ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Νευρωνικά Δίκτυα και Ευφυή Υπολογιστικά Συστήματα

9ο Εξαμηνο ΗΜΜΥ

Συνεργάτες: Γιαννιός Γεώργιος-Ταξιάρχης, Α.Μ.: 03116156
Μπέτζελος Γιώργος, Α.Μ.: 03117442
Μπέτζελος Χρήστος, Α.Μ.: 03116067

Ομάδα: 70

Άσκηση 4: Παίζοντας Atari με Βαθιά Ενισχυτική Μάθηση

Περίληψη - Στόχος:

Στόχος της άσκησης είναι η βελτιστοποίηση αλγορίθμων βαθιάς ενισχυτικής μάθησης, ώστε ο πράκτορας (2) παιχνιδιών Atari να επιτυγχάνει υψηλό σκορ. Αρχικά εξετάσαμε την επίδοση ενός πράκτορα (χωρίς μάθηση) παράλληλα με την επιδοση κάποιου ανθρώπου και καταγράψαμε τα αντίστοιχα σκορ. Στη συνέχεια εκπαιδεύσαμε πράκτορες με τεχνικές ενισχυτικής μάθησης και εξετάσαμε (συγκριτικά) την επίδραση τόσο διαφορετικών environments όσο και διαφορετικών αλγορίθμων. Το υψηλότερο σκορ (4396.0 ± 963.07) καταγράφηκε στο περιβάλλον Deterministic-v4 με τον αλγόριθμο DQN, μετά από κατάλληλη βελτιστοποίηση.

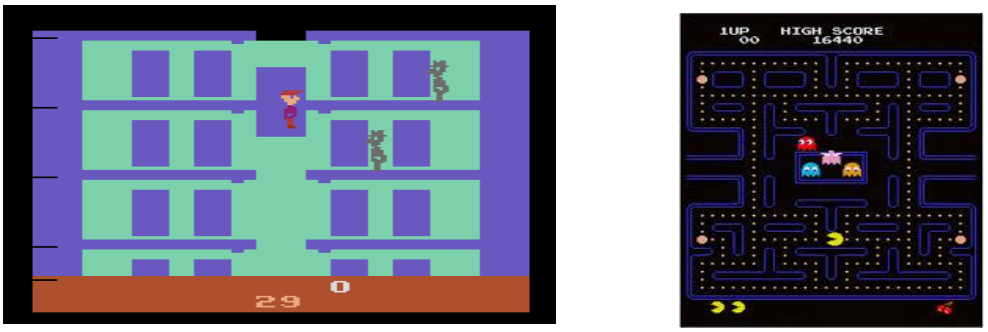
Περιγραφή Παιχνιδιών:

Παιχνίδι 1: Elevator Action

Στο συγκεκριμένο παιχνίδι, σκοπός του πράκτορα είναι να συγκεντρώσει πόντους, σκοτώνοντας αντιπάλους που εμφανίζονται σε διάφορους ορόφους. Οι δυνατές κινήσεις είναι: άλμα, σκύψιμο, βολή, χειρισμός ασανσερ (πάνω-κάτω μόνο όσο είναι μέσα σε αυτό), πλοήγηση δεξιά- αριστερά σε έναν όροφο ή στην οροφή του κτιρίου. Κατα την αρχικοποίηση του παιχνιδιού, ο πράκτορας βρίσκεται στην οροφή του κτιρίου, όπου δεν εμφανίζεται κάποιος αντίπαλος.

Παιχνίδι 2: MsPacman

Το συγκεκριμένο παιχνίδι είναι παρόμοιο με το κλασσικό Pac-Man. Ο πράκτορας κερδίζει πόντους τρώγοντας “καρπούς” και αποφεύγοντας κάποια τέρατα. Πέρα από τους κλασικούς καρπούς, υπάρχουν και πιο ειδικοί οι οποίοι δίνουν την δυνατότητα στο παίκτη να φάει ακόμα και τα τέρατα και να κερδίσει περισσότερους πόντους (blue mode). Κατά τη διάρκεια του παιχνιδιού εμφανίζονται και κάποια φρούτα τα οποία προσφέρουν διπλάσιους πόντους. Με το πέρασμα του χρόνου αυξάνεται και η ταχύτητα του gameplay, ενώ μειώνεται και η διάρκεια του blue mode.



Εικόνα 1: Στιγμιότυπο του παιχνιδιού “Elevator Action” (αριστερά) και του “Pacman” (δεξιά)

Πράκτορας χωρίς μάθηση

Αρχικά ορίζουμε έναν πράκτορα χωρίς μάθηση. Το score (μέση ανταμοιβή) που επιτυγχάνεται στο test_env για το παιχνίδι ElevatorAction είναι 0.0 (+/-0.0) ενώ για το MsPacman 112.0 (+/-67.35). Το βίντεο αυτό, παραδίδεται στο αρχείο videos/MsPacman_Random_Agent.mp4

Σύγκριση Επίδοσης Αλγορίθμων

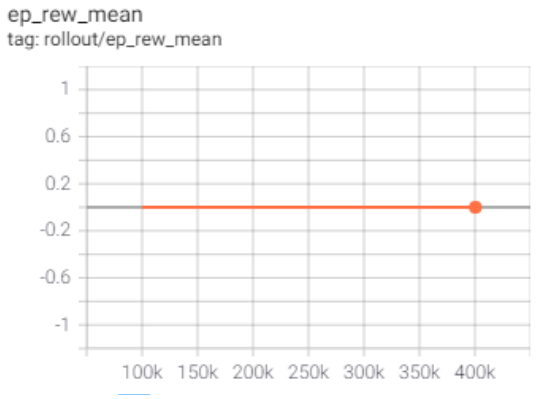
Το περιβάλλον εργασίας (Kaggle) στο οποίο υλοποιείται ο πράκτορας, ορίζει κάποιους περιορισμούς τόσο σε υπολογιστική ισχύ όσο και σε χρόνους εκτέλεσης. Για το λόγο αυτό, κρίνεται αναγκαίο να εξετάσουμε από πριν (συγκριτικά) τη συμπεριφορά διαφόρων μοντέλων. Χρησιμοποιώντας λοιπόν το *TensorBoard*, λαμβάνουμε κάποιες εκτιμήσεις για τη μετρική που μας ενδιαφέρει. Η μετρική αυτή είναι η μέση ανταμοιβή ή αλλιώς score.

Οι αλγόριθμοι ενισχυτικής μάθησης που εξετάζουμε είναι τρεις: A2C (actor critic), DQN (deep Q learning) και PPO (policy optimization) ενώ τα τα διαφορετικά περιβάλλοντα συνοψίζονται ακολούθως:

- Deterministic-v4: Ίδια ενέργεια για 4 frames (Χωρίς στοχαστικότητα - ο πράκτορας θα κάνει την ίδια κίνηση)
- NoFrameskip-v4: Η ενέργεια αποφασίζεται για κάθε frame (Χωρίς στοχαστικότητα - ο πράκτορας θα κάνει την ίδια κίνηση)
- Empty-v4: Η ενέργεια επιλέγεται τυχαία κάθε 2 ή 4 frames.

Παιχνίδι 1: Elevator Action

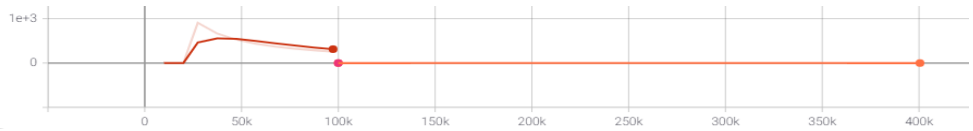
Στο συγκεκριμένο παιχνίδι βρεθήκαμε αντιμέτωποι με το πρόβλημα "*cold start*". Ξεκινώντας λοιπόν απο ντετερμινιστικό περιβάλλον και με αλγόριθμο DQN παρατηρήσαμε ότι ο πράκτορας δεν εκπαιδεύεται ακόμα και για μεγάλο αριθμό επαναλήψεων (500K)



Εικόνα 2: Μέση ανταμοιβή για εκπαίδευση 500K σε ντετερμινιστικό περιβάλλον (DQN)

Παρομοίως, και οι υπόλοιποι αλγόριθμοι, (PPO, A2C) παρουσίαζαν το ίδιο πρόβλημα. (Μηδενική μέση ανταμοιβή ακόμα και για μεγάλο αριθμό επαναλήψεων). Κατά το βίντεο, όπως ήταν αναμενόμενο, ο πράκτορας επέλεγε την εξής στρατηγική: Να παραμείνει στον πρώτο όροφο, να μη χρησιμοποιήσει το ασανσερ (αφού έτσι δεν χάνει ποτέ). Κάτι τέτοιο αποκλίνει προφανώς απο τη στρατηγική που ακολουθεί ο άνθρωπος όταν παίζει κάποιο παιχνίδι.

Τα παραπάνω μας οδήγησαν στο συμπέρασμα, ότι για τη μηδενική μέση ανταμοιβή οφείλεται η μη υπαρξη στοχαστικότητας. Έτσι λοιπόν οι υπόλοιποι πειραματισμοί μας αφορούσαν περιβάλλον v0. Στο συγκεκριμένο περιβάλλον, σημειώθηκαν κάποια ενδιαφέροντα αποτελέσματα κυρίως στην περίπτωση ElevatorAction-v0:



Εικόνα 3: Μέση ανταμοιβή (κόκκινο χρώμα) για εκπαίδευση 500K σε μη *ντετερμινιστικό περιβάλλον (DQN)*

Ωστόσο, επειδή δουλεύουμε σε μη ντετερμινιστικό περιβάλλον, (και το *test_env* διαφέρει εν γένει απο το *learning_env*), ο πράκτορας στο βίντεο παρέμενε και πάλι ακίνητος (ίσως λόγω διαφορετικής αρχικοποιησης στο test_env). Η στρατηγική αυτή είχε το αποτέλεσμα να μη χάνει ποτέ. Απο την στιγμή που δεν έχανε λοιπόν δεν γινόταν και κάποιο update στα βάρη του δικτύου.

Όλα αυτά σε συνδυασμό με την παρακάτω αναφορά:

"Finally, we discovered that on some games the actual optimal strategy is by doing a loop over and over giving a small amount of reward. In Elevator Action the agent learn to stay at the first floor and kill over and over the first enemy. This behavior cannot be seen as an actual issue as the agent is basically optimizing score but this is definitely not the intended goal. A human player would never perform this way"

Πηγή: Is Deep Reinforcement Learning Really Superhuman on Atari (Marin Toromanoff, MINES ParisTech)

, μας οδήγησαν στο συμπέρασμα ότι το συγκεκριμένο παιχνίδι αντιμετωπίζει το πρόβλημα της "*κρυας εκκίνησης*". Άλλωστε ούτε κατά το βίντεο, ούτε κατά το gameplay (σε προσομοιωτή), είδαμε να εμφανίζεται αντίπαλος στην οροφή του κτιρίου. (*Πιθανώς σε κάποια άλλη version*). Η ανάλυση στο εξής θα αφορά το παιχνίδι MsPacman.

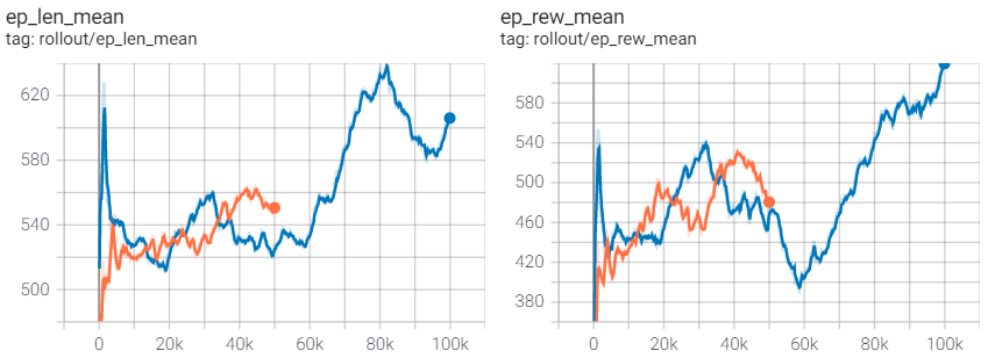
Παιχνίδι 2: MsPacman

A. Deterministic-v4

A1. DQN Deterministic-v4

Ξεκινήσαμε από 50K επαναλήψεις και παρατηρήσαμε ένα σχετικά χαμηλό σκορ γύρω στο 70 όπου ο πράκτορας μαθαίνει και επαναλαμβάνει μια συγκεκριμένη κίνηση. Αυξάνοντας στο διπλάσιο τις επαναλήψεις, βλέπουμε ότι συγκεντρώνονται περισσότεροι πόντοι, κοντά στο 300.

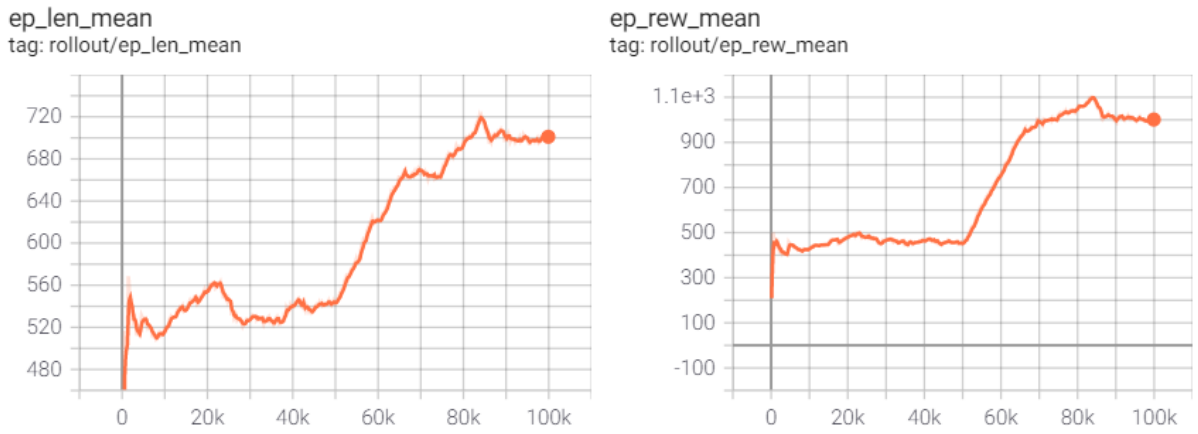
```
Eval reward: 279.8 (+/-106.34378214075329)
```



Εικόνα 4: Εκπαίδευση για 50K timesteps (πορτοκαλί χρώμα) & 100K (μπλε χρώμα) timesteps σε ντετερμινιστικό περιβάλλον με αλγόριθμο DQN.

Από τα παραπάνω διαγράμματα παρατηρούμε ότι έχουμε το περιθώριο να αυξήσουμε κι άλλο τις επαναλήψεις, ώστε να ανέβει περισσότερο η μέση ανταμοιβή. Γι' αυτό και εκπαιδεύουμε το μοντέλο μας για 500K επαναλήψεις και στη συνέχεια και για άλλες 100K. Το evaluation reward σκαρφάλωσε σε τιμή λίγο μεγαλύτερη από 1000.

```
Eval reward: 1024 (+/-428.51371039909566)
```

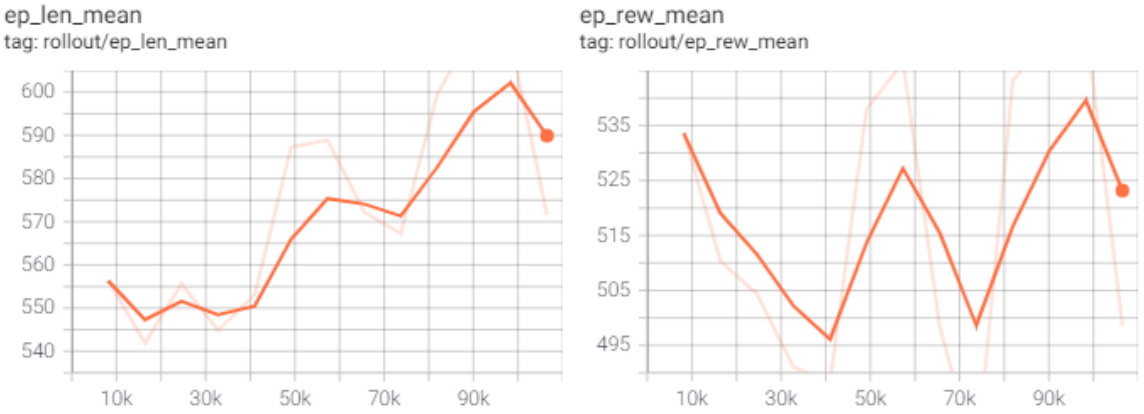


Εικόνα 5: Εκπαίδευση για επιπλέον 100K timesteps σε ντετερμινιστικό περιβάλλον με αλγόριθμο DQN.

Με βάση τα παραπάνω διαγράμματα παρατηρούμε έναν **κορεσμό** από ένα σημείο και μετά, όπου το μέσο μήκος επεισοδίου και η μέση ανταμοιβή παραμένουν σχεδόν σταθερά χωρίς ιδιαίτερη βελτίωση. Επομένως, αντιλαμβανόμαστε ότι δεν αξίζει να αυξήσουμε περαιτέρω τις επαναλήψεις.

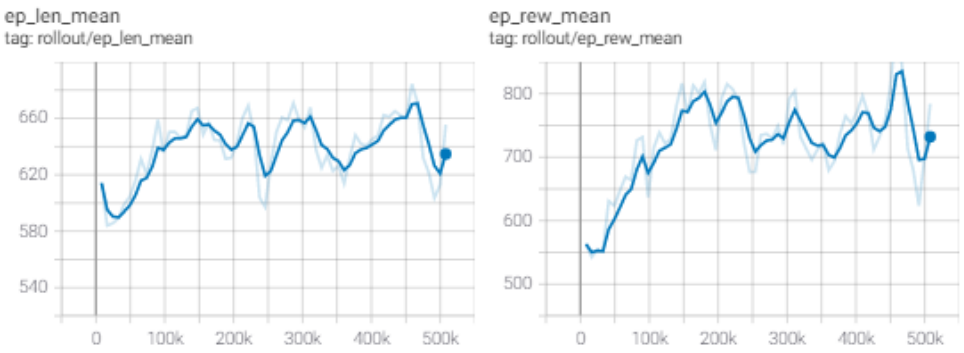
A2. PPO

Παρόμοια στρατηγική ακολουθήσαμε και για τον αλγόριθμο PPO, με τη μόνη διαφορά ότι ήταν εφικτή η εκπαίδευση σε πολλαπλά (4) περιβάλλοντα. Συγκριτικά λοιπόν με τον DQN παρατηρούμε ότι για 100K επαναλήψεις έχουμε μεγάλη διασπορά στα τελικά αποτελέσματα ανταμοιβής. (Με τον DQN είχαμε μόνο ορισμένες μικρές κυματώσεις).



Εικόνα 6: Εκπαίδευση για 100K timesteps σε ντετερμινιστικό περιβάλλον με PPO.

Η αύξηση των βημάτων εκπαίδευσης θα μας δώσει πιο ξεκάθαρη εικόνα. Για 500K επαναλήψεις (επιπλέον), παίρνουμε κατα την εκπαίδευση τα ακόλουθα διαγράμματα απο το TensorBoard.



Εικόνα 7: Εκπαίδευση για 500K timesteps σε ντετερμινιστικό περιβάλλον με PPO.

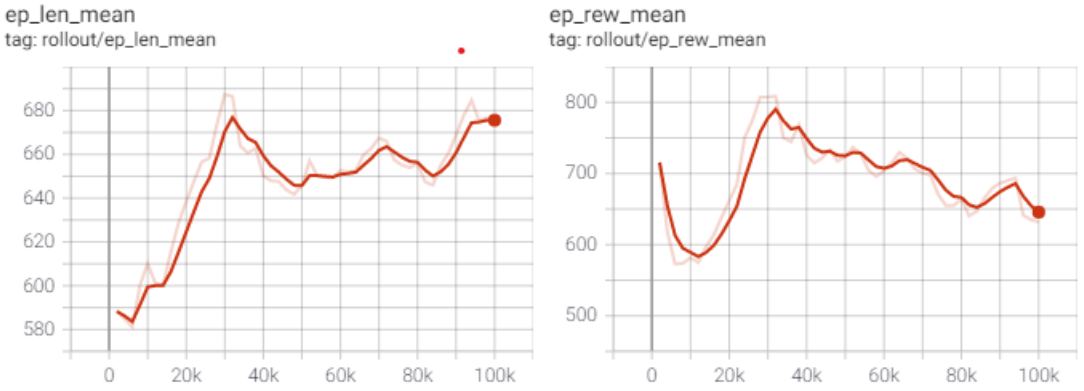
Και στη συγκεκριμένη περίπτωση προβάλλοντας το βιντεο, παρατηρούμε ότι ο πράκτορας ανεβαινει προς τα πάνω για να αποφύγει τους αντιπάλους και εγκλωβίζεται εκεί. Σε κάθε episode η αμοιβή του είναι 210 χωρίς διακυμάνσεις

```
Eval reward: 210.0 (+/-0.0)
```

A3. A2C Deterministic-v4

Ξεκινώντας από 1K επαναλήψεις, παρατηρήσαμε ότι δεν αρκούν για την επιτυχή εκπαίδευση του πράκτορα. (Καμία έξοδος σε tensorboard). Έτσι λοιπόν, αυξάνουμε κατά μια τάξη μεγέθους. Στη περίπτωση αυτή, παρατηρούμε ότι ο agent πλεον μαθαίνει να αποφεύγει τους αντιπάλους και να συγκεντρώνει πόντους (γύρω στους 400 - 500).

Όσο αυξάνουμε τον αριθμό των επαναλήψεων, τόσο αυξάνεται και η αμοιβή του πράκτορα. Βέβαια παρατηρούνται και κάποιες διακυμάνσεις γύρω από μια θέση ισορροπίας. Οι διακυμάνσεις αυτές πιθανως να οφείλονται στην ύπαρξη τεσσάρων διαφορετικών environments (στα οποία προφανώς σημειώνεται διαφορετικό σκορ κατά την εκπαίδευση). Από ένα σημείο και μετά όμως, η καμπύλη της μέσης ανταμοιβής αρχίζει να λαμβάνει τιμές γύρω από το 650. Αυτό επιβεβαιώνεται και από τα παρακάτω διάγραμμα:



Εικόνα 8: Εκπαίδευση για 500K timesteps σε ντετερμινιστικό περιβάλλον με αλγόριθμο A2C.

Από τα παραπάνω διαγράμματα, εξάγουμε το συμπέρασμα ότι δεν συνιστά καλή τακτική η επιλογή του συγκεκριμένου αλγορίθμου, αφού η ανταμοιβή από ένα σημείο και μετά μειώνεται συνεχώς. Επίσης το γεγονός ότι το μέσο μήκος επεισοδίου αυξάνεται, σηματοδοτεί ότι ο πράκτορας αποφευγει επιτυχώς τους αντιπάλους αλλά δεν συγκεντρώνει επιπλέον πόντους. Στο συγκεκριμένο παιχνίδι (pacman), όπως παρατηρούμε και απο το βίντεο, περνάει από μονοπάτια που έχει ήδη επισκεφτεί (στα οποία πλέον, προφανώς και δεν υπάρχουν πόντοι).

Τέλος ως προς αμοιβή στο περιβάλλον test λαμβάνουμε:

```
Eval reward: 70.0 (+/-0.0)
```

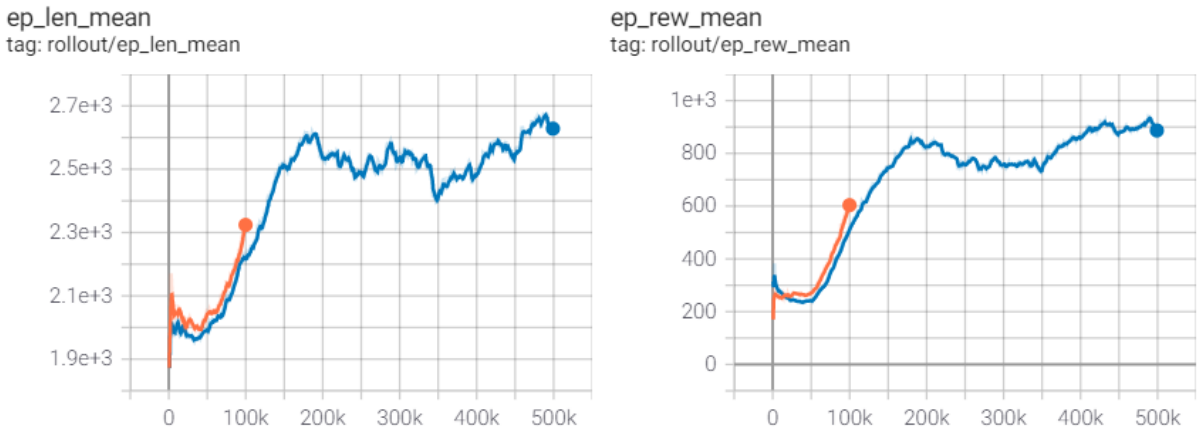
Πράγματι επιβεβαιώνουμε ότι θα πρέπει να μελετήσουμε και άλλα envs - algorithms

Συμπέρασμα: Στο περιβάλλον Deterministic καλύτερος αλγόριθμος ήταν ο **DQN**.

B. NoFrameskip-v4

B1. DQN NoFrameskip-v4

Ξεκινήσαμε να πειραματιζόμαστε αρχικά με σχετικά χαμηλό αριθμό *timesteps* (100K) ώστε να οδηγηθούμε από νωρίς σε ορισμένα συμπεράσματα για τον συγκεκριμένο αλγόριθμο. Όπως βλέπουμε και από τα παρακάτω διαγράμματα, έχουμε ανοδική τάση τόσο στη μέση ανταμοιβή, όσο και στο μέσο μήκος επεισοδίου. Επομένως αυξήσαμε κι άλλο τον αριθμό των επαναλήψεων στις 500K.



Εικόνα 9: Εκπαίδευση για 500K timesteps σε NoFrameskip περιβάλλον με DQN.

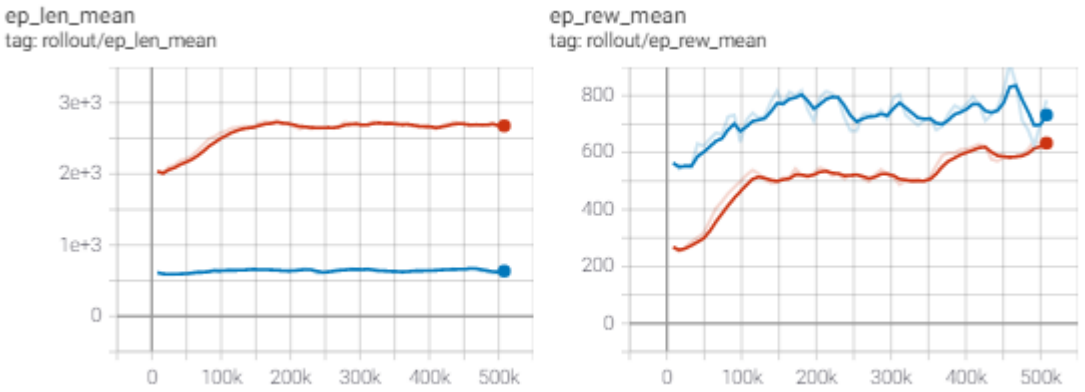
Το eval reward που πετυχαίνει είναι:

```
Eval reward: 537.0 (+/-130.770791845885847)
```

Όπως φαίνεται λοιπόν από τα παραπάνω διαγράμματα, αλλά και από το βίντεο, ο συγκεκριμένος αλγόριθμος στο περιβάλλον *NoFrameskip-v4*, μαθαίνει τον πράκτορα να μένει περισσότερη ώρα στο παιχνίδι (μεγάλο *ep_len_mean*) αποφεύγοντας τους αντιπάλους του, χωρίς όμως να συλλέγει αρκετούς πόντους παράλληλα. Περιορίζεται δηλαδή σε απόμακρα σημεία στο χάρτη (μένει σε παρόμοια μέρη) ώστε να αποφύγει τους αντιπάλους χωρίς να εξερευνά άλλα μονοπάτια στα οποία υπάρχουν διαθέσιμοι πόντοι. Μάλιστα σε κάποιες περιπτώσεις, ο πράκτορας ταλαντώνεται γύρω από μια θέση. Μια περαιτέρω αύξηση των *timesteps* ενδεχομένως να οδηγήσει σε μια μικρή αύξηση του σκορ, η οποία όμως δεν θα είναι πολύ σημαντική σύμφωνα με τις παραπάνω μετρήσεις.

B2. PPO NoFrameskip-v4

Ο πρώτος πειραματισμός μας για το συγκεκριμένο αλγόριθμο πραγματοποιήθηκε για μεγάλο αριθμό επαναλήψεων (500K). Στην παρακάτω γραφική, βλέπουμε ότι έχει αρκετά καλή δυναμική ο συγκεκριμένος αλγόριθμος.



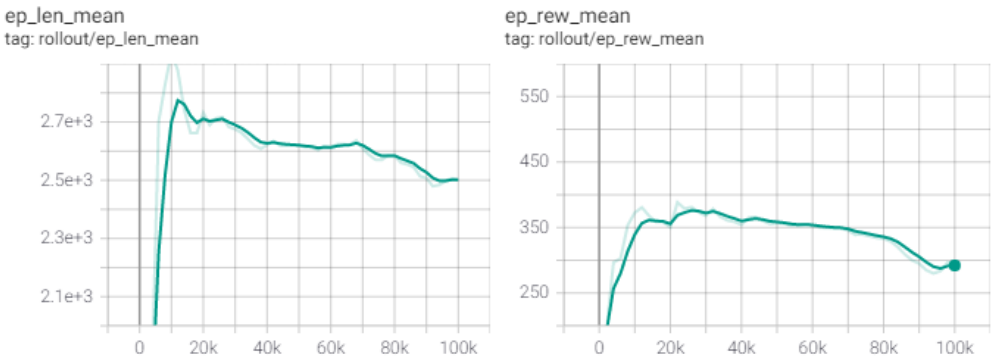
Εικόνα 10: Μέση ανταμοιβή για εκπαίδευση 500K σε ντετερμινιστικό περιβάλλον (μπλε χρώμα) και σε μη ντετερμινιστικό (κόκκινο χρώμα) για αλγόριθμο PPO.

Αξιζει να σταθεί κανείς στο γεγονός ότι η εκπαίδευση σε περιβάλλον ντετερμινιστικό πραγματοποιείται πίο γρήγορα (ήδη απο 100K επαναλήψεις επιτυγχάνεται σκορ ίσο με 800 κατά την εκπαίδευση). Αυτό μάλιστα επιβεβαιώνεται και από το γεγονός ότι στη περίπτωση ντετερμινιστικού περιβάλλοντος, το μήκος επεισοδίου είναι κοντά στο 500, υποτετραπλασιο (σχεδόν) από το αντίστοιχο μέσο μήκος του NoFrameSkip περιβάλλοντος.

Ωστόσο συγκριτικά με τον προηγούμενο αλγόριθμο (DQN), φαίνεται να υστερεί, οπότε δεν πειραματιζόμαστε περαιτέρω με τον συγκεκριμένο αλγόριθμο (PPO).

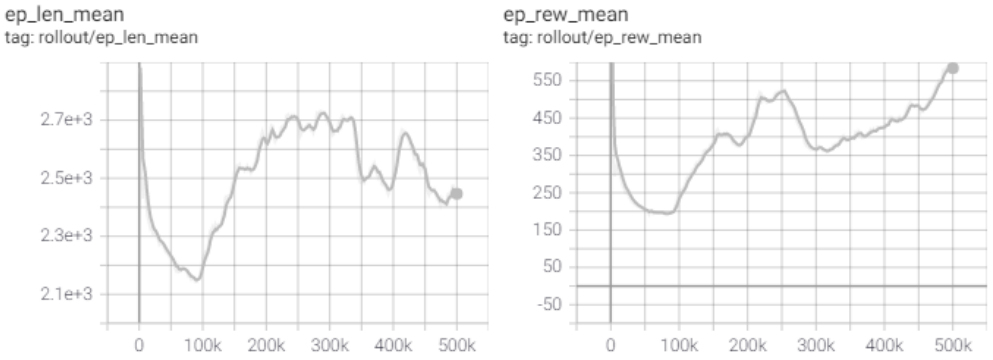
B3. A2C NoFrameskip-v4

Αρχικά πειραματιστήκαμε με λίγα *timesteps* ώστε και πάλι να εξαγάγουμε από νωρίς συμπεράσματα για τον συγκεκριμένο αλγόριθμο στο περιβάλλον *NoFrameskip.* Αυτό που παρατηρήσαμε ήδη από τις 100K επαναλήψεις ήταν ότι ο αλγόριθμος δεν είναι αρκετά υποσχόμενος. Στα παρακάτω διαγράμματα φαίνεται μια καθοδική τάση στην μέση ανταμοιβή.



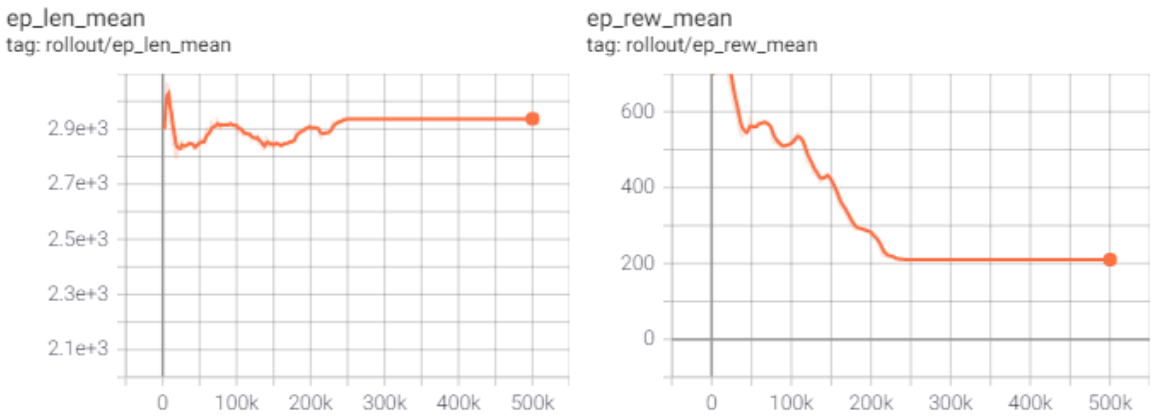
Εικόνα 11: Εκπαίδευση για 100K timesteps σε NoFrameskip περιβάλλον με A2C.

Θα δοκιμάσουμε ωστόσο να αυξήσουμε και άλλο το χρόνο εκπαίδευσης καθώς η μορφή της γραφικής (raw_mean) δεν αποκλείει το ενδεχόμενο να αυξηθεί περαιτέρω.



Εικόνα 12: Συνέχιση εκπαίδευσης για 500K timesteps σε NoFrameskip με A2C.

Πράγματι, όπως φαίνεται και παραπάνω, η ανταμοιβή μπορεί αρχικά να μειώνεται, αλλά στη συνέχεια παρουσιάζει ανοδική τάση. Οπότε θα ήταν παράλειψη αν δεν διερευνούσαμε περαιτέρω την απόδοση του αλγορίθμου, για περισσότερες επαναλήψεις. Συνεχίζουμε και πάλι την εκπαίδευση για άλλες 500K επαναλήψεις, ώστε να δούμε αν εν τέλει φτάνουμε σε κορεσμό. Όπως φαίνεται και απο την παρακάτω γραφική αναπαράσταση, αυτό συμβαίνει. (Η ανταμοιβή συγκλίνει σε μια ευθεία).



Εικόνα 13: Συνέχιση εκπαίδευσης για 500K timesteps σε NoFrameskip περιβάλλον με A2C.

Η ανταμοιβή στο περιβάλλον test (οπως ήταν αναμενόμενο απο τη γραφική στη TensorBoard), σημειώθηκε ίση με:

```
Eval reward: 210.0 (+/-0.0)
```

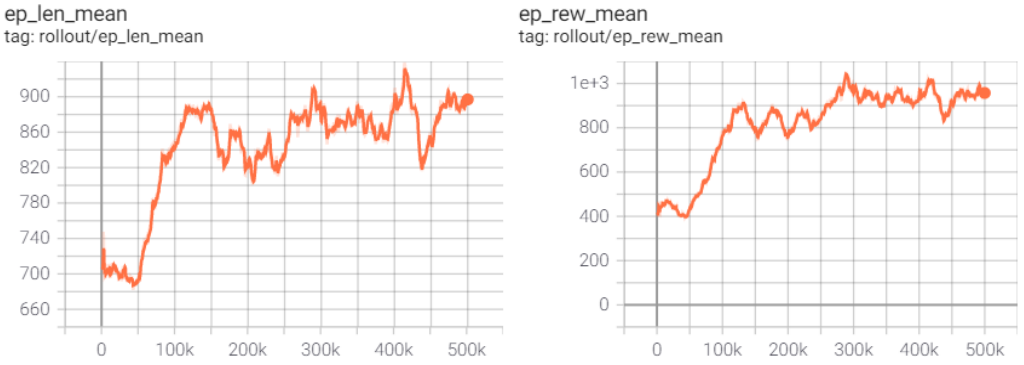
Συμπέρασμα: Στο περιβάλλον NoFrameSkip καλύτερος αλγόριθμος ήταν ο **DQN**.

C. Empty-v4

C1. DQN

Empty-v4

Πειραματιστήκαμε με 500K *timesteps* και όπως παρατηρούμε και από τα παρακάτω διαγράμματα, δεν αξίζει να τα αυξήσουμε περαιτέρω, καθώς το σύστημα μας φτάνει σε κορεσμό. Δηλαδή, τόσο το μέσο μήκος επεισοδίου, όσο και η μέση ανταμοιβή, έπειτα από τις 300K, δεν σημειώνουν σημαντική άνοδο.



Εικόνα 14: Εκπαίδευση για 500K timesteps σε Empty περιβάλλον με αλγόριθμο DQN.

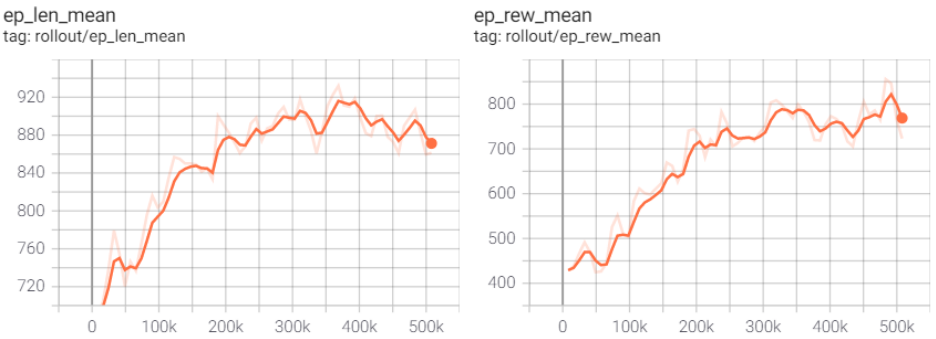
Το evaluation reward που πετυχαίνουμε με τον συγκεκριμένο αλγόριθμο στο σκέτο περιβάλλον είναι το εξής:

```
Eval reward: 975.0 (+/-320.69455873151327)
```

C2. PPO Empty-v4

Για 500K επαναλήψεις παίρνουμε κατα την εκπαίδευση τα παρακάτω διαγράμματα απο το TensorBoard. Στον συγκεκριμένο αλγόριθμο, στο σκέτο περιβάλλον, προβάλλοντας το βιντεο, παρατηρούμε ότι ο πράκτορας ανεβαίνει προς τα πάνω για να αποφύγει τους αντιπάλους και εγκλωβίζεται εκεί. Σε κάθε episode η αμοιβή του είναι 210 χωρίς καθόλου διακυμάνσεις. Όπως μας φανερώνουν και τα διαγράμματα, δεν θα ωφελήσει η περαιτέρω αύξηση του αριθμού των *timesteps*.

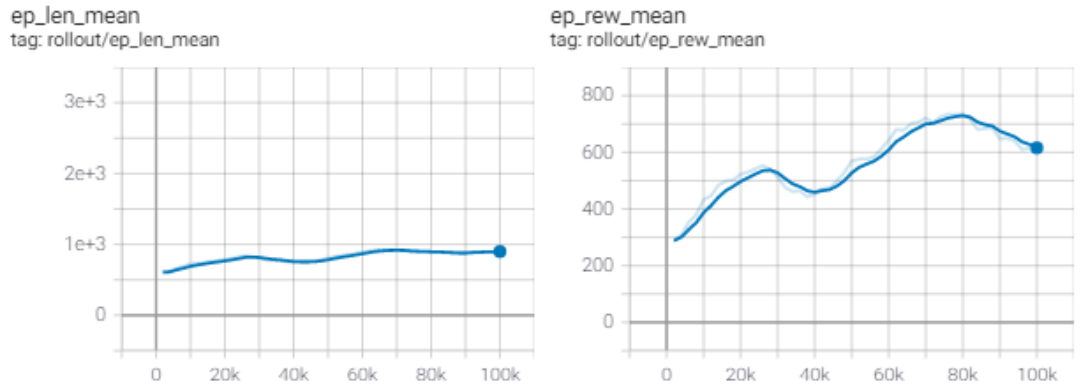
```
Eval reward: 210.0 (+/0.0)
```



Εικόνα 14: Εκπαίδευση για 500K timesteps σε Empty περιβάλλον με αλγόριθμο PPO.

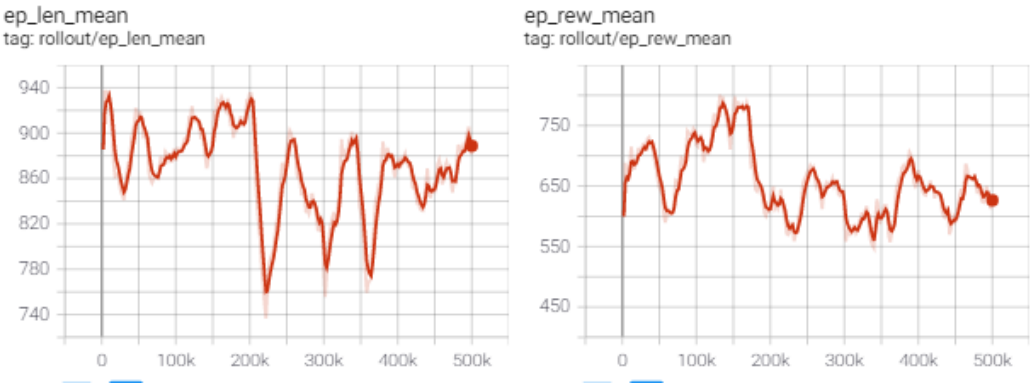
C3. A2C Empty-v4

Ομοίως με τα προηγούμενα envs, και εδώ ξεκινήσαμε με λίγες επαναλήψεις και είδαμε ότι ο αλγόριθμος συμβάλει καθοριστικά στην αύξηση της ανταμοιβής, ενώ παράλληλα το μήκος του επεισοδίου κυμαίνεται γύρω από μια σταθερή τιμή.



Εικόνα 15: Εκπαίδευση για 100K timesteps σε Empty περιβάλλον με αλγόριθμο A2C.

Έτσι συνεχίζουμε να αυξάνουμε και άλλο τα βήματα εκπαίδευσης, ώστε να φτάσουμε σε κορεσμό (όπως φαίνεται και παρακάτω).



Εικόνα 16: Συνέχιση εκπαίδευσης για 500K timesteps σε Empty περιβάλλον με αλγόριθμο A2C.

Η ανταμοιβή στο περιβάλλον test (οπως ήταν αναμενόμενο), σημειώθηκε ίση με :

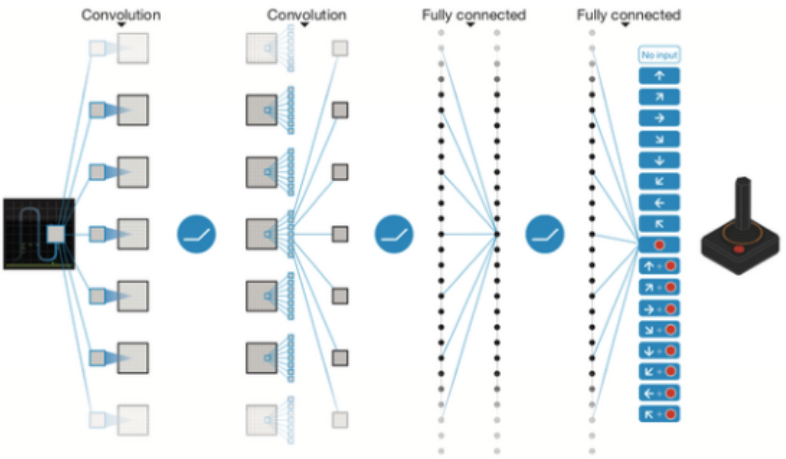
```
Eval reward: 210.0 (+/-0.0)
```

Παρατηρούμε παρόμοια συμπεριφορά με την παραμετροποίηση “NoFrameskip-v4”, οπότε τα συμπεράσματα μας δεν επεκτείνονται. Άλλωστε τα δυο αυτά environments είναι παρόμοια (στο ένα η ενέργεια αποφασίζεται σε κάθε frame, στο άλλο αν 2 - 4 frames).

Συμπέρασμα: Στο περιβάλλον “Empty” καλύτερος αλγόριθμος ήταν ο **DQN**.

Επιλογή Καλύτερων Αλγορίθμων

Και στα τρία διαφορετικά προβλήματα (που ορίζει αντίστοιχα το κάθε περιβάλλον), ξεχώρισε ο αλγόριθμος DQN. Πρόκειται για έναν αλγόριθμο ενισχυτικής μάθησης (*Reinforcement Learning*), η αρχιτεκτονική του οποίου μπορεί να περιγραφεί από τα ακόλουθα δύο βήματα. Στο πρώτο βήμα μια σειρά απο συνελικτικά δίκτυα μαθαίνει να εξάγει χαρακτηριστικά από την εικόνα εισόδου του παιχνιδιού. Στο δεύτερο βήμα ακολουθεί ένας ταξινομητής (dense) ο οποίος αντιστοιχίζει τη τρέχουσα παρατήρηση σε ένα κόμβο του τελικού layer. Κάθε κόμβος του τελικού layer αντιστοιχεί σε κάποια κίνηση του atari joystick.



Εικόνα 17:Δομή Νευρωνικού Δικτύου σύμφωνα με το paper της DeepMind.

Οι τιμές εξόδου αντιπροσωπέυουν λοιπόν τη καλύτερη πρόβλεψη που κάνει ο πράκτορας για την επικείμενη ανταμοιβή. Αυτή η προσέγγιση (κατάσταση - πρόβλεψη) είναι γνωστή σαν value - based learning. Μαθηματικά, χρησιμοποιείται μια συνάρτηση Q(s,a), όπου s η κατάσταση και a η ενέργεια (action). Στόχος λοιπόν είναι η βελτιστοποίηση της συνάρτησης:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s,\, a_t = a,\, \pi\right],$$

όπου γ ένας παράγοντας μείωσης, και r_{t+n}, μελλοντικές αμοιβές που δείχνουν πόσο μακριά μπορεί να φτάσει ο πράκτορας. Με άλλα λόγια, σε κάθε βήμα (κατάσταση) παίρνουμε την ενέργεια που οδηγεί στη μέγιστη ανταμοιβή. Ωστόσο ο πράκτοράς μας δεν μπορεί να δει το μέλλον (!), και έτσι δεν μπορεί να αναθέσει μια συγκεκριμένη τιμή στη παράσταση Q για κάθε μια από τις καταστάσεις. Για το λόγο αυτό έρχονται τα νευρωνικά δίκτυα. Αντιστοιχίζοντας λοιπόν εικόνες απο pixel σε τιμές Q, το νευρωνικο μας συμπεριφέρεται σαν ένας εκτιμητής Q τιμών. Οπότε αν και αδυνατεί να δει το μέλλον, μπορεί να το προβλέψει.

Αναφορικά με τη συνάρτηση Loss, πρόκειται για τη διαφορά μεταξύ των πραγματικών τιμών Q και των προβλέψεων. Βέβαια η πραγματική τιμή Q δεν είναι τίποτα άλλο πέρα από την αμοιβή στη τρέχουσα κατάσταση, συν την τιμή Q στην κατάσταση που θα πάει ο πράκτορας βάσει κάποιας ενέργειας. Βλέπουμε λοιπόν ότι ακόμα και οι πραγματικές μας τιμές υπολογίζονται από το δίκτυο κατά την εκπαίδευση. Και εδώ λοιπόν έγκειται η διαφορά μεταξύ supervised learning (οι ετικέτες - δίνονται εξ αρχής στο dataset) και reinforcement learning (οι ετικέτες ή καλύτερα πραγματικές τιμές παράγονται από το δίκτυο). Στη δεύτερη περίπτωση χρησιμοποιούνται τρεις τεχνικές

Η πρώτη τεχνική είναι ο “διπλασιασμός” του νευρωνικού μας. Με το ένα αντίγραφο λοιπόν, δημιουργούμε τις “πραγματικές” τιμές Q και με το άλλο (online network) κάνουμε τις εκτιμήσεις μας. Μόνο το δεύτερο εκπαιδεύεται, αλλά τα βάρη του αντιγράφονται στο target network.

Η δεύτερη τεχνική είναι η χρήση Επιπλέον ενός buffer εμπειρίας. Ο buffer αυτός, είναι στην ουσία ένα dataset απο τις προηγούμενες εμπειρίες του πράκτορα (Τρέχουσα_Κατάσταση, Ενέργεια, Αμοιβή, Τερματισμός_Επεισοδίου (true/false), Επόμενη_Κατάσταση). Αντί λοιπόν ο πράκτορας να περιπλανάται σε ένα περιβάλλον χωρίς γνώση, χρησιμοποιεί αναμνήσεις (τις οποίες αντλεί από το dataset αυτό). Κάτι τέτοιο προσιδιάζει στον άνθρωπο που ανακαλεί μνήμες και αποκτά έτσι εμπειρία.

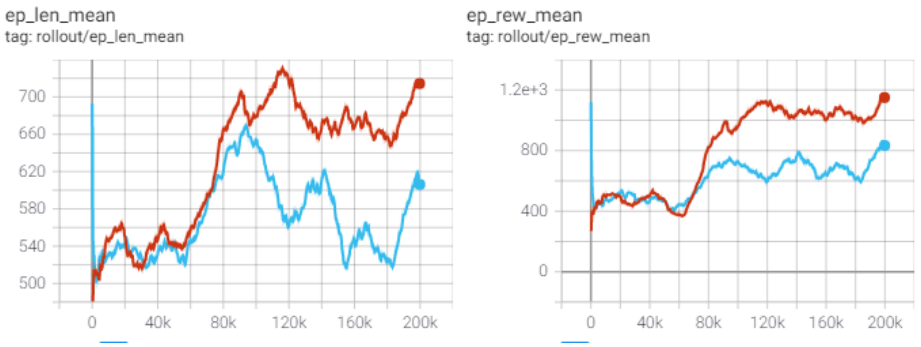
Η τρίτη τεχνική είναι η *epsilon_greedy,* η οποία εγγυάται ότι ο πράκτορας pacman δε θα "κολλήσει" σε κάποια γωνία ή ότι δε θα κάνει ταλαντώσεις. Η συγκεκριμένη τεχνικη ορίζει ότι κατά 90% ο πράκτορας θα κάνει τη πράξη που του υποδεικνύει το νευρωνικό και κατά 10% μια τυχαία. Κάτι τέτοιο είναι αρκετά σημαντικό αφού στους άλλους αλγορίθμους (A2C, PPO), είδαμε τον πράκτορα να στέκεται ακίνητος σε ορισμένες θέσεις. (Βλ βίντεο στο αρχείο videos/MsPacman_PPO_Bad_Case.mp4)

Βελτιστοποίηση Καλύτερων Αλγορίθμων

Από τη στιγμή που και στα τρια διαφορετικά προβλήματα καλύτερος αλγόριθμος ήταν ο DQN, επιλέγουμε να βελτιστοποιήσουμε αυτόν. Το περιβάλλον στο οποίο πραγματοποιήθηκε η βελτιστοποίηση αυτή ήταν το Deterministic καθώς ήταν ταχύτερη η εκπαίδευση και σημειώθηκε το υψηλότερο σκορ.

Χρησιμοποιώντας τις διαθέσιμες παραμέτρους από το documentation του SB3, βελτιστοποιούμε την επίδοση του παραπάνω αλγορίθμου ως προς το score στο test environment. Οι παράμετροι με τις οποίες πειραματιστήκαμε ήταν οι εξής:

policy: Οι δυνατές πολιτικές που μπορούσε να χρησιμοποιήσει το μοντέλο μας ήταν δυο: MLP ή CNN. Στο παρακάτω διάγραμμα, εκπαιδεύσαμε από την αρχή δύο μοντέλα χρησιμποιώντας κάθε φορά διαφορετική στρατηγικη.

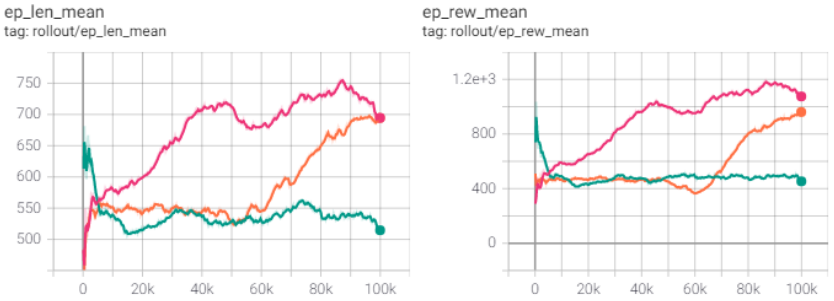


Εικόνα 17: Συγκριτική μελέτη επίδοσης CNN (κόκκινο χρώμα) έναντι MLP (μπλε χρώμα)

Προτιμήσαμε την στρατηγική CNN, αφού μέσω αυτής πετυχαίναμε υψηλότερα σκορ. Βέβαια στις περιπτώσεις αυτές, απαιτούνταν περισσότεροι πόροι κυρίως σε μνήμη - χωρητικότητα (7 GB / 16 GB στο περιβάλλον kaggle). **Τα παρακάτω βήματα αφορούν μοντέλο με στρατηγική CNN.**

learning_rate: Κάθε φορά που συνεχίζουμε την εκπαίδευση, φροντίζουμε να μειώνουμε το ρυθμό εκμάθησης. Συγκεκριμένα, για τις πρώτες 1M επαναλήψεις ο ρυθμός εκμάθησης τέθηκε ίσος με 10^{-4}, ενώ στη συνέχεια , για τις επόμενες 2M επαναλήψεις μειώθηκε κατά μία τάξη μεγέθους ($= 10^{-5}$). Κάθε φορά λοιπόν που φορτώνουμε ένα προεκπαιδευμένο μοντέλο, φροντίζουμε να μειώσουμε και το learning rate.

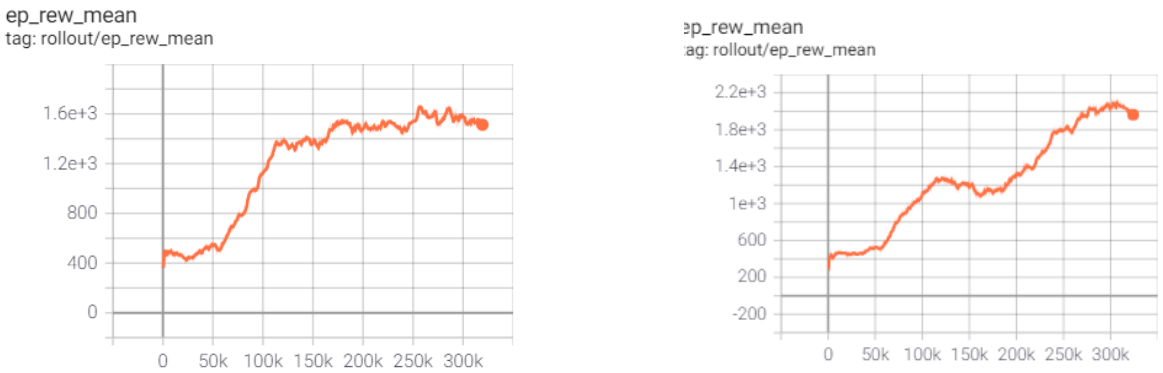
learning_starts: Με τη συγκεκριμενη παράμετρο, ορίζουμε πόσα βήματα θα συλλέγει το μοντέλο μας (σχετικά με τις μεταβάσεις), προτού ξεκινήσει η εκπαίδευση. Η default τιμή ήταν $5 \bullet 10^4$. Θα εξετάσουμε άλλες δύο τιμές (μια μικρότερη και μια μεγαλύτερη, για να δούμε πώς επηρεάζει η συγκεκριμένη παράμετρος την εκπαίδευση).



Εικόνα 18: Συγκριτική μελέτη επίδοσης CNN μοντέλου για learning starts = 500 (ροζ χρώμα), $5 \bullet 10^4$ *(πορτοκαλί χρώμα) και* $5 \bullet 10^6$ *(μπλε χρώμα)*

Παρατηρεί κανείς ότι για την default τιμή της παραμέτρου learning starts ($= 5 \bullet 10^4$), έχουμε τη καλύτερη (συγκριτικά με τις άλλες περιπτώσεις) συμπεριφορά. Πιο ειδικά, η πορτοκαλί καμπύλη έχει την προδιάθεση να σημειώσει υψηλότερες τιμές στον κατακόρυφο άξονα, παρόλο που για λίγες επαναλήψεις βρίσκεται κάτω από τη ροζ καμπύλη. **Τα παρακάτω βήματα αφορούν μοντέλο με στρατηγική CNN και learning_starts = 50000.**

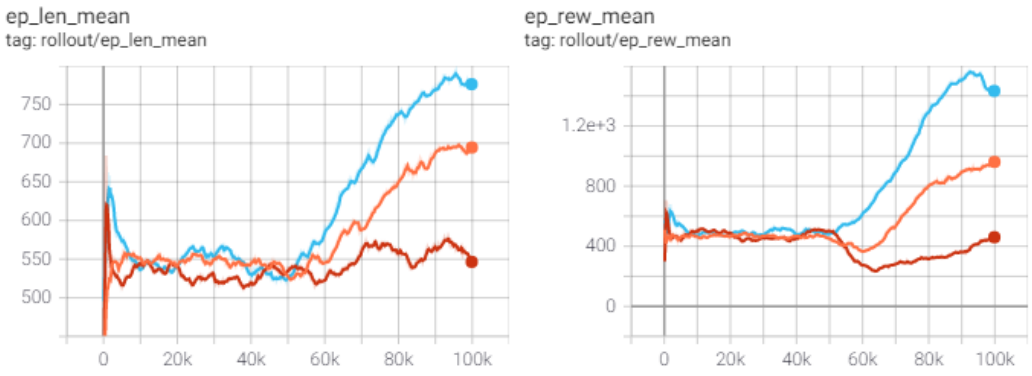
tau: Πρόκειται για ένα συντελεστή, ο οποίος καθορίζει αν κατά την εκπαίδευση θα συντελείται *soft update* (tau = 0) ή *hard update* (tau = 1).



Εικόνα 19: Συγκριτική μελέτη επίδοσης CNN μοντέλου για learning starts = $5 \bullet 10^4$ *και soft update (αριστερά) και hard update (δεξιά)*

Γίνεται αντιληπτό ότι, παρόλο που το soft update επιτυγχάνει ταχύτερη σύγκλιση, οδηγείται και πιο γρήγορα σε κορεσμό. Για το λόγο αυτό, επιλέγουμε hard update παραμέτρων. **Το παρακάτω βήμα αφορά μοντέλο με στρατηγική CNN, learning starts = 50000 και hard update (tau = 1).**

train_freq: By default, το μοντέλο ενημερώνεται κάθε 4 steps. Δοκιμάζουμε το update αυτό να συντελείται με μεγαλύτερο ή μικροτερο ρυθμο.



Εικόνα 20: Συγκριτική μελέτη επίδοσης CNN μοντέλου για learning starts = $5 \bullet 10^4$*, hard update και train_freq = 1 (μπλε χρώμα), train_freq = 4 (πορτοκαλί χρώμα), και train_freq = 20 (κόκκινο χρώμα)*

Απο το παραπάνω διάγραμμα συμπεραίνουμε ότι καλύτερη δυναμική για το παιχνίδι μας, έχει η παραμετροποίηση train_freq = 1. Ωστόσο ήδη από τις 100K επαναλήψεις, η RAM του kaggle φαινόταν ανεπαρκής, για να υποστηρίξει τη συγκεκριμένη εκπαίδευση. Οπότε επιλέγουμε train freq = 2.

Σύνοψη

Το μοντέλο λοιπόν που σχεδιάστηκε, ακολουθούσε στρατηγική CNN, με hard update παραμέτρων, εκκίνηση εκπαίδευσης μετά από 50000 μεταβάσεις και ανανέωση παραμέτρων κάθε 2 βήματα. Το μέσο σκορ που σημειώθηκε ήταν **4396**.

```
Eval reward: 4396.0 (+/-963.0700909071987)
```

Το τελικό βίντεο μετά την βελτιστοποίηση του αλγορίθμου, φαίνεται στο αρχείο: videos/MsPacman_DQN_Best_Case.mp4

Πειραματισμός με Στοχαστικότητα

Απο τη στιγμή που επιτύχαμε ικανοποιητικό score σε περιβάλλον χωρίς στοχαστικότητα, δοκιμάζουμε τον αλγόριθμο σε χώρο με sticky actions (ο πράκτορας στην επόμενη κίνηση δεν θα κάνει απαραίτητα αυτό που έχει αποφασίσει, αλλά με μια μικρή πιθανότητα p θα κάνει την προηγούμενη ενέργειά του).

Το μοντέλο λοιπόν που σημείωσε το υψηλότερο σκορ (βλ Σύνοψή) δοκιμάστηκε σε περιβάλλον v0 και πέτυχε σκορ:

```
Eval reward: 1914.0 (+/-524.1698198103359)
```

Το βίντεο μετά την εκπαίδευση σε περιβάλλον v0, φαίνεται στο αρχείο:

videos/MsPacman_DQN_v0.mp4

Σύγκριση με State-of-the-Art συστήματα

Αναζητώντας τα μοντέλα με τα υψηλότερα score στη σελίδα paperswithcode, παρατηρήσαμε ότι το MuZero model είναι αυτό που πετυχαίνει το μεγαλύτερο reward (243401.10). Όμως, όσον αφορά τα συστήματα που χρησιμοποίησαν όπως κι εμείς τον αλγόριθμο DQN, βλέπουμε ότι το μεγαλύτερο score που έχει επιτευχθεί είναι το 5821 (QR-DQN-1) το οποίο δεν απέχει ιδιαίτερα από τα δικά μας αποτελέσματα. Στο συγκεκριμένο μοντέλο χρησιμοποιήθηκε η πλατφόρμα “ReAgent”, η οποία αναπτύχθηκε και χρησιμοποιείται από το Facebook.

Προφανώς, λόγω των περιορισμών που έχουμε στους πόρους (RAM, CPU, GPU) καθώς και του περιβάλλοντος cloud που δουλέψαμε, είναι απολύτως λογικό να έχουμε χαμηλότερη απόδοση από τα State-of-the-Art μοντέλα. Για παράδειγμα, παρόλο που χρησιμοποιήσαμε πολιτική CNN, αν επιθυμούσαμε να τροποποιήσουμε την αρχιτεκτονική του, θα έπρεπε να το κάνουμε μέσω της παραμέτρου αρχικοποίησης *policy_kwards* (περνώντας τα επιμέρους ορίσματα σε dictionary). Τα ορίσματα αυτά αφορούν το δίκτυο μόνο και έτσι είναι αδύνατη κάποια προεπεξεργασία (πχ resize εικόνων, μετατροπή σε grayscale). Επίσης το γεγονός ότι δεν διατίθεται documentation για τα επιμέρους ορίσματα του dict, παρά μόνο ο κώδικας git της βιβλιοθήκης, καθιστά πιο δύσκολους τους πειραματισμούς απο τη πλευρά του προγραμματιστή. Τέλος, στα περισσότερα papers χρησιμοποιείται η βιβλιοθήκη PyTorch αντί για Tensorflow, η οποία λόγω της μεγαλύτερης “ευελιξίας” της μπορεί να βοήθησε στην αποτελεσματικότερη δημιουργία βέλτιστων συστημάτων.

Οι ισχυρισμοί μας για μεγάλη υπολογιστική ισχύ τεκμηριώνονται και απο την ακόλουθη αναφορά.

“Something to know about DQNs (and RL algorithms in general) is that they are sample inefficient and expensive to train. The standard approach in the DQN literature is to run 200 million frame training sessions. That’s about 930 hours of human-speed gameplay or roughly 20 days of training on a single P4000 GPU (at least for the final DQN variant we’ll be getting to). We don’t have the resources to pull something like that off for every version of the algorithm. Instead, we chose to run these comparison experiments for 10 million frames (enough to highlight the discrepancies), and to train our final, best-performing agent for a more extended 40 million step run.”

Πηγή: Advanced DQNs: Playing Pac-man with Deep Reinforcement Learning

---