Πρώτη Εργασία στο μάθημα της Υπολογιστικής Νοημοσύνης - Βαθιά Ενισχυτική Μάθηση

Ονοματεπώνυμο: Σταύρος Νικολαΐδης
ΑΕΜ: 3975

Μάιος 2024

1 Εισαγωγή

Σε αυτή την εργασία θα υλοποιήσουμε έναν αλγόριθμο βαθιάς ενισχυτικής μάθησης με σκοπό να μάθει να παίζει κάποιο παιχνίδι από την κλασσική παιχνιδομηχανή Atari. Επομένως, έχουμε να κάνουμε με ένα πρόβλημα ελέγχου.

2 Περιβάλλον

Θα χρησιμοποιήσουμε το πακέτο Gymnasium της Farama Foundation [3] (πρώην πακέτο gym της OpenAI) και το πακέτο τους Arcade Learning Enviroment [1]. Ουσιαστικά το πρώτο πακέτο μας δίνει πρόσβαση στις έτοιμες μεθόδους για το περιβάλλον (π.χ. Reward Wrappers) ενώ το δεύτερο μας δίνει πρόσβαση στα κλασσικά παιχνίδια της παιχνιδομηχανής Atari, όπως για παράδειγμα το Space Invaders και το Ms.Pacman.

3 Προεπεξεργασία Δεδομένων

Είναι σημαντικό να εξηγήσουμε την βασική προεπεξεργασία των δεδομένων μας. Στην συγκεκριμένη υλοποίηση τα δεδομένα εισόδου είναι εικόνες (screenshots) από το παιχνίδι, τα οποία μας δίνουν την κατάσταση του παιχνιδιού. Όμως, έπειδη είναι απίστευτα κοστοβόρο να δώσουμε σε ένα Συνελικτικό Νευρωνικό Δίκτυο μια έγχρωμη εικόνα μεγέθους (210x160x3) θα κάνουμε κάποια προεπεξεργασία ώστε και να αφαιρέσουμε περιττή πληροφορία (π.χ. το κάτω μέρος της οθόνης που δεν προσφέρει κάποια σημαντική πληροφορία στον παίκτη) αλλά και να μειώσουμε το υπολογιστικό κόστος. Η διαδικασία ακολουθεί τα παρακάτω βήματα:

- Κάνουμε την εικόνα ασπρόμαυρη.
- Περικόπτουμε την εικόνα για να αφαιρέσουμε το κάτω κομμάτι της οθόνης.
- Κανονικοποιούμε τις τιμές των pixels.

- Κάνουμε downscale (resize) την εικόνα στο μέγεθος (84x84).

Στο Figure 1 βλέπουμε την διαφορά ανάμεσα στην αρχική εικόνα και την επεξεργασμένη.

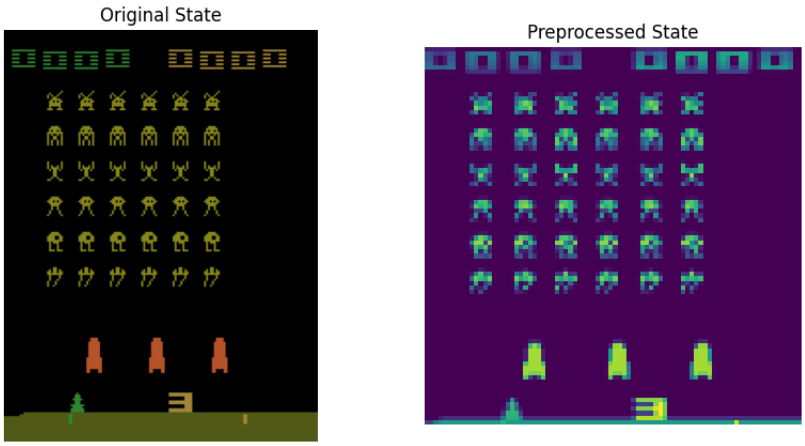

Figure 1: Αρχική και Επεξεργασμένη Κατάσταση Space Invaders

4 Υλοποίηση και Αρχιτεκτονική Συνελικτικού Νευρωνικού Δικτύου

Τόσο για την υλοποίηση όσο και για την εκπαίδευση του Συνελικτικού Νευρωνικού Δικτύου (CNN από εδώ και πέρα) θα χρησιμοποιήσουμε το framework Pytorch. Το Συνελικτικό Δίκτυο θα λειτουργεί ως ο estimator για τον αλγόριθμο Deep Q-Learning (βλέπε Ενότητα 5). Βέβαια υπάρχουν πολλές επιλογές για το είδος του δικτύου που θα χρησιμοποιήσουμε κυρίως όσων αφορά την αρχιτεκτονική του. Παρακάτω παρουσιάζουμε 2 από αυτές και έπειτα μέσα από τα πειράματα που θα τρέξουμε για αυτές θα συγκρίνουμε ποια είναι καλύτερη ώστε να την επιλέξουμε και να την επεκτείνουμε με περισσότερα χαρακατηριστικά.

4.1 Απλή Αρχιτεκτονική CNN

Η πιο απλή επιλογή που μπορούμε να κάνουμε είναι η χρήση ενός απλού CNN το οποίο δέχεται ως είσοδο τα 4 τελευταία καρέ του παιχνιδιού προεπεξεργασμένα και έχει στην αρχή συνελικτικά επίπεδα και στο τέλος 1 ή 2 πλήρως συνδεδεμένα επίπεδα πριν το επίπεδο εξόδου. Ένα παράδειγμα μιας τέτοιας αρχιτεκτονικής φαίνεται στο Figure 2.

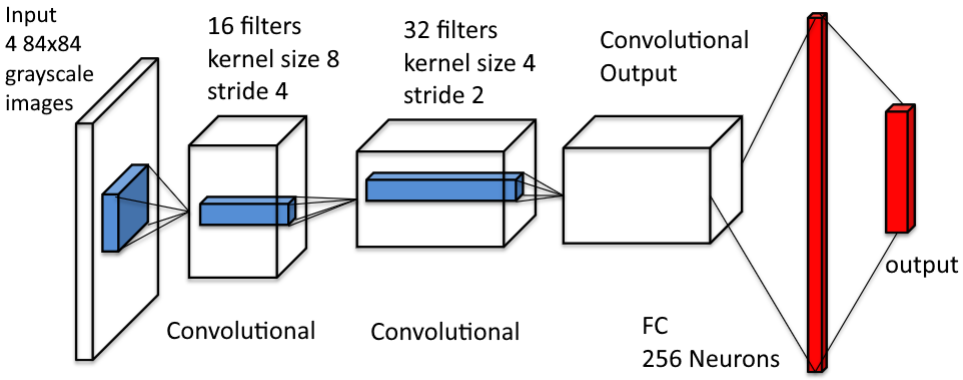

Figure 2: Απλή Αρχιτεκτονική CNN

4.2 Αρχιτεκτονική Dueling Network

Σύμφωνα με την ενότητα 3 του [4] προτείνεται η χρήση ενός Dueling Network έναντι ενός απλού CNN. Ουσιαστικά πρόκειται για μια επέκταση ενός CNN όπου αντί να έχουμε 1 Fully Connected επίπεδο ανάμεσα στα Συνελικτικά επίπεδα και το Output επίπεδο έχουμε 2 παράλληλα Fully Connected επίπεδα, όπου το καθένα από αυτά εξυπηρετεί διαφορετικό σκοπό. Αυτό που προτείνει δηλαδή η αρχιτεκτονική αυτή είναι ο διαχωρισμός της προσέγγισης των Q-Values ανάμεσα σε 2 παράλληλα επίπεδα και συνάθροιση των αποτελεσμάτων των 2 επίπεδων.

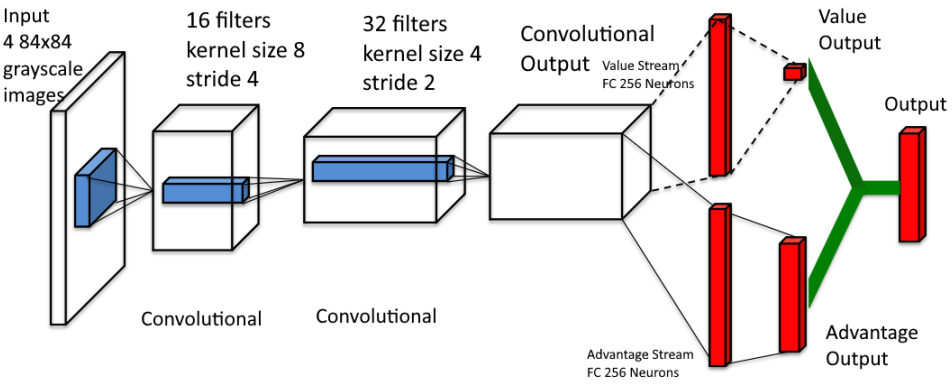

Figure 3: Αρχιτεκτονική Dueling Network

4.2.1 Value Fully Connected Επίπεδο

Αυτό το επίπεδο έχει 1 έξοδο και μας δίνει μια **Αξία** για την κατάσταση στην οποία βρίσκεται ο πράκτορας που παίζει το παιχνίδι.

4.2.2 Advantages Fully Connected Επίπεδο

Αυτό το επίπεδο έχει αριθμό εξόδων όσες είναι και οι πιθανές κινήσεις που μπορεί να κάνει ο πράκτορας (π.χ. στο Space Invaders 6). Μας δίνει εκτιμήσεις για το ποια επιλογή κίνησης είναι καλύτερη με βάση την συγκεκριμένη κατάσταση.

4.2.3 Συνάθροιση των 2 Επιπέδων

Η συνάθροιση των δύο επιπέδων γίνεται με σκοπό την παραγωγή των τελικών Q-Values για κάθε πιθανή κίνηση. Συγκεκριμένα, η τιμή Q για κάθε δράση a σε μια δεδομένη κατάσταση s προκύπτει από τον συνδυασμό της **Αξίας** (Value) της κατάστασης και των **Πλεονεκτημάτων** (Advantages) των ενεργειών. Ο μαθηματικός τύπος για τον υπολογισμό των Q-Values είναι:

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a') \right)$$

όπου:

- $Q(s,a)$ είναι η εκτίμηση της ποιότητας (Q-Value) της κατάστασης s και της δράσης a.
- $V(s)$ είναι η αξία της κατάστασης s από το Value Fully Connected Επίπεδο.
- $A(s,a)$ είναι το πλεονέκτημα της δράσης a στην κατάσταση s από το Advantages Fully Connected Επίπεδο.
- $\mathcal{A}$ είναι το σύνολο των πιθανών ενεργειών (π.χ. 6 για το Space Invaders).
- $\frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a')$ είναι ο μέσος όρος των πλεονεκτημάτων όλων των πιθανών ενεργειών στην κατάσταση s.

Ο λόγος που αφαιρείται ο μέσος όρος των πλεονεκτημάτων είναι για να εξασφαλιστεί ότι η εκτίμηση της αξίας της κατάστασης $V(s)$ δεν επηρεάζεται από την επιλογή της δράσης, παρέχοντας έτσι μια σταθερή βάση για την αξιολόγηση των πλεονεκτημάτων των ενεργειών.

Με αυτόν τον τρόπο, η αρχιτεκτονική Dueling Network επιτρέπει στο δίκτυο να διαχωρίζει την εκτίμηση της συνολικής αξίας μιας κατάστασης από τις διαφορές μεταξύ των αξιών των επιμέρους ενεργειών, βελτιώνοντας την απόδοση και τη σταθερότητα της εκπαίδευσης σε διάφορα προβλήματα ενισχυτικής μάθησης.

5 Αλγόριθμος Deep Q-Learning

Ο Αλγόριθμος Q-Learning είναι ένας αλγόριθμος ενισχυτικής μάθησης που χρησιμοποιείται για να βρει την βέλτιστη πολιτική επιλογής ενεργειών για μια δεδομένη πεπερασμένη διαδικασία Markov (MDP). Χρησιμοποιεί την εξίσωση Bellman για να ενημερώσει τις τιμές Q, οι οποίες αντιπροσωπεύουν τις αναμενόμενες μελλοντικές ανταμοιβές για τη λήψη μιας συγκεκριμένης δράσης σε μια δεδομένη κατάσταση. Ο Deep Q-Learning χρησιμοποιεί ένα νευρωνικό δίκτυο ως estimator των τιμών Q.

5.1 Σημαντικά Στοιχεία του Αλγορίθμου

5.1.1 Αναπαράσταση Q-Values

Οι τιμές Q προσεγγίζονται χρησιμοποιώντας είτε το CNN Deep Q-Network είτε το Dueling Deep Q-Network, το οποίο είναι ένα νευρωνικό δίκτυο που προβλέπει τιμές Q για κάθε ενέργεια δεδομένης μιας κατάστασης.

5.1.2 Επιλογή Ενεργειών

Ο πράκτορας επιλέγει ενέργειες χρησιμοποιώντας μια πολιτική epsilon-greedy. Με πιθανότητα epsilon, επιλέγει μια τυχαία ενέργεια (εξερεύνηση), και με πιθανότητα 1-epsilon, επιλέγει την ενέργεια με την υψηλότερη προβλεπόμενη τιμή Q (εκμετάλλευση).

5.1.3 Αποθήκευση Εμπειρίας (Μνήμη)

Ο πράκτορας αποθηκεύει τις x πιο πρόσφατες εμπειρίες του (state, action, reward, next state, done) σε έναν replay buffer χρησιμοποιώντας τη μέθοδο store_transition.

1. Απλή μνήμη
    - Δομή Ουράς
    - Συγκεκριμένο Μέγεθος
    - Τυχαία δειγματοληψία minibatch για το training
2. Prioritized Experience Replay
    - Δομή Sum Tree
    - Συγκεκριμένο Μέγεθος
    - Δειγματοληψία minibatch για το training, μέσω υπολογισμού των προτεραιοτήτων των αποθηκευμένων εμπειριών.
    - κάθε εμπειρία έχει το δικό της σφάλμα χρονικής διαφοράς (TD error) το οποίο χρησιμοποιείται για τον υπολογισμό της προτεραιότητάς της.
    - εμπειρίες με υψηλό TD error έχουν να προσφέρουν περισσότερα στην διαδικασία μάθησης.

5.1.4 Ενημέρωση Q-Value

Οι τιμές Q ενημερώνονται κατά τη διάρκεια της διαδικασίας εκπαίδευσης, η οποία συμβαίνει στη μέθοδο train.

5.1.5 Ανταμοιβές

Για τις ανταμοιβές που θα λαμβάνει ο agent από το περιβάλλον υπάρχουν διάφορες τεχνικές.

1. Προκαθορισμένες Ανταμοιβές Περιβάλλοντος
   - Η ανταμοιβή μιας κατάστασης είναι ίση με το score που κέρδισε ο παίκτης.
     Πιθανές Ανταμοιβές: 0, 5, 10, 15, 20, 25, 30, 200
   - Δεν υπάρχουν αρνητικές ανταμοιβές
2. Κανονικοποιημένες Ανταμοιβές
   - Η θετική ανταμοιβή μιας κατάστασης είναι ίση με το score που κέρδισε ο παίκτης, ενώ υπάρχουν και αρνητικές ανταμοιβές στην περίπτωση που ο agent χάσει μια ζωή (-5) και στην περίπτωση που ο agent χάσει την τελευταία ζωή του (-10).
     Πιθανές Ανταμοιβές: -10, -5, 0, 5, 10, 15, 20, 25, 30, 200
   - Κανονικοποιούμε τις τιμές των ανταμοιβών στα διαστήματα:
     Αρνητικές Ανταμοιβές: [-1, 0)
     Ανταμοιβή 0: 0
     Θετικές Ανταμοιβές (εκτός 200): (0, 1]
     Θετικές Ανταμοιβή 200: 1

Πληροφοριακά να σημειώσουμε, ότι δοκιμάστηκαν και άλλες προσεγγίσεις, όπως το να δίνουμε, πέρα από τα άλλα rewards, μια πολύ μικρή αρνητική ή θετική ανταμοιβή όσο μένει ο Agent ζωντανός.

Η λογική πίσω από την επιλογή **θετικής** ανταμοιβής είναι πως ο agent θα προσπαθεί να μείνει όλο και περισσότερη ώρα ζωντανός με σκοπό να μεγιστοποιήσει την συνολική ανταμοιβή. Όμως αυτό αποδείχτηκε κακή επιλογή γιατί οι agents που εκπαιδεύτηκαν με αυτόν τον τρόπο έμεναν στο ίδιο σημείο και πυροβολούσαν. Θεωρητικά κάναν σωστά αυτό που τους λέγαμε μέσω των ανταμοιβών όμως δεν παίζαν σωστά το παιχνίδι.

Οπότε κάποιος θα μπορούσε να υποθέσει ότι η αντίθετη επιλογή, δηλαδή μια μικρή **αρνητική** ανταμοιβή για όσο ο agent δεν κερδίζει ανταμοιβές, θα ήταν πιο αποτελεσματική. Στην πραγματικότητα ούτε αυτό φάνηκε να βοηθάει διότι οι agents άρχισαν να κινούνται με υπερβολικά ριψοκίνδυνο τρόπο, στην προσπάθεια τους να μειώσουν τις αρνητικές ανταμοιβές χωρίς να αποφεύουν τα πυρά των αντιπάλων.

5.2 Αναλυτική Παρουσίαση Υλοποίησης

5.2.1 Αρχικοποίηση Q-values

Αρχικοποιείται το κύριο Q-network (main_network) και το target Q-network (target_network), και τα δύο είναι στιγμιότυπα της κλάσης εκάστοτε αρχιτεκτονικής που χρησιμοποιούμε.

5.2.2 Επιλογή Ενέργειας

Για την τρέχουσα κατάσταση, επιλέγεται μια ενέργεια χρησιμοποιώντας την πολιτική epsilon-greedy (epsilon_greedy μέθοδος).

5.2.3 Εκτέλεση Ενέργειας και Παρατήρηση

Εκτελείται η ενέργεια στο περιβάλλον για να ληφθεί η επόμενη κατάσταση και η ανταμοιβή. Αποθηκεύεται η εμπειρία (state, action, reward, next state, done) στον replay buffer.

5.2.4 Δειγματοληψία Mini-Batch

Δειγματοληψία ενός mini-batch εμπειριών από τον replay buffer, είτε με τυχαίο τρόπο είτε με την χρήση μιας δομής Prioritized Experience Replay.

5.2.5 Υπολογισμός Target Q-Value

Για κάθε εμπειρία στο mini-batch, υπολογίζεται η target Q-value χρησιμοποιώντας την εξίσωση Bellman:

$$target_Q = reward + \gamma max Q'(next_state, \alpha)$$

Εάν το επεισόδιο έχει τελειώσει, η target Q-value είναι απλά η ανταμοιβή:

$$target_Q = reward$$

5.2.6 Ενημέρωση Q-Network

Υπολογίζεται η απώλεια μεταξύ της τρέχουσας Q-value και της target Q-value. Η Q-value για το τρέχον ζεύγος κατάστασης-δράσης είναι:

$$Q(state, action)$$

Πραγματοποιείται ένα βήμα gradient descent για να ελαχιστοποιηθεί αυτή η απώλεια. Αυτό περιλαμβάνει τον υπολογισμό των gradient και την ενημέρωση των βαρών του κύριου Q-network.

5.2.7 Ενημέρωση Target Network

Περιοδικά, το target Q-network ενημερώνεται αντιγράφοντας τα βάρη από το κύριο Q-network στο target Q-network μέσω της μεθόδου update_target_network. Ο λόγος που χρησιμοποιούμε δύο δίκτυα, ένα βασικό και ένα δίκτυο στόχο, είναι για να σταθεροποιήσουμε τον στόχο κατά τη διάρκεια της εκπαίδευσης. Όταν ανανεώνεται το target network, ο στόχος για την ενημέρωση των Q-τιμών παραμένει σταθερός για ένα ορισμένο αριθμό βημάτων, επιτρέποντας μια πιο σταθερή και αποτελεσματική προσέγγιση της αντικειμενικής συνάρτησης μέσω gradient descent. Με αυτόν τον τρόπο, αποφεύγουμε τις ταλαντώσεις και την αστάθεια που

μπορεί να προκύψουν από τη συνεχή αλλαγή των Q-τιμών στόχων κατά τη διάρκεια της εκπαίδευσης.

5.2.8 Epsilon Decay

Πρόκειται για μια συνάρτηση η οποία μειώνει την τιμή epsilon ελάχιστα για να μειωθεί ο ρυθμός εξερεύνησης με την πάροδο του χρόνου (decay_epsilon μέθοδος). Η συνάρτηση μπορεί να καλείται είτε μετά από κάθε βήμα είτε μετά από κάθε επεισόδιο ανάλογα την επιλογή που θα κάνουμε στο εκάστωτε πείραμα.

6 Πειράματα και Αποτελέσματα

6.1 Πείραμα 0

Εδώ θα επιλέγουμε εντελώς τυχαία την κάθε επιλογή του παίκτη για να δούμε πως αποδίδει ένα μοντέλο το οποίο δεν έχει εκπαιδευτεί καθόλου. Τα αποτελέσματα:

Min Score	Max Score	Median Score	Average Score
0	185	50	62.85

6.2 CNN vs Dueling Network

Αρχικά θα τρέξουμε 2 πειράματα με ίδιες παραμέτρους αλλά με διαφορετική αρχιτεκτονική δικτύου για να δούμε ποια είναι καλύτερη για να την επιλέξουμε για τα επόμενα πειράματα. Και για τα 2 πειράματα θα δίνουμε τις προεπιλεγμένες ανταμοιβές του περιβάλλοντος χωρίς καμία κανονικοποίηση. Για όλα τα πειράματα θα εκπαιδεύσουμε το δίκτυο σε περιβάλλον με hard δυσκολία.

6.3 Πείραμα 1

Υπερ-παράμετροι:

- Αρχιτεκτονική: CNN
- Επεισόδια = 1000
- Μέγεθος Μνήμης = 10000 (Τυχαία Δειγματοληψία)
- batch_size = 8
- $\gamma = 0.99$
- ε_start = 1.0
- ε_min = 0.1
- ε_decay = 0.995 (ανά επεισόδιο)

- update_rate = 1000
- learning_rate = 0.00025
- Ανταμοιβές: Προκαθορισμένες Ανταμοιβές Περιβάλλοντος
- Δυσκολία: Hard

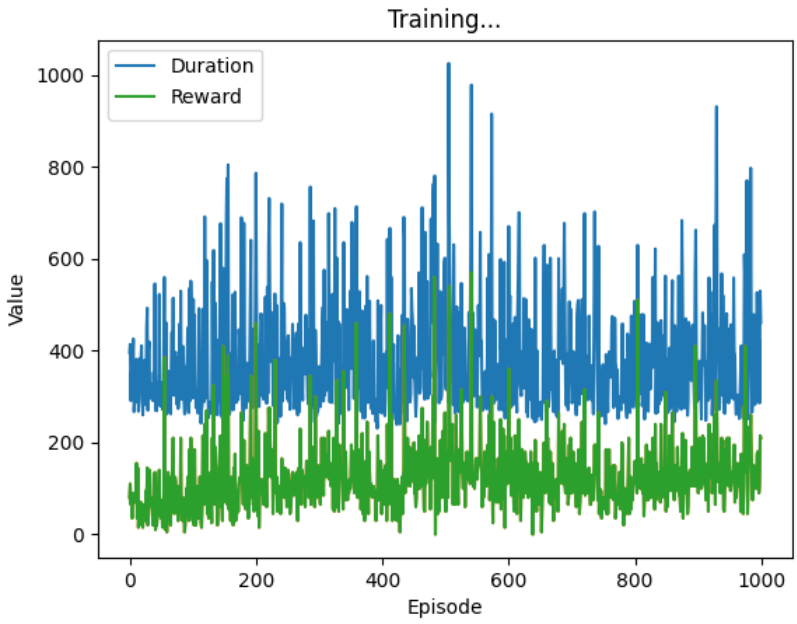

Figure 4: (Πείραμα 1) Score και διάρκεια ζωής agent ανά επεισόδιο εκπαίδευσης

6.4 Πείραμα 2

Υπερ-παράμετροι:

- Αρχιτεκτονική: Dueling Network
- Επεισόδια = 1000
- Μέγεθος Μνήμης = 10000 (Τυχαία Δειγματοληψία)
- batch_size = 8
- $\gamma = 0.99$
- ε_start = 1.0

Table 1: Απόδοση μοντέλων Πειράματος 1 (CNN) για 100 επεισόδια (Hard δυσκολία)

Episodes Trained	Min Score	Max Score	Median Score	Average Score
100	20	495	95	121.05
200	90	430	165	175.65
300	50	415	127.5	141.15
400	60	450	120	143.65
500	30	405	90	105.45
600	5	355	75	103.7
700	65	410	135	166.35
800	30	210	90	94.1
900	50	415	120	131.2
1000	105	550	157.5	186.95

- ε_min = 0.1
- ε_decay = 0.995 (ανά επεισόδιο)
- update_rate = 1000
- learning_rate = 0.00025
- Ανταμοιβές: Προκαθορισμένες Ανταμοιβές Περιβάλλοντος
- Δυσκολία: Hard

Table 2: Απόδοση μοντέλων Πειράματος 2 (Dueling Network) για 100 επεισόδια (Hard δυσκολία)

Episodes Trained	Min Score	Max Score	Median Score	Average Score
100	75	210	120	125.2
200	45	470	135	138.15
300	80	430	155	160.8
400	120	490	270	282.75
500	90	590	260	263.45
600	125	645	215	262.35
700	65	745	210	252.4
800	45	745	180	210.9
900	25	370	60	97.45
1000	90	470	210	214.2

Όπως παρατηρούμε από τους πίνακες 1 και 2, η αρχιτεκτονική Dueling Network έχει πολύ καλύτερα αποτελέσματα από την αρχιτεκτονική CNN στα 8 από τα 10 μοντέλα. Οπότε είναι ξεκάθαρο πως θα χρησιμοποιήσουμε αυτήν για τα επόμενα πειράματα. Χρησιμοποιούμε τον διάμεσο για να μην επηρεαζόμαστε από

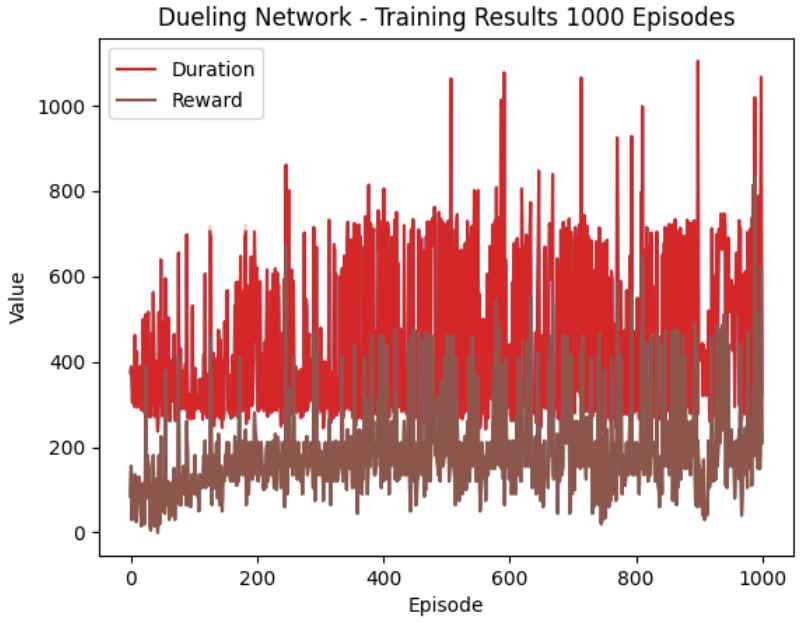

Figure 5: (Πείραμα 2) Score και διάρκεια ζωής agent ανά επεισόδιο εκπαίδευσης

ακραίες τιμές, δηλαδή παρτίδες όπου ο agent πήρε είτε πολύ υψηλό είτε πολύ χαμηλό score. Βέβαια να σημειωθεί πως οι τιμές της διάμεσου και του μέσου όρου είναι αρκετά κοντά τις περισσότερες φορές πράγμα που σημαίνει ότι δεν έχουμε μεγάλη απόκλιση.

Table 3: Σύγκριση CNN και Dueling Network Median Scores

Episodes Trained	Median Scores ανά Μοντέλο									
	100	200	300	400	500	600	700	800	900	1000
CNN	95	165	127.5	120	90	75	135	90	120	157.5
Dueling	120	135	155	270	260	215	210	180	60	210

6.5 Από τα δύσκολα στα εύκολα

Κάτι που είναι επίσης ενδιαφέρον να μελετήσουμε είναι το αν ένας agent εκπαιδευμένος στην hard δυσκολία μπορεί να αποδώσει το ίδιο ή και ακόμα καλύτερα στην easy δυσκολία.

Σε γενικές γραμμές, συγκρίνοντας τα αποτελέσματα των πινάκων 4 και 5 με αυτά των αρχικών πειραμάτων, βλέπουμε και για τις 2 υλοποιήσεις καλύτερα αποτελέσματα. Φυσικά αυτό δεν είναι πάντα ο κανόνας καθώς το ότι ο agent έμαθε να παίζει σε ένα πιο δύσκολο περιβάλλον για εμάς τους ανθρώπους δεν

Table 4: Απόδοση μοντέλων Πειράματος 1 (CNN) για 100 επεισόδια (Easy δυσκολία)

Episodes Trained	Min Score	Max Score	Median Score	Average Score
100	50	770	180	194.3
200	75	740	180	196.05
300	50	465	155	170.65
400	50	635	185	208.7
500	5	585	125	144.75
600	155	260	155	162.25
700	75	530	180	201.05
800	35	645	135	162.65
900	30	605	140	172.6
1000	15	560	110	154.5

Table 5: Απόδοση μοντέλων Πειράματος 2 (Dueling Network) για 100 επεισόδια (Easy δυσκολία)

Episodes Trained	Min Score	Max Score	Median Score	Average Score
100	30	280	90	95.05
200	110	525	142.5	167.15
300	20	460	100	122.25
400	65	835	292.5	333.15
500	75	595	260	279.25
600	135	1065	225	282.55
700	120	755	155	237.45
800	35	695	95	156.7
900	30	710	210	228.7
1000	80	845	315	347.8

τον κάνει άμεσα και καλύτερο στο περιβάλλον χαμηλότερης δυσκολίας. Είναι σαν ένας άνθρωπος να έχει εκπαιδευτεί μόνο στον Διαφορικό Λογισμό και να καλείται ξαφνικά να λύσει για πρώτη φορά στην ζωή του ένα απλό πρόβλημα Διακριτών Μαθηματικών. Είναι πιθανό να δυσκολευτεί λόγω της φύσης του νέου περιβάλλοντος, όσο και αν αυτό έχει κάποια κοινά χαρακτηριστικά με το περιβάλλον στο οποίο εκπαιδεύτηκε (π.χ. αριθμοί και στα 2 προβλήματα). Από την άλλη μπορεί με κάποιον τρόπο τα ερεθίσματα που έλαβε στο πιο δύσκολο περιβάλλον να τον έχουν κάνει καλύτερο με έναν πιο γενικευμένο τρόπο και να μπορεί να ανταπεξέλθει και στο πιο εύκολο περιβάλλον.

6.6 Πείραμα 3

Παίρνοντας έμπνευση από το [2] ορίζουμε τις υπερπαραμέτρους:

- Αρχιτεκτονική: Dueling Network

- Επεισόδια = 2000
- Μέγεθος Μνήμης = 20000 (Prioritized Experience Replay)
- batch_size = 32 (train ανά 4 steps)
- $\gamma = 0.99$
- ε_start = 1.0
- ε_min = 0.1
- ε_decay = 0.99 (ανά επεισόδιο)
- update_rate = 10000
- learning_rate = 0.00025
- Ανταμοιβές: Κανονικοποιημένες Ανταμοιβές
- Δυσκολία: Hard

Βελτιώσεις: Έχουμε αναπτύξει μια Prioritized Experience Replay μνήμη. Κάνουμε train ανά 4 βήματα. Κανονικοποιήσαμε τις ανταμοιβές στο [-1, 1].

Όπως παρατηρούμε στο Figure 6 βλέπουμε μια αργή αλλά σταδιακή άνοδο στο γράφημα και άρα μια συνεχόμενη βελτίωση κατά την διάρκεια του training οπότε μπορούμε να πούμε πως η προσέγγιση μας λειτουργεί και σίγουρα θα μπορούσε να φέρει και καλύτερα αποτελέσματα για περισσότερα επεισόδια εκπαίδευσης.

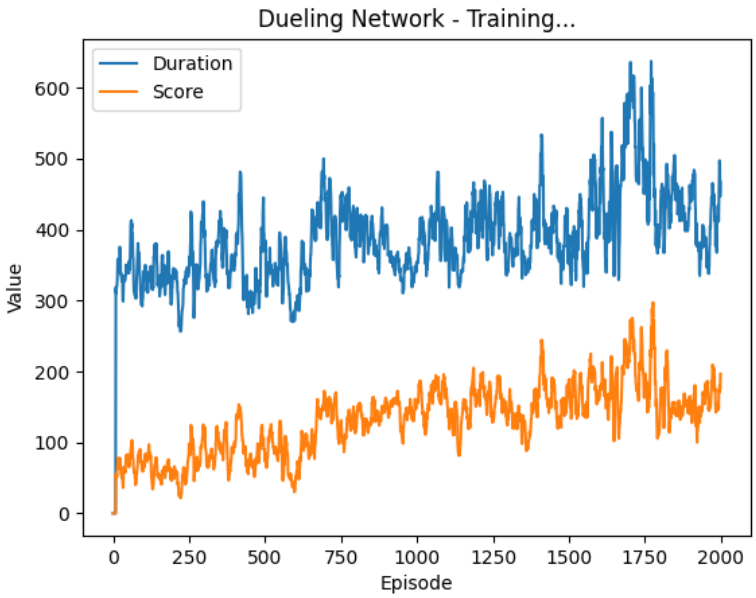

Figure 6: (Πείραμα 3) Μέσο Score και μέση διάρκεια ζωής agent ανά 10 επεισόδια εκπαίδευσης

6.7 Πείραμα 4

Αυτό το πείραμα προέκυψε από διάφορους πειραματισμούς που έκανα, τόσο με τις υπερ-παραμέτρους όσο και με την αρχιτεκτονική του δικτύου αλλά και τον ορισμό των ανταμοιβών υπό ένα νέο πρίσμα.

- Αρχιτεκτονική: Dueling Network (Διπλάσια Φίλτρα, Νέο Συνελικτικό Επίπεδο και Διπλάσιοι Νευρώνες στα FC Επίπεδα)
- Επεισόδια = 1000
- Μέγεθος Μνήμης = 100000 (Prioritized Experience Replay)
- batch_size = 8 (train ανά 4 steps)
- $\gamma = 0.99$
- ε_start = 1.0
- ε_min = 0.1
- ε_decay = 0.99999 (ανά βήμα)
- update_rate = 10000

Table 6: Απόδοση μοντέλων Πειράματος 3 για 100 επεισόδια (Hard δυσκολία)

Episodes Trained	Min Score	Max Score	Median Score	Average Score
100	0	160	57.5	61.8
200	0	310	35	42.6
300	5	160	20	34.7
400	20	415	82.5	92.95
500	55	255	120	122
600	5	210	75	75.2
700	105	515	197.5	209.8
800	20	215	135	127.45
900	90	410	155	155.25
1000	105	240	180	173.05
1100	75	410	155	157.15
1200	35	425	155	150.6
1300	30	415	180	191
1400	35	425	190	177.9
1500	40	610	152.5	167.25
1600	35	460	180	184.5
1700	30	535	210	220.1
1800	40	545	155	207.55
1900	10	430	132.5	131.6
2000	90	410	210	204.5

- learning_rate = 0.0001
- Ανταμοιβές: -1 αν χάσει ζωή, 1 αν εξολοθρεύσει αντίπαλο, 0 σε κάθε άλλη περίπτωση
- Δυσκολία: Hard

Table 7: Απόδοση μοντέλων Πειράματος 4 για 100 επεισόδια (Hard δυσκολία)

Episodes Trained	Min Score	Max Score	Median Score	Average Score
100	5	345	55	79.5
200	60	410	127.5	137.15
300	240	265	240	247.95
400	30	420	135	141.15
500	60	165	60	81.75
600	45	395	60	69.25
700	95	790	245	254.1
800	80	565	230	235.25
900	75	490	150	161.55
1000	90	510	180	191.65

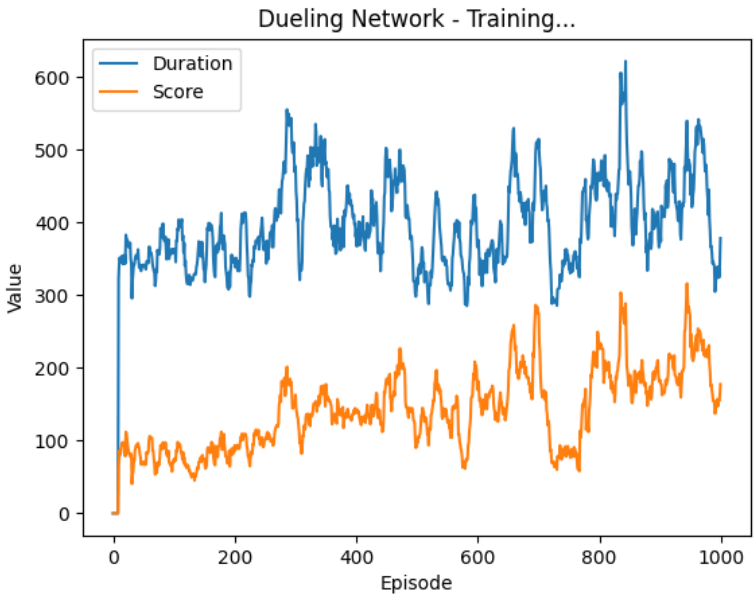

Figure 7: (Πείραμα 4) Μέσο Score και μέση διάρκεια ζωής agent ανά 10 επεισόδια εκπαίδευσης

Αντί να προσπαθούμε να μεγιστοποιήσουμε το σκορ με διάφορες μεθόδους ανταμοιβών, απλοποιούμε το πρόβλημα λέγοντας πως ο Agent θα δέχεται μια θετική ανταμοιβή ίση με 1 για κάθε αντίπαλο που εξολοθρεύει, μια αρνητική ανταμοιβή ίση με -1 κάθε φορά που χάνει μια ζωή και 0 για κάθε άλλη περίπτωση. Έτσι το πρόβλημα πλέον δεν είναι το να μεγιστοποιήσει το σκορ αλλά να εξολοθρεύσει όσο το δυνατόν περισσότερους αντιπάλους μπορεί. Έμεσσα αυτό επιφέρει και μεγαλύτερο σκορ.

Ιδιαίτερο ενδιαφέρον έχει το μοντέλο που προέκυψε μετά από 300 επεισόδια, καθώς όπως φαίνεται και από το γράφημα αλλά και από τον πίνακα, καταφαίρνει ένα συγκεκριμένο σκορ περίπου 240 στα περισσότερα επεισόδια. Θα μπορούσαμε να πούμε ότι είναι ένα αρκετά robust μοντέλο. Το ειρωνικό είναι πως αυτό το μοντέλο παίζει σαν αυτά του Πειράματος 3, δηλαδή μένει περισσότερη ώρα στην αριστερή μεριά της οθόνης και περιμένει τους αντιπάλους.

Μετά τα 300 επεισόδια εκπαίδευσης βλέπουμε στο Figure 9 ότι το loss δεν μειώνεται (αντίθετα αρχίζει και ανεβαίνει) οπότε αναμενόμενο τα επόμενα μοντέλα που παράχθηκαν να μην είναι καλύτερα.

Παρόλα αυτά, το μοντέλο μετά τα 700 επεισόδια έχει μια εξαιρετική επίδοση καθώς σε πολλές περιπτώσεις καταφαίρνει να εξολοθρεύσει περίπου τους μισούς αντιπάλους της οθόνης.

Μετά τα 700 επεισόδια παρατηρούμε άνοδο του loss και πτώση των rewards, της διάρκειας και του σκορ, μέχρι να έρθει ξανά μια ύψωση.

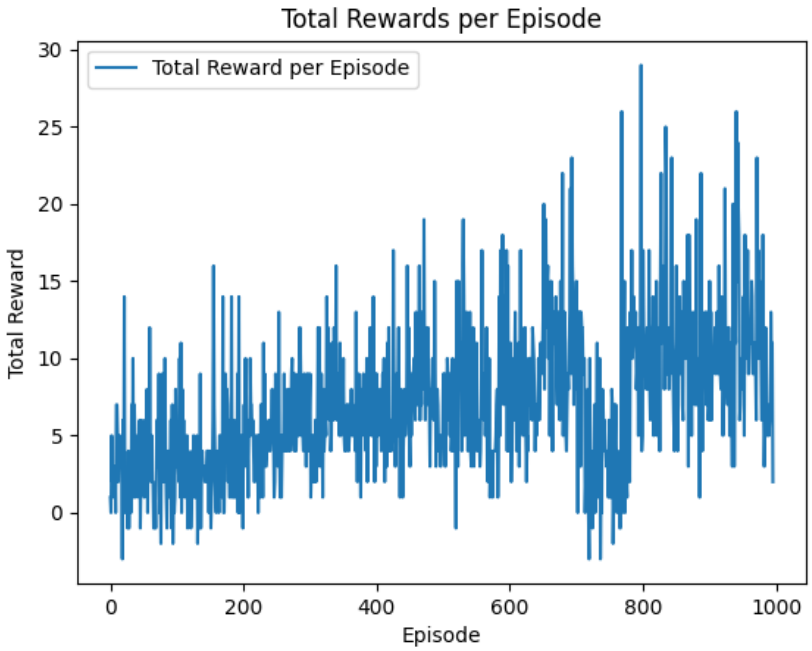

Figure 8: (Πείραμα 4) Συνολική Ανταμοιβή ανά επεισόδιο εκπαίδευσης

Σε γενικές γραμμές το μοντέλο βελτιώνεται όσο περνάνε τα επεισόδια. Αυτό επιβεβαιώνεται από τα Figure 7, 8 και 9. Στα 2 πρώτα βλέπουμε γενικά ανοδική πορεία και στο τελευταίο γενικά βλέπουμε κάθοδο.

Κάτι επίσης πολύ ενδιαφέρον είναι να οπτικοποιήσουμε το ποια σημεία της εικόνας εισόδου του νευρωνικού δικτύου μας δίνουν περισσότερη πληροφορία για την λήψη της απόφασης. Για αυτό κάνουμε χρήση των Saliency Maps και όπως βλέπουμε στα Figure 10 και 11, τα σημεία με πιο έντονη απόχρωση παίζουν μεγαλύτερο ρόλο στην λήψη απόφασης της επόμενης κίνησης. Τέτοιες περιοχές είναι κοντά στον παίκτη, στις σφαίρες του και προς τα σημεία όπου εξολοθρεύτηκε πρόσφατα κάποιος αντίπαλος.

Να σημειωθεί πως τα Saliency Maps σε αυτήν την φάση έχουν ακόμα αρκετό θόρυβο. Το αναμενόμενο είναι με την πάροδο της εκπαίδευσης να εστιάζουν σε πολύ συγκεκριμένα μέρη της οθόνης για την λήψη απόφασης.

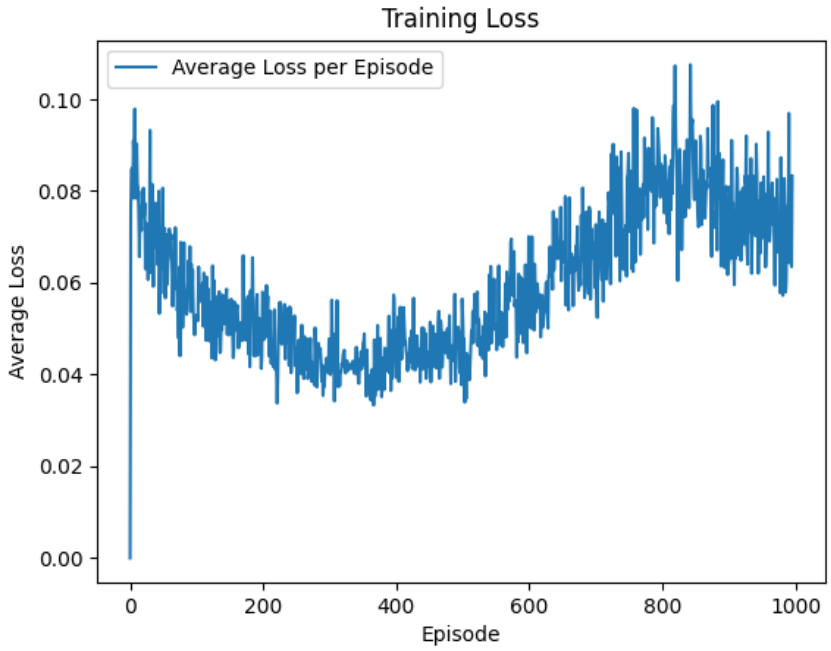

Figure 9: (Πείραμα 4) Μέσο Loss ανά επεισόδιο εκπαίδευσης

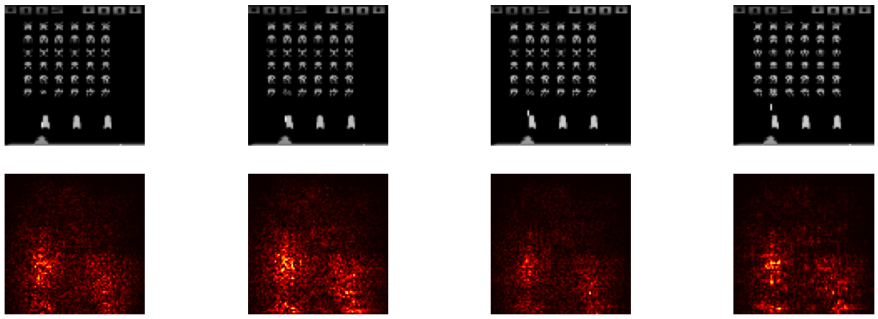

Figure 10: Saliency Maps για μια κατάσταση του μοντέλου των 700 επεισόδιων του πειράματος 4

7 Τελικός Σχολιασμός Αποτελεσμάτων

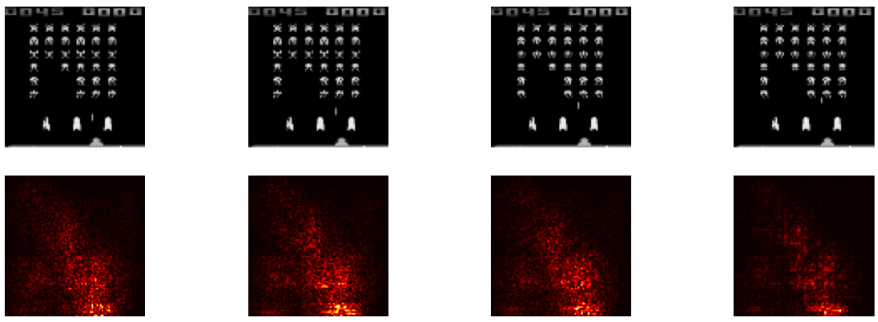

Figure 11: Saliency Maps για μια κατάσταση του μοντέλου των 700 επεισόδιων του πειράματος 4

- Η αρχιτεκτονική Dueling Network δίνει καλύτερα αποτελέσματα και πιο γρήγορα από την αρχιτεκτονική ενός απλού CNN μέσω της εκπαίδευσης με τον αλγόριθμο Deep Q Learning.
- Μια Prioritized Experience Replay Μνήμη βοηθάει τον αλγόριθμο Deep Q Learning να ανακτήσει τις καταστάσεις με μεγαλύτερο TD error και άρα να επωφεληθεί από αυτές για την εκπαίδευση του νευρωνικού.
- Ο Αλγόριθμος Deep Q Learning αλλά και τα νευρωνικά δίκτυα είναι αρκετά ασταθή και χρειάζονται σημαντικό χειροκίνητο fine-tuning για να επιτευχθεί καλύτερη εκπαίδευση για κάθε task.
- Χρησιμοποιώντας Saliency Maps μπορούμε να δούμε ποια σημεία της εικόνας εισόδου παρέχουν την σημαντικότερη πληροφορία για την λήψη απόφασης μέσω του νευρωνικού.

8 Οδηγίες Εκτέλεσης Κώδικα

1. Καταρχάς χρειαζόμαστε ένα περιβάλλον Python σε έκδοση 3.11 (π.χ. PyCharm) για να φορτώσουμε τα αρχεία του κώδικα και των φακέλων.
2. Έπειτα κάνουμε install τα requirements του project μέσω της εντολής:

   `pip install -r requirements.txt`

3. Τρέχουμε όποιο από τα `.py` αρχεία θέλουμε για να δούμε ένα αποθηκευμένο μοντέλο από κάθε πείραμα να παίζει το παιχνίδι.
   (a) Στο αρχείο `saliency_maps.py` φορτώνουμε ένα μοντέλο του Πειράματος 4 και οπτικοποιούμε τα Saliency Maps του.
   (b) Μπορούμε να αλλάξουμε το ποιο μοντέλο θέλουμε να ελέγξουμε αλλάζοντας τις παραμέτρους μέσα στο πρόγραμμα.

Αναφορές

[1] M. G. Bellemare et al. “The Arcade Learning Environment: An Evaluation Platform for General Agents”. In: *Journal of Artificial Intelligence Research* 47 (June 2013), pp. 253–279.

[2] H. V. Hasselt, Arthur Guez, and David Silver. “Deep Reinforcement Learning with Double Q-Learning”. In: *AAAI Conference on Artificial Intelligence*. 2015. URL: https://api.semanticscholar.org/CorpusID:6208256.

[3] Mark Towers et al. “Gymnasium”. In: (Mar. 2023).

[4] Ziyun Wang et al. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *International Conference on Machine Learning*. 2015. URL: https://api.semanticscholar.org/CorpusID:5389801.