ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΑΝΑΠΤΥΞΗ ΠΡΑΚΤΟΡΩΝ ΕΝΙΣΧΥΤΙΚΗΣ ΜΑΘΗΣΗΣ ΣΕ ΤΡΙΣΔΙΑΣΤΑΤΑ ΕΙΚΟΝΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

Πτυχιακή εργασία

ΑΛΕΞΑΝΔΡΟΣ-ΠΑΝΑΓΙΩΤΗΣ ΚΡΗΤΙΚΟΣ

Αθήνα, Σεπτέμβριος 2019

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Βαρλάμης Ηρακλής
Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Καραγιώργου Σοφία
Επισκέπτης Λέκτορας,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Κωνσταντίνος Τσερπές
Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Ο Αλέξανδρος Κρητικός

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Σελίδα για Αφιέρωση (προαιρετικά)

Σελίδα για γνωμικό (προαιρετικά)

Ευχαριστίες

Ολοκληρώνοντας τη πτυχιακή μου εργασία, θα ήθελα να ευχαριστήσω όλους όσους βοήθησαν σε όλα τα στάδια ανάπτυξής της. Αρχικά, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια κ. Καραγιώργου Σοφία για την εξαιρετική συνεργασία καθώς και τις εύστοχες υποδείξεις καθ' όλη τη διάρκεια της εκπόνησης της εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω τους κ. Βαρλάμη Ηρακλή και κ. Τσερπέ Κωνσταντίνο για την βοήθεια τους και την ευκαιρία που μου έδωσαν να ασχοληθώ με το συγκεκριμένο θέμα. Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για τη διαρκή υποστήριξη και βοήθεια κατά τη διάρκεια της εργασίας καθώς και την παροχή ερεθισμάτων και συνεργασίας καθ' όλη τη διάρκεια των σπουδών μου.

Πίνακας Περιεχομένων

Περίληψη

Η σχεδίαση και υλοποίηση έξυπνων και αυτόνομων πρακτόρων Τεχνητής Νοημοσύνης, και ιδιαίτερα όσων αξιοποιούν τη Μηχανική Μάθηση (Machine Learning), αποτελεί μια σημαντική δοκιμασία για όσους τους ενδιαφέρει το γνωστικό αυτό πεδίο. Προαπαιτεί γνώσεις από διάφορους τομείς της Επιστήμης των Υπολογιστών, όπως είναι για παράδειγμα ο τομέας των Πιθανοτήτων και της Στατιστικής, ενώ στην παρούσα περίπτωση γίνονται σημαντικές αναφορές στην έννοια του Δυναμικού Προγραμματισμού.

Η παρούσα εργασία περιλαμβάνει την ανάλυση του χώρου καταστάσεων, των ενεργειών και άλλων μετρήσεων που αποτελούν ένα τρισδιάστατο περιβάλλον, πιο στοχευμένα το εικονικό τρισδιάστατο περιβάλλον του ηλεκτρονικού παιχνιδιού Doom (1993), ενώ επιδεικνύει τη χρήση σύγχρονων αλγορίθμων που ανήκουν στην υποκατηγορία της Μηχανικής Μάθησης που ονομάζεται Ενισχυτική Μάθηση (Reinforcement Learning). Γίνεται αξιοποίηση της πλατφόρμας μηχανικής μάθησης Tensorflow, καθώς και του Keras API που τρέχει πάνω από την πλατφόρμα αυτή και αποτελεί το πλέον δημοφιλέστερο abstraction API για υλοποιήσεις βαθιών νευρωνικών δικτύων (deep neural networks). Για το πρόβλημα το οποίο διερευνά η συγκεκριμένη εργασία, εφαρμόζονται τεχνικές που αξιοποιούν Deep-Q-Networks. Η παροχή του περιβάλλοντος, το οποίο τροφοδοτεί το πρακτορικό σύστημα με πληροφορίες για το χώρο, την κατάσταση της τρέχουσας συνεδρίας παιχνιδιού και την αξία των ενεργειών του συστήματος, γίνεται μέσω της πλατφόρμας ViZDoom (Kempka et al. 2016).

Όπως είναι λογικό, ο χώρος καταστάσεων στα τρισδιάστατα περιβάλλοντα ηλεκτρονικών βιντεοπαιχνιδιών μπορεί να κριθεί μη μετρήσιμος. Αυτό έχει ως αποτέλεσμα την αυξημένη πολυπλοκότητα στους υπολογισμούς που επιχειρούν τα πρακτορικά συστήματα ενισχυτικής μάθησης, ώστε να μπορέσουν να προχωρήσουν στην εξαγωγή αποφάσεων για την καταλληλότερη ενέργεια που απαιτείται σε κάθε δεδομένη κατάσταση. Οι προαναφερθείσες τεχνικές επιχειρούν να βελτιώσουν την απόδοση στην επεξεργασία τέτοιων περιβαλλόντων από πρακτορικά συστήματα, όπως επίσης και την κρίση τους για την εξαγωγή των κατάλληλων κινήσεων. Στο τέλος, αυτό που καταλήγουν να κάνουν τα συστήματα αυτά είναι η επίτευξη στόχων, όμοιων με εκείνους που επιδιώκει ένας

άνθρωπος κατά την ενασχόληση του με τα ανταγωνιστικά βιντεοπαιχνίδια, δηλαδή την προσωπική επιτυχία ή την νίκη του παίκτη.

Λέξεις – κλειδιά: Ενισχυτική μάθηση, τρισδιάστατα περιβάλλοντα, ViZDoom, Tensorflow, Keras

Abstract

The design and implementation of smart and autonomous Artificial Intelligence agents, more specifically of those who also exploit Machine Learning, is considered a great challenge for those who are interested in this specific field. It requires knowledge from various fields of Computer Science, for instance the field of Probabilities and Statistics, while there are important references to the concept of Dynamic Programming.

The current thesis includes the analysis of the state space and the actions that make up a three-dimensional environment, more specifically a virtual three-dimensional environment of the video-game Doom (1993), whereas it demonstrates the usage of state-of-the-art algorithms which belong to the subcategory of Machine Learning named Reinforcement Learning**.** The well-known Tensorflow machine learning platform is being utilized, as well as the Keras API which runs on top of the Tensorflow platform and is considered as the most popular abstraction API for deep neural network implementations. For the problem which this thesis investigates, various techniques that belong to the domain of Q-Learning are applied. The environment, which sends feedback related to the game state, the current game session and the reward for a given action back to the Reinforcement Learning agent, is provided by the ViZDoom platform (Kempka et al. 2016).

As it is quite obvious, the state space in three-dimensional video-game environments may be considered as non-measurable and complex. As a consequence, this leads to increased complexity in the computations that these agent systems make, in order to be able to extract an optimal decision-making policy for a given environment state. The previously mentioned techniques aim to improve the agent performance in processing such environments, as well as their judgement in the prediction of the appropriate actions. At last, what these agent systems tend to achieve is the completion of certain goals, identical to the goals a human has, during his avocation with adversarial video-games, which in most of the cases are centered around the idea of personal success or player's victory.

Keywords: Reinforcement Learning, three-dimensional environments, ViZDoom, Tensorflow, Keras

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΠΑΡΑΡΤΗΜΑΤΑ ΚΩΔΙΚΑ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

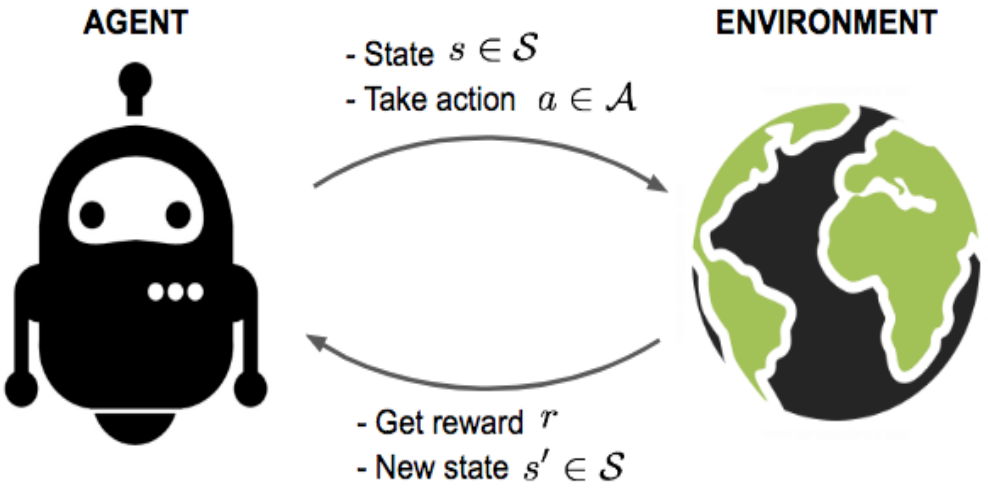
AI	Artificial Intelligence
ML	Machine Learning
SL	Supervised Learning
RL	Reinforcement Learning
API	Application Programming Interface
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DQN	Deep Q Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
TD	Temporal Difference
FPS	First-Person Shooter
MCTS	Monte Carlo Tree Search
TDP	Thermal Design Power
CPU	Central Processing Unit
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
MOBA	Multiplayer Online Battle Arena
RTS	Real-Time Strategy
APM	Actions Per Minute
RGB	Red-Green-Blue
ReLU	Rectified Linear Unit
MSE	Mean Squared Error

1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μια εισαγωγή στη θεματική ενότητα των Reinforcement Learning (RL) αλγορίθμων και συγκεκριμένα στον τρόπο με τον οποίο χρησιμοποιούν οι RL πράκτορες (agents) τους αλγορίθμους αυτούς για να μεγιστοποιήσουν την επίδοση τους σε μια δεδομένη συνεδρία παιχνιδιού (game session), μέσα από μία συνεχή αλληλεπίδραση του πράκτορα με το περιβάλλον (environment) του παιχνιδιού. Γίνεται μία παρουσίαση της εξέλιξης των αλγορίθμων αυτών στην πάροδο του χρόνου, αναλύεται η έννοια της βαθιάς μάθησης (deep learning), ενώ γίνεται και μία αναφορά στους τομείς που εφαρμόζονται οι αλγόριθμοι ενισχυτικής μάθησης και βαθιάς μάθησης. Τέλος, περιγράφεται το αντικείμενο της πτυχιακής εργασίας και δίνεται μια μικρή εικόνα για μελλοντικές της βελτιώσεις.

1.1 Ενισχυτική μάθηση

Η ενισχυτική μάθηση (reinforcement learning) στην επιστήμη των υπολογιστών είναι ένας γενικός όρος που έχει δοθεί σε μια οικογένεια τεχνικών στις οποίες το σύστημα μάθησης προσπαθεί να μάθει μέσα από την άμεση αλληλεπίδραση με το περιβάλλον. Αυτό που κάνει την ενισχυτική μάθηση να διαφέρει από την μάθηση με επίβλεψη (supervised learning) είναι το γεγονός πως απουσιάζει η έννοια των ζευγαριών εισόδου/εξόδου βασισμένων σε χαρακτηριστικά (labels).



Εικόνα 1: Αναπαράσταση ενισχυτικής μάθησης

1.1.1 Ορισμός ενισχυτικής μάθησης

Η έννοια της ενισχυτικής μάθησης είναι εμπνευσμένη από τα αντίστοιχα ανάλογα της μάθησης με επιβράβευση και τιμωρία που συναντώνται ως μοντέλα μάθησης των έμβιων όντων. Σκοπός του συστήματος μάθησης είναι να μεγιστοποιήσει μια συνάρτηση του αριθμητικού σήματος ενίσχυσης (ανταμοιβή), για παράδειγμα την αναμενόμενη τιμή του σήματος ενίσχυσης στο επόμενο βήμα. Το σύστημα δεν καθοδηγείται από κάποιον εξωτερικό χρήστη για το ποια ενέργεια θα πρέπει να ακολουθήσει αλλά πρέπει να ανακαλύψει μόνο του ποιες ενέργειες είναι αυτές που θα του αποφέρουν το μεγαλύτερο κέρδος. Η τεχνική αυτή είναι πολύ σημαντική για τα σημερινά παιχνίδια.

Ο πράκτορας (agent) είναι η οντότητα που μαθαίνει. Οτιδήποτε άλλο διαφορετικό από αυτό αποτελεί μέρος του περιβάλλοντος (environment). Περιβάλλον και πράκτορας αλληλεπιδρούν συνεχώς. Το πρώτο επιστρέφει ανταμοιβές (rewards) στο δεύτερο και ο δεύτερος επιλέγει ενέργειες (actions). Ο πράκτορας μαθαίνει από τις προηγούμενες εμπειρίες του κάτι το οποίο μακροπρόθεσμα του επιτρέπει να επιλέγει βέλτιστα τα actions. Αυτό επιτυγχάνεται με τη βόηθεια των rewards που λαμβάνονται για την κατάσταση που βρίσκεται ο πράκτορας. Μία δημοφιλής τεχνική ενισχυτικής μάθησης είναι το Q-learning. Η τεχνική αυτή είναι model free και μπορεί να χρησιμοποιηθεί για να βρεθεί μία βέλτιστη τακτική ενεργειών- δράσεων για πεπερασμένες Μαρκοβιανές Διεργασίες Απόφασης (MDP). Αυτή είναι και η τεχνική που χρησιμοποιήθηκε στον πράκτορα για το παιχνίδι που δημιουργήθηκε.

1.1.2 Σημαντικά επιτεύγματα

Τα τελευταία χρόνια έχουν γίνει σημαντικές βελτιώσεις στις τεχνικές και τους αλγορίθμους, που χρησιμοποιούνται από RL πράκτορες για την επίλυση προβλημάτων σχετικών με την εξεύρεση βέλτιστών τακτικών (policies) σε περιβάλλοντα ηλεκτρονικών βιντεοπαιχνιδιών. Ο *AlphaGo* (Google DeepMind 2015), ένας AI πράκτορας εκπαιδευμένος πάνω στο παιχνίδι GO μέσω ενισχυτικής μάθησης, κατέχει τον τίτλο του πρώτου λογισμικού που κατάφερε να ξεπεράσει τις επιδόσεις του ανθρώπου πάνω στο συγκεκριμένο παιχνίδι.

Μία ακόμα μεγάλη και σημαντική πρόοδο στον τομέα της ενισχυτικής μάθησης θεωρείται το **Five**, δουλειά μιας ερευνητικής ομάδας στην OpenAI (2017). Πρόκειται για

την υλοποίηση ενός RL πράκτορα, ο οποίος κατάφερε να κερδίσει σε μονομαχίες ένας-με-έναν πολλούς από τους κορυφαίους επαγγελματίες στο δημοφιλές και μεγάλης πολυπλοκότητας παιχνίδι Dota 2.

Πρόσφατη υλοποιήση με μεγάλη επιτυχία αποτελεί ο πράκτορας *AlphaStar* (Google DeepMind 2019), ο οποίος έχει σημειώσει σημαντικές νίκες απέναντι στους καλύτερους παίκτες του StarCraft II.

Φυσικά, άξιος αναφοράς θεωρείται και ο διαγωνισμός *Visual Doom AI Competition* (Wydmuch, Kempka, Mohanty, Jaskowski) που έχει αποτελέσει μία πολύ καλή ευκαιρία για τους ειδικούς της ενισχυτικής μάθησης να αξιολογήσουν νέους αλγορίθμους Q-Learning και Policy Gradient.

Εικόνα 2: Περιβάλλον του παιχνιδιού Starcraft II στο οποίο συμμετέχει ο πράκτορας AlphaStar της DeepMind

1.1.3 Βαθιά μάθηση (Deep Learning)

Ο όρος βαθιά μάθηση (deep learning) έρχεται από την ιδέα πως ένα νευρωνικό δίκτυο (neural network) αποτελείται από πολλά κρυφά επίπεδα (hidden layers) και ενσωματώνει ένα ευρύ φάσμα αρχιτεκτονικών. Καθιστάται απαραίτητη η χρήση μεγάλου πλήθους δεδομένων αλλά και υψηλή υπολογιστική ισχύ για την καλύτερη απόδοση σε deep-q-network αρχιτεκτονικές.

1.1.4 Εφαρμογές ενισχυτικής μάθησης

Η ενισχυτική μάθηση βρίσκει εφαρμογές στον έλεγχο κίνησης ρομπότ, στη βελτιστοποίηση εργασιών σε εργοστάσια, στη μάθηση επιτραπέζιων παιχνιδιών και φυσικά στην εύρεση βέλτιστων στρατηγικών σε ανταγωνιστικά ηλεκτρονικά βιντεοπαιχνίδια, όπως περιγράφει εξάλλου η συγκεκριμένη πτυχιακή εργασία.

1.2 Αντικείμενο πτυχιακής εργασίας

Απώτερος στόχος της συγκεκριμένης εργασίας είναι η υλοποίηση έξυπνων πρακτόρων οι οποίοι χρησιμοποιούν διάφορες αρχιτεκτονικές νευρωνικών δικτύων για να εντοπίσουν, με αποδοτικό τρόπο, βέλτιστες στρατηγικές που θα τους οδηγήσουν στη μεγιστοποίηση των μακροπρόθεσμων ανταμοιβών που δέχονται από το περιβάλλον με το οποίο αλληλεπιδρούν. Με πιο απλά λόγια, αναμένουμε οι πράκτορες αυτοί να καταλήξουν να γίνουν ανταγωνιστικοί, μέσα από μία επαναληπτική διαδικασία εκπαίδευσης (training).

Καταλληλότερο παιχνίδι για την εφαρμογή και την αξιολόγηση τέτοιων συστημάτων κρίθηκε το γνωστό FPS παιχνίδι Doom (id Software 1993). Πρόκειται για ένα από τα ιστορικότερα παιχνίδια στο είδος του, μιας και αυτό μαζί με ορισμένα ακόμα καθιέρωσαν τα δημοφιλή παιχνίδια *βολών πρώτου προσώπου* (first-person shooter). Για να μπορέσουν οι πράκτορες να αποκτήσουν πρόσβαση στην μηχανή του παιχνιδιού και στις διάφορες πληροφοριακές πηγές που παρέχονται, έγινε απαραίτητη η χρήση της πλατφόρμας ViZDoom, η οποία διευκολύνει στην ανάπτυξη AI συστημάτων που είναι ικανά να παίζουν το παιχνίδι, έχοντας ως οπτική πληροφορία την οθόνη (screen buffer). Εκεί που φάνηκε ιδιαίτερα χρήσιμη η πλατφόρμα ήταν στο ότι προσέφερε ένα πλήθος σεναρίων με παραμετροποιήσεις για το καθένα, αλλά και χρήσιμες μεθόδους για την διαχείριση του περιβάλλοντος και των σεναρίων αυτών.

1.2.1 Ερευνητικό πρόβλημα

Το ερευνητικό πρόβλημα εστιάζει στην αποδοτική απεικόνιση της όθονης του παιχνιδιού ως την αίσθηση που λαμβάνει το πρακτορικό σύστημα για το περιβάλλον στο οποίο καλείται να λειτουργήσει, όπως επίσης και στην εύρεση των κινήσεων που θεωρούνται καλύτερες στην δεδομένη κατάσταση της συνεδρίας παιχνιδιού, μέσα από μία διαδικασία προσπάθειας και λάθους (trial and error). Έχοντας μία χαμηλής πολυπλοκότητας αντίληψη για τον χώρο που δρα ο πράκτορας, μπορεί να οδηγηθεί σε μία

λιγότερο δαπανήρη εκτίμηση για τις ενέργειες που είναι προτιμότερο να εκτελεστούν σε κάθε χρονική στιγμή, καθώς επίσης θα χρειαστεί σαφώς λιγότερος χρόνος για να παραχθεί η συγκεκριμένη εκτίμηση. Καθιστάται λοιπόν απαραίτητη η σχεδίαση μίας μεθόδου που να μπορεί να απλοποιεί το πλήθος των δεδομένων που παράγονται και μεταφέρονται από την μηχανή του παιχνιδιού στον ίδιο τον πράκτορα. Η μεγάλη πρόκληση βρίσκεται στην υλοποίηση κατάλληλων τεχνικών για την αποδοχή των δεδομένων αυτών, αλλά και των διαθέσιμων κινήσεων που μπορεί να χρησιμοποίησει ο πράκτορας, με κύριο στόχο την παραγωγή εκτιμήσεων που μεγιστοποιούν τις μελλοντικές ανταμοιβές που λαμβάνει ο πράκτορας από το περιβάλλον.

1.2.2 Συνεισφορά

Στην συγκεκριμένη υποενότητα καταγράφονται με αριθμητική σειρά τα στάδια προσέγγισης και επίλυσης του προβλήματος που μελετά η εργασία αυτή. Η συνεισφορά της αναλυτικότερα έχει ως εξής:

1. Ανάλυση και επιλογή των σεναρίων που προσφέρονται από την πλατφόρμα ViZDoom και κρίθηκαν κατάλληλα για την εφαρμογή των RL τεχνικών.
2. Παραμετροποίηση των ρυθμίσεων (configuration files) που διατίθονται για κάθε ένα από τα παραπάνω σενάρια, ώστε να συμβαδίζουν με τις απαιτήσεις των RL υλοποιήσεων.
3. Υλοποίηση μεθόδων προεπεξεργασίας του περιβάλλοντος παιχνιδιού ωστέ η πληροφορία της εικόνας του παιχνιδιού να καταλήξει ως μία απλοποιημένη είσοδος για το δεδομένο νευρωνικό δίκτυο
4. Μελέτη και επιλογή ορισμένων από τους πιο σύγχρονους αλγορίθμους ενισχυτικής μάθησης. Η πλειοψηφία των αλγορίθμων αυτών ανήκει στο πεδίο του Q-Learning, ενώ εμφανίζεται και μία προσέγγιση που κληρονομεί SL τεχνικές και τις διατυπώνει στο πλαίσιο της ενισχυτικής μάθησης.
5. Κατασκευή βαθέων νευρωνικών δικτύων (deep neural networks) τα οποία ενσωματώνουν τις παραπάνω Q-Learning τεχνικες. Τα δίκτυα αυτά δημιουργήθηκαν με τη χρήση της deep learning βιβλιοθήκης Tensorflow και του Keras API. Η γλώσσα προγραμματισμού που προτιμήθηκε είναι η Python.

6. Σχεδίαση και υλοποίηση πρακτορικών οντοτήτων που ενσωματώνουν τις παραπάνω DNN αρχιτεκτονικές, καθώς και κλήση αυτών των οντοτήτων εντός μίας συνεδρίας παιχνιδιού.
7. Επαναληπτική εκπαίδευση του πράκτορα και αποθήκευση των παραμέτρων του μοντέλου του για μελλοντική χρήση και αξιολόγηση της επίδοσης του.
8. Παρακολούθηση του πράκτορα στην οθόνη παιχνιδιού και στατιστική ανάλυση των μετρικών και των επιδόσεων που αυτός παράγει εν ώρα δράσης

1.2.3 Περιορισμοί

Όπως προαναφέρθηκε, μεγάλη δυσκολία αποτελεί η εύρεση αποδοτικών τεχνικών που θα επιφέρουν μία πιο απλοποιημένη μορφή εισόδου στο νευρωνικό δίκτυο που καλείται να λειτουργήσει σε ένα οποιοδήποτε σενάριο. Ως είσοδο σε ένα DNN δίκτυο που αλληλεπιδρά με ένα περιβάλλον παιχνιδιού ορίζεται η εικόνα που αναπαράγεται από την μηχανή γραφικών του υποκείμενου σε μελέτη παιχνιδιού, αλλά και οι διαθέσιμες κινήσεις που μπορούν να επιλεχθούν για το συγκεκριμένο περιβάλλον. Οι είσοδοι αυτοί αναπαριστώνται ως διανύσματα (vectors), ή, για να χρησιμοποιούμε deep learning ορολογία, ως τένσορες (tensors). Η είσοδος της οθόνης, ή αλλιώς η είσοδος κατάστασης (state input) μπορεί να αναπαρασταθεί ως ένα διάνυσμα από pixels. Για υψηλές αναλύσεις οθόνης, στις οποίες εμπλέκεται και η έννοια του χρώματος, το διάνυσμα αυτό γίνεται υπερβολικά μεγάλο. Αν σκεφτούμε πως και η ύπαρξη των διαθέσιμων κινήσεων είναι και αυτή μια ακόμα είσοδος, η συγκέντρωση τόσο πολύπλοκων διανυσμάτων επηρεάζει αρνητικά τους υπολογισμούς του DNN.

Ακόμα, περιορισμός θεωρείται και η ύπαρξη της απαραίτητης υπολογιστικής ισχύος ώστε να υπάρχει βελτίωση στην απόδοση του υπό διεργασία συστήματος. Γενικώς η διαδικασία της εκπαίδευσης των DNN θεωρείται χρονοβόρα. Ο χρόνος εκπαίδευσης τέτοιων ML συστημάτων κυμαίνεται συνήθως από μερικές ώρες εώς και κάποιες εβδομάδες, ανάλογα βέβαια με την φύση του περιβάλλοντος και την αρχιτεκτονική που έχει επιλεχθεί για την επίλυση ενός συγκεκριμένου προβλήματος σε αυτό το περιβάλλον. Στην περίπτωση του Doom, η διάρκεια των συνεδριών εκπαίδευσης σε πολύπλοκα περιβάλλοντα δεν ξεπέρασε ποτέ την μία μέρα. Δεν παύει όμως να αποτελεί σημαντικό πρόβλημα για την παραγωγή και αξιολόγηση τέτοιων συστημάτων.

Εικόνα 3: Στιγμιότυπο από μία αρχική κατάσταση στο περιβάλλον παιχνιδιού Deadly Corridor. Ως κατάσταση ορίζεται μια γραφική αναπαράσταση της οθόνης σε μία δεδομένη χρονική στιγμή. Η κατάσταση είναι η είσοδος του νευρωνικού δικτύου

1.3 Οργάνωση κειμένου

Ακολουθεί η δομή με βάση την οποία οργανώνονται τα κεφάλαια της παρούσας πτυχιακής εργασίας. Στο Κεφάλαιο 2 γίνεται μία αναλυτικότερη περιγραφή από άποψη αλγοριθμικών υλοποιήσεων, σχετική με τις μελέτες και εργασίες των προαναφερθέντων ερευνητκών ομάδων της Google DeepMind και της OpenAI. Στο Κεφάλαιο 3 γίνεται η εμβάθυνση στις τεχνολογίες και τα εργαλεία που χρησιμοποήθηκαν για την κάλυψη των αναγκών της παρούσας εργασίας. Έπειτα, δίνεται η διερεύνηση των αλγορίθμων ενισχυτικής μάθησης που εφαρμόστηκαν στα επιλεγμένα περιβάλλοντα του Doom καθώς

και τα επιμέρους στάδια των εξαγόμενων στατιστικών αποτελεσμάτων. Το Κεφάλαιο 4 παρουσιάζει τα αποτελέσματα των πειραμάτων που έγιναν πάνω στα υλοποιημένα συστήματα συνοπτικά. Εν κατακλείδι, στο Κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα από την παρούσα πτυχιακή εργασία καθώς και οι επεκτάσεις που θα μπορούσαν να προκύψουν στο μέλλον. Γίνεται επίσης καταγραφή της βιβλιογραφίας που χρησιμοποιήθηκε για την συλλογή των απαραίτητων γνώσεων.

2 Σχετικές δραστηριότητες και εργασίες

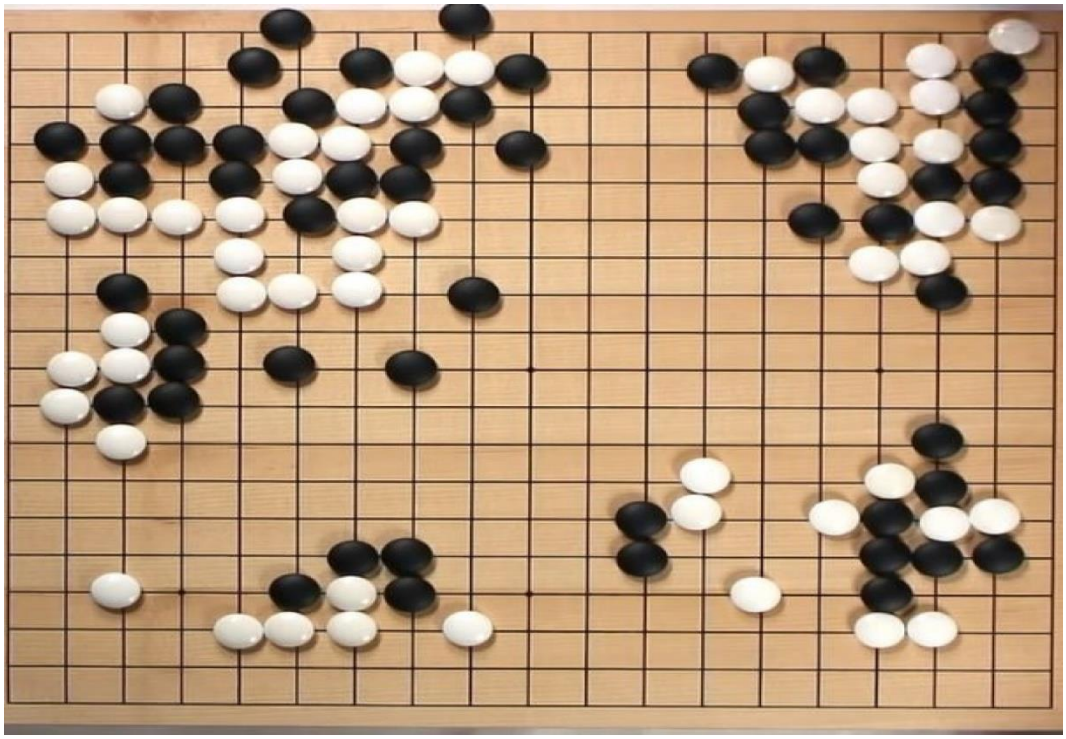
Τα τελευταία χρόνια έχει αυξηθεί δραματικά το πλήθος των καινοτομιών στο πλαίσιο της μηχανικής μάθησης και πιο συγκεκριμένα στον τομέα του reinforcement learning, από διάφορες ερευνητικές ομάδες. Επίσης, σήμερα παρέχονται αισθητά περισσότερα frameworks ανοιχτού κώδικα που καθιστούν ευκολότερη την υλοποίηση τέτοιων καινοτομιών, σε σύγκριση με το παρελθόν. Βέβαια, οι ανακαλύψεις αυτές δεν θα ήταν εφικτές αν δεν είχε προηγηθεί η σημαντική πρόοδος στην τεχνολογία υλικού (hardware). Παρακάτω εξηγούνται αναλυτικότερα οι μέθοδοι που αναφέρθηκαν στην ενότητα 1.1.2, σχετικά με τις πρόσφατες δραστηριότητες στον χώρο της ενισχυτικής μάθησης και των παιχνιδιών.

2.1 AlphaGo (Go)

Πρόκειται για ένα πρακτορικό RL σύστημα που αναπτύχθηκε από την Google DeepMind για να παίξει το επιτραπέζιο παιχνίδι Go. Τον Οκτώβριο του 2015, έγινε ο πρώτος υπολογιστής Go που νίκησε επαγγελματία παίκτη Go επί ίσοις όροις σε ένα σε φυσικό μέγεθος 19 × 19 ταμπλό. Τον Μάρτιο του 2016 ο υπολογιστής AlphaGo νίκησε τον επαγγελματία παίχτη Lee Sedol σε ματς 5 παιχνιδιών, η πρώτη φορά που πρόγραμμα υπολογιστή Go νικάει επαγγελματία 9 νταν χωρίς κάποιος να αρχίζει με μειονέκτημα. Παρόλο που ο υπολογιστής έχασε το τέταρτο παιχνίδι, ο Sedol παραιτήθηκε στο τελευταίο παιχνίδι, διαμορφώνοντας το τελικό 4-1 υπέρ του AlphaGo. Σε αναγνώριση της νίκης του επί του Sedol, απονεμήθηκε τιμητικά στον AlphaGo η κατηγορία 9 νταν, από την Κορεατική Ένωση Μπαντούκ (Korea Baduk Association). Το AlphaGo επιλέχθηκε από το περιοδικό Science ως ένας από τους 9 υποψήφιους πρωτοπόρους της χρονιάς στις 22 Δεκεμβρίου 2016.

Ο πράκτορας AlphaGo χρησιμοποιεί έναν αλγόριθμο MCTS (Monte Carlo Tree Search) με συνδυασμό τεχνικών μηχανικής μάθησης με διακλαδώσεις, και ειδικότερα με εντατική εξάσκηση σε παιχνίδια με ανθρώπους και σε παιχνίδια με τον υπολογιστή. Γίνεται επίσης χρήση DNN δικτύων. Αυτά τα δίκτυα λαμβάνουν ως είσοδο μία περιγραφή της

απεικόνισης του ταμπλό του παιχνιδιού και ύστερα ακολουθεί μία επεξεργασία των δεδομένων αυτής της εισόδου μέσα από ένα πλήθος διαφορετικών κρυφών επιπέδων, κάθε ένα από τα οποία περιλαμβάνει μερικά εκατομμύρια νευρωνικές συνδέσεις. Ένα νευρωνικό δίκτυο *τακτικής* (policy network), επιλέγει την επόμενη προς εκτέλεση κίνηση, ενώ ένα δεύτερο δίκτυο, το νευρωνικό δίκτυο *αξίας* (value network), προβλέπει τον νικητή του παιχνιδιού.

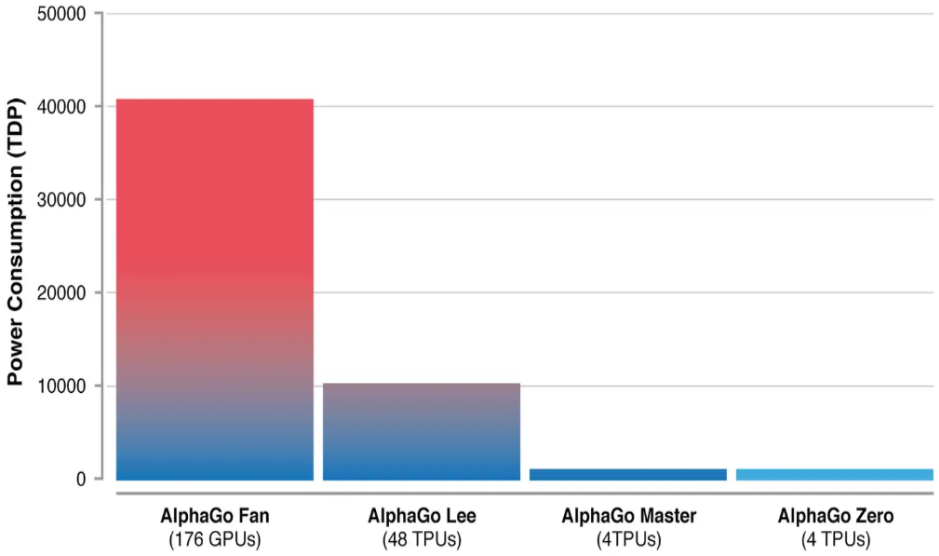


Εικόνα 4: Απεικόνιση του ταμπλό ενός παιχνιδιού Go

2.1.1 AlphaGo Zero

Μια πρόσφατη εξέλιξη του AlphaGo αποτελεί ο AlphaGo Zero. Σε αντίθεση με τον AlphaGo, ο οποίος μάθαινε να παίζει επαγγελματικά μετά από χιλιάδες παρτίδες με αρχάριους και επαγγελματίες παίχτες, ο AlphaGo Zero μαθαίνει παίζοντας ενάντια στον εαυτό του, ξεκινώντας με εντελώς τυχαίες κινήσεις. Ο πράκτορας αυτός μπόρεσε και συσσώρευσε χιλιάδες χρόνια ανθρώπινης εμπειρίας εντός μίας περιόδου λίγων ημερών και έμαθε να παίζει με την βοήθεια του καλύτερου παίχτη στον κόσμο, που δεν ήταν άλλος από τον ίδιο τον AlphaGo.

Κατάφερε μέσα σε μικρό χρονικό διάστημα και ξεπέρασε τις επιδόσεις όλων των προηγούμενων εκδόσεων του και μπόρεσε επίσης να χρησιμοποιήσει νέα γνώση, αλλά και να ανακαλύψει καινούργιες τακτικές και νέες κινήσεις πέρα από αυτές με τις οποίες κέρδισε τους προηγούμενους παγκόσμιους πρωταθλητές. Η επιτυχία του Zero βασίζεται στην εξής λογική. Κατά τη διάρκεια του παιχνιδιού, υπάρχει ένα μοναδικό νευρωνικό δίκτυο, το οποίο αρχικά δεν γνωρίζει τίποτα για το Go. Καταλήγει να παίζει με τον εαυτό του, συνδυάζοντας το νευρωνικό αυτό με έναν ισχυρό αλγόριθμο αναζήτησης Το νευρωνικό δίκτυο ρυθμίζεται και ενημερώνεται για να εκτιμά κινήσεις, καθώς και τον νικητή. Το ενημερωμένο αυτό δίκτυο συνδυάζεται ξανά με τον ίδιο αλγόριθμο για να προκύψει ένας ισχυρότερος Zero. Η διαδικασία αυτή γίνεται για πολλές επαναλήψεις, όπου σε κάθε επανάληψη οι επιδόσεις του συστήματος κατά ένα μικρό ποσοστό αυξάνονται.



Διάγραμμα 1: Ο AlphaGo Zero χρησιμοποιεί μόλις 4 TPUs, καθιστώντας τον μαζί με τον AlphaGo Master ως το πιο αποδοτικό σύστημα σε θέμα κατανάλωσης ισχύος

Μία βασική διαφοροποίηση από τον κλασσικό AlphaGo είναι πως ο Zero χρησιμοποιεί ως είσοδο μόνο τις άσπρες και μαύρες πέτρες του ταμπλό του Go. Επίσης, ο Zero χρησιμοποιεί ένα ενιαίο νευρωνικό δίκτυο που συνδυάζει την λογική των δύο policy και value networks που είχαν οι αρχικές υλοποιήσεις. Τέτοιες αλλαγές οδήγησαν τον Zero να εξελιχθεί σε θέμα αλγοριθμικής ισχυρότητας, όπως επίσης και σε καλύτερη διαχείριση

υπολογιστικής ισχύος. Το παρακάτω διάγραμμα απεικονίζει την κατανάλωση ενέργειας σε ποσοστό TDP, σε σύγκριση με προηγούμενες υλοποιήσεις του AlphaGo.

2.2 OpenAI Five (Dota 2)

Το Dota 2 (Valve Corporation 2013) είναι ένα MOBA παιχνίδι το οπόιο φημίζεται, μεταξύ άλλων παιχνιδιών που ανήκουν στην ίδια κατηγορία, ως ένα ιδιαίτερα απαιτητικό βιντεοπαιχνίδι, κυρίως για το μεγάλο πλήθος των κινήσεων που έχει ο παίχτης στην κατοχή του, τις διαφορετικές μετρήσεις που έχει να λάβει υπόψιν του και τις εναλλαγές στους στόχους που έχει κατά τη διάρκεια μίας αναμέτρησης. Οι παίχτες και τα διάφορα τμήματα του παιχνιδιού (units) έχουν ορατότητα μόνο σε περιοχές του χάρτη που βρίσκονται γύρω τους, το οποίο κάνει το περιβάλλον μερικώς – παρατηρήσιμο (partially - observable). Ο βαθύς και πολύπλοκος τρόπος παιχνιδιού του έχει ως αποτέλεσμα μία ανηφορική καμπύλη μάθησης (learning curve). Το Five της OpenAI καταφέρνει και αναιρεί αυτή την πολυπλοκότητα, καταφέρνωντας να κερδίσει σε αναμετρήσεις 5 πρακτόρων εναντίον 5 ανθρώπων τους παγκόσμιους πρωταθλητές OG, στα OpenAI Five Finals στις 13 Απριλίου του 2019.

Το Five αποτελείται από μία ομάδα πέντε νευρωνικών δικτύων που παρατηρούν το περιβάλλον παιχνιδιού ως μια λίστα από 20000 αριθμούς (είσοδος) που κωδικοποιεί το ορατό πεδίο του παιχνιδιού και ενεργούν επιλέγοντας κινήσεις από μία λίστα 8 αριθμών (έξοδος).

Η γενική ιδέα του συστήματος είναι ότι παίζει αυτοδίδακτα, ξεκινώντας με τυχαίες παραμέτρους, αξιοποιώντας μία έκδοση του αλγορίθμου PPO (Proximal Policy Optimization, OpenAI 2017). Κάθε νευρωνικό δίκτυο στην ομάδα του Five είναι ένα LSTM δίκτυο (Hochreiter, Schmidhuber 1997) που περιέχει ένα επίπεδο 1024 τμημάτων (units) που λαμβάνουν την κατάσταση του παιχνιδιού μέσω ενός Bot API (Valve Corporation) και εξάγουν κινήσεις όπου κάθε μία έχει σημασιολογική αξία. Η αξία αυτή για παράδειγμα μετριέται στον αριθμό των ticks* που χρειάζονται για να καθυστερήσει η δεδομένη κίνηση, ποια κίνηση θεωρείται ως η κατάλληλη, στις συντεταγμένες X και Y που βρίσκονται γύρω από το τμήμα στο οποίο θα εκτελεστεί η κίνηση αυτή κτλ. Οι κινήσεις υπολογίζονται από κάθε νευρωνικό δίκτυο ανεξάρτητα (κάθε πράκτορας αποφασίζει για

τις δικές του κινήσεις με βάση τον τρέχον στόχο της αναμέτρησης και με το αν κάποιος συμπαίκτης πράκτορας χρειάζεται βοήθεια). Μέσω κατάλληλης διαμόρφωσης ανταμοιβής (reward shaping), το Five ήταν ικανό να αντιληφθεί καταστάσεις οι οποίες θεωρούνται επικίνδυνες, όπως για παράδειγμα μια πτώση πυρομαχικών στην περιοχή που ήταν τοποθετημένος ο πράκτορας, λαμβάνωντας υπόψιν του μετρήσεις, όπως είναι η πτώση της υγείας του. Μέσω μίας εκπαίδευσης 80% παίζοντας ενάντια στον εαυτό του και 20% παίζοντας ενάντια σε προηγούμενες εκδόσεις του, το σύστημα μπόρεσε ύστερα από αρκετές μέρες να προσαρμοστεί σε προχωρημένες παιχτικές πρακτικές, όπως η ομαδική πίεση προς την αντίπαλη περιοχή (5-hero push) και η κλοπή κρίσιμων αντικειμένων (Bounty runes) από τον αντίπαλο. Σημαντικό ρόλο έχει και η έννοια της εξερεύνησης σε ένα περιβάλλον στρατηγικής, όπως είναι αυτό του Dota 2. Για να αναγκαστούν οι πράκτορες να εξερευνήσουν στρατηγικά το χώρο, μόνο κατά τη διάρκεια της εκπαίδευσης, ιδιότητες όπως η υγεία, η ταχύτητα και το αρχικό επίπεδο του πράκτορα έλαβαν τυχαίες τιμές.

1. μονάδα μέτρησης στην ανάπτυξη παιχνιδιών που εκφράζει την χρονική περίοδο που καταναλώνει μία κίνηση

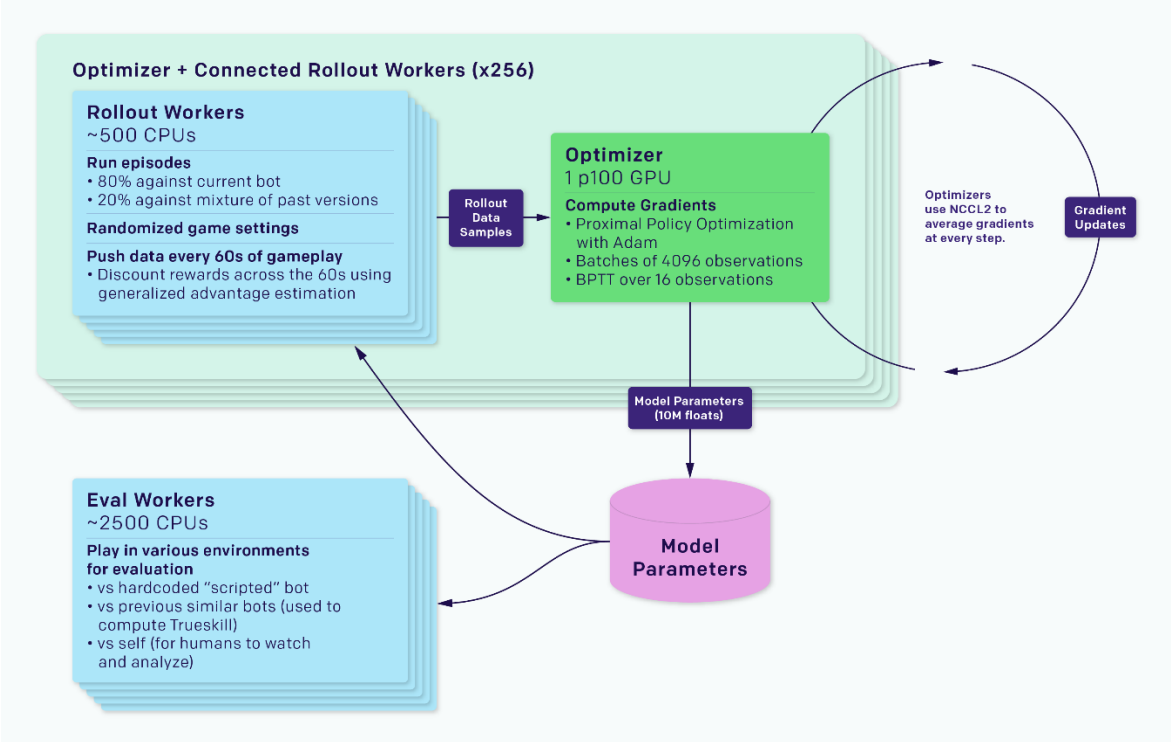
Έχοντας χάσει αρκετές 1 εναντίον 1 μονομαχίες με έναν πειραματικό παίχτη, οι τυχαίες τιμές της εκπαίδευσης αυξήθηκαν, επιτρέποντας στο σύστημα να αρχίζει να κερδίζει. Σχετικά με το θέμα της συνεργασίας, το OpenAI Five δεν παρέχει κάποιο άμεσο κανάλι επικοινωνίας μεταξύ των πρακτόρων – παικτών. Η μεταξύ τους ομαδικότητα ελέγχεται μέσω μίας υπερπαραμέτρου (hyperparameter). Η υπερπαράμετρος αυτή δέχεται τιμές από 0 εώς 1 και καθορίζει την βαρύτητα που πρέπει να ρίχνουν οι πράκτορες στην ατομική τους συνάρτηση ανταμοιβής έναντι της μέσης τιμής ομαδικής συνάρτησης ανταμοιβής. Η τιμή της υπερπαραμέτρου αλλάζει ανωδικά από 0 σε 1 κατά τη διάρκεια της εκπαίδευσης. Έπειτα από χρονοβόρα εκπαίδευση με χρήση ενός μεγάλου πλήθους από CPUs (128,000 πυρήνες στην Google Cloud Platform) και GPUs (256 P100 GPUs στην Google Cloud Platform), μπόρεσε να κληρονομήσει στρατηγικές υψηλού επιπέδου, αντίστοιχες με εκείνες που εφαρμόζουν οι καλύτεροι παίχτες του κόσμου.

Εικόνα 5: Στιγμιότυπο από μία μάχη 5 εναντίον 5 (άνθρωποι εναντίον OpenAI bots). Οι παίχτες με την πράσινη μπάρα ζωής είναι οι άνθρωποι, οι παίχτες με την κόκκινη μπάρα είναι τα OpenAI Five bots. Τα bots συνεργάζονται για να εξοντώσουν τον αντίπαλο

2.2.1 Σύστημα Rapid

Το OpenAI Five βασίζεται σε μία υλοποίηση ενός συστήματος RL εκπαίδευσης γενικού σκοπού που ονομάζεται Rapid. Χρησιμοποιήθηκε από την OpenAI Five για την επίλυση ζητημάτων, συμπεριλαμβανομένου και του ανταγωνιστικού αυτόνομου παιχνιδιού. Το σύστημα χωρίζεται σε workers, όπου ο καθένας περιέχει ένα αντίγραφο του Dota 2, έναν πράκτορα και κόμβους βελτιστοποίησης (optimizers), οι οποίοι με χρήση πολλαπλών GPUs υπολογίζουν σύγχρονα την μείωση της κλίσης (synchronous gradient descent). Οι workers συγχρονίζουν την εμπειρία τους μέσω του Redis* στους optimizers. Επίσης, κάθε πείραμα περιέχει και workers που αξιολογούν τον εκπαιδευόμενο πράκτορα σε σύγκριση με διαφορετικούς πράκτορες, καθώς και λογισμικό παρακολούθησης γραφημάτων, όπως είναι για παράδειγμα το Tensorboard. Κάθε GPU υπολογίζει, στο ποσοστό δέσμης που της αναλογεί, μία μείωση κλίσης και όλες οι συνολικές μειώσεις υπολογίζονται κατά μέσο όρο. Στην επόμενη σελίδα εμφανίζεται διαγραμματικά η αρχιτεκτονική του συστήματος Rapid.

1. Open-source, δομή αποθήκευσης, που χρησιμοποιείται ως βάση δεδομένων ή ως σύστημα caching



Σχήμα 1: Αριτεκτονική συστήματος Rapid

2.3 AlphaStar (StarCraft II)

Και στην περίπτωση του StraCraft II (Blizzard Entertainment 2010) έχουμε να κάνουμε με ένα RTS παιχνίδι υψηλής πολυπλοκότητας και ανταγωνισμού, που θεωρείται ένα από τα παιχνίδια με τις περισσότερες ώρες ενασχόλησης σε διαγωνισμούς esports και αποτελεί μία μεγάλη πρόκληση για τις ερευνητικές ομάδες τεχνητής νοημοσύνης. Παιχτικά δεν υπάρχει κάποια στρατηγική που να θεωρείται η καλύτερη. Για άλλη μια φορά, έχουμε να κάνουμε με ένα μερικώς - παρατηρήσιμο περιβάλλον που ο παίχτης καλείται να εξερευνήσει. Ο χώρος κινήσεων εξαρτάται από δύο παράγοντες. Υπάρχουν πολλά διαχειρίσιμα τμήματα που επιφέρουν πολλές συνδυαστικές ενέργειες. Επίσης, οι κινήσεις είναι ιεραρχικές και μπορούν να παραμετροποιηθούν ή να αυξηθούν με βάση την κατάσταση του παιχνιδιού.

Η DeepMind της Google παρουσίασε το 2019 την υλοποίηση του AlphaStar, της πρώτης AI που κέρδισε τον Grzegorz Komincz, έναν από τους καλύτερους παίχτες του StarCraft II σε μία ακολουθία από πειραματικούς αγώνες που πραγματοποιήθηκαν στις 19

Δεκεμβρίου του 2019, υπό συνθήκες αγώνων επαγγελματικής κατάταξης, με σκορ 5 – 0 υπέρ του AlphaStar.

2.3.1 Εκπαίδευση AlphaStar

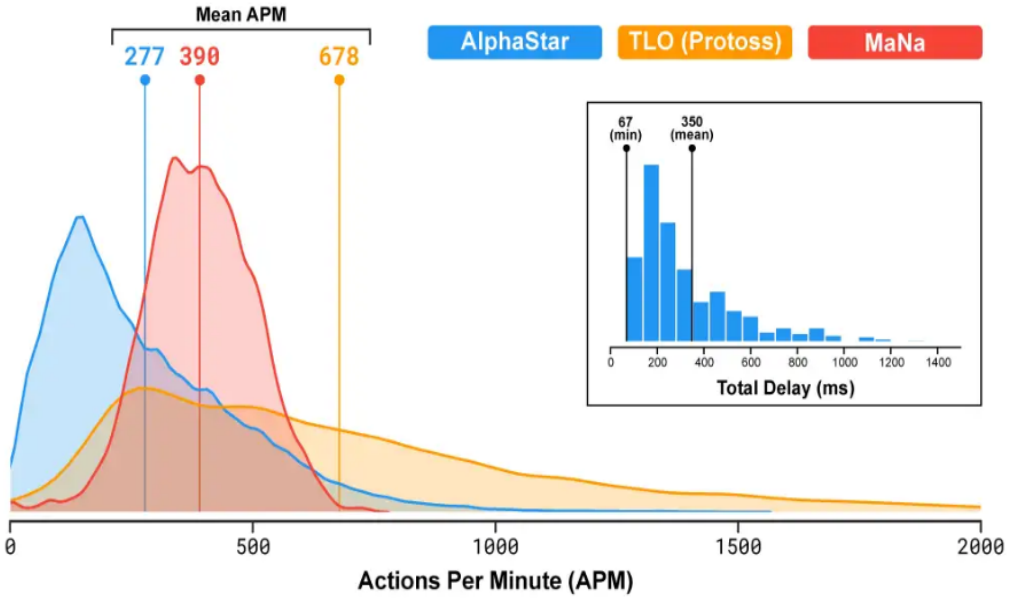
Η συγκεκριμένη τεχνητή νοημοσύνη, η οποία αξιοποιεί SL και RL τεχνικές, χρησιμοποιεί DNN που δέχεται ως είσοδο ανεπεξέργαστα δεδομένα παιχνιδιού (στο περιβάλλον του StarCraft II, τα δεδομένα αυτά μεταφράζονται ως μία λίστα από τα διαθέσιμα τμήματα και τις ιδιότητες τους) και παράγει ως έξοδο ένα σύνολο από εντολές που απαρτίζουν την κίνηση που εκτελείται σε κάθε χρονικό βήμα. Η αρχιτεκτονική εφαρμόζει μία ιδέα βασισμένη σε μηχανισμούς προσοχής που διανέμονται με χρήση RNNs (Recurrent Neural Networks) και CNNs (Convolutional Neural Networks), την οποία ιδέα η DeepMind την αποκαλεί Transformer. Ο κορμός του Transformer, σε συνδυασμό με έναν βαθύ LSTM πυρήνα, μία πολιτική αυτορυθμιζόμενης τακτικής (Google DeepMind 2017) συνοδευόμενη από ένα δίκτυο δεικτών (Vinyals, Fortunato, Jaitly) και μία εκτίμηση συγκεντρωτικής τιμής (University of Oxford). Ο Alphastar αξιοποιεί έναν αλγόριθμο πολυπρακτορικής εκπαίδευσης. Το δίκτυο αρχικά εκπαιδεύτηκε μέσω μάθησης με επίβλεψη (supervised learning), παρατηρώντας παιχνίδια που είχαν παίξει άνθρωποι και μπόρεσε να μιμηθεί βραχυπρόθεσμες και μακροπρόθεσμες στρατηγικές τους. Τα δεδομένα της SL εκπαίδευσης τροφοδότησαν στη συνέχεια ένα πολυπρακτορικό σύστημα ενισχυτικής μάθησης. Για τη διαδικασία αυτή, πραγματοποιήθηκαν για 14 μέρες συνεχείς αναμετρήσεις με ανταγωνιστές πράκτορες, σε έναν διαγωνισμό αποτελούμενο αποκλειστικά από AI υλοποιήσεις, όπου προστίθεντο δυναμικά νέοι διαγωνιζόμενοι, που αποτελούσαν διακλαδώσεις ήδη υπαρχόντων πρακτόρων. Κάθε πράκτορας, που ως στόχο είχε να νικήσει είτε έναν είτε ένα σύνολο από ανταγωνιστές, μάθαινε από τις αναμετρήσεις του, το οποίο βοήθησε στην εξερεύνηση ενός τεράστιου χώρου στρατηγικών σχετικών με τον τρόπο παιχνιδιού του StarCraft II, ενώ ταυτόχρονα σιγούρευε πως κάθε διαγωνιζόμενος λειτουργούσε αποτελεσματικά απέναντι σε δυνατότερες στρατηγικές και δεν ξεχνούσε πως να αντιμετωπίζει τις πιο αδύναμες παρελθοντικές. Οι σχετικά πιο πρόσφατοι διαγωνιζόμενοι εκτιμούσαν εξελιγμένες εκδόσεις παλαιότερων στρατηγικών, την ώρα που οι πιο έμπειροι δοκίμαζαν πρωτοπόρες στρατηγικές που βασίζονταν σε εντελώς νέες μετρήσεις, διατάξεις τμημάτων και διαχειρίσεις πόρων. Μια τέτοια προσέγγιση δεν

διαφέρει δραματικά από τον τρόπο με τον οποίο οι άνθρωποι ανακαλύπτουν νέες καλύτερες στρατηγικές και θεωρούν τις ήδη γνώριμες ως απαρχαιομένες. Μέσα από τις διάφορες αναμετρήσεις και με τη βοήθεια της ενισχυτικής μάθησης, τα βάρη των νευρωνικών δικτύων κάθε πράκτορα ανανεώνονταν ώστε να βελτιστοποιηθεί η επίτευξη του πρακτορικού στόχου. Η ανανέωση αυτή γίνεται με τη χρήση ενός RL αλγορίθμου δράστη – κριτή εκτός πολιτικής (off – policy actor - critic) σε συνδυασμό με επανάληψη εμπειρίας (experience replay), μάθηση αυτό – μίμησης και μεταφορά τακτικής σε νέο νευρωνικό δίκτυο (policy distillation).

Η εκπαίδευση του AlphaStar πραγματοποιήθηκε σε ένα κατανεμημένο περιβάλλον που χρησιμοποιούσε v3 TPUs της Google με υποστήριξη πολλαπλής μάθησης πρακτόρων σε παράλληλες εκτελέσεις του StarCraft II. Για κάθε πράκτορα χρησιμοποιήθηκαν 16 TPUs και κάθε πράκτορας έλαβε εμπειρία εώς και 200 ετών πραγματικού χρόνου του παιχνιδιού, σε διάστημα μόλις 14 ημερών.

2.3.2 Αντίληψη παιχνιδιού

Το σύστημα αναμετρήθηκε με δύο επαγγελματίες του StarCraft II. Κατά την διάρκεια των αγώνων, ο AlphaStar αλληλεπιδρούσε απευθείας με την διεπαφή της μηχανής του παιχνιδιού, το οποίο είχε ως αποτέλεσμα την άμεση παρατήρηση των δικών του τμημάτων παιχνιδιού αλλά και του αντίπαλου, χωρίς την χρήση περαιτέρω ενεργειών που προαπαιτεί ο ανθρώπινος χειρισμός την ώρα της αναμέτρησης (πχ. σωστή τοποθέτηση κάμερας). Συνέπεια αυτού ήταν ο πράκτορας να έχει ένα μέσο APM (actions per minute) περίπου 280 κινήσεων, την ώρα που οι άνθρωποι – αντίπαλοι του είχαν σημαντικά μεγαλύτερο μέσο όρο κινήσεων.



Διάγραμμα 2: Μέσος όρος κινήσεων ανά λεπτό. Μπορεί ο AlphaStar να σημείωσε τον μικρότερο μέσο όρο σε σύγκριση με τους αντιπάλους του, όμως αυτό είναι συνέπεια της άμεσης επαφής με το παιχνίδι

3 Τεχνολογικό υπόβαθρο

Στο συγκεκριμένο κεφάλαιο αναλύεται η προσέγγιση και ανάπτυξη αυτόνομων συστημάτων ενισχυτικής μάθησης σε βάθος, στο πλαίσιο των ηλεκτρονικών παιχνιδιών βολών πρώτου προσώπου (first-person shooters) και πιο συγκεκριμένα στο πλαίσιο του παιχνιδιού Doom (id Software 1993). Σε αντίθεση με τα περισσότερα παιχνίδια στα οποία έχουν εξεταστεί RL τεχνικές, τα οποία χαρακτηρίζονται από δισδιάστατα πλήρως – παρατηρήσιμα (fully - observable) περιβάλλοντα, στην περίπτωση του Doom έχουμε ένα τρισδιάστατο μερικώς – παρατηρήσιμο περιβάλλον. Οι υλοποιήσεις που παρουσιάζονται στην παρούσα πτυχιακή εργασία έχει αποδειχθεί πως μπορούν να ανταπεξέλθουν αποτελεσματικά σε αυτού του είδους τα περιβάλλοντα. Στις ενότητες που ακολουθούν περιγράφεται η διαδικασία επιλογής των τεχνολογιών στις οποίες βασίστηκε η ανάπτυξη των πρακτορικών συστημάτων, δίνονται εξηγήσεις για τους διάφορους μαθηματικούς ορισμούς που χρησιμοποιούνται από τέτοια συστήματα, ενώ γίνεται μία λεπτομερής ανάλυση του περιβάλλοντος του παιχνιδιού και της λειτουργικότητας στην οποία στηρίζονται τα συστήματα αυτά (αντίληψη καταστάσεων, μοντέλα εκτίμησης κινήσεων, διαμόρφωση ανταμοιβών κτλ.) για την αλληλεπίδραση τους με το περιβάλλον αυτό.

Εικόνα 6: Το παιχνίδι Doom (1993)

3.1 Ανάλυση τεχνολογιών

Για την φάση της υλοποίησης, έπρεπε να ληφθεί υπόψιν η χρήση μιας γλώσσας προγραμματισμού ικανής να αποτυπώσει με απλότητα, μέσω παρεχόμενων βιβλιοθηκών, την λογική των συστημάτων ενισχυτικής μάθησης, καθώς και η αξιοποίηση των κατάλληλων frameworks τα οποία ενσωματώνουν την γλώσσα αυτή για την εκμετάλευση της διεπαφής που επιτρέπει την επικοινωνία μεταξύ ενός περιβάλλοντος και ενός πρακτορικού συστήματος, όπως και για την δημιουργία αποτελεσματικών μοντέλων μηχανικής μάθησης γενικότερα.

3.1.1 Python

Εικόνα 7: Γλώσσα προγραμματισμού Python

Ο λόγος που επιλέχθηκε αυτή η γλώσσα προγραμματισμού έναντι άλλων γλωσσών ήταν η παροχή ενός εύρους βιβλιοθηκών διαχείρισης διανυσμάτων, επεξεργασίας εικόνων, αναπαράστασης γραφημάτων κ.α. Επίσης, η Python έχει ενσωματωμένες κάποιες δομές δεδομένων που χρησίμευσαν στην αναπαράσταση των δεδομένων που αφορούν τους τένσορες εισόδου των νευρωνικών δικτύων, όπως είναι για παράδειγμα τα λεξικά (dictionaries). Ο πίνακας που ακολουθεί αναφέρει ονομαστικά τις βιβλιοθήκες και την χρήση αυτών για την παρούσα εργασία.

Βιβλιοθήκες	Χρήση
numpy	Επεξεργασία πολυδιάστατων πινάκων
matplotlib	Δημιουργία στατιστικών διαγραμμάτων

skimage	Επεξεργασία εικόνων
collections	Παροχή δομών δεδομένων
random	Παραγωγή τυχαίων αριθμών
os	Διαχείριση πόρων λειτουργικού συστήματος
math	Υλοποιήσεις μαθηματικών συναρτήσεων
itertools	Συναρτήσεις επαναλήψεων
__future__	Ειδικές συναρτήσεις εκτύπωσης
tqdm	Δημιουργία μπάρας προόδου
time	Χρονικές συναρτήσεις
argparser	Ανάλυση παραμέτρων γραμμής εντολών

Πίνακας 1: Βιβλιοθήκες που αξιοποιήθηκαν στην γραφή των python scripts. Από τον συγκεκριμένο πίνακα λείπουν οι TensorFlow και Keras βιβλιοθήκες, καθώς αποτελούν ξεχωριστή υποενότητα η καθεμία

3.1.2 TensorFlow

Εικόνα 8: Πλατφόρμα μηχανικής μάθησης TensorFlow

Από τα αρχικά κιόλας στάδια της παρούσας πτυχιακής εργασίας, κρίθηκε καταλληλότερο framework το TensorFlow (Google Brain 2015), κυρίως επειδή καλύπτει ένα μεγάλο πλήθος υλοποιήσεων για νευρωνικά δίκτυα, συναρτήσεις ενεργοποίησης

(activation functions), αρχικοποιητές βαρών (weight initializers) κ.α., ενώ προσφέρει και APIs υψηλού επιπέδου που διευκολύνουν το χτίσιμο και την εκπαίδευση μοντέλων βαθιάς μάθησης, όπως είναι για παράδειγμα το Keras που αναλύεται παρακάτω. Όλα τα μοντέλα που δημιουργήθηκαν για τη μελέτη αυτή βασίζονται ως ένα βαθμό σε μεθόδους που παρέχουν οι βιβλιοθήκες του TensorFlow.

3.1.3 Keras

Εικόνα 9: Keras API

Παρόλο που ως βασική δομή της υλοποίησης των μοντέλων στηρίζονται στο TensorFlow, η κατασκευή των πιο πρόσφατων νευρωνικών δικτύων της μελέτης έγινε με τη χρήση του Keras (Francois Chollet 2015), ενός API υψηλού επιπέδου που τρέχει πάνω από το TensorFlow. Υποστηρίζει υλοποιήσεις CNN και RNN δικτύων και τρέχει αποτελεσματικά σε CPUs και GPUs.

3.1.4 ViZDoom

Για να μπορέσουν τα πρακτορικά συστήματα να έχουν μία επικοινωνία με το περιβάλλον του Doom και να αποκτήσουν πρόσβαση στις μεταβλητές του παιχνιδιού, προαπαιτείται η χρήση της πλατφόρμας ViZDoom. Είναι βασισμένη στο ZDoom, την πιο δημοφιλή πηγαία μεταφορά (source - port) του παιχνιδιού, που έχει ως αποτέλεσμα τη συμβατότητα με πολλά εργαλεία δημιουργίας προσωποποιημένων σεναρίων. Μέσω της πλατφόρμας αυτής, παρέχεται η οπτική πληροφορία (σε μορφή pixel) που χρειάζεται ο ενεργός πράκτορας για να πράξει πάνω στο περιβάλλον. Δίνει τη δυνατότητα εφαρμογής πρακτικών σε ένα πλήθος από διαφορετικά σενάρια, ταυτοποιεί αυτόματα τα αντικείμενα

που εμφανίζονται εντός ενός καρέ (frame) και επιτρέπει την δημιουργία εικόνας εκτός όθονης (off – screen rendering).

3.2 Περιβάλλον Doom

Απαραίτητη προυπόθεση για να ξεκινήσει η αλληλεπίδραση του πράκτορα με το περιβάλλον του είναι η αρχικοποίηση του παιχνιδιού (βλ. Παράρτημα Α). Για την αρχικοποίηση απαιτείται ένα αρχείο ρυθμίσεων (configuration file), το οποίο εξηγεί στην πλατφόρμα με τι παραμέτρους θα εκκινήσει το παιχνίδι για ένα δεδομένο σενάριο. Κάθε αρχείο ρυθμίσεων περιέχει το σχετικό μονοπάτι στο οποίο βρίσκεται το αρχείο σεναρίου (με κατάληξη .wad), την προτεινόμενη ανάλυση της οθόνης παιχνιδιού, περιγράφει τις τιμές των ανταμοιβών που δίνονται στον πράκτορα για τις συνέπειες των κινήσεων που αυτός εκτελεί (ανταμοιβή επιβίωσης, αρνητική ανταμοιβή θανάτου κτλ.), ενώ δίνονται επίσης σε μορφή πίνακα διάφορες μεταβλητές παιχνιδιού (αριθμός εξοντώσεων, υγεία πράκτορα, διαθέσιμες σφαίρες) και το σύνολο των διαθέσιμων κινήσεων (βλ. Παράρτημα Ε). Επίσης παρέχεται η επιλογή εμφάνισης ή απόκρυψης της οθόνης παιχνιδιού, η οποία βοηθάει στη διαδικασία εκπαίδευσης του πράκτορα, καθώς δεν επιθυμούμε να εμφανίζεται η οθόνη κατά τη διαδικασία αυτή. Ο λόγος είναι διότι η απεικόνιση της οθόνης επιβραδύνει σημαντικά την εκπαίδευση, μιας και το σύστημα καλείται να δημιουργεί καρέ την ώρα που υπολογίζονται οι μεταβλητές του νευρωνικού δικτύου.

Από τη στιγμή που η αρχικοποίηση του περιβάλλοντος έχει ολοκληρωθεί, μπορεί να ξεκινήσει η λήψη καταστάσεων παιχνιδιού. Το ViZDoom API παρέχει ένα σύνολο από μεθόδους, η χρήση των οποίων δίνει πρόσβαση στις τιμές των μεταβλητών που ορίζονται από το αρχείο ρυθμίσεων του δεδομένου περιβάλλοντος. Σε αυτές περιλαμβάνονται και μέθοδοι εκκίνησης νέου επεισοδίου παιχνιδιού (game episode), όπως και για την εφαρμογή κινήσεων πάνω στο περιβάλλον (DoomGame.set_action(list actions), DoomGame.advance_action(int tics, bool updateState)). Ως επεισόδιο παιχνιδιού ορίζεται ένα σύνολο από καταστάσεις, οι οποίες θέτουν το παιχνίδι σε εξέλιξη. Το πλήθος των καταστάσεων αυτών, ή ακόμα και το πλήθος των επεισοδίων μπορεί να προκαθοριστεί πριν από την διαδικασία εκπαίδευσης ή αξιολόγησης του πρακτορικού συστήματος. Το τελός ενός επεισοδίου σηματοδοτείται μόλις φτάσει το περιβάλλον σε τερματική κατάσταση. Τερματική κατάσταση σε ένα περιβάλλον ενισχυτικής μάθησης θεωρείται η κατάσταση

στην οποία έχει συμβεί κάποιο κρίσιμο για το παιχνίδι γεγονός, όπως για παράδειγμα η εξουδετέρωση του πράκτορα από τους εχθρούς του αλλά και η ξεπέραση του ορίου των συνολικών καταστάσεων παιχνιδιού.

Όσον αφορά τις ιδιότητες του περιβάλλοντος του παιχνιδιού Doom, πρόκειται για ένα μερικώς – παρατηρήσιμο και στοχαστικό περιβάλλον. Θεωρούμε το συγκεκριμένο περιβάλλον μερικώς – παρατηρήσιμο διότι ο RL πράκτορας μας δεν είναι σε θέση να αντιληφθεί πλήρως κάθε δεδομένη κατάσταση που λαμβάνει. Για παράδειγμα, σε ένα Doom σενάριο, ένας εχθρός μπορεί κάποια χρονική στιγμή να είναι τοποθετημένος πίσω από τη θέση του πράκτορα. Λόγω του ότι ο πράκτορας αντιλαμβάνεται ως κατάσταση το διάνυσμα των pixels της οθόνης για εκείνη τη στιγμή, είναι ανίκανος να αισθανθεί την ύπαρξη του εχθρού στο πίσω μέρος του. Για την εξαγωγή βέλτιστων αποφάσεων σε ένα μερικώς – παρατηρήσιμο περιβάλλον, εμφανίζεται η ανάγκη αποθήκευσης των πρόσφατων καταστάσεων στη μνήμη (σε επόμενη ενότητα εξηγείται αυτή η τεχνική). Η στοχαστικότητα του Doom περιβάλλοντος οφείλεται στο γεγονός ότι για μία κίνηση που εκτελεί το πρακτορικό σύστημα σε μία δεδομένη κατάσταση, είναι αβέβαιη η επόμενη κατάσταση στην οποία θα μεταβεί το περιβάλλον. Σε ένα στοχαστικό περιβάλλον, η εκτίμηση των κινήσεων που μεγιστοποιούν την επίδοση του RL συστήματος είναι βασισμένη σε μοντέλα πιθανοτήτων.

3.2.1 Κατάσταση παιχνιδιού

Ως κατάσταση παιχνιδιού s ορίζεται η αναπαράσταση του περιβάλλοντος ενός παιχνιδιού για μία χρονική στιγμή t. Η αναπαράσταση αυτή μπορεί να περιγράφει τη θέση των αντικειμένων ή των πρακτόρων που ενεργούν σε αυτό το περιβάλλον, την ταχύτητα με την οποία κινούνται στο χώρο, όπως και άλλες χρήσιμες πληροφορίες τις οποίες καλείται να αντιληφθεί ένα σύστημα ενισχυτικής μάθησης. Στην περίπτωση του Doom, τα καρέ της οθόνης παιχνιδιού σε κάθε χρονική στιγμή αντιπροσωπεύουν τις καταστάσεις.

Μία αναπαράσταση κατάστασης ***x*** περιέχει την ιδιότητα Markov (Andrey Markov 1906), το οποίο σημαίνει πως η εμφάνιση αυτής της τρέχουσας κατάστασης οφείλεται στις καταστάσεις που προηγήθηκαν εκείνης και έχουν εξάρτηση μαζί της. Για μία ακολουθία καταστάσεων $\{x_1, x_2, x_{3,\dots}, x_t\}$ η ιδιότητα Markov μπορεί να αναπαρασταθεί ως μία πιθανότητα της ακόλουθης μορφής:

$$p\{x_t \mid x_{t-1,\ldots,} x_{t-2,} x_1\} = p\{x_t \mid x_{t-1}\}$$

Η στοχαστική διαδικασία των καταστάσεων του παιχνιδιού Doom που πληρεί την παραπάνω ιδιότητα θεωρείται μία *Διαδικασία Μαρκοβιανών Αποφάσεων* (Markov Decision Process).

3.2.2 Διαδικασία Μαρκοβιανών Αποφάσεων

Πέρα από τα δύο βασικά στοιχεία, δηλαδή το πρακτορικό σύστημα και το περιβάλλον λειτουργίας του, μία MDP διαδικασία αποτελείται από μερικές έννοιες σχετικές με σήματα που σχηματίζονται από το ίδιο το περιβάλλον. Τα σήματα αυτά στο πλαίσιο των περιβαλλόντων που εφαρμόζονται οι πρακτικές συστημάτων της ενισχυτικής μάθησης ορίζουν την πλειάδα <s, a, r, s'>. Ακολουθούν οι αντιστοιχίες των συμβόλων:

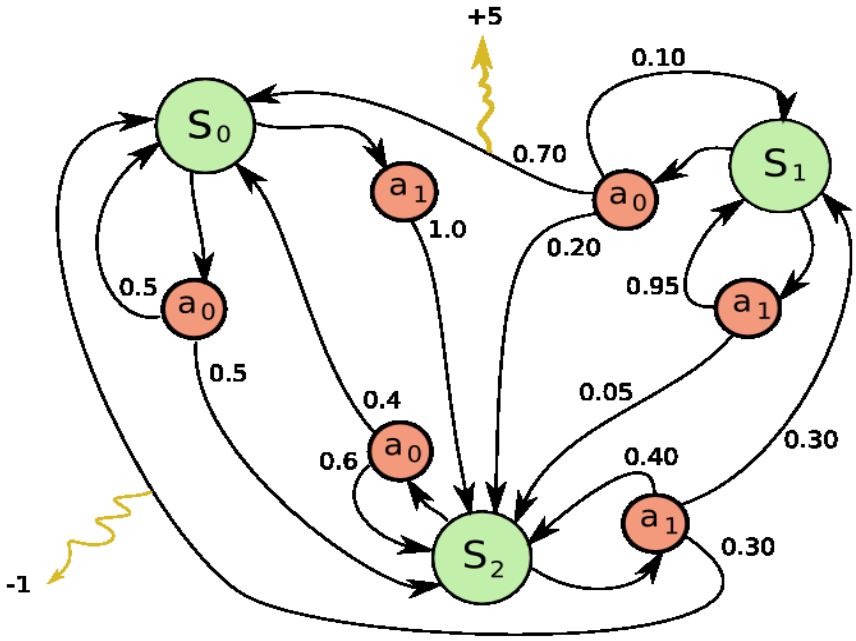
1. Κατάσταση
2. Ενέργειες
3. Ανταμοιβές
4. Επόμενη κατάσταση

Η έννοια της κατάστασης αναλύθηκε στην προηγούμενη υποενότητα. Τις ενέργειες μίας MDP διαδικασίας αποτελούν όλες οι διαθέσιμες κινήσεις που μπορεί να εκτελέσει ο πράκτορας και έχουν εξάρτηση στην τρέχουσα κατάσταση (ορισμένες κινήσεις ενδέχεται να μην είναι έγκυρες για μία δεδομένη κατάσταση). Το στοιχείο της ανταμοιβής, ή αλλιώς της συνάρτησης ανταμοιβής, εξηγεί στο πρακτορικό σύστημα πόσο θετική ή αρνητική επίπτωση είχε η εκτελούμενη ενέργεια για τον κύριο στόχο του. Στόχος του συστήματος είναι η μεγιστοποίηση αυτής της συνάρτησης, η οποία ισούται με το άθροισμα όλων των ανταμοιβών σε κάθε χρονική στιγμή. Εκτελώντας μία ενέργεια, το περιβάλλον οδηγείται στον υπολογισμό μίας νέας κατάστασης, γνωστής και ως επόμενης κατάστασης. Οι επόμενες καταστάσεις μοντελοποιούνται ως πιθανότητες μετάβασης από μία κατάσταση s_1 σε μία νέα κατάσταση s_2, μέσω μίας ενέργειας a. Η συνάρτηση πιθανότητας μετάβασης ορίζεται ως εξής:

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

Πρόκειται για μία πιθανοτική κατανομή όλων των πιθανών επόμενων καταστάσεων δοθέντος της τωρινής κατάστασης. Αναλύοντας το περιβάλλον του Doom, καταλήγουμε

στο συμπέρασμα πως ο χώρος καταστάσεων του είναι υπερβολικά μεγάλος, οπότε η απαράθμιση όλων των πιθανών μεταβάσεων δεν αποτελεί καλή πρακτική. Η ικανότητα μάθησης των ιδανικών περιβαλλοντικών μεταβάσεων έγκειται στην ικανότητα του συστήματος να εκτιμήσει κατάλληλες ενέργειες. Μία επόμενη κατάσταση μπορεί να χαρακτηριστεί επίσης ως τερματική (terminal state), ανάλογα με τα γεγονότα που λαμβάνουν μέρος στο παιχνίδι, όπως για παράδειγμα η εξουδετέρωση του πράκτορα.



Σχήμα 2: Διαδικασία Μαρκοβιανών Αποφάσεων. Οι ακμές ορίζουν την πιθσνότητα μετάβασης από μία κατάσταση σε μία άλλη μέσω μίας ενέργειας

3.2.3 Διαδικασία Μαρκοβιανών Ανταμοιβών

Μία Διαδικασία Μαρκοβιανών Αποφάσεων (Markov Reward Process) ορίζεται ως η πλειάδα <S, P, R, γ>, όπου S είναι ο χώρος καταστάσεων του περιβάλλοντος, P είναι η συνάρτηση πιθανότητας μετάβασης καταστάσεων, R είναι η συνάρτηση ανταμοιβής και γ είναι ο παράγοντας μείωσης (discount factor) και ισχύει $\gamma \in [0, 1]$. Πιο συγκεκριμένα, η συνάρτηση ανταμοιβής αναφέρεται στην τιμή της άμεσης ανταμοιβής που αναμένεται από μία δεδομένη κατάσταση s και περιγράφεται από την ακόλουθη ισότητα:

$$Rs = \mathbb{E}[r_{t+1} \mid \boldsymbol{s}_t = S]$$

Ο παράγοντας μείωσης εξηγεί στο RL σύστημα πόση βαρύτητα πρέπει να δώσει στις ανταμοιβές που λαμβάνονται σε μελλοντικές καταστάσεις. Όσο πιο κοντά στο 0 βρίσκεται το γ, τόσο λιγότερο λαμβάνονται υπόψη οι μελλοντικές ανταμοιβές, ενώ το αντίθετο ισχύει για περιπτώσεις που το γ πλησιάζει το 1. Για τις τεχνικές που εφαρμόστηκαν στην παρούσα μελέτη και για το Doom περιβάλλον, μετά από πειραματισμούς κρίθηκε καταλληλότερη τιμή ο δεκαδικός αριθμός 0.99, επειδή στην αρχή το σύστημα είναι αναμενόμενο να συμπεριφέρεται απρόβλεπτα εκτελώντας τυχαίες κινήσεις, αξιοποιώντας την έννοια της εξερεύνησης (exploration). Άρα, η αξία των ανταμοιβών σε αρχικές καταστάσεις δεν λαμβάνεται σοβαρά. Εφαρμόζοντας τον παράγοντα μείωσης στη συνάρτηση ανταμοιβής με τη μορφή γινομένου, προκύπτει η επιστροφή του συνολικού αθροίσματος μειωμένων μελλοντικών ανταμοιβών G_t, όπου t είναι μία δεδομένη χρονική στιγμή. Πιο αναλυτικά ισχύει ότι:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots + \gamma^k R_{t+n}$$

Ο στόχος του συστήματος είναι να μεγιστοποιήσει αυτό το άθροισμα. Συνήθως, στα σενάρια του Doom, μεγαλύτερες τιμές ανταμοιβής επιτυγχάνονται με την εξόντωση αντιπάλων ή την επιβίωση σε ένα επικίνδυνο περιβάλλον. Ακολουθούν κάποιες ενδεικτικές τιμές ανταμοιβών για ορισμένες περιπτώσεις που εμφανίζονται σε διάφορα Doom σενάρια.

	Basic	Defend The Center	Health Gathering
Living reward	-1	-	+1
Death penalty	-	+1	+100
Kill reward	+1	+1	-

Πίνακας 2: Οι γραμμές περιγράφουν τις διαφορετικές ανταμοιβές/ποινές, οι στήλες περιγράφουν διαφορετικά Doom σενάρια. Η ποινή (penalty) προσμετράται αρνητικά στην ανταμοιβή μιας κατάστασης

Συνοψίζοντας, μπορούμε να περιγράψουμε ένα οποιοδήποτε περιβάλλον ενισχυτικής μάθησης, όπως είναι και το Doom περιβάλλον, ως μία MDP διαδικασία. Προαπαιτείται βέβαια η ύπαρξη ενός πράκτορα, έτοιμου να επιλύσει μία τέτοιου είδους διαδικασία. Οι τεχνικές λεπτομέρειες για την λειτουργία του πράκτορα περιγράφονται στην επόμενη ενότητα.

3.3 Πράκτορες ενισχυτικής μάθησης

Έχοντας μια MDP διαδικασία όπως αυτή που προαναφέρθηκε, ο σκοπός είναι να βρεθούν οι κατάλληλες ενέργειες οι οποίες βελτιστοποιούν το άθροισμα μειωμένων μελλοντικών ανταμοιβών G_t. Αυτός είναι και ο λόγος ύπαρξης των πρακτόρων ενισχυτικής μάθησης, οι οποίοι υλοποιούνται για να πετυχαίνουν αυτό το σκοπό. Η φάση επιλογής μίας ενέργειας για μία κατάσταση ονομάζεται τακτική (policy). Για την εύρεση της καλύτερης τακτικής σε διεργασίες που σχετίζονται με το Doom και αφορούν τη συγκεκριμένη πτυχιακή μελέτη, εφαρμόστηκε ο αλγόριθμος Q-Learning (Watkins, 1989). Για μία δεδομένη κατάσταση και μία ενέργεια που εκτελείται σε αυτή την κατάσταση, ο αλγόριθμος προσπαθεί να εκτιμήσει μία τιμή επιστροφής, γνωστή και ως τιμή Q (Q-value). Ουσιαστικά, η τιμή αυτή περιγράφει την ποιότητα που έχει η εκτελεσμένη ενέργεια για την τρέχουσα κατάσταση και αναπαριστάται ύπο την μορφή πίνακα |S * A| διαστάσεων (γνωστού και ως πίνακα Q), όπου S είναι ο χώρος καταστάσεων και A είναι το σύνολο των διαθέσιμων προς εκτέλεση ενεργειών. Αρχικά, ο πίνακας Q αρχικοποιείται με μηδενικές τιμές. Κατά την διάρκεια της εκπαίδευσης, ο πίνακας ανανεώνεται επαναληπτικά για κάθε ζεύγος κατάστασης – ενέργειας που επισκέπτεται ο πράκτορας. Εφόσον η εύρεση όλων των πιθανοτήτων μεταβάσεων για υπερβολικά μεγάλους χώρους καταστάσεων θεωρείται άσκοπη διαδικασία, ο αλγόριθμος Q-Learning κάνει μία εκτίμηση των Q-τιμών και προσπαθεί σταδιακά να τις βελτιώνει. Για την ενημέρωση των τιμών εφαρμόζεται ο ακόλουθος κανόνας ενημέρωσης:

$$Q(s,a) := r(s',a) + \gamma \max Q(s',a)$$

s είναι η τρέχουσα κατάσταση, a είναι η ενέργεια που εκτελείται στην κατάσταση αυτή, r είναι η ανταμοιβή που λαμβάνεται κατά την επόμενη κατάσταση, γ είναι ο παράγοντας μείωσης που πολλαπλασιάζεται με την μέγιστη τιμή Q από όλες τις

καταστάσεις στις οποίες οδηγούμαστε μέσω της κίνησης a. Ο συγκεκριμένος κανόνας βασίζεται στην εξίσωση Bellman (Richard Bellman 1957) και εισάγει την έννοια του Δυναμικού Προγραμματισμού για την βελτιστοποίηση προβλημάτων διακριτού χρόνου με τη χρήση αναδρομικών συναρτήσεων. Η έννοια της αναδρομής εμφανίζεται και στην παραπάνω ισότητα στον υπολογισμό της μέγιστης τιμής της συνάρτησης *Q* για τις επόμενες καταστάσεις. Στην περίπτωση που ο πράκτορας φτάσει σε τερματική κατάσταση, μιας και δεν υπάρχουν επόμενες καταστάσεις, ο παραπάνω κανόνας καταλήγει να έχει την εξής μορφή:

$$Q(s,a) := r(s',a)$$

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Εικόνα 10: Παράδειγμα πίνακα Q

Χρειάζεται να γίνει αναφορά στον τρόπο με τον οποίο το σύστημα επιλέγει τις προς εκτέλεση ενέργειες για κάθε δεδομένη χρονική στιγμή, από την αρχή της εκπαίδευσης εώς και την φάση της αξιολόγησης. Γενικώς, στις Q-Learning τεχνικές προτιμάται η πολιτική Epsilon – Greedy. Η συγκεκριμένη πολιτική βασίζεται σε δύο διακλαδώσεις που ενσωματώνουν στο πλαίσιο της εκπαίδευσης την εναλλαγή εξερεύνησης – εκμετάλλευσης (exploration - exploitation). Δεδομένης μίας μεταβλητής e, όπου ισχύει $0 \leq e \leq 1$ και ένος τυχαίου σπόρου rand, οι διακλαδώσεις αυτές ορίζονται ως εξής:

1. Epsilon: αν rand ≤ e, επέλεξε μία τυχαία ενέργεια μέσα από το σύνολο των διαθέσιμων ενεργειών A (exploration)

2. Greedy: αν rand > e, επέλεξε την ενέργεια η οποία έχει τη μεγαλύτερη τιμή Q για τη δεδομένη κατάσταση (exploitation)

Βέβαια, χρειάζεται και ένας μηχανισμός σταδιακής μείωσης της μεταβλητής e. Αυτό είναι απαραίτητο επειδή, υποθετικά η τιμή της e θα είναι μεγαλύτερη από την τιμή του σπόρου και έτσι ο πράκτορας στις αρχικές καταστάσεις κυρίως θα εξερευνεί το περιβάλλον εκτελώντας τυχαίες ενέργειες. Από ένα σημείο και έπειτα, έχοντας εξερευνήσει ένα πλήθος καταστάσεων, είναι επιθυμητό ο πράκτορας να αρχίσει να εκμεταλλεύεται τη γνώση που έχει αποκτήσει ανατρέχοντας τις τιμές του πίνακα Q. Για τις Doom εφαρμογές, η αρχική τιμή της e ορίζεται ως 1 και στην πορεία του χρόνου και εφόσον έχει αποθηκευτεί η τρέχουσα πλειάδα <s, a, r ,s', d>, όπου d μία ιδιότητα που περιγράφει αν η τρέχουσα κατάσταση s είναι τερματική, ελλατώνεται μέχρι να πλησιάσει μία τιμή κοντά 0. Έχοντας καταλήξει σε αυτή την μικρή τελική τιμή, θεωρητικά ο πράκτορας βρίσκεται αποκλειστικά σε μία φάση εκμετάλλευσης της γνώσης του και δεν βασίζεται σε πειραματισμούς κινήσεων. Τεχνικές λεπτομέρειες υπό μοφρή κώδικα εμφανίζονται στο Παράρτημα ΣΤ.

Σχετικά με την αποθήκευση της περιβαλλοντικής γνώσης, η οποία περιγράφεται από την παραπάνω πλείαδα για μία δεδομένη χρονική στιγμή, υλοποιείται μία δομή μνήμης επανάληψης (replay memory). Η συγκεκριμένη μνήμη είναι σταθερού μεγέθους και αποτελείται από όλες τις πρόσφατες μεταβάσεις <s, a, r ,s', d>, τις οποίες αντιλήφθηκε ο πράκτορας. Για τις πρακτορικές υλοποιήσεις της παρούσας μελέτης, το προτεινόμενο μέγιστο μήκος μνήμης ήταν 50.000 μεταβάσεις, καθώς μεγαλύτερα ποσά προκαλούσαν υπερχείλιση για το υλικό στο οποίο πραγματοποιήθηκαν οι δοκιμές. Ακολουθούνται ορισμένα βήματα για τα στάδια της αποθήκευσης μεταβάσεων στην μνήμη και της δειγματοληψίας μεταβάσεων από αυτήν. Τα βήματα αυτά έχουν ως εξής:

1. Ο πράκτορας συγκεντρώνει μία πλειάδα της μορφής <τωρινή κατάσταση, ενέργεια, ανταμοιβή, επόμενη κατάσταση, τερματική ιδιότητα> που αντιπροσωπεύει την τρέχουσα περιβαλλοντική μετάβαση και την αποθηκεύει στη μνήμη ως εγγραφή. Αν οι συνολικές εγγραφές μεταβάσεων ξεπεράσουν το μέγεθος της μνήμης επανάληψης, αφαιρούνται οι παλαιότερες εγγραφές που είναι ήδη αποθηκευμένες.

2. Έπειτα εφαρμόζει τυχαία δειγματοληψία από τη μνήμη επανάληψης, όπου το μέγεθος του δείγματος είναι ίσο με μία δέσμη μνήμης (προτιμούνται μεγέθη δέσμης 32 ή 64, ανάλογα με την υλοποίηση του πρακτορικού συστήματος και τους περιορισμούς του υλικού). Η τεχνική αυτή είναι γνωστή και ως επανάληψη εμπειρίας (experience replay).
3. Με βάση το δείγμα μνήμης που λήφθηκε, γίνεται ενημέρωση του πίνακα Q.

Ο λόγος που οι εγγραφές μεταβάσεων δεν επιλέγονται σειριακά και προτιμάται τυχαία δειγματοληψία, είναι διότι στα στατιστικά προβλήματα και τα προβλήματα βελτιστοποίησης όπως είναι αυτά των RL συστημάτων είναι καταλλήτερο τα δεδομένα να *κατανέμονται ανεξάρτητα.* Δεν είναι επιθυμητό τα δεδομένα των μεταβάσεων να σχετίζονται μεταξύ τους, όπως συμβαίνει δηλαδή στη σειριακή λήψη. Με τη λογική της ανεξάρτητης κατανομής αποφεύγονται οι ταλαντεύσεις στο RL μοντέλο, οι οποίες προκύπτουν από την συσχέτιση των δεδομένων. Η ιδέα της ανάκτησης εμπερίας από μία τέτοια δομή μνήμης για τον σκοπό της εκπαίδευσης, είναι που ενσωματώνει SL λογική στον τομέα της ενισχυτικής μάθησης, μιας και το σύστημα πλέον έχει μία βάση γνώσης ώστε να μπορεί να προβλέψει Q τιμές που τείνουν να πλησιάζουν πραγματικές Q τιμές. Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Z.

Σχετικά με την ενημέρωση των τιμών του πίνακα Q, απαιτείται ο υπολογισμός της συνάρτησης σφάλματος (loss function). Ο ρόλος της συνάρτησης αυτής είναι να εξηγεί πόσο απέχει η προβλεπόμενη τρέχουσα τιμή Q που εκτιμά ο αλγόριθμος από την πραγματική τιμή targetQ για την οποία ισχύει:

$$\text{Yi} = \begin{cases} Rt \ \alpha\nu \ s \ = \ \tau\varepsilon\rho\mu\alpha\tau\iota\kappa\acute{\eta} \\ Rt + 1 \ + \ \gamma \ maxQ(st + 1 \ \alpha') \ \alpha\nu \ s \ \neq \ \tau\varepsilon\rho\mu\alpha\tau\iota\kappa\acute{\eta} \end{cases}$$

Η συνάρτηση σφάλματος $L(\theta_i)$ υπολογίζεται από το ελάχιστο τετραγωνικό σφάλμα (mean squared error) μεταξύ της προβλεπόμενης τιμής Q και της πραγματικής τιμής target ως εξής:

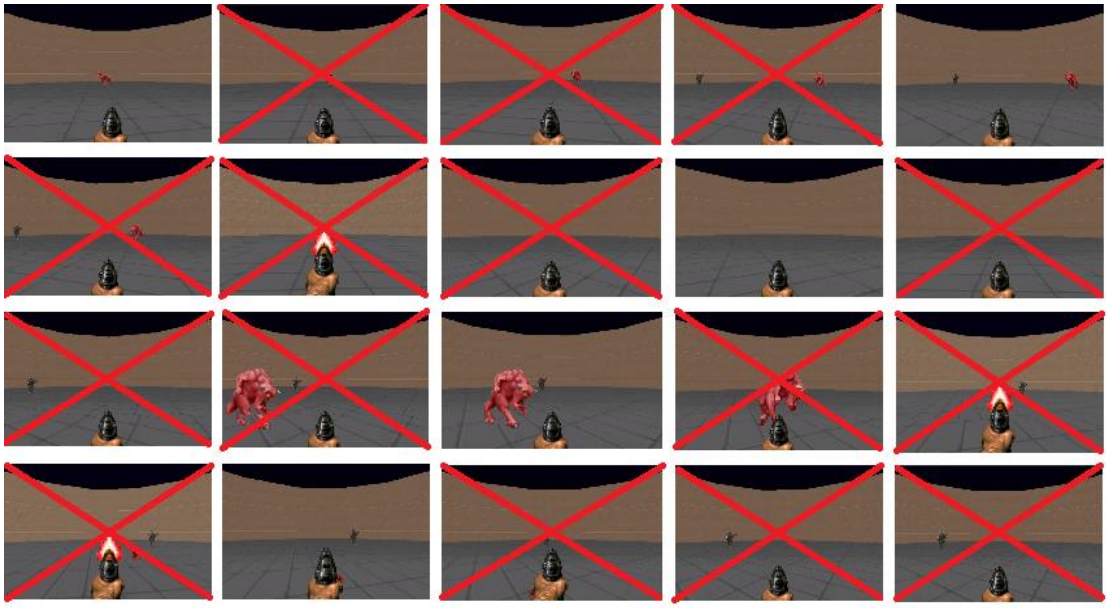
$$L(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot)}[(y_i - Q(s,a;\theta_i))^2]$$

όπου $E_{s,\,\alpha}$ είναι μία εκτίμηση για την τιμή της διαφοράς Q - targetQ της τρέχουσας κατάστασης και της ενέργειας που εκτελείται σε αυτή, η οποία ενέργεια ακολουθεί μία

πιθανοτική κατανομή. Ως θ ορίζουμε μία υπερπαράμετρο η οποία χρησιμεύει στην βαθμωτή πτώση (gradient descent) της τιμής που επιστρέφει η συνάρτηση σφάλματος και συμβολίζει τον ρυθμό μάθησης (learning rate).

3.3.1 Βαθιά Q Δίκτυα

Λόγω του μεγάλου γινομένου *καταστάσεων* επί *ενεργειών* που εμφανίζεται στο Doom περιβάλλον, γίνεται απαραίτητη η συγχώνευση της έννοιας της βαθιάς μάθησης (deep learning) και της Q-Learning τεχνικής, εισάγωντας την ιδέα των βαθέων νευρωνικών δικτύων (deep neural networks Mnih et al, 2015). Το αποτέλεσμα του συνδυασμού αυτού αποτελεί η ύπαρξη των DQNs (Deep Q Networks). Η χρησιμότητα τους δεν διαφέρει σημαντικά από εκείνη που έχουν τα CNNs, δηλαδή, στην ανάλυση εικόνων που αντιστοιχούν σε μεγάλα διανύσματα RGB pixels. Στο Doom περιβάλλον, κάθε δεδομένη κατάσταση s αναπαριστάται ως μία εικόνα (καρέ) ή ως ένα στιγμιότυπο υψηλών διαστάσεων του παιχνιδιού. Έχοντας όμως ως είσοδο ένα μοναδικό καρέ, το νευρωνικό δίκτυο δεν είναι ικανό να αντιληφθεί την κίνηση στο περιβάλλον του. Έτσι, γίνεται απαραίτητη η υλοποίηση της τεχνικής στοίβας καρέ (frame stack, DeepMind 2013). Ουσιαστικά, στοιβάζονται 4 μη-διαδοχικά καρέ, στα οποία έχει προηγηθεί κάποια προεπεξεργασία για κάθε μεμονωμένο στιγμιότυπο. Για την ακρίβεια, προσίθεται στην στοίβα ένα καρέ για κάθε 4 συνεχόμενα καρέ (frame skipping), κάνοντας πιο αισθητή την έννοια της κίνησης στον πράκτορα. Μόλις η στοίβα ξεπεράσει το όριο των 4 καρέ, αφαιρείται το παλαιότερο που έχει προστεθεί.



Εικόνα 11: Ενδεικτική αναπαράσταση του τρόπου προσθήκης των καρέ στη στοίβα. Σε τέσσερα συνεχόμενα στιγμιότυπα οθόνης δεν διακρίνεται εύκολα η κίνηση.

Με εφαρμογή αυτής της μεθόδου, η τελική αναπαράσταση μίας ατομικής κατάστασης περιβάλλοντος περιγράφεται από τέσσερα στοιβαγμένα καρέ. Χρειάζεται όπως προαναφέρθηκε μία προεπεξεργασία πάνω σε κάθε ένα από αυτά, πριν προστεθεί στην δομή. Η ανεπεξέργαστη είσοδος που καλείται να τροφοδοτήσει το DQN είναι ίση με το γινόμενο που προκύπτει από τα παρακάτω ποσά:

Πλάτος εικόνας	**640**
Ύψος εικόνας	480
Αριθμός καρέ	4

Πίνακας 3: Κάθε μεμονωμένο ανεπεξέργαστο καρέ έχει ανάλυση 640x480

Πέρα από το γεγονός ότι αυτό το μέγεθος φέρνει ως αποτέλεσμα πολύπλοκους υπολογισμούς για το DQN, εμπεριέχει αρκετή ασήμαντη πληροφορία. Για παράδειγμα, οι χρωματικές τιμές RGB που περιγράφουν ένα στιγμιότυπο δεν προσφέρουν κάποια αξία στο δίκτυο. Πιο αναλυτικά για κάθε καρέ:

1. Γίνεται αλλαγή του μεγέθους ώστε να σμικρυνθεί το διάνυσμα της εικόνας. Ανάλογα με το περιβάλλον, επιλέγεται η αποκοπή των αρχικών pixels του άξονα y (αντιστοιχούν στην περιοχή που περιγράφει το ταβάνι του παιχνιδιού)
2. Μετατρέπεται η μορφή οθόνης από RGB σε grayscale
3. Το τρέχον καρέ προστίθεται στην στοίβα. Μόνο στην περίπτωση της πρώτης εικόνας που παράγεται, γίνεται προσθήκη αυτής στην στοίβα τέσσερις φορές για να συμπληρωθούν τα καρέ που απαιτούνται για την αρχική κατάσταση

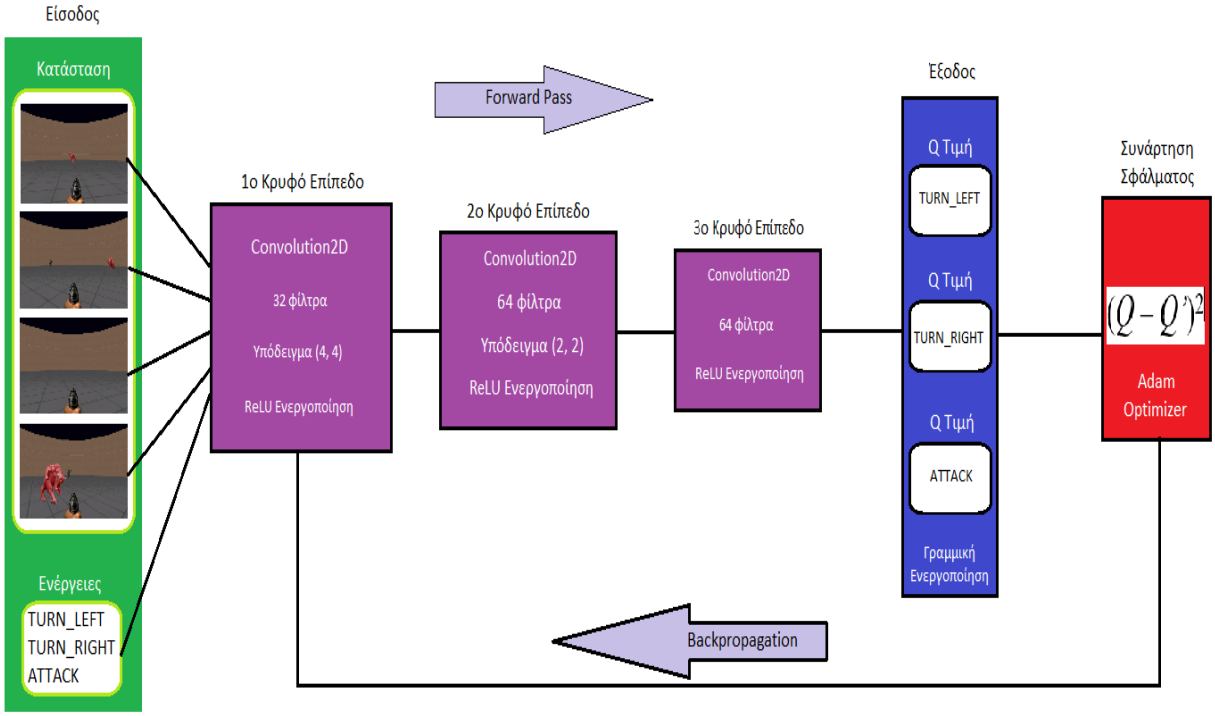
Λεπτομέρειες σχετικά με την υλοποίηση της παραπάνω διαδικασίας σε python βρίσκονται στο Παράρτημα Β. Στον παρακάτω πίνακα φαίνεται η απλοποιημένη είσοδος που καταλήγει στο νευρωνικό δίκτυο, έπειτα από προεπεξεργασία με τις προτεινόμενες τιμές των διαστάσεων του κάθε καρε.

Πλάτος εικόνας	**64**
Ύψος εικόνας	64
Αριθμός καρέ	4

Πίνακας 4: Η επεξεργασμένη κατάσταση που αποτελεί την είσοδο του δικτύου εκφράζεται από το γινόμενο 64x64x4

Με την ολοκλήρωση της παραπάνω διαδικασίας, η κατάσταση – είσοδος λαμβάνεται από το DQN ως ένας τένσορας 64x64x4 διαστάσεων. Αρχικά, οι τιμές των βαρών του δικτύου που συμβάλλουν στους υπολογισμούς εισόδου αρχικοποιούνται τυχαία, με τη χρήση κατάλληλων μεθόδων που παρέχονται από το Keras. Τα βάρη που υπάρχουν στις διασυνδέσεις κάθε νευρώνα εκφράζουν την ένταση της μεταξύ τους σύνδεσης και αποφασίζουν την επιρροή της εισόδου στην τελική εκτίμηση του δικτύου. Με την εφαρμογή κατάλληλου μεγέθους φίλτρων, σωστού υποδείγματος και με τη χρήση της σωστής συνάρτησης ενεργοποίησης (activation function), η είσοδος περνάει προς τα επόμενα κρυφά επίπεδα του δικτύου (forward pass). Τα ενδιάμεσα αυτά επίπεδα θεωρούνται πλήρως διασυνδεδεμένα, που σημαίνει ότι κάθε νευρώνας έχει σύνδεση προς όλους τους νευρώνες του επόμενου επιπέδου. Στο τελικό επίπεδο του δικτύου (output layer) παράγεται ο τένσορας εξόδου που αποτελεί πιθανοτική κατανομή των τιμών Q που

αντιστοιχούν σε κάθε διαθέσιμη ενέργεια. Πρόκειται για μία εκτίμηση που εξάγει το DQN για αυτές τιμές, με την προϋπόθεση πως στην είσοδο έχει περαστεί και το σύνολο των διαθέσιμων ενεργειών. Ύστερα, με βάση την εξίσωση Bellman υπολογίζεται η συνάρτηση σφάλματος ως το ελάχιστο τετραγωνικό σφάλμα μεταξύ των εκτιμόμενων τιμών Q και των πραγματικών τιμών Q που προέρχονται από δειγματοληψία μνήμης και αντιπροσωπεύουν επόμενες της τρέχουσας καταστάσεις. Τέλος, η τιμή της συνάρτησης σφάλματος μεταφέρεται οπισθοδρομικά στα προηγούμενα επίπεδα για να γίνει ενημέρωση των βαρών του δικτύου (backpropagation).



Σχήμα 3: Αναπαράσταση ενός νευρωνικού δικτύου DQN. Η είσοδος (κατάσταση παιχνιδιού, διαθέσιμες ενέργειες) κατευθύνεται προς τα κρυφά επίπεδα όπου γίνεται η επεξεργασία της. Το επίπεδο εξόδου υπολογίζει τις πιθανότητες εκτέλεσης κάθε διαθέσιμης ενέργειας. Η τιμή της συνάρτησης σφάλματος οπισθοδρομείται στα προηγούμενα επίπεδα για παραμετροποίηση των βαρών.

3.3.2 Διπλά DQN

Ένα σημαντικό πρόβλημα που προκύπτει από την χρήση των κλασσικών δικτύων DQN είναι η υπερτίμηση των μέγιστων τιμών Q, που έχουν ως αποτέλεσμα αρνητικές επιπτώσεις στην επίδοση του RL συστήματος. Αυτό είναι λογικό, αφού οι προβλεπόμενες αλλά και οι πραγματικές τιμές υπολογίζονται από το ίδιο το RL σύστημα, με αποτέλεσμα να διαφέρουν μετά από κάθε υπολογιστική επανάληψη. Τέτοια συμπεριφορά δεν είναι επιθυμητή, διότι είναι σημαντικό να υπάρχει μία σχετική σταθερότητα στις πραγματικές τιμές σε χρόνο εκπαίδευσης. Η ιδέα των διπλών Deep Q Networks (van Hasselt, 2010), έρχεται για να ελαττώσει το πρόβλημα της υπερτίμησης σε περιβάλλοντα μεγάλης κλίμακας, έχοντας ως συνέπεια αυξημένη σταθερότητα και αξιοπιστία στη διαδικασία της μάθησης. Η συγκεκριμένη τεχνική δείχνει πως οδηγεί στην εύρεση καλύτερων τακτικών για την μεγιστοποίηση της συνάρτησης ανταμοιβής στο Doom περιβάλλον.

Η λογική της συγκεκριμένης τεχνικής λέει πως συμπληρωματικά με ένα αρχικό μοντέλο δικτύου θ, χρησιμοποιείται ένα δεύτερο μοντέλο θ' που χρησιμοποιεί την ίδια αρχιτεκτονική και τις ίδιες υπερ-παραμέτρους με το πρώτο. Ο ρόλος του δεύτερου μοντέλου είναι να αξιολογεί τις εκτιμήσεις των ζευγών καταστάσεων – ενεργειών που παράγει το αρχικό μοντέλο. Αυτό επιτυγχάνεται με τη χρήση του συνόλου των βαρών των θ και θ' Η διαδικασία της εκτίμησης των τιμών από το αρχικό δίκτυο θ μπορεί να αναπαρασταθεί κατά τον κλασσικό τρόπο ως εξής:

$$Y^{Q}_{t} = r(s',a) + \gamma Q(s_{t+1}. \operatorname{argmax} Q(s',a;\theta_t);\theta_t)$$

Η σφαλματική υπερτίμηση για την τεχνική Double DQN μπορεί να γραφτεί ως:

$$Y^{DoubleDQN}_{t} = r(s',a) + \gamma Q(s_{t+1}. \operatorname{argmax} Q(s',a;\theta_t);\theta_t')$$

υποθέτωντας πως η επιλογή της βέλτιστης κίνησης γίνεται με βάση τις παραμέτρους του θ και αξιολογούνται με βάση τις παραμέτρους του θ' για τη δεδομένη χρονική στιγμή t. Επίσης, η σφαλματική υπερτίμηση λαμβάνει τιμές στο διάστημα [-1, 1]. Χρειάζεται να ελαχιστοποιηθεί το ελάχιστο τετραγωνικό σφάλμα μεταξύ των δύο μοντέλων. Αυτό γίνεται υπολογίζοντας τις Q τιμές της τρέχουσας αλλά και της επόμενης περιβαλλοντικής κατάστασης και για τα δύο μοντέλα. Προτιμάται ο ελάχιστος υπολογισμός των Q τιμών της επόμενης κατάστασης για την εύρεση της αναμενόμενης Q τιμής. Οι τιμές των βαρών

του δικτύου αξιολόγησης ενημερώνονται περιοδικά κατά ένα σταθερό αριθμό χρονικών επαναλήψεων.

Όπως προαναφέρθηκε, τα δύο αυτά μοντέλα DQN δικτύων περιγράφονται από την ίδια αρχιτεκτονική και το ίδιο σύνολο υπερ-παραμέτρων. Στο Παράρτημα Γ βρίσκεται ο κώδικας που υλοποιεί το DQN δίκτυο. Στους παρακάτω πίνακες καταγράφονται η από άκρο-σε-άκρο αρχιτεκτονική, η προσέγγιση του υπολογισμού σφάλματος και οι υπερ-παράμετροι που χρησιμοποιήθηκαν στην υλοποίηση των Διπλών DQN για την παρούσα πτυχιακή μελέτη.

Υπερ-παράμετροι	Τιμές
Παράγοντας μείωσης	0.99
Ρυθμός μάθησης	0.0001
Αρχικό έψιλον	1
Τελικό έψιλον	0.0001
Συνολικό μέγεθος μνήμης	50000
Μέγεθος δέσμης μνήμης	32
Αριθμός καρέ ανά ενέργεια	4
Συχνότητα ενημέρωσης δικτύου αξιολόγησης (χρονικά βήματα)	3000
Συνολικά βήματα εκπαίδευσης	300000 (1200000 σύνολο καρέ)

Πίνακας 5: Υπερ-παράμετροι Διπλών DQN

Επίπεδα Δικτύου	Μέγεθος φίλτρων	Μέγεθος kernel	Μέγεθος stride	Μέγεθος υπο-δείγματος	Συνάρτηση ενεργοποίησης	Τμήματα εξόδου
1ο κρυφό επίπεδο	32	8	8	(4, 4)	ReLU	-
2ο κρυφό επίπεδο	64	4	4	(2, 2)	ReLU	-
3ο κρυφό επίπεδο	64	3	3	-	ReLU	-

Πλήρως-συνδεδεμένο επίπεδο	-	-	-	-	ReLU	512
Επίπεδο εξόδου	-	-	-	-	Γραμμική	Μέγεθος διαθέσιμων ενεργειών

Πίνακας 6: Αρχιτεκτονική δικτύου DQN

Παράμετρος	Τιμή
Βελτιστοποιητής	Adam
Ρυθμός μάθησης	0.0001
Συνάρτηση σφάλματος	MSE

Πίνακας 7: Παράμετροι υπολογισμού σφάλματος δικτύου DQN

3.3.3 Αλγόριθμος C51

Μία βελτιωμένη έκδοση του αλγόριθμου Q-Learning αποτελεί η μοντελοποίηση μίας κατανομής των συνολικών μελλοντικών ανταμοιβών που αντικαθιστά την κατά προσέγγιση εκτίμηση. Όπως έχει εξηγηθεί, οι μελλοντικές ανταμοιβές υπολογίζονται μέσα από μία αναδρομική διαδικασία που περιγράφεται από την εξίσωση Bellman. Ο αλγόριθμος C51 (Bellemare et al. 2017) βασίζεται σε αυτή την ιδέα, με τη σημαντική διαφοροποίηση πως αντί να βασίζεται σε μία εκτίμηση τιμής, χρησιμοποιεί μία κατανομή εκτιμήσεων για τη διαδικασία της μάθησης. Η στοχαστικότητα του Doom περιβάλλοντος μπορεί σε αρκετές περιπτώσεις να φέρει απρόσμενες για το σύστημα καταστάσεις (απρόβλεπτη εξόντωση πράκτορα, μετατόπιση εχθρών σε ανεξερεύνητο σημείο), ειδικά αν η τακτική που ακολουθείται για την επιλογή της βέλτιστης κίνησης είναι βασισμένη σε μία εκτίμηση τιμής. Μαθαίνοντας από μία πολυτροπική κατανομή με τη χρήση του συγκεκριμένου αλγορίθμου, ο RL πράκτορας αποκτά πρόσβαση σε ένα πλουσιότερο σύνολο εκτιμήσεων που θα έχει ως συνέπεια την εύρεση αποτελεσματικότερων τακτικών, αν συγκριθεί η παρούσα ιδέα με τις ήδη υπάρχουσες που ανήκουν στο πεδίο του Q-Learning. Στη θέση της εξίσωσης Bellman, όπου η εκτίμηση περιγραφόταν από μία

μεταβλητή $Q(s, \alpha)$ για μία δεδομένη κατάσταση s και μία ενέργεια α, έρχεται ο ορισμός της κατανομής Bellman και μπορεί να χαρακτηριστεί από μία τυχαία μεταβλητή $Z(s, \alpha)$ που υποδηλώνει την κατανομή όλων των μελλοντικών ανταμοιβών. Η μεταβλητή υπολογίζεται παρόμοια την τιμή Q από την εξίσωση της κατανομής Bellman ως εξής:

$$Z(s, \alpha) = r + \gamma Z(s', \alpha')$$

Πρόκειται για ισότητα μεταξύ του τρέχοντος Z και των επομένων Z' τα οποία υπολογίζονται αναδρομικά. Πλέον, στα ίδια πλαίσια με τον κλασσικό Q-Learning αλγόριθμο, μπορεί να εκτιμηθεί επαναληπτικά η κατανομή τιμής Z. Η αναπαράσταση της περιγράφεται από μία διακριτή κατανομή όπου το πεδίο ορισμού της περιγράφεται από ένα πλήθος διακριτών τιμών ομοιόμορφα αραιωμένων. Το σύνολο αυτών των διακριτών τιμών είναι κοινό και για την τρέχουσα Z αλλά και για την επόμενη Z. Αυτό επιτρέπει τον υπολογισμό της απόστασης μεταξύ των δύο κατανομών με τη χρήση της συνάρτησης σφάλματος κατηγορηματικής διασταυρωμένης εντροπίας (categorical cross-entropy). Κάθε χρονικό βήμα της εξίσωσης κατανομής Bellman αναλύεται από τα παρακάτω βήματα:

1. Λαμβάνεται δείγμα της κατάστασης s στην οποία εκτελείται ενέργεια α και επιστρέφεται από το περιβάλλον η επόμενη κατάσταση s' και η ανταμοιβή r, όπως ορίζονται από την MDP διασικασία. Η κατανομή της επόμενης κατάστασης ορίζεται ως $Z(s', \alpha')$
2. Η κατανομή επόμενης κατάστασης κλιμακώνεται με τη χρήση του παράγοντα μείωσης γ
3. Η ίδια κατανομή μετατοπίζεται δεξιά κατά r (ανταμοιβή) και προκύπτει η πραγματική κατανομή Z' για την οποία ισχύει $Z' = r(s, \alpha) + \gamma Z'(s', a')$
4. Πραγματοποιείται προβολή της πραγματικής κατανομής Z' στις διακριτές τιμές της προβλεπόμενης κατανομής Z, ελαχιστοποιώντας το σφάλμα της συνάρτησης διασταυρωμένης εντροπίας μεταξύ Z και Z'

Μετά από πειραματισμούς με διαφορετικό πλήθος διακριτών τιμών, έχει αποδειχθεί πως ο παρόν αλγόριθμος λειτουργεί βέλτιστα με χρήση 51 διακριτών τιμών, οι οποίες αναφέρονται στη βιβλιογραφία συχνά ως atoms. Η συγκεκριμένη μοντελοποίηση θα

μπορούσε να χαρακτηριστεί και ως ένα πρόβλημα κατηγοριοποίησης εικόνων (image classification), με την εικόνα οθόνης (κατάσταση) παιχνιδιου να αποτελεί το σύνολο δεδομένων εισόδου και τις 51 διακριτές τιμές να αποτελούν τα χαρακτηριστικά κατηγοριοποίησης. Οι πιθανοτικές τιμές που προκύπτουν από κάθε χαρακτηριστικό της κατανομής *Z'* λειτουργούν ως τα επιθυμητά αποτελέσματα για την καθοδήγηση της εκπαίδευσης. Στο Παράρτημα Δ βρίσκεται ο κώδικας που υλοποιεί το δίκτυο κατανομής τιμών το οποίο χρησιμοποιεί ο αλγόριθμος C51. Παρακάτω εμφανίζονται οι υπερ-παράμετροι και η αρχιτεκτονική του δικτύου κατανομής τιμών που εφαρμόστηκαν κατά την υλοποίηση του αλγόριθμου C51.

Υπερ-παράμετροι	Τιμές
Παράγοντας μείωσης	0.99
Ρυθμός μάθησης	0.0001
Αρχικό έψιλον	1
Τελικό έψιλον	0.0001
Συνολικό μέγεθος μνήμης	50000
Μέγεθος δέσμης μνήμης	32
Αριθμός καρέ ανά ενέργεια	4
Συχνότητα ενημέρωσης δικτύου αξιολόγησης (χρονικά βήματα)	3000
Πλήθος διακριτών τιμών	51
Συνολικά βήματα εκπαίδευσης	300000 (1200000 σύνολο καρέ)

Πίνακας 8: Υπερ-παράμετροι δικτύου κατανομής τιμών

Επίπεδα Δικτύου	Μέγεθος φίλτρων	Μέγεθος kernel	Μέγεθος stride	Μέγεθος υπο-δείγματος	Συνάρτηση ενεργοποίησης	Τμήματα εξόδου
1ο κρυφό επίπεδο	32	8	8	(4, 4)	ReLU	-
2ο κρυφό επίπεδο	64	4	4	(2, 2)	ReLU	-

3ο κρυφό επίπεδο	64	3	3	-	ReLU	-
Πλήρως-συνδεδεμένο επίπεδο	-	-	-	-	ReLU	512
Επίπεδο εξόδου	-	-	-	-	Softmax	A x 51

Πίνακας 9: Αρχιτεκτονική δικτύου κατανομής τιμών. Ως *A* ορίζεται το μέγεθος των διαθέσιμων ενεργειών. Σε κάθε ενέργεια ανατίθεται μία κατανομή τιμής με πεδίο 51 διακριτών τιμών (atoms).

Παράμετρος	Τιμή
Βελτιστοποιητής	Adam
Ρυθμός μάθησης	0.0001
Συνάρτηση σφάλματος	Κατηγορηματική Διασταυρωμενή Εντροπία

Πίνακας 10: Παράμετροι υπολογισμού σφάλματος δικτύου κατανομής τιμών

4 Στατιστικά αποτελέσματα

Το περιεχόμενο του συγκεκριμένου κεφαλαίου αποτελείται από διαγράμματα των αποτελεσμάτων τα οποία προέκυψαν από την εξέταση των υλοποιήσεων που ερευνήθηκαν και αναλύθηκαν στο προηγούμενο κεφάλαιο. Εξηγείται ο τρόπος με τον οποίο ρυθμίστηκαν οι όποιες παράμετροι χρειάστηκαν στο στάδιο αξιολόγησης ώστε να μπορέσουν να παραχθούν οι μετρήσεις που προέρχονται από την επαναληπτική διαδικασία της εκπαίδευσης των πρακτορικών συστημάτων που μελετήθηκαν. Επίσης, σχολιάζονται τα προαναφερθέντα στατιστικά μοντέλα, με βάση τι θεωρήθηκε αναμενόμενο πως θα προκύψει και τι όχι. Χρειάζεται σε αυτό το σημείο να δοθεί έμφαση στο μεγάλο πλήθος των δυνατοτήτων εξέλιξης των παραπάνω μεθοδολογιών με σκοπό την ύπαρξη πιο λεπτομερών και εύστοχων αποτελεσμάτων, αλλά και την αποτελεσματικότερη αλληλεπίδραση με το περιβάλλον μαθησης. Η πολυπλοκότητα των παραπάνω αλγορίθμων ενισχυτικής μάθησης σε συνδυασμό με την έλλειψη παραγωγικότητας που εμφανίζεται λόγω του ότι η φάση της εκπαίδευσης τέτοιων συστημάτων καταλαμβάνει μεγάλο χρονικό διάστημα, ειδικά σε μηχανήματα καθημερινής χρήσης, καθιστούν σαφές το γεγονός της ύπαρξης περιθωρίου βελτίωσης των διαδικασιών και συνεπώς την εξαγωγή πιο λογικών και ακριβέστερων στατιστικών μετρήσεων.

4.1 Ρύθμιση περιβάλλοντος αξιολόγησης

Κάτι που δεν αναφέρθηκε στις προηγούμενες ενότητες και αξίζει αναφορά είναι πως κατά την διάρκεια της εκπαίδευσης ενός RL πράκτορα, τα ενημερωμένα βάρη του νευρωνικού δικτύου του τρέχοντος συστήματος αποθηκεύονται σε ξεχωριστό αρχείο (.h5 μορφή) για μελλοντική παρακολούθηση και αξιολόγηση της επίδοσης που αναπτύχθηκε. Για την φόρτωση των δεδομένων των βαρών και του εκτιμώμενου πίνακα *Q*, αρκεί απλώς να κληθεί η κατάλληλη συνάρτηση που θα φέρει στην εκτελούμενη εφαρμογή τα ζητούμενα δεδομένα. Ύστερα, ακολουθεί μία σειρά από βήματα για την παρακολούθηση της λειτουργίας του πράκτορα που έχει ως εξής:

1. Αρχικοποίηση περιβάλλοντος Doom
2. Καθορισμός επεισοδίων (παρτίδες παιχνιδιού) τα οποία περιγράφουν τη διάρκεια της διαδικασίας παρακολούθησης

3. Ο πράκτορας ξεκινάει να δέχεται καταστάσεις από το περιβάλλον, προϋποθέτοντας την ύπαρξη της προεπεξεργασμένης στοίβας καρέ
4. Ο πράκτορας επιλέγει την καλύτερη ενέργεια, αυτή τη φορά όμως βασισμένος μοναχά στην greedy πολιτική. Με άλλα λόγια, επιλέγει πάντα την ενέργεια που φέρνει την μέγιστη ανταμοιβή για τη δεδομένη κατάσταση και δεν λειτουργεί επιλέγοντας τυχαίες κινήσεις (e = 0)

Επίσης, πρέπει να αναφερθεί και η διαδικασία καταγραφής των μετρικών που πληθαίνουν τα στατιστιστικά μοντέλα που προέκυψαν. Κατά το στάδιο της υλοποίησης, αποφασίστηκε πως η εγγραφή των αποτελεσμάτων θα γίνεται σε ξεχωριστό αρχείο κειμένου (.txt μορφή), ανά 50 επεισόδια παιχνιδιού. Ένα μεμονωμένο επεισόδιο παιχνιδιού αποτελείται από το σύνολο των διαδοχικών (s, α) που προηγούνται μίας τερματικής κατάστασης. Ουσιαστικά, με την άφιξη της τερματικής κατάστασης στο πρακτορικό σύστημα σημάνεται και η λήξη του επεισοδίου. Τα αποτελέσματα που καταγράφηκαν ανήκουν στο δίαστημα των 1500 επεισοδίων παιχνιδιού.

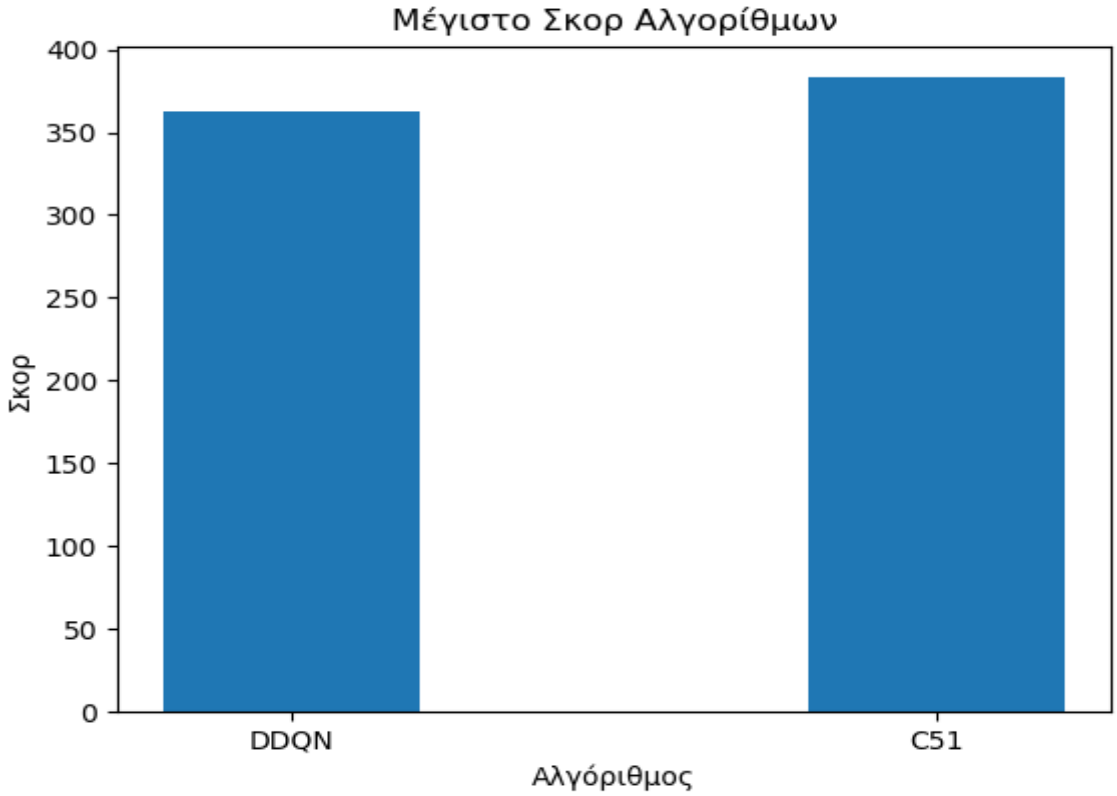
4.2 Αποτύπωση γραφημάτων

Με την ολοκλήρωση της διαδικασίας αξιολόγησης των RL συστημάτων-μεθοδολογιών που καλύφθηκαν παραπάνω, χρειάζεται να ακολουθήσει μία παρουσίαση των γραφικών παραστάσεων που περιγράφουν τις διαφορετικές τιμές που σημείωσαν οι δύο τεχνικές οι οποίες αξιοποιήθηκαν στην παρούσα πτυχιακή εργασία. Η χρήση των μεθόδων που είναι ενσωματωμένες στην standard βιβλιοθήκη της Python για τη διαχείριση εισόδου/εξόδου, όπως και της βιβλιοθήκης matplotlib για την σχεδίαση διαγραμμάτων, συνέβαλε σε ικανοποιητικό βαθμό στην αποτύπωση των τιμών σε γραφική μορφή. Στις επόμενες υποενότητες εμφανίζονται τα δεδομένα που παρήχθησαν, συνοδευόμενα από τα γραφήματα τους, ενώ γίνεται σχολιασμός των αποτελεσμάτων.

4.2.1 Μέγιστο Αλγοριθμικό Σκορ

Μέγιστο σκορ πράκτορα DDQN: 363

Μέγιστο σκορ πράκτορα C51: 383



Διάγραμμα 3: Σύγκριση του μέγιστου σκορ των πρακτόρων DDQN και C51

Αρχικά πρέπει να δοθεί ένας ορισμός για το την μεταβλητή του σκορ. Πρόκειται για μία μετρική που εκφράζει την μέγιστη υγεία (health μεταβλητή παιχνιδιου). Η μετρική αυτή αυξάνεται κατά ένα για κάθε κατάσταση εντός επεισοδίου παιχνιδιού η οποία δεν θεωρείται τερματική. Ουσιαστικά λειτουργεί ως μεσολαβητής της επίδοσης του πράκτορα στο τρέχον περιβάλλον. Από την παραπάνω απεικόνιση φαίνεται πως ο πράκτορας C51 καταφέρνει να πετύχει ελαφρώς μεγαλύτερο σκορ από τον πράκτορα DDQN, το οποίο ήταν και το αναμενόμενο αποτέλεσμα.

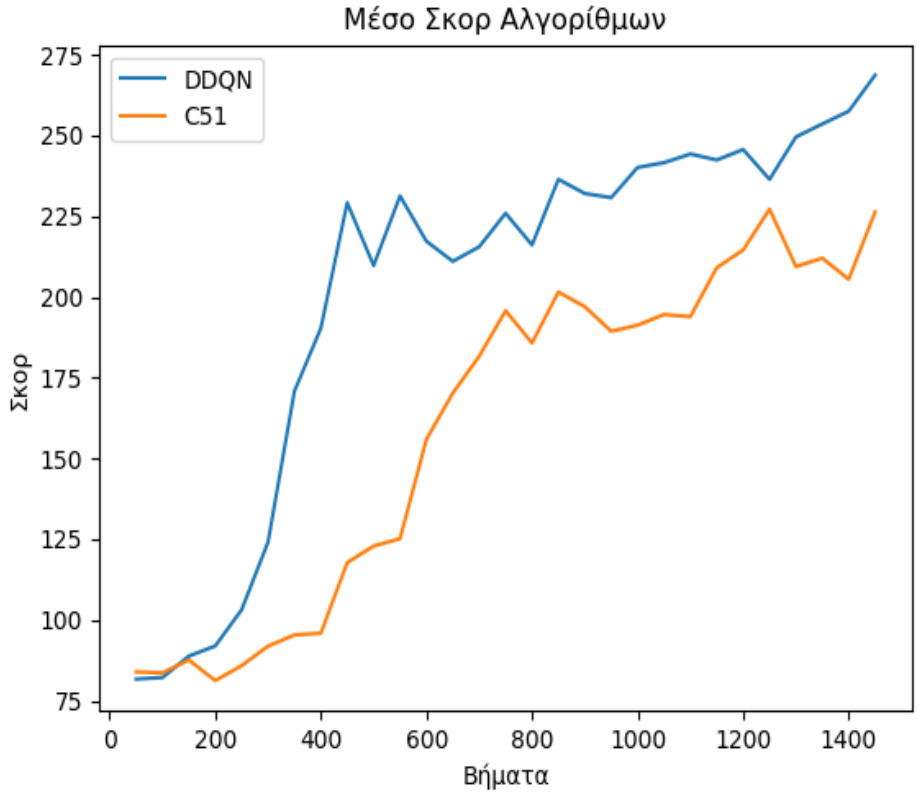
4.2.2 Μέσο Αλγοριθμικό Σκορ

Ακολουθούν για κάθε μία από τις υλοποιημένες μεθοδολογίες οι λίστες με τα δεδομένα των μέσων όρων των σκορ που προέκυπταν κάθε 50 επεισόδια παιχνιδιών.

Μέσες σκορ τιμές πράκτορα DDQN: [81.65, 82.14, 88.78, 91.92, 103.14, 124.08, 170.84, 190.32, 229.2, 209.74, 231.3, 217.28, 211.04, 215.56, 225.94, 216.12, 236.46, 232.06, 230.78, 240.12, 241.6, 244.3, 242.44, 245.68, 236.44, 249.58, 253.6, 257.56, 268.74]

Μέσες σκορ τιμές πράκτορα C51: [83.9, 83.58, 87.58, 81.2, 85.82, 91.86, 95.34, 95.86, 117.74, 122.86, 125.14, 155.9, 170.24, 181.66, 195.74, 185.78, 201.54, 197.02, 189.4, 191.26, 194.58, 193.92, 209.08, 214.56, 227.2, 209.42, 212.04, 205.48, 226.34]

Στην επόμενη σελίδα απεικονίζονται σε διαγραμματική μορφή τα δεδομένα και από τους δύο RL πράκτορες συγκριτικά.



Διάγραμμα 4: Απεικόνιση των μέσων τιμών των σκορ των δύο πρακτόρων

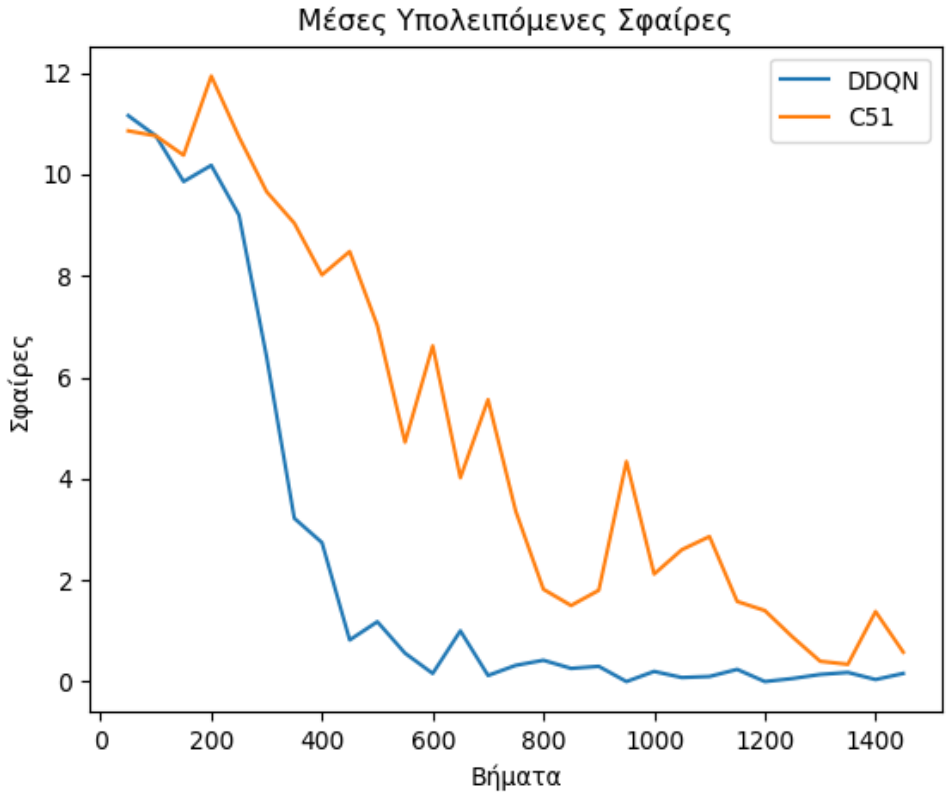
Το μέσο σκορ ακολουθεί την ίδια λογική με το μέγιστο πρακτορικό σκορ που αναλύθηκε παραπάνω. Παρόλο που ο πράκτορας C51 είχε σημειώσει μεγαλύτερο σκορ

από τον DDQN, τα αποτελέσματα του διαγράμματος 5 δεν είναι αντιπροσωπευτικά του μέγιστου σκορ του συγκεκριμένου συστήματος. Έτσι, ο DDQN δείχνει να εμφανίζει μεγαλύτερα ποσά μέσων σκορ. Θεωρητικά τα αποτελέσματα αυτά έχουν ως ένα βαθμό συσχέτιση με τη φύση των καταστάσεων του Doom περιβάλλοντος.

4.2.3 Μέσος Αριθμός Υπολειπόμενων Σφαιρών

Μέσες υπολειπόμενες σφαίρες πράκτορα DDQN: [11.16, 10.76, 9.86, 10.18, 9.2, 6.4, 3.22, 2.74, 0.82, 1.18, 0.56, 0.16, 1.0, 0.12, 0.32, 0.42, 0.26, 0.3, 0.0, 0.2, 0.08, 0.1, 0.24, 0.0, 0.06, 0.14, 0.18, 0.04, 0.16]

Μέσες υπολειπόμενες σφαίρες πράκτορα C51: [10.86, 10.76, 10.38, 11.94, 10.74, 9.66, 9.04, 8.02, 8.48, 7.02, 4.72, 6.62, 4.02, 5.56, 3.36, 1.82, 1.5, 1.8, 4.34, 2.12, 2.6, 2.86, 1.58, 1.4, 0.88, 0.4, 0.34, 1.38, 0.58]



Διάγραμμα 5: Απεικόνιση των μέσων τιμών των σφαιρών των δύο πρακτόρων

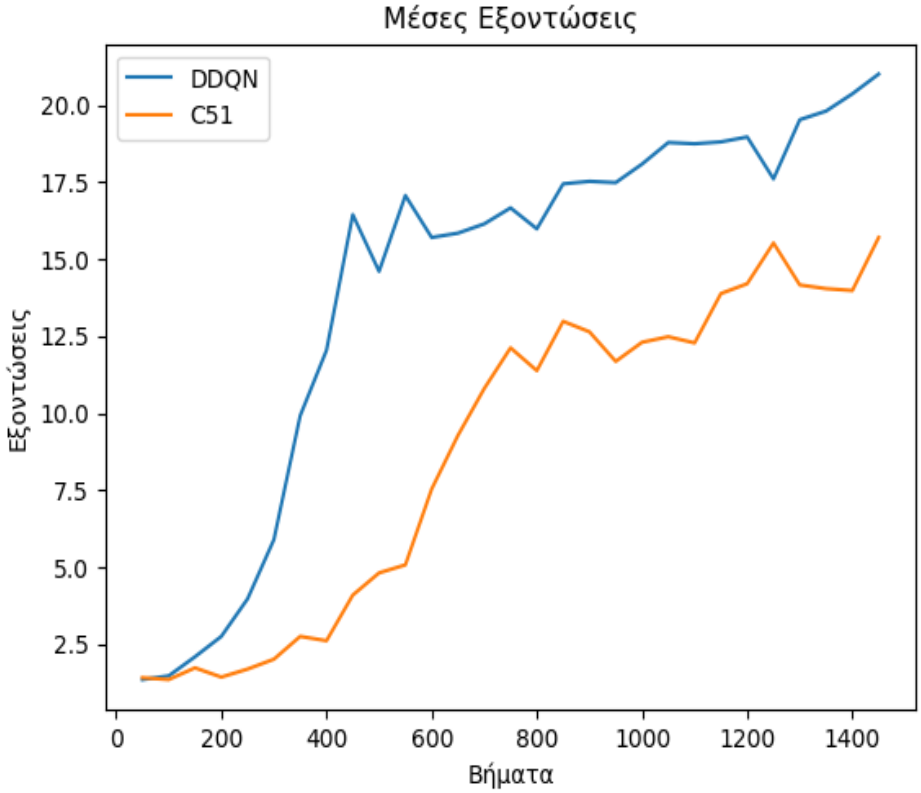
Κατά την εκκίνηση ενός επεισοδίου, ο πράκτορας λαμβάνει ένα σύνολο από σφαίρες τις οποίες καλείται να αξιοποιήσει με αποδοτικό τρόπο για να εξουδετερώσει τους εχθρούς που συναντά στο Doom περιβάλλον. Είναι επιθυμητό η τιμή της μεταβλητής των σφαιρών να μειώνεται με την πάροδο των επεισοδίων παιχνιδιού, καθώς αυτό υπονοεί πως ο πράκτορας, έπειτα από εκτέλεση βέλτιστης τακτικής, προλαβαίνει στην προθεσμία του ενός επεισοδίου και χρησιμοποιεί μεγαλύτερο πλήθος των σφαιρών του νικώντας τους αντιπάλους του πριν ο ίδιος εξοντωθεί. Πράγματι, και οι δύο πράκτορες καταφέρνουν σε μελλοντικό και τερματίζουν τα επεισόδια με μικρότερο αριθμό σφαιρών, το οποίο αποτελεί θετικό γεγονός.

4.2.4 Μέσος Αριθμός Εξοντώσεων

Σε κάθε βιντεοπαιχνίδι βολών, ο παίκτης επιδιώκει πάντα να πετυχαίνει μεγάλο αριθμό εξοντώσεων εντός μίας συνεδρίας παιχνιδιού. Τον ίδιο σκοπό προσπαθεί να πετύχει και ένας RL πράκτορας λειτουργώντας σε ένα περιβάλλον όπως είναι αυτό του Doom. Οι επόμενες μετρήσεις περιγράφουν κατά μέσο όρο τα ποσά των εξοντώσεων που σημείωσαν οι δύο πράκτορες κατά τη διαδικασία λειτουργίας τους.

Μέσοι όροι εξοντώσεων πράκτορα DDQN: [1.36, 1.48, 2.1, 2.76, 3.98, 5.9, 9.92, 12.04, 16.44, 14.6, 17.06, 15.7, 15.84, 16.14, 16.66, 15.98, 17.44, 17.52, 17.48, 18.08, 18.78, 18.74, 18.8, 18.96, 17.6, 19.52, 19.8, 20.36, 21.0]

Μέσοι όροι εξοντώσεων πράκτορα C51: [1.42, 1.36, 1.74, 1.44, 1.7, 2.02, 2.76, 2.62, 4.1, 4.82, 5.08, 7.54, 9.28, 10.8, 12.12, 11.38, 12.98, 12.64, 11.68, 12.3, 12.48, 12.28, 13.88, 14.2, 15.52, 14.16, 14.04, 13.98, 15.7]

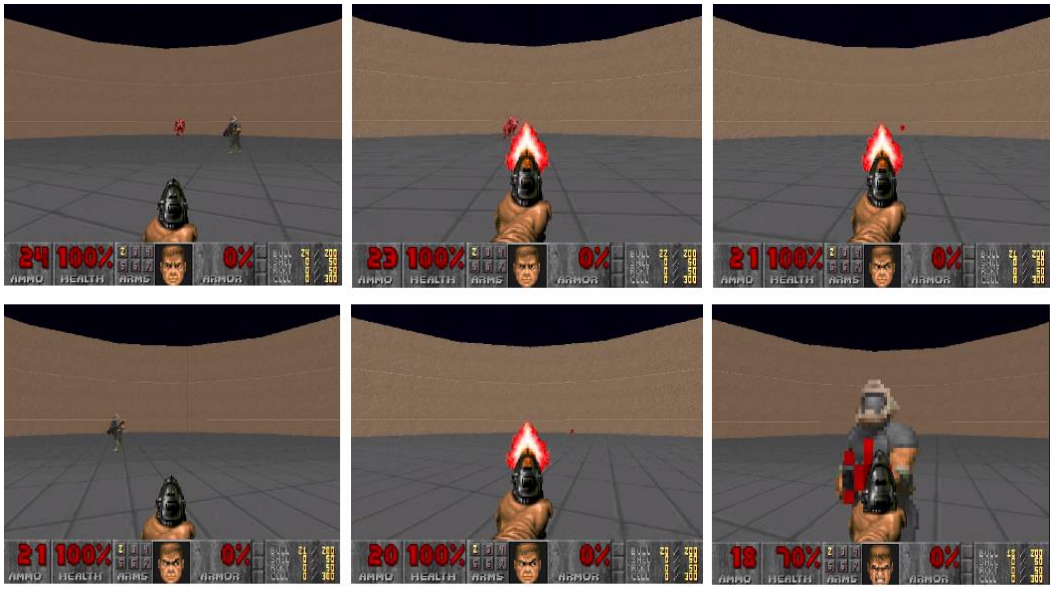


Διάγραμμα 6: Μέσοι όροι εξοντώσεων των δύο πρακτόρων

Καθώς ανεβαίνει ο αριθμός των επεισοδίων, αυξάνει και ο αριθμός των εξοντώσεων που κατορθώνουν τα πρακτορικά συστήματα. Αυτά τα ποσά είναι αντιστρόφως ανάλογα της κατανάλωσης των σφαιρών που αναλύθηκε στο προηγούμενο διάγραμμα.

4.2.5 Παραδείγματα Καλών και Κακών Επιλογών Αλγορίθμων

Και στις δύο μεθόδους που αναλύονται στις παραπάνω ενότητες, μια τακτική θεωρείται καλή όταν γίνεται σωστή διαχείριση των πόρων του παιχνιδιού, όπως είναι για παράδειγμα η υγεία του πράκτορα και οι σφαίρες που διαθέτει. Είναι επιθυμητό οι σφαίρες να καταναλώνονται αποκλειστικά για την εξόντωση εχθρών, χωρίς να επιτρέπεται η σπατάλη υγείας του πράκτορα. Η παρακάτω εικόνα αντιστοιχεί σε διαδοχικά στιγμιότυπα παιχνιδιού, στα οποία απεικονίζονται καταστάσεις πριν και μετά την εξόντωση αντιπάλων από το πρακτορικό σύστημα και αφορά παραδείγματα καλών επιλογών των αλγοριθμικών προσεγγίσεων πάνω στο Doom περιβάλλον.



Εικόνα 12: Παράδειγμα καλής αλγοριθμικής τακτικής

Αν παρατηρηθούν οι μετρήσεις των σφαιρών (AMMO) και της υγείας (HEALTH), προκύπτει το συμπέρασμα ότι γίνεται σωστή αξιοποίηση τους από τα συστήματα, με τις σφαίρες να χρησιμοποιούνται κυρίως για την εξόντωση των αντιπάλων, χωρίς να προκαλείται κάποια ζημιά στην υγεία. Παρατηρείται ότι δεν γίνεται αχρείαστη σπατάλη στις σφαίρες σε σημεία που δεν εμφανίζονται οι εχθροί και αυτό δείχνει πως οι ενέργειες των συστημάτων DDQN και C51 δεν βασίζονται στην τυχαιότητα επιλογής, αλλά στην επιλογή βέλτιστων εκτιμήσεων που αντιστοιχούν στις παραπάνω καταστάσεις του περιβάλλοντος. Στις παρακάτω εικόνες εμφανίζονται κακές πρακτικές που εμφανίστηκαν στα αποτελέσματα κατά την επιλογή ενεργειών από τα συστήματα.

Εικόνα 13: Σπατάλη σφαιρών και μειωμένη υγεία

Εικόνα 14: Μειωμένη απόσταση του αντιπάλου σε σχέση με τον πράκτορα

Ένα επιθυμητό αποτέλεσμα θα ήταν οι αντίπαλοι να μην πλησιάζουν σε πολύ κοντινή απόσταση τον πράκτορα, αφήνωντας τον εκτεθειμένο σε εχθρικές επιθέσεις. Τέτοιες

καταστάσεις είναι λογικό να έχουν ως αποτέλεσμα την μείωση της υγείας του πράκτορα. Οι παραπάνω εικόνες δείχνουν την ανικανότητα των πρακτορικών συστημάτων σε ορισμένες περιπτώσεις να χειριστούν αποδοτικά την έννοια της επιβίωσης στο δεδομένο περιβάλλον. Τέτοια κακή συμπεριφορά οφείλεται στο γεγονός πως ορισμένες από τις καταστάσεις του υπερβολικά μεγάλου χώρου καταστάσεων του παιχνιδιού ενδεχομένως έχουν μείνει ανεξερεύνητες ή δεν έχει γίνει σωστή εκτίμηση των Q τιμών που αντιστοιχούν στις ενέργειες αυτών των καταστάσεων. Πιθανώς απαιτείται αρκετά μεγαλύτερο πλήθος επαναλήψεων εκπαίδευσης για την επικάλυψη αυτών των σφαλμάτων.

5 Συμπεράσματα και μελλοντικές επεκτάσεις

Στα πλαίσια της παρούσας εργασίας πραγματοποιήθηκε προσπάθεια ανάπτυξης αυτόνομων πρακτορικών συστημάτων ενισχυτικής μάθησης, με σκοπό την αποτελεσματική επίτευξη στόχων χωρίς την επίβλεψη κάποιου τρίτου προσώπου. Οι πρόσφατες βελτιώσεις που έχουν δημοσιευθεί από διάφορες ερευνητικές ομάδες και γενικά η άνθηση του κλάδου της ενισχυτικής μάθησης τα τελευταία χρόνια, αποτέλεσε σημαντική πηγή έμπνευσης της εργασίας αυτής. Πιο συγκεκριμένα, αυτά τα RL συστήματα δείχνουν τις ικανότητες τους λειτουργώντας πάνω σε τρισδιάστατα περιβάλλοντα βιντεοπαιχνιδιών που χαρακτηρίζονται ως στοχαστικά και μερικώς παρατηρήσιμα. Τέτοια χαρακτηριστικά είναι που δημιουργούν μεγάλη πρόκληση στην ανάπτυξη έξυπνων ML συστημάτων και απαιτούν μεγάλά ποσά υπολογιστικών πόρων για την αποδοτική και παραγωγική εκπαίδευση και αξιολόγηση τους. Για το σκοπό αυτής της μελέτης, το περιβάλλον που αναλύθηκε και στο οποίο δοκιμάστηκαν τα υλοποιηθέντα συστήματα ήταν εκείνο του παιχνιδιού βολών τρίτου προσώπου Doom.

Μέσα από τους διαθέσιμους πόρους που παρέχονται από την ερευνητική πλατφόρμα ViZDoom, έγινε μία ανάλυση του χώρου καταστάσεων που αποτελούν το περιβάλλον του παιχνιδιού, καθώς και του συνόλου των εκτελέσιμων κινήσεων και της συνάρτησης ανταμοιβής. Αυτές οι ιδιότητες αποκτούν σημασία στο πλαίσιο των Διαδικασιών Μαρκοβιανών Αποφάσεων (MDP) και των Διαδικασιών Μαρκοβιανών Ανταμοιβών (MRP). Με την αξιοποίηση των μεθόδων που προσφέρει το ViZDoom API, κρίθηκε εύκολη η αρχικοποίηση του περιβάλλοντος παιχνιδιού δοθέντος ενός σεναριακού αρχείου ρυθμίσεων, παραμετροποιώντας κατάλληλα το παράθυρο οθόνης, το οποιό ουσιαστικά τροφοδοτεί την είσοδο των νευρωνικών δικτύων που αναπτύχθηκαν. Αποδείχθηκε ότι το διάνυσμα που προκύπτει από το γινόμενο των περιβαλλοντικών καταστάσεων και των διαθέσιμων κινήσεων προκαλεί πολύπλοκους υπολογισμούς για την εκτίμηση των βέλτιστων κινήσεων σε κάθε περίπτωση.

Για την αποτελεσματική επίτευξη στόχων στο Doom περιβάλλον, σε σύγκριση με άλλες μεθόδους (Policy Gradients, Actor-Critic κτλ.) αποδείχθηκε πως καλύτερη λύση

αποτελεί η αξιοποίηση του αλγορίθμου Q-Learning και η συγχώνευση του με την ιδεά του Deep Learning για την υλοποίηση των Deep Q Networks. Ο συγκεκριμένος αλγόριθμος έπειτα από πειραματισμούς έδειξε πως καταφέρνει και καταλήγει γρήγορα σε μία βέλτιστη τακτική που μεγιστοποιεί το μελλοντικό αποτέλεσμα του πράκτορα. Η συγκεκριμένη κατηγορία τεχνητών νευρωνικών δικτύων, η οποία φέρει αρκετές ομοιότητες με τα Convolutional Neural Networks, κρίθηκε η κατάλληλη επιλογή για την ανάλυση εικόνων και πιο συγκεκριμένα για την ανάλυση των καρέ που παράγει η μηχανή γραφικών του παιχνιδιού Doom. Προτάθηκαν δύο παραλλαγές του αλγορίθμου Q-Learning. Η πρώτη αντιστοιχεί στην έννοια των Double Deep Q Networks, η οποία επιλύει το πρόβλημα της υπερτίμησης των προβλεπόμενων τιμών Q και των πραγματικών τιμών Q. Η δεύτερη πρόταση στηρίζεται στην ιδέα ενός δικτύου κατανομής τιμών που αξιοποιείται από τον αλγόριθμο C51, οποίος χρησιμοποιεί ένα σύνολο 51 διακριτών τιμών για να προσφέρει κατανεμημένες εκτιμήσεις για κάθε διαθέσιμη ενέργεια.

Με αρκετή βεβαιότητα μπορεί να ειπωθεί πως και στις δύο διαφορετικές μεθόδους θα μπορούσε να υπάρξει μελλοντικά κάποια βελτιστοποίηση στον τρόπο με τον οποίο αυτές υλοποιούνται, ώστε να εξαχθούν πιο επιθυμητά αποτελέσματα. Επίσης, η χρήση των RNN δικτύων θα μπορούσε να αποτελέσει μία αξιοπρεπή επέκταση της συγκεκριμένης εργασίας, μιας και έχει αποδειχθεί πως υλοποιήσεις που ενσωματώνουν ένα τέτοιου είδους νευρωνικό δίκτυο είναι ικανές να προσαρμοστούν καλύτερα σε περιβαλλοντικές αλλαγές.

Βιβλιογραφία

[1] Tensorflow [Online]. Available: https://www.tensorflow.org/

[2] Keras API [Online]. Available: https://keras.io/

[3] ViZDoom Platform [Online]. Available: http://vizdoom.cs.put.edu.pl/

[4] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek & Wojciech Jaskowski: ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning (2016).

[5] Wikipedia, "Ενισχυτική μάθηση", 2013, [Online]. Available: https://el.wikipedia.org/wiki/Ενισχυτική_μάθηση

[6] Alexey Dosovitskiy, Vladlen Koltun: Learning To Act By Predicting The Future (2017)

[7] OpenAI DOTA 2 [Online]. Available: https://openai.com/blog/dota-2/

[8] Visual Doom AI Competition [Online]. Available: http://vizdoom.cs.put.edu.pl/competitions/vdaic-2018-cig

[9] Wikipedia, "AlphaGo", 2016, [Online]. Available:

https://el.wikipedia.org/wiki/AlphaGo

[10] AlphaGo Google DeepMind [Online]. Available:

https://deepmind.com/research/case-studies/alphago-the-story-so-far

[11] OpenAI Five [Online]. Available: https://openai.com/five/#overview

[12] OpenAI Five approach [Online]. Available:

https://openai.com/blog/openai-five/#coordination

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov: Proximal Policy Optimization Algorithms (2017)

[14] DeepMind AlphaStar [Online]. Available:

https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser: Attention Is All You Need (2014)

[16] Sepp Hochreiter, J• urgen Schmidhuber: Long Short-Term Memory (1997)

[17] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Kuttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, Rodney Tsing: StarCraft II: A New Challenge for Reinforcement Learning (2017).

[18] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, Koray Kavukcuoglu: IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures (2018)

[19] Long-Ji Lin: Self-Improving Reactive Agents Based On Reinforcement Learning,

Planning and Teaching (1992)

[20] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu & Raia Hadsell: Policy Distillation (2016)

[21] Guillaume Lample, Devendra Singh Chaplot: Playing FPS Games with Deep Reinforcement Learning (2016)

[22] Wikipedia, “Markov Property”, 1950, [Online]. Available:

https://en.wikipedia.org/wiki/Markov_property

[23] Wikipedia, “Markov Decision Process”, 1950, [Online]. Available:

https://en.wikipedia.org/wiki/Markov_decision_process

[24] Deep Q-Learning [Online]. Available:

https://brandonmorris.dev/2018/10/09/dql-vizdoom/

[25] Experience Replay [Online]. Available:

https://towardsdatascience.com/dqn-part-1-vanilla-deep-q-networks-6eb4a00febfb

[26] Frame Stack [Online]. Available:

https://danieltakeshi.github.io/2016/11/25/frame-skipping-and-preprocessing-for-deep-q-networks-on-atari-2600-games/

[27] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver & Daan Wierstra: Continuous Control With Deep Reinforcement Learning (2016)

[28] Marc G. Bellemare, Will Dabney, Remi Munos: A Distributional Perspective on Reinforcement Learning (2017)

Παραρτήματα κώδικα

Παράρτημα Α

Αρχικοποίηση περιβάλλοντος Doom

```
def initialize_vizdoom(config_file_path, is_rgb):
    print("Initializing doom...")
    game = vzd.DoomGame()
    game.load_config(config_file_path)
    game.set_window_visible(False)
    game.set_mode(vzd.Mode.PLAYER)
    if not is_rgb:
        game.set_screen_format(vzd.ScreenFormat.GRAY8)
    game.set_screen_resolution(vzd.ScreenResolution.RES_640X480)
    game.init()
    print("Doom initialized.")
    return game
```

Παράρτημα Β

Προεπεξεργασία καρέ

```
def preprocess_frame_v2(frame, size):
    frame = np.rollaxis(frame, 0, 3)
    frame = skimage.transform.resize(frame, size)
    frame = skimage.color.rgb2gray(frame)
    return frame
```

Παράρτημα Γ

Υλοποίηση μοντέλου νευρωνικού δικτύου DQN

```
def dqn(input_shape, action_size, alpha):
    model = Sequential()
    model.add(Convolution2D(32, 8, 8, subsample=(4, 4), activation='relu', input_shape=input_shape))
    model.add(Convolution2D(64, 4, 4, subsample=(2, 2), activation='relu'))
    model.add(Convolution2D(64, 3, 3, activation='relu'))
    model.add(Flatten())
    model.add(Dense(output_dim=512, activation='relu'))
    model.add(Dense(output_dim=action_size, activation='linear'))

    adam = Adam(lr=alpha)
    model.compile(loss='mse', optimizer=adam)
    return model
```

Παράρτημα Δ

Υλοποίηση μοντέλου νευρωνικού δικτύου κατανομής τιμών

```
def value_distribution_network(input_shape, num_atoms, action_size, alpha):
    state_input = Input(shape=(input_shape))
    cnn_feature = Convolution2D(32, 8, 8, subsample=(4, 4), activation='relu')(state_input)
    cnn_feature = Convolution2D(64, 4, 4, subsample=(2, 2), activation='relu')(cnn_feature)
    cnn_feature = Convolution2D(64, 3, 3, activation='relu')(cnn_feature)
    cnn_feature = Flatten()(cnn_feature)
    cnn_feature = Dense(512, activation='relu')(cnn_feature)

    distribution_list = []
    for i in range(action_size):
        distribution_list.append(Dense(num_atoms, activation='softmax')(cnn_feature))

    model = Model(input=state_input, output=distribution_list)

    adam = Adam(lr=alpha)
    model.compile(loss='categorical_crossentropy', optimizer=adam)

    return model
```

Παράρτημα Ε

Αρχείο ρυθμίσεων περιβαλλοντικού σεναρίου

```
doom_scenario_path = ../wad/defend_the_center.wad

death_penalty = 1

screen_resolution = RES_640X480
screen_format = CRCGCB
render_hud = True
render_crosshair = false
render_weapon = true
render_decals = false
render_particles = false
window_visible = true

episode_start_time = 10

episode_timeout = 2100

available_buttons =
    {
        TURN_LEFT
        TURN_RIGHT
        ATTACK
    }

available_game_variables = { KILLCOUNT AMMO2 HEALTH }

mode = PLAYER
doom_skill = 3
```

Παράρτημα ΣΤ

Επιλογή ενέργειας με βάση την πολιτική epsilon - greedy

```
def get_action(self, state):
    if np.random.rand() <= self.epsilon:
        action_idx = random.randrange(self.action_size)
    else:
        action_idx = self.get_optimal_action(state)

    return action_idx
```

Παράρτημα Ζ

Αποθήκευση μετάβασης στη μνήμη επανάληψης

```
def replay_memory(self, s_t, action_idx, r_t, s_t1, is_terminated, t):
    self.memory.append((s_t, action_idx, r_t, s_t1, is_terminated))
    if self.epsilon > self.final_epsilon and t > self.observe:
        self.epsilon -= (self.initial_epsilon - self.final_epsilon) / self.explore
    if len(self.memory) > self.max_memory:
        self.memory.popleft()
    if t % self.update_target_freq == 0:
        self.update_target_model()
```