ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΠΡΑΚΤΟΡΑ ΓΙΑ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΑ ΜΕ ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΕΚΜΑΘΗΣΗΣ

ΑΛΕΞΙΟΥ ΦΑΒΙΟΛΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Σταμούλης Γεώργιος
Καθηγητής

Λαμία, Οκτώβριος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΠΡΑΚΤΟΡΑ ΓΙΑ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΑ ΜΕ ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΕΚΜΑΘΗΣΗΣ

ΑΛΕΞΙΟΥ ΦΑΒΙΟΛΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Σταμούλης Γεώργιος
Καθηγητής

Λαμία, Οκτώβριος 2020

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE &
TELECOMMUNICATIONS

AGENT DEVELOPMENT FOR VIDEO GAMES WITH MACHINE LEARNING TECHNIQUES

ALEXIOU FAVIOLA

FINAL THESIS

ADVISOR

Stamoulis George
Professor

Lamia, October 2020

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

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

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

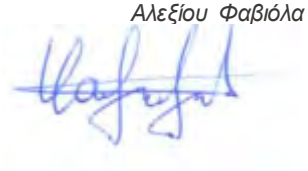
3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κ.λπ.), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη.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

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

Ημερομηνία: 08/10/2020

Η Δηλούσα

Αλεξίου Φαβιόλα



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

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία με θέμα «Ανάπτυξη πράκτορα για βιντεοπαιχνίδια με τεχνικές μηχανικής εκμάθησης», εκπονήθηκε από την φοιτήτρια Αλεξίου Φαβιόλα του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Θεσσαλίας κατά το ακαδημαϊκό έτος 2019-2020.

Θα ήθελα λοιπόν να εκφράσω τις ειλικρινείς και θερμές ευχαριστίες μου σε όλους όσους συμπαραστάθηκαν και συνέβαλλαν στην ολοκλήρωση αυτής της προσπάθειας. Κατά κύριο λόγο θα ήθελα να ευχαριστήσω τον Διδάκτορα κ. Φλώρο Γεώργιο για την πολύτιμη προσφορά του και την μετάδοση γνώσεων καθοριστικών για την εκπόνηση της παρούσας πτυχιακής εργασίας.

Ιδιαίτερες ευχαριστίες θα ήθελα να απευθύνω στον επιβλέποντα της πτυχιακής μου εργασίας Καθηγητή κ. Σταμούλη Γεώργιο, του οποίου η αρωγή και η υποστήριξη υπήρξε πολύτιμη καθ' όλη τη διάρκεια της εκπόνησης της πτυχιακής εργασίας.

Τέλος ευχαριστώ θερμά τα άτομα από το οικογενειακό και φιλικό μου περιβάλλον, που ήταν δίπλα μου σε όλη τη διάρκεια των σπουδών μου αλλά και στην προσπάθεια της αποπεράτωσης της παρούσας πτυχιακής εργασίας, παρέχοντας απεριόριστη κατανόηση και ψυχολογική υποστήριξη.

ΠΕΡΙΛΗΨΗ

Σκοπός αυτής της πτυχιακής εργασίας είναι η εκπαίδευση ενός ευφυούς πράκτορα, ο οποίος μαθαίνει να τερματίζει επιτυχώς την πίστα Luigi's Raceway του παιχνιδιού Mario Kart 64 μέσω αυτόνομης οδήγησης χωρίς την παρέμβαση του χρήστη. Σε πρώτο στάδιο ήταν απαραίτητο να συγκεντρωθούν δεδομένα από το τρέξιμο της πίστας, όπου χειριστής είναι ο προγραμματιστής. Με την χρήση ενός εξομοιωτή του παιχνιδιού δίνεται δυνατότητα στον χειριστή να συλλέξει στιγμιότυπα της πίστας και να καταγράψει τις κινήσεις που εκτέλεσε καθ' όλη τη διάρκεια του παιχνιδιού. Αποσκοπώντας στην πιο ορθή εκμάθηση και στη βέλτιστη επίδοση του πράκτορα, η διαδικασία αυτή επαναλήφθηκε πολλές φορές με σκοπό να συγκεντρωθούν όσο το δυνατόν περισσότερα δεδομένα από τις κινήσεις και τις εικόνες που καταγράφηκαν. Η εκμάθηση του πράκτορα βασίστηκε σε μια αρχιτεκτονική συνελικτικού τεχνητού νευρωνικού δικτύου στο οποίο εισάγονται τα δεδομένα εικόνας και κινήσεων χειριστηρίου που αποθηκεύτηκαν κατά την εξομοίωση, και στη συνέχεια εκπαιδεύτηκε σύμφωνα με αυτά. Αναλύθηκε η μέθοδος της υλοποίησης του τεχνητού νευρωνικού δικτύου και ο τρόπος εκπαίδευσης του με τη χρήση σύγχρονων τεχνικών. Το αποτέλεσμα είναι η δημιουργία ενός αυτόνομου πράκτορα που μπορεί να αναγνωρίσει χαρακτηριστικά του δρόμου του Luigi's Raceway και να επιλέξει μόνος του τον κατάλληλο τρόπο οδήγησης για την ομαλή διεκπεραίωση της πίστας. Τελικά, εάν και ο πράκτορας δεν έχει τη δυνατότητα επιλογής αντικειμένων, εκτέλεσης αλμάτων και χρήσης του φρένου, τα πειραματικά αποτελέσματα δείχνουν ότι ο πράκτορας που αναπτύχθηκε μπορεί να οδηγήσει επιτυχώς την πίστα του παιχνιδιού στην οποία εκπαιδεύτηκε.

Λέξεις κλειδιά: τεχνητή νοημοσύνη, πράκτορας, συνελικτικά νευρωνικά δίκτυα, ψηφιακά παιχνίδια

ABSTRACT

The purpose of this thesis is the training of an intelligent agent, that has the ability to successfully terminate Mario Kart's 64 level named Luigi's Raceway, through autonomous driving, without any human intervention. First of all, the gathering of dataset had to be completed while a human-player, the programmer, is in control of Mario Kart's gameplay. An emulator is used in order to collect screenshots from the particular level of the game and record the commands executed during the game. Aiming to the better training of the agent and his optimal performance, the whole process needed to be repeated several times, so as to gather as much data as possible. The training of the AI model is based on a convolutional neural network (CNN) architecture, which is used to load all of the image data and the controller's actions saved during the simulation, according to which the CNN is trained. The method used for developing the artificial neural network and the contemporary technique used to train the neural network was analyzed. As a result, an autopilot agent was constructed that can recognize the Luigi's Raceway road characteristics and choose the best way to drive so as to smoothly complete the level. Finally, even though the agent cannot opt to choose objects, do leaps or use the break command, the results of the experiment show that the developed agent can successfully drive the level of the game to which it was trained.

Keywords: artificial intelligence, agent, convolutional neural network, video games

ΕΥΧΑΡΙΣΤΙΕΣ I
ΠΕΡΙΛΗΨΗ III
ABSTRACT V

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ 1

1.1 Εισαγωγη 1
1.2 Δομη της πτυχιακης εργασιας 3

ΚΕΦΑΛΑΙΟ 2 DEEP LEARNING 5

2.1 Εισαγωγη στη μηχανικη μαθηση 5
2.1.1 Ορισμος 5
2.1.2 Περιγραφη τεχνητου νευρωνα 5
2.1.3 Περιγραφη της βαθεας μαθησης 7
2.2 Τεχνικες νευρωνικων δικτυων 9
2.3 Συναρτησεις υπολογισμου απωλειας 13
2.4 Βιβλιοθηκες ανοικτου λογισμικου για νευρωνικα δικτυα 13
2.4.1 Εξελιξη του ανοιχτου λογισμικου 13
2.4.2 TensorFlow 14
2.4.3 Keras API 14

ΚΕΦΑΛΑΙΟ 3 ΜΕΘΟΔΟΛΟΓΙΑ 17

3.1 Συλλογη δεδομενων προς εκπαιδευση 17
3.2 Αρχιτεκτονικη συνελικτικων νευρωνικων δικτυων 17
3.2.1 Εισαγωγη 17
3.2.2 Μοντελο αναπτυξης τεχνητου νευρωνικου δικτυου 19
3.2.3 Υλοποιηση της αρχιτεκτονικης συνελικτικου νευρωνικου δικτυου 19
3.3 Τεχνικες εκμαθησης 25
3.3.1 Μαθηση μεσω μιμησης: εφαρμογη του αλγοριθμου Dagger 25
3.3.2 Ενισχυτικη μαθηση 26
3.4 Υλοποιηση ευφυους πρακτορα 26

ΚΕΦΑΛΑΙΟ 4 ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ 31

4.1 Περιγραφη τρεχιματος πιστας και αποτελεσματα 31
4.2 Συμπερασματα 31

ΚΕΦΑΛΑΙΟ 5 ΕΠΙΛΟΓΟΣ 35

ΒΙΒΛΙΟΓΡΑΦΙΑ 37

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

1.1 Εισαγωγή

Μια από τις μεγαλύτερες προκλήσεις στον κλάδο της επιστήμης των υπολογιστών είναι η δημιουργία ενός υπολογιστή που θα μπορεί να μιμηθεί την ανθρώπινη νοημοσύνη. Για την επίτευξη αυτού του στόχου, έχει αναπτυχθεί ο κλάδος της τεχνητής νοημοσύνης (artificial intelligence) [1], ο οποίος στοχεύει στη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που προσομοιάζουν την ανθρώπινη συμπεριφορά, υπονοώντας έτσι κάποια στοιχειώδη ευφυΐα. Η εξέλιξη του κλάδου αυτού, έδωσε τη δυνατότητα στους ερευνητές να «εκπαιδεύσουν», δηλαδή να προγραμματίσουν, τους υπολογιστές έτσι ώστε να αναγνωρίζουν εικόνες και ήχους, να συνδυάζουν λογική και αντίληψη, ακόμη και να επιλύουν προβλήματα.

Η τεχνητή νοημοσύνη διαχωρίζεται στη συμβολική και στην υποσυμβολική νοημοσύνη. Η συμβολική τεχνητή νοημοσύνη προσπαθεί να εξομοιώσει την ανθρώπινη νοημοσύνη αλγοριθμικά, χρησιμοποιώντας λογικούς κανόνες υψηλού επιπέδου και σύμβολα. Η υποσυμβολική τεχνητή νοημοσύνη έχει σκοπό την αναπαραγωγή της ανθρώπινης ευφυΐας, χρησιμοποιώντας αριθμητικά μοντέλα για την δημιουργία εκλογικευμένων συμπεριφορών. Οι δύο αυτές κατηγορίες διαφοροποιούν την τεχνητή νοημοσύνη ανάλογα με τα εργαλεία και τις τεχνικές που χρησιμοποιούνται για την επίλυση των προβλημάτων.

Η πρώτη εφαρμογή της τεχνητής νοημοσύνης ήταν η μηχανή Turing, η οποία αναπτύχθηκε από τον Alan Turing το 1936, κατά τη διάρκεια του Δεύτερου Παγκόσμιου Πολέμου, με στόχο την αποκωδικοποίηση του κώδικα Enigma. Αποτελούσε μια μηχανή με γενικό μοντέλο υπολογισμού που μπορεί να υλοποιήσει οποιοδήποτε υπολογίσιμο αλγόριθμο με την χρήση συμβόλων σύμφωνα με ορισμένο σύνολο κανόνων. Σε μεταγενέστερο χρόνο, το 1950, προτάθηκε από τον Alan Turing η δοκιμή Turing, διαδικασία για την διαπίστωση εάν μια μηχανή διαθέτει ευφυΐα. Τελικά, η τεχνητή νοημοσύνη θεμελιώθηκε τυπικά ως πεδίο, από σύνολο ορισμένων Αμερικανών επιστημόνων σχετικών με τον τομέα (Τζον Μακαρθι, Μαρβιν Μίνσκυ, Κλοντ Σάνον κ.ά.) το 1956 [1].

Η σύγχρονη τεχνητή νοημοσύνη συνδυάζει εφαρμοσμένα μαθηματικά, πιθανότητες, αλγοριθμικές μεθόδους, λογική, μάθηση, ακόμη και αντίληψη. Βασίζεται σε μεθόδους μηχανικής μάθησης (machine learning), οι οποίες κυρίως χαρακτηρίζονται από μαθηματικούς αλγόριθμους που έχουν την ικανότητα να πραγματοποιούν προβλέψεις σύμφωνα με ένα σύνολο δεδομένων. Κατά αυτό τον τρόπο, ο υπολογιστής μπορεί να προσαρμοστεί σε νέες συνθήκες, να μάθει νέες πληροφορίες και να αναγνωρίσει νέα περιβάλλοντα ή να συμπεράνει πρότυπα. Συνεπώς, το πεδίο της τεχνητής νοημοσύνης αποσκοπεί στην κατασκευή νοήμων

οντοτήτων που θα προβλέπουν νέες καταστάσεις, σύμφωνα με τα δεδομένα που τους παρέχονται.

Τα βιντεοπαιχνίδια είναι άμεσα συνδεδεμένα με τον κλάδο της επιστήμης των υπολογιστών και αποτελούν πλέον ξεχωριστό πεδίο της. Με τη ραγδαία ανάπτυξη της τεχνολογίας και την απόκτηση ηλεκτρονικών υπολογιστών από το μεγαλύτερο σύνολο του πληθυσμού, η βιομηχανία των ψηφιακών παιχνιδιών αποτελεί πλέον κύριο κομμάτι της ενασχόλησης και της ψυχαγωγίας των ανθρώπων. Εντούτοις, η ανάπτυξη και η εξέλιξη αυτής της βιομηχανίας αποτελεί επίσης και ένα από τα κύρια πεδία έρευνας, η οποία βασίζεται στην τεχνητή νοημοσύνη. Από πολύ νωρίς οι ερευνητές γράφουν προγράμματα τα οποία δημιουργούν, επιλύουν και παίζουν παιχνίδια. Σκοπός τους είναι να δοκιμάσουν αν οι ηλεκτρονικοί υπολογιστές μπορούν να λύσουν προβλήματα, να αναγνωρίσουν πρότυπα, να συγκρατήσουν πληροφορίες, να αντιδρούν γρήγορα στις μεταβολές του περιβάλλοντος, δηλαδή να αποκτήσουν στοιχειώδη «νοημοσύνη». Χαρακτηριστικό παράδειγμα αποτελεί το Chinook το πρώτο πρόγραμμα ηλεκτρονικού υπολογιστή που κατάφερε να νικήσει τον τίτλο παγκόσμιου πρωταθλητή σε αγώνα Ντάμας το 1994, ενάντια στον Marion Tinsley, αλλά και ο Deep Blue της IBM [2], υπολογιστής βασισμένος σε τεχνητή νοημοσύνη, που επικράτησε σε σκακιστικό αγώνα ενάντια στον Garry Kasparov το 1997.

Αρχικά, η τεχνητή νοημοσύνη χρησιμοποιήθηκε σε παιχνίδια στρατηγικής (όπως το σκάκι). Με την πάροδο των χρόνων και την ανάπτυξη της τεχνολογίας, η τεχνητή νοημοσύνη αποκτά νέες εφαρμογές στα ψηφιακά παιχνίδια, κερδίζοντας το ενδιαφέρον πολλών ερευνητών οι οποίοι κατάφεραν τον εμπλουτισμό των περιβαλλόντων τους με τη δημιουργία πολυπλοκότερων γραφικών αλλά και την προσθήκη επιπλέον χαρακτήρων. Η γενιά των ηλεκτρονικών παιχνιδιών αποτελεί το επόμενο βήμα της τεχνητής νοημοσύνης όπου νέοι μέθοδοι αναπτύσσονται, πέραν των τεχνικών που κάνουν χρήση κανόνων βασισμένων σε λογική. Τα βιντεοπαιχνίδια γίνονται ολοένα και πιο περίπλοκα και απαιτητικά, ενώ πολλά από αυτά, όπως παιχνίδια στρατηγικής και οδήγησης, προσομοιώνουν συνθήκες και εικόνες από τον πραγματικό κόσμο. Συγκεκριμένα, περιέχουν στοιχεία που καθορίζουν τις προκλήσεις και τα προβλήματα που ο παίκτης καλείται να αντιμετωπίσει, όπως η χρήση αντικειμένων, η σωστή επιλογή διαδρομής, η τήρηση μιας σειράς από κανόνες ακόμα και η επιλογή του επιπέδου δυσκολίας και των αντιπάλων. Συμπερασματικά, η εξέλιξη της τεχνητής νοημοσύνης και των μεθόδων της βοήθησε στη δημιουργία δυσκολότερων, εξυπνότερων και πιο ρεαλιστικών παιχνιδιών.

Το πεδίο της τεχνητής νοημοσύνης είναι ρητά συνδεδεμένο με την κατασκευή ευφυών πρακτόρων (intelligent agents). Στον τομέα των ψηφιακών παιχνιδιών, οι πράκτορες ενεργούν αποσκοπώντας στην περάτωση του παιχνιδιού με το καλύτερο δυνατό αποτέλεσμα, ιδιαίτερα όταν υπάρχει κάποια μεταβολή του περιβάλλοντος τους. Ο όρος ευφυής πράκτορας [3] στην τεχνητή νοημοσύνη χρησιμοποιείται για να περιγράψει το σύνολο των προγραμμάτων που έχει την ικανότητα να αναγνωρίζει δεδομένα από το περιβάλλον που βρίσκεται, να μαθαίνει από

αυτά, να προβλέπει καταστάσεις ανάλογα με τις ενέργειές του, και να εξάγει συμπεράσματα τα οποία χρησιμοποιεί για να επιτύχει συγκεκριμένους στόχους και να διεκπεραιώσει εργασίες. Οι πράκτορες διαφέρουν από τα απλά προγράμματα, κυρίως στον τρόπο λειτουργίας τους, καθώς διαθέτουν την ικανότητα αυτόνομου ελέγχου και είναι σε θέση να υιοθετήσουν ανθρώπινες συμπεριφορές και αντιδράσεις. Για αυτό το λόγο βρίσκουν μεγάλη απήχηση στην υλοποίηση λογισμικών που έχουν τη δυνατότητα να οδηγούν αυτόνομα, ενώ βρίσκονται σε περιβάλλοντα τα οποία περιλαμβάνουν προκλήσεις οι οποίες απαιτούν αντίληψη και πρόβλεψη, όπως η επιλογή σωστής κλήσης γωνίας της κατεύθυνσης, κατάλληλης ταχύτητας και η αναγνώριση του δρόμου.

Στα πλαίσια αυτής της πτυχιακής εργασίας, χρησιμοποιώντας τεχνικές βαθέας μάθησης, αναπτύξαμε ένα πράκτορα που μπορεί να παίξει και να τερματίσει επιτυχώς την πίστα Luigi's Raceway του παιχνιδιού MarioKart 64. Όπως θα περιγράψουμε και παρακάτω, αναπτύξαμε ένα νευρωνικό δίκτυο το οποίο το εκπαιδεύσαμε με δεδομένα που συλλέξαμε από στιγμιότυπα του παιχνιδιού κατόπιν συνεχόμενων καταγραφών από πραγματικό χρήστη. Τα αποτελέσματα δείχνουν ότι τα συνελικτικά νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν αποτελεσματικά στον τομέα της αυτόματης οδήγησης.

1.2 Δομή της πτυχιακής εργασίας

Τα επόμενα κεφάλαια της πτυχιακής εργασίας είναι οργανωμένα όπως περιγράφουμε παρακάτω.

Αρχικά, στο κεφάλαιο 2 περιγράφονται βασικές έννοιες μηχανικής μάθησης, όπως οι τεχνητοί νευρώνες και τα τεχνητά νευρωνικά δίκτυα. Παράλληλα, παρέχεται σύντομη περιγραφή στη βαθέα μάθηση (deep learning). Επιπλέον, περιγράφονται οι βιβλιοθήκες Keras και TensorFlow, που χρησιμοποιήθηκαν για την υλοποίηση του τεχνητού νευρωνικού δικτύου.

Στο κεφάλαιο 3 γίνεται ανάπτυξη της μεθοδολογίας για τη δημιουργία του πράκτορα και τη συλλογή των απαραίτητων δεδομένων. Επιπλέον, παρουσιάζεται και αναλύεται η αρχιτεκτονική των συνελικτικών νευρωνικών δικτύων και οι μέθοδοι εκμάθησης του πράκτορα.

Στο κεφάλαιο 4 παρουσιάζονται πειράματα που αφορούν τη συμπεριφορά του πράκτορα στην πίστα Luigi's Raceway του Mario Kart 64 και τα αντίστοιχα αποτελέσματα.

Τέλος, στο κεφάλαιο 5 δίνεται ο επίλογος και αναφέρονται πιθανές μελλοντικές προεκτάσεις.

ΚΕΦΑΛΑΙΟ 2 Deep learning

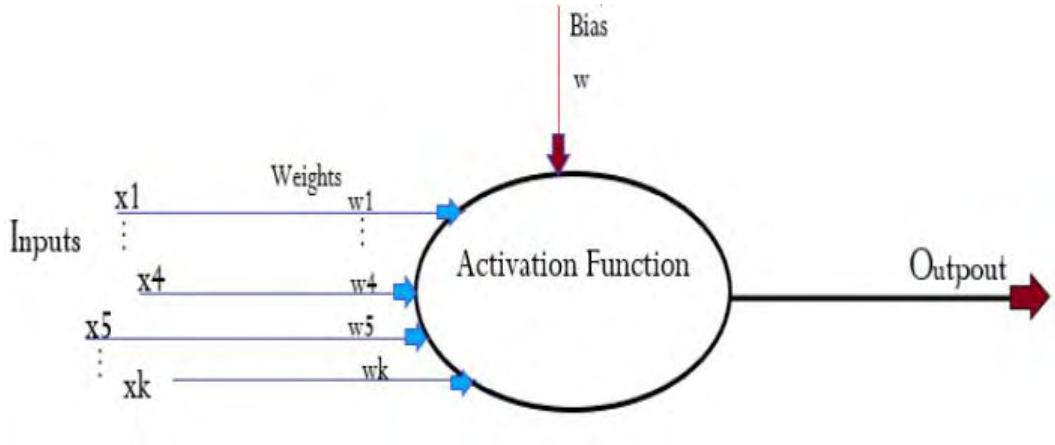
2.1 Εισαγωγή στη μηχανική μάθηση

2.1.1 Ορισμός

Ένα από τα πεδία της τεχνητής νοημοσύνης είναι η μηχανική μάθηση (machine learning) [3], η οποία αντιπροσωπεύει μια ευρύτερη ομάδα αλγορίθμων που βασίζεται στη μάθηση από δεδομένα και έχει την ιδιότητα εξαγωγής πληροφοριών και συμπερασμάτων από αυτά. Ως βαθέα μάθηση (deep learning), χαρακτηρίζουμε μια τεχνική μηχανικής μάθησης η οποία πραγματεύεται τη δημιουργία τεχνητών νευρωνικών δικτύων, την υλοποίηση αλγορίθμων, καθώς και την εκμάθηση χαρακτηριστικών από ένα σύνολο δεδομένων. Στόχος αυτής είναι οι υπολογιστές να λύσουν περίπλοκα προβλήματα που απαιτούν ανθρώπινη νοημοσύνη. Τα σύγχρονα μοντέλα βαθέας μάθησης βασίζονται σε τεχνητά νευρωνικά δίκτυα, τα οποία αποτελούνται από πολλαπλά επίπεδα νευρώνων και προσπαθούν να εξάγουν χαρακτηριστικά από την είσοδο των δεδομένων. Η διαδικασία της μάθησης των νευρωνικών δικτύων γίνεται αυτόματα, δίνοντας ως εισόδους τα δεδομένα και τις ετικέτες. Τα δεδομένα αυτά διατρέχουν όλα τα επίπεδα του τεχνητού νευρωνικού δικτύου καταλήγοντας στην έξοδο σε μια υψηλού επιπέδου αναπαράσταση.

2.1.2 Περιγραφή τεχνητού νευρώνα

Ένα τεχνητό νευρωνικό δίκτυο απαρτίζεται από πολλούς τεχνητούς νευρώνες, οι οποίοι αποτελούν και το δομικό του στοιχείο. Ο τεχνητός νευρώνας είναι μια μαθηματική συνάρτηση που δέχεται μία ή και περισσότερες τιμές, όπως φαίνεται στην Εικόνα 1, οι οποίες αντιπροσωπεύουν πληροφορίες. Χαρακτηρίζονται και ως κόμβοι που δέχονται αριθμητικές εισόδους είτε από το περιβάλλον είτε από άλλους νευρώνες και έχουν έξοδο η οποία είτε δίνει πληροφορία προς το περιβάλλον είτε προς άλλο νευρώνα. Για την μεταβίβαση των τιμών οι κόμβοι συνδέονται μεταξύ τους με χρήση συναπτικών ακμών, οι οποίες καθορίζουν τον βαθμό αλληλεπίδρασης κάθε ζεύγους ανάλογα με τα συναπτικά τους βάρη. Η κωδικοποίηση της γνώσης του δικτύου πραγματοποιείται από τα συναπτικά βάρη, τα οποία μεταβάλλονται για την αποδυνάμωση ή ενδυνάμωση της ισχύος των συναπτικών ακμών. Η μαθηματική συνάρτηση που ονομάζουμε τεχνητό νευρώνα πήρε το όνομά της από τον βιολογικό νευρώνα του ανθρώπινου εγκεφάλου, διότι έχει παρόμοιο τρόπο λειτουργίας και αρχιτεκτονική δομή.



Εικόνα 1. Τεχνητός νευρώνας

Οι νευρώνες διαχωρίζονται στις εξής κατηγορίες:

- Νευρώνες εισόδου
- Νευρώνες εξόδου
- Κρυφοί/Υπολογιστικοί νευρώνες

Ένας νευρώνας απαρτίζεται από:

- Τις εισόδους (inputs) του νευρώνα
- Τα συναπτικά βάρη (weights) των εισόδων
- Το σώμα του τεχνητού νευρώνα, που περιλαμβάνει:
  - τον αθροιστή, όπου πραγματοποιείται πολλαπλασιασμός των εισόδων με τα βάρη τους και υπολογίζεται το άθροισμά τους
  - τη συνάρτηση ενεργοποίησης ή μεταφοράς, από την οποία υπολογίζεται η τελική έξοδος
- Την έξοδο (output) του νευρώνα, η οποία μπορεί να έχει και περισσότερες από μια εξόδους αλλά είναι όλες ισότιμες
- Την ειδική είσοδο (bias), η οποία μπορεί να πάρει τρείς διαφορετικές τιμές. Όταν ισούται με "0" την αγνοούμε, εάν ισούται με "-1" τότε ονομάζεται κατώφλι, και τέλος αν ισούται με "1" τότε την αποκαλούμε πόλωση.

Η μαθηματική συνάρτηση που χαρακτηρίζει ένα νευρώνα είναι η εξής:

$$y_k = \varphi\left(\sum_{i=0}^{N} x_{ki} w_{ki}\right)$$

όπου:

- y είναι η έξοδος του νευρώνα k
- w είναι το i-οστό βάρος του νευρώνα k
- x είναι η i-οστή είσοδος του νευρώνα k
- φ είναι η συνάρτηση ενεργοποίησης

Η συνάρτηση ενεργοποίησης (μεταφοράς), η οποία παράγει και την έξοδο του νευρώνα, μπορεί να έχει μια από τις ακόλουθες μορφές:

- Βηματική (step transfer function)
- Γραμμική (linear transfer function)
- Μη γραμμική (non-linear transfer function)
- Στοχαστική (stochastic transfer function)

Η μορφή της συνάρτησης ενεργοποίησης καθορίζει τα όρια των εξόδων των νευρώνων, ενώ παράλληλα επηρεάζει τη μορφοποίηση των δεδομένων εισόδου πριν την επεξεργασία.

2.1.3 Περιγραφή της βαθέας μάθησης

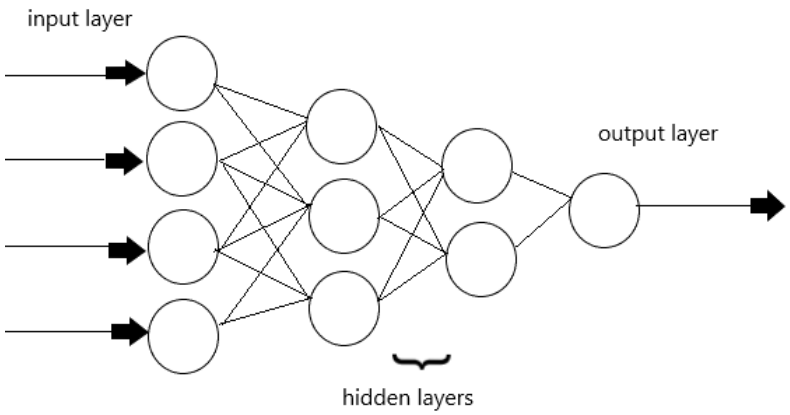
Η ανάγκη της αναπαράστασης δεδομένων που περιέχουν γνώση για τον κόσμο και η μετάδοσή τους στα υπολογιστικά συστήματα, έτσι ώστε να εξάγουν συμπεράσματα, οδήγησε στην ανάπτυξη των μοντέλων βαθέας μάθησης. Πιο αναλυτικά, η καθημερινή ζωή του ανθρώπου χαρακτηρίζεται από ένα αχανές σύνολο πληροφοριών που αφορούν τη λειτουργία ολόκληρου του κόσμου. Ο άνθρωπος κατέχει γνώση η οποία αποκτάται τόσο μέσω της νοημοσύνης του, όσο και μέσω προσωπικών εμπειριών. Η γνώση αυτή χαρακτηρίζεται είτε ως αντικειμενική είτε ως υποκειμενική βάση της προαίσθησής του, και είναι δύσκολο στο σύνολό της να κανονικοποιηθεί και να κατηγοριοποιηθεί. Ένα από τα πιο απαιτητικά χαρακτηριστικά της τεχνητής νοημοσύνης είναι πως οι υπολογιστές καλούνται να υιοθετήσουν αυτή τη γνώση, ώστε να «συμπεριφερθούν» με έξυπνο τρόπο. Αρχικά, η μέθοδος που υιοθετήθηκε για την μετάδοση της γνώσης και των δεδομένων στα υπολογιστικά συστήματα ήταν η χρήση κανόνων βασισμένοι σε λογική.

Ένα από τα πιο γνωστά project βασισμένο στην τεχνητή νοημοσύνη είναι το Cyc, που ξεκίνησε από τον Lenat το 1989 [4]. Το Cyc είναι ένα σύστημα τεχνητής νοημοσύνης το οποίο περιλαμβάνει μια τεράστια βάση δεδομένων από κανόνες που αναπαριστούν τον κόσμο και βασίζονται σε κοινή λογική. Στοχεύει στον συνδυασμό ενός συνόλου αμέτρητων γνώσεων και ενός συνόλου κανόνων, για την παράγωγή ποικίλων σεναρίων που αναπαριστούν μια πληροφορία. Οι κανόνες εισάγονται από προγραμματιστές και είναι περίπλοκοι αλλά και ακριβείς ως προς την περιγραφή τους για τον κόσμο. Σε πείραμα που πραγματοποιήθηκε το 1992, το Cyc απέτυχε να καταλάβει το πρόβλημα “Fred shaving”. Αφορούσε έναν άνθρωπο ο οποίος ξυριζόταν με βοήθεια ηλεκτρικής συσκευής, το Cyc δεν μπόρεσε να αναγνωρίσει αν ο Fred είναι άνθρωπος ή ηλεκτρονική συσκευή ενώ ξυριζόταν, διότι κρατούσε μια μηχανή ξυρίσματος. Συνεπώς, οι ερευνητές κατέληξαν στο συμπέρασμα ότι τα συστήματα τεχνητής νοημοσύνης θα πρέπει να είναι σε θέση να μπορούν να διαμορφώνουν το πεδίο γνώσης τους δημιουργώντας σχηματισμό μοτίβων από μια σειρά εισερχόμενων δεδομένων. Η μηχανική μάθηση ορίζει αυτή την ικανότητα, δηλαδή την επίλυση προβλημάτων τα οποία σχετίζονται με γνώσεις του πραγματικού κόσμου και επιφέρουν αποφάσεις που εμφανίζουν υποκειμενικότητα. Χαρακτηριστικό παράδειγμα αποτελεί

ο αλγόριθμος μηχανικής μάθησης naive Bayes [5], ο οποίος για παράδειγμα έχει χρησιμοποιηθεί επιτυχώς στη διάκριση των ανεπιθύμητων e-mail.

Κύριο ζήτημα για την ορθή λειτουργία και λήψη αποφάσεων από τα υπολογιστικά συστήματα αποτελεί η αποδοτικότητα των αλγορίθμων τους, η οποία εξαρτάται από την ικανότητα να εξάγουν χαρακτηριστικά από τα δεδομένα εκμάθησης. Ο όρος βαθέα χρησιμοποιείται για να προσδιοριστεί η ύπαρξη πολλαπλών επιπέδων νευρώνων τα οποία σχετίζονται μεταξύ τους. Εξαιτίας της ύπαρξης τους, το υπολογιστικό σύστημα μπορεί να δεχθεί πολλαπλές πληροφορίες. Κάθε ένα από αυτά συνδέεται με προηγούμενα επίπεδα νευρώνων, από τα οποία λαμβάνει τα δεδομένα εισόδου. Τα χαρακτηριστικά σε κάθε επίπεδο υπόκεινται σε υπολογισμούς και διοχετεύονται μέσω συνδέσεων σε επόμενα επίπεδα. Εξαιτίας των ακριβέστερων αναπαραστάσεων δεδομένων που προκύπτουν από την ύπαρξη πολλαπλών επιπέδων η τεχνική της βαθέας μάθησης επιτυγχάνει την ευέλικτη και γρήγορη εκμάθηση νευρωνικών δικτύων.

Τα μοντέλα της βαθέας μάθησης έχουν την ιδιότητα να επιλέγουν από μόνα τους τα κατάλληλα χαρακτηριστικά από ένα σύνολο πληροφοριών που δίνεται ως είσοδος από τον προγραμματιστή. Ένα από τα πρώτα είδη τεχνητού νευρωνικού δικτύου, στο οποίο βασίστηκαν τα σημερινά βαθέα νευρωνικά δίκτυα, είναι το πολυεπίπεδο perceptron (multilayer perceptron/MLP) [6]. Αποτελεί ένα μοντέλο μηχανικής μάθησης και υλοποιεί μια μαθηματική συνάρτηση που επεξεργάζεται πολλά εισερχόμενα δεδομένα. Συγκεκριμένα, αποτελείται από τουλάχιστον τρία επίπεδα και χαρακτηρίζεται ως ένα τεχνητό νευρωνικό δίκτυο το οποίο περνά τις πληροφορίες του από το πρώτο στο τελευταίο επίπεδο (δίκτυο πρόσθιας τροφοδότησης). Οι νευρώνες οποιουδήποτε επιπέδου τροφοδοτούν αποκλειστικά τους νευρώνες του επόμενου επιπέδου με μια τιμή, που αναπαριστά εισερχόμενη πληροφορία, και τροφοδοτούνται αποκλειστικά από νευρώνες προηγούμενου επιπέδου, όπως φαίνεται και στην Εικόνα 2. Λόγω της ύπαρξης πολλών επιπέδων στο δίκτυο αυτό, δίνεται και η δυνατότητα παράλληλης επεξεργασίας, αξιοποιώντας πλήρως τις σύγχρονες μαζικά παράλληλες αρχιτεκτονικές (π.χ. κάρτες γραφικών [7]) για αποδοτικότερη υλοποίηση. Τελικά, στο τελευταίο επίπεδο εξάγεται μια τιμή η οποία χρησιμοποιείται για την κατηγοριοποίηση της εισόδου σε μια από τις κλάσεις, βάση των οποίων εκπαιδεύσαμε το νευρωνικό δίκτυο.

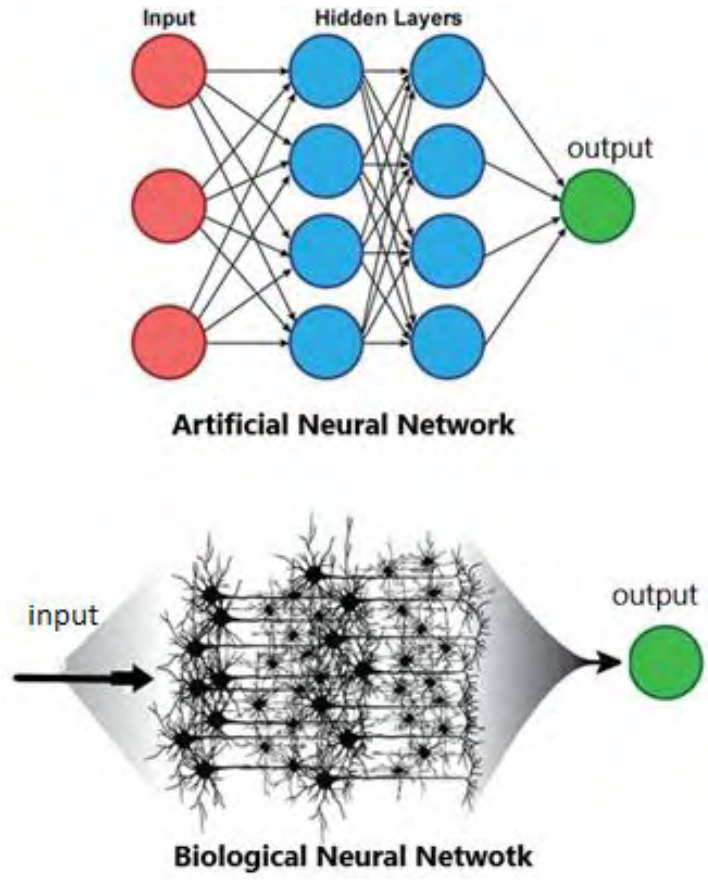


Εικόνα 2. Πολυεπίπεδο perceptron

Επιπροσθέτως, στα μοντέλα της βαθέας μάθησης υπάρχει η δυνατότητα προσθήκης επιπλέον επιπέδων και επιπλέον πληροφοριών εντός των επιπέδων, δημιουργώντας έτσι μια συνάρτηση συνεχούς αυξανόμενης πολυπλοκότητας. Με αυτό τον τρόπο, προβλήματα με μεγάλο όγκο δεδομένων μπορούν να επιλυθούν από μεγαλύτερα μοντέλα ύστερα από πολλές εκπαιδεύσεις.

2.2 Τεχνικές νευρωνικών δικτύων

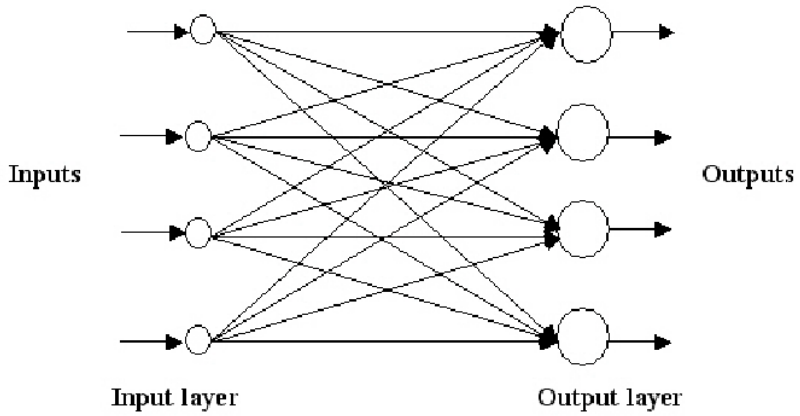
Χαρακτηρίζουμε το τεχνητό νευρωνικό δίκτυο ως ένα κύκλωμα νευρώνων με διάφορες τοπολογίες, το οποίο τις τελευταίες δεκαετίες αποτελεί βασικό εργαλείο της μηχανικής μάθησης. Είναι ένα υπολογιστικό σύστημα το οποίο προσομοιώνει/προσεγγίζει τη λειτουργία των νευρώνων του εγκεφάλου. Όπως φαίνεται στην Εικόνα 3, η δομή του τεχνητού νευρωνικού δικτύου είναι εμπνευσμένη από αυτή του ανθρώπινου βιολογικού νευρωνικού δικτύου. Σκοπός των τεχνητών νευρωνικών δικτύων είναι η εξαγωγή πληροφοριών η οποία σχετίζεται με τα ερεθίσματα που εισέρχονται σε αυτά. Συγκεκριμένα έχει δειχθεί ότι τα μοντέλα αυτά μπορούν να επεξεργαστούν δεδομένα και να κάνουν υπολογισμούς που κάνει και ο ανθρώπινος εγκέφαλος.



Εικόνα 3. Στο επάνω μέρος της εικόνας βλέπουμε τη δομή ενός τεχνητού νευρωνικού δικτύου. Στο κάτω μέρος διακρίνεται η δομή ενός βιολογικού νευρωνικού δικτύου

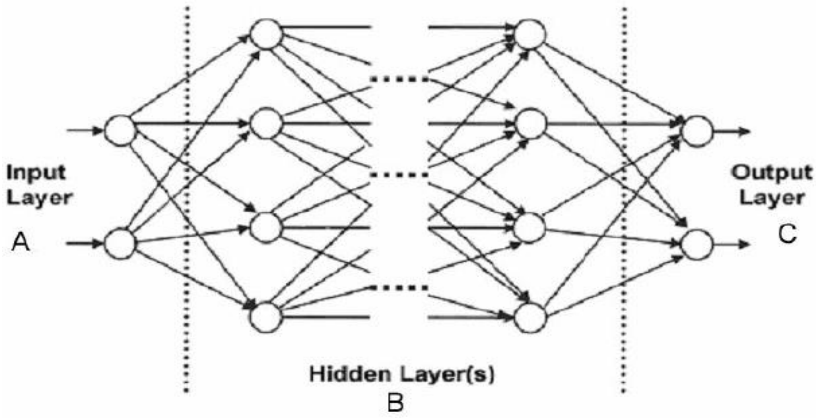
Τα δίκτυα εκπαιδεύονται με σκοπό να αποκτήσουν γνώση και ως εκ τούτου να μπορέσουν να επιλύσουν μια διεργασία και να επιφέρουν το καλύτερο δυνατό αποτέλεσμα. Η εκπαίδευσή τους πραγματοποιείται με τη βοήθεια παραδειγμάτων, ώστε να είναι σε θέση να μάθουν το περιβάλλον που βρίσκονται. Για την ανάπτυξη αλγορίθμων εκπαίδευσης είναι απαραίτητο να οριστεί η μέθοδος που ακολουθείται, δηλαδή αν το δίκτυο θα εκπαιδεύεται με επίβλεψη, θα έχει συγκεκριμένο κριτήριο/στόχο, είτε θα οργανώνεται και εκπαιδεύεται μόνο του. Η αποθήκευση της γνώσης που αποκτάται από την εκπαίδευση πραγματοποιείται μέσω των δυνάμεων σύνδεσης των νευρώνων, δηλαδή των συναπτικών βαρών. Η δομή των τεχνητών νευρωνικών δικτύων αποτελεί έναν κατευθυνόμενο γράφο που έχει ως κόμβους τους τεχνητούς νευρώνες, και ως ακμές τις συναπτικές συνδέσεις και συνδέσεις ενεργοποίησης ή μεταφοράς. Τα τεχνητά νευρωνικά δίκτυα διακρίνονται σε διάφορες κατηγορίες ανάλογα με την αρχιτεκτονική δομή τους και την εκπαίδευσή τους. Οι κατηγορίες αρχιτεκτονικής των τεχνητών νευρωνικών δικτύων είναι τέσσερις:

- *Δίκτυα πρόσθιας τροφοδότησης ενός επιπέδου* (feedforward neural networks). Είναι η πιο απλή μορφή νευρωνικού δικτύου. Αποτελείται από ένα επίπεδο εισόδου το οποίο μεταφέρει τα σήματα εισόδου στο επόμενο επίπεδο το οποίο είναι το επίπεδο των νευρώνων εξόδου, οι οποίοι είναι και οι υπολογιστικοί νευρώνες.



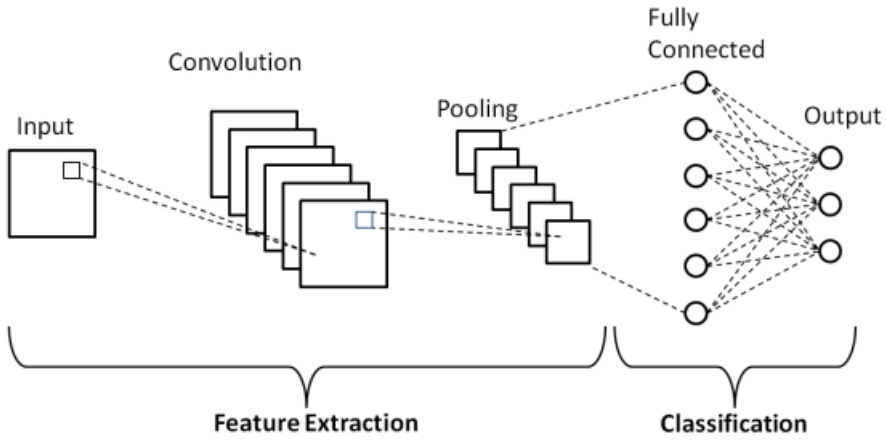
Εικόνα 4. Δίκτυο πρόσθιας τροφοδότησης ενός επιπέδου

- *Πολυεπίπεδα δίκτυα πρόσθιας τροφοδότησης* (multilayer neural networks). Περιέχουν ένα ή και περισσότερα κρυφά επίπεδα νευρώνων. Είσοδος των νευρώνων, σε κάθε επίπεδο, είναι τα σήματα εξόδων των νευρώνων του προηγούμενου επιπέδου. Πιο αναλυτικά, οι νευρώνες εισόδων μεταφέρουν τις πληροφορίες χωρίς καμία επεξεργασία στους νευρώνες κρυφούς νευρώνες. Οι κρυφοί νευρώνες όπως και οι νευρώνες εξόδου είναι οι υπολογιστικοί.



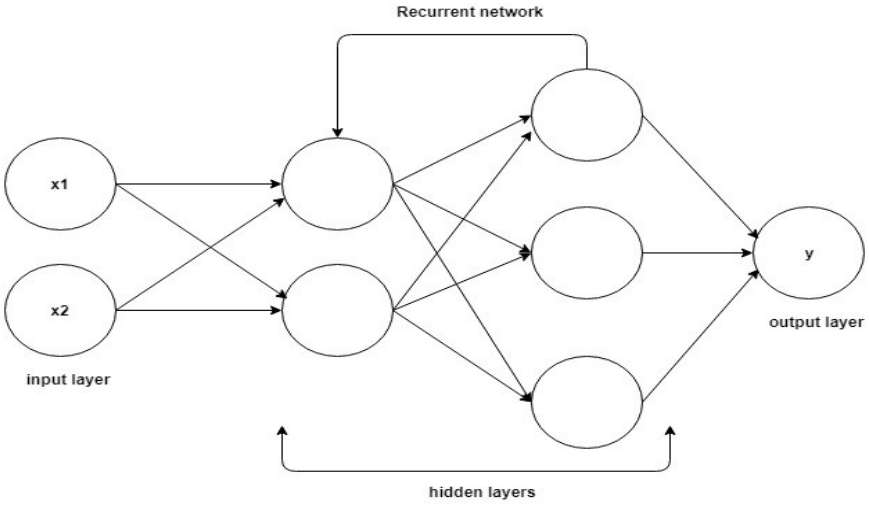
Εικόνα 5. Πολυεπίπεδο δίκτυο πρόσθιας τροφοδότησης

- *Συνελικτικά δίκτυα* (convolutional neural networks). Περιλαμβάνουν τρία διαφορετικά επίπεδα νευρώνων, και συνήθως χρησιμοποιούνται για την αναγνώριση εικόνων τις οποίες δέχονται ως είσοδο. Το πρώτο επίπεδο νευρώνων ονομάζεται συνελικτικό επίπεδο και σε αυτό εισέρχονται χαρακτηριστικά εικόνων στα οποία εφαρμόζονται φίλτρα. Τα δεδομένα που εξάγονται περνούν στο επόμενο κύριο επίπεδο, το επίπεδο συγκέντρωσης, όπου πραγματοποιείται μείωση των υπολογισμών του δικτύου με χρήση παραμέτρων. Τα αποτελέσματα αυτά προωθούνται στο πλήρως συνδεδεμένο επίπεδο, το οποίο παράγει και την τελική έξοδο του νευρωνικού δικτύου.



Εικόνα 6. Συνελικτικό δίκτυο

- *Αναδρομικά δίκτυα* (recurrent neural networks/RNN). Η έξοδος κάθε νευρώνα έχει την ιδιότητα να ανατροφοδοτεί την είσοδο των άλλων που βρίσκονται στο ίδιο επίπεδο αλλά μερικές φορές και τη δική τους είσοδο. Σε αυτή την κατηγορία των νευρωνικών δικτύων υπάρχει τουλάχιστον ένας βρόχος ανάδρασης, οποίος επηρεάζει την απόδοση και τη δυνατότητα μάθησης του νευρωνικού δικτύου.



Εικόνα 7. Αναδρομικό δίκτυο

Επιπροσθέτως, τα τεχνητά νευρωνικά δίκτυα αποτελούν άρτια συστήματα για την εύρεση πολύπλοκων και πολυάριθμων μοτίβων που είναι δύσκολο να αναπτυχθούν από τους προγραμματιστές. Δυο βασικές λειτουργίες του τεχνητού νευρωνικού δικτύου είναι η μάθηση και η ανάκληση. Αναφερόμενοι στον όρο ανάκληση υπονοούμε την διαδικασία επεξεργασίας ενός διανύσματος εισόδου σε συνάρτηση των βαρών, δίνοντας ως αποτέλεσμα τον υπολογισμό της τιμής εξόδου. Η μάθηση περιγράφει τη διαδικασία τροποποίησης των βαρών ώστε για κάθε σήμα εισόδου να παράγεται αντίστοιχα ένα συγκεκριμένο σήμα εξόδου. Επιπλέον, περιγράφει την εφαρμογή παραδειγμάτων και αλγορίθμων εκπαίδευσης στα

τεχνητά νευρωνικά δίκτυα η οποία βελτιστοποιεί την απόδοση τους, έχοντας ως γνώμονα την ελαχιστοποίηση των σφαλμάτων.

Οι παράμετροι των νευρωνικών δικτύων μπορούν να αλλάξουν με πολλούς τρόπους, επομένως υπάρχουν πολλοί διαφορετικοί μέθοδοι μάθησης. Παρόλα αυτά, η μέθοδος που τροποποιούνται τα βάρη κατά την εκπαίδευση του επιτρέπει την διάκριση της μάθησης σε τρία είδη:

- Επιβλεπόμενη μάθηση (Supervised learning)
- Μάθηση χωρίς επίβλεψη (Unsupervised learning)
- Ενισχυτική μάθηση (Reinforcement learning)

2.3 Συναρτήσεις υπολογισμού απώλειας

Η διαδικασία της εκπαίδευσης ενός νευρωνικού δικτύου ουσιαστικά αναπαριστά την προσπάθεια βελτιστοποίησης του αποτελέσματος του προβλήματος, δηλαδή την εύρεση των ακριβέστερων βαρών που πρέπει να εφαρμόζονται στα δεδομένα εισόδου του κάθε νευρώνα. Για να μετρηθεί η επίδοση του μοντέλου χρησιμοποιούμε μια συνάρτηση υπολογισμού απώλειας (loss function) η οποία προσπαθεί να εκτιμήσει τη διαφορά της προβλεπόμενης από την επιθυμητή έξοδο. Πιο απλά η συνάρτηση αυτή δείχνει το πόσο ορθά έχουν επιλεχθεί τα βάρη των συνάψεων για κάθε επίπεδο. Ο αλγόριθμος βελτιστοποίησης ανανεώνει τα βάρη κατά τη διαδικασία της εκπαίδευσης έχοντας ως στόχο την ελαχιστοποίηση του αποτελέσματος της συνάρτησης απώλειας. Η επιλογή των συναρτήσεων των απωλειών γίνεται ανάλογα με τον τρόπο μάθησης του νευρωνικού δικτύου και το επιθυμητό αποτέλεσμα. Παρακάτω αναφέρονται δύο από τις πιο συνήθεις συναρτήσεις απώλειας:

- Το Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error). Χρησιμοποιείται σε μοντέλα πρόβλεψης τιμών (regression loss functions).

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

- Η Απώλεια Διεντροπίας (Cross Entropy Loss). Εφαρμόζεται σε μοντέλα συσχέτισης μιας ετικέτας με την είσοδο (binary/Multi-Class classification loss functions).

$$L_{cross-entropy}(\hat{y}, y) = -\sum_{i}^{k} y_i \log(y_i)$$

2.4 Βιβλιοθήκες ανοικτού λογισμικού για νευρωνικά δίκτυα

2.4.1 Εξέλιξη του ανοιχτού λογισμικού

Από τις αρχές της εμφάνισης των υπολογιστικών συστημάτων, ερευνητές και προγραμματιστές μοιραζόταν μεταξύ τους και έκαναν χρήση κοινών μεθόδων, προγραμμάτων και λογισμικών με σκοπό να

αλληλοβοηθηθούν και να εξελίξουν τεχνικές, μεθόδους και ιδέες επάνω στον κλάδο της πληροφορικής. Με αυτό τον τρόπο δημιουργήθηκε και το λογισμικό ανοιχτού κώδικα (Open source software) του οποίου ο πηγαίος κώδικας (source code), συνοδεύεται από κάποιες άδειες χρήσης και μπορεί ελεύθερα να χρησιμοποιηθεί από κάποιο τρίτο, με στόχο την μελέτη, τον πειραματισμό και τη δημιουργία νέων προγραμμάτων και υπολογιστικών μεθόδων. Παράδειγμα open source code αποτελεί και το GNU το οποίο έκανε την εμφάνισή του το 1983 από τον Richard Stallman, και χαρακτηρίζεται ως η βάση της εξέλιξης της οικογένειας του λειτουργικού συστήματος Linux. Ένα ακόμη σημαντικό γεγονός για την κοινωνία των ερευνητών και το προγραμματιστών είναι η ίδρυση του μη κερδοσκοπικού οργανισμού Free Software Foundation, από τον Stallman το 1985, δίνοντας έτσι το έναυσμα και την ελευθερία για πειραματισμό, δημιουργία, εξάσκηση και τροποποίηση του λογισμικού αυτού. Στις μέρες πλέον σύνηθες φαινόμενο είναι τμήματα κώδικα, σε διάφορες γλώσσες (C, C++, Python, Java κ.ά.), να είναι κοινόχρηστα και ελεύθερα να χρησιμοποιηθούν από τους προγραμματιστές παρέχοντας συντομία και ευχρηστία στην υλοποίηση προγραμματιστικών μεθόδων.

Η ανάγκη ανάπτυξης μοντέλων που υλοποιούν τεχνητά νευρωνικά δίκτυα οδήγησε στη δημιουργία βιβλιοθηκών ανοικτού λογισμικού, όπως το TensorFlow και το Keras API που αναφέρονται παρακάτω.

2.4.2 TensorFlow

Η μαθηματική βιβλιοθήκη TensorFlow [8] αναπτύχθηκε από την ομάδα τεχνητής νοημοσύνης Google Brain [9], της Google, και διατέθηκε για δημόσια χρήση το Νοέμβριο του 2015. Είναι το αποτέλεσμα που επέφερε η προσπάθεια εξέλιξης και αναδιάταξης του κώδικα του DistBelief [10], λογισμικού της Google, με σκοπό την δημιουργία μιας ισχυρότερης βιβλιοθήκης. Χρησιμοποιείται για την πραγματοποίηση υπολογισμών που λαμβάνουν μέρος στα νευρωνικά δίκτυα όπως για παράδειγμα ο προγραμματισμός και η εκπαίδευση των μοντέλων (π.χ. για αναγνώριση εικόνων). Πιο αναλυτικά, βοηθά στην υλοποίηση δομών στις οποίες εισέρχονται και επεξεργάζονται ροές από πληροφορίες και στη διαδικασία δημιουργίας κόμβων στους οποίους μεταφέρεται η ανάλογη ροή δεδομένων. Ουσιαστικά οδηγεί στον σχηματισμό γράφου που αναπαριστά το νευρωνικό δίκτυο. Η ανάπτυξη μοντέλων στο περιβάλλον του TensorFlow γίνεται μέσω της γλώσσας προγραμματισμού Python.

2.4.3 Keras API

Το Keras API [11] [12] είναι βιβλιοθήκη πηγαίου κώδικα γραμμένη σε γλώσσα προγραμματισμού Python, και χρησιμοποιείται για την παροχή μεθόδων και συναρτήσεων κατά την ανάπτυξη ενός τεχνητού νευρωνικού δικτύου (open-source neural network). Υλοποιήθηκε από τον Francois Chollet το 2015 με σκοπό την εφαρμογή ταχύτερων πειραμάτων πάνω στα τεχνητά νευρωνικά δίκτυα. Ουσιαστικά το Keras συγκροτεί μια διεπαφή προγραμματισμού εφαρμογών η οποία παρέχει ένα σύνολο

διεργασιών/υπηρεσιών σε άλλα προγράμματα. Καθορίζει τις κλήσεις του συστήματος, το σύνολο των δεδομένων και τις συνδέσεις που πρέπει να πραγματοποιηθούν, μεταξύ πολλαπλών ενδιάμεσων λογισμικών. Τρέχει πάνω σε προγράμματα όπως τα TensorFlow, Microsoft Cognitive Toolkit [13], R [14], Theano [15]και PlaidML [16]. Η ανακάλυψη του ήταν αποτέλεσμα του ερευνητικού project Open-ended Neuro-Electronic Intelligent Robot Operating System. Η χρήση του Keras είναι μια από τις βέλτιστες λύσεις, καθώς σχεδιάστηκε ώστε να διευκολύνει τον προγραμματιστή παρέχοντας υποστήριξη για την ανάπτυξη συνελικτικών νευρωνικών δικτύων (CNN) ή ανατροφοδοτούμενων (RNN) ή και τον συνδυασμό των δύο (RCNN) καθώς περιλαμβάνει πολυάριθμες υλοποιήσεις των συνηθισμένων τεχνητών και των συναρτήσεων που απαιτούνται. Επιτρέπει επίσης στον κώδικα να τρέχει αδιάκοπα και σε συνεχή χρόνο στη CPU ή στη GPU. Υποστηρίζει την γρήγορη ανάπτυξη και εκπαίδευση νευρωνικών δικτύων βαθέας μάθησης, περιλαμβάνοντας διάφορες αρχιτεκτονικές (layer sharing, multi-input multi-output, model sharing κ.λπ.) και εργαλεία που καθιστούν ευκολότερη την επεξεργασία δεδομένων (κειμένου και εικόνας), απλοποιώντας τον απαιτούμενο κώδικα με χρήση διεπαφών υψηλού επιπέδου.

ΚΕΦΑΛΑΙΟ 3 Μεθοδολογία

3.1 Συλλογή δεδομένων προς εκπαίδευση

Η ορθή συλλογή των δεδομένων που χρησιμοποιούνται ως είσοδοι (inputs) στα νευρωνικά δίκτυα αποτελεί ένα από τα βασικότερα κομμάτια της ανάπτυξης ενός μοντέλου που οδηγεί αυτόνομα, δηλαδή τη δημιουργία ενός πράκτορα που έχει ιδιότητες αυτόματου πιλότου. Η υλοποίηση μιας εφαρμογής που θα έχει μηδενικό ποσοστό λαθών είναι ακατόρθωτη καθώς απαιτείται η καταγραφή τεράστιου όγκου δεδομένων του οποίου η συλλογή δεν μπορεί να πραγματοποιηθεί σε πραγματικό χρόνο.

Για τη δημιουργία της συλλογής δεδομένων για την εκπαίδευση του νευρωνικού δικτύου διατρέχθηκε η πίστα Luigi's Raceway του Mario Kart 64 χρησιμοποιώντας τον εξομοιωτή Bizhawk [17], όπου ο χειρισμός ανατέθηκε σε πραγματικό χρήστη. Με αυτό τον τρόπο καταγράφηκαν τα δεδομένα, τα οποία είναι μια σειρά από εικόνες και κινήσεις που θα χρησιμοποιηθούν ως είσοδος για την εκμάθηση του νευρωνικού δικτύου. Παρέχει τη δυνατότητα χρήσης ενός script γραμμένου σε γλώσσα προγραμματισμού Lua το οποίο βρίσκεται στο μενού του Bizhawk Console και παρέχει τη δυνατότητα καταγραφής και χειρισμού των κινήσεων του παίκτη. Για την καταγραφή των δεδομένων, αρχικά πρέπει να φορτώσουμε την πίστα Luigi's Raceway και να τη θέσουμε σε κατάσταση αναμονής. Επόμενο βήμα είναι να πατήσουμε "Start" για να αρχίσει η καταγραφή, και στη συνέχεια να εκκινήσουμε την εξομοίωση της πίστας. Στο τέλος κάθε τρεξίματος υπάρχουν μερικά λεπτά καθυστέρησης όπου πραγματοποιείται αποθήκευση δεδομένων σε αρχείο .txt. Ακόμα, μας δίνεται η δυνατότητα αποθήκευσης καταστάσεων κατά τη διάρκεια κάθε τρεξίματος του Luigi's Raceway. Επιπλέον για την αποθήκευση πληροφοριών που θα βοηθήσουν στην ανάκαμψη του μοντέλου από εσφαλμένες καταστάσεις, μπορούμε κατά την εκπαίδευση σκόπιμα να οδηγήσουμε το Kart σε αυτές και να καταγράψουμε πλαίσια πιέζοντας τα πλήκτρα L & R του χειριστηρίου. Στις εισόδους αυτές δεν προσδίδονται βάρη κατά την είσοδό τους στα νευρωνικά δίκτυα. Για να σταματήσουμε την καταγραφή πατάμε το "Stop".

3.2 Αρχιτεκτονική συνελικτικών νευρωνικών δικτύων

3.2.1 Εισαγωγή

Τα συνελικτικά νευρωνικά δίκτυα αποτελούν μια από τις πιο σύγχρονες αρχιτεκτονικές που χρησιμοποιούνται στα τεχνητά νευρωνικά δίκτυα για την αναγνώριση προτύπων και την ανάλυση άλλων ζητημάτων όρασης υπολογιστών (computer vision), όπως η αναγνώριση βίντεο και εικόνων. Η τεχνική των συνελικτικών νευρωνικών δικτύων αναδείχθηκε ως η κυρίαρχος μέθοδος διεξαγωγής εργασιών πάνω στο τομέα της υπολογιστικής όρασης από τον Alex Krizhevsky στο διαγωνισμό ImageNet Large Scale Visual Recognition Competition [18] το 2012, καταφέρνοντας να μειώσει το

ποσοστό των λαθών από το 26% στο 15%. Τα συνελικτικά νευρωνικά δίκτυα δέχονται κυρίως ως είσοδο δεδομένα από συλλογές εικόνων, τα οποία αναλύονται σε πολλές περιπτώσεις βάση τρισδιάστατων δομών σύμφωνα με το ύψος, το βάθος και το πλάτος της εικόνας. Απαρτίζονται από νευρώνες εισόδου, νευρώνες εξόδου και πολλαπλά κρυφά επίπεδα νευρώνων. Τα μοντέλα αυτών των νευρωνικών δικτύων ακολουθούν την αρχιτεκτονική πρόσθιας τροφοδότησης, δηλαδή η ροή δεδομένων διατρέχει τα μοντέλα χωρίς να υπάρχουν ανατροφοδοτήσεις. Η δομή των συνελικτικών νευρωνικών δικτύων είναι εμπνευσμένη από τη δομή του οπτικού φλοιού του εγκεφάλου, ο οποίος περιέχει τμήματα κυττάρων τα οποία δέχονται ερεθίσματα από το πεδίο όρασης και για παράδειγμα ενεργοποιούνται από την παρουσία οριζόντιων ή κάθετων γραμμών. Σε γενικές γραμμές, σκοπός των συνελικτικών νευρωνικών δικτύων είναι να μπορέσουν να αναλύσουν μια εικόνα που έχουν ως είσοδο, δηλαδή να εξάγουν και κατηγοριοποιήσουν χαρακτηριστικά ξεκινώντας από αυτά χαμηλότερου επιπέδου, όπως καμπύλες και ακμές, με τη βοήθεια μιας σειράς συνελικτικών επιπέδων (image classification). Η ποικιλία των πληροφοριών που εξάγονται από την ανάλυση μιας εικόνας δημιουργεί ανάγκη για την υπολογιστική επεξεργασία μεγάλων μοντέλων και δεδομένων η οποία επιλύεται με την χρήση πολλών αντίγραφων των ίδιων νευρώνων, διατηρώντας έτσι χαμηλό τον αριθμό των προοδιοριστικών παραμέτρων.

Γενικότερα, τα συνελικτικά νευρωνικά δίκτυα αποτελούν παραλλαγή των πολυεπίπεδων perceptron, με τα τελευταία επίπεδα νευρώνων των συνελικτικών νευρωνικών δικτύων δημιουργούν ένα δίκτυο MLP. Τα δίκτυα αυτά απαρτίζονται από τρία διαφορετικά επίπεδα:

1. *Τα Συνελικτικά Επίπεδα (Convolution layers).* Είναι τα κύρια δομικά επίπεδα των συνελικτικών νευρωνικών δικτύων, ενώ μπορεί να αποτελούνται από ένα ή και περισσότερα επίπεδα με διαφορετικά μεγέθη. Κάθε συνελικτικό επίπεδο είναι είσοδος ενός δομικού μπλοκ του επόμενου επιπέδου, ενώ τα βάρη διατηρούνται ίδια για όλους τους επόμενους νευρώνες. Τείνουν να τοποθετούνται στην αρχή έτσι ώστε να αποδομήσουν την πολυπλοκότητα της εισόδου μειώνοντας τον διανυσματικό χώρο της. Γίνεται χρήση ενός συνόλου φίλτρων που ξεχωρίζουν μοτίβα και χαρακτηριστικά των εικόνων της εισόδου και παράγουν χάρτες χαρακτηριστικών. Τα φίλτρα αυτά διαθέτουν 3 διαστάσεις, όπως η εικόνα, και συνήθως μικρότερες από αυτής διατηρώντας το μέγεθος του βάθους ίδιο, αφού κάθε ένα από αυτά συνελίσσεται με την κάθε εικόνα.
2. *Τα Επίπεδα Συγκέντρωσης (Pooling layers).* Τοποθετούνται συνήθως μετά από τα συνελικτικά επίπεδα, αλλά και ανάμεσά τους. Η είσοδος που δέχονται είναι η έξοδος του προηγούμενου συνελικτικού επιπέδου και παράγουν μια μοναδική τιμή ως έξοδο. Σκοπός τους είναι η μείωση του αριθμού των παραμέτρων άρα και των υπολογισμών του συνελικτικού νευρωνικού δικτύου. Η έξοδος που παράγουν τα επίπεδα συγκέντρωσης μπορεί να είναι η μέγιστη τιμή, η μέση τιμή, μία γραμμική παλινδρόμηση κ.λπ.

3. *Πλήρως συνδεδεμένα επίπεδα (Fully connected layers)*. Τα επίπεδα αυτά αποτελούν πλέγματα νευρώνων που υιοθετούν τη δομή του MLP. Οι νευρώνες έχουν πλήρης συνδέσεις με όλες τις εξόδους των νευρώνων προηγούμενου επιπέδου. Στόχος αυτού του επιπέδου είναι η κατηγοριοποίηση της εικόνας εισόδου σε μια από τις κατηγορίες που δόθηκαν κατά την εκπαίδευση του τεχνητού νευρωνικού δικτύου, σύμφωνα με τα χαρακτηριστικά υψηλού επιπέδου που δόθηκαν από το συνελικτικό και συγκεντρωτικό επίπεδο. Δημιουργείται δηλαδή ένα μοντέλο απόφασης.

3.2.2 Μοντέλο ανάπτυξης τεχνητού νευρωνικού δικτύου

Για την εκπαίδευση του πράκτορα που αναπτύξαμε, χρησιμοποιήσαμε το μοντέλο που προτάθηκε από την NVIDIA το 2016, το οποίο αναπτύχθηκε για υλοποίηση πράκτορα αυτόματης οδήγησης και βασίζεται σε συνελικτικά νευρωνικά δίκτυα [19]. Συγκεκριμένα, χαρτογραφεί δεδομένα εικόνας που λαμβάνει από κάμερα που είναι τοποθετημένη στη μπροστινή όψη του αμαξιού. Το μοντέλο αυτό απαιτεί τη συλλογή δεδομένων ύστερα από αρκετές ώρες οδήγησης, καθώς επίσης εμπλουτίζεται και με ενέργειες που βοηθούν στην ανάκαμψη από εσφαλμένες καταστάσεις, όπως όταν ο πράκτορας βρίσκεται σε λάθος κατεύθυνση ή σε λάθος θέση. Συνεπώς, δημιουργεί μεγαλύτερη συλλογή δεδομένων για την εκπαίδευση του συνελικτικού νευρωνικού δικτύου. Για τον υπολογισμό της απώλειας των κινήσεων, το μοντέλο υπολογίζει τη διαφορά ανάμεσα στη προβλεπόμενη και καταγεγραμμένη γωνία πλοήγησης. Στα αρχικά στάδια της εκπαίδευσης είναι απαραίτητη η παρέμβαση του χειριστή για την ανάκαμψη από εσφαλμένες καταστάσεις.

3.2.3 Υλοποίηση της αρχιτεκτονικής συνελικτικού νευρωνικού δικτύου

Στα πλαίσια της παρούσας εργασίας αναπτύχθηκε ένα συνελικτικό νευρωνικό δίκτυο, χρησιμοποιώντας την διεπαφή του Keras, το οποίο βασίζεται στην αρχιτεκτονική του ήδη ανεπτυγμένου μοντέλου νευρωνικού δικτύου TensorKart. Παρακάτω, στον Αλγόριθμο 1, παραθέτουμε τμήμα κώδικα από τη συνάρτηση που αναπτύχθηκε για τη δημιουργία του νευρωνικού δικτύου και αναπαριστά μία από τις κυριότερες συναρτήσεις του script, την *def create_model(keep_prob=0.8)*.

```
def create_model(keep_prob = 0.8):
model = Sequential()

model.add(Conv2D(24, kernel_size=(5, 5), strides=(2, 2),
activation='relu', input_shape= INPUT_SHAPE))

model.add(Conv2D(36, kernel_size=(5, 5), strides=(2, 2),
activation='relu'))
```

```
    model.add(Conv2D(48, kernel_size=(5, 5), strides=(2, 2),
    activation='relu'))

    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(Flatten())
    model.add(Dense(1164, activation='relu'))

    drop_out = 1 - keep_prob

    model.add(Dropout(drop_out))
    model.add(Dense(100, activation='relu'))
    model.add(Dropout(drop_out))
    model.add(Dense(50, activation='relu'))
    model.add(Dropout(drop_out))
    model.add(Dense(10, activation='relu'))
    model.add(Dropout(drop_out))
    model.add(Dense(OUT_SHAPE, activation='softsign'))


    return model
```

Αλγόριθμος 1. Δημιουργία μοντέλου νευρωνικού δικτύου

Πιο αναλυτικά, θα μελετήσουμε τις συναρτήσεις και τις τεχνικές που χρησιμοποιήθηκαν για τη υλοποίηση της αρχιτεκτονικής του μοντέλου.

1. Η συνάρτηση *model.add(Conv2D(…))* χρησιμοποιείται για την προσθήκη πέντε συνεχόμενων συνελικτικών επιπέδων (2D-convolutional layers) σειριακά. Εντός της συνάρτησης περνιούνται παράμετροι των δικτύων, ενώ στο πρώτο επίπεδο ορίζεται και το μέγεθος της εικόνας εισόδου.
2. Η συνάρτηση *model.add(Flatten())* προσθέτει ένα επίπεδο ισοπεδωτικής συνάρτησης (flatten) μετά από το πέμπτο συνελικτικό επίπεδο και πριν το πρώτο πλήρως συνδεδεμένο επίπεδο. Το επίπεδο αυτό μετατρέπει τον δυσδιάστατο πίνακα των χαρακτηριστικών σε ένα διάνυσμα τιμών το οποίο θα εισαχθεί στο πλήρως συνδεδεμένο επίπεδο.
3. Η *model.add(Dense(…))* προσθέτει πέντε πλήρως συνδεδεμένα επίπεδα (dense layers). Τα επίπεδα αυτά χρησιμοποιούν τα αποτελέσματα που δόθηκαν από το συνελικτικό επίπεδο ύστερα από την επεξεργασία τους από την ισοπεδωτική συνάρτηση με σκοπό να αντιστοιχίσουν την εικόνα με μία ετικέτα.
4. Η *model.add(Dropout(dropout))* προσθέτει τέσσερα επίπεδα περιορισμού ενεργοποίησης (dropout layers), τοποθετημένα το κάθε ένα ύστερα από κάθε πλήρως συνδεδεμένο επίπεδο.

Χρησιμοποιούνται για την αποφυγή της υπερποσαρμογής (overfitting). Σε αυτό το επίπεδο απενεργοποιούνται τυχαία νευρώνες σε κάθε επανάληψη της εκπαίδευσης, με σκοπό να μειώσουμε τα σφάλματα της.

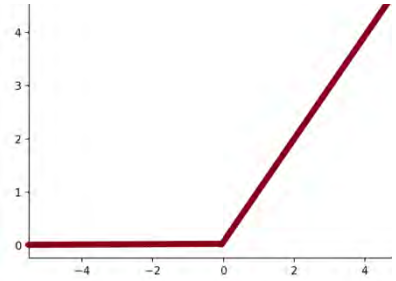
Στον Πίνακα 1 περιγράφονται οι παράμετροι της συνάρτησης πρόσθεσης των συνελικτικών επιπέδων (Conv2D(filters, kernel_size, strides, activation, input_shape)).

Παράμετροι Συναρτήσεων	**Περιγραφή**
Filters	Δεκαδικός αριθμός που υποδεικνύει τον αριθμό των φίλτρων κατά τη διαδικασία της συνέλιξης. Ο αριθμός που επιλέγουμε εξαρτάται από την πολυπλοκότητα του συνόλου δεδομένων και το βάθος του νευρωνικού δικτύου. Τα συνελικτικά επίπεδα που βρίσκονται πιο κοντά στην έξοδο του δικτύου έχουν και μεγαλύτερο αριθμό φίλτρων
kernel_size=(k,l)	Περιγράφει το μέγεθος των φίλτρων (ύψος και πλάτος) των συνελικτικών επιπέδων. Το μέγεθος αυτό εξαρτάται από την αρχιτεκτονική που χρησιμοποιούμε, και είναι περιττός αριθμός.
Activation	Ορίζει τη συνάρτηση ενεργοποίησης.
strides= (m,n)	Περιγράφει το βήμα που πρέπει να ακολουθήσει το φίλτρο πάνω στους άξονες ύψους και πλάτους της εικόνας εισόδου (βήμα της συνέλιξης). Για να μειωθεί ο θόρυβος της εικόνας θέτουμε m=n=2.
input_shape	Global μεταβλητή (1x3 πίνακας). Ορίζεται ως είσοδος της εικόνας στο πρώτο συνελικτικό επίπεδο.

Πίνακας 1. Παράμετροι συναρτήσεων

Η συνάρτηση ενεργοποίησης που χρησιμοποιήθηκε στα πέντε συνελικτικά επίπεδα είναι η ReLu (Εικόνα 8), η οποία είναι και η συνηθέστερη διότι έχει αποδειχθεί ότι συχνά αποδίδει καλύτερα από τις υπόλοιπες συναρτήσεις [20]. Είναι μια συνάρτηση η οποία επιστρέφει μηδέν αν λάβει αρνητική είσοδο και για οποιαδήποτε θετική είσοδο επιστρέφει την ίδια τιμή. Εφαρμόζεται στους πίνακες των χαρακτηριστικών, οι οποίοι έχουν παραχθεί από την εφαρμογή των φίλτρων στην εικόνα εισόδου.

Συμβολίζεται ως:
f(x)=max(0,x)



Εικόνα 8. Αναπαράσταση της ReLu

Η δεύτερη κυριότερη συνάρτηση του script που δημιουργήθηκε για την εκπαίδευση του νευρωνικού δικτύου είναι η συνάρτηση απώλειας (loss function) και ορίζεται στον κώδικα ως *def customized_loss()*. Για να κατανοηθεί η χρήση αυτής της συνάρτηση γίνεται μια σύντομη αναφορά στην τεχνική του regularization, μέθοδος η οποία βοηθά στην αντιμετώπιση του προβλήματος της υπερποσαρμογής.

Η υπερπροσαρμογή (overfitting) αποτελεί ένα από τα σημαντικότερα προβλήματα στην υλοποίηση τεχνικών νευρωνικών δικτύων και προκύπτει όταν το δίκτυο τροφοδοτείται με μεγάλο όγκο δεδομένων προς εκπαίδευση σε σχέση με το μέγεθός του. Συγκεκριμένα, όταν το σύνολο των δεδομένων περιέχουν θόρυβο ή όταν δεν έχει απομονωθεί μέσω των επιπέδων περιορισμού ενεργοποίησης μεγάλος αριθμός νευρώνων, υπάρχει πιθανότητα το δίκτυο να απομνημονεύσει τα δεδομένα εκπαίδευσης χωρίς να τα έχει γενικεύσει και να μάθει νέες καταστάσεις. Συνεπώς, θα παρέχει σωστές προβλέψεις για τα εισερχόμενα δεδομένα εκπαίδευσης αλλά λανθασμένες προβλέψεις για άλλες εισόδους που δεν έχει εκπαιδευτεί. Ο πιο σύνηθες τρόπος αντιμετώπισης της υπερποσαρμογής είναι η τεχνική του regularization, δηλαδή η εφαρμογή περιορισμών που επιτρέπουν στους παραμέτρους του νευρωνικού δικτύου (βάρη και κατώφλια) να επιλέγουν τις χαμηλότερες τιμές. Για να επιτευχθεί αυτό εφαρμόζουμε μια συνάρτηση κόστους (cost function) που σχετίζεται με τα βάρη του νευρωνικού δικτύου για να υπολογίσουμε την απώλεια από τις επιθυμητές προβλέψεις. Υπάρχουν δύο είδη συναρτήσεων κόστους που λαμβάνουν μέρος στη μέθοδο regularization:

- L1_norm Cost, αποτελεί το άθροισμα των απόλυτων τιμών της διαφοράς των αντίστοιχων επιθυμητών τιμών από τις τιμές που υπολογίστηκαν.

$$J(\Theta) = \sum_{i=1}^{m} |y_i - h_\theta(x_i)|$$

- L2_norm Cost, είναι το άθροισμα των τετραγώνων των διαφορών μεταξύ των αντίστοιχων επιθυμητών τιμών από τις τιμές που υπολογίστηκαν.

$$J(\theta) = \sum_{i=1}^{m} (y_i - h_\theta(x_i))^2$$

Παρακάτω, στον Αλγόριθμο 2, παρουσιάζεται η υλοποίηση της συνάρτησης απώλειας (*def customized_loss(y_true, y_pred, loss='euclidean'))*.

```
def customized_loss(y_true, y_pred, loss='euclidean'):

if loss == 'L2':
L2_norm_cost = 0.001
val = K.mean(K.square((y_pred  - y_true)), axis=-1)  \
+ K.sum(K.square(y_pred),  axis=-1)/2  * L2_norm_cost

elif loss == 'euclidean':

val = K.sqrt(K.sum(K.square(y_pred-y_true),  axis=-1))

return val
```

Αλγόριθμος 2. Υπολογισμός συνάρτησης απώλειας

Πιο συγκεκριμένα, για τον υπολογισμό της διαφοράς ανάμεσα στην προβλεπόμενη και καταγεγραμμένη κίνηση, η συνάρτηση απώλειας υπολογίζεται με τη χρήση του Mean Squared Error ή εναλλακτικά χρησιμοποιώντας την Ευκλείδεια απόσταση μεταξύ της καταγεγραμμένης κίνησης και αυτής που έχει προβλεφθεί. Το Keras διαθέτει μια βιβλιοθήκη που ονομάζεται backend, η οποία χρησιμοποιείται για την κλήση συναρτήσεων, όπως της μέσης τιμής *mean()*, του αθροίσματος *sum()*, της τετραγωνικής ρίζας *sqrt()* και του τετραγώνου *square()*, οι οποίες χρησιμοποιούνται στους υπολογισμούς της *customized_loss()*.

Για να ξεκινήσει η διαδικασία της εκπαίδευσης, όπως βλέπουμε στον Αλγόριθμο 3, αρχικά στη main φορτώνουμε τα δεδομένα προς εκπαίδευση (training data), τα οποία καταγράφηκαν από το προσομοιωτή Bizhawk.

```
if __name == '__main__':

x_train = np.load("data/X.npy")
y_train = np.load("data/y.npy")

print(x_train.shape[0],  'train samples')

epochs = 100
batch_size = 50

model = create_model()
```

```
model.compile(loss=customized_loss, optimizer=optimizers.adam())

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
shuffle=True, validation_split=0.1)

model.save_weights('model_weights.h5')
```

Αλγόριθμος 3. Εκπαίδευση νευρωνικού δικτύου

Με τις παρακάτω εντολές επιτυγχάνεται η φόρτωση των δεδομένων:

- *x_train = np.load("data/X.npy")*
- *y_train = np.load("data/y.npy")*

Οι εικόνες που θέλουμε να φορτώσουμε στο δίκτυο είναι αποθηκευμένες σε μορφή πίνακα από pixels στο αρχείο X.npy. Σε κάθε μία από αυτές αντιστοιχεί και μία ετικέτα (δηλαδή η είσοδος από το χειριστήριο). Το σύνολο των ετικετών είναι αποθηκευμένο σε μορφή πίνακα στο αρχείο Y.npy**.**

Έχοντας ορίσει τον αριθμό των πακέτων δεδομένων (*batch_size*) και επαναλήψεων/εποχών (*epochs*) τους κατά τη διάρκεια κάθε γύρου της εκπαίδευσης, επόμενο στάδιο είναι η δημιουργία του μοντέλου με την εντολή *model=create_model()* το οποίο στη συνέχεια θα εκπαιδευτεί. Η εκπαίδευση συμβαίνει με τη χρήση της *model.compile(),* η οποία δέχεται ως παραμέτρους την τιμή που επέστρεψε η συνάρτηση απώλειας και την επιλογή του αλγορίθμου βελτιστοποίησης του Adam (Adam optimizer). Σκοπός είναι η βελτιστοποίηση της επίδοσης του μοντέλου μέσα από την αλλαγή των βαρών και των κατωφλίων, και κατά συνέπεια η βελτιστοποίηση της loss function.

Το τελικό στάδιο είναι να προσαρμόσουμε στο μοντέλο τα δεδομένα προς εκπαίδευση. Αυτό υλοποιείται με τη συνάρτηση *model.fit()* η οποία δέχεται ως παραμέτρους:

- Τα δεδομένα εκπαίδευσης (*x_train)*
- Μία στήλη για την αποθήκευση των προβλέψεων (*y_train*)
- Τον αριθμό των δεδομένων που διατρέχουν το τεχνητό νευρωνικό δίκτυο σε κάθε γύρο εκπαίδευσης (*batch_size*)
- Τον αριθμό που δηλώνει πόσες φορές τα δεδομένα θα διαπεράσουν το δίκτυο (*epochs*)
- Τα προς επαλήθευση δεδομένα (*validation_data*), για την εκτίμηση του κάθε γύρου εκπαίδευσης. Έχουν το ίδιο μέγεθος με τα πακέτα των χαρακτηριστικών προς εκπαίδευση. Τα λάθη της εκπαίδευσης που ανιχνεύονται κατά την επαλήθευση αποθηκεύονται σε ειδικό φάκελο, όταν παρατηρηθεί μείωση της τιμής της συνάρτησης απώλειας. Σε περίπτωση που επιθυμούμε να διακόψουμε τη διαδικασία της εκπαίδευσης πατάμε ctrl+Break, δεν πραγματοποιείται αποθήκευση και έχουμε τη δυνατότητα συνέχισης της εκπαίδευσης από το τελευταίο σημείο ελέγχου του φακέλου.

Τέλος, τα βάρη του μοντέλου και η αρχιτεκτονική του αποθηκεύονται μέσω της εντολής *model.save_weights('model_weights.h5')*.

3.3 Τεχνικές εκμάθησης

3.3.1 Μάθηση μέσω μίμησης: εφαρμογή του αλγόριθμου Dagger

Η εκπαίδευση ενός αυτόνομου πράκτορα που διατρέχει μια πίστα Mario Kart 64 σε πραγματικό χρόνο συνήθως πραγματοποιείται με τη μέθοδο μάθησης μέσω μίμησης (imitation learning). Πιο συγκεκριμένα, είναι ένα είδος μάθησης με επίβλεψη, έχοντας ως ροές πληροφοριών δεδομένα που καταγράφηκαν από κάποιον χειριστή. Χαρακτηριστικό της μάθησης μέσω μίμησης είναι ότι βασίζεται στην αλληλεπίδραση ευφυών πρακτόρων, όπου ο ένας είναι teacher-agent και ο δεύτερος ο student-agent. Το ρόλο του teacher-agent τον αναλαμβάνει ο χειριστής. Για να καταγραφούν τα απαραίτητα δεδομένα πού θα εισαχθούν στο νευρωνικό δίκτυο πρέπει πρώτα ο χειριστής να διατρέξει την πίστα, όπου κατά τη διάρκεια του παιχνιδιού σε κάθε βήμα αποθηκεύονται χαρακτηριστικά του περιβάλλοντος, παρατηρήσεις, κίνδυνοι που πρέπει να αποφευχθούν κ.ά. Έτσι αυτές οι καταγραφές εισάγονται ως πληροφορίες στα δίκτυα τα οποία θα μιμηθούν τις κινήσεις του χειριστή. Το γεγονός ότι ο άνθρωπος μπορεί να αποφύγει πολλά λάθη καταγράφοντας έτσι λιγότερες πληροφορίες για την επαναφορά τους από αυτά οδηγεί στο γεγονός ότι ο χειριστής που δημιουργείται από το δίκτυο δεν μπορεί να διορθώσει τον εαυτό του ή να επανέλθει.

Για να επιλυθεί αυτό το πρόβλημα χρησιμοποιήθηκε ο αλγόριθμος Dagger [21], που ακολουθεί επαναλαμβανόμενη εκπαίδευση. Το πλεονέκτημα είναι ότι ο χειριστής έχει τη δυνατότητα να διορθώσει τα σφάλματα του νευρωνικού δικτύου. Αρχικά τα δεδομένα στα οποία εκπαιδεύεται το δίκτυο είναι αυτά που καταγράφηκαν από το τρέξιμο της πίστας από τον ίδιο. Το νευρωνικό δίκτυο τερματίζοντας, με τα δεδομένα της αρχικής κατάστασης, μας επιτρέπει να σημειώσουμε ποιες ενέργειες το οδήγησαν σε εσφαλμένη κατάσταση. Κατά αυτό τον τρόπο, ο χειριστής συγκεντρώνει νέα δεδομένα, τα οποία εμπεριέχουν νέες κινήσεις για την αποφυγή των λανθασμένων ενεργειών ή για την ανάκαμψη του νευρωνικού δικτύου από εσφαλμένες καταστάσεις. Αυτά τα δεδομένα εισέρχονται στο νευρωνικό δίκτυο και ύστερα από την εκ νέου εκπαίδευσή του, οδηγούμαστε σε νέα κατάσταση. Η νέα κατάσταση που δημιουργήθηκε περιέχει τα δεδομένα από το τρέξιμο της πίστας από τον πράκτορα, αλλά και τις ενέργειες που καταγράφηκαν από τον χειριστή για την αποφυγή εσφαλμένων καταστάσεων. Το δίκτυο εκπαιδεύεται στη νέα κατάσταση για να αποθηκευτούν τα δεδομένα που προστέθηκαν. Αυτή η διαδικασία επαναλαμβάνεται αρκετές φορές και ο αλγόριθμος επιστρέφει μια λίστα με κανόνες που προήλθαν κατά την εκπαίδευση, ώστε ο χειριστής να μπορέσει να επιλέξει αυτόν που θα επιφέρει λιγότερα σφάλματα.

3.3.2 Ενισχυτική μάθηση

Πριν την ανάλυση του κώδικα για την υλοποίηση του πράκτορα θα αναφερθούμε στην ενισχυτική μάθηση, η οποία εστιάζει στην ικανότητα του πράκτορα να εκτελεί ενέργειες χωρίς να αντιγράφει άλλα πρότυπα, αλλά σύμφωνα με επιβραβεύσεις που λαμβάνει ύστερα από την αλληλεπίδραση του με το περιβάλλον που βρίσκεται. Η μέθοδος αυτή χρησιμοποιείται σε μοντέλα μηχανικής μάθησης που πρέπει να πάρουν μια σειρά από αποφάσεις και περιλαμβάνει συναρτήσεις επιβράβευσης ή τιμωρίας (rewards/penalties) για τις ενέργειες που εκτελεί ο πράκτορας. Η συνάρτηση επιβράβευσης (reward function) αντιστοιχεί την κάθε κατάσταση που βρίσκεται ο πράκτορας με έναν αριθμό, δηλώνοντας με αυτό τον τρόπο εάν η συγκεκριμένη κατάσταση που βρίσκεται είναι επιθυμητή. Κατά την αλληλεπίδραση του με το περιβάλλον, στόχος του πράκτορα είναι να μεγιστοποιήσει την τιμή του reward, ύστερα από πολλές δοκιμές. Πιο αναλυτικά, ο πράκτορας συνδέεται με το περιβάλλον μέσω των παρατηρήσεων (observations), οι οποίες αποθηκεύονται σε πίνακα από pixels, και των κινήσεών του (actions). Σε κάθε κατάσταση που βρίσκεται, παίρνει μία παρατήρηση και αποφασίζει ποια θα είναι η κίνηση που θα εκτελέσει, η οποία αποτελεί και την έξοδό του. Έτσι μεταφέρεται σε μία νέα κατάσταση, ενώ παράλληλα του επιστρέφεται ένα reward ως βαθμολόγηση της κατάστασης που έχει βρεθεί. Ο πράκτορας προσπαθεί να επιλέξει κινήσεις που θα αυξήσουν το άθροισμα των επιβραβεύσεών του (total_reward). Η εκπαίδευση ολοκληρώνεται ύστερα από πολλά βήματα.

3.4 Υλοποίηση ευφυούς πράκτορα

Για τη δημιουργία πράκτορα, υλοποιήθηκε ένα script γραμμένο σε γλώσσα προγραμματισμού Python, το οποίο δημιουργεί μια κλάση Agent (*class Actor(object)*). Στον Αλγόριθμο 4 παραθέτουμε τη συνάρτηση δημιουργίας/αρχικοποίησης του πράκτορα την *def __init__(self)*:

```
def __init__(self):

self.model = create_model(keep_prob=1)  # no dropout
self.model.load_weights('model_weights.h5')

self.real_controller = XboxController()
```

Αλγόριθμος 4. Δημιουργία πράκτορα

Με τη χρήση της λέξη-κλειδί *self* αποκτάμε πρόσβαση σε όλα τα χαρακτηριστικά και τις μεθόδους που ορίζονται στη κλάση. Η μέθοδος *__init__* στην Python αποτελεί constructor αντικειμένων και καλείται όταν δημιουργούνται τα αντικείμενα μιας κλάσης, παρέχοντας τη δυνατότητα αρχικοποίησης των χαρακτηριστικών της.

- *self.model = create_model (keep_prob=1)*

Με αυτό τον τρόπο φορτώνουμε το μοντέλο που εκπαιδεύθηκε στο script που περιεγράφηκε προηγουμένως θέτοντας το *keep_prob* ίσο με "1" για να μην έχουμε dropout και να έχουμε ακρίβεια όταν τρέχει ο πράκτορας. Στο script έχουμε τον υπολογισμό του dropout από την φόρμουλα *drop_out = 1 - keep_prob*, οδηγώντας σε μηδενικό dropout κατά την εκτέλεση του πράκτορα. Η τεχνική του dropout είναι μέθοδος regularization και χρησιμοποιείται μόνο κατά την εκπαίδευση του μοντέλου, διότι η τεχνική αυτή ουσιαστικά οδηγεί σε χάσιμο πληροφοριών, το οποίο συνεπάγεται και χάσιμο ακρίβειας.

- *self.model.load_weights ('model_weights.h5')*

Με τη συνάρτηση *load_weights()* φορτώνουμε στο μοντέλο τα βάρη που αποθηκεύτηκαν (στο αρχείο *'model_weights.h5'*) από την εκπαίδευσή του. Το Keras API μας επιτρέπει να αποθηκεύσουμε όλα ή μερικά στοιχεία του μοντέλου (όπως η αρχιτεκτονική, τα βάρη και τα δεδομένα) σε ένα αρχείο τύπου HDF5, ύστερα από την εκπαίδευση του νευρωνικού δικτύου.

Στη συνέχεια, θέτουμε ως τον πραγματικό χειριστή τις εισόδους από το χειριστήριο με τη χρήση της συνάρτησης *self.real_controller=Xbox Controller()*. Επιπροσθέτως, θα πρέπει να ορίσουμε με ποιο τρόπο θα αποφασιστεί αν οι ενέργειες που θα εκτελούνται θα είναι από την είσοδο ενός ανθρώπινου χειριστή ή από τις προβλέψεις του πράκτορα. Για αυτό το λόγο ορίζεται η συνάρτηση *get_action(self, obs)* με παραμέτρους το μοντέλο του πράκτορα που δημιουργήθηκε και τις παρατηρήσεις που λαμβάνει από το περιβάλλον.

Στον Αλγόριθμο 5 βρίσκεται ο κώδικας της δεύτερης κύριας συνάρτησης *def get_action()*, όπου ορίζεται ποιος ενεργεί και επιστρέφει τις κατάλληλες ενέργειες.

```
def get_action(self, obs):
manual_override = self.real_controller.LeftBumper == 1

if not manual_override:

    vec = resize_image(obs)
    vec = np.expand_dims(vec, axis=0)
   joystick = self.model.predict(vec, batch_size=1)[0]

else:

   joystick = self.real_controller.read()
   joystick[1] *= -1
```

Αλγόριθμος 5. Ενέργειες που εκτελεί ο πράκτορας

Σε αυτό το τμήμα κώδικα θέτουμε ως *manual_override* παράμετρο η οποία επιστρέφει την τιμή "1" όταν πατηθεί το πλήκτρο του χειριστηρίου που ορίστηκε, και "0" σε διαφορετική περίπτωση.

Στη περίπτωση που *manual_override*==0, ο πράκτορας ενεργεί αυτόνομα. Μέσω των παρατηρήσεων (*obs*), καταχωρούμε στιγμιότυπο της κατάστασης στην οποία βρισκόμαστε και περνάμε στο vec τις διαμορφωμένες διαστάσεις της εικόνας. Η εντολή *joystick = self.model.predict(vec, batch_size=1)[0]* χρησιμοποιεί το μοντέλο για να προβλέψει την επόμενη κίνηση. Η *get action* επιστρέφει ένα πίνακα από ακέραιες τιμές, όπου πραγματοποιείται η χαρτογράφηση και στρογγυλοποίηση των τιμών των κινήσεων, έτσι ώστε να περαστούν στη συνέχεια στην main όπου θα κατευθύνουν την εξέλιξη του παιχνιδιού.

Στη περίπτωση που *manual_override==1*, διαβάζουμε από τη μεταβλητή *real controller* την ενέργεια που καταγράφηκε.

Εντός της main, όπως βλέπουμε στον Αλγόριθμο 6, η διαδικασία ξεκινά με την προετοιμασία του περιβάλλοντος της πίστας Luigi's Raceway. Η βιβλιοθήκη Gym περιλαμβάνει πολλά ενσωματωμένα περιβάλλοντα τα οποία διαθέτουν κοινή διεπαφή και επιτρέπουν την ανάπτυξη πολλών αλγορίθμων [22]. Ακολουθούν τέσσερις συναρτήσεις της βιβλιοθήκης Gym που συμβάλουν στην προετοιμασία του περιβάλλοντος:

- *env = gym.make()* , η οποία φορτώνει τα στιγμιότυπα του περιβάλλοντος
- *env.reset(),* η οποία επιστρέφει την αρχική παρατήρηση
- *env.render(), η οποία* απεικονίζει την κατάσταση, δηλαδή ένα πλαίσιο του περιβάλλοντος
- *env.step(action),* η οποία οδηγεί το περιβάλλον στην επόμενη κατάσταση ανάλογα με την κίνηση που δόθηκε. Επιστρέφει τις εξής τέσσερις τιμές:
    - *obs* (object), ένα αντικείμενο που αναπαριστά τις παρατηρήσεις του περιβάλλοντος
    - *reward* (float), τιμή επιβράβευσης της προηγούμενης κίνησης που πραγματοποιήθηκε
    - *end_episode* (boolean), τιμή που υποδηλώνει εάν έχει τερματίσει η πίστα
    - *info,* αναπαριστά πληροφορία χρήσιμη για debugging

```
if __name == '__main__':
env = gym.make('Mario-Kart-Royal-Raceway-v0')

obs = env.reset()
env.render()
print('env ready!')

actor = Actor()
print('actor ready!')

print('beginning episode loop')
total_reward = 0
end_episode = False
while not end_episode:
    action = actor.get_action(obs)
    obs, reward, end_episode, info = env.step(action)
    env.render()
    total_reward += reward

print('end episode... total reward: ' + str(total_reward))
```

Αλγόριθμος 6. Εφαρμογή ενεργειών του πράκτορα

Έχοντας καλέσει και δημιουργήσει τον actor, βρισκόμαστε σε ένα while loop όπου μέσω της εντολής *action = actor.get_action(obs)* λαμβάνουμε τις επόμενες κινήσεις και καταστάσεις, έως το τέλος του τρεξίματος της πίστας (*end_episode*). Κάθε κίνηση του πράκτορα οδηγεί και σε μία καινούρια κατάσταση (που υπολογίζεται με την *env.step(action)*). Οι κινήσεις του πράκτορα στέλνονται στο περιβάλλον, το οποίο επιστρέφει παρατηρήσεις και μια τιμή *reward* που βαθμολογεί την επίδοση της ενέργειας του.

Τελικά, υπολογίζεται η τιμή μιας μεταβλητής της *total_reward* η οποία και εκτυπώνεται. Αποτελεί το άθροισμα των επιδόσεων των κινήσεων που λήφθηκαν από κάθε κατάσταση, και υποδηλώνει πόσο καλά απέδωσε ο πράκτορας.

ΚΕΦΑΛΑΙΟ 4 Αποτελέσματα και Συμπεράσματα

4.1 Περιγραφή τρεξίματος πίστας και αποτελέσματα

Για την εκπαίδευση πρέπει να ξεκινήσουμε συλλέγοντας δεδομένα, τα οποία θα χρησιμοποιηθούν ως είσοδος για την εκμάθηση του τεχνητού νευρωνικού δικτύου που αναπτύξαμε. Στο script που υλοποιήθηκε για τη δημιουργία και την εκπαίδευση του νευρωνικού δικτύου φορτώνονται μέσω της βιβλιοθήκης NumPy οι πίνακες των στιγμιότυπων του περιβάλλοντος και των ετικετών που προέρχονται τις κινήσεις του χειριστή από κάθε βήμα. Έπειτα, στα πλαίσια της πτυχιακής εργασίας δημιουργήθηκε ένας πράκτορας βασισμένος στην αρχιτεκτονική του TensorKart [23]. Το script που αναπτύχθηκε για την δημιουργία του πράκτορα και την εκτέλεση των κινήσεών του, έχει υλοποιηθεί σύμφωνα με το μοντέλο της αρχιτεκτονικής του νευρωνικού δικτύου που εκπαιδεύθηκε προηγουμένως. Ο πράκτορας προβλέπει και εκτελεί κινήσεις σύμφωνα με τα εισερχόμενα δεδομένα εκπαίδευσης του δικτύου χωρίς να έχει δυνατότητα χρήσης φρένου και επιλογής χρήσιμων αντικειμένων της πίστας. Με τη βοήθεια της βιβλιοθήκης Gym της Python απεικονίζονται εικόνες του περιβάλλοντος του πράκτορα, οι οποίες προκύπτουν από κάθε κίνηση που εκτελεί. Στα πρώτα πειράματα παρατηρήθηκε η αδυναμία του πράκτορα να πραγματοποιήσει στροφές, με αποτέλεσμα να ακολουθεί μόνο ευθεία πορεία και να συγκρούεται με τοίχους και δέντρα. Για αυτό το λόγο κρίθηκε αναγκαία η παρέμβαση του χειριστή, ο οποίος παίρνει τον έλεγχο και πραγματοποιεί τις κατάλληλες κινήσεις. Αποτέλεσμα αυτού ήταν να πραγματοποιηθούν αρκετές επαναλήψεις της προσομοίωσης τρεξίματος της πίστας Luigi's Raceway για τη δημιουργία μιας εκτενούς συλλογής δεδομένων προς εκπαίδευση. Όσες περισσότερες προσομοιώσεις πραγματοποιηθούν τόσα περισσότερα δεδομένα συλλέγονται για την πιο πλήρη και συνεπώς πιο ορθή εκμάθηση του πράκτορα. Σε επόμενες επαναλήψεις χρειάστηκε να συγκεντρωθούν πληροφορίες για την επιλογή της γωνίας πλοήγησης, δηλαδή σωστή επιλογή κατεύθυνσης, για αποφυγή εμποδίων και για την επιλογή κινήσεων για ανάκαμψη από εσφαλμένες καταστάσεις (οδήγηση εντός δρόμου, ξεμπλοκάρισμα από σύγκρουση με τοίχο κ.ά.). Συνεπώς, το τεχνητό νευρωνικό δίκτυο που δημιουργήθηκε, εκπαιδεύεται και μαθαίνει σύμφωνα με τα καταγεγραμμένα δεδομένα.

4.2 Συμπεράσματα

Παρατηρήθηκε πως η απόδοση του πράκτορα είναι σχεδόν ακατόρθωτο να έχει μηδενικό ποσοστό λαθών. Αυτό συμβαίνει διότι η καταγραφή των δεδομένων στα οποία εκπαιδεύεται το δίκτυο δεν επαρκεί για τη συλλογή όλων των απαραίτητων πληροφοριών με στόχο την αποφυγή εσφαλμένων καταστάσεων. Αυτό θα απαιτούσε μεγάλο χρονικό διάστημα για την καταγραφή όλων των πιθανών καταστάσεων που θα μπορούσε να εκτελέσει ο

πράκτορας. Το τρέξιμο της πίστας, από το χειριστή, επαναλήφθηκε δέκα φορές και μετρήθηκε ο μέσος χρόνος που χρειάστηκε για τον επιτυχή τερματισμό. Στη συνέχεια μετρήθηκε και ο μέσος χρόνος της απόδοσης του πράκτορα για άλλα δέκα τρεξίματα. Παρατηρήθηκε ότι ο υπολογιστής μπορεί να ανταγωνιστεί τον άνθρωπο, παρουσιάζοντας πολύ κοντινούς χρόνους με αυτούς τους χειριστή, όπως βλέπουμε στον Πίνακα 2.

Μέσος Χρόνος Απόδοσης Ανθρώπου	Μέσος Χρόνος Απόδοσης Πράκτορα
129,40 sec	134,20 sec

Πίνακας 2. Μέσος χρόνος τρεξιμάτων

Η μικρή απόκλιση που παρατηρούμε στους χρόνους απόδοσης οφείλεται σε προβλήματα που αντιμετωπίζει ο πράκτορας κατά την επιλογή της κλήσης της γωνίας για την πραγματοποίηση στροφών, στην οδήγηση σε άμμο ή γρασίδι και στην πρόσκρουσή του με τοίχους. Αναλυτικότερα, κατά τη διάρκεια της απόδοσης του πράκτορα παρατηρείται πως δεν επιλέγει πάντα τη βέλτιστη κλήση οδήγησης όταν θα πρέπει να πραγματοποιήσει μια ανοιχτή στροφή. Αυτό συμβαίνει διότι σε μια κλειστή στροφή είναι πιο εμφανής η κλήση που πρέπει να επιλέξει λόγω της εικόνας που λαμβάνει από τα στιγμιότυπα της πίστας. Συνεπώς, σε ανοιχτές στροφές τείνει να κατευθύνεται με ταχύτητα προς κάποιο τοίχο ή εκτός πορείας. Έχοντας ως στόχο την αύξηση της πιθανότητας ανάκαμψης του πράκτορα από εσφαλμένες καταστάσεις, και της ικανότητας του να αναγνωρίσει και να διορθώσει μόνος του λανθασμένες ενέργειες, χρειάστηκε να εξομοιώσουμε την πίστα επιλέγοντας τυχαίες καταστάσεις του παιχνιδιού στις οποίες βρισκόταν ο παίκτης έτσι ώστε να συγκεντρωθούν περισσότερα δεδομένα προς εκπαίδευση. Παρόλα αυτά, σε αρκετές περιπτώσεις ο πράκτορα δεν αναγνωρίζει επικίνδυνες συμπεριφορές, όπως ο άνθρωπος, με αποτέλεσμα να οδηγεί είτε παράλληλα είτε πάνω σε γρασίδι η άμμο επιβραδύνοντας την απόδοσή του, όπως διακρίνουμε στις Εικόνες 9 και 10.

Εικόνα 9. Ο πράκτορας οδηγεί "επικίνδυνα" έχοντας επιλέξει τροχιά πάνω σε γρασίδι με αποτέλεσμα την επιβράδυνση του

Εικόνα 10. Ο πράκτορας οδηγεί "επικίνδυνα" έχοντας επιλέξει τροχιά πάνω σε άμμο κατά την εκτέλεση της στροφής με αποτέλεσμα να επιβραδύνει

Ακόμα ένα σημαντικό γεγονός που προκαλεί την επιβράδυνση του πράκτορα είναι η αδυναμία επιλογής της κατάλληλης ενέργειας ύστερα από πρόσκρουση με τοίχο, όπως φαίνεται στην Εικόνα 11. Όταν ο πράκτορας βρίσκεται σε θέση τέτοια ώστε το στιγμιότυπο της εικόνας που λαμβάνει να είναι μόνο ένας τοίχος, όπως διακρίνουμε στην Εικόνα 12, τότε αργεί να επιλέξει την κατάλληλη ενέργεια για ανάκαμψη και βρίσκεται σε αδιέξοδο.

Εικόνα 11. Ο πράκτορας ύστερα από πρόσκρουση με τοίχο

Εικόνα 12. Πρόσκρουση με τοίχο με αδυναμία επιλογής της σωστής κίνησης για ανάκαμψη από αυτή την κατάσταση

Εικόνα 13. Ο πράκτορας βρίσκεται σε αντίθετη κατεύθυνση μετά την πρόσκρουση του με τοίχο

Η επιλογή αντίθετης κατεύθυνσης, όπως στην Εικόνα 13, θεωρείται ως αρνητική απόδοση και οδηγεί σε διακοπή του παιχνιδιού καθώς ο πράκτορας δεν έχει την ικανότητα να αναγνωρίσει το λάθος του. Επιπλέον, ένα ακόμη μειονέκτημα του πράκτορα που οδηγεί στην απώλεια χρόνου είναι το γεγονός ότι δεν μπορεί να πατήσει φρένο με αποτέλεσμα πολλές φορές να ολισθαίνει, όπως στην Εικόνα 14. Συνεπώς, σε καταστάσεις που η αναμενόμενη συμπεριφορά του είναι να παραμένει σταθερός στην κλήση που έχει επιλέξει χωρίς να προκαλεί μείωση της ταχύτητας του, ο πράκτορας παρασέρνεται από την πορεία του και επιβραδύνει

Εικόνα 14. "Επικίνδυνη" οδήγηση ύστερα από ολίσθηση και εκτροχιασμό

ΚΕΦΑΛΑΙΟ 5 Επίλογος

Παρατηρώντας την απόδοση του ευφυούς πράκτορα και συγκρίνοντας τους χρόνους για τον τερματισμό της πίστας μεταξύ αυτού και του χειριστή μπορούμε να συμπεράνουμε ότι ο άνθρωπος εξαιτίας της ικανότητάς του να αντιληφθεί και να εκμεταλλευτεί την σμίκρυνση του χάρτη της πίστας καταφέρνει να ολοκληρώσει την διαδρομή σε μικρότερο χρονικό διάστημα. Πιο συγκεκριμένα, η αναπαράσταση της πίστας τον ωφελεί στο να αποφύγει την σύγκρουση με τοίχους και να πραγματοποιήσει καλύτερα τις στροφές προσαρμόζοντας τις κινήσεις του χειριστηρίου. Σε αντίθεση με τον ανθρώπινο παίκτη, ο πράκτορας δεν μπορεί να αναγνωρίσει τον χάρτη που βρίσκεται σε μικρογραφία στο κάτω μέρος της εικόνας που λαμβάνει ως δεδομένο. Συνεπώς, θα μπορούσαν να χρησιμοποιηθούν επιπλέον μέθοδοι της υπολογιστικής όρασης ώστε ο πράκτορας να λαμβάνει πληροφορίες εκ των προτέρων για την κλήση της γωνίας που χρειάζεται να επιλέξει, αυτές μπορούν να δοθούν ως επιπλέον χαρακτηριστικά για την εκπαίδευση του νευρωνικού δικτύου. Ακόμα, θα μπορούσαν να εισαχθούν και επιπλέον χαρακτηριστικά που θα περιέχουν πληροφορίες για την αναγνώριση χρήσιμων αντικειμένων ή των εμπόδιων που εμφανίζονται κατά μήκος της διαδρομής. Επιπλέον, ωφέλιμη θα ήταν και η εισαγωγή επιπλέον κινήσεων του χειριστηρίου, οι οποίες θα δίνουν την ικανότητα στον πράκτορα να εκτελεί άλματα για να χρησιμοποιήσει τα αντικείμενα ή να αποφύγει τα εμπόδια που εμφανίζονται στη διαδρομή του. Επιπλέον, μια διαφορετική προσέγγιση στην υλοποίηση του πράκτορα θα ήταν η προσθήκη ενός επιπλέον επιπέδου νευρώνων που στηρίζεται στη μέθοδο της ενισχυτικής μάθησης. Ο πράκτορας δεν θα τρέχει σε πραγματικό χρόνο αλλά θα έχει την ιδιότητα να διαβάζει τη μνήμη όπου περιέχονται πληροφορίες που περιγράφουν το περιβάλλον και θα διατηρείται ένα σύνολο επιθυμητών μελλοντικών καταστάσεων. Σε κάθε βήμα το περιβάλλον θα επιστρέφει στον πράκτορα μια μεταβλητή επιβράβευσης, δίνοντας του τη δυνατότητα να αξιολογεί τις κινήσεις του πριν τις εκτελέσει. Στόχος του πράκτορα θα είναι να βελτιστοποιήσει την τιμή που λαμβάνει, επιλέγοντας ανάμεσα από τις προβλεπόμενες, την κίνηση που έχει επιφέρει την καλύτερη δυνατή τιμή επιβράβευσης. Συμπερασματικά, ο πράκτορας θα εκπαιδεύεται μόνος του έχοντας κάποιο στόχο και βελτιστοποιώντας παράλληλα το χρόνο της απόδοσής του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Artificial Intelligence,» Wikipedia, [Ηλεκτρονικό].
Available: https://en.wikipedia.org/wiki/Artificial_intelligence.

[2] «IBM Deep Blue,» [Ηλεκτρονικό].
Available: https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/.

[3] Stuart Russell, Peter Norvig, Τεχνητή Νοημοσύνη: Μια Σύγχρονη Προσέγγιση, 2η επιμ., ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ, p. 1200.

[4] Stephen Lucci, Danny Kopec, «Artificial Intelligence in the 21st Century,» σε *Artificial Intelligence in the 21st Century*, Stylus Publishing, 2015, p. 615.

[5] Amit Sharma, Sudesh Kumar, Mohammed Aslam, "A Comparative Study between Naïve Bayes and Neural Network (MLP) Classifier for Spam Email Detection," *Int. J. Comput.Appl,* 2014.

[6] Trevor Hastie, Robert Tibshiran, Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer Science & Business Media, 2009.

[7] «AI and Data Science,» NVIDIA, [Ηλεκτρονικό].
Available: https://www.nvidia.com/en-us/ai-data-science/.

[8] «TensorFlow,» [Ηλεκτρονικό]. Available: https://www.tensorflow.org/.

[9] «Tensorflow,» Wikipedia, [Ηλεκτρονικό].
Available: https://en.wikipedia.org/wiki/TensorFlow.

[10] Dean Jeffrey κ.ά., «Large Scale Distributed Deep Networks.,» *Advances in neural information processing systems,* pp. 1223-1231, 2012.

[11] «Keras API,» [Ηλεκτρονικό]. Available: https://keras.io/about/.

[12] Wikipedia, «Keras API,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Keras.

[13] «Cognitive toolkit,» Microsoft, [Ηλεκτρονικό].
Available: https://www.microsoft.com/en-us/research/product/cognitive-toolkit/.

[14] «R-project,» R Core Team, [Ηλεκτρονικό]. Available: https://www.r-project.org/.

[15] «Theano,» [Ηλεκτρονικό]. Available: http://www.deeplearning.net/software/theano/.

[16] «PlaidML,» Intel, [Ηλεκτρονικό].
Available: https://www.intel.com/content/www/us/en/artificial-intelligence/plaidml.html.

[17] «Bizhawk,» [Ηλεκτρονικό]. Available: http://tasvideos.org/Bizhawk.html.

[18] «ImageNet,» Wikipedia, [Ηλεκτρονικό].
Available: https://en.wikipedia.org/wiki/ImageNet#ImageNet_Challenge.

[19] Mariusz Bojarski κ.ά., «End to End Learning for Self-Driving Cars,» 2016. [Ηλεκτρονικό].
Available: arXiv:1604.07316 [cs.CV].

[20] M. Lanham, «Training a perceptron in Python,» σε *Hands-On Deep Learning for Games: Leverage the power of neural networks and reinforcement learning to build intelligent games*, Packt Publishing Ltd, 2019, p. 392.

[21] Ross Stéphane, Geoffrey Gordon, Drew Bagnell, «A reduction of imitation learning and structured prediction to no-regret online learning,» *Proceedings of the fourteenth international conference on artificial intelligence and statistics.,* pp. 627-635, 2011.

[22] «Gym toolkit,» [Ηλεκτρονικό]. Available: https://gym.openai.com/docs/.

[23] K. Hughes, «TensorKart,» Github, 2017. [Ηλεκτρονικό].
Available: https://github.com/kevinhughes27/TensorKart.