        # print a statement telling what the name of the controller is
        print ("Detected joystick " + joysticks[-1].get_name(), "")

    while keepPlaying:
        clock.tick(60)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                print ("Received event 'Quit', exiting.")
                keepPlaying = False

            elif event.type == pygame.JOYAXISMOTION:
                if (event.axis == 0):
                    with open('current_steering.txt', 'w') as file:
                        file.write(str(event.value))

main()
pygame.quit()
```

B.4 create_data_batches.py

```

import numpy as np
import math

for counter in range(10,70,10):

    counter = 76

    print(str(counter))
    print("loading data...")

    #load all data in RAM (~5GB)
    TRAINING_DATA_FILES = counter
    data = np.load("data/telemetry_training_data-{}.npz".format(counter-6))

    #data = np.load("normalized_data-1.npz")

    for i in range(counter-5,TRAINING_DATA_FILES + 1 ):
        data = np.append(data,np.load('data/telemetry_training_data-{}.npz'.format(i)), axis=0 )

    data_len = len(data)

    #keep only the interesting part (ROI)
    for i in range(data_len):
        data[i][0] = data[i][0][:100]

    #pre calculated means and stds for the whole dataset

    sum1 = 91461941044.18457
    std_sum1 = 1875687355110.2168

    sum2 = 123215856697.59067
    std_sum2 = 113146464589.96437

    sum3 = 117359336885.57675
    std_sum3 = 128011048786.3762

    x_sum = 2030407.3156801462
    x_std_sum = 4384368263.496551

    speed_sum = 2087542.642747879
    speed_std_sum = 10022033.821311038

    y_sum = -586132.7977940273
    y_std_sum = 12510483.537866063

    dataset_data_len = 38000

    mean1 = sum1 / (dataset_data_len*100*250)
    std_1 = math.sqrt(std_sum1/(dataset_data_len*100*250))

```

```

mean2 = sum2 / (dataset_data_len*100*250)
std_2 = math.sqrt(std_sum2/(dataset_data_len*100*250))

mean3 = sum3 / (dataset_data_len*100*250)
std_3 = math.sqrt(std_sum3/(dataset_data_len*100*250))

speed_mean = speed_sum / dataset_data_len
speed_std = math.sqrt(speed_std_sum/dataset_data_len)

x_mean = x_sum / dataset_data_len
x_std = math.sqrt(x_std_sum/dataset_data_len)

y_mean = y_sum / dataset_data_len
y_std = math.sqrt(y_std_sum/dataset_data_len)

#normalize images
for i in range(data_len):
    data[i][0] = np.array(data[i][0], dtype='float32')

    data[i][0][...,0] = ( data[i][0][...,0] - mean1 ) / std_1
    data[i][0][...,1] = ( data[i][0][...,1] - mean2 ) / std_2
    data[i][0][...,2] = ( data[i][0][...,2] - mean3 ) / std_3

    data[i][1]['speed'] = ( data[i][1]['speed'] - speed_mean ) / speed_std
    data[i][1]['x'] = ( data[i][1]['x'] - x_mean ) / x_std
    data[i][1]['y'] = ( data[i][1]['y'] - y_mean ) / y_std

HEIGHT = 100
WIDTH = 250

X = []

for i in range(data_len):
    if (i%200 == 0):
        print(str(i) + " \ " + str(data_len))
    cur_x = data[i][1]['x']
    cur_y = data[i][1]['y']
    cur_speed = data[i][1]['speed']
    x_channel = np.zeros( (HEIGHT,WIDTH) )
    y_channel = np.zeros( (HEIGHT,WIDTH) )
    speed_channel = np.zeros( (HEIGHT,WIDTH) )
    for h in range(HEIGHT):
        for w in range(WIDTH):
            x_channel[h][w] = cur_x
            y_channel[h][w] = cur_y
            speed_channel[h][w] = cur_speed
    temp = np.concatenate( ( data[i][0], np.expand_dims(x_channel,axis=2) ), axis=2 )
    temp = np.concatenate( (temp, np.expand_dims(y_channel,axis=2)), axis=2 )
    temp = np.concatenate( (temp, np.expand_dims(speed_channel,axis=2)), axis=2 )
    X.append(temp)

```

```

print("converting X to numpy array")

X = np.array(X)

Y = []

for i in range(data_len):
    currency_Y = []
    currency_Y = np.array(currency_Y)
    for key,value in sorted(data[i][2].items()):
        currency_Y = np.append(currency_Y, value)
    Y.append(np.array(currency_Y))

Y = np.array(Y, dtype='float32')

#keep only the sterring for this model (for now)

#new_Y = []

#for i in range(len(Y)):
#    new_Y.append(Y[i][1])

#new_Y = np.array(new_Y)

#YY = []

#for i in range(len(new_Y)):
#    temp = []
#    temp.append(new_Y[i])
#    temp = np.array(temp)
#    YY.append(temp)

#YY = np.array(YY)

np.save("X2_norm_final_data_{}.np".format(counter+4/10), X)
np.save("Y2_norm_final_data_{}.np".format(counter+4/10), Y)

data = {}

break

```

B.5 train_model.py

```
import numpy as np
from random import shuffle

from multinomal import model_fc_2

VERSION = 2
TRAINING_DATA_FILES = 8
WIDTH = 250
HEIGHT = 100
LR = 1e-3
EPOCHS = 30

MODEL_NAME= 'final_model_001-formula-1-car_v{}-{}-{}'.format(VERSION,LR,EPOCHS)

print("loading data...")

#CHANGE THIS IN EVERY PHASE
file_counter = 6300
model = model_fc_2(output=3)

for i in range(1,7):

    print(file_counter)

    print(str(i) + "/6")

    #load all data in RAM (~5GB)
    X = None
    Y = None
    X = np.load("X2_norm_final_data_{}.0.npy".format(i))
    Y = np.load("Y2_norm_final_data_{}.0.npy".format(i))

    #custom data shuffling
    combined = list(zip(X,Y))
    shuffle(combined)
    X[:, Y[:]] = zip(*combined)

    #80% for train
    split_data_boundry = int(len(X) * 0.8)

    #LOAD previews trained model on the fly
    if i != 1:
        model.load("model_spirosnet-{}".format(file_counter))
        file_counter += 6300
```

```

model.fit(X[:split_data_boundry],Y[:split_data_boundry], n_epoch=EPOCHS, validation_set=(
X[split_data_boundry:], Y[split_data_boundry:]),
        shuffle = False,snapshot_step=2500, show_metric=True, run_id=MODEL_NAME)

```

B.6 model.py

```

import tflearn
import tensorflow as tf

from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.conv import conv_2d

def model_fc_2(output=1):

    network = input_data(shape=[None, 100, 250, 6])

    part_one = network[...,:3]
    part_two = network[:,3:]

    network1 = conv_2d(part_one, 24, 5, strides=5, activation='elu')
    network2 = conv_2d(part_two, 24, 5, strides=5, activation='elu')

    network = tflearn.merge([network1, network2], 'concat')

    #from basic network
    #network = conv_2d(network, 24, 5, strides=5, activation='elu' )
    network = conv_2d(network, 36, 5, strides=5, activation='elu' )
    network = conv_2d(network, 48, 5, strides=5, activation='elu' )
    network = conv_2d(network, 64, 3, strides=3, activation='elu' )
    network = conv_2d(network, 64, 3, strides=3, activation='elu' )

    network = dropout(network, 0.5)

    network = fully_connected(network, 100, activation='elu')
    network = fully_connected(network, 50, activation='elu')
    network = fully_connected(network, 10, activation='elu')
    network = fully_connected(network, output, activation='linear')

    #momentum = tflearn.Momentum(learning_rate=0.00001, lr_decay=0.96, decay_step=1500)
    #network = tflearn.regression(network, optimizer=momentum, loss='huber_loss',

```



```

metric=None)

network = tflearn.regression(network, optimizer='adam',
                             loss='huber_loss', metric=None, learning_rate = 0.0001)

model = tflearn.DNN(network, checkpoint_path='model_spiroset',
                    max_checkpoints=1, tensorboard_verbose=2, tensorboard_dir='log')

return model

```

B.7 evaluation_screen.py (evaluation tool)

```

from PyQt5 import QtCore, QtGui, QtWidgets
import numpy as np
from PIL import Image
import time
import math

import threading
# from models.final_model_001.multinomial import model_fc_2

def RGB2YUV( rgb ):

    m = np.array([[ 0.29900, -0.16874,  0.50000],
                  [ 0.58700, -0.33126, -0.41869],
                  [ 0.11400, 0.50000, -0.08131]])

    yuv = np.dot(rgb,m)
    yuv[:,1:] += 128.0
    return yuv

sum1 = 91461941044.18457
std_sum1 = 1875687355110.2168

sum2 = 123215856697.59067
std_sum2 = 113146464589.96437

sum3 = 117359336885.57675
std_sum3 = 128011048786.3762

x_sum = 2030407.3156801462
x_std_sum = 4384368263.496551

speed_sum = 2087542.642747879

```

```

speed_std_sum = 10022033.821311038

y_sum = -586132.7977940273
y_std_sum = 12510483.537866063

dataset_data_len = 38000

mean1 = sum1 / (dataset_data_len*100*250)
std_1 = math.sqrt(std_sum1/(dataset_data_len*100*250))

mean2 = sum2 / (dataset_data_len*100*250)
std_2 = math.sqrt(std_sum2/(dataset_data_len*100*250))

mean3 = sum3 / (dataset_data_len*100*250)
std_3 = math.sqrt(std_sum3/(dataset_data_len*100*250))

speed_mean = speed_sum / dataset_data_len
speed_std = math.sqrt(speed_std_sum/dataset_data_len)

x_mean = x_sum / dataset_data_len
x_std = math.sqrt(x_std_sum/dataset_data_len)

y_mean = y_sum / dataset_data_len
y_std = math.sqrt(y_std_sum/dataset_data_len)

def normalize_data(screen, telemetry_values):

    HEIGHT = 100
    WIDTH = 250

    screen = np.array(screen)

    screen[...,0] = ( screen[...,0] - mean1 ) / std_1
    screen[...,1] = ( screen[...,1] - mean2 ) / std_2
    screen[...,2] = ( screen[...,2] - mean3 ) / std_3

    #normalize the features
    telemetry_values['speed'] = ( telemetry_values['speed'] - speed_mean ) / speed_std
    telemetry_values['x'] = ( telemetry_values['x'] - x_mean ) / x_std
    telemetry_values['y'] = ( telemetry_values['y'] - y_mean ) / y_std

    #create the X values (screen with x,y,speed channels)
    X = []
    x_channel = np.zeros( (HEIGHT,WIDTH) )
    y_channel = np.zeros( (HEIGHT,WIDTH) )
    speed_channel = np.zeros( (HEIGHT,WIDTH) )
    for h in range(HEIGHT):
        for w in range(WIDTH):
            x_channel[h][w] = telemetry_values['x']
            y_channel[h][w] = telemetry_values['y']
            speed_channel[h][w] = telemetry_values['speed']

    temp = np.concatenate( ( screen , np.expand_dims(x_channel,axis=2) ), axis=2 )
    temp = np.concatenate( (temp, np.expand_dims(y_channel,axis=2)), axis=2 )
    temp = np.concatenate( (temp, np.expand_dims(speed_channel,axis=2)), axis=2 )
    X.append(temp)

```

```
X = np.array(X)

return X

class Ui_MainWindow_Evaluation(object):

    def __init__(self, model, testing_file):

        self.model = model
        self.testing_file = testing_file

        try:
            pre_load_thread = threading.Thread(target=self.load_model_and_data)
            pre_load_thread.start()
        except (KeyboardInterrupt, SystemExit):
            cleanup_stop_thread()
            sys.exit()

    def load_model_and_data(self):

        #load model
        package = "{ }.multinomial".format(self.model.replace("/", "."))
        name = "model_fc_2"

        the_model = getattr(__import__(package, fromlist=[name]), name)
        self.model_dnn = the_model(3)
        self.model_dnn.load(str(self.model)+"/model_spiroset-11340")

        self.label_2.setText("Live Input Data")

        try:
            t1 = threading.Thread(target=self.runLiveOriginalImage)
            t1.start()

        except (KeyboardInterrupt, SystemExit):
            cleanup_stop_thread()
            sys.exit()

    def runLiveOriginalImage(self):

        telemetry_values = {}
```

```

sumA = 0
sumB = 0
sumC = 0

videos = 5
for j in range(1,videos+1):

    self.testing = np.load(self.testing_file.replace("1", str(j)))

    for i in range(len(self.testing)):

        img = Image.fromarray(self.testing[i][0][:100], 'RGB')
        img.save('current_image.png')

        telemetry_values['x'] = self.testing[i][1]['x']
        telemetry_values['y'] = self.testing[i][1]['y']
        telemetry_values['speed'] = self.testing[i][1]['speed']

        X = normalize_data(self.testing[i][0][:100], telemetry_values)

        game_image = Image.fromarray(X[0][...,3].astype('uint8'))
        features_image = Image.fromarray(X[0][...,3:].astype('uint8'))

        game_image.save('current_image_norm.png')
        features_image.save('features_image.png')

        self.norm_image.setPixmap(
            QtGui.QPixmap("current_image_norm.png"))

        self.norm_features.setPixmap(
            QtGui.QPixmap("features_image.png"))

        self.original_image.setPixmap(
            QtGui.QPixmap("current_image.png"))

        #real values
        real_steering = self.testing[i][2]['steering']
        real_throttle = self.testing[i][2]['throttle']
        real_braking = self.testing[i][2]['brakes']

        #predicted values
        prediction = self.model_dnn.predict([X.reshape(100,250,6)])[0]

        prediction[0] = - prediction[0]
        prediction[2] = - prediction[2]
        for k in range(3):
            if prediction[k] > 1:
                prediction[k] = 1
            if prediction[k] < -1:
                prediction[k] = -1

        #set them in the ui
        #sorry for the naming :)
        self.label.setText("Real Steering: " + real_steering)
        self.label_6.setText("Real Throttle: " + real_throttle)
        self.label_8.setText("Real Brakes:" + real_braking)

```

```

self.label_3.setText("Predicted Steering: " + str(prediction[1]))
self.label_7.setText("Predicted Throttle:" + str(prediction[2]))
self.label_9.setText("Predicted Brakes:" + str(prediction[0]))

#differences
se = abs(float(prediction[1]) - float(real_steering) )
te = abs(float(prediction[2]) - float(real_throttle))
be = abs( float(prediction[0]) - float(real_braking))

self.label_10.setText("Steering Error: " + str(se) )
self.label_11.setText("Throttle Error: " + str(te) )
self.label_12.setText("Braking Error: " + str(be) )

sumA += se
sumB += te
sumC += be

avgA = se/(i*j+1)
avgB = te/(i*j+1)
avgC = be/(i*j+1)

self.label_13.setText("Mean Error: " + str((avgA+avgB+avgC)/3) )

time.sleep(0.1)

def center(self, window):
    qr = window.frameGeometry()
    cp = QDesktopWidget().availableGeometry().center()
    qr.moveCenter(cp)
    window.move(qr.topLeft())

def setup(self, MainWindow):
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(1280, 805)
    self.centralwidget = QtWidgets.QWidget(MainWindow)
    self.centralwidget.setObjectName("centralwidget")
    self.horizontalLayout_2 = QtWidgets.QHBoxLayout(self.centralwidget)
    self.horizontalLayout_2.setObjectName("horizontalLayout_2")
    self.dockWidget_6 = QtWidgets.QDockWidget(self.centralwidget)
    self.dockWidget_6.setMaximumSize(QtCore.QSize(500, 700))
    self.dockWidget_6.setObjectName("dockWidget_6")
    self.dockWidgetContents_10 = QtWidgets.QWidget()
    self.dockWidgetContents_10.setObjectName("dockWidgetContents_10")
    self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.dockWidgetContents_10)
    self.verticalLayout_2.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)
    self.verticalLayout_2.setObjectName("verticalLayout_2")
    self.verticalLayout = QtWidgets.QVBoxLayout()
    self.verticalLayout.setObjectName("verticalLayout")
    self.label = QtWidgets.QLabel(self.dockWidgetContents_10)
    font = QtGui.QFont()

```

```

font.setPointSize(14)
self.label.setFont(font)
self.label.setObjectName("label")
self.verticalLayout.addWidget(self.label)
self.label_3 = QtWidgets.QLabel(self.dockWidgetContents_10)
font = QtGui.QFont()
font.setPointSize(14)
self.label_3.setFont(font)
self.label_3.setObjectName("label_3")
self.verticalLayout.addWidget(self.label_3)
self.line = QtWidgets.QFrame(self.dockWidgetContents_10)
self.line.setFrameShape(QtWidgets.QFrame.HLine)
self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line.setObjectName("line")
self.verticalLayout.addWidget(self.line)
self.label_6 = QtWidgets.QLabel(self.dockWidgetContents_10)
font = QtGui.QFont()
font.setPointSize(14)
self.label_6.setFont(font)
self.label_6.setObjectName("label_6")
self.verticalLayout.addWidget(self.label_6)
self.label_7 = QtWidgets.QLabel(self.dockWidgetContents_10)
font = QtGui.QFont()
font.setPointSize(14)
self.label_7.setFont(font)
self.label_7.setObjectName("label_7")
self.verticalLayout.addWidget(self.label_7)
self.line_4 = QtWidgets.QFrame(self.dockWidgetContents_10)
self.line_4.setFrameShape(QtWidgets.QFrame.HLine)
self.line_4.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line_4.setObjectName("line_4")
self.verticalLayout.addWidget(self.line_4)
self.label_8 = QtWidgets.QLabel(self.dockWidgetContents_10)
font = QtGui.QFont()
font.setPointSize(14)
self.label_8.setFont(font)
self.label_8.setObjectName("label_8")
self.verticalLayout.addWidget(self.label_8)
self.label_9 = QtWidgets.QLabel(self.dockWidgetContents_10)
font = QtGui.QFont()
font.setPointSize(14)
self.label_9.setFont(font)
self.label_9.setObjectName("label_9")
self.verticalLayout.addWidget(self.label_9)
self.line_5 = QtWidgets.QFrame(self.dockWidgetContents_10)
self.line_5.setFrameShape(QtWidgets.QFrame.HLine)
self.line_5.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line_5.setObjectName("line_5")
self.verticalLayout.addWidget(self.line_5)
self.label_10 = QtWidgets.QLabel(self.dockWidgetContents_10)
self.label_10.setObjectName("label_10")
self.verticalLayout.addWidget(self.label_10)
self.label_11 = QtWidgets.QLabel(self.dockWidgetContents_10)
self.label_11.setObjectName("label_11")
self.verticalLayout.addWidget(self.label_11)
self.label_12 = QtWidgets.QLabel(self.dockWidgetContents_10)

```

```
self.label_12.setObjectName("label_12")
self.verticalLayout.addWidget(self.label_12)
self.line_6 = QtWidgets.QFrame(self.dockWidgetContents_10)
self.line_6.setFrameShape(QtWidgets.QFrame.HLine)
self.line_6.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line_6.setObjectName("line_6")
self.verticalLayout.addWidget(self.line_6)
self.label_13 = QtWidgets.QLabel(self.dockWidgetContents_10)
self.label_13.setObjectName("label_13")

font = QtGui.QFont()
font.setPointSize(17)
self.label_13.setFont(font)

self.verticalLayout.addWidget(self.label_13)
self.verticalLayout_2.addLayout(self.verticalLayout)
self.dockWidget_6.setWidget(self.dockWidgetContents_10)
self.horizontalLayout_2.addWidget(self.dockWidget_6)
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 1280, 28))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)
self.dockWidget_4 = QtWidgets.QDockWidget(MainWindow)
self.dockWidget_4.setObjectName("dockWidget_4")
self.dockWidgetContents_8 = QtWidgets.QWidget()
self.dockWidgetContents_8.setObjectName("dockWidgetContents_8")
self.gridLayout_9 = QtWidgets.QGridLayout(self.dockWidgetContents_8)
self.gridLayout_9.setObjectName("gridLayout_9")
self.stackedWidget = QtWidgets.QStackedWidget(self.dockWidgetContents_8)
self.stackedWidget.setMinimumSize(QtCore.QSize(650, 500))
self.stackedWidget.setMaximumSize(QtCore.QSize(900, 700))
self.stackedWidget.setFrameShape(QtWidgets.QFrame.Panel)
self.stackedWidget.setFrameShadow(QtWidgets.QFrame.Plain)
self.stackedWidget.setLineWidth(1)
self.stackedWidget.setObjectName("stackedWidget")
self.inventory = QtWidgets.QWidget()
self.inventory.setObjectName("inventory")
self.verticalLayout_4 = QtWidgets.QVBoxLayout(self.inventory)
self.verticalLayout_4.setObjectName("verticalLayout_4")
self.label_2 = QtWidgets.QLabel(self.inventory)
self.label_2.setMinimumSize(QtCore.QSize(0, 0))
self.label_2.setMaximumSize(QtCore.QSize(900, 100))
font = QtGui.QFont()
font.setPointSize(26)
self.label_2.setFont(font)
self.label_2.setScaledContents(False)
self.label_2.setAlignment(QtCore.Qt.AlignCenter)
self.label_2.setWordWrap(True)
self.label_2.setObjectName("label_2")
```

```

self.verticalLayout_4.addWidget(self.label_2)
self.label_4 = QtWidgets.QLabel(self.inventory)
font = QtGui.QFont()
font.setPointSize(14)
self.label_4.setFont(font)
self.label_4.setObjectName("label_4")
self.verticalLayout_4.addWidget(self.label_4)
self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
self.horizontalLayout_4.setContentsMargins(200, -1, 200, -1)
self.horizontalLayout_4.setObjectName("horizontalLayout_4")
self.norm_image = QtWidgets.QLabel(self.inventory)
self.norm_image.setMinimumSize(QtCore.QSize(0, 50))
self.norm_image.setMaximumSize(QtCore.QSize(16777215, 16777215))
self.norm_image.setText("")
self.norm_image.setPixmap(QtGui.QPixmap("../images/wemall.png"))
self.norm_image.setScaledContents(False)
self.norm_image.setObjectName("norm_image")
self.norm_image.setScaledContents(True)
self.horizontalLayout_4.addWidget(self.norm_image)
self.verticalLayout_4.addLayout(self.horizontalLayout_4)
self.horizontalLayout_10 = QtWidgets.QHBoxLayout()
self.horizontalLayout_10.setContentsMargins(200, -1, 200, -1)
self.horizontalLayout_10.setObjectName("horizontalLayout_10")
self.norm_features = QtWidgets.QLabel(self.inventory)
self.norm_features.setMinimumSize(QtCore.QSize(0, 50))
self.norm_features.setMaximumSize(QtCore.QSize(16777215, 16777215))
self.norm_features.setText("")
self.norm_features.setPixmap(QtGui.QPixmap("../images/wemall.png"))
self.norm_features.setScaledContents(False)
self.norm_features.setScaledContents(True)
self.norm_features.setObjectName("norm_features")
self.horizontalLayout_10.addWidget(self.norm_features)
self.verticalLayout_4.addLayout(self.horizontalLayout_10)
self.label_5 = QtWidgets.QLabel(self.inventory)
font = QtGui.QFont()
font.setPointSize(14)
self.label_5.setFont(font)
self.label_5.setObjectName("label_5")
self.verticalLayout_4.addWidget(self.label_5)
self.horizontalLayout_9 = QtWidgets.QHBoxLayout()
self.horizontalLayout_9.setContentsMargins(200, -1, 200, 20)
self.horizontalLayout_9.setObjectName("horizontalLayout_9")
self.original_image = QtWidgets.QLabel(self.inventory)
self.original_image.setMinimumSize(QtCore.QSize(0, 50))
self.original_image.setMaximumSize(QtCore.QSize(16777215, 16777215))
self.original_image.setText("")
self.original_image.setPixmap(QtGui.QPixmap("../images/welove_original.png"))
self.original_image.setScaledContents(False)
self.original_image.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|
QtCore.Qt.AlignVCenter)
self.original_image.setObjectName("original_image")
self.original_image.setScaledContents(True)
self.horizontalLayout_9.addWidget(self.original_image)
self.verticalLayout_4.addLayout(self.horizontalLayout_9)
self.verticalLayout_4.setStretch(0, 1)
self.stackedWidget.addWidget(self.inventory)

```



```

self.gridLayout_9.addWidget(self.stackedWidget, 0, 0, 1, 1)
self.dockWidget_4.setWidget(self.dockWidgetContents_8)
MainWindow.addDockWidget(QtCore.Qt.DockWidgetArea(1), self.dockWidget_4)

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

#self.runLiveEvaluation()

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Evaluation Screen"))
    self.label.setText(_translate("MainWindow", "Real Steering: "))
    self.label_3.setText(_translate("MainWindow", "Predicted Steering: "))
    self.label_6.setText(_translate("MainWindow", "Real Throttle:"))
    self.label_7.setText(_translate("MainWindow", "Predicted Throttle:"))
    self.label_8.setText(_translate("MainWindow", "Real Brakes:"))
    self.label_9.setText(_translate("MainWindow", "Predicted Brakes:"))
    self.label_10.setText(_translate("MainWindow", "Steering Error:"))
    self.label_11.setText(_translate("MainWindow", "Throttle Error:"))
    self.label_12.setText(_translate("MainWindow", "Braking Error:"))
    self.label_13.setText(_translate("MainWindow", "Mean Error: "))
    self.label_2.setText(_translate("MainWindow", "Please Wait..."))
    self.label_4.setText(_translate("MainWindow", "Normalized data"))
    self.label_5.setText(_translate("MainWindow", "Original data"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```