Αριθμός Ομάδας: 66
Τσιλιβής Θεόδωρος 03116032
Στόικου Θεοδότη 03117085
Ποταμίτου Νεφέλη 03117709

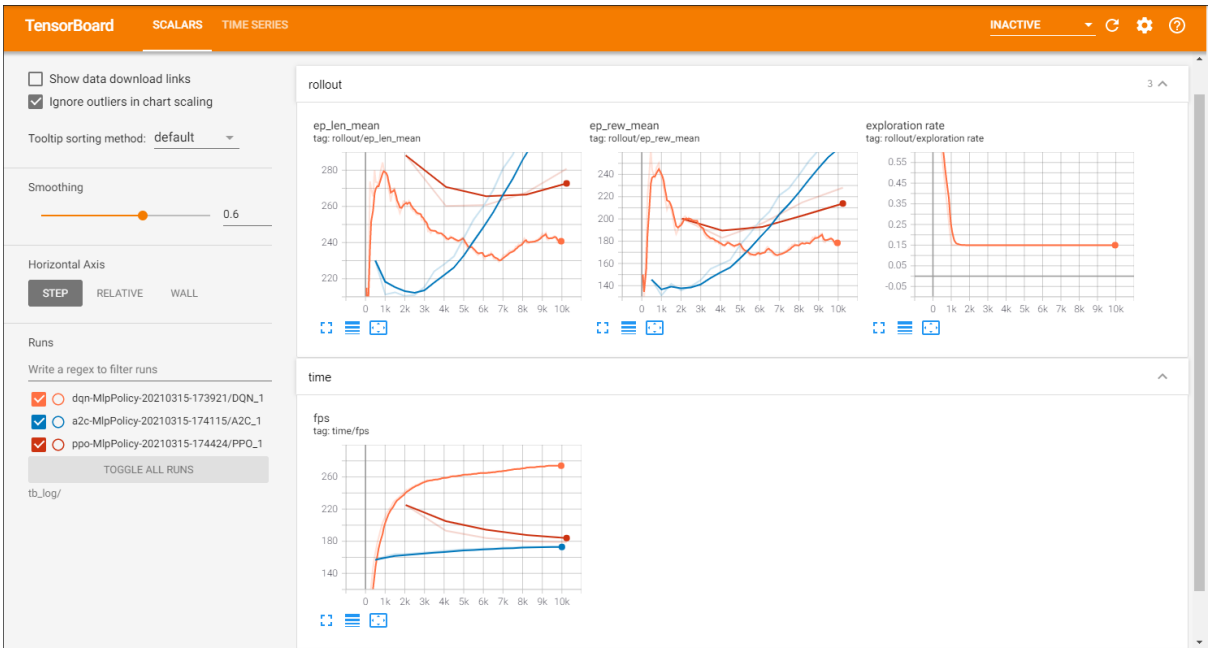
4ή Εργασία Νευρωνικών Δικτύων 2020-2021

Ο κυριότερος λόγος ύπαρξης της παρούσας αναφοράς είναι να μπορούμε να παρουσιάσουμε τα γραφήματα και ένας γρήγορος σχολιασμός των αποτελεσμάτων μας. Συγκεκριμένα για του αλγόριθμους μάθησης DNQ, A2C, PPO. Να σημειωθεί πως δεν εκπαιδεύσαμε με τον αλγόριθμο (QR-DQN) διότι crash-αρε το google colab. Μαζί με την αναφορά θα επισυνάψουμε και μερικά video agent καθώς και τον κώδικα ο οποίος όμως δεν θα έχει το output για λόγους χώρου (10MB).

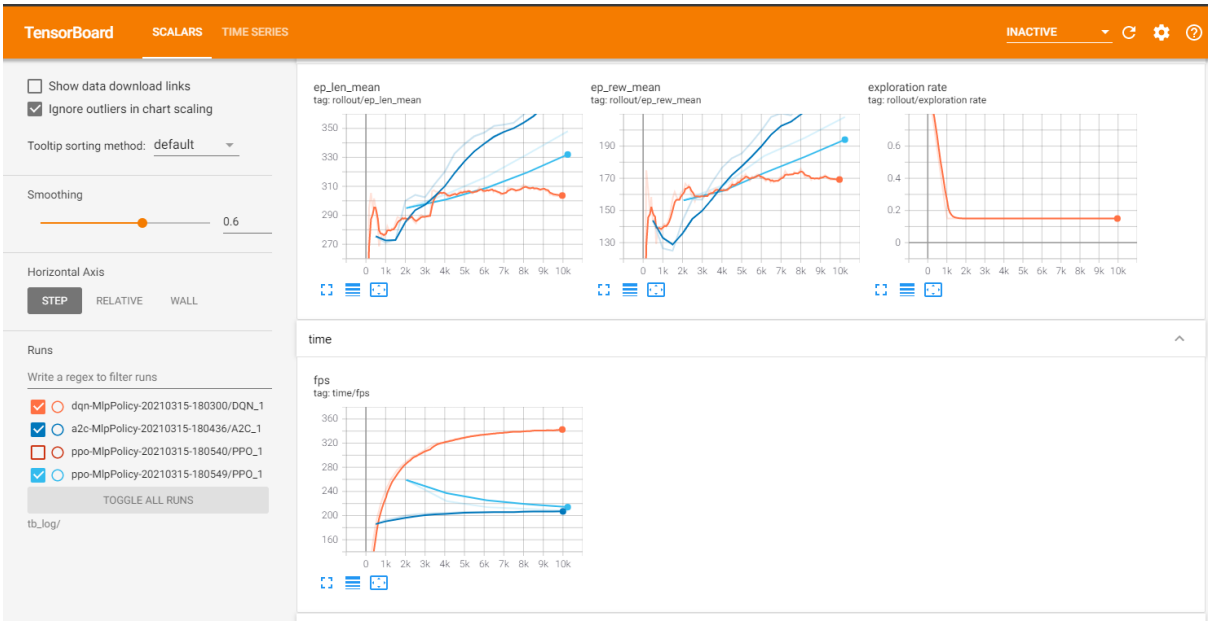
Ως ενδεικτικά αποτελέσματα από τον agent με random κινήσεις που υλοποιήσαμε καταλαβαίνουμε πως το παιχνίδι που πρέπει να “παίξουμε” είναι αρκετά πολύπλοκο. Ο random agent πετυχαίνει στις περισσότερες των περιπτώσεων score 0 ή 50 το οποίο αντιστοιχεί σε μία επιτυχή πράξη πριν χάσει ο παίκτης.

Στην συνέχεια θα χρησιμοποιήσουμε το Tensorboard για να εντοπίσουμε ποιος αλγόριθμος θα μας δώσει τα καλύτερα δεδομένα και άρα σε ποιον θα στηριχτούμε για να προχωρήσουμε. Εκπαιδεύουμε με κάθε αλγόριθμο στα μοντέλα v4 με επιλογές Deterministic, Simple και No Frameskip

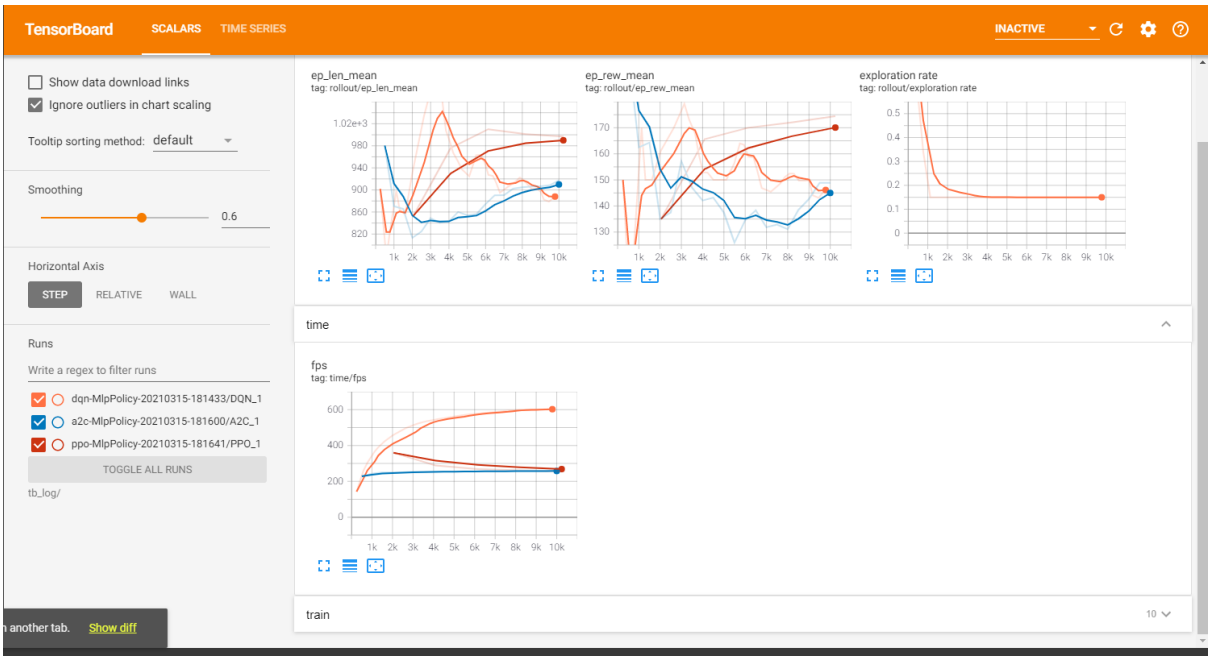
Deterministic



Simple



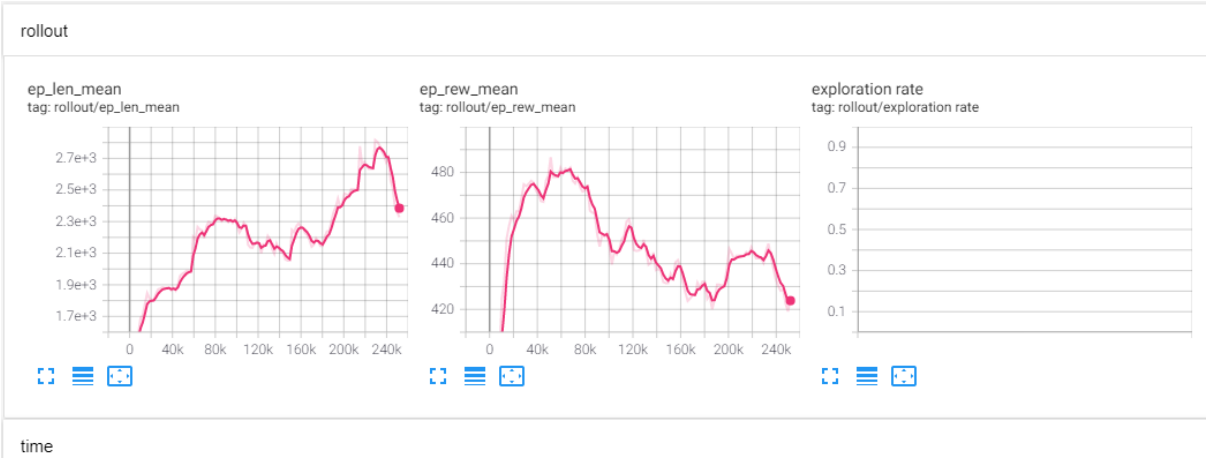
No-Frameskip



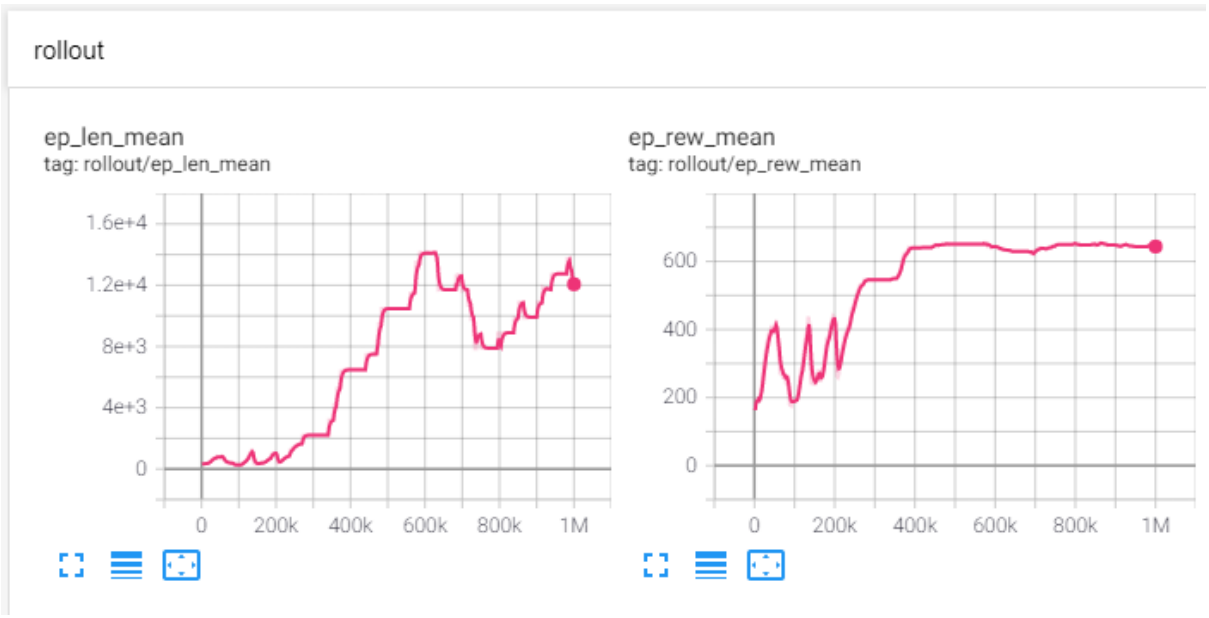
Τα παραπάνω αποτελέσματα μας δείχνουν πως σε όλες τις περιπτώσεις ο agent που εκπαιδεύτηκε με το απλό DQN είναι χειρότερος από τους agent των άλλων δύο αλγορίθμων. Επειδή τα παραπάνω αποτελέσματα δεν δίνουν σαφή εξήγηση για το ποιος απο τους PPO και A2C αλγόριθμους είναι ο καλύτερος θα προσπαθήσουμε να εξετάσουμε την συμπεριφορά και των δύο. Σε γενικές γραμμές βλέπουμε πως στο No-Frameskip έχουμε υψηλότερα score.

Θα εκπαιδεύσουμε ένα μοντέλο PPO για No-Frameskip με 500.000 max steps και ένα μοντέλο A2C για Simple με 1.000.000 max steps

Τα αποτελέσματα για το PPO μοντέλο είναι τα εξής :



Ενώ για το A2C μοντέλο :



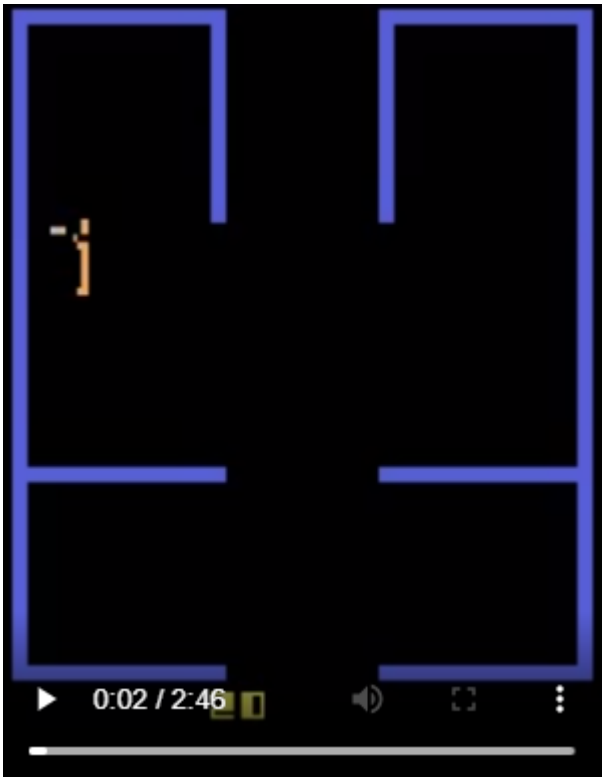

```
Eval reward: 610.0 (+/-73.48469228349535)
```

Παρατηρούμε πως το μοντέλο A2C το οποίο βέβαια εκπαιδεύτηκε για πιο πολύ χρόνο πετυχαίνει καλύτερα αποτελέσματα οπότε θα συνεχίσουμε την εκπαίδευση με αυτό το μοντέλο.

Βελτιστοποίηση Αλγορίθμου A2C και Στοχαστικότητα

Στην βελτιστοποίηση του αλγορίθμου A2C επιχειρήσαμε να αλλάξουμε διάφορες παραμέτρους του και να συγκρίνουμε την επιρροή αυτών. Το πρόβλημα στους agent που

εκπαιδεύουμε μέχρι στιγμής είναι κατά κύριο λόγο πως δεν κατανοούν την ανάγκη κίνησης τους όταν το πεδίο δράσης είναι χωρίς αντιπάλους:



Πεδίο χωρίς αντιπάλους

Πεδία με αντιπάλους

Δυστυχώς όλες οι προσπάθειες μας για καλύτερη εκπαίδευση πράκτορα φαίνονται να καταλήγουν στο κενό, οι agents δεν φαίνονται να μπορούν να κατανοήσουν το παιχνίδι παρά μόνο καταλήγουν συνήθως να πυροβολούν προς τυχαία κατεύθυνση.
Επιχειρήσαμε να μειώσουμε το learning rate κατά μια τάξη μεγέθους, να αυξήσουμε το learning rate κατά μια τάξη μεγέθους, να αλλάξουμε τον παράγοντα gamma, CnnPolicy και MlpPolicy και συνδυασμό αυτών.

Ενδεικτικά το evaluation που προκύπτει για μεγαλύτερο learning rate και μικρότερο gamma είναι:

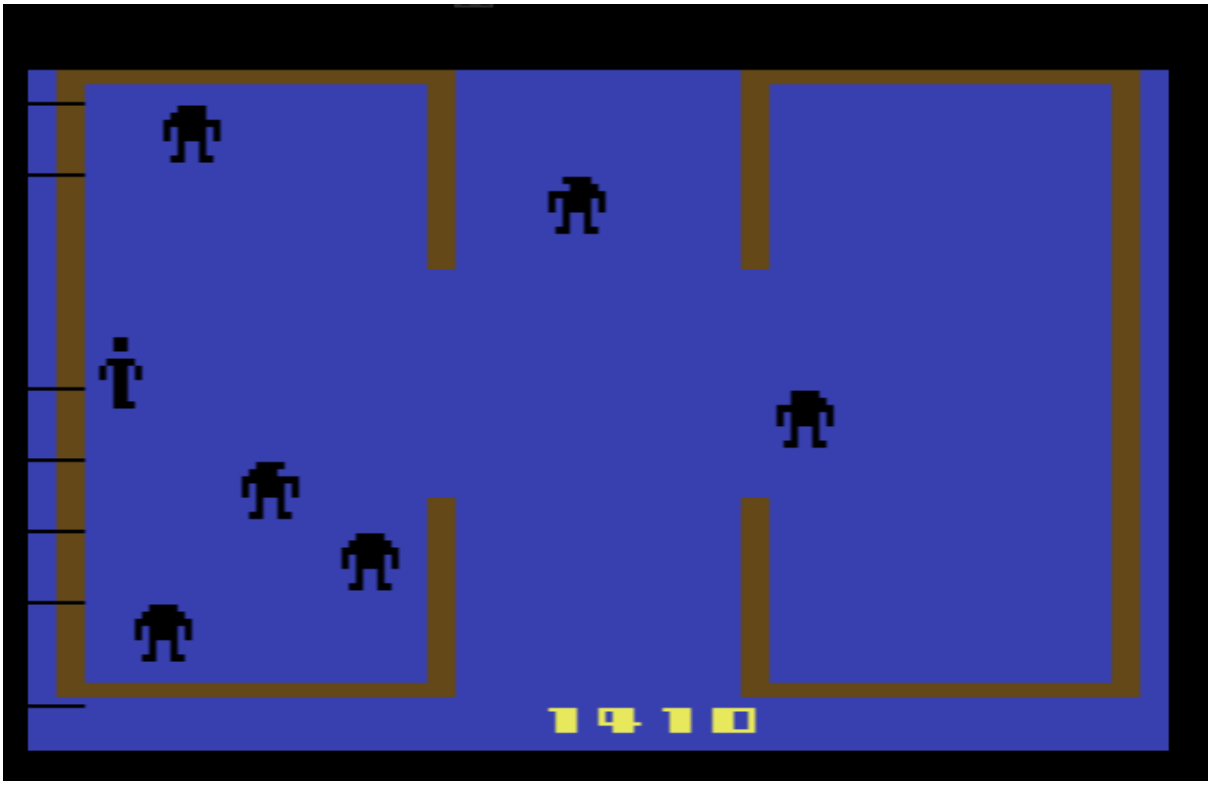
```
Eval reward: 555.0 (+/-127.37739202856997)
```

Αυτό φυσικά μας προδικάζει σε πολύ κακά αποτελέσματα και στην περίπτωση που παίζουμε το παιχνίδι με στοχαστικότητα. Ενδεικτικά εκπαιδεύσαμε ένα μοντέλο A2C με στοχαστικότητα για να συγκρίνουμε τα αποτελέσματα και δεν παρατηρήσαμε κάποια βελτίωση.

State-of-the-art

Συγκρίνοντας με τα state of the art αποτελέσματα παρατηρούμε πως τα μοντέλα μας έχουμε παρόμοια score με τα χαμηλότερα στο συνολικό ranking. Φυσικά με πιο εξελιγμένα μοντέλα και τεχνικές προκύπτουν όπως παρατηρούμε και 2 τάξης μεγέθους καλύτερα αποτελέσματα. Συμπέρασμα αυτών είναι η αρχική μας παρατήρηση πως το παιχνίδι Berzerk είναι ένα αρκετά δύσκολο παιχνίδι για reinforcement learning. Συγκρίνοντας με έναν πραγματικό

χρήστη παίξαμε εμείς το παιχνίδι και μετά από πολύ λίγες προσπάθειες πετύχαμε πολύ καλύτερα αποτελέσματα από τους agents μας.



Το καλύτερο score μας σε 5-6 προσπάθειες

Αλγόριθμος A2C

Ο βέλτιστος αλγόριθμος για το παιχνίδι μας είναι ο A2C.
Το policy gradient δίνεται από τη σχέση

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t]$$

Οι μέθοδοι vanilla policy gradient όμως παρουσιάζουν αστάθεια και αργή σύγκλιση, εξαιτίας της μεγάλης διακύμανσης στις log πιθανότητες και στις αθροιστικές reward values και της παρατήρησης ότι οι τροχιές έχουν αθροιστικό reward ισο με 0.
Ένας τρόπος βελτίωσης του Policy gradient είναι να μειώσουμε τη διακύμανση με χρήση ενός baseline, δηλαδή:

$$\text{Introducing baseline } b(s):$$
$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))]$$

Έτσι, θα έχουμε μικρότερο αθροιστικό reward, δηλαδή μικρότερα gradients και συνεπώς μικρότερες και πιο ευσταθείς ανανεώσεις. Το baseline που χρησιμοποιούμε μπορεί να παίρνει διάφορες μορφές.
Το vanilla policy gradient γράφεται ως εξής:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_0,a_0,\ldots,s_t,a_t}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)]\mathbb{E}_{r_{t+1},s_{t+1},\ldots,r_T,s_T}[G_t]$$

Επίσης ισχύει:

$$\mathbb{E}_{r_{t+1},s_{t+1},\ldots,r_T,s_T}[G_t] = Q(s_t, a_t)$$

Άρα, λαμβάνουμε:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_0,a_0,\ldots,s_t,a_t}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)]Q_w(s_t, a_t)$$

$$= \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)Q_w(s_t, a_t)]$$

Το Q value μπορεί να εκπαιδευτεί παραμετροποιώντας τη συνάρτηση Q με ένα νευρωνικό δίκτυο. Εδώ χρησιμοποιούνται οι μέθοδοι actor critic, με τις ακόλουθες παραδοχές:
Το critic υπολογίζει τη συνάρτηση value (Q value ή V value)
Το actor ανανεώνει την κατανομή του policy στην κατεύθυνση που προτείνεται από το critic.
Τόσο η critic οσο και η actor συνάρτηση παραμετροποιούνται με νευρωνικά δίκτυα και σε κάθε βήμα ανανεώνουμε τόσο τη μέθοδο critic όσο και το Value network.
Χρησιμοποιώντας την V function ως baseline function, ορίζουμε το advantage value, που αναφέρεται στο κατά πόσο είναι καλύτερο να κάνουμε μια συγκεκριμένη ενέργεια σε σύγκριση με τη μέση ενέργεια στη δεδομένη κατάσταση:

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t)$$

Από Bellman optimality equation, έχουμε μια σχέση για τα Q και V:

$$Q(s_t, a_t) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1})]$$

Και επομένως:

$$A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$$

Με αυτόν τον τρόπο, μπορούμε να χρησιμοποιήσουμε μόνο ένα νευρωνικό δίκτυο για τη συνάρτηση V, με ανανεώσεις:

$$\begin{aligned}\nabla_\theta J(\theta) &\sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)) \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)\end{aligned}$$

Ο παραπάνω αλγόριθμος είναι ο Advantage Actor Critic (A2C).
Έχουμε λοιπόν τη νέα εξίσωση ανανέωσης, στην οποία αντικαθιστούμε το μειωμένο αθροιστικό award του vanilla policy gradient με τη συνάρτηση Advantage:

$$\nabla_\theta J(\theta) \sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)$$

Σε κάθε βήμα της εκπαίδευσης, ανανεώνουμε τόσο την παράμετρο actor (με policy gradients και advantage value) όσο και την παράμετρο critic (ελαχιστοποιώντας το μέσο τετραγωνικό σφάλμα με την εξίσωση ανανέωσης Bellman).