ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Μεταπτυχιακή Εργασία

Εξερεύνηση των Δυνατοτήτων του Αλγορίθμου Ενισχυτικής Μάθησης Dreamerv3 στο Δυναμικό Περιβάλλον Παιχνιδιού Obstacle Tower

Βαγγέλης Τσιουρτής

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

12 Ιανουαρίου 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εξερεύνηση των Δυνατοτήτων του Αλγορίθμου Ενισχυτικής Μάθησης Dreamerv3 στο Δυναμικό Περιβάλλον Παιχνιδιού Obstacle Tower

Βαγγέλης Τσιουρτής

Επιβλέπων: Καθηγητής Χριστοδούλου Χρίστος

Δρ. Βασιλειάδης Βασίλης

Η Μεταπτυχιακή Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του μεταπτυχιακού Επιστήμη της Πληροφορικής του Τμήματος Πληροφορικής

12 Ιανουαρίου 2024

ΕΥΧΑΡΙΣΤΙΕΣ

Πρωτίστως, θα ήθελα να εκφράσω την εκτίμησή μου προς τον επιβλέποντα καθηγητή μου, Δρ. Χριστοδούλου Χρίστο, ο οποίος με εμπιστεύτηκε προσφέροντάς μου το θέμα αυτής της μεταπτυχιακής εργασίας. Επιπλέον, με υποστήριξε και με ενθάρρυνε σε όλα τα στάδια της.

Επίσης, εκφράζω τις ευχαριστίες μου προς τον Δρ. Βασιλειάδη Βασίλη για το ενδιαφέρον που έδειξε καθ' όλη τη διάρκεια του χρόνου, τις συμβουλές, τη στήριξη, και την απεριόριστη συνεισφορά του μέσα από μία εξαιρετική συνεργασία.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ την οικογένειά μου και τους φίλους μου, οι οποίοι παρέμειναν στο πλευρό μου καθ' όλη τη διάρκεια της διπλωματικής μου εργασίας, προσφέροντας συνεχή υποστήριξη.

ABSTRACT

Οι αλγόριθμοι ενισχυτικής μάθησης (RL) έχουν αναδειχθεί ως ισχυρά εργαλεία στον τομέα της τεχνητής νοημοσύνης, βρίσκοντας εκτεταμένη εφαρμογή στην επίλυση πολύπλοκων προβλημάτων, ιδίως στον τομέα των προσομοιώσεων παιχνιδιών. Σε αυτό το πλαίσιο, οι αλγόριθμοι μαθαίνουν να προβαίνουν σε αποφάσεις και να βελτιστοποιούν στρατηγικές μέσω της αλληλεπίδρασης με ένα δυναμικό περιβάλλον. Αυτή η συνεργασία όχι μόνο έχει ωθήσει την πρόοδο στις τεχνολογίες παιχνιδιών, αλλά έχει επίσης ανοίξει τον δρόμο για καινοτόμες λύσεις σε ευρύτερα προβλήματα σε τομείς όπως η ρομποτική και τα αυτόνομα συστήματα. Στο πλαίσιο των αλγορίθμων RL, ο Dreamer V3 ξεχωρίζει ως ένα προηγμένο μοντέλο με τη δυνατότητα να αντιμετωπίζει περίπλοκες σειριακές διαδικασίες λήψης αποφάσεων. Αυτή η μεταπτυχιακή μελέτη εξετάζει την εφαρμογή του αλγορίθμου Dreamer V3 στην αντιμετώπιση των πολύπλοκων προκλήσεων που παρουσιάζει το περιβάλλον παιχνιδιού του Obstacle Tower. Το Obstacle Tower παρουσιάζει ένα δυναμικό και πολυεπίπεδο περιβάλλον, με εμπόδια, δομές πολλαπλών επιπέδων και περιορισμούς χρόνου που απαιτούν ευφυή λήψη αποφάσεων. Μέσω μιας σειράς εκτενών πειραμάτων, αξιολογούμε την αποτελεσματικότητα του Dreamer V3 στην πλοήγηση στο Obstacle Tower, επιδεικνύοντας την προσαρμοστικότητα και την αξιοπιστία του. Παράλληλα, βάσει κριτηρίων, όπως η υψηλότερη τελική απόδοση, η αποδοτικότητα των δεδομένων, καθώς και ο μέσος αριθμός ορόφων και το μέσο reward, εξετάζουμε εάν ο αλγόριθμος έχει τη δυνατότητα να υπερβεί την απόδοση άλλων προηγούμενων αλγορίθμων στο ίδιο περιβάλλον. Τα αποτελέσματά μας αναδεικνύουν την προσαρμοστικότητα και την επεκτασιμότητά του αλγορίθμου Dreamer V3 και, συγκεκριμένα, φανερώνουν ότι μεγαλύτερα μοντέλα και Training Ratio=16 συνδέονται με υψηλότερη αποδοτικότητα. Επίσης, ο συνδυασμός Large μοντέλου με Training Ratio=16 και first-person perspective επιφέρει τα καλύτερα αποτελέσματα. Επίσης, η μείωση των κινήσεων στο Action space βελτιώνει την αποδοτικότητα δεδομένων, επιφέρει μία από τις υψηλότερες τελικές τιμές ενω ταυτόχρονα απαιτεί

και λιγότερο χρόνο εκτέλεσης για να επιτύχει. Αυτή η έρευνα συνεισφέρει στον εξελισσόμενο χώρο της ενισχυτικής μάθησης, παρέχοντας ενδιαφέρουσες προοπτικές στη δυνατότητα των προηγμένων αλγορίθμων όπως το Dreamer V3 να ξεχωρίζει σε δυναμικά και πολύπλοκα εικονικά περιβάλλοντα, και επισημαίνοντας τις ευρύτερες του επιπτώσεις για εφαρμογές στον πραγματικό κόσμο που απαιτούν προηγμένες ικανότητες λήψης αποφάσεων.

Περίληψη Σημειογραφίας

s_t	state at time t
a_t	action at time t or an action
r_t	reward at time t
$\mathcal{S}$	set of all nonterminal states
$\mathcal{S}^+$	set of all states (including terminal states)
$\mathcal{A}(s)$	set of all actions available in state s
$\mathcal{R}$	set of all rewards
t	discrete time step
$T, T(t)$	final time step of an episode, or of the episode including time step t
S_t	state at time t
A_t	action at time t
R_t	reward at time t
G_t	discounted return at time t ($\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$)
γ	discount rate (where $0 \leq \gamma \leq 1$)
ε	probability of taking a random action in an ε-greedy policy
η	learning rate
γ	discount-rate parameter
λ	decay-rate parameter for Generalized Advantage Estimation and N-step returns

$\in$ is an element of; e.g., $s \in S$, $r \in R$

π policy

$\pi(s)$ action taken in state s under deterministic policy π

$\pi(a|s)$ probability of taking action a in state s under stochastic policy π

$p(s', r|s, a)$ probability of next state s' and reward r, given current state s and action a ($\mathbb{P}(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$)

$v_\pi(s)$ state-value function for policy π ($v_\pi(s) \doteq \mathbb{E}[G_t|S_t = s]$ for all $s \in \mathcal{S}$)

$v_*(s)$ optimal state-value function ($v_*(s) \doteq \max_\pi v_\pi(s)$ for all $s \in \mathcal{S}$)

q_π action-value function for policy π ($q_\pi(s, a) \doteq \mathbb{E}[G_t|S_t = s, A_t = a]$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)

$q_*(s)$ optimal action-value function ($q_*(s, a) \doteq \max_\pi q_\pi(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)

$\mathbb{E}[X]$ expectation of a random variable X, i.e., $\mathbb{E}[X] = \sum_x p(x)x$

$\mathbb{R}$ set of real numbers

$|S|$ number of elements in set S

h horizon, the time step one looks up to in a forward view

∂ parameter vector of target policy

∂_k parameter vector of target policy at the kth optimization step

$\pi(a|s; \partial)$ probability of taking action a in state s given parameter vector ∂

π_∂ policy corresponding to parameter (vector) ∂

$\nabla\pi(a|s; \partial)$ column vector of partial derivatives of $\pi(a|s; \partial)$ with respect to ∂

$J_\partial(\cdot)$ Objective of policy π_∂ evaluated at the input

$\nabla J_\partial(\cdot)$ column vector of partial derivatives of J_∂ with respect to ∂ evaluated at the input

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή (Introduction)

1.1 Θεωρητικό Υπόβαθρο

Εν όψει της συνεχούς, ραγδαίας εξέλιξης στον τομέα της Μηχανικής Μάθησης (ΜΜ) και συγκεκριμένα στο πεδίο της Ενισχυτικής Μάθησης (ΕΜ), έχουν επιτευχθεί πρωτόγνωρες επιδόσεις από τους υπολογιστές σε διάφορους τομείς, όπως το σκάκι [1], το Dota 2 [2], και το GO [3], οι οποίες ξεπερνούν ακόμη και τον ανθρώπινο παράγοντα.

Ένας πρωτοποριακός αλγόριθμος βασισμένος σε μοντέλα στον τομέα της Ενισχυτικής Μάθησης (ΕΜ) θεωρείται ο DreamerV3 [4], αρχικά λόγω της ιδιότητάς του ως γενικευμένος και επεκτάσιμος αλγόριθμος, αλλά και λόγω του γεγονότος ότι ξεπερνά το πρόβλημα της ανάγκης τεράστιων υπολογιστικών πόρων και της κατοχής μεγάλης εμπειρικής γνώσης για την ρύθμιση (tuning) των υπερπαραμέτρων των αλγορίθμων μέσω της παροχής των σταθερών υπερπαραμέτρων του. Επίσης, ο DreamerV3 είναι ικανός να αντιμετωπίζει μακροπρόθεσμα προβλήματα (long-horizon tasks) μέσω της ¨φαντασίας,¨ η οποία αναδύεται μέσω του "world model," [5] στο οποίο κάνει χρήση ο DreamerV3. Το world model προβλέπει τα μελλοντικά αποτελέσματα πιθανών ενεργειών, δημιουργώντας ένα εσωτερικό μοντέλο του περιβάλλοντος. Αυτή η δυνατότητα πρόβλεψης επιτρέπει στον αλγόριθμο να λαμβάνει αποφάσεις με μακροπρόθεσμες επιδράσεις, δημιουργώντας τροχιές, μέσω των οποίων θα επιτυγχάνονται καλύτερες αποδόσεις σε ποικίλες και περίπλοκες καταστάσεις.

Ο DreamerV3 έχει δοκιμαστεί και αξιολογηθεί σε ένα ευρύ φάσμα περιβάλλοντα με διαφορετικά χαρακτηριστικά, όπως διακριτά και συνεχή περιβάλλοντα, περιβάλλοντα χαμηλών και υψηλών διαστάσεων εισόδου, 2D και 3D περιβάλλοντα, καθώς και σε αραιές και πυκνές ανταμοιβές, επιδεικνύοντας υψηλότερη απόδοση από άλλους εξειδικευμένους αλγορίθμους - είτε βασισμένους σε μοντέλα είτε όχι - χωρίς την ανάγκη συντονισμού (tuning) τους [6]. Αυτές οι υψηλές αποδόσεις απέδειξαν ότι ο DreamerV3 έχει ξεπεράσει το εμπόδιο της εξειδικευμένης γνώσης, καθιστώντας τον ευρέως εφαρμόσιμο σε δύσκολα προβλήματα λήψης αποφάσεων [6].

Στην προσπάθεια τους για την αξιολόγηση των νέων υποσχόμενων αλγορίθμων της ΕΜ, οι ερευνητές αναζητούν όλο και πιο περίπλοκα βιντεοπαιχνίδια για να σπρώξουν τους αλγορίθμους στα όρια τους, με απώτερο στόχο την απόδειξη των πραγματικών δυνατοτήτων τους. Ένα τέτοιο παιχνίδι είναι το Obstacle Tower, το οποίο θεωρείται περίπλοκο, δυναμικό και τρισδιάστατο. Αποτελείται από 99 ορόφους, όπου ο κάθε όροφος παρουσιάζει ένα ευρύ φάσμα προκλήσεων (π.χ. πλοήγηση σε πλατφόρμες με διαφορετικά ύψη, κινούμενες πλατφόρμες, στενά περάσματα και εμπόδια που απαιτούν ακριβή άλματα, αποφυγή εχθρών κλπ.) [6]. Σε κάθε επεισόδιο παρατηρείται αλλαγή στη διάταξη των ορόφων του Obstacle Tower, διασφαλίζοντας έτσι την ικανότητα προσαρμογής και γενίκευσης του πράκτορα. Το Obstacle Tower εν τέλει προσφέρει δίκαιες συγκρίσεις, ενώ ταυτόχρονα αποκαλύπτει πληροφορίες σχετικά με τα δυνατά και τα αδύνατα σημεία των διαφόρων αλγορίθμων.

Μέχρι στιγμής, οι καλύτεροι αλγόριθμοι που έχουν εφαρμοστεί στο περιβάλλον Obstacle Tower έχουν επιτύχει να επιλύσουν πάνω από 10 ορόφους, με τον καλύτερο αλγόριθμο να επιλύει κατά μέσο όρο 19.4 ορόφους και μέσο όρο του reward 35.86, απόδοση που είναι παρόμοια με την απόδοση έμπειρων ανθρώπων [7]. Για να επιτύχουν υψηλότερες αποδόσεις, απαιτείται η χρήση νέων αλγορίθμων με θεμελιώδεις βελτιώσεις.

1.2 Στόχος Μεταπτυχιακής Εργασίας

Ο στόχος αυτής της διπλωματικής εργασίας είναι διπλός: 1) να εξερευνήσει και αξιολογήσει την απόδοση του αλγορίθμου DreamerV3 στις προκλήσεις που περιλαμβάνει το περιβάλλον του παιχνιδιού Obstacle Tower, 2) να διερευνήσει κατά πόσο η χρήση αυτού του αλγορίθμου μπορεί να φτάσει και να ξεπεράσει τον μέσο αριθμό ορόφων και το μέσο του ŕeward που κατάφεραν άλλοι κορυφαίοι αλγόριθμοι μέχρι τώρα στο περιβάλλον Obstacle Tower.

Για την επίτευξη αυτού του στόχου, διεξάγονται πειράματα όπου προσαρμόζονται οι εξής παράμετροι του αλγορίθμου DreamerV3: το Training Ratio, το Model Size, καθώς και οι παράμετροι Agent Perspective και το Action Space του Obstacle Tower.

Συνολικά πραγματοποιούνται 9 πειράματα με διάφορες ρυθμίσεις. Για την ανάλυση των αποτελεσμάτων, πραγματοποιείται σύγκριση με βάση την υψηλότερη τελική απόδοση (higher final performance), την αποδοτικότητα των δεδομένων (higher data-efficiency), καθώς και τον μέσο αριθμό ορόφων και το μέσο reward.

1.3 Αναμενόμενα Αποτελέσματα

Αναμένουμε ότι τα αποτελέσματα από την εφαρμογή του αλγορίθμου DreamerV3 στο περιβάλλον θα παρέχουν μια σαφή εικόνα της απόδοσής του έναντι των προκλήσεων του Obstacle Tower, καθώς γίνονται μεταβολές στις παραμέτρους του DreamerV3, όπως το Training Ratio, το Model Size, καθώς και οι παράμετροι Agent Perspective και το Action Space του Obstacle Tower. Επιπλέον, προσδοκούμε έναν υψηλότερο μέσο όρο, περίπου 19.4 ορόφους, και ένα μέσο όρο του reward περίπου 35.86.

Τα αποτελέσματα αυτής της μελέτης θα παράσχουν πολύτιμες πληροφορίες για την αποτελεσματικότητα του νέου αλγορίθμου DreamerV3 σε πολύπλοκα και δυναμικά περιβάλλοντα όπως ο Obstacle Tower. Επίσης τα αποτελέσματα αναμένεται να προσκομίσουν αναγνώριση πιθανών ζητημάτων που μπορεί να προκύψουν κατά την εφαρμογή του DreamerV3 στο Obstacle tower. Τα ευρήματα αυτα θα συμβάλουν στην προώθηση του πεδίου της ΕΜ και θα ρίξουν φως στις δυνατότητες του αλγόριθμου για εφαρμογές πραγματικού κόσμου που απαιτούν έξυπνη λήψη αποφάσεων σε πολύπλοκα σενάρια (πχ.ρομποτικός πλοήγησης).

Λέξεις-κλειδιά : ενισχυτική μάθηση, DreamerV3, Obstacle tower, σύνθετο περιβάλλον, βάσει μοντέλου, χωρίς μοντέλα, μετρήσεις απόδοσης.

Κεφάλαιο 2

Ανασκόπηση της Βιβλιογραφίας (Literature review)

2.1 Εισαγωγή

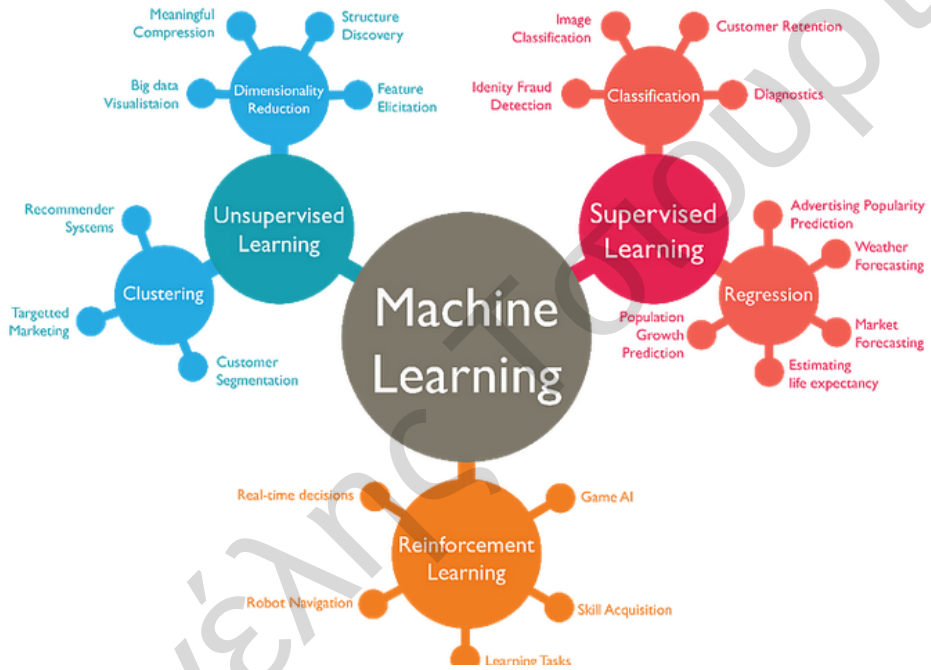
Στο συγκεκριμένο κεφάλαιο, πραγματοποιείται μια εκτενής βιβλιογραφική ανασκόπηση, ξεκινώντας με την παρουσίαση βασικών έννοιών και αρχών τόσο της ενισχυτικής μάθησης (ΕΜ) όσο και της βαθιάς μάθησης (ΒΜ), που είναι απαραίτητες για την μετέπειτα κατανόηση του υπόλοιπου περιεχομένου. Το κεφάλαιο 2.2 επικεντρώνεται στην ΕΜ, εξετάζοντας βασικά στοιχεία, διακριτά περιβάλλοντα, Μαρκοβιανές Διαδικασίες Απόφασης, Exploitation και Exploration, εξισώσεις Bellman, καθώς και διάφορες μέθοδοι ΕΜ. Ακολούθως, το κεφάλαιο 2.3 εισάγει και εμβαθύνει στη ΒΜ, καλύπτοντας ιστορική εξέλιξη, βασικά στοιχεία νευρωνικών δικτύων, συμπεριλαμβανομένων των Perceptrons και των συναρτήσεων ενεργοποίησης, καθώς και διάφορα Φραμεωορκς και μεθόδους εφαρμογής της ΒΜ.

Τα κεφάλαια 2.4 και 2.5 είναι αφιερωμένα στην παρουσίαση και λεπτομερή ανάλυση των κύριων αλγορίθμων ΒΕΜ, περιλαμβάνοντας TRPO, PPO, DQN, Rainbow DQN, DDPG, TD3, και SAC. Το τελευταίο αλλά εξίσου σημαντικό κεφάλαιο, 2.6, παρέχει ποικίλα παραδείγματα εφαρμογών αλγορίθμων ΒΜ και ΒΕΜ σε παιχνίδια όπως τα ντάμα, σκάκι, Atari Games, Go, Dota 2, και Obstacle Tower.

2.2 Ενισχυτική Μάθηση (Reinforcement Learning)

Η Ενισχυτική Μάθηση (ΕΜ) είναι μία εκ των τρίων κύριων αλγοριθμικών κατηγοριών της Μηχανικής Μάθησης (ΜΜ) η οποία διαχωρίζεται με βάση τον τρόπο που μαθαίνει η κάθε μία διαφορετικά [8].

1. Επιβλεπόμενη Μάθηση (Supervised Learning)
2. Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)
3. Ενισχυτική Μάθηση (Reinforcement Learning)



Σχήμα 2.1: Διαγραμμα διαχωρισμού της Μηχανικής Μάθησης στα υποπεδία της [9].

Η Ενισχυτική Μάθηση (ΕΜ) είναι κατά κάποιον τρόπο εμπνευσμένη από τον τρόπο που μαθαίνουν οι άνθρωποι και τα ζώα, μέσα από την διαδικασία δοκιμής και λάθους (trail and error) η οποία εμφανίζεται στην προσπάθεια τους να δοκιμάσουν νέες στρατηγικές με αποτέλεσμα να απορρίπτονται επιλογές που είναι λανθασμένες ή δεν επιφέρουν το επιθυμητό αποτέλεσμα (δεν οδηγούν σε υψηλότερες αποδόσεις) [10].

Το κυριότερο πρόβλημα είναι να εφαρμόσουμε ένα τρόπο ελέγχου στο σύστημα (πράκτορα) ώστε να μεγιστοποιηθεί η συνολική ανταμοιβή όταν επιτύχει τον στόχο του. [11]

2.2.1 Βασικά Στοιχεία

1. **Χώρος Καταστάσεων - State Space**

   Είναι το σύνολο που αποτελείται από όλες τις πιθανές καταστάσεις στις οποίες μπορεί να μεταβεί ο πράκτορας αλληλοεπιδρώντας με το περιβάλλον. Ο χώρος κατάστασης τυπικά ορίζεται από το περιβάλλον.

2. **Χώρος Ενεργειών - Action Space**

   Είναι το σύνολο που αποτελείται από όλες τις πιθανές ενέργειες που μπορεί να επιλέξει να εκτελέσει ο πράκτορας σε κάθε μία συγκεκριμένη κατάσταση. Ο χώρος δράσης τυπικά ορίζεται από το περιβάλλον.

3. **Τροχιά - Trajectory-Episodes**

   Μια τροχιά (trajectory) τ είναι μια ακολουθία η οποία εμπεριέχει τις καταστάσεις στις οποίες μεταβαίνει ο πράκτορας καθώς και τις ενέργειες τις οποίες λαμβάνει ο πράκτορας από αυτές τις καταστάσεις. Η μορφή που ακολουθεί η τροχία είναι κατάσταση→ενεργειών→νέα κατάσταση:

   $$\tau = (s_0, a_0, s_1, a_1, \ldots).$$

   Η αρχική κατάσταση s_0 λαμβάνεται από μια κατανομή κατάστασης έναρξης μ: $s_0 \sim m(;)$.

   Οι μεταβάσεις κατάστασης εξαρτώνται μόνο από την πιο πρόσφατη κατάσταση και δράση. Θα μπορούσαν να είναι:

   (α΄) ντετερμινιστική:
   $s_{(t+1)} = f(s_t, a_t),$

   (β΄) ή στοχαστική:
   $s_{(t+1)} \sim P(\cdot|s_t, a_t)$

   Το τι επιστρέφει μία τροχιά (trajectory) είναι το μέτρο της σωρευτικής ανταμοιβής (cumulative reward)

Στόχος της ενισχυτικής μάθησης όπως έχει αναφερθεί προηγουμένως, είναι η μεγιστοποίηση του ληφθέντος ποσού της σωρευτικής ανταμοιβής σε μακροπρόθεσμη βάση. Σύμφωνα με την παραπάνω θεώρηση και συμβολίζοντας τις ανταμοιβές μετά από κάθε χρονική στιγμή τ με $r_{(t+1)}, r_{(t+2)}, \ldots$, τότε ο πράκτορας προσπαθεί να μεγιστοποιήσει την αναμενόμενη επιστροφή (expected return) T είναι το τελικό (τερματικό) χρονικό βήμα. Υπάρχουν δύο κύριοι τρόποι υπολογισμού της επιστροφής ανάλογα με το τύπο της εφαρμογής:

(α΄) Finite horizon undiscounted sum of rewards

Οταν η αλληλεπίδραση μεταξύ πράκτορα και περιβάλλοντος διαχωρίζεται σε υποακολουθίες που καλούνται τροχίες ή επεισόδια (επισοδες) τα οποία έχουν τερματικό σημείο τότε οι ανταμοιβές μπορούν να εκφραστούν ώς:

$$R(\tau) = \sum_{t=0}^{\tau} r_t$$

(β΄) Infinite horizon discounted sum of rewards

Οταν η αλληλεπίδραση μεταξύ πράκτορα και περιβάλλοντος δέν διαχωρίζεται σε υποακολουθίες που καλούνται τροχίες ή επεισόδια (επισοδες) αλλά συνεχίζει επ'άπειρον τότε η ανταμοιβές μπορούν να εκφραστούν ώς

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

Όπου $\gamma \in (0, 1)$ ο συντελεστής μείωσης ανταμοιβής. Το πώς θα οριστεί το γ καθορίζει εάν πράκτορας βλέπει μυοπικά ή όχι. Αν $\gamma = 0$ τότε ο πράκτορας καθιστά τις μελλοντικές ανταμοιβές μικρότερης αξίας και συνεπώς δίνει μεγαλύτερη βαρύτητα στις ανταμοιβές που λαμβάνονται άμεσα παρά σε αυτές που έρχονται μελλοντικά. Αντίθετα αν το γ πλησιάζει στο 1 τότε ο πράκτορας καθιστά τις μελλοντικές ανταμοιβές μεγαλύτερης αξίας και συνεπώς δίνει μεγαλύτερη βαρύτητα στις ανταμοιβές που λαμβάνονται μελλοντικά παρά τις άμεσες.

4. **Πολιτική**(π) **- Policy**

Είναι μία συνάρτηση που ορίζει την συμπεριφορά του πράκτορα σε κάθε κατάσταση στην οποία θα βρεθεί μέσα στο περιβάλλον [12]. Εμμέσως πλην σαφώς υπαγορεύει στο πράκτορα ποιά είναι η βέλτιστη ενέργεια που θα εκτελέσει όταν βρίσκεται σε μία

συγκεκριμένη κατάσταση με στόχο να μεγιστοποιήσει την συνολική σωρευτική ανταμοιβή [13]. Η πολιτική μπορεί να θεωρηθεί ως μία κατανομή πιθανοτήτων πάνω στις διαθέσιμες ενέργειες που μπορεί να λάβει ο πράκτορας σε κάθε κατάσταση που βρίσκεται. Μια πολιτική είναι μια συνάρτηση που μπορεί να είναι είτε ντετερμινιστική είτε στοχαστική.

5. **Συνάρτηση Αξίας Ενέργειας-Action value Function(Q-Function)**

Η Συνάρτηση αξίας ενέργειας Q(s,a) εκτιμά το αναμενόμενο άθροισμα ανταμοιβών που μπορεί να λάβει ένας πράκτορας ξεκινώντας από μια συγκεκριμένη κατάσταση ($S_t = s$) εκτελώντας αρχικά μία ενέργεια ($A_t = a$) και ακολουθώντας μια συγκεκριμένη πολιτική π. Στην ουσία η συνάρτηση αξίας ενέργειας επιστρέφει το αναμενόμενο άθροισμα ανταμοιβής ενός ζεύγους κατάστασης ($S_t = s$)- ενέργειας ($A_t = a$) [12].

6. **Συνάρτηση Αξίας Κατάστασης-State-value Function(V-function)**

Η Συνάρτηση αξίας κατάστασης V(s) εκτιμά το μελλοντικό αναμενόμενο συσσωρευτικό άθροισμα ανταμοιβών που μπορεί να λάβει ένας πράκτορας ξεκινώντας από μια συγκεκριμένη κατάσταση ($S_t = s$) και ακολουθώντας μια συγκεκριμένη πολιτική π εκτελώντας μία συγκεκριμένη ενέργεια α $A_t = a$. Μπορεί να θεωρηθεί και ώς η αξία της κάθε κατάστασης [12].

$$V(s) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1})|S_t = s] \tag{2.1}$$

7. **Συνάρτηση Πλεονεκτήματος-Advantage function** $A(s, a)$

Η Συνάρτηση Πλεονεκτήματος (ΣΠ) $A(s, a)$ χρησιμοποιείται για να αξιολογηθεί πόσο καλή είναι η λήψη μίας συγκεκριμένης ενέργειας a από μία συγκεκριμένη κατάσταση s σε σχέση με τη μέση τιμή που θα επέστρεγε το σύνολο όλων των διαθέσιμων ενεργειών στην κατάσταση s. Στην ουσία η ΣΠ αξιολογεί την κάθε πιθανή λήψη ενέργειας με απώτερο στόχο την ελαχιστοποιηση των απωλειών ή την αύξηση των κέρδων.

$$A(s, a) = Q(s, a) - V(s) \tag{2.2}$$

Όπου:

(α') $A(s, a)$ είναι η ΣΠ για την ενέργεια a στην κατάσταση s.

(β') $Q(s, a)$ είναι το Q-Function για την ενέργεια a στην κατάσταση s.

(γ') $V(s)$ είναι η εκτιμημένη αξία V-function) της κατάστασης s, δηλαδή η μέση τιμή που επιστρέφει το σύνολο όλων των πιθανών ενεργειών στην κατάσταση s.

8. **Συνάρτηση ανταμοιβής- Reward Function**

Όταν ένας πράκτορας εκτελέσει μια συγκεκριμένη ενέργεια σε μια συγκεκριμένη κατάσταση λαμβάνει μια ανατροφοδότηση από το περιβάλλον, μία ανταμοιβή. Εδώ ο όρος «ανταμοιβή» περιγράφει την ανατροφοδότηση που λαμβάνει από το περιβάλλον ο πράκτορας μετά από κάθε ενέργεια που λαμβάνει σε αυτό. Μια ανταμοιβή μπορεί να είναι θετική ή αρνητική. Όταν η ανταμοιβή είναι θετική συνεπάγεται ότι η ενέργεια που επέλεξε να εκτέλεσε σε μία συγκεκριμένη κατάσταση ο πράκτορας ήταν ορθή. Όταν η ανταμοιβή είναι αρνητική συνεπάγεται ότι η ενέργεια που επέλεξε να εκτέλεσε ο πράκτορας σε μία συγκεκριμένη κατάσταση ήταν λανθάνουσα.

Η συνάρτηση ανταμοιβής υπολογίζει την συνολική ανταμοιβή που αποκτά ο πράκτορας από τις ενέργειες σε μια τροχιά. Επειδή η απόδοση του πράκτορα βασίζεται εξ' ολοκλήρου στην ανταμοιβή που κερδίζεται από τις ενέργειές του, είναι απαραίτητο να υπάρχει η καλύτερη δυνατή λειτουργία ανταμοιβής.

2.2.2 Διακριτά Περιβάλλοντα Vs Συνεχές Περιβάλλοντα

Στην ΕΜ, ο Χώρος Καταστάσεων (State Space) και Χώρος Δράσης (Action Space) - μπορεί να διακριθεί είτε ως διακριτός είτε ως συνεχές. Αυτός ο διαχωρισμός οφείλεται στο γεγονός ότι η διακριτικότητα και η συνέχεια παρέχουν διαφορετικούς τύπους προβλημάτων με διαφορετικούς περιορισμούς και επιπτώσεις στους αλγορίθμους που θα εφαρμοστούν σε αυτά. Ανάλογα στο σε πια κατηγορία εμπίπτει ο χώρος κατάστασης και δράσης καθορίζεται και ο αντίστοιχος τρόπος σχεδιασμού του αλγορίθμου της ΕΜ.

Ακολουθεί μια εξήγηση των διαφορών μεταξύ διακριτών και συνεχών χώρων στην ΕΜ:

1. **Διακριτός χώρος**

   Σε έναν διακριτό χώρο, ο πράκτορας έχει ένα πεπερασμένο (μετρήσιμο) σύνολο πιθανών ενεργειών. Για παράδειγμα στο σκάκι υπάρχει ένας πεπερασμένος αριθμός πιθανών κινήσεων σε οποιαδήποτε κατάσταση και αν βρεθεί ο πράκτορας. Γενικά ο χειρισμός των διακριτών χώρων θεωρείται σχετικά πιο εύκολα διαχειρίσιμος, λόγω του ότι υπάρχει η δυνατότητα χρήσης τεχνικών όπως το Q-learning ή οι μέθοδοι πινάκων (tabular) για την εκτίμηση των τιμών για κάθε ζεύγος ενέργειας-κατάστασης.

2. **Συνεχής χώρος**

   Σε έναν συνεχές χώρο, ο πράκτορας μπορεί να επιλέξει από έναν άπειρο αριθμό πιθανών ενεργειών. Για παράδειγμα, ο χειρισμός της κίνησης ενός ρομποτικού βραχίονα μέσω των γωνιών που υπάρχουν στις αρθρώσεις του μπορεί να θεωρηθεί ως ένας συνεχές χώρος. Γενικά ο χειρισμός των συνεχών χώρων θέτουν υπολογιστικές και μαθηματικές προκλήσεις. Οι παραδοσιακοί μέθοδοι (π.χ. πινάκων (Tabular), Q learning) που εφαρμόζονται με ευκολία σε διακριτά περιβάλλοντα, δεν είναι κατάλληλοι στους συνεχείς χώρους και η προσέγγιση των τιμών Χ ή των πολιτικών γίνεται πιο περίπλοκη. Συχνά χρησιμοποιούνται αλγόριθμοι όπως οι Gradient Policy.

Συνοψίζοντας και αναφέροντας μαθηματική αναπαράσταση

Οι χώροι δράσης και χώροι καταστάσεων είναι είτε διακριτοί ή συνεχής.

1.Διακριτός Action space:

Το σύνολο των πιθανών ενεργειών συμβολίζεται ως Α και είναι πεπερασμένο ή μετρήσιμο.

$$A = a_1, a_2, \ldots, a_N$$

όπου:

1. a_i αντιπροσωπεύει την κάθε διακριτή ενέργεια ξεχωριστά
2. Ν είναι ο συνολικός αριθμός των πιθανών ενεργειών στο χώρο

Η Action value Function (συνάρτηση Q) για έναν διακριτό Action space μπορεί να οριστεί ως:

$$Q(s, a) = E[\, R_t s_t = s, a_t = a]$$

όπου:

1. Q(s,a) αντιπροσωπεύει την αναμενόμενη σωρευτική ανταμοιβή λαμβάνοντας μία ενέργεια a στην κατάσταση s
2. Ε την αναμενόμενη σωρευτική ανταμοιβή για όλη την ακολουθία μίας πολιτικής.

2. Διακριτό State space:

Το σύνολο των πιθανών καταστάσεων που συμβολίζεται ως S και είναι πεπερασμένο ή μετρήσιμο.

$$S = s_1, s_2, \ldots, s_M$$

όπου:

1. s_i αντιπροσωπεύει μεμονωμένες διακριτές καταστάσεις
2. M είναι ο συνολικός αριθμός των πιθανών καταστάσεων στον χώρο.

Η συνάρτηση State-value (συνάρτηση V) για έναν διακριτό χώρο κατάστασης μπορεί να οριστεί ως:

$$V_(s) = E[R_t s_t = s]$$

όπου:

1. V(s) αντιπροσωπεύει την αναμενόμενη σωρευτική ανταμοιβή όταν ξεκινάμε από μία συγκεκριμένη κατάσταση s
2. Ε την αναμενόμενη σωρευτική ανταμοιβή για όλη την ακολουθία μίας πολιτικής.

3. Συνεχές Action space:

Το σύνολο των πιθανών ενεργειών συμβολίζεται ως A και αναπαρίσταται ως διάνυσμα πραγματικών αριθμών.

$$A = Rn$$

όπου:

1. R αντιπροσωπεύει το σύνολο των πραγματικών αριθμών
2. n είναι η διάσταση του χώρου συνεχούς δράσης.

4. Συνεχές State space:

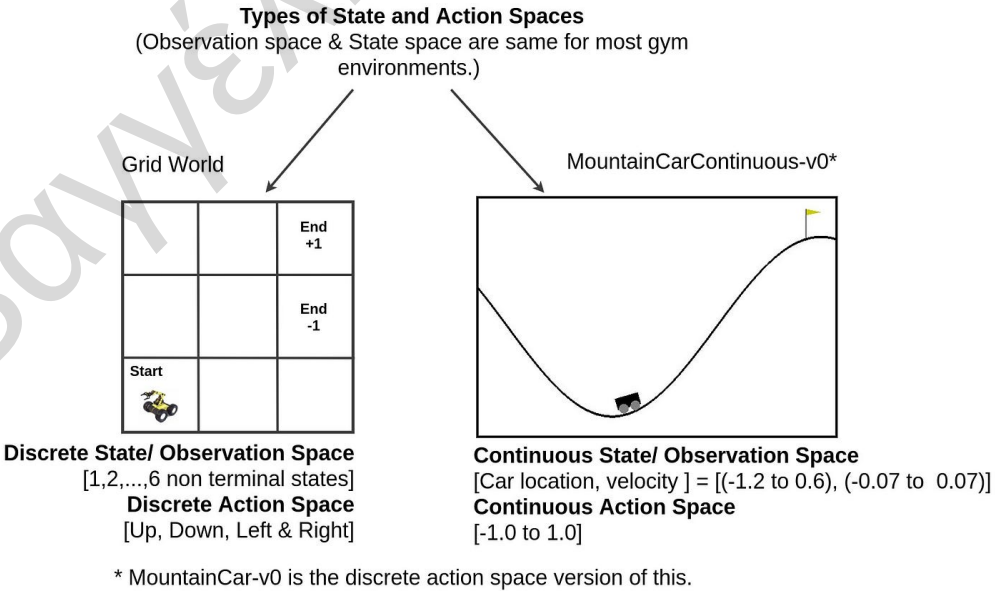
Το σύνολο των πιθανών καταστάσεων συμβολίζεται ως S και αναπαρίσταται ως διάνυσμα πραγματικών αριθμών.

$$S = Rm$$

όπου:

1. R αντιπροσωπεύει το σύνολο των πραγματικών αριθμών
2. m είναι η διάσταση του χώρου συνεχούς κατάστασης

Οι αλγόριθμοι ΕΜ μπορούν να χειριστούν τόσο διακριτούς όσο και συνεχείς χώρους κατάστασης και η επιλογή εξαρτάται από το συγκεκριμένο πρόβλημα που αντιμετωπίζεται.



Σχήμα 2.2: Αριστερά απεικονίζετε ένα διακριτό περιβάλλον ενώ στα δεξιά ένα συνεχές περιβάλλον. Για το κάθε περιβάλλον παρουσιάζεται το Observation Space και το Action Space του, καθώς και οι πιθανές τιμές που μπορεί να λάβει το κάθε ένα [9]

2.2.3 Exploitation and Exploration

Στο Exploitation (εκμετάλλευση) ο πράκτορας κάνει επιλογή ενεργειών οι οποίες μεγιστοποιούν την αναμενόμενη άμεση αθροιστική ανταμοιβή για την τρέχουν κατάσταση στην οποία βρίσκεται βάση της τρέχουσας γνώσης που αποκτήθηκε από προηγουμένη εμπειρία (Greedy action).

Σε αντίθεση στην εξερεύνηση (Exploration) ο πράκτορας κάνει επιλογή ενεργειών που δεν έχουν επιλεχθεί προηγουμένως, δηλαδή είναι πρωτόγνωρες για τον πράκτορα και συνεπώς δεν υπάρχει προϋπάρχουσα γνώση. Ο πράκτορας επιλέγει ενέργειες που μπορεί να μην έχουν υψηλή αναμενόμενη ανταμοιβή, αλλά έχουν τη δυνατότητα να παρέχουν πολύτιμες νέες πληροφορίες για το περιβάλλον. Στο exploration η άμεση αθροιστική ανταμοιβή μπορεί να μην είναι η βέλτιστη λόγω της επιλογής μία άγνωστης ενέργειας αλλά μπορεί να δώσει καλύτερη απόδοση μακροπρόθεσμα.

Όταν το περιβάλλον στο οποίο ο πράκτορας θα αλληλεπιδράσει είναι άγνωστο θεωρείτε μεγάλο ζήτημα μεταξύ Exploitation και Exploration καθώς

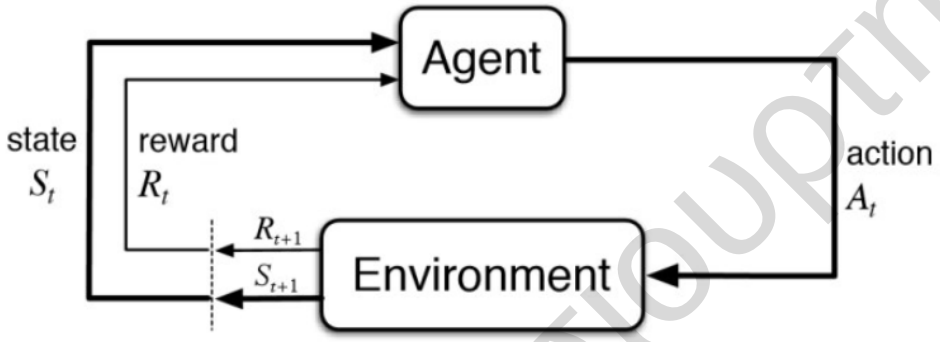
1. χωρίς αρκετό Exploration, δεν μπορούμε να μάθουμε το περιβάλλον αρκετά καλά [13].Σημ(Το Exploration βοηθά κυρίως όταν απαιτείτε μεγιστοποίηση των άμεσων ανταμοιβών, αλλά η υπερβολική χρήση του Exploration μπορεί να μας αποτρέψει στο να επιτύχουμε καύτερα αποτελέσματα μακροπρόθεσμα αφού λειτουργεί μυωπικά). [14]

2. χωρίς αρκετό Exploitation, δεν μπορούμε να επιτύχουμε μεγιστοποίηση των άμεσων ανταμοιβών (reward optimization task) [13].Το Exploitation, σπρώχνει τον πράκτορα να ανακαλύψει νέες και καλύτερες ενέργειες, αλλά η υπερβολική χρήση του Exploitation μπορεί να οδηγήσει σε σπατάλη χρόνου εάν δεν υπάρχουν καλύτερες ενέργειες [14].

Στις διαδικασίες λήψης αποφάσεων, η εύρεση της σωστής ισορροπίας μεταξύ εκμετάλλευσης και εξερεύνησης είναι ζωτικής σημασίας. Ως εκ τούτου, για να επιτευχθεί η βέλτιστη απόδοση, οι πράκτορες πρέπει να εξισορροπούν την αντιστάθμιση μεταξύ εκμετάλλευσης και εξερεύνησης. Μια κοινή προσέγγιση είναι η χρήση μιας στρατηγικής εξερεύνησης, όπως το epsilon-greedy ή το soft-max, που επιλέγει τυχαία ενέργειες με μια συγκεκριμένη πιθανότητα. Αυτή η προσέγγιση διασφαλίζει ότι ο πράκτορας εξερευνά νέες ενέργειες ενώ εξακολουθεί να εκμεταλλεύεται τις πιο γνωστές ενέργειες [14].

2.2.4 Μαρκοβιανές Διαδικασίες Απόφασης

Οι μαρκοβιανές διαδικασίες απόφασης (Markov decision process – MDP) είναι η μαθηματική αναπαράσταση μιας πολύπλοκης διαδικασίας λήψης αποφάσεων όπου τα αποτελέσματα των ενεργειών ή είναι τυχαία ή είναι ελεγχόμενα από αυτόν που παίρνει τις αποφάσεις. Όλες οι διαδοχικές λήψεις αποφάσεων στην ενισχυτική μάθηση συνήθως μοντελοποιούνται με μαρκοβιανές διαδικασίες απόφασης [11].



Σχήμα 2.3: Συνδιασμός των βασικών στοιχείων που αποτελούν ένα μοντέλο ΕΜ. [15]

.

Ένα MDP αποτελείται από πέντε στοιχεία M = ⟨S, A, P, R, γ⟩

1. S- ένα πεπερασμένο πλήθος πιθανών καταστάσεων.
2. A- ένα που είναι ένα πεπερασμένο πλήθος ενεργειών.
3. P- συνάρτηση πιθανότητας μετάβασηςπεριγράφει την πιθανότητα μετάβασης από τη μια κατάσταση στην άλλη μετά τη λήψη μιας συγκεκριμένης ενέργειας.)
4. R- συνάρτηση ανταμοιβής.
5. γ- παράγοντας έκπτωσης για μελλοντικές ανταμοιβές.

• $P(s'|s, a) = \Pr(s_{t+1} = s'|s_t = s, a_t = a)$

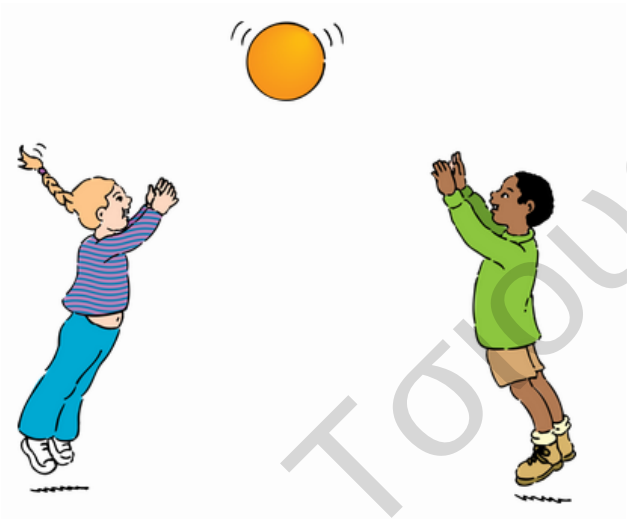
Είναι η πιθανότητα να συμβεί η ενέργεια a στην κατάσταση s την χρονική στιγμή t και να οδηγήσει στην κατάσταση s' την χρονική στιγμή t+1.

• Το $R_a(s, s')$ είναι η άμεση ανταμοιβή (ή η αναμενόμενη άμεση ανταμοιβή) που αποκτάται από την μετάβαση από την κατάσταση ς στην κατάσταση s' λόγω τις ενέργειας a [13] [16].

2.2.5 POMDP (Partially Observable MDP)

Σε ένα περιβάλλον με Μαρκοβιανές Διαδικασίες Απόφασης σε κάποιες περιπτώσεις προϋποθέτουν την πλήρης γνώση του περιβάλλοντος ενώ σε περίπλοκα περιβάλλοντα αυτό δεν είναι εφικτό.

Για παράδειγμα μια μεμονωμένη οθόνη παιχνιδιού όπως διακρίνεται στο πιο κάτω σχήμα δεν αρκεί για να προσδιορίσει την κατάσταση του συστήματος [17].



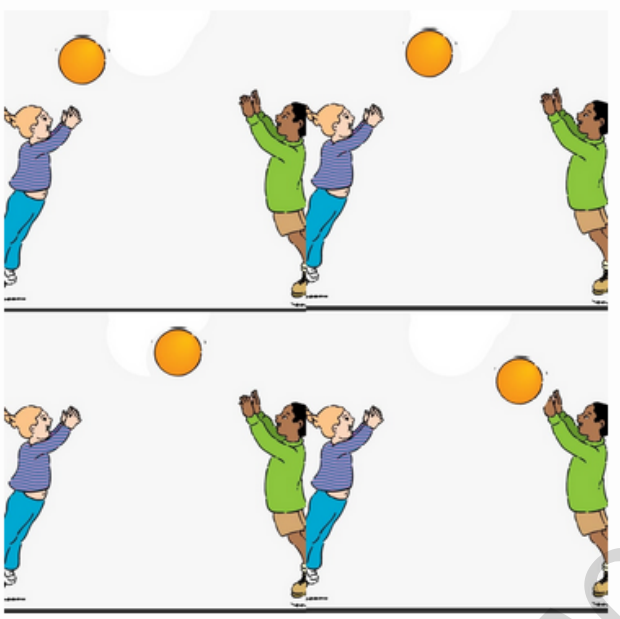
Σχήμα 2.4: Μεμονωμένο στιγμιότυπο από βιντεοπαιχνίδι [17]

Παρατηρώντας το πιο πάνω Σχήμα2.4 δεν μπορούμε να πούμε με σαφήνεια εάν η μπάλα κινείται προς το κορίτσι ή προς το αγόρι. Επομένως, αυτή η έλλειψη πληροφορίας, μας οδηγεί στην μερική γνώση του περιβάλλοντος(POMDP).

Άρα ο πράκτορας δεν είναι σε θέση να αναπτύξει μία πολιτική σε ένα περιβάλλον το οποίο δεν του παρέχεται πλήρης πληροφόρηση για την τρέχουσα κατάσταση. Για να καλύψει αυτή την έλλειψη πληροφορίας και να αναπτύξει ορθότερη πολιτική αναγκάζεται να βασιστεί στο να θυμάται τι έχει δει και μάθει από το παρελθόν.

Παράδειγμα:

Για το συγκεκριμένο παράδειγμα με την μπάλα, μέσω τεσσάρων διαδοχικών εικόνων στο Σχήμα2.5 του παρελθόντος μπορούμε να διακρίνουμε προς ποια κατεύθυνση κινείται η μπάλα. Σύμφωνα με το άρθρο [18] η απομνημόνευση τεσσάρων εικόνων από το παρελθόν κάνει όλα τα απλά παιχνίδια(π.χ. Αταρι 2600 [19]) MDP. Αυτό δεν ισχύει για πιο σύνθετα περιβάλλοντα όπως του πραγματικού κόσμου, αφού ο πράκτορας μπορεί να χρειαστεί να



Σχήμα 2.5: Διαδοχικά στιγμιότυπα από βιντεοπαιχνίδι [17]

θυμηθεί κάτι που συνέβη πριν από πολλά βήματα για να καταλάβει την τρέχουσα κατάσταση. Σε αυτή την περίπτωση τέτοιων POMDP, ο αλγόριθμος DRQN ο οποίος παρουσιάζεται μετέπειτα λειτουργεί καλύτερα από τις άλλες παραλλαγές του DQN [17].

Ένα POMDP διακριτού χρόνου μοντελοποιεί τη σχέση μεταξύ ενός πράκτορα και του περιβάλλοντος του [20]. Τυπικά μπορεί να μοντελοποιηθεί ως ένα σύνολο 7 παραμέτρων (S,A,T,R,Ω,O,γ) όπου:

1. S είναι το σύνολο των καταστάσεων
2. A είναι το σύνολο των ενεργειών
3. T είναι το σύνολο πιθανοτήτων υπο όρους μετάβασης μεταξύ καταστάσεων
4. $R : S \times A \to \mathbb{R}$ είναι η συνάρτηση ανταμοιβής
5. Ω είναι το σύνολο ανταμοιβής
6. O είναι ένα σύνολο πιθανοτήτων υπό όρους παρατήρησης
7. γ [0,1] είναι ο παράγοντα έκπτωσης

Σε κάθε χρονική περίοδο, το περιβάλλον βρίσκεται σε κάποια κατάσταση $s \in S$. Ο πράκτορας αναλαμβάνει δράση $a \in A$, που προκαλεί τη μετάβαση του περιβάλλοντος σε κατάσταση s' πιθανότητα $T(s' \mid s, a)$.

Την ίδια στιγμή, ο πράκτορας λαμβάνει μια παρατήρηση $O \in \Omega$ που εξαρτάται από τη νέα κατάσταση του περιβάλλοντος, ς′ και σχετικά με τις ενέργειες που μόλις ελήφθησαν, a, με πιθανότητα O($o \mid s', a$). Τέλος, ο πράκτορας λαμβάνει μια ανταμοιβή r ίσο με $R(s, a)$. Στη συνέχεια η διαδικασία επαναλαμβάνεται Ο στόχος είναι ο πράκτορα να επιλέγει ενέργειες σε κάθε χρονικό βήμα που μεγιστοποιούν την μελλοντική του ανταμοιβή με έκπτωση:

$E[\sum_{t=0}^{\infty} \gamma^t]$, όπου το r_t είναι η ανταμοιβή που κερδίζεται τη στιγμή t. Ο παράγοντας έκπτωσης (discount factor) γ καθορίζει πόσο ευνοούνται οι άμεσες ανταμοιβές έναντι των μελλοντικών ανταμοιβών.

Όταν $\gamma = 0$
Ο πράκτορας ενδιαφέρεται μόνο για το ποια ενέργεια θα αποφέρει τη μεγαλύτερη άμεση ανταμοιβή.

Όταν $\gamma \to 1$
Ο πράκτορας ενδιαφέρεται για τη μεγιστοποίηση του αναμενόμενου αθροίσματος των μελλοντικών ανταμοιβών.

2.2.6 Εξισώσεις Bellman

Οι εξισώσεις Bellman είναι ένα σύνολο μαθηματικών εξισώσεων οι οποίες εφαρμόζονται στην Ενισχυτική Μάθηση για την επίλυση μιας Διαδικασίας Απόφασης Markov [10] . Οι εξισώσεις αυτές μπορούν να θεωρηθούν ώς ένα σύνολλο εξισώσεων που έχουν στόχο να αποσυνθέσουν τις συνάρτησεις συνάρτησης αξίας κατάστασης και συνάρτησης αξίας ενέργειας ως προς τις αναμενόμενες ανταμοιβές συν τις εκπτωτικές μελλοντικές τιμές των μελλοντικών καταστάσεων [13].

Bellman Expectation Equations

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s'))$$

$$q_\pi(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_\pi(s', a'))$$

Bellman Optimality Equations

$$\boxed{v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_*(s'))}$$

$$\boxed{q_*(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a'))}$$

Useful Formulas for Deriving the Bellman Equations

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) q_\pi(s, a)$$

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_*(s, a)$$

$$q_\pi(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s'))$$

$$q_*(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_*(s'))$$

$$\begin{aligned}
q_\pi(s, a) &\doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] && (1)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} \mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a) \mathbb{E}_\pi[G_t | S_t = s, A_t = a, S_{t+1} = s', R_{t+1} = r] && (2)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a) \mathbb{E}_\pi[G_t | S_t = s, A_t = a, S_{t+1} = s', R_{t+1} = r] && (3)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a) \mathbb{E}_\pi[G_t | S_{t+1} = s', R_{t+1} = r] && (4)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a) \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_{t+1} = s', R_{t+1} = r] && (5)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']) && (6)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s')) && (7)
\end{aligned}$$

2.2.7 Ταξινόμηση Μεθόδων-Methods Taxonomy

Τα είδη των μεθόδων της Ενισχυτικής Μάθησης τα οποία προκύπτουν από τις κύριες διαφορετικές σκοπιές παρατήρησης.

2.2.7.1 Μέθοδοι χωρίς μοντέλα (Model-Free) vs Μέθοδοι που βασίζονται σε μοντέλα (Model-Based)

Η πρώτη κατηγορία μεθόδων βασίζεται στο εάν ο πράκτορας έχει πρόσβαση στο μοντέλο του περιβάλλοντος ή όχι, δηλαδή εάν έχει τις κατάλληλες πληροφορίες έτσι ώστε να μπορέσει να προβλέψει ακριβώς το πώς το περιβάλλον θα ανταποκριθεί στην ενέργεια του πράκτορα μέσα σε αυτό. Με βάση τα πιο πάνω μπορούμε να διακρίνουμε δύο κατηγορίες μεθόδων:

- Ενισχυτική μάθηση χωρίς μοντέλα-Model-Free:

  Οι μέθοδοι χωρίς μοντέλα, μαθαίνουν άμεσα μια συνάρτηση πολιτικής ή μία συνάρτηση αξίας μέσω της αλληλεπίδραση τους με το περιβάλλον χωρίς όμως να δημιουργούν ένα ακριβές μοντέλο του περιβάλλοντος(προσεγγιστικό) [21]. Ο τρόπος με τον οποίο μαθαίνει ο πράκτορας στην ενισχυτική μάθηση χωρίς μοντέλο είναι κυρίως μέσω του trial-and-error, δηλαδή ο πράκτορας αλληλεπιδρά με το περιβάλλον, εκτελεί κάποιες ενέργειες και λαμβάνει ανατροφοδότηση από το περιβάλλον με τη μορφή ανταμοιβών ή ποινών [22]. Με βάση αυτές τις ανταμοιβές, ο πράκτορας ενημερώνει την πολιτική ή τη συνάρτηση αξίας του για να βελτιώσει τη λήψη αποφάσεων. Οι μέθοδοι χωρίς μοντέλα μπορούν να ονομαστούν και ως indirect, planing methods [10]. Μεθόδοι χωρίς μοντέλα είναι Monte Carlo, Temporal difference methods-Μαθηση Χρονικών Διαφορών).

- Ενισχυτική μάθηση με βάση το μοντέλο-Model-Based:

  Οι μέθοδοι που βασίζονται σε μοντέλα, δημιουργούν ένα ρητό μοντέλου του περιβάλλοντος και τη χρήση του για το σχεδιασμό και τη λήψη αποφάσεων. Κάνοντας χρήση του γνωστού αυτού μοντέλου ο πράκτορας κάνει ένα σχεδιασμό (planning) για το ποιές ενέργειες θα λάβει σε κάθε κατάσταση με απώτερο στόχο την μεγιστοποίηση της σωρευτικής ανταμοιβής [23]. Αυτό το είδος αλγορίθμων μπορούν να επιτύχουν εκμάθηση βέλτιστων πολιτικών μέσω ελάχιστης αλληλεπίδρασης με το περιβάλλον. Οι μέθοδοι με βάση το μοντέλο μπορούν να ονομαστούν και ως direct learning methods [10]. Μεθόδοι με βάση το μοντέλο είναι ο Δυναμικός Προγραμματισμός (Dynamic program-

ming) και η Ευρετική αναζήτηση (heuristic search). Η ταξινόμηση γίνεται ως προς τέσσερις κύριες κατηγορίες (Value Function Methods,Policy Search, Transition Models, Return Function) ανάλογα με το πρόβλημα που επιλύει η κάθε μία.

2.2.7.2 Βάση αξίας (Value-Based) vs Βάση πολιτικής (Policy-Based)

Μία άλλη οπτική με την οποία μπορούμε να ταξινομήσουμε τους αλγορίθμους της ΕΜ είναι λαμβάνοντας υπόψη ποιο στοιχείο βελτιστοποιείται από τον αλγόριθμο, συνάρτηση αξίας τιμής ή η πολιτική [22].

Οι αλγόριθμοι που είναι βάση αξίας στοχεύουν να μάθουν τις συναρτήσεις αξίας και κατάστασεις αξίας προκειμένου να δημιουργήσουν τη βέλτιστη πολιτική (δηλαδή η βέλτιστη πολιτική δημιουργείται implicitly).

Οι αλγόριθμοι που είναι βάση πολιτικής στοχεύουν να μάθουν την πολιτική απευθείας χρησιμοποιώντας μια παραμετροποιημένη συνάρτηση.

2.2.7.3 On-policy vs off-policy

Η ταξινόμηση αλγόριθμων μπορεί να βασιστεί στον διαφορετικό τρόπο προσέγγισης για την ενημέρωση και τη χρήση της πολιτικής από ένα πράκτορα, δηλαδή στο πώς ο πράκτορας λαμβάνει τις αποφάσεις για τις ενέργειες που εκτελεί σε ένα περιβάλλον. Οι δύο διαφορετικές προσεγγίσεις βασίζονται στο εάν ο πράκτορας είναι εντός πολιτικής (On-policy) ή εκτός πολιτικής (Off-policy) [22].

Εδω υπάρχουν δύο ήδη πολιτικών:

1. Πολιτική στόχου-(Target Policy) : η πολιτική που επιδιώκει να αποκτήσει ένας πράκτορας μέσω της συνάρτησης αξίας η οποία εγκαθιδρύεται μέσω της, αλληλεπιδρώντας με το περιβάλλον. Συνεπώς, όσο περισσότερο αλλιλεπιδρά με το περιβάλλον τόσο βελτιώνει την πολιτική αυτή [24].

2. Πολιτική συμπεριφοράς (Behavior Policy): η πολιτική που χρησιμοποιείται από έναν πράκτορα για την επιλογή των δράσεών του. Ο πράκτορας ακολουθεί αυτήν την πολιτική προκειμένου να αλληλεπιδράσει με το περιβάλλον. [24].

Εντός πολιτικής-(On-Policy): Στη μάθηση ενίσχυσης εντός πολιτικής, ο πράκτορας ενημερώνει την πολιτική στόχου του με βάση τις εμπειρίες που συγκεντρώνει ακολουθώντας την

τρέχουσα πολιτική του (πολιτική στόχου). Ο πράκτορας αλληλεπιδρά με το περιβάλλον, συλλέγει δεδομένα (π.χ. καταστάσεις, ενέργειες, ανταμοιβές) και χρησιμοποιεί αυτά τα δεδομένα για να ενημερώσει την πολιτική του. Στη συνέχεια, η ενημερωμένη πολιτική χρησιμοποιείται για την επόμενη επιλογή ενέργειας. Αυτό σημαίνει ότι η πολιτική του πράκτορα ενημερώνεται συνεχώς και βελτιστοποιείται με βάση την τρέχουσα συμπεριφορά του [24]. Εν ολίγοις, [Target Policy = Behavior Policy]. Μερικά παραδείγματα αλγορίθμων On-Policy είναι το Policy Iteration, Value Iteration, Monte Carlo for On-Policy [22].

Εκτός πολιτικής-(Off-Policy): Στην εκμάθηση ενίσχυσης εκτός πολιτικής, ο πράκτορας ενημερώνει την πολιτική στόχου του χρησιμοποιώντας εμπειρίες που συλλέχθηκαν ακολουθώντας μια διαφορετική πολιτική (Πολιτική συμπεριφοράς) από αυτήν που ενημερώνεται επί του παρόντος. Αυτό σημαίνει ότι ο πράκτορας μπορεί να μάθει από προηγούμενες εμπειρίες που είναι αποθηκευμένες, ακόμα κι αν ακολουθεί διαφορετική πολιτική τη στιγμή της ενημέρωσης. Αυτό επιτρέπει μεγαλύτερη ευελιξία στη συλλογή και εκμάθηση δεδομένων, καθώς ο πράκτορας μπορεί να επαναχρησιμοποιήσει προηγούμενες εμπειρίες [24]. Εν ολίγοις, [Target Policy = Behavior Policy]. Μερικά παραδείγματα αλγορίθμων μάθησης εκτός πολιτικής είναι η εκμάθηση Q και η sarsa [22].

2.2.8 Μεθόδοι Ενισχυτικής Μάθησης

2.2.8.1 Δυναμικός Προγραμματισμός - Dynamic Programming

Ο δυναμικός προγραμματισμός (γνωστός και ως δυναμική βελτιστοποίηση -dynamic optimization) είναι μια μέθοδος για την επίλυση ενός σύνθετου προβλήματος με τη διάσπασή του σε μια συλλογή απλούστερων υποπροβλημάτων, ακολουθεί η επίλυση καθενός από αυτά τα υποπροβλήματα και τέλος η αποθήκευση αυτών των λύσεών τους [25].

Οι αλγόριθμοι που χρησιμοποιούνται στο δυναμικό προγραμματισμό έχουν ως πυρήνα τους τις εξισώσεις Bellman, υπολογίζονται οι βέλτιστες αξίας- ενέργειας και η βέλτιστη συνάρτηση αξίας. Βασικοί όροι για το δυναμικό προγραμματισμό είναι [10]:

1. Αξιολόγηση πολιτικής (policy evaluation)
2. Βελτίωση πολιτικής (policy improvement)
3. Επανάληψη ως προς την πολιτική (policy iteration)
4. Επανάληψη ως προς την αξία(value iteration)

1) Αξιολόγηση πολιτικής (policy evaluation)

Η διαδικασία για την αξιολόγηση μίας δεδομένης πολιτικής επιτυγχάνεται εκτελώντας επαναληπτικό υπολογισμό των συναρτήσεων αξίας για καθε κατάσταση μέσω της εξισωσης Bellman αυτής της πολιτικής.

2) Βελτίωση πολιτικής (policy improvement)

Η διαδικασία βελτίωσης πολιτικής έχει ως σκοπό τον υπολογισμό του value function για μία πολιτική με απώτερο στόχο την εύρεση μίας καλύτερης πολιτικής από την ήδη υφιστάμενη. Για να διαπιστωθεί εάν υπάρχει καλύτερη πολιτική π' από την ήδη υφιστάμενη π, αρκεί να εκτελέσουμε μία ντετερμινιστική ενέργεια a ($a \neq \pi(s)$) από μια κατάσταση s και κατόπιν να γίνει εφαρμογή της τρέχουσας πολιτικής π. Υπολογίζοντας την συνάρτηση αξίας για αυτή την ενέργεια σε αυτή την κατάσταση και κάνοντας την πίο κάτω σύγκριση μπορούμε να διαπιστώσουμε εάν υπάρχει η βέλτιστη πολιτική.

$Q_\pi(s, a) \geq V_\pi(s)$

Η πιο πάνω ανισότητα μπορεί να γενικευτεί για συγκρίση δύο οποιοδήποτε ντετερμινιστικές πολιτικές π και π', ως ειδική περίπτωση του λεγόμενου θεωρήματος βελτίωσης πολιτικής (policy improvement theorem).

$Q_\pi(s, \pi(s)) \geq V_\pi(s)$, $s \in S$, τότε $V'_\pi \geq V_\pi$

3)Επανάληψη ως προς την πολιτική (policy iteration)

Ο συνδυασμός αξιολόγησης πολιτικής (policy evaluation) και της βελτίωσης πολιτικής (policy improvement) μας παρέχουν δύο νέες μεθόδους δυναμικού προγραμματισμού.

1. επανάληψη ως προς την πολιτική (policy iteration)
2. επανάληψη ως προς την αξία(value iteration)

Αυτές οι δύο νέες μεθόδοι μας παρέχουν μέσω υπολογισμών τις βέλτιστες συναρτήσεις αξίας αλλά και τις βέλτιστες πολιτικές.

Η επανάληψη ως προς την πολιτική, επιτυγχάνεται μέσω των δύο πιο πάνω διαδικασιών που αναφέραμε πιο πάνω

1. αξιολόγηση της παρούσας πολιτικής -policy evaluation
2. βελτίωση της παρούσας πολιτικής -policy improvement

Αυτή η διαδικασία επαναλαμβάνεται μέχρι να επέλθει σύγκλιση σε μία βέλτιστη πολίτικη.

Εδώ πρέπει να σημειωθεί ότι ο αλγόριθμος επανάληψης ως προς την πολιτική μαστίζεται απο το μειονέκτημα ότι σε κάθε βήμα αξιολόγησης της παρούσας πολιτικής το οποίο εκτελείται επαναληπτικά χρειάζεται μεγάλος χρόνος για να συγκλίνει ακριβώς σε ένα V_p, και συνεπώς απαιτεί μεγάλο υπολογιστικό κόστος. Επανάληψη ως προς την αξία(value iteration),

4)Επανάληψη ως προς την αξία(value iteration)

Εδώ έρχεται ο αλγόριθμος επανάληψης ως προς την αξία(Value iteration) έτσι ώστε να βελτιώσει την παραπάνω διαδικασία, σταματόντας το policy evaluation αμέσως μετά από την πρώτη σάρωση(one backup of each state). Αυτή η διακοπή αμέσως μετά από την πρώτη επανάληψη δεν επηρεάζει την σύγκλιση του policy improvement αφού ο υπολογισμός της νέας τιμής V_{k+1} value function έιναι εκτιμώμενη.

2.2.8.2 Monte Carlo

Το Monte Carlo στην ενισχυτική μάθηση αναφέρεται σε μια μέθοδο που χρησιμοποιείται για την εκτίμηση των τιμών των καταστάσεων ή των ζευγών καταστάσεων-δράσεων σε μια διαδικασία απόφασης Markov (MDP) με σκοπό την μάθηση των συναρτήσεων αξίας και βέλτιστων πολιτικών. Σε αντίθεση με τον δυναμικό προγραμματισμό που αναφέρθηκε προηγουμένως στη μέθοδο Monte Carlo το περιβάλλον είναι πλήρως άγνωστο (ο πράκτορας δεν γνωρίζει τη μετάβαση και τις ανταμοιβές του MDP) με αποτέλεσμα η εκμάθηση του περιβάλλοντος να επιτυγχάνεται μέσω δειγμάτων επεισοδίων (sample episodes) [10].

Δηλαδή το μόνο που απαιτείται ουσιαστικά είναι δείγματα ακολουθιών(καταστάσεις, ενέργειες και ανταμοιβές) που λαμβάνονται μέσω της αλληλεπίδρασης του πράκτορα με ένα περιβάλλον εκτελώντας ενέργειες από μια δεδομένη κατάσταση, λαμβάνοντας ανταμοιβές και μεταβαίνοντας σε νέες καταστάσεις μέχρι την τερματική κατάσταση [26].

Πίο συγκεκριμένα η μέθοδος MC αρχίζει από μία τυχαία κατάσταση και συνεχίζει για ένα πεπερασμένη χρονική διάρκεια(πεπερασμένα βήματα) μέχρι να καταλήξει στην τερματική κατάσταση, καταγράφει τις καταστάσεις από τις οποίες πέρασε καθώς και την ενέργεια και την ανταμοιβή που πήρε στην κάθε μία από αυτές [27]. Κατά την λήξη κάθε επεισοδίου κάνοντας χρήση των αυτών των δεδομένων γίνεται εκτίμηση των μέσων τιμών για την συναρτήσεων αξίας V(s) και συνάρτηση αξίας κατάστασης Q(s) και ακολούθως γίνονται οι αλλαγές των πολιτικών.

Τα κύρια μειονεκτήματα αυτής της μεθόδου είναι :

1. ότι δεν εγγυάται ότι θα περάσει από όλα τα πιθανά σενάρια, αυτό μπορεί να 'διορθωθεί' θέτοντας ένα καλό ισοζύγιο μεταξύ exploring(epsilon) και exploit(1-epsilon)

2. ότι να για ανανεωθούν οι τιμές των συναρτήσεων αξίας και συναρτήσεων κατάστασης αξίας πρέπει πρώτα να τελειώσει ένα επεισόδιο, που αυτό σε παιχνίδια χωρίς τέλος είναι πρόβλημα.

2.2.8.3 Μάθηση Χρονικών Διαφορών-Temporal-Difference Learning

Όπως και στις προηγούμενες μεθόδους έτσι και στη ΜΧΔ ένας πράκτορας αλληλεπιδρά με ένα περιβάλλον εκτελώντας κάποια ενέργεια σε κάθε κατάσταση και λαμβάνει ως ανατροφοδότηση μία ανταμοιβή. Ο στόχος του πράκτορα είναι να μάθει μια πολιτική η οποία θα μεγιστοποιεί τις σωρευτικές ανταμοιβές με την πάροδο του χρόνου. Οι αλγόριθμοι Χ υπολογίζουν τη συνάρτηση τιμής, η οποία είναι ένα μέτρο των αναμενόμενων σωρευτικών ανταμοιβών από μια συγκεκριμένη κατάσταση, και τη χρησιμοποιούν για να ενημερώσουν την πολιτική [10]. Η πολιτική αυτή αναφέρεται στις αποφάσεις ή τις ενέργειες που ορίζονται βάσει των υπολογισμένων τιμών της συνάρτησης τιμής.

Η μάθηση χρονικών διαφορών (temporal difference learning) μπορεί να θεωρηθεί ως ένας συνδυασμός μεταξύ των μεθόδων Δυναμικού προγραμματισμού και MC. Όπως και στη μέθοδο MC το μοντέλο του περιβάλλοντος είναι άγνωστο και η εκμάθηση του επιτυγχάνεται μέσω ημιτελών επεισοδιακών δειγμάτων [27]. Η κύρια διαφορά μεταξύ ΜΧΔ και της MC είναι στο πότε γίνεται η εκτίμηση και ενημέρωση της συνάρτησης αξίας τους. Συγκεκριμένα στο MC ο πράκτορας πρέπει να μεταβεί στην τελική κατάσταση για να κάνει εκτίμηση και ενημέρωση τις συναρτήσεις αξίας σε αντίθεση με τη ΜΧΔ στην οποία η εκτίμηση και ενημέρωση της συναρτήσης αξίας γίνεται μετά από κάθε χρονική στιγμή.

Άρα στη ΜΧΔ η επανεκτίμηση τις συνάρτησης αξίας γίνεται βάσει της ανταμοιβής που παρέχεται ανά χρονική μέσω της αλληλεπίδρασης του πράκτορα με το περιβάλλον. Η επανεκτίμηση της συνάρτησης τιμής βασίζεται στο σφάλμα χρονικής διαφοράς δ(temporal difference error – TD error) που είναι η διαφορά μεταξύ της εκτιμώμενης τιμής μιας κατάστασης ή ενέργειας και της πραγματικής παρατηρούμενης τιμής, η οποία είναι συνήθως ένας συνδυασμός της ανταμοιβής που λαμβάνεται και της εκτιμώμενης τιμής της επόμενης κατάστασης [26]. Sarsa: On-Policy TD control

Ο SARSA είναι αλγόριθμος εντός πολιτικής (on-policy) και συνεπώς η πολιτική συμπεριφοράς που κάνει χρήση ο πράκτορας για την επιλογή των ενεργειών του είναι η ίδια με την πολιτική στόχου που αξιολογεί και βελτιστοποιεί σε κάθε βήμα. Για την αξιολόγηση και βελτίωση της πολιτικής στόχου ο πράκτορας εφαρμόζει την εξίσωση

$$[\underbrace{\text{New}Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underbrace{\alpha}_{\text{New Q-Value}}[\underbrace{R(s,a)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount rate}} \overbrace{Q'(s',a')}^{\substack{\text{Predicted reward, given}\\ \text{new state and all possible actions}}} - Q(s,a)]$$

η οποία απαιτεί την τρέχουσα κατάσταση, την ενέργεια που πραγματοποιήθηκε, την ανταμοιβή που έλαβε, την επόμενη κατάσταση και την ενέργεια που επιλέχθηκε στην επόμενη κατάσταση. Οι ενέργειες επιλέγονται από την πολιτική η οποία ταυτόχρονα αξιολογείται μέσω των υπολογισμών της Q value.

Στην ουσία ο SARSA υπολογίζει μία τιμή Q value ενός ζεύγους κατάστασης-ενέργειας $Q(s_t, a_t)$, χρησιμοποιώντας εκτιμήσεις άλλων ζευγών $Q(s_{t+1}, a_{t+1})$ προσθέτοντας σε αυτό την άμεση ανταμοιβή r+$Q(s_{t+1}, a_{t+1})$ [10].

Q-Learning: Off-Policy TD control

Ο Q-Learning είναι αλγόριθμος εκτός πολιτικής(off-Policy) και συνεπώς η πολιτική συμπεριφοράς που κάνει χρήση ο πράκτορας για την επιλογή των ενεργειών κατά τη διαδικασία εκμάθησης δεν είναι ίδια με την πολιτική στόχου που αξιολογεί και βελτιστοποιεί σε κάθε βήμα. Για την αξιολόγηση και βελτίωση της πολιτικής στόχου ο πράκτορας εφαρμόζει την εξίσωση

$$[\underbrace{\text{New}Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underbrace{\alpha}_{\text{New Q-Value}}[\underbrace{R(s,a)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount rate}} \overbrace{\max Q'(s',a')}^{\substack{\text{Maximum predicted reward, given}\\ \text{new state and all possible actions}}} - Q(s,a)]$$

Ο Q-Learning σε αντίθεση με τον SARSA επιχειρεί να προσεγγίσει απευθείας τη βέλτιστη συνάρτηση Q*, ανεξαρτήτως από την πολιτική που ακολουθείται από τον πράκτορα, αφού σε κάθε κατάσταση επιλέγει άπληστα την μεγαλύτερη τιμή της συνάρτησης Q((γ$maxQ(s_{t+1,a})$))που είναι διαθέσιμη για την κατάσταση s_{t+1} χωρίς να δει μακροπρόθεσμα.

Στην ουσία ο Q-Learning υπολογίζει μία τιμή Q value ενός ζεύγους κατάστασης-ενέργειας $Q(s_t, a_t)$, χρησιμοποιώντας τη μέγιστη τιμή Q της επόμενης κατάστασης άλλων ζευγών max Q(s_{t+1}, a_{t+1}) προσθέτοντας σε αυτό την άμεση ανταμοιβή r +$Q(s_{t+1}, a_{t+1})$ [10].

2.2.9 Function Approximation

Όλοι οι μέθοδοι που έχουν αναφερθεί μέχρι στιγμής απαιτούν την αποθήκευση όλων των δεδομένων (s,a) ∈ S A σε ένα πίνακα στη μνήμη. Αυτό δεν είναι εφικτό σε πραγματικά προβλήματα στα οποία ο χώρος καταστάσεων και των ενεργειών είναι πολύ μεγάλος. Επομένως η μέθοδος πινάκων (tabular methods) γίνεται ανεπαρκής και ακατάλληλος. Για την αντιμετώπιση αυτού του προβλήματος εφαρμόζεται μία νέα μέθοδος η οποία βασίζεται στην εκτίμησης της τιμής κάθε καταστάσεις βάση αρακτηριστικών(παραμετρικές συναρτήσεις) [28] [10].

Μέσω των παραμετρικών συναρτήσεων θα υπολογίζεται η αξία κάθε κατάστασης η οποία θα είναι προσεγγιστική και όχι η ακριβής τιμή της. Για την υλοποίηση αυτής της προσεγγιστικής διαδικασίας υπάρχουν οι θεμελιώδης συναρτήσεις προσέγγισης.

1. Linear combinations of features
2. Neural networks
3. Decision Tree
4. nonlinear gradient-descent
5. Radial Basis Functions (RBFs)

2.2.10 Προγραμματισμός-Planning

Οι επιστήμονες στον τομέα της ΕΜ τις τελευταίες δεκαετίες έχουν ανάπτυξη διάφορες μεθόδους για την εκπαίδευση ευφυών πρακτόρων. Μία από αυτές τις μεθόδους είναι ο προγραμαμτικσμός Planning ο οποίος βασίζεται σε μοντέλαModel-Based. Ο προγραμματισμόςPlanning θεωρείτε μια φυσική και ισχυρή προσέγγιση για την λήψη αποφάσεων σε προβλήματα με γνωστές δυναμικές, όπως για παράδειγμα στα βιντεοπαιχνίδια (πχ go ,atari) και προσομοιωμένος έλεγχος ρομπότ ([29] [3] [30]).

Βάση του [10] με το όρο προγραμματισμός (Planning) στην ΕΜ αναφερόμαστε σε οποιαδήποτε υπολογιστική διαδικασία που λαμβάνει ένα μοντέλο ως είσοδο και παράγει ή βελτιώνει μια πολιτική η οποία θα αλληλεπιδρά με το συγκεκριμένο μοντέλο περιβάλλοντος. Με πιο απλά λόγια ο προγραμματισμός στην ΕΜ είναι η ικανότητα του πράκτορα να χαρτογραφεί στρατηγικά τις ενέργειές, κατασκευάζοντας μια ακολουθία ενεργειών η οποία συχνά ανα-

φέρεται ως πολιτική, προκειμένου να επιτύχει τους στόχους του ενώ ταυτόχρονα προσπαθεί να μεγιστοποιήσει τις μακροπρόθεσμες ανταμοιβές.

Σε αντίθεση με άλλες προσεγγίσεις στην ΕΜ, όπως οι μέθοδοι χωρίς μοντέλα (Model-free) που βασίζονται στη δοκιμή και το σφάλμα, ο σχεδιασμός προσφέρει πολλά πλεονεκτήματα όπως αναφέρεται στο [31] :

1. **Υψηλότερη Απόδοση Δεδομένων- Higher Final Performance**

   Αυτό προκύπτει λόγο της αξιοποίησης του σήματος ανατροφοδότησης(σήμα εκπαίδευσης-Btraining signal) και δεν απαιτεί ανταμοιβές διάδοσης (propagating rewards) μέσω των Bellmans backups*.

2. **Υψηλότερη Τελική Απόδοση-Higher data-efficiency**

   Το Planning προσφέρει αύξηση της απόδοσης καθώς αυξάνεται το computational budget για την αναζήτηση μελλοντικών ενεργειών υψηλότερης απόδοσης στην προσπάθεια μεγιστοποίησης και επιλογής της επόμενης καλύτερης κινησης χωρίς να είναι απαραίτητα τα ανθρώπινα δεδομένα(human demonstration) [3].

3. **Μεταφορα Γνωσης-(Knowledge Transfer)**.

   Η εκπαίδευση του αλγορίθμου για την εκμάθηση της δυναμικής ενός συγκεκριμένου περιβάλλοντος πολλές φορές προϋποθέτει την κατανόηση υποπροβλημάτων τα οποία είναι ανεξάρτητα μεταξύ τους. Αυτή η γνώση μπορεί να μεταφερθεί καλά σε άλλες εργασίες στο περιβάλλοντα που περιέχουν τα ίδια ή παρόμοια υποπροβλήματα τα οποία έχουν ήδη κατανοήθεί και επιλύθει.

Αρχικά στο Planning ο πράκτορας αλληλεπιδρά με το περιβάλλον και συλλέγει χρήσιμες πληροφορίες. Κάνοντας χρήση αυτών των πληροφοριών επιτυγχάνεται η κατασκευή ενός μοντέλου του περιβάλλοντος, το οποίο αντικατοπτρίζει την δυναμική -συμπεριφορά του επιβάλλοντος όπως ο ίδιος ο πράκτορα την έχει αντιληφθεί. Αυτό το μοντέλο μπορεί να λάβει διαφορετικές μορφές, που κυμαίνονται από ένα απλό ντετερμινιστικό έως ένα σύνθετο στοχαστικό (πιθανοτικό μοντέλο) μοντέλο. Ο πράκτορας μπορεί στη συνέχεια να χρησιμοποιήσει αυτό το μοντέλο για να δημιουργήσει πιθανές τροχιές ενεργειών και να εκτιμήσει τις αναμενόμενες συνέπειες και ανταμοιβής της κάθε τροχιάς. Εξερευνώντας πιθανές μελλοντικές καταστάσεις και τις αντίστοιχες ανταμοιβές τους, ο πράκτορας μπορεί να αξιολογήσει και να συγκρίνει διαφορετικές ακολουθίες ενεργειών για να καθορίσει την πιο 'βέλτιστει' πορεία

δράσης.

Εν τέλη το Planning επιτρέπει στους πράκτορες να λαμβάνουν έξυπνες αποφάσεις διερευνώντας στρατηγικά τις συνέπειες των πράξεών τους και αποτελεί βασικό συστατικό για τη δημιουργία αποτελεσματικών και αυτόνομων συστημάτων ΕΜ.

Συνοψίζοντας το Planning στην ΕΜ θεωρείτε ένα κρίσιμο στοιχείο σε πολλές εφαρμογές του πραγματικού κόσμου, συμπεριλαμβανομένης της ρομποτικής, της αυτόνομης οδήγησης και του παιχνιδιού. Επιτρέπει στους πράκτορες να αιτιολογήσουν τις ενέργειές τους, να προβλέψουν τις συνέπειες των αποφάσεών τους και τελικά να μάθουν αποτελεσματικές στρατηγικές για την επίτευξη των στόχων τους.

2.2.11 Ιεραρχική Ενισχυτική Μάθηση(Hierarchical Reinforcement Learning)

Αν και η ΕΜ έχει επιτύχει πολλά την τελευταία δεκαετία έχει διαπιστωθεί ότι πάσχει από μια ποικιλία ελαττωμάτων κυρίως όταν τα περιβάλλοντα με τα οποία αλληλεπιδρά ο πράκτορας είναι πολύ περίπλοκα. Συγκεκριμένα στη προσπάθεια του ο πράκτορας όταν εξερευνά το περιβάλλον για εύρεση πιθανών λύσεων συχνά αποτυγχάνει ή είναι πολύ αναποτελεσματικός, απαιτώντας υπερβολική ποσότητα αλληλεπίδρασης με το περιβάλλον [32]. Αυτή η χρονοβόρα αλληλεπίδραση μεταξύ πράκτορα και περιβάλλοντος έχει ως αποτέλεσμα την παρεμπόδιση της μάθησης και την εφαρμογή της σε πιο περίπλοκα περιβάλλοντα.

Η ιεραρχική ενισχυτική μάθηση (ΙΕΜ) είναι μια υπολογιστική προσέγγιση που προορίζεται να αντιμετωπίσει αυτά τα ζητήματα μαθαίνοντας να λειτουργεί σε διαφορετικά επίπεδα χρονικής αφαίρεσης, που επιτρέπουν την αποσύνθεση δύσκολων εργασιών λήψης αποφάσεων μεγάλου βάθους σε απλούστερες δευτερεύουσες εργασίες [33],καθώς οι επιμέρους εργασίες που προκύπτουν μπορούν να επιλυθούν αποτελεσματικά με προσεγγίσεις ΕΜ (καλύτερη αναπαράσταση γνώσης), ενώ ταυτόχρονα ανοίγει το δρόμο για επαναχρησιμοποίηση συμπεριφοράς και αυξημένη ερμηνεία των συστημάτων ΕΜ [32].

Σε αυτό το κρίσιμο σημείο έρχεται η ΙΕΜ η οποίο στοχεύει να μειώσει ακόμη και να εξαλείψει αυτά τα προβλήματα κάνοντας την διαδικασία μάθησης απλούστερη, αναλύοντας τον τρόπο μάθησης σε συγκεκριμένα μέρη. Οι κύριες αδυναμίες της ΕΜ τις οποίες η ΙΕΜ μπορεί να αντιμετώπιση είναι [34]:

1. Κατάρα της διαστασιμότητας (curse of dimensionality)
2. Κλιμάκωση-Scaling up
3. Γενίκευση-Generalization
4. Αφαίρετικότητα-Abstraction

Έχοντας βάση τα πιο πάνω η ιεραρχική ενισχυτική μάθηση (IEM) μπορεί να θεωρηθεί ως μια υπολογιστική προσέγγιση που προορίζεται να αντιμετωπίσει αυτά τα ζητήματα μαθαίνοντας να λειτουργεί σε διαφορετικά επίπεδα χρονικής αφαίρεσης(temporal abstraction, τα οποία μας επιτρέπουν την απλούστευση του προβλήματος, καθώς οι επιμέρους εργασίες αποσυνθέτονται σε μικρότερες υποπροεργασίες οι οποίες είναι πιο διαχειρίσιμες μαθαίνοντας στην κάθε μία την δική της πολιτική μάθησης και ταυτόχρονα η κάθε μία από αυτές μπορεί να χρησιμοποιηθεί σε πολλαπλά επίπεδα αφαίρεσης.

Το IEM μπορεί να θεωρηθεί ότι προσπαθεί να μιμηθεί τον τρόπο με τον οποίο οι άνθρωποι συμπεριφορά εμφανίζει ιεραρχική δομή , δηλαδή θέτουν αυθαίρετα υποστόχος εντός του πεδίου γνώσεων τους για να επιτύχουν ένα μεγαλύτερο στόχο. [35]. Για την επίτευξη αυτών των νέων στόχων εκτελούνται απλές ενέργειες που συνεργάζονται για την επίτευξη των γενικών στόχων εργασίας. Έτσι οι ευφυείς πράκτορες συχνά επιλύουν εργασίες ιεραρχικά κάνοντας χρήση της αφαιρετικότητας ως προς το χρόνο (temporal abstraction), όπου δεν είναι απαραίτητη η λήψη αποφάσεων σε κάθε χρονική στιγμή, αλλά επιτρέπεται η επίκληση της εκτέλεσής εκτεταμένων χρονικά δραστηριοτήτων, που ακολουθούν τις δικές τους πολιτικές μέχρι να τερματιστούν [36]. Με αυτό τον τρόπο υλοποιούνται οι ιεραρχικές αρχιτεκτονικές και αλγόριθμοι μάθησης στην IEM.

Στην IEM, ένας πράκτορας ΕΜ για να εφαρμόσει αυτή την χρονική αφαιρετικότητα μαθαίνει να αποσυνθέτει σύνθετες εργασίες σε μικρότερες-πίο διαχειρίσιμες και μαθαίνει χρονικές εκτεταμένες δράσεις για την ολοκλήρωση κάθε δευτερεύουσας εργασίας. Επομένως κάθε πράκτορας είναι οργανωμένος σε μια ιεραρχία πολλαπλών επιπέδων. Στο υψηλότερο επίπεδο, ο πράκτορας, πέραν των βασικών ενεργειών (elementary actions), έχει τη δυνατότητα να επιλέγει και μακροτελεστές (macro-operators) ή μακροενέργειες (macro-actions), αλλιώς γνωστές ως απλά macros [37]. Οι μακροτελεστές ουσιαστικά είναι μια πολιτική-μία ακολουθία ενεργειών – τελεστών που μπορούν να κληθούν αναπάσα στιγμή για επίλυση ενός προβλήματος για το οποίο υπάρχει ήδη η βέλτιστη πολιτική. Οι μακροτελεστές επιτρέπουν

στο σύστημα να αγνοήσει λεπτομέρειες που είναι άσχετες με το προς επίλυση πρόβλημα και να επικεντρωθεί στην ουσία ενώ ταυτόχρονα του επιτρέπουν να λαμβάνει αποφάσεις σε μεγαλύτερους χρονικούς ορίζοντες.

Κατά συνέπεια τα macros συνθέτουν τη βάση για την IEM καθώς μέσων αυτών διασπάται το κύριο πρόβλημα σε μικροπροβλήματα τα οποία αντιμετωπίζονται αναξάρτητα μεταξύ τους.

Στο χαμηλότερο επίπεδο, ο πράκτορας επιλέγει πρωτόγονες ενέργειες primitive actions, οι οποίες είναι ενέργειες μικρής διάρκειας που χρησιμοποιούνται για την εκτέλεση απλών ενεργειών που έχουν άμεση αλληλεπίδραση με το περιβάλλον.

Για την προσέγγιση της της χρονική αφαιρετικότητας και εφαρμογής της στην ΕΜ είναι η χρήση των ημι-Μαρκοβιανών διαδικασίων απόφασης(ΗΜΔΑ (semi-Markov Decision Processes – SMDPs)) [37] [10]. Τα ΗΜΔΑ μπορούν να θεωρηθούν ως μια γενικευμένη εκδοχή των ΜΔΑ με το κύριω χαρακτηριστικό του είναι το το ποσό χρόνου που παρεμβάλλεται μεταξύ μιας απόφασης και της επόμενης μπορεί να ορίζεται ως:

- Τυχαία μεταβλητή
- Ακέραια σταθερά
- Πραγματική σταθερά

σε αντίθεση με τα ΜΔΑ όπου το ποσό του χρόνου μεταξύ δύο χρονικών βημάτων κατά τα οποία λαμβάνεται απόφαση δεν παίζει ρόλο αφού θεωρείτε σειριακή και σταθερή [37].

Για την προσέγγιση και υλοποίηση της IEM εγκαθιδρυθήκαν κάποιες θεμελιώδεις μέθοδοι [34] [32]:

1. Φεουδαρχική Μάθηση-Feudal Learning
2. Πλαίσιο επιλογών-Options Framework
3. Ιεραρχίες αφηρημένων μηχανών-Hierarchical Abstract Machines
4. Αποδόμηση Συναρτήσων Αξίας-MAXQ (MAXQ Value Function Decomposition))

2.3 Βαθιά Μάθηση (Deep Learning)

2.3.1 Εισαγωγή

Στόχος του συγκεκριμένου κεφαλαίου, είναι να περιγραφούν και να γίνουν κατανοητές οι βασικές πτυχές της βαθιάς μάθησης. Αρχικά θα γίνει μία ιστορική αναφορά της ανάγκης του ανθρώπου για την ανάπτυξη επιτευγμάτων στον τομέα της ΤΝ και πώς αυτό οδήγησε στην εξέλιξη της βαθιάς μάθησης. Ακολούθως αφού αναλυθεί η έννοια της βαθιάς μάθησης και η συνεισφορά της, θα γίνει αναφορά στις βασικές έννοιες της, στους θεμελιώδεις τύπους τεχνιτών νευρωνικών δικτύων, στους αλγόριθμους βελτιστοποίησης των ΤΝΔ, στα πλαίσια βαθιάς μάθησης και τέλος μία μικρή αναφορά στα παραγωγικά μοντέλα.

2.3.2 Εισαγωγή στη Βαθιά Μάθηση (Introduction to Deep Learning)

2.3.2.1 Ιστορική Επισκόπηση (Historical Overview)

Η τεχνητή νοημοσύνη είναι μια ειδικότητα της επιστήμης των υπολογιστών που ασχολείται με τη δημιουργία συστημάτων που μπορούν να αναπαράγουν την ανθρώπινη νοημοσύνη και τις ικανότητες επίλυσης προβλημάτων [38]. Για να επιτευχθεί αυτό χρειάζεται να λάβουν μεγάλο όγκο δεδομένων τα οποία και επεξεργάζονται, ενώ ταυτόχρονα μαθαίνουν από αυτά με στόχο να βελτιωθούν στο μέλλον. Αντίθετα στον κλασικό προγραμματισμό ένα κανονικό πρόγραμμα υπολογιστή θα χρειαζόταν ανθρώπινη παρέμβαση για να διορθώσει σφάλματα και να βελτιώσει τις διαδικασίες [38].

Η ιδέα της «τεχνητής νοημοσύνης» πηγαίνει χιλιάδες χρόνια πίσω αφού απασχόλησε εφευρέτες που ονειρεύονταν από καιρό να δημιουργήσουν μηχανές οι οποίες θα μπορούν να σκέφτονται αυτόνομα. Η πρώτη καταγεγραμμένη εμφάνιση μηχανής η οποία θεωρείτο «αυτόνομη» («δρώντας με τη θέλησή της») έχει καταγραφεί από τον Όμηρο για να περιγράψει ένα αυτόματο άνοιγμα πόρτας, [39] [5.749] ή αυτόματη κίνηση τροχοφόρων τρίποδων [39][18.376].

Με την πάροδο του χρόνου και τη δημιουργία των πρώτων προγραμματιζόμενων υπολογιστών, οι άνθρωποι διερωτήθηκαν κατά πόσο είναι εφικτό να δημιουργηθούν υπολογιστές τόσο με τις δεξιότητες όσο και την ευφυΐα του ανθρώπου. Ήδη από το 1900, στην προσπάθεια τους οι άνθρωποι να κατασκευάσουν τέτοιου είδους υπολογιστές εγκαθιδρύθηκε ο όρος ρομπότ(η λέξη επινοήθηκε σε ένα τσέχικο θεατρικό έργο το 1921) ενώ το 1929 ο Makoto Nishimura

δημιούργησε το πρώτο ρομπότ ονόματι Gakutensoku [40]. Μέσα στις επόμενες δύο δεκαετίες και συγκεκριμένα το 1949 ο Edmund Callis Berkley δημοσίευσε το βιβλίο «Giant Brains, or Machines that Think» το οποίο θεωρείται ένα από τα πρώτα βιβλία που διερευνά την έννοια κατά πόσο η τεχνητή νοημοσύνη έχει την δυνατότητα δημιουργίας υπολογιστών που μπορούν να προσομοιώσουν τις διαδικασίες ανθρώπινης σκέψης, συμπεριλαμβανομένης της συλλογιστικής, της επίλυσης προβλημάτων και της λήψης αποφάσεων [41].

Αν και ο όρος «τεχνητή νοημοσύνη» Νοημοσύνη-ΤΝ (Artificial Intelligence - AI) τυπικά διατυπώθηκε για πρώτη φορά στο Dartmouth College το 1956, σε συνέδριο ερευνητών από τον χώρο της Ψυχολογίας και των Μαθηματικών για τη μελέτη δυνατοτήτων χρήσης των υπολογιστών ως προς την προσομοίωση της ανθρώπινης νοημοσύνης (McCarthy, Allen Newell, Herbert Simon και Marvin Minsky) είχε ήδη αρχίσει να εμφανίζεται από τις αρχές της δεκαετίας του 40, αφού το ενδιαφέρον της επιστημονικής κοινότητας ήταν στραμμένο προς την τεχνητή νοημοσύνη. Το 1950 ο Άλαν Τούρινγκ δημοσίευσε το έργο του «Μηχανήματα Υπολογιστών και Νοημοσύνη» που τελικά έγινε το Τεστ Τούρινγκ (αρχικά ονομάστηκε παιχνίδι μίμησης «imitation game») [42] το οποίο είναι μια δοκιμή της ικανότητας μιας μηχανής να επιδεικνύει ευφυή συμπεριφορά ισοδύναμη ή αδιάκριτη από αυτή ενός ανθρώπου. Η βασική προϋπόθεση του Τεστ Τούρινγκ είναι η εξής:

1. Ένας αξιολογητής (άνθρωπος) συμμετέχει σε συνομιλίες φυσικής γλώσσας τόσο με άνθρωπο όσο και με μηχανή (συχνά μέσω μιας διεπαφής υπολογιστή).

2. Ο αξιολογητής δεν γνωρίζει ποιός είναι ο άνθρωπος και ποιός είναι η μηχανή.

3. Ο στόχος του μηχανήματος είναι να πείσει τον αξιολογητή ότι είναι ο άνθρωπος κατά τη διάρκεια της συνομιλίας.

4. Εάν ο αξιολογητής δεν είναι σε θέση να ξεχωρίσει σταθερά τη μηχανή από τον άνθρωπο με βάση τις απαντήσεις, τότε η μηχανή λέγεται ότι έχει περάσει το τεστ Τυρινγ και παρουσίασε ένα επίπεδο τεχνητής νοημοσύνης συγκρίσιμο με την ανθρώπινη νοημοσύνη.

Στα πρώτα της βήματα, η τεχνητή νοημοσύνη, κατάφερε να αντιμετωπίσει με επιτυχία προβλήματα τα οποία για τον ανθρώπινο παράγοντα ήταν διανοητικά δύσκολα αλλά εύκολα στον προσδιορισμό τους μέσω μαθηματικών κανόνων, τα οποία ικανοποιούσαν τα πρόβλημματα ικανοποίησης περιορισμών(constraint satisfaction problem). Η πραγματική πρόκληση για την τεχνητή νοημοσύνη αποδείχθηκε να ειναι το οτι, άν και ικανή να επιλύσει τις απλές

λειτουργίες που εκτελεί ο άνθρωπος διαισθητικά στην καθημερινότητα του (αναγνωρίζουμε προφορικές λέξεις ή πρόσωπα σε εικόνες) καθίσταται δύσκολο να περιγραφούν μέσω μαθηματικών όρων [38]. Στην προσπάθεια τους οι επιστήμονες να αντιμετωπίσουν αυτή τη νέα πρόκληση που παρουσιάστηκε στράφηκαν στη μελέτη του τρόπου με τον οποίο ο ίδιος ο άνθρωπος μαθαίνει. Η Μάθηση (Learning) είναι μία από τις θεμελιώδεις ιδιότητες της νοήμονος συμπεριφοράς του ανθρώπου κατά την οποία ο άνθρωπος βελτιώνει την απόδοσή του κατά την εκτέλεση μιας συγκεκριμένης εργασίας, χωρίς να υπάρχει ανάγκη να 'προγραμματιστεί' εκ νέου. Παρά τις μελέτες και τις έρευνες επί χρόνια από τους επιστήμονες του πεδίου της Γνωστικής Ψυχολογίας και τους φιλοσόφους, η έννοια της μάθησης δεν έχει γίνει πλήρως κατανοητή. Έχοντας ως αναπάντητο το ερώτημα του μηχανισμού μάθησης του ανθρώπου οι επιστήμονες του χώρου της ΤΝ δεν ήταν σε θέση να δημιουργήσουν υπολογιστικά συστήματα ικανά να κατακτήσουν τη Μηχανική Μάθηση (Machine Learning).

Βάση του ορισμού του Mitchell (1997), «Ένα πρόγραμμα υπολογιστή λέμε ότι μαθαίνει από την εμπειρία E ως προς κάποια κλάση εργασιών T και μέτρο απόδοσης P, αν η απόδοσή του σε εργασίες από το T, όπως μετριέται από το P, βελτιώνεται μέσω της εμπειρίας E.» [43].

Η Μηχανική μάθηση έχει ως στόχο τη δημιουργία ικανών μηχανών οι οποίες κάνοντας χρήση προγενέστερης γνώσης και εμπειρίας θα μαθαίνουν και θα βελτιώνουν την απόδοση τους σε συγκεκριμένα προβλήματα.

Στην αναζήτηση αυτής της γνώσης η μελέτη των βιολογικών νευρωνικών δικτύων στον ανθρώπινο εγκέφαλο θεωρήθηκε αναπόφευκτη με την ελπίδα κατανόησης του τρόπου λειτουργίας **του** στον τομέα μάθησης. Ήδη από τον 19ο αιώνα οι επιστήμονες παραδέχονται ότι ο εγκέφαλος αποτελείται από διακριτά στοιχεία, τους νευρώνες (neurons), που επικοινωνούν το ένα με το άλλο, οι οποίοι και θεωρούνταν ο βασικός πυλώνας για την λειτουργία του εγκεφάλου και συνεπώς για την διαδικασία μάθησης. Στα μέσα της δεκαετίας του 1940 και του 1950, ερευνητές όπως ο (Warren McCulloch και ο Walter Pitts ανέπτυξαν το πρώτο επίσημο μοντέλο τεχνητών νευρώνων, γνωστό ως μοντέλο (McCulloch-Pitts [44]. Αυτό το πρώιμο μοντέλο έθεσε τις βάσεις για μελλοντικές εξελίξεις στα νευρωνικά δίκτυα.

Αντίθετα με την πρώτη περίοδο της τεχνητής νοημοσύνης όπου χρησιμοποιούνταν κυρίως μαθηματικοί κανόνες για την αντιμετώπιση προβλημάτων, η βαθιά μάθηση επιλύει αυτό το κεντρικό πρόβλημα με τη χρήση τεχνητών νευρωνικών δικτύων με πολλαπλά κρυφά επίπεδα (νευρωνικά δίκτυα). Αυτό επιτρέπει την αναγνώριση και κατανόηση πολύπλοκων προτύπων

και αναπαραστάσεων σε μεγάλα σύνολα δεδομένων τα οποία εκφράζονται με όρους άλλων, απλούστερων αναπαραστάσεων (γωνίες-ακμές, περιγράμματα κτλ).

Συνοψίζοντας η βαθιά μάθηση είναι ένα υποσύνολο της μηχανικής μάθησης που αποτελείται από νευρωνικά δίκτυα με τρία ή περισσότερα επίπεδα που επιχειρούν να μιμηθούν τη συμπεριφορά του ανθρώπινου εγκεφάλου μέσω ενός συνδυασμού εισροών μεγάλων όγκων δεδομένων, βαρών επιτρέποντάς του να «μάθει».

Η βαθιά μάθηση έγινε οδηγός για πολλές εφαρμογές και υπηρεσίες τεχνητής νοημοσύνης (AI) οι οποίες βελτιώνουν την αυτοματοποίηση, εκτελώντας αναλυτικές και φυσικές εργασίες χωρίς ανθρώπινη παρέμβαση. Σήμερα, η εφαρμογή της βαθιάς μάθησης στην τεχνητή νοημοσύνη (AI) είναι ένας ακμάζων τομέας με πολλές πρακτικές εφαρμογές και ενεργά ερευνητικά θέματα (κατανόηση ομιλίας και εικόνων, διαγνώσεις στην ιατρική, αυτοοδηγούμενα αυτοκίνητα).

2.3.3 Βασικές Αρχές Νευρωνικών Δικτύων (Neural Networks Fundamentals)

Ο όρος Νευρωνικά Δίκτυα αναφέρεται σε ένα σύνολο μαθηματικών μοντέλων τα οποία είναι εμπνευσμένα από βιολογικά μοντέλα νευρώνων του ανθρώπινου εγκεφάλου και στοχεύουν να μιμηθούν την συμπεριφορά- λειτουργία τους.

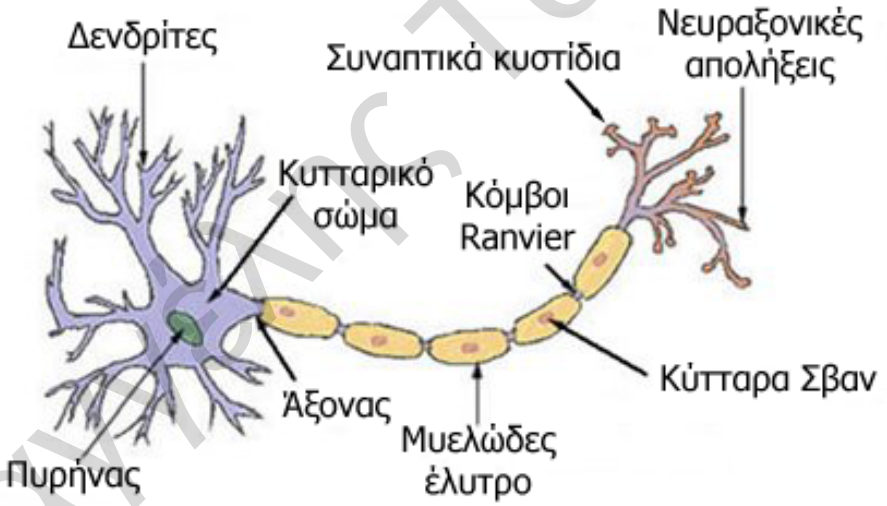
Στα τέλη του 19ου και στις αρχές του 20ου αιώνα, ο Santiago Ramón y Cajal, μαζί με τον Camillo Golgi Golgi, επικεντρώθηκαν στη δομή και οργάνωση τόσο του εγκεφάλου όσο και της οργάνωσης του νευρικού συστήματος. Η σημαντικότερη συνεισφορά τους ήταν η ανακάλυψη ότι ο εγκέφαλος αποτελείται από νευρώνες και η παρατήρηση και περιγραφή της μορφολογίας των νευρώνων, των περίπλοκων δικτύων τους και των συνδέσεων τους μέσω συνάψεων.

Το 1888, ο Santiago Ramón y Cajal δημοσίευσε τη μελέτη με τίτλο "Estructura de los centros nervios de las aves" (Δομή των νευρικών κέντρων των πτηνών), η οποία θεωρήθηκε επαναστατική λόγω του ότι παρουσίασε το γεγονός ότι το νευρικό σύστημα αποτελείται από μεμονωμένα κύτταρα (νευρώνες) αμφισβητώντας την επικρατούσα θεωρία της εποχής ότι το νευρικό σύστημα ήταν ένα συνεχές δίκτυο ιστών γνωστό ως δικτυωτή θεωρία (reticular theory) [45].

Συγκεκριμένα απέδειξε μέσα από τις σχολαστικές μελέτες και τα σχέδιά του ότι οι νευρώνες είναι διακριτές οντότητες με ξεχωριστούς δενδρίτες, κυτταρικά σώματα και άξονες. Βάση των πιο πάνω οι νευρώνες συνιστούν το βασικό δομικό κομμάτι του ανθρώπινου εγκεφάλου. Υπολογίζεται ότι ο εγκέφαλος περιέχει 10 δισ. περίπου νευρώνες τοποθετημένους σε ομάδες, καθεμία από τις οποίες συνιστά ένα φυσικό νευρωνικό δίκτυο. Έτσι, ο ανθρώπινος εγκέφαλος περιέχει εκατοντάδες φυσικά νευρωνικά δίκτυα, καθένα από τα οποία περιέχει χιλιάδες διασυνδεδεμένους νευρώνες με μέσο αριθμό διασυνδέσεων ανά νευρώνα 1000 με 10.000 [46].

Η δομή του νευρώνα απεικονίζεται στο πιο κάτω σχήμα2.6 και αποτελείται απο 3 κύρια τμήματα:

1. τους δενδρίτες (dendrites), οι οποίοι λειτουργούν ως κανάλια εισόδου για το νευρώνα
2. το κυτταρικό σώμα (cell body),
3. τον άξονα του κυττάρου-νευροάξονα (axon), που συνδέει ένα νευρώνα με άλλους



Σχήμα 2.6: Σχηματικό διάγραμμα νευρώνα [47].

Κάθε νευρώνας είναι συνδεδεμένος με τους γειτονικούς νευρώνες μεσώ των δενδριτών και επομένως μέσω αυτών γίνεται η ανταλλαγή σημάτων. Τα σήματα από τους γειτονικούς νευρώνες μεταφέρονται στο σώμα μέσω του άξονα τα οποία συναθροίζονται σε αυτό. Τα σήματα που φτάνουν στο σώμα είναι είτε ανασταλτικά μετασυναπτικά δυναμικά (IPSPs) είτε διεγερτικά μετασυναπτικά δυναμικά (EPSPs). Εάν το άθροισμα των ερεθισμάτων είναι αρκετά μεγάλο έτσι ώστε να ξεπεραστεί το κατώφλι δυναμικού (τιμή κατωφλίου) τότε ο νευρώνας πυροδοτεί δημιουργώντας μια έξοδο (με τη μορφή ηλεκτρικού σήματος) στον άξονά του, η οποία στη

συνεχεία μέσω των συνάψεων θα μεταφερθεί στους γειτονικούς νευρώνες.

Μοντέλα βιολογικού νευρώνα

Κατά τη δεκαετία του 1950 πολλοί προσπάθησαν να κατανοήσουν και να εξηγήσουν του μηχανισμούς που κρύβονταν πίσω από την επικοινωνία μεταξύ των νευρώνων, δηλαδή το πώς δημιουργούνταν τα δυναμικά ενεργείας. Ο Alan Hodgkin και ο Andrew Huxley στην προσπάθεια τους να βρουν τους μηχανισμούς που κρύβονται πίσω από την έναρξη και τη διάδοση των δυναμικών ενεργείας για να εξηγήσουν τους ιοντικούς μηχανισμούς [48], το 1952 διεξήγαγαν πειράματα σε ένα γιγάντιο άξονα καλαμαριού με αποτέλεσμα να δημιούργησουν το μοντέλο Hodgkin-Huxley [49] [50].

Πιο συγκεκριμένα το μοντέλο Hodgkin-Huxley είναι ένα μαθηματικό μοντέλο που βασίζεται στην αγωγιμότητα και περιγράφει τον τρόπο με τον οποίο ξεκινούν και διαδίδονται τα δυναμικά ενεργείας στους νευρώνες. Είναι ένα σύνολο μη γραμμικών διαφορικών εξισώσεων που προσεγγίζει τα ηλεκτρικά χαρακτηριστικά διεγέρσιμων κυττάρων όπως οι νευρώνες και τα καρδιακά μυοκύτταρα [49] [50]. Είναι ένα δυναμικό σύστημα συνεχούς χρόνου με τέσσερις διαφορικές εξισώσεις που απεικονίζονται στο Σχ.2.7

$$I = C_M \frac{dV}{dt} + \bar{g}_K n^4 (V - V_K) + \bar{g}_{Na} m^3 h (V - V_{Na}) + \bar{g}_l (V - V_l),$$

$$dn/dt = \alpha_n (1 - n) - \beta_n n,$$

$$dm/dt = \alpha_m (1 - m) - \beta_m m,$$

$$dh/dt = \alpha_h (1 - h) - \beta_h h,$$

Σχήμα 2.7: Hodgkin and Huxley Equations [48]

Αν και το Hodgkin-Huxley μοντέλο είναι αρκετά καλό, έχει αρκετές αδυναμίες. Για παράδειγμα [51]:

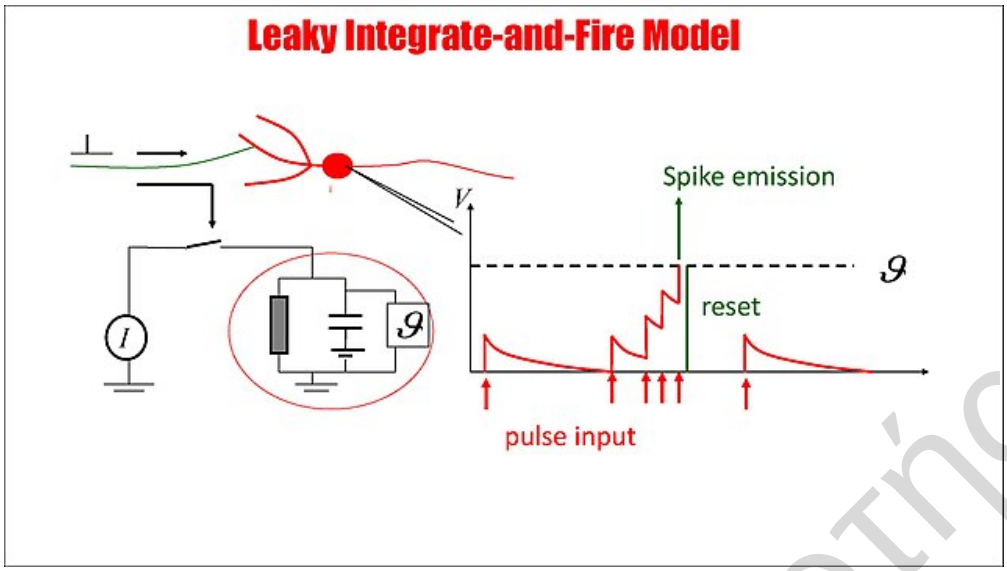
- Λόγω της μεγάλης πολυπλόκότητας που οφείλεται στις τέσσερεις διαφορικές εξισώσεις του, πολλοί ερευνητές σπάνια το χρησιμοποιούν
- Οι εξισώσεις λαμβάνουν υπόψη τους μόνο δύο εξαρτώμενα ρεύματα, αλλά υπάρχει η δυνατότητα να προσθέτουν νέα ρεύματα όπως απαιτείται από τη συγκεκριμένη μεμβράνη που μοντελοποιείται

- Επιπλέον, αυτές οι ντετερμινιστικές εξισώσεις δεν μπορούν να εξηγήσουν τον ιοντικό «θόρυβο του καναλιού» [52] [53].

Λόγω αυτών των μειονεκτημάτων οι ερευνητές για τη μελέτη πολύπλοκων συστημάτων όπως τα εγκεφαλικά νευρωνικά δίκτυα, κάνουν χρήση πιο απλών μοντέλων όπως το Integrate and fire (I&F).

Integrate and fire(I&F)

Ένα από τα πιο ευρέως χρησιμοποιούμενα μοντέλα για την αξιολόγηση της συμπεριφοράς των εγκεφαλικών συστημάτων είναι το μοντέλο νευρώνων integrate-and-fire. Το συγκεκριμένο μοντέλο χαρακτηρίζεται από την απλότητα του λόγω του ότι αποτελείται από μια γραμμική διαφορική εξίσωση (τύπου I). Η κύρια διαφορά του από το μοντέλο H&H είναι ότι αφαιρέθηκαν κάποιοι βιοφυσικοί μηχανισμοί όπως το νάτριο και το κάλιο (στο μοντέλο H&H υπάρχουν 3 διαφορικές εξισώσεις για αυτούς τους μηχανισμούς) ούτως ώστε να γίνει πιο απλό. Όταν το δυναμικό της μεμβράνης φτάσει σε ένα κατώφλι, σχηματίζεται ένα δυναμικό ενεργείας (ακίδα), αλλά οι πραγματικές αλλαγές στην τάση της μεμβράνης και στις αγωγιμότητες που οδηγούν το δυναμικό ενεργείας δεν περιλαμβάνονται στο μοντέλο (βιοφυσικοί μηχανισμοί όπως νάτριο, κάλιο). Οι συναπτικές εισροές στον νευρώνα χαρακτηρίζονται ως μια χρονικά ομοιογενής διαδικασία Poisson και θεωρούνται στοχαστικές. Στην προσέγγιση της διάχυσης, όταν οι μεμονωμένες συνεισφορές στο μετασυναπτικό δυναμικό είναι μέτριες, διερευνώνται μέθοδοι και αποτελέσματα τόσο για τις τρέχουσες συνάψεις όσο και για τις συνάψεις αγωγιμότητας. Το μοντέλο νευρώνων integrate-and-fire έχει καθιερωθεί ως κανονικό μοντέλο για την εξήγηση των νευρώνων με αιχμή. Λόγω της απλότητας του, μπορεί να εξεταστεί μαθηματικά και είναι επίσης αρκετά ικανό ώστε να αντιπροσωπεύει πολλές από τις σημαντικές ιδιότητες της επεξεργασίας του εγκεφάλου. Γενικότερα, το συγκεκριμένο μοντέλο όταν ξεπεράσει το κατώφλι (-50 mV) τότε θα πυροδοτήσει και αμέσως θα επανέλθει στο δυναμικό ισορροπίας όπως φαίνεται και στο Σχ.2.10 [54] [55] [56].



Σχήμα 2.8: Integrate and Fire model [57].

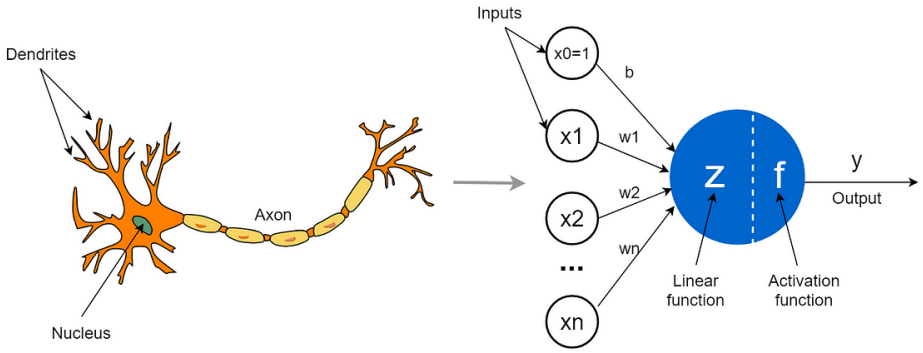
$$\tau_m \frac{dV(t)}{dt} = E_L - V(t) + R_m I_e(t)$$

Σχήμα 2.9: Integrate and Fire Equation [57].

2.3.3.1 Perceptrons and Artificial Neurons

Το 1957 ο Rosenblatt παρουσιάζει το ΤΝΔ Perceptron που είναι προσομοίωση ενός φυσικού νευρώνα. Το ΤΝΔ Perceptron αποτελείται από έναν μόνο νευρώνα και χαρακτηρίζεται ως ένας γραμμικός ταξινομητής (linear classifier).

Γενικά ένας τεχνητός νευρώνας αποτελεί απλοποιημένο μοντέλο του φυσικού νευρώνα. Τα βάρη διασύνδεσης αναπαριστούν τα ηλεκτρικά χαρακτηριστικά της επαφής της σύναψης και η τιμή κατωφλίου προσομοιώνει τη συμπεριφορά κορεσμού του φυσικού νευρώνα [46].



Σχήμα 2.10: Στα αριστερά απεικονίζεται ένας φυσικός νευρώνας, ενώ στα δεξιά διακρίνεται ο τρόπος με τον οποίο αναπαριστάτε έναν τεχνητό νευρώνα *Περσεπτρον* [58].

Βάση των πιο πάνω οι θεμελιώδεις τρόποι λειτουργίας για ένα ΤΝΔ είναι οι εξής:

1. Κάθε τεχνητός νευρώνας αποτελείται από πολλές εισόδους xi και μία μόνο έξοδο y.
2. Κάθε είσοδος xi πολλαπλασιάζεται με το αντίστοιχο βάρος wi
3. Τα αποτελέσματα αθροίζονται μέσω της συνάρτησης αθροίσματος (summation function) F:

$$y = f(\sum_{i=1}^{n} x_i w_i + b) \tag{2.3}$$

4. Ο τεχνητός νευρώνας δίνει έξοδο μέσω της συνάρτησης μετάβασης (transfer function), μόνο όταν το άθροισμα των εισόδων επί των βαρών είναι μεγαλύτερο μιας ορισμένης τιμής κατωφλίου (threshold value) ϑ, δηλαδή όταν:

$$f(\sum_{i=1}^{n} x_i w_i + b) - \vartheta > 0 \tag{2.4}$$

Αρχιτεκτονική ΤΝΔ

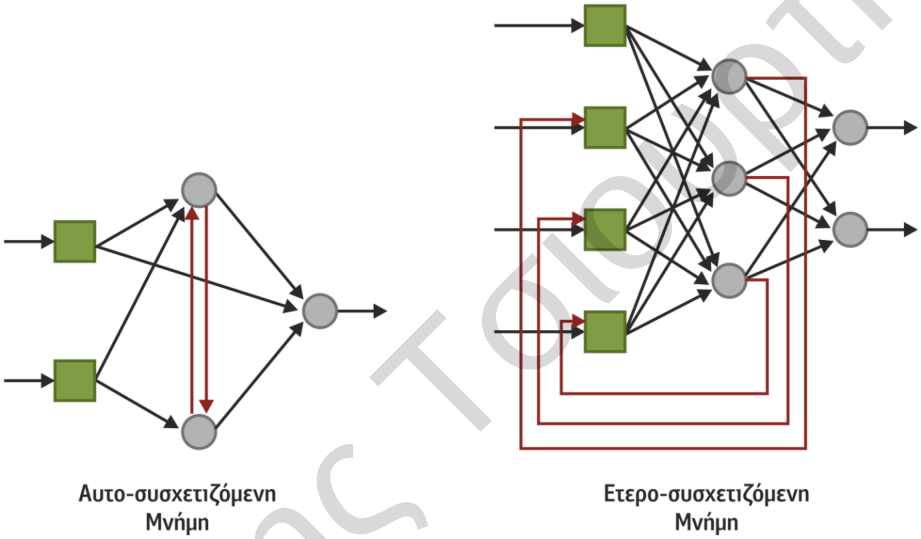
Ο κύριος τρόπος με τον οποίο οι νευρώνες σε ένα νευρωνικό δίκτυο συνδέονται μεταξύ τους καθιστούν ικανό των διαχωρισμό των ΤΝΔ σε δύο κύριες κατηγορίες:

1. πρόσθιας τροφοδότησης (feed forward) και
2. οπίσθιας τροφοδότησης (feed backward)

Στα νευρωνικά δίκτυα πρόσθιας τροφοδότησης (feed forward), οι νευρώνες σε κάθε κρυφό επίπεδο τροφοδοτούνται και τροφοδοτούν τους νευρώνες του αμέσως επόμενου κρυφού επι-

πέδου. Δηλαδή, καμία έξοδος ενός νευρώνα σε οποιοδήποτε κρυφό επιπέδο δεν είναι είσοδος κάποιου νέρωνα του ίδιου ή προηγούμενων επιπέδων. Σχήμα 2.11 .

Σε αντίθεση με τα οπίσθια τροφοδοτούμενα (φεεδ βασκωαρδ) δίκτυα κάποιοι νευρώνες σε κάποια κρυφά επίπεδα τροφοδοτούν τους νευρώνες του ίδιου επιπέδου ή νευρώνες σε άλλα επίπεδα. Εάν ο νευρώνας τροφοδοτεί νευρώνα του ίδιου επιπέδου τότε τα ΤΝΔ καλούνται δίκτυα αυτοσυσχετιζόμενης μνήμης (autoassociated memories). Σε αντίθετη περίπτωση καλούνται δίκτυα ετεροσυσχετιζόμενης μνήμης (heteroassociated memories) Σχήμα 2.11.



Σχήμα 2.11: Παράδειγμα ανατροφοδοτούμενων ΤΝΔ [46].

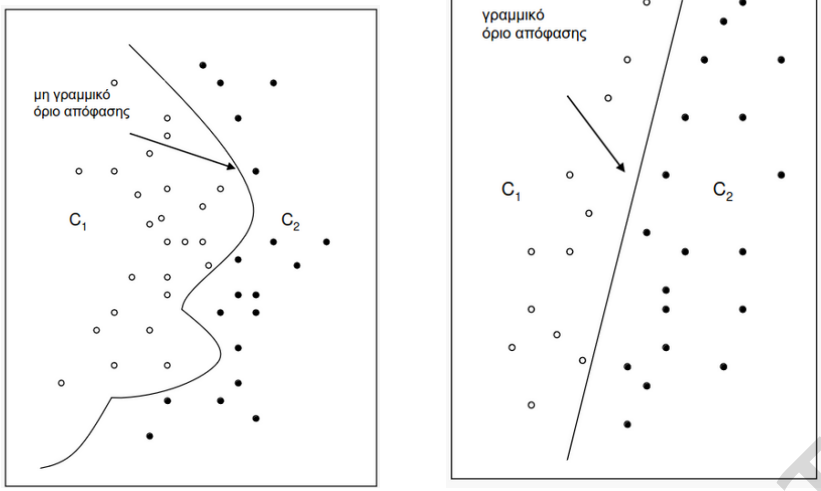
Αυτά τα ανατροφοδοτούμενα ΤΝΔ ονομαζονται και RNNs (recurrent Neural Networks) πχ Jordan, elman, τα οποία θα αναλυθούν περαιτέρω σε μεταγενέστερο στάδιο.

Μέσα στις επόμενες δεκαετίες που ακολούθησαν ο Minsky και Papert [59] το 1969 αποδεικνύουν μαθηματικά ότι τα ΤΝΔ ενός επιπέδου όπως τα δίκτυα Perceptron δεν είναι ικανά να επιλύσουν μη γραμμικά προβλήματα.

Μη γραμμικά προβλήματα

Ένα σύνολο δεδομένων δύο κατηγοριών ονομάζεται γραμμικά διαχωρίσιμο, εάν υπάρχει υπερεπίπεδο που διαχωρίζει τις κατηγορίες. Δηλαδή αφήνει όλα τα πρότυπα της μιας κατηγορίας στο θετικό ημιχώρο και όλα τα πρότυπα της άλλης κατηγορίας στον αρνητικό ημιχώρο

1. Αν δεν ικανοποιείται η παραπάνω ιδιότητα το σύνολο δεδομένων ονομάζεται μη γραμμικά διαχωρίσιμο.



(α') Μή Γραμμικά Διαχωρίσιμο Πρόβλημα

(β') Γραμμικά Διαχωρίσιμο Πρόβλημα

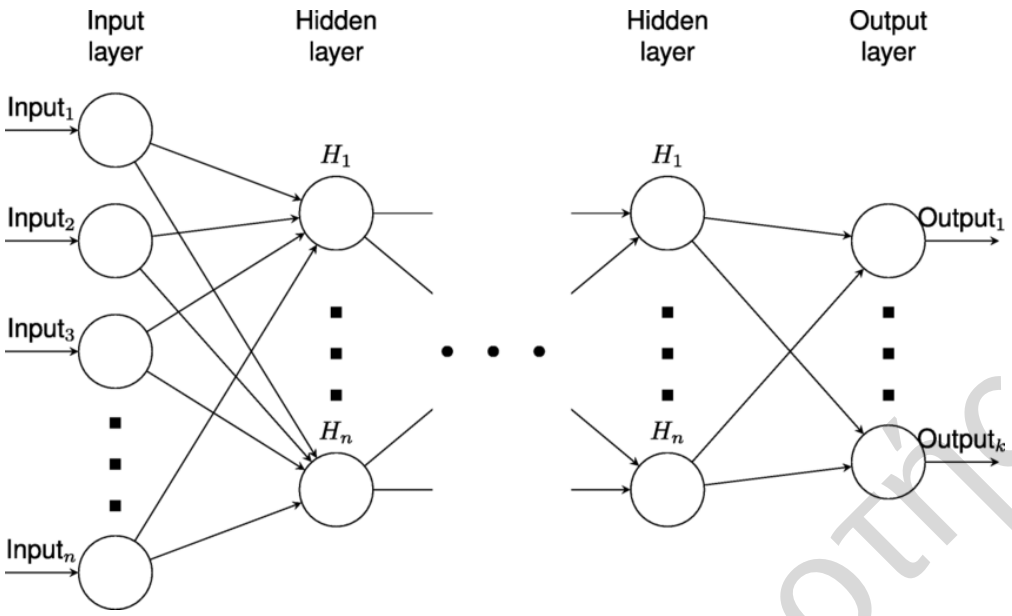
Σχήμα 2.12: Στα αριστερά απεικονίζεται ένα μή γραμμικά διαχωρίσιμο πρόβλημα, ενώ στα δεξιά ένα γραμμικά διαχωρίσιμο πρόβλημα [60]

2. Το διαχωριστικό υπερεπίπεδο εάν υπάρχει δεν είναι μοναδικό.

Πολυεπίπεδα ΤΝΔ

Για την επίλυση μή γραμμικών προβλημάτων εισήχθησαν τα πολυεπίπεδα ΤΝΔ. Με τον όρο πολυεπίπεδα ΤΝΔ ενοούμε ένα νευρωνικό δίκτυο το οποίο διαθέτει τουλάχιστο ένα κρυφό επίπεδο Σχήμα2.13. Τα κύρια χαρακτηριστικά μέσω των οποίο διαχωρίζονται είναι τα εξής:

1. πλήρως συνδεδεμένοι (fully connected) ή μερικώς συνδεδεμένοι (partially connected)
2. πρόσθιας τροφοδότησης (feed forward) ή οπίσθιας τροφοδότησης (feed backward ή recurrent)



Σχήμα 2.13: Παράδειγμα πολυεπίπεδου ΤΝ [61]

2.3.3.2 Activation Functions

Σε όλα τα πιο πάνω ΤΝΔ ο κάθε νευρώνας λειτουργεί ως μια υπολογιστική μονάδα που λαμβάνει εισόδους πολλαπλασιασμένες με τα αντίστοιχα βάρη της κάθε μία και το άθροισμα αυτό περνά μέσω μίας μαθηματικής συναρτήσηςς που μεταφράζει τα σήματα εισόδου σε σήματα εξόδου.

Αυτές οι μαθηματικές συναρτήσεις ονομάζονται συνάρτηση μετάβασης-συνάρτηση ενεργοποίησης (transfer functions-activation functions).

Μέσω αυτών των συναρτήσεων εισάγεται η μη γραμμικότητα στο μοντέλο, επιτρέποντας στα νευρωνικά δίκτυα να προσεγγίζουν και να αναπαριστούν μη γραμμικές σχέσεις στα δεδομένα. Σε περίπτωση μη εισαγωγής των συναρτήσεων μετάβασης στο νευρωνικό δίκτυο θα έχει γραμμική συμπεριφορά με επακόλουθο τον περιορισμό της ικανότητας να μαθαίνει πολύπλοκα και σύνθετα μοτίβα.

Οι κύριοι τύποι γραμμικών συναρτήσεων μεταφοράς είναι:

1. Βηματικές συναρτήσεις ή συναρτήσεις κατωφλίου (threshold functions)
2. Συναρτήσεις προσήμου (sign functions)
3. Συναρτήσεις βηματικής μεταβολής (hard limiter functions)
4. Συναρτήσεις αναρρίχησης (ramping functions)

Συνηθέστερα, όμως, χρησιμοποιούνται μη γραμμικές συναρτήσεις, όπως οι σιγμοειδείς συναρτήσεις (sigmoid functions) και οι Γκαουσιανές συναρτήσεις (Gaussian functions) αφού παρέχουν τα πιο κάτω πλεονεκτήματα:

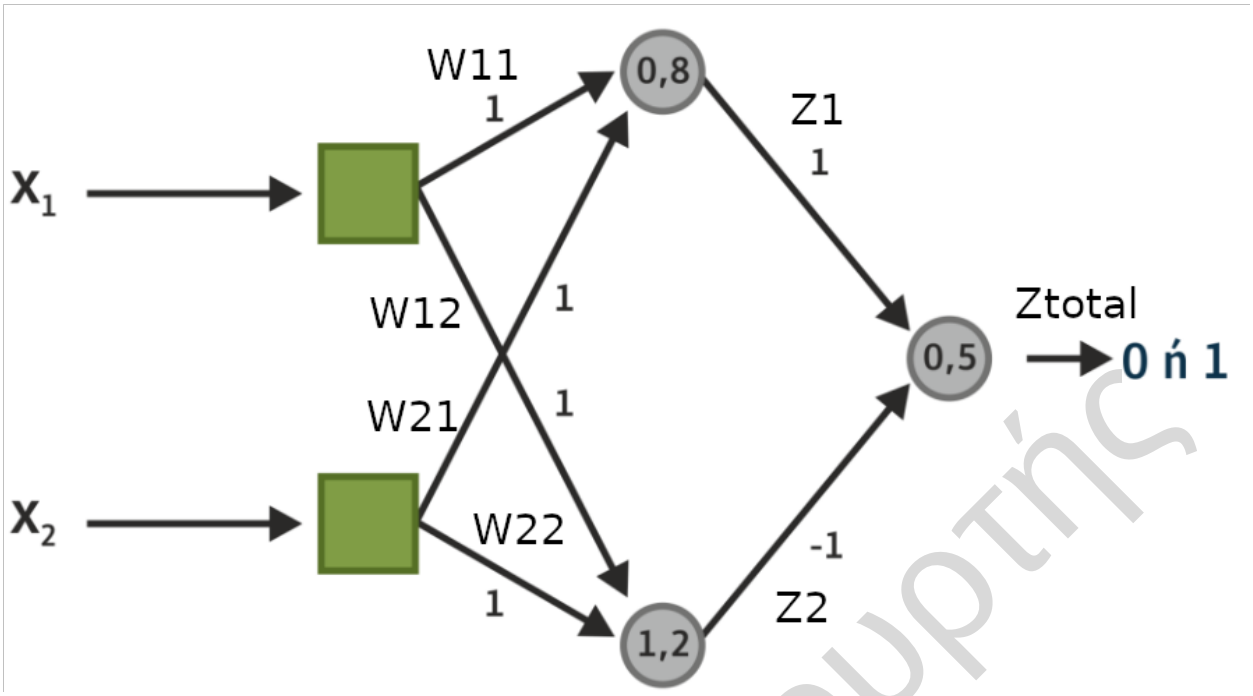
1. Ομαλή και διαφοροποιήσημη (συνεχές):Οι συναρτήσεις Γαυσσιαν και σιγμοειδούς είναι ομαλές και διαφοροποιήσιμες, σε αντίθεση με τη βηματική συνάρτηση η οποία είναι ασυνεχής. Αυτά τα δύο χαρακτηριστικά είναι καθοριστικά για αλγόριθμους βελτιστοποίησης, όπως τον γραδιεντ δεσεντ, οποίοι είναι βασισμένη σε παραγώγους για την ενημέρωση των βαρών του μοντέλου κατά τη διάρκεια της εκπαίδευσης. Αυτό επιτρέπει πιο αποτελεσματική και σταθερή μάθηση στα νευρωνικά δίκτυα.

2. Η έξοδος της σιγμοειδούς συνάρτησης είναι μεταξύ των τιμών 0 και 1 σε αντίθεση με την βηματική συνάρτηση που δίνει τιμές 0 ή 1. Λόγω αυτής της ιδιότητας χρησιμοποιείται συχνά ως συνάρτηση ενεργοποίησης στο επίπεδο εξόδου των νευρωνικών δικτύων για εργασίες δυαδικής ταξινόμησης. Τεμαχίζει την έξοδο σε ένα εύρος μεταξύ 0 και 1, καθιστώντας το κατάλληλο για την εκτίμηση των πιθανοτήτων δυαδικών γεγονότων

3. Η Gaussian συνάρτηση, γνωστή και ως κανονική κατανομή, χρησιμοποιείται ευρέως λόγω του ότι το κέντρο της είναι η αρχή των αξόνων. Συχνά χρησιμοποιείται για τη μοντελοποίηση συνεχών δεδομένων με καλά καθορισμένο μέσο όρο και τυπική απόκλιση.

2.3.3.3 Feedforward Neural Networks (FNNs)

Σε ένα νευρωνικό δίκτυο το πρώτο βήμα για τον υπολογισμό των παραμέτρων του είναι το μπροστινό πέρασμα (Forward Propagation). Τα δεδομένα εισόδου διαδίδονται από το επίπεδο εισόδου στα κρυφά επίπεδα του δικτύου και στο τέλος στο επίπεδο εξόδου. Για κάθε δεδομένο εισόδου υπολογίζεται ένα σταθμισμένο άθροισμα που είναι η είσοδος πολλαπλασιασμένη με το αντίστοιχο βάρος. Στο πιο κάτω Σχήμα2.14 διακρίνεται ένα από τα απλούστερα πολυεπίπεδα ΤΝΔ που αποτελείται από ένα κρυφό επίπεδο, πλήρως συνδεδεμένο και εκτελεί πρόσθια τροφοδότηση.



Σχήμα 2.14: Πολυεπίπεδο ΤΝΔ για το xor [46]

Όπως διακρίνεται στο Σχήμα η είσοδος είναι ένα σύνολο χαρακτηριστικών εισόδου [x1, x2]. Το σταθμισμένο άθροισμα (ΣΑ) για κάθε χαρακτηριστικό εισόδου, προκύπτει από τον υπολογισμό του πολλαπλασιασμού μεταξύ της εισόδου και του αντίστοιχου βάρους και δίνεται από τον εξίσωση:

$$Z1 = (w11 * x1) + (w12 * x2) + b$$

$$Z2 = (w21 * x1) + (w22 * x2) + b$$

όπου (b is the bias term).

Το ΣΑ z1 αντιπροσωπεύει μια γραμμική συνάρτηση των εισόδων, που σημαίνει ότι η σχέση μεταξύ των εισόδων και της εξόδου του perceptron θα είναι γραμμική.

Ωστόσο, η συνάρτηση ενεργοποίησης, που συμβολίζεται ως f(z), εφαρμόζεται στο σταθμισμένο άθροισμα z για την εισαγωγή μη γραμμικότητας.

Ακολούθως το ΣΑ z περνάει μέσα από τη συνάρτηση ενεργοποίησης. Σε αυτό το παράδειγμα η συνάρτηση ενεργοποίησης είναι μία απλή βηματική.

Η συνάρτηση ενεργοποίησης εισάγει μη γραμμικότητα στο perceptron, επιτρέποντάς του να μαθαίνει πολύπλοκα μοτίβα από τα δεδομένα εισόδου. Η τελική έξοδος του είναι

$$Ztotal = fx(Z1 + Z2)$$

Για να είναι ικανό το ΤΝΔ να μας παρέχει σωστές εξόδους ανάλογα με τις εισόδους που λαμβάνουν απαιτείται η εκπαίδευση τους. Κατά τη διάρκεια της εκπαιδευτικής διαδικασίας, τα βάρη του perceptron προσαρμόζονται επαναληπτικά χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης, η οποία θα αναλυθεί μετέπειτα. Αυτή η επαναληπτική διαδικασία συνεχίζεται έως ότου το perceptron μπορεί να ταξινομήσει με ακρίβεια τα δεδομένα εισόδου, μαθαίνοντας αποτελεσματικά τα υποκείμενα μοτίβα και τις σχέσεις στα δεδομένα.

2.3.4 Αλγόριθμος Ανάστροφης Διάδοσης και Βελτιστοποίηση (Back-propagation and Optimization)

Ο όρος βελτιστοποίηση στο πεδίο της μηχανικής μάθησης διαδραματίζει καθοριστικό ρόλο στη εκπαίδευση των μοντέλων και έχει ως στόχο την αύξηση της απόδοσης τους στα προβλήματα με τα οποία αλληλεπιδρούν. Η διαδικασία της μάθηση και της βελτιστοποίησης θεωρείται αλληλένδετη αλλά όχι πανομοιότυπη. Η μάθηση μπορεί να θεωρηθεί ως πρόβλημα βελτιστοποίησης για την εύρεση των παραμέτρων του μοντέλου. Η εύρεση των κατάλληλων παραμέτρων για το μοντέλο ελαχιστοποιεί μια συγκεκριμένη συνάρτηση απώλειας, μειώνοντας τη διαφορά μεταξύ των προβλέψιμων τιμών του μοντέλου και των πραγματικών τιμών στόχου στα δεδομένα εκπαίδευσης και ως επακόλουθο αυξάνει την απόδοση του μοντέλου.

Πιο συγκεκριμένα για την αύξηση της απόδοσης εκτός από την εύρεση των κατάλληλων παραμέτρων που γίνεται συνήθως μέσω της ορθής επιλογής αλγορίθμου βελτιστοποίησης, η ορθή επιλογή του ρυθμού εκμάθησης και άλλων υπερπαραμέτρων επηρεάζει σημαντικά τη διαδικασία εκμάθησης, την ταχύτητα σύγκλισης, τη σταθερότητα και τη συνολική απόδοση του μοντέλου στη δεδομένη εργασία. Οι αποτελεσματικές τεχνικές βελτιστοποίησης και οι καλά συντονισμένες υπερπαράμετροι είναι απαραίτητες για την επιτυχή εκμάθηση και την ανάπτυξη ακριβών και υψηλής απόδοσης μοντέλων μηχανικής εκμάθησης.

2.3.4.1 Backpropagation Algorithm

Ο Backpropagation αλγόριθμος, θεωρείται ως η βάση για τη διαδικασία μάθησης στα νευρωνικά δίκτυα και είναι απαραίτητο για τη βελτιστοποίηση της απόδοσης τους. Συγκεκριμένα είναι ο μηχανισμός για τον υπολογισμό των κλίσεων κατά τη διάρκεια της εκπαιδευτικής διαδικασίας, οι οποίες υποδεικνύουν την κατεύθυνση της πιο απότομης αύξησης στη συνάρτηση απώλειας προσαρμόζοντας επαναληπτικά τα βάρη του ΤΝΔ.

Ο Backpropagation αποτελείται εκτελείται σε δύο φάσεις:

1. Ένα πέρασμα προς τα εμπρός

   Είναι η διαδικασία που αναφέρθηκε προηγούμενος στα Feedforward Neural Networks.Τα δεδομένα εισόδου αφού πολλαπλασιαστούν με τα αντίστοιχα βάρη και περάσουν από της συναρτήσεις ενεργοποίησης θα καταλήξουν να δημιουργήσουν τις εξόδους του δικτύου. Στη συνέχεια, αυτές οι εξόδοι συγκρίνονται με τις πραγματικές τιμές στόχου και η απόκλιση μεταξύ τους ποσοτικοποιείται χρησιμοποιώντας μια επιλεγμένη συνάρτηση απώλειας.

2. Ένα πέρασμα προς τα πίσω

   Στο πίσω πέρασμα, ο αλγόριθμος υπολογίζει τις κλίσεις της απώλειας σε σχέση με κάθε παράμετρο, λειτουργώντας αναδρομικά από το επίπεδο εξόδου πίσω στο επίπεδο εισόδου. Αυτές οι διαβαθμίσεις υποδεικνύουν πόσο συμβάλλει κάθε παράμετρος στο σφάλμα. Οι υπολογιζόμενες διαβαθμίσεις χρησιμοποιούνται στη συνέχεια για την ενημέρωση των βαρών και των προκαταλήψεων του δικτύου χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης(BGD,SGD,Adam..), που ωθεί τις παραμέτρους σε μια κατεύθυνση που μειώνει το σφάλμα. Με την επανάληψη αυτής της διαδικασίας σε πολλαπλές επαναλήψεις, ο backpropagation επιτρέπει στο δίκτυο να μάθει και να προσαρμόσει τις παραμέτρους του για να προσεγγίσει καλύτερα τα υποκείμενα μοτίβα στα δεδομένα. Είναι μια ισχυρή τεχνική που στηρίζει την εκπαίδευση σύνθετων νευρωνικών αρχιτεκτονικών, δίνοντάς τους τη δυνατότητα να υπερέχουν σε ένα ευρύ φάσμα εργασιών, από την αναγνώριση εικόνας έως την επεξεργασία φυσικής γλώσσας.

Περιληπτικά ο backpropagation είναι υπεύθυνος για τον υπολογισμό των κλίσεων που υποδεικνύουν τον τρόπο προσαρμογής των βαρών του δικτύου για τη μείωση του σφάλματος μεταξύ προβλέψεων και τιμών στόχου. Στη συνέχεια οι αλγόριθμοι βελτιστοποίησης χρησιμοποιούν αυτές τις διαβαθμίσεις για να ενημερώσουν τα βάρη με στόχο την βελτίωση της

απόδοσης του ΤΝΔ. Αυτή η αλληλένδετη συνεργασία μεταξύ του backpropagation και των αλγορίθμων βελτιστοποίησης για την εκπαιδεύσει των ΤΝΔ μέσα από την επαναληπτική μεταβολή των βαρών βάση τις υπολογισμένες κλίσεις, οδηγεί σε ολοένα και πιο βελτιωμένες και γενικευμένες εξόδους των ΤΝΔ.

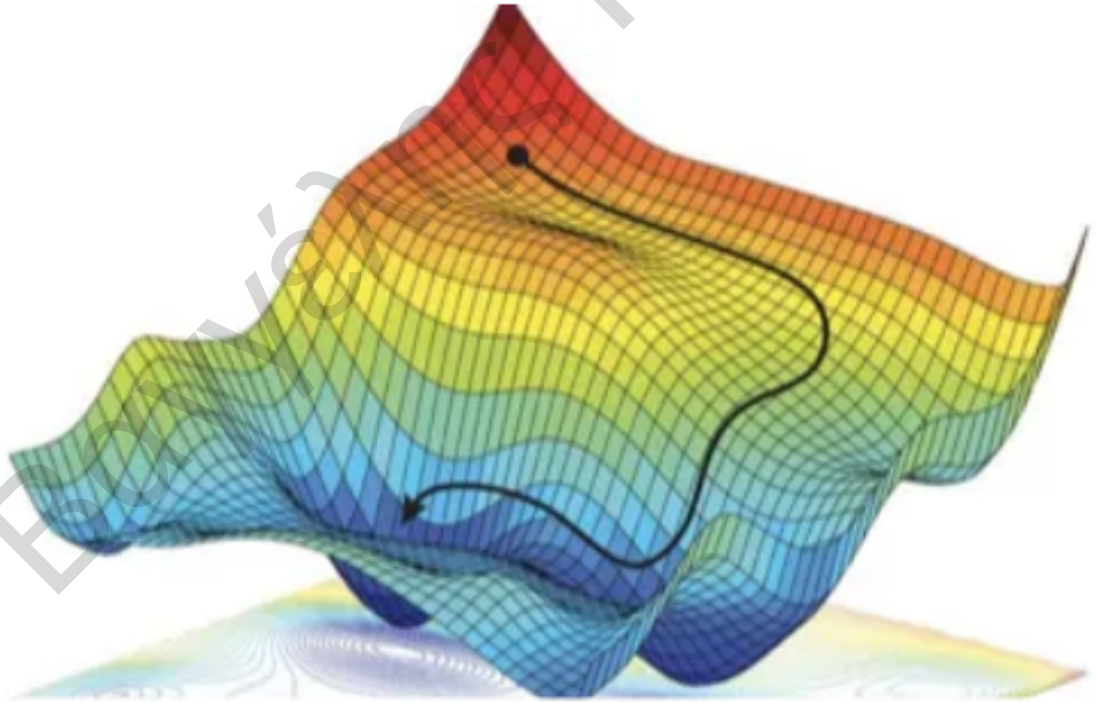
Στα επόμενα υποκεφάλαια θα παρουσιαστούν οι κυριότεροι αλγόριθμοι βελτιστοποίησης.

2.3.4.2 Gradient Descent (GD) - Gradient Ascent (GA)

Το Gradient Descent και το Gradient Ascent είναι δύο τεχνικές βελτιστοποίησης που χρησιμοποιούνται στους αλγόριθμους ΜΜ και μπορεί να θεωρηθούν ουσιαστικά αντίθετες μεταξύ τους.

Gradient Descent (GD)

Ο αλγόριθμος Gradient Descent (GD) μπορεί να θεωρηθεί ως ένας από τους θεμελιώδεις αλγορίθμους βελτιστοποίησης ο οποίος χρησιμοποιείται ευρέως στη ΜΜ βρίσκοντας το ελάχιστο μίας συνάρτησης.



Σχήμα 2.15: Κάθοδική κλίση σε χώρο τριών διαστάσεων για ελαχιστοποίηση της συνάρτησης [62]

Η βασική λειτουργίας του αλγορίθμου επιτυγχάνεται μέσω επαναληπτικής διαδικασίας. Συγκεκριμένα ενημερώνει τις παραμέτρους (βάρη και bias) του μοντέλου προς την κατεύθυνση της μεγαλύτερης αρνητικής κλίσης(πιο απότομη μείωση) της παραγώγου της συνάρτησης απώλειας. Ακολουθώντας αυτή την απότομη κάθοδο της κλίσης με μέγεθος βημάτων ανάλογα με τον ρυθμό εκμάθησης, ο αλγόριθμος καταλήγει σε ένα τοπικό ελάχιστο (local minimum) ή το συνολικό ελάχιστο της συνάρτησης ελαχιστοποιώντας το error function E.

$$w := w - a \cdot \nabla_w f(w)$$

w is the parameter vector being updated

a learning rate

$\nabla_w f(w)$ is the gradient of the function f with respect to the parameters w

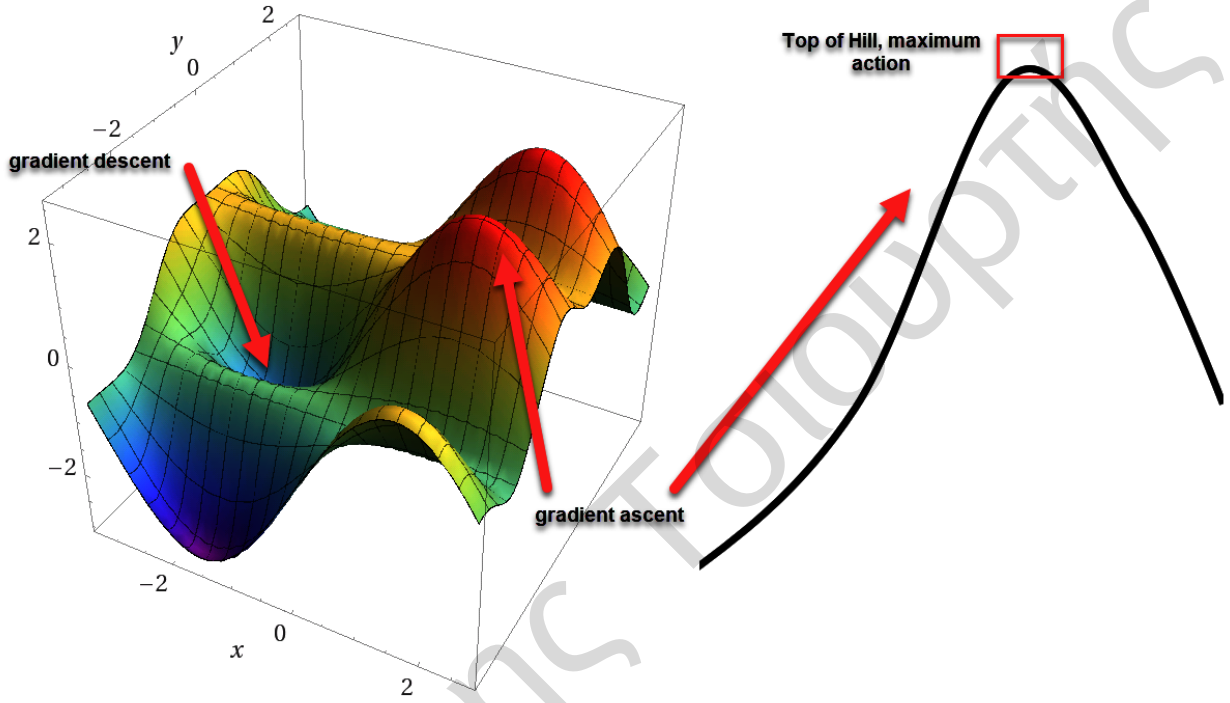
Βασισμένος στην απλότητα, την επιτυχία εφαρμογής του σε διάφορους τομείς αλλά και την αποτελεσματικότητα του σε αυτούς, ο αλγόριθμος Γραδιεντ Δεσεντ το εγκαθιδρύθηκε ως βασικό εργαλείο στην ΜΜ. Ο Gradient Descent μπορεί να παρουσιαστεί και ως η βάση κπαίδευσης διαφόρων μοντέλων μηχανικής μάθησης, συμπεριλαμβανομένων των νευρωνικών δικτύων, καθώς τους επιτρέπει να βελτιώνουν επαναληπτικά τις παραμέτρους τους για να ταιριάζουν στα δεδομένα και να βελτιώνουν την απόδοσή τους σε μια δεδομένη εργασία. Έχοντας αντιληφθεί όλα αυτά τα προτερήματα του ΓΔ οι ερευνητές προσπάθησαν να βελτιώσουν περαιτέρω την αποτελεσματικότητα του και να γενικευτεί για να εφαρμοστεί σε όλο και πιο περίπλοκα, μεγάλα προβλήματα της ΜΜ. Στην προσπάθεια τους αυτή δημιουργήθηκαν παραλλαγές του:

1. Batch Gradient Descent
2. Stochastic Gradient Descent (SGD)
3. Mini-Batch Gradient Descent

καθεμία από αυτές μετά πλεονεκτήματα και τα μειονέκτημα της. Οι ερευνητές και οι επαγγελματίες συνεχίζουν να εξερευνούν νέες παραλλαγές και τεχνικές για να βελτιστοποιήσουν περαιτέρω το Gradient Descent, με στόχο να βελτιώσουν την αποτελεσματικότητά του και την εφαρμογή του σε όλο και πιο περίπλοκα και μεγάλης κλίμακας προβλήματα μηχανικής μάθησης.

Gradient Ascent GA)

Σε αντίθεση με τον GD ο οποίος προσπαθεί να ελαχιστοποιήσει μία συνάρτηση, ο GA χρησιμοποιείται για τη μεγιστοποίηση μιας συνάρτησης. Συγκεκριμένα ο GA αντί να κινείται προς την κατεύθυνση με μεγαλύτερη αρνητική (πιο απότομη μείωση) κλίση της συνάρτησης, όπως στο GD, κινείται προς την κατεύθυνση της πιο απότομης αύξησης της συνάρτησης.



Σχήμα 2.16: Ανοδική κλίση σε χώρο τριών διαστάσεων για μεγιστοποίηση της συνάρτησης [63]

Εδώ πρέπει να αναφερθεί ότι αυτή η τεχνική χρησιμοποιείται λιγότερο συχνά σε περιβάλλοντα ΜΜ (πχ. Generative Models Policy Gradients, Natural Language Processing (NLP)) σε σύγκριση με το GD ο οποίος όπως αναφέρθηκε τέθηκε η βάση για ανάπτυξη πολλών νέων αλγορίθμων βελτιστοποίησης. Ο GA χρησιμοποιείται συχνά όταν αντιμετωπίζουμε προβλήματα βελτιστοποίησης όπου ο στόχος είναι η εύρεση παραμέτρων που μεγιστοποιούν μια συγκεκριμένη αντικειμενική συνάρτηση.

$$w := w + a \cdot \nabla_w f(w)$$

w is the parameter vector being updated

a learning rate

$\nabla_w f(w)$ is the gradient of the function f with respect to the parameters w

2.3.4.3 Stochastic Gradient Descent

Το Stochastic Gradient Descent (SGD) θεωρείτε πλέον μια δημοφιλής τεχνική βελτιστοποίησης μέσω ελαχιστοποίησης της συνάρτησης απώλειας κατά την εκπαίδευση μοντέλων μηχανικής μάθησης. Ο κύριος λόγος που αναπτύχθηκε ο SGD είναι για να ξεπεράσει τους περιορισμούς και προκλήσεις που αντιμετωπίζει ο GD κατά την εκπαίδευση μοντέλων μηχανικής μάθησης μεγάλης κλίμακας και πολύπλοκων σεναρίων. Για την αντιμετώπιση αυτών των προκλήσεων ο SGD δεν κάνει επεξεργασία ολόκληρου του συνόλου δεδομένων αλλά ενός τυχαίου υποσυνόλου το οποίο αποκαλείται ¨μίνι-παρτίδα¨(Minibatches). Μέσω αυτής της στοχαστικής δειγματοληψίας δεδομένων επιτυγχάνεται ως επακόλουθο και η ταχύτερη σύγκλιση και καλύτερη γενίκευση.

Επιλέγοντας αυτές τις μίνι-παρτίδες η ενημέρωση των βαρών γίνεται μόνο με τον υπολογισμό της κλίσης σε αυτό το σύνολο δεδομένων. Η λειτουργία μείωσης της κλίσης μπορεί να επιτευχθεί είτε αθροιστικά είτε με μέσο όρο. Η επιλογή με πιο τρόπο θα υπολογιστεί η μείωση της κλίσης συνήθως εξαρτάται κυρίως από:

1. Μέγεθος των δεδομένων
2. Σύνολο υπολογιστικών πόρων

2.3.5 Τύποι Νευρωνικών Δικτύων (Types of Neural Networks)

2.3.5.1 Convolutional Neural Networks (CNNs)

Αν και τα πολυεπίπεδα νευρωνικά δίκτυα έχουν επιτύχει να επιλύουν σύνθετα και μη γραμμικά προβλήματα, στην εφαρμογή τους σε οπτικά δεδομένα δεν επιτυγχάνουν τόσο καλά αποτελέσματα. Το κύριο μειονέκτημα τους είναι ότι δεν είναι ικανά να αντιληφθούν ότι μία εικόνα είναι η ίδια απλά μετατοπισμένη κατά έναν αριθμό εικονοστοιχείων. Εδώ έρχονται τα βαθιά Συνελικτικά Νευρωνικά Δίκτυα (CNN) τα οποία έχουν αναδειχθεί μέσω της ικανότητας τους να επιτρέπουν στις μηχανές να εντοπίζουν και να αναλύουν οπτικά δεδομένα (πχ φωτεινά ή σκοτεινά σημεία, ακμές σε διάφορους προσανατολισμούς, μοτίβα)σε εικόνες και βίντεο με πρωτοφανή ακρίβεια. Μέσω της ικανότητας εξαγωγής οπτικών δεδομένων τα δίκτυα είναι ικανά να εκτελέσουν εργασίες όπως ταξινόμηση εικόνων, ανίχνευση αντικειμένων κτλ. Εδώ πρέπει να αναφερθεί ότι τα ΝΝ είναι εμπνευσμένα από την οργάνωση του οπτικού φλοιού των θηλαστικών. Δηλαδή συγκεκριμένη νευρώνες ενεργοποιούνται σε συγκεκριμένα εξωτερικά ερεθίσματα, που απορρέει από το πείραμα του Huble & Wiesel το 1962 σε μία γάτα [64].

Η αρχιτεκτονική ενός CNN αποτελείται από πολλά βασικά στοιχεία όπως διακρίνεται και στο Σχήμα 2.17:

1. Επίπεδο εισόδου - Input Layer: Είναι το επίπεδο το οποίο δέχεται τα ακατέργαστα δεδομένα μία εικόνας.

2. Συνελικτικό στρώμα - Convolutional Layer: Συνελικτικά Φίλτρα τα οποία εφαρμόζονται στα δεδομένα εισόδου για την εξαγωγή χαρακτηριστικών.

3. Συνάρτηση ενεργοποίησης - Activation Function: Οι συναρτήσεις ενεργοποίησης όπως αναφέρθηκε και προηγουμένως εισάγει την μη γραμμικότητα στο δίκτυο καθιστώντας το ικανό να μαθαίνει πολύπλοκα μοτίβα δεδομένων. Η συνηθέστερη συνάρτηση ενεργοποίησης είναι η ReLU (Rectified Linear Unit) η οποία επιτρέπει ταχύτερη και πιο αποτελεσματική εκπαίδευση σε μεγάλα και πολύπλοκα σύνολα δεδομένων.

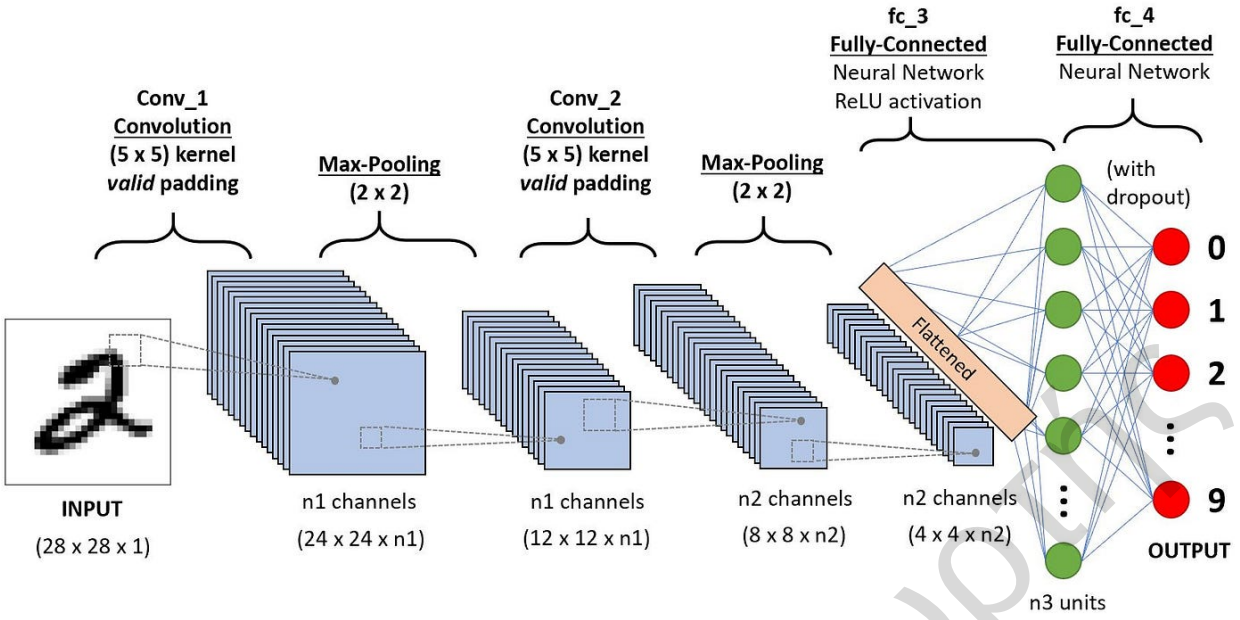
4. Επίπεδο συγκέντρωσης - Pooling Layer: Το Poolong Layer κάνει μείωση των δειγμάτων που λήφθηκαν ως είσοδος κρατώντας τις σημαντικές πληροφορίες. Αυτό βοηθά στη μείωση της πολυπλοκότητας αφού υπάρχουν λιγότερες παράμετροι χαρακτηριστικών ενώ παράλληλα παρέχεται μία υποδειγματολιπριμένει εικόνα της αρχικής η οποία είναι

αναλλοίωτη στις μικρές μετατοπίσεις των γειτονικών εικονοστοιχείων. Υπάρχουν κυρίως δύο τρόποι που εκτελείται το Pooling:

- Max pooling. Επιλέγει το στοιχείο με την μεγαλύτερη τιμή
- Average polling Επιλέγει το στοιχείο με την μέση τιμή όλων των στοιχείων του πίνακα.

5. Ισοπέδωση - Flatten: Πριν την είσοδο των δεδομένα στα πλήρως συνδεδεμένα επίπεδα, οι πίνακες χαρακτηριστικών που περιλαμβάνουν τα δεδομένα και έχουν την μορφή δύο(2D) ή τριών(3D) διαστάσεων μετασχηματίζονται σε ένα διάνυσμα 1D.
6. Πλήρως συνδεδεμένο (FC) στρώμα: Κάθε νευρώνας συνδέεται με κάθε νευρώνα στο προηγούμενο στρώμα, σχηματίζοντας ένα πλήρως συνδεδεμένο γράφημα. Τα πλήρως συνδεδεμένα στρώματα χρησιμοποιούνται για ταξινόμηση των εικόνων.
7. Επίπεδο εξόδου: Το Το τελικό επίπεδο που παράγει τις προβλέψεις του δικτύου. Ο αριθμός των νευρώνων σε αυτό το στρώμα εξαρτάται από τη συγκεκριμένη εργασία. Για παράδειγμα, όπως διακρίνεται στο Σχήμα υπάρχουν μία έξοδος που αντιστοιχεί ` στο κάθε μέσω μεταφοράς.

Η εκπαίδευση των CNN γίνεται μέσω αγλορίθμων βελτιστοποίησης όπως ο Stochastic Gradient Descent (SGD) ή ο Adam. Η ενημέρωση των βαρών του δικτύου επιτυγχάνεται μέσω του backpropagation ο οποίος υπολογίζει τις διαβαθμίσεις των κλήσεων.



Σχήμα 2.17: Αρχιτεκτονική ενός CNN [65]

Τα κύρια μειονεκτήματα που αντιμετωπίζουν τα CNN είναι:

1. Μπορούν να εφαρμοστούν μόνο σε ομοιογενής δεδομένα(πχ εικόνες, βίντεο κτλ)
2. Ευαίσθητα στο θόρυβο
3. Δεν παρέχεται επεξήγηση για τα αποτελέσματα τα οποία παρέχουν, δηλαδή για πιο λόγο έλαβαν αυτήν την απόφαση. Αυτό έχει οδηγήσει στην δημιουργία του νέου πεδίου επεξηγηματικότατης της τεχνητής νοημοσύνης το eXplainable AI(XAI).

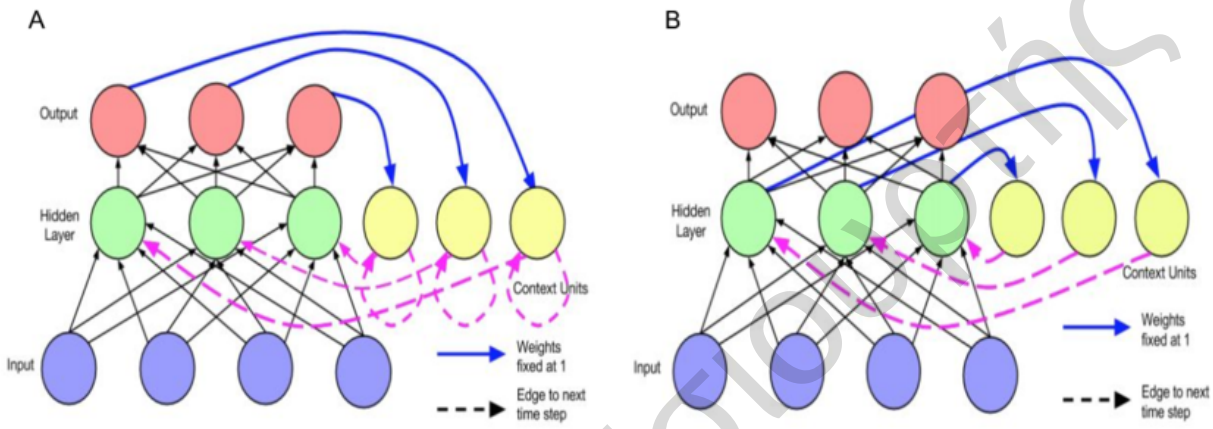
2.3.5.2 Recurrent Neural Networks (RNNs)

Τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) έχουν βρει εφαρμογή κυρίως στο τομέα ανάλυσης δεδομένων λόγο της ικανότητας τους να εντοπίζουν χρονικές εξαρτήσεις και μοτίβα μέσα σε ακολουθίες δεδομένων. Για να το επιτύχουν αυτό τα PNN κάνουν χρήση εσωτερικής κατάστασης μνήμης η οποία καταγράφει τις κύριες αλλαγές στο περιβάλλον που θεωρούνται κρίσιμες για την ανάλυση των ακολουθιών των δεδομένων. Ο υπολογισμός του διανύσματος επόμενης κατάστασης μέσω του συνδυασμού του διανύσματος τρέχουσας κατάστασης και του διανύσματος εισόδου του δικτύου.

Το κύριο χαρακτηριστικό στην αρχιτεκτονική των RNN είναι η ανατροφοδότηση της εξόδου του δικτύου πίσω στον εαυτό του δημιουργώντας ένα βρόχο που διευκολύνει τη διατήρηση

πληροφοριών και επεξεργασίας μεταβλητού μήκους ακολουθιών. Βάση των πιο πάνω τα RNN εφαρμόζονται για επεξεργασία φυσικής γλώσσας(πχ κείμενα), αναγνώριση ομιλίας και πρόβλεψη χρονοσειρών (πχ μελλοντικές τιμές μετοχών).

Δυο κύρια RNN είναι το Jordan και ELMA τα οποία σχεδιάστηκαν για να καταγράφουν χρονικές εξαρτήσεις σε διαδοχικά δεδομένα, το καθένα με τη δική του ξεχωριστή δομή και χαρακτηριστικά.



Σχήμα 2.18: Το Σχήμα Α στα αριστερά αναπαριστά την αρχιτεκτονική Jordan RNN.
Το Σχήμα Β στα δεξιά αναπαριστά την αρχιτεκτονική Elman RNN [66]

Jordan

Όπως διακρίνεται από το σχήμα η **έξοδος του ΤΝΔ** σε κάθε χρονικό βήμα γίνεται ξανά ως είσοδος στο σύστημα μέσω ενός επιπέδου νευρώνων που ονομάζεται context layer. Επιπλέον χαρακτηριστικά:

1. Όσο πιο πολύ νευρώνες υπάρχουν στο κάθε κάθε κρυφό επίπεδο τόσο πιο παλιά δεδομένα είναι ικανό το δίκτυο να θυμάται.
2. Ο αριθμός εξόδων είναι ίσος με τον αριθμών των νευρώνων στο context layer
3. Τα βάρη μεταξύ της εξόδου και του context layer είναι ίσο με 1 λόγο του ότι θέλουμε να παραμείνει αναλλοίωτη η πληροφορία
4. Η τρέχουσα κατάσταση είναι συνάρτηση της προηγούμενης εξόδου του ΤΝΔ και τρέχουσας εισόδου του ΤΝΔ

Elman

Όπως διακρίνεται από το σχήμα η **έξοδος του κρυφού επιπέδου** γίνεται ξανά ως είσοδος στο σύστημα μέσω ενός επιπέδου νευρώνων που ονομάζεται Context Units layer or state units layer. Επιπλέον χαρακτηριστικά :

1. Όσο πιο πολύ νευρώνες υπάρχουν στο κάθε κάθε κρυφό επίπεδο τόσο πιο παλιά δεδομένα είναι ικανό το δίκτυο να θυμάται.
2. Ο αριθμός εξόδων είναι ίσος με τον αριθμών των νευρώνων στο context Unit layer
3. Τα βάρη μεταξύ της εξόδου και του context unite είναι ίσο με 1 λόγο του ότι θέλουμε να παραμείνει αναλλοίωτη η πληροφορία
4. Η τρέχουσα κατάσταση είναι συνάρτηση της προηγούμενης εξόδου του κρυφού επιπέδου και τρέχουσας εισόδου του ΤΝΔ

Η εκπαίδευση των RNN περιλαμβάνει περάσματα προς τα εμπρός και προς τα πίσω για τον υπολογισμό των κλίσεων, επιτρέποντας ενημερώσεις βάρους μέσω αλγορίθμων βελτιστοποίησης όπως ο SGD ή ο Adam. Καθώς τα δεδομένα γίνονται πιο δυναμικά και εξαρτώνται από το περιβάλλον, η σημασία των RNN αυξάνεται, οδηγώντας την καινοτομία και τις ανακαλύψεις στην κατανόηση διαδοχικών δομών δεδομένων.

2.3.5.3 Long Short-Term Memory Networks (LSTM)

Vanishing gradient problem (VGP)

Το πρόβλημα της κλίσης εξαφάνισης (VGP) είναι ένα φαινόμενο που εμφανίζεται κατά την εκπαίδευση των βαθιών νευρωνικών δικτύων όπως τα deep feedforward networks και RNN. Υπενθυμίζουμε ότι κατά τη διαδικασία εκπαίδευσης του νευρωνικού δικτύου, στόχος είναι η ελαχιστοποίηση μιας συνάρτησης απώλειας προσαρμόζοντας τα βάρη του δικτύου με στόχο το πραγματικό αποτέλεσμα (actual output) του δικτύου να είναι όσο το δυνατό πιο κοντά στο επιθυμητό (target output). Ο αλγόριθμος backpropogation υπολογίζει αυτές τις κλίσεις μεταδίδοντας το σφάλμα από το επίπεδο εξόδου προς τα πίσω μέχρι το επίπεδο εισόδου.

Υπενθύμιση διαδικασίας:

1. **Forward pass**

   Η είσοδος διαδίδεται μέσω του δικτύου δια μέσου όλων των κρυφών επιπέδων, για τον υπολογισμό των τιμών εξόδου.

2. **Loss calculation**

   Αφού γίνει το πέρασμα προς τα εμπρός, οι υπολογιζόμενες εξόδοι συγκρίνονται με τις επιθυμητές τιμές για τον υπολογισμό της συνάρτησης απώλειας (loss fnction).

3. **Backpropogation**

   Υπολογισμός των κλίσεων της συνάρτησης απώλειας σε σχέση με τα βάρη του δικτύου.

4. **Weight update**

   Μετά τον υπολογισμό των κλίσεων, τα βάρη του δικτύου ενημερώνονται με σκοπό να να ελαχιστοποιηθεί η συνάρτηση απώλειας.

Το VGP εμφανίζεται όταν κλίσεις (gradients) της συνάρτησης απώλειας που υπολογίζονται και χρησιμοποιούνται για την ενημέρωση των παραμέτρων του δικτύου γίνονται εξαιρετικά μικρές ή «εξαφανίζονται» σε σχέση με αυτές καθώς μεταδίδονται από τα στρώματα εξόδου στα προηγούμενα στρώματα.

Το πρόβλημα της διαβάθμισης εξαφάνισης είναι ιδιαίτερα έντονο σε δίκτυα που χρησιμοποιούν συναρτήσεις ενεργοποίησης με παράγωγα που είναι κοντά στο μηδέν για ένα σημαντικό μέρος του τομέα εισόδου τους.

1. Sigmoid Activation (Logistic Activation)
2. Hyperbolic Tangent Activation (tanh)

Παράδειγμα με την sigmoid activation function:
$\sigma(x) = \frac{1}{1+e^{-x}}$
και η παράγωγος του είναι :
$\sigma'(x) = \sigma(x).(1 - \sigma(x))$

Εδω μπορούν να παρατηρηθούν δύο περιπτώσεις:

1. Για κάθε πολύ μεγάλη θετική τιμή του x

   $x : lim_{\to+\infty}\sigma(x) = 1$

   Επομένως όταν το ξ παίρνει μία μεγάλη θετική τιμή, το e^{-x} τίνει το μηδέν, καθιστώντας τον παρονομαστή της σιγμοειδούς συνάρτησης πολύ μεγάλο. Ως αποτέλεσμα, η ίδια η σιγμοειδής συνάρτηση πλησιάζει το 1, γεγονός που καθιστά την παράγωγο $\sigma'(x)$ πολύ κοντά στο 0.

2. Για κάθε πολύ μεγάλη αρνητική τιμή του x

   $x : lim_{\to-\infty}\sigma(x) = 0$

   Ομοίως, πιο πάνω όταν το x παίρνει μία μεγάλη αρνητική τιμή, το e^{-x} γίνεται πολύ μεγάλο, με αποτέλεσμα η σιγμοειδής συνάρτηση να μην μπρεί να εφαρμοστεί.

Σε τέτοιες περιπτώσεις, οι κλίσεις τείνουν να συρρικνώνονται εκθετικά καθώς διαδίδονται πίσω από τα στρώματα εξόδου στα στρώματα εισόδου. Αυτό το ζήτημα είναι πιο εμφανής στις αρχιτεκτονικές με πολλαπλά κρυφά επίπεδα, επειδή οι διαβαθμίσεις πρέπει να περάσουν από όλα τα κρυφά επίπεδα, με αποτέλεσμα οι τιμές τους να μειώνονται εκθετικά με κάθε επίπεδο.

Ως αποτέλεσμα, τα βάρη του δικτύου ενημερώνονται πολύ αργά ή καθόλου, οδηγώντας σε προβλήματα όπως:

1. Αργή σύγκλιση: Όταν οι κλίσεις είναι μικρές, οι ενημερώσεις βάρους είναι ελάχιστες, οδηγώντας σε αργή σύγκλιση.

2. Στασιμότητα: Σε σοβαρές περιπτώσεις, οι κλίσεις γίνονται τόσο μικρές που τα βάρη σταματούν ουσιαστικά να ενημερώνονται, με αποτέλεσμα το μοντέλο να μην βελτιώνεται.

3. Δυσκολία σύλληψης μεγάλων εξαρτήσεων: Στην περίπτωση των επαναλαμβανόμενων νευρωνικών δικτύων (RNN), το πρόβλημα της εξαφάνισης της κλίσης μπορεί να εμποδίσει την ικανότητα του δικτύου να διαχειρίζεται μεγάλες ακολουθίες διαδοχικών δεδομένων.

Για τον ξεπεραστεί το πρόβλημα του VGP, γίνεται χρήση διάφορων μεθόδων όπως:

1. Συναρτήσεις ενεργοποίησης που δεν επιφέρουν πρόβλημα αυτό, (Rectified Linear Units (ReLUs))
2. Προηγμένες αρχιτεκτονικές όπως LSTM
3. Τεχνικές όπως Batch normalization και Weight initialization

(LSTM)

Τα παραδοσιακά RNN τα οποία αναφέρθηκαν πιο πάνω υποφέρουν από το πρόβλημα VGP . Για να αντιμετωπιστεί αυτό, εισήχθησαν πιο εξελιγμένες αρχιτεκτονικές όπως τα δίκτυα μακράς βραχυπρόθεσμης μνήμης (LSTM) απο τους Hochreiter & Schmidhuber (1997) [67]. Τα LSTM είναι ένας τύπος αρχιτεκτονικής (RNN) που έχουν σχεδιαστεί για να καταγράφουν μακροπρόθεσμες εξαρτήσεις μεταξύ των δεδομένων, επιτρέποντας στις πληροφορίες του παρελθόντος να παραμένουν για πολλά χρονικά βήματα 'αποθηκευμένες' στη μνήμη. Αυτό επιτρέπει την αποτελεσματική εκπαίδευση στα DRNN (Deep RNN)και τη μοντελοποίηση πολύπλοκων χρονικών εξαρτήσεων σε διαδοχικά δεδομένα .

Τα LSTM είναι ιδιαίτερα αποτελεσματικά στο χειρισμό δεδομένων με εξαρτήσεις μεγάλης εμβέλειας, γεγονός που τα καθιστά κατάλληλα για εργασίες που περιλαμβάνουν διαδοχική λήψη αποφάσεων, πρόβλεψη μοτίβων σε διαδοχικά δεδομένα όπως χρονοσειρές, φυσικής γλώσσας-κείμενο και ομιλία .

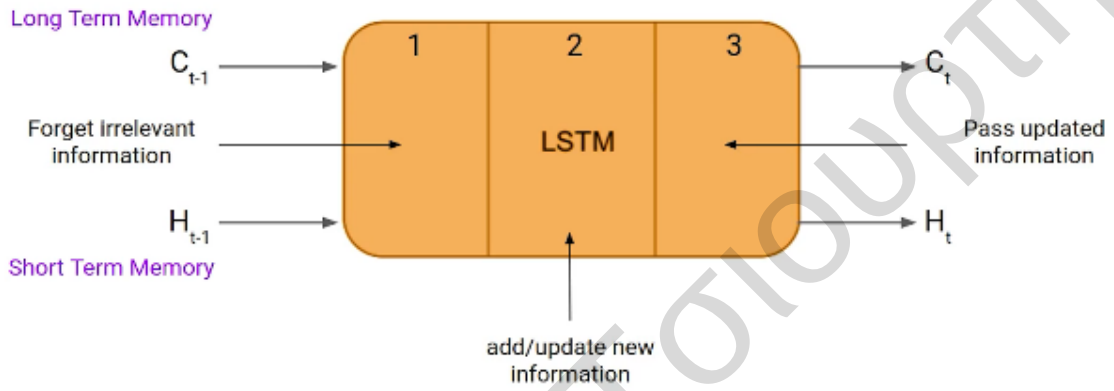
Στην ΕΜ η τρέχουσα κατάσταση του περιβάλλοντος εξαρτάται από προηγούμενες καταστάσεις και ενέργειες. Η χρήση των LSTM επιτρέπει στον πράκτορα την ανάκτηση πληροφοριών από προηγούμενα βήματα της ακολουθίας και μέσω αυτού μαθαίνει και αιτιολογεί τις μακροπρόθεσμες εξαρτήσεις στα δεδομένα.

Συνολικά, τα LSTM στην ενισχυτική μάθηση παρέχουν έναν τρόπο μοντελοποίησης και μάθησης από διαδοχικά και χρονικά δεδομένα, επιτρέποντας στους πράκτορες να λαμβάνουν πιο ενημερωμένες αποφάσεις με βάση το ιστορικό των αλληλεπιδράσεων με το περιβάλλον.

Αρχιτεκτονική LSTM

Όλα τα επαναλαμβανόμενα νευρωνικά δίκτυα έχουν τη μορφή μιας αλυσίδας επαναλαμβανόμενων μονάδων νευρωνικού δικτύου. Στα τυπικά RNN, αυτή η επαναλαμβανόμενη ενότητα θα έχει μια πολύ απλή δομή.

Η αρχιτεκτονική δικτύου LSTM αποτελείται από τρεις πύλες και μία μονάδα μνήμης ή μονάδα lstm, όπως φαίνεται στην παρακάτω Σχήμα2.19, και κάθε τμήμα εκτελεί μια μεμονωμένη λειτουργία.



Σχήμα 2.19: Αρχιτεκτονική LSTM RNN [66]

Η εσωτερική λειτουργία του δικτύου LSTM. Εδώ η κρυφή κατάσταση H(t) και H(t-1) είναι γνωστή ως βραχυπρόθεσμη μνήμη και η κατάσταση κυψέλης C(t) και C(t-1) είναι γνωστή ως μακροπρόθεσμη μνήμη. Αυτά τα τρία μέρη μιας μονάδας LSTM είναι γνωστά ως πύλες. Ελέγχουν τη ροή πληροφοριών μέσα και έξω από το κελί μνήμης ή το κελί lstm.

1. **Forget gate**

   Το πρώτο μέρος επιλέγει εάν οι πληροφορίες που προέρχονται από την προηγούμενη χρονική σήμανση πρέπει να απομνημονεύονται ή είναι άσχετες και μπορούν να ξεχαστούν.

2. **Input gate-Πύλη εισόδου**

   Στο δεύτερο μέρος, το κελί προσπαθεί να μάθει νέες πληροφορίες από την είσοδο σε αυτό το κελί.

3. **Output gate-Πύλη εξόδου**

   Στο τρίτο μέρος, το κελί μεταβιβάζει τις ενημερωμένες πληροφορίες από την τρέχουσα χρονική σήμανση στην επόμενη χρονική σήμανση. Αυτός ο ένας κύκλος του LSTM

Θεωρείται ένα βήμα μίας χρήσης.

Τα LSTM όπως έχουν δύο διαφορετικά είδη καταστάσεων:

1. Κρυφή κατάσταση)- H(t) και H(t-1) (eden States)
   Οι κρυφές καταστάσεις H(t-1) και H(t) καταγράφουν βραχυπρόθεσμες σχετικές πληροφορίες που έχει μάθει το LSTM σχετικά με την ακολουθία των εισόδων μέχρι την τρέχον χρονική στιγμή. Αφού γίνει η καταγραφή των σχετικών πληροφοριών από προηγούμενα χρονικά βήματα, ακολούθως μεταφέρονται προς τα εμπρός στο τρέχον χρονικό βήμα. Αυτές οι κρυφές καταστάσεις χρησιμοποιούνται κυρίως προβλέψεις.

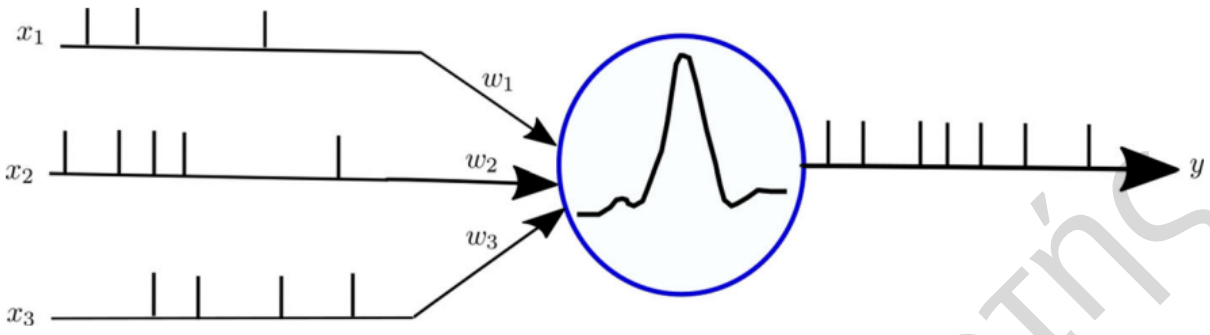
2. Κατάσταση κυψέλης- Ct και C(t-1) (Cell states)
   Οι κατάστασεις κυψέλης- C(t-1) και C(t) καταγράφουν μακροπρόθεσμες σχετικές πληροφορίες που έχει μάθει το LSTM σχετικά με την ακολουθία των εισόδων μέχρι την τρέχον χρονική στιγμή. Σε αντίθεση με τις κρυφές κατάστασεις, οι οποίες μεταφέρουν πληροφορίες που μπορεί να είναι σχετικές για την λειτουργία της πρόβλεψης, τα Cell states επικεντρώνονται στη διατήρηση και την επιλεκτική ενημέρωση των πληροφοριών μέσω των τριών πύλων του LSTM.

2.3.5.4 Spiking Neural Networks (SNN)

Τα νευρωνικά δίκτυα Spiking (SNN) είναι μια ειδική κατηγορία τεχνητών νευρωνικών δικτύων (ANN), που συνήθως αναφέρονται και ως η τρίτη γενιά ANN, όπου οι νευρωνικές μονάδες επικοινωνούν χρησιμοποιώντας διακριτές αλληλουχίες ακίδων, όπως φαίνεται στο Σχ.2.20. Ανάλογο με ένα βιολογικό νευρώνα, οι είσοδοι σε έναν αιχμηρό νευρώνα είναι διακριτές αιχμές, οι οποίες στη συνέχεια συνδυάζονται για να παράγουν μια ακίδα εξόδου εάν ξεπεραστεί ένα συγκεκριμένο όριο. Διαφορετικά, η έξοδος είναι μηδέν. Ως εκ τούτου, τα SNN περιέχουν επίσης χρονική δυναμική, η οποία τα καθιστά κατάλληλα για λειτουργία σε πραγματικό χρόνο, κάνοντας ενημερώσεις που βασίζονται αποκλειστικά σε γεγονότα και δεδομένα, σε αντίθεση με την επαναλαμβανόμενη και συχνά περιττή διαδικασία ενημέρωσης των βαρών στα ANN, η οποία είναι η υπολογιστική συμφόρηση για εργασίες που απαιτούν αλληλεπίδραση σε πραγματικό χρόνο με το περιβάλλον.

Τα SNN σε νευρομορφικό υλικό παρουσιάζουν ευνοϊκές ιδιότητες όπως χαμηλή κατανάλωση ενέργειας, γρήγορη εξαγωγή συμπερασμάτων και επεξεργασία πληροφοριών βάσει συμβάντων. Αυτό τα καθιστά ενδιαφέροντες υποψηφίους για την αποτελεσματική υλοποίηση των

βαθιών νευρωνικών δικτύων, της μεθόδου επιλογής για πολλές εργασίες μηχανικής μάθησης [68] [69] [70].



Σχήμα 2.20: Μοντέλο ενός αιχμηρού νευρώνα LIF [68].

2.3.6 Frameworks Βαθίας Μάθησης (Deep Learning Frameworks)

Η δημιουργία βαθιά ΤΝΔ τα οποία είναι εμπνευσμένα από την περιπλοκή αρχιτεκτονική του ανθρώπινου εγκεφάλου, έχουν αποδειχθεί ικανά να επιλύσουν πολύπλοκες εργασίες. Ωστόσο η ανάπτυξη τέτοιων δικτύων θεωρείται αρκετά πολύπλοκη και χρονοβόρα λόγω της πολυεπίπεδης δομής τους και των περίπλοκων μαθηματικών λειτουργιών τους. Για την αντιμετώπιση αυτού του προβλήματος αναπτύχθηκαν τα Deep learning framework τα οποία μετριάζουν αυτήν την πολυπλοκότητα, αφαιρώντας την πολυπλοκότητα χαμηλού επιπέδου της υλοποίησης νευρωνικών δικτύων και μαθηματικών λειτουργιών, επιτρέποντας στους επαγγελματίες να επικεντρωθούν στο σχεδιασμό καινοτόμων αρχιτεκτονικών και στον πειραματισμό με διάφορες διαμορφώσεις. Παράλληλα παρέχουν ένα υψηλότερο επίπεδο αφαίρεσης μέσω έξυπνων API, επιτρέποντας στους χρήστες μια ολοκληρωμένη σειρά εργαλείων, βιβλιοθηκών και βελτιστοποιημένων λειτουργιών που διευκολύνουν την κατασκευή και τη χρήση αυτών των νευρωνικών αρχιτεκτονικών με αξιοσημείωτη ευκολία

Εν τάχει ένα πλαίσιο βαθιάς μάθησης στην ουσία του, λειτουργεί ως γέφυρα μεταξύ του εννοιολογικού σχεδιασμού των νευρωνικών δικτύων και της πρακτικής εφαρμογής τους, παρέχοντας ένα εξελιγμένο και δομημένο περιβάλλον για τη δημιουργία, την εκπαίδευση και την ανάπτυξη περίπλοκων βαθιών νευρωνικών δικτύων.

Επίσης μία από τις κυριότερες λειτουργίες που παρέχουν(πχ. Visualization and Monitoring ,Transfer Learning, Deployment, Community and Resources κτλ.) τα Deep learning frame-

work η οποία και πρέπει να σημειωθεί είναι ότι έχουν σχεδιαστεί έτσι ώστε να ενσωματώσουν και να κάνουν χρήση μονάδων Επεξεργασίας Γραφικών (GPU) και των μονάδων επεξεργασίας Tensrs (TPU). Οι χρήση αυτών των επιταχυντών έχουν επιφέρει μείωση του χρόνου εκτέλεσης των λειτουργιών τους(πχ εκπαίδεση) σε μεγάλο βαθμό και ειδικά για μοντέλα μεγάλης κλίμακας που απαιτούν εκτεταμένη υπολογιστική ισχύ.

Στη συνέχεια θα γίνει μία μικρή παρουσίαση στα κυριότερα Deep learning framework (TensorFlow, PyTorch, Keras (το οποίο αποτελεί πλέον μέρος του TensorFlow), spinningup. τα οποία έχουν παίξει καθοριστικό ρόλο στην ταχεία προόδο της έρευνας και των εφαρμογών βαθιάς μάθησης σε ένα ευρύ φάσμα πεδίων, όπως η υπολογιστή όραση, επεξεργασία φυσικής γλώσσας, αναγνώριση ομιλίας και πολλά άλλα.

2.3.6.1 TensorFlow

Το TensorFlow είναι ένα Deep learning framework ανοιχτού κώδικα που αναπτύχθηκε από την Google Brain Team [71]. Χρησιμοποιείται ευρέως σε διάφορους τομείς της μηχανικής μάθησης, συμπεριλαμβανομένης της εποπτευόμενης μάθησης, της μάθησης χωρίς επίβλεψη και της ΕΜ. Θεωρείτε ένα επαναστατικό μέσω λόγο του τρόπου με τον οποίο αναπτύσσονταικαι εκπαιδεύονται τα μοντέλα μηχανικής μάθησης.

Το TensorFlow περιλαμβάνει προκατασκευασμένα μοντέλα και επίπεδα, μέσω των οποίων διευκολύνεται η ανάπτυξη εφαρμογών για εργασίες όπως η αναγνώριση εικόνας και ομιλίας, η επεξεργασία φυσικής γλώσσας και τα συστήματα συστάσεων.

Συγκεκριμένα όσον αφορά την ΕΜ, η εκπαίδευση και ενημέρωση των παραμέτρων των βαθιών ΤΝΔ γίνεται βάση την ανατροφοδότηση από το περιβάλλον μεσώ της χρήσης αλγορίθμων βελτιστοποίησης όπως:

- Q-learning
- SARSA
- Deep Q-Networks (DQN
- Actor-Critic Algorithms

Τέλος ένας από τους κύριους λόγους χρήσης του είναι η αποτελεσματική κατανομή υπολογισμών σε διάφορους επιταχυντές υλικού, όπως GPU και TPU.

2.3.6.2 Keras

Όπως αναφέρθηκε πιο πάνω η δύναμη του TensorFlow έγκειται στην προσαρμοστικότητά του σε πολλούς τομείς της ΜΜ. Ένα από τα ΑΠΙ υψηλού επιπέδου που παρέχει είναι το Keras [72]. Το Keras είναι ένα API υψηλού επιπέδου νευρωνικών δικτύων ανοιχτού κώδικα γραμμένο σε Πψτηον το οποίο απλοποιεί τη δημιουργία νευρωνικών δικτύων και άλλων μοντέλων ΜΜ. Ταυτόχρονα μέσω του keras παρέχεται μία φιλική προς το χρήστη διεπαφή για την κατασκευή μοντέλων για γρήγορο πειραματισμό με αρχιτεκτονικές (πχ. CNN,RNNM MLPs), επίπεδα και λειτουργίες απώλειας.

2.3.6.3 PyTorch

Το PyTorch αναπτύχθηκε το 2017 από την ερευνητική ομάδα AI του Facebook και είναι ένα Deep learning framework ανοιχτού κώδικα [73]. Έγινε τόσο δημοφιλές λόγω της ευκολίας χρήσης του, της ευελιξίας και της δυναμικής του για τη δημιουργία μοντέλων βαθιάς μάθησης σε σε διάφορους τομείς της μηχανικής μάθησης, συμπεριλαμβανομένης της εποπτευόμενης μάθησης, της μάθησης χωρίς επίβλεψη και της ΕΜ καθώς και για την εφαρμογή του στην αναγνώριση εικόνων και επεξεργασία γλώσσας.

Στην ΕΜ, το PyTorch χρησιμοποιείται συχνά για τον ορισμό και την εκπαίδευση νευρωνικών δικτύων για την προσέγγιση της βέλτιστης συνάρτησης πολιτικής ή αξίας ενός πράκτορα. Το PyTorch επιτρέπει την εύκολη εφαρμογή πολύπλοκων αλγορίθμων όπως η βαθιά ενισχυτική εκμάθηση, όπου η αρχιτεκτονική δικτύου και ο αλγόριθμος εκμάθησης ενημερώνονται ταυτόχρονα.

Επιπλέον, το PyTorch ενσωματώνεται με την βιβλιοθήκη ενισχυτικής μάθησης OpenAI Gym, καθιστώντας εύκολη την εφαρμογή και τη δοκιμή διαφόρων αλγορίθμων μάθησης ενίσχυσης σε διαφορετικά περιβάλλοντα.

Τέλος ένας από τους κύριους λόγους χρήσης του όπως και το Tensorflow είναι η αποτελεσματική κατανομή υπολογισμών σε διάφορους επιταχυντές υλικού, όπως GPU και TPU.

2.3.6.4 Spinning Up in Deep RL

Σε αντίθεση με το Tensorflow και Pytorch τα οποία εφαρμόζονται σε διάφορους τομείς της μηχανικής μάθησης, το "Spinning Up in Deep Reinforcement Learning" είναι ένα εκπαιδευτικό λογισμικό που αναπτύχθηκε από την OpenAI και έχει σχεδιαστεί για να βοηθήσει τα

άτομα να μάθουν και να κατανοήσουν τις βασικές αρχές της βαθιάς ΕΜ [74].

Το Spinning Up παρέχει:

- Εκπαιδευτικό Υλικό: Το Spinning Up παρέχει λεπτομερώς υλικό το οποίο επεξηγεί βασικές έννοιες στη βαθιά ενισχυτική μάθηση, συμπεριλαμβανομένων αλγορίθμων, θεωρίας και πρακτικών λεπτομερειών εφαρμογής καθώς και οδηγό λειτουργίας του ίδιου του spinningup.
- Παραδείγματα κώδικα: Για την κατανόηση των εννοιών και αλγορίθμων παρέχεται κώδικας διαφόρων αλγορίθμων επιτρέποντας στους χρήστες να πειραματιστούν και να μάθουν από τα παραδείγματα.

Όλοι οι πίο κάτω αλγορίθμων που παρέχονται από το spinningup και διαθέτουν δύο εκδοχές:

1. Vanilla Policy Gradient (VPG)
2. Trust Region Policy Optimization (TRPO)
3. Proximal Policy Optimization (PPO)
4. Deep Deterministic Policy Gradient (DDPG)
5. Twin Delayed DDPG (TD3)
6. Soft Actor-Critic (SAC)

Μία που υλοποιήθηκε βάση του PyTorch (εκτός από τον TRPO) και μία που υλοποιήθηκε στο Tensorflow. Όλοι οι πιο πάνω αλγόριθμοι εφαρμόζονται στα περιβάλλοντα της OpenAI Gym με στόχο την καλύτερη κατανόηση τους από τους χρήστες μέσω των αποτελεσμάτων και της οπτικοποίησης τους καθώς τρέχουν.

2.3.7 Μεταφορά Μάθησης και Προ-εκπαιδεθμαένα Μοντέλα (Transfer Learning and Pre-trained Models)

2.3.7.1 Pre-trained Models

Με τον όρο προεκπαιδευμένο μοντέλο (Pre-trained models) αναφερόμαστε σε ένα μοντέλο μηχανική μάθησης που έχει εκπαιδευτεί σε ένα μεγάλο όγκο δεδομένων για μία συγκεκριμένη εργασία (πχ επεξεργασίας φυσικής γλώσσας (NLP), υπολογιστική όραση) σε ένα συγκεκριμένο τομέα. Ο χρόνος εκπαίδευσης των μοντέλων για ένα μεγάλο όγκο δεδομένων μπορεί να διαρκέσει από μερικές μέρες μέχρι εβδομάδες. Έχοντας στην κατοχή μας αυτά τα ήδη εκπαιδευμένα μοντέλα μπορούμε να τα εφαρμοστούν και σε άλλος τομείς και εργασίες μέσω κάποιας προσαρμογής τους. Αυτό περιλαμβάνει περαιτέρω εκπαίδευση του μοντέλου σε ένα μικρότερο σύνολο δεδομένων που σχετίζεται με τη συγκεκριμένη εργασία, επιτρέποντας στο μοντέλο να εξειδικεύεται σε αυτήν την εργασία. Αυτή η επιπλέον ρύθμιση είναι ιδιαίτερα χρήσιμη όταν τα διαθέσιμα δεδομένα για συγκεκριμένες εργασίες είναι περιορισμένα.

Συνεπώς η χρήση των προεκπαιδευμένων μοντέλων σε νέους τομείς και εργασίες μέσω περαιτέρω εκπαίδευσης και ρύθμισης τους επιλύει πολλά καίρια προβλήματα τα οποία ερχόμασταν αντιμέτωποι μέχρι στιγμής όπως:

1. Αποδοτικότητα και ταχύτητα: Η χρήση ενός προεκπαιδευμένου μοντέλου μας εξοικονομεί σημαντικό χρόνο και υπολογιστικούς πόρους αφού η διαδικασία εκπαίδευσης απο την αρχή ενός μοντέλου είναι χρονοβόρα και με υψηλό κόστος.

2. Μεταφερόμενη μάθηση- Τρανσφερ λεαρνινγ: Αυτό επιτυγχάνεται κάνοντας χρήση ενός προοεκπαιδευμένου μοντέλου το οποίο εκπαιδεύτηκε σε δεδομένα παρόμοιας εργασίας με αυτή που θέλουμε να εκτελέσουμε(πχ Object Detection,Image Classification,Natural Language Processing (NLP))

3. Περιορισμένα δεδομένα- Fine tuning: Όταν τα δεδομένα μας για ένα συγκεκριμένο πρόβλημα είναι περιορισμένα ακολουθούμε την διαδικασία Φινε τυνινγ. Συγκεκριμένα παίρνουμε το ήδη προεκπαιδευμένο μοντέλο και το εκπαιδεύουμε στο μικρό σύνολο δεδομένων το οποίο κατέχουμε για τη συγκεκριμένη εργασία. Με αυτό τον τρόπο 'προσαρμόζουμε' το μοντέλο μας στα συγκεκριμένα δεδομένα σας.

4. Γενίκευση: Τα προεκπαιδευμένα μοντέλα μέσω της της εκπαίδευσης τους απο δεδομένα διαφορετικών τύπων δεδομένων αναπτύσουν την ικανότητα να γενικεύουν καλά

νέα δεδομένα. Αυτό είναι ιδιαίτερα πολύτιμο σε εργασίες NLP, όπου τα γλωσσικά μοτίβα μπορεί να διαφέρουν πολύ.

Άρα τα προεκπαιδευμένα μοντέλα θεωρούνται ως ισχυρά εργαλεία για την ταχεία ανάπτυξη εφαρμογών τεχνητής νοημοσύνης και μηχανικής εκμάθησης, επιτρέποντάς μας να επιτύχετε εντυπωσιακά αποτελέσματα με μειωμένη προσπάθεια και πόρους. Τα Deep learning framework που αναφέρθηκαν προηγουμένως όπως το TensorFlow, το PyTorch διευκολύνουν τη διαδικασία χρήσης και εκπαίδευσης προεκπαιδευμένων μοντέλων.

2.3.7.2 Fine-tuning

Για να επιλύσουμε ένα πρόβλημα στη ΒΜΜ επιβάλετέ η εκπαίδευση ενός ΝΔ στο σύνολο των δεδομένων που παρέχονται για αυτό το πρόβλημα. Η διαδικασία εκπαίδευσης για επιτύχει μία καλή γενίκευση στα ΤΝΔ τα οποία έχουν τεράστιο αριθμό παραμέτρων απαιτεί μεγάλος όγκος δεδομένων και συνεπώς καθίσταται αρκετά χρονοβόρα. Ιδικά στις περιπτώσεις όπου πρέπει να εκπαιδευτεί ένα ΣΝΝ αυτό ο χρόνος αυξάνεται δραματικά.

Σε περίπτωση εκπαίδευσης ενός ΣΝΝ σε ένα μικρό σύνολο δεδομένων (ένα που είναι μικρότερο από τον αριθμό των παραμέτρων) επηρεάζεται η ικανότητα γενίκευσης του, και συχνά οδηγεί σε υπερβολική προσαρμογή [75].

Ως εκ τούτου, στην πράξη δεν συνηθίζεται η εκπαίδευση ενός ΤΝΔ από την αρχήα αλλά γίνεται χρήση ήδη προ εκπαιδευόμενων ΝΔ (πχ. ImageNet εκπαιδεύτικε σε ένα σύνολο 1,2 εκ. Εικόνων με ετικέτες, BERT). Αυτα τα προεκπαιδευόμενα ΝΔ τυχαίνουν κάποιας επιπλέον εκπαίδευσης με ένα μικρότερο σύνολο δεδομένων σχετικό με το πεδίο που εμπίπτει το πρόβλημα που θα επιλύσουμε.

Η προϋπόθεση για την πιο πάνω διαδικασία, είναι ότι το μικρό σύνολο δεδομένων μας δεν διαφέρει δραματικά σε σχέση με το αρχικό σύνολο δεδομένων (π.χ. ImageNet). Δηλαδή το προεκπαιδευμένο μοντέλο θα έχει ήδη μάθει τα κύρια χαρακτηριστικά(πχ ακμές, καμπύλες κτλ) που σχετίζονται με το δικό μας πρόβλημα ταξινόμησης [75] και με την επιπλέον εκπαίδευση προσθέτουμε κάποιες λεπτομέρειες που επιθυμούμε. Ο μόνος λόγος για να αποφευχθεί η πιο πάνω διαδικασία είναι να μην μπορούν να βρεθούν προεκπαιδευμένα δίκτυα στον τομέα που θέλουμε, τότε θα πρέπει να εξετάσουμε το ενδεχόμενο εκπαίδευσης του δικτύου από την αρχή.

Πριν την εφαρμογή του ΤΝΔ σε ένα νέο πρόβλημα απαιτούνται και κάποιες άλλες τροποποιήσεις όπως για παράδειγμα:

1. Αλλαγή του επιπέδου εξόδου του προεκπαιδευμένου δικτύου (softmax) με ένα επίπεδο που πληρεί τις προδιαγραφές του νέου προβλήματος. Για παράδειγμα το ImageNet στο επίπεδο εξόδου έχει 1000 νευρώνες εξόδου και συνεπώς ταξινομεί τις εισόδου του σε 1000 κατηγορίες. Επομένως εάν η δική μας ταξινόμηση εμπίπτει σε λιγότερες κατηγορίες θα πρέπει να αλλάξει.

2. Χρήση μικρότερου ρυθμού εκμάθησης(εμπειρικά συνήθος 10 φορές μικρότερο από την τιμή που είχε στην διαδικασία εκπαίδευσης) για την εκπαίδευση του δικτύου αφού δεν θέλουμε να αλλάξουν τα βάρη σε μεγάλο βαθμό αφού είναι ήδη εκπαιδευμένο το δίκτυο. Χρειάζεται να αλλάξουν ελάχιστο οι τιμές των βαρών έτσι ώστε να αντιλαμβάνονται τις επιπλέων λεπτομέρειες που χρειάζεται.

3. Κρατάμε σταθερά τα βάρη στα αρχικά επίπεδα του προεκπαιδευμένου δικτύου. Αυτό γίνεται λόγο του ότι τα πρώτα επίπεδα είναι υπευφθηνα για την αναγνώριση των κύριων χαρακτηριστικών και όχι λεπτομέρειων, επομένος δεν χρειάζεται να μεταβληθούν.

2.3.7.3 Tuning

Ο ορισμός Tuning (συντονισμός) στην ΕΜ αναφέρεται στην διαδικασία ρύθμισης των υπερπαραμέτρων(Learning Rate, Discount Factor (Gamma, Exploration Rate (Epsilon) κτλ) ενός αλγορίθμου με απώτερο στόχο τη βελτιστοποίηση της απόδοσης του. Με τον όρο υπερπαραμέτροι αναφερόμαστε σε μεταβλητές οι οποίες χρησιμοποιούνται μέσα στον αλγόριθμο και έχουν σημαντικό αντίκτυπο στο μέτρο της απόδοση του αλγορίθμου -πώς μαθαίνει ο αλγόριθμος .

Ο στόχος του συντονισμού είναι να βρεθεί το βέλτιστο σύνολο όλων των τιμών των υπερπαραμέτρεων που θα επιτρέψει στον αλγόριθμο να μάθει όσο το δυνατών πιο αποτελεσματικά την εργασία που εκτελεί.

Η διαδικασία συντονισμού περιλαμβάνει τα ακόλουθα βήματα 1. τυχαία επιλογή ενός συνόλου υπερπαραμέτρων, 2. εκτέλεση του αλγορίθμου με αυτές τις υπερπαραμέτρους 3. αξιολόγηση της απόδοσής του

Η πιο πάνω διαδικασία γίνεται επαναληπτικά εως ότου βρεθεί το βέλτιστο σύνολο υπερπαραμέτρων.

Υπάρχουν διάφορες μέθοδοι για τον συντονισμό υπερπαραμέτρων στην ενισχυτική μάθηση:

1. Grid search – Δοκιμή όλων των πιθανών συνδυασμών των υπερπαραμέτρων που μπορεί να λάβει κάθε υπερπαράμετρως(predefined range-καθορίζετε ένα εύρος τιμών που θέλουμε να εξερευνήσουμε για κάθε υπερπαράμετρων ξεχωριστά)
2. Random search - Δοκιμή τυχαίων πιθανόν συνδυασμών των υπερπαραμέτρων που μπορεί να λάβει κάθε υπερπαράμετρως(όχι δοκιμή όλων των πιθανών συνδυασμών που γίνεται στο Grid Search)
3. Bayes optimazation - χρησιμοποιεί ένα πιθανό μοντέλο για να καθοδηγήσει την αναζήτηση βέλτιστων υπερπαραμέτρων.

.

Συνοπτικά, ο συντονισμός είναι μία απαραίτητη διαδικασία στην ΕΜ,καθώς βοηθά στην διασφάλιση του ότι ο αλγόριθμος θα μάθει όσο το δυνατών πιο αποτελεσματικά την εργασία που εκτελείται.

2.3.8 Generative Models

Μέχρι στιγμής έχουν παρουσιαστεί μοντέλα ΤΝΔ τα οποία επικεντρώνονται στην πρόβλεψη και ταξινόμηση δεδομένων. Μία νέα κατηγορία αλγορίθμων ΜΜ τα generative models διαθέτουν την ικανότητα να μαθαίνουν τα δεδομένων του συνόλου εκπαίδευσης, και ακολούθως δημιουργούν νέα καινούργια δεδομένων που μοιάζουν πολύ με τα αρχικά δεδομένα με ορισμένες παραλλαγές. Ο τρόπος εκπαίδευσης τους επιτυγχάνεται μέσω μη επιβλεπόμενης μάθησης. Η εκμάθηση των δεδομένων εκπαίδευσης στοχεύει στην αποτύπωση-κατανόηση της ουσίας-μοτίβο των δεδομένων και κάνοντας χρήση αυτής της γνώσης και στη συνέχεια να παράγουν νέα δεδομένα που είναι ταυτόχρονα ρεαλιστικά και καινοτόμα.

Η εκμάθηση της ακριβής δομής των δεδομένων δεν είναι πάντα εφικτή, αλλά είναι όσο το δυνατόν πιο προσεγγιστική σε αυτά . Για αυτό τον λόγο, μπορούμε γίνεται χρήση των νευρωνικών δικτύων τα οποία παρέχουν μία συνάρτηση που μπορεί να προσεγγίσει την κατανομή του μοντέλου στην πραγματική κατανομή.

Βάση όλο των πιο πάνω, στόχος των generative models είναι να αναπτύξουν την ικανότητα της δημιουργικότητας στα μοντέλα και εφαρμογής τους σε ένα ευρύ φάσμα πεδίων.

Τα μοντέλα αυτά είναι ικανά μέχρι στιγμής να παράγουν Εικόνες, Κείμενο, Μουσική, Βίντεο μαθαίνοντας από τεράστιους όγκους υπαρχόντων δεδομένων. Οι δύο πιο συχνά χρησιμοποιούμενες και αποτελεσματικές τεχνικές και αρχιτεκτονικές, είναι οι Variational Autoencoders (VAE) και τα Generative Adversarial Networks (GAN). Κάθε μία από αυτές τις δύο θα αναλυθεί παρουσιάζοντας τις δυνατότητες αλλά και τις εφαρμογές τους.

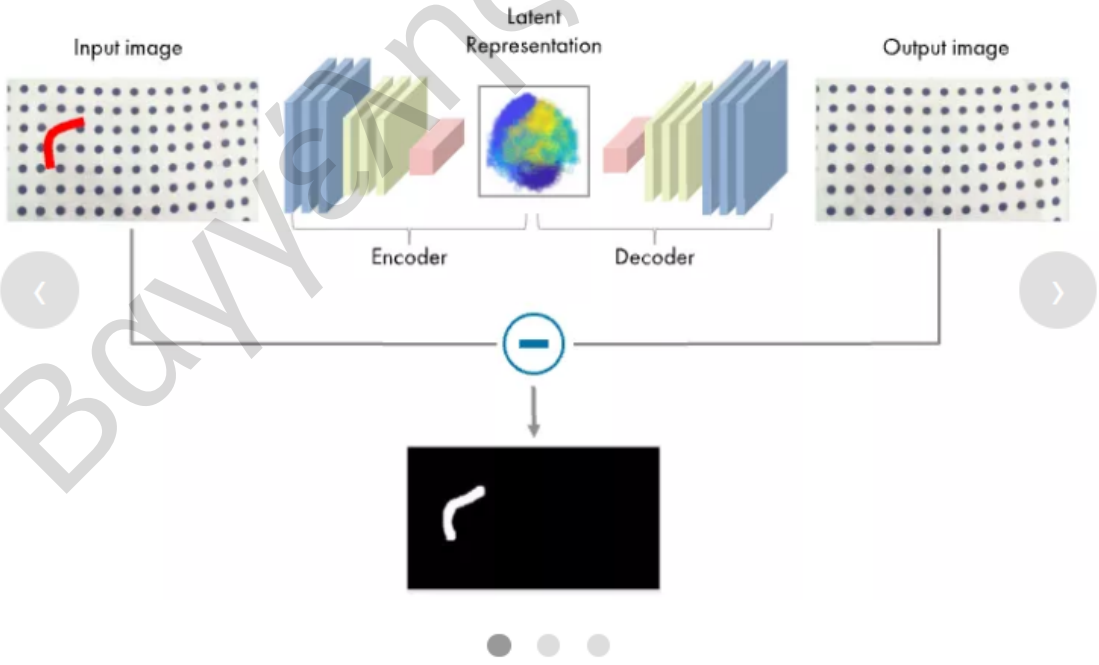
2.3.8.1 Autoencoders

Ο Autoencoders (αυτοκωδικοποιητές) με συνάρτηση

$$F(x; \theta, \phi) = D(E(x; \theta); \phi) : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

είναι ένα είδος αρχιτεκτονικής ΤΝΔ που εκπαιδεύεται μέσω μη επιβλεπόμενης μάθηση. Αποτελείται από δύο μέρη:

1. έναν κωδικοποιητή
2. έναν αποκωδικοποιητή



Σχήμα 2.21: Αρχιτεκτονική αυτόματου Κωδικοποιητή [76]

Κωδικοποιητής $D(E(x;\partial))$

Ο κωδικοποιητής λαμβάνει δεδομένα εισόδου από τον αρχικό χώρο δεδομένων $\mathbb{R}^n$και τα μετατρέπει-αντιστοιχίζει-συμπίεζει σε ένα χώρο χαμηλότερης διάστασης $\mathbb{R}^z$, που ονομάζεται λανθάνον χώρος(λατεντ σπασε). Αυτή η αντιστοίχιση γίνεται μέσω της συνάρτησης κωδικοποίησης.

$$E(x;\partial)) : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

Εδώ, το ϑ αντιπροσωπεύει τις παραμέτρους του κωδικοποιητή και το z είναι η διάσταση του λανθάνοντος χώρου.

Αποκωδικοποιητή $F(x;\partial,\phi)$

Ο αποκωδικοποιητής επιχειρεί να ανακατασκευάσει-αποσυμπιέζει τα αρχικά δεδομένα εισόδου πίσω στον αρχικό χώρο δεδομένων από αυτή το latent space προσπαθώντας ταυτόχρονα να ελαχιστοποιήσει το σφάλμα ανακατασκευής.

$$D(Z;\phi)) : \mathbb{R}^z \rightarrow \mathbb{R}^n$$

Εδω, το Φ αντιπροσωπεύει τις παραμέτρους του αποκωδικοποιητή.

Εκπαίδευση

Κατά τη διάρκεια της εκπαίδευσης, ο αυτόματος κωδικοποιητής προσπαθεί να ελαχιστοποιήσει την απόκλιση μεταξύ των δεδομένων εισόδου $F(x;\Theta,\Phi)$ και της ανακατασκευασμένης εξόδου $F(x;\Theta,\Phi)$. Αυτό επιτυγχάνεται χρησιμοποιώντας μια συνάρτηση απώλειας όπως το μέσο τετράγωνο σφάλμα (MSE).

$\mathcal{L}(\partial,\phi) = \frac{1}{n}\sum_{i=1}^{n}(x_i - F(x_i;\partial,\phi))^2$

Εδώ,n είναι ο αριθμός των δειγμάτων δεδομένων.

Προσαρμόζοντας επαναληπτικά τα βάρη μαθαίνει τις βέλτιστες τιμές των Θ (κωδικοποιητή) και Φ (αποκωδικοποιητή).Η επαναληπτική προσαρμογή των βαρών γίνεται κυρίως μέσω χρησης της backpropagation και μιας επαναληπτικής διαδικασίας βελτιστοποίησης (πχ. gradient descent). Το μοντέλο μαθαίνει να δημιουργεί μια ουσιαστική λανθάνουσα αναπαράσταση που καταγράφει τα βασικά χαρακτηριστικά των δεδομένων. Με άλλα λόγια, για ένα δεδομένο σύνολο πιθανών κωδικοποιητών και αποκωδικοποιητών, αναζητούμε το ζεύγος που διατηρεί το μέγιστο των πληροφοριών κατά την κωδικοποίηση και, επομένως, έχει το ελάχιστο σφάλμα ανακατασκευής κατά την αποκωδικοποίηση

Αυτά τα μοντέλα βρίσκουν εφαρμογές σε διάφορους τομείς όπως:

1. **Στη μείωση των διαστάσεων.**

   Στη μηχανική μάθηση, η μείωση διαστάσεων είναι η διαδικασία μείωσης του αριθμού των χαρακτηριστικών που περιγράφουν ορισμένα δεδομένα. Στην ουσία γίνεται αναπαράσταση των δεδομένων εισόδου σε ένα χώρο μικρότερων διαστάσεων(πχ λατεντ σπαςε). Αυτή η μείωση είναι χρήσιμη σε πολλές περιπτώσεις που απαιτούν δεδομένα χαμηλών διαστάσεων (πχ οπτικοποίηση δεδομένων, δεδομένα αποθήκευση, βαρύς υπολογισμός κτλ.).

2. **Η ανίχνευση ανωμαλιών.**

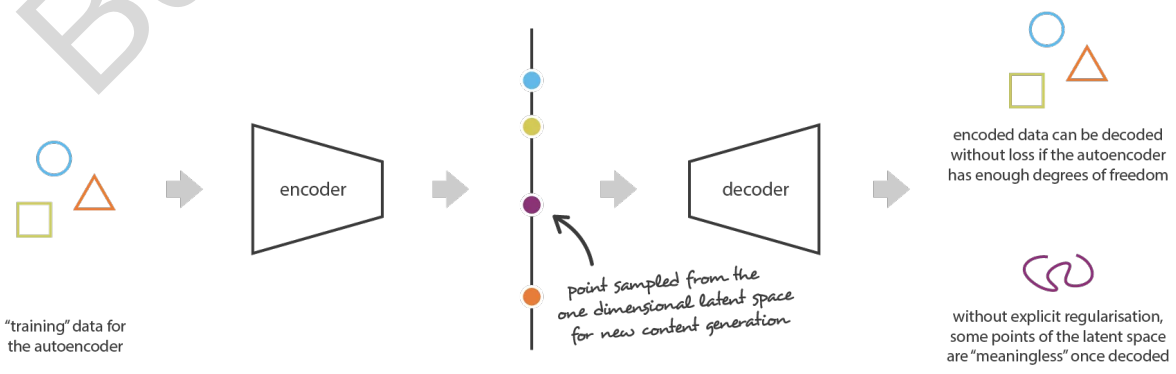
   Ανίχνευση μοτίβων (παττερνς) ακόμη και αν υπάρχουν αλλοιώσεις σε αυτά

3. **Η αφαίρεση θορύβου εικόνας.**

   Αφαίρεση του θορύβου στις εικόνες αυξάνοντας την ποιότητα τους

2.3.8.2 Variational Autoencoders (VAE)

Όπως αναφέρθηκε πιο πάνω οι αυτόματοι κωδικοποιητές προσπαθούν να φτάσουν στο ελάχιστο σφάλμα ανακατασκευής κατά την αποκωδικοποίηση. Επομένως αυτή η κωδικοποίηση και αποκωδικοποίηση χωρίς απώλεια πληροφοριών (παρά τη χαμηλή διάσταση του λανθάνοντος χώρου) οδηγεί σε μια σοβαρή υπερπροσαρμογή που υποδηλώνει ότι ορισμένα σημεία του λανθάνοντος χώρου θα δώσουν περιεχόμενο χωρίς νόημα μόλις αποκωδικοποιηθεί Σχήμα2.22. Έτσι, είναι αρκετά δύσκολο να διασφαλιστεί, εκ των προτέρων, ότι ο κωδικοποιητής θα οργανώσει τον λανθάνοντα χώρο με έξυπνο και συμβατό τρόπο με τη διαδικασία παραγωγής.



Σχήμα 2.22: Variational Autoencoders (VAE) [14]

οι μεταβλητοί αυτοκωδικοποιητές (VAEs) είναι αυτοκωδικοποιητές που αντιμετωπίζουν το πρόβλημα της ανωμαλίας του λανθάνοντος χώρου κάνοντας τον κωδικοποιητή να επιστρέφει μια κατανομή στον λανθάνοντα χώρο αντί για ένα μόνο σημείο και προσθέτοντας στη συνάρτηση απώλειας έναν όρο τακτοποίησης σε αυτήν την επιστρεφόμενη κατανομή προκειμένου να διασφαλιστεί καλύτερη οργάνωση του λανθάνοντος χώρου

Εδώ έρχονται οι μεταβλητοί αυτοκωδικοποιητές (VAEs) οι οποίοι θεωρούνται μία επέκταση των κλασικών αυτόματων κωδικοποιητών με την διαφορά ότι εφαρμόζουν προσέγγιση πιθανοτικής κωδικοποίησης(probabilistic encoding approach) στο λανθάνοντα χώρο και προσθέτουν στη συνάρτηση απώλειας έναν όρο τακτοποίησης (regularisation term). Η έννοια πιθανοτικής κωδικοποίησης χρησιμοποιείται στους Αυτοκωδικοποιητές Μεταβλητών (VAE), για να αναπαραστήσουν δεδομένα όχι ως ένα σταθερό σημείο σε έναν λανθάνοντα χώρο αλλά ως κατανομή πιθανότητας σε αυτόν τον χώρο.

Μέσω αυτής της κατανομής στο λανθάνοντα χώρο και του όρου τακτοποίησης διασφαλίζεται η καλύτερη οργάνωση του λανθάνοντος χώρου αλλά και η δημιουργία νέων δειγμάτων δεδομένων.

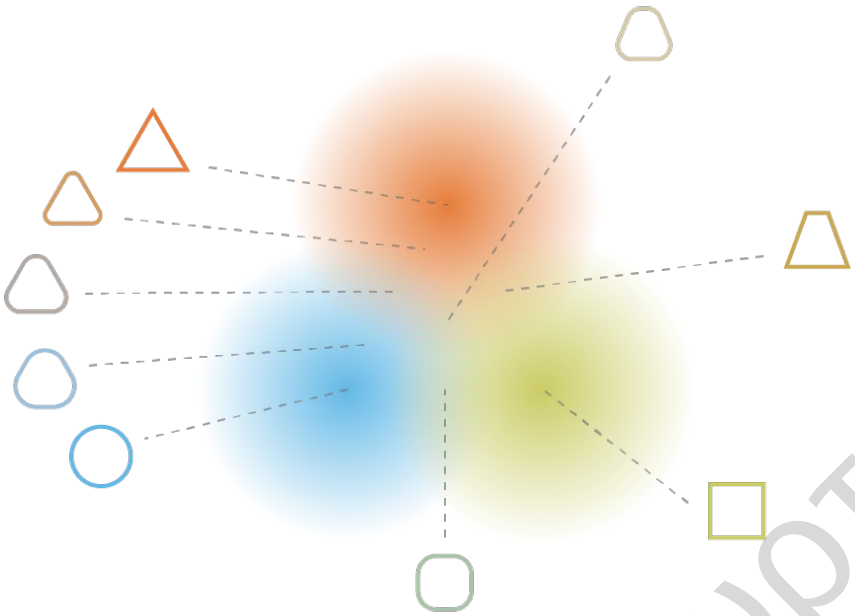
Ο νέος αυτός λανθάνον χώρος μπορεί να εκφραστεί μέσω δύο βασικών ιδιοτήτων:

1. Συνέχειας (δύο στενά σημεία στον λανθάνοντα χώρο δεν πρέπει να δίνουν δύο εντελώς διαφορετικά περιεχόμενα μετά την αποκωδικοποίηση)

2. Πληρότητας (για μια επιλεγμένη κατανομή , ένα σημείο δειγματοληψίας από τον λανθάνοντα χώρο θα πρέπει να δίνει «νόημα» περιεχόμενο μόλις αποκωδικοποιηθεί).

Για να ισχίου τα πιο πάνω και να μην λειτουργεί ο VAEs ως ένα κλασικός AEs πρέπει να ρυθμίσουμε τόσο τον πίνακα διακύμανσης όσο και τον μέσο όρο τις κάθε κατανομής που επιστρέφεται από τον κωδικοποιητή καθώς και την απόκλιση KL μεταξύ της επιστρεφόμενης κατανομής και μίας τυπικής κατανομής Gaussian.

Επομένως ένας μεταβλητός αυτόματος κωδικοποιητής μπορεί να οριστεί ως ένας αυτόματος κωδικοποιητής του οποίου η εκπαίδευση είναι ρυθμισμένη για να αποφευχθεί η υπερβολική προσαρμογή και να διασφαλιστεί ότι ο λανθάνοντας χώρος έχει καλές ιδιότητες που επιτρέπουν τη διαδικασία παραγωγής.



Σχήμα 2.23: Απεικόνιση του Χώρου Λαμβάνοντας Υπόψη την Συνέχεια και Πληρότητα [14]

2.4 Policy Gradients

2.4.1 Εισαγωγή Policy Gradients

Στην ΕΜ η τεχνική Policy Gradient (PG) είναι μία μέθοδος βελτιστοποίησης της πολιτικής-στρατηγικής ενός πράκτορα. Μέσω αυτής της πολιτικής ο πράκτορας θα είναι ικανός να επιλέξει τις βέλτιστες ενέργειες εντός του περιβάλλοντος που αλληλεπιδρά. Αυτές οι ενέργειες δεν θα επιφέρουν μόνο την επίτευξη ενός συγκεκριμένου στόχου αλλά θα επιτύχουν και μεγιστοποίηση των σωρευτικών ανταμοιβών με την πάροδο του χρόνου. Η καινοτομία που προσφέρει η PG, είναι ότι αντί να μαθαίνουν μια συνάρτηση αξίας η οποία εκτιμάται από τις αναμενόμενες μελλοντικές ανταμοιβές και έπειτα να βελτιώνει την πολιτική του όπως οι άλλοι αλγόριθμοι, επικεντρώνονται στην άμεση βελτιστοποίηση της ίδιας της πολιτικής.

Ο στόχος του PG είναι να βρει τις τιμές των παραμέτρων της πολιτικής οι οποίες θα βελτιώσουν την απόδοση της και συνεπώς θα μεγιστοποιήσουν τη συνολική ανταμοιβή που θα επιτύχει ο πράκτορας όταν ακολουθεί αυτήν την πολιτική. Η ενημέρωση των παραμέτρων της πολιτικής επιτυγχάνεται μέσα από την επαναληπτική χρήση της μεθόδου κλίσης ανάβασης (gradient ascent) τρόπος που βελτιώνει την απόδοση της πολιτικής. Στην συνέχεια αυτού το κεφαλαίου θα παρουσιαστούν οι πιο βασικοί αλγόριθμοι πολιτικής αξιολόγησης (Policy Gradient - PG).

2.4.2 REINFORCE Algorithm Vanilla Policy Gradient (VPG)

Ο αλγόριθμος Vanilla Policy Gradient (VPG) γνωστός και ως REINFORCE είναι μία PG μέθοδος. Εδώ πρέπει να αναφερθεί ότι η πολιτική του VPG ενεργεί βάσει ενός ΝΔ το οποίο καθορίζει τον τρόπο λειτουργίας της πολιτικής (μέσω αυτού εξάγονται οι πιθανότητες για την επιλογή κάθε ενέργειας δεδομένης της τρέχουσας κατάστασης)

Μέσω της μεταβολής των παραμέτρων αυτού του ΝΔ ο VPG προσπαθεί να προσαρμόσει την πολιτική έτσι ώστε να παράγει τροχιές (αυξάνει την πιθανότητα κάποιων συγκεκριμένων ενεργειών) που θα επιφέρουν όσο το δυνατό μεγαλύτερη αναμενόμενη σωρευτική ανταμοιβή.

Για να επιτύχει αυτές της μεταβολές στις παραμέτρους του δικτύου (ενημέρωση της πολιτικής) κάνει χρήση του θεωρήματος policy gradient. Η πιο κάτω εξίσωση υπολογίζει την πιθανότητα για να συμβεί μια τροχιά Τ.

$$P(T;\theta) = p(S_0)\prod_{t=1}^{T} p(S_t|S_{t-1},A_{t-1})\pi_\theta(A_{t-1}|S_{t-1}) \tag{2.5}$$

Όπου:

- $P(T;\theta)$ αναπαριστά την πιθανότητα της τροχιάς να συμβεί δεδομένης μιας πολιτικής παραμετροποιημένη βάση του θ.
- $p(S_0)$ είναι η πιθανότητα η κατάσταση S_0 να είναι η εναρκτήρια κατάσταση.
- Το γινόμενο ($\prod_{t=1}^{T}$) με όρια από $t = 1$ στο T αντιπροσωπεύει την πιθανότητα μετάβασης από την κατάσταση S_{t-1} στην κατάσταση S_t κατά την λήψει μίας ενέργειας A_{t-1} στη χρονική στιγμή $t-1$, και συμβολίζεται ως $p(S_t|S_{t-1},A_{t-1})$.
- Ο όρος $\pi_\theta(A_{t-1}|S_{t-1})$ είναι η πιθανότητα λήψης της ενέργειας A_{t-1} απο την κατάσταση S_{t-1} δεδομένης μιας πολιτικής που παραμετροποιείται από θ.

Εφαρμόζοντας την ιδιότητα των λογαρίθμων($log(AB) = log(A) + log(B)$) και παράγωγο ως προς την παράμετρο πολιτικής ϑ στην εξίσωση 3.1 καταλήγουμε:

$$\nabla_\theta \log P(T;\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t|S_t) \tag{2.6}$$

Όπου:

- $\nabla_\partial \log P(T;\partial)$ αντιπροσωπεύει την κλίση του λογαρίθμου της πιθανότητας της τροχιάς T σε σχέση με τις παραμέτρους πολιτικής ∂.
- Το άθροισμα ($\sum_{t=0}^{T}$) με όρια από $t = 0$ έως T περιλαμβάνει την κλίση του λογαρίθμου των πιθανοτήτων ενεργειών της πολιτικής, $\pi_\partial(A_t|S_t)$, λαμβάνοντας υπόψη τις παραμέτρους της πολιτικής ∂ σε κάθε βήμα t.

Στην προσπάθεια μας να μεγιστοποιήσουμε την αναμενόμενη σωρευτική ανταμοιβή κάνουμε χρήση της συνάρτησης ανταμοιβής εξίσωσης:

$$J(\partial) = E_{\pi_\partial}[G_t] \tag{2.7}$$

Όπου G_T είναι οι αναμενόμενες σωρευτικές ανταμοιβές για την τροχιά T

$$G_t = \sum_{t=1}^{T} \gamma^{t-1} R_t^T \tag{2.8}$$

και το R_t^T είναι η ανταμοιβή για τη στιγμή t για την τροχιά.

Στη συνέχεια, για να υπολογίσουμε τον gradient της συνάρτησης J(ϑ) ως προς την παράμετρο πολιτικής ϑ, χρησιμοποιούμε τον κανόνα Expectation Over Trajectories:

$$\nabla_\partial J(D;\partial) = E_{\pi_\partial}[\nabla_\partial(\sum_{t=0}^{T} \gamma^t R_t)] \tag{2.9}$$

Όπου:

1. D είναι το σύνολο των τροχιών που χρησιμοποιούνται για την εκτίμηση του gradient,
2. R_t είναι η ανταμοιβή στο χρονικό βήμα t, και γ είναι ο discount factor

Αντικαθιστούμε την ανταμοιβή R_t με το G_t , και αντικαθιστούμε τον όρο E_{π_∂} με τον όρο $\frac{1}{|D|}\sum_{T\in D}$, όπου $|D|$ είναι ο αριθμός των τροχιών στο σύνολο D καταλήγοντας στην εξίσωση:

$$\nabla_\partial J(D;\partial) = \frac{1}{|D|}\sum_{T\in D}\sum_{t=0}^{T} \nabla_\partial \log \pi_\partial(A_t|S_t)\gamma^t G_t \tag{2.10}$$

Συνοπτικά, μέσα από την πιο πάνω εξίσωση ο VPG υπολογίζει επαναληπτικά την κλίση της αναμενόμενης απόδοσης και βάση αυτής γίνονται οι προσαρμογές στις παραμέτρους

του δικτύου της πολιτικής για να επιτευχθεί μεγιστοποίηση της αναμενόμενης σωρευτικής ανταμοιβής. Επίσης λόγω του ότι η εκτίμηση αυτής της κλίσης πηγάζει από ένα σύνολο πιθανών τροχιών θεωρείτε πίο αξιόπιστη. Στόχος του VPG είναι να βρεθεί μια πολιτική που να μεγιστοποιεί την αναμενόμενη απόδοση με την πάροδο του χρόνου.

2.4.3 Trust Region Policy Optimization (TRPO)

Κάθε αλγόριθμος PG διέπεται απο κάποιους περιορισμούς όπως την αποδοτικότητα, την σταθερότητα, σύγκλιση σε τοπικά ελάχιστα κτλ. Ο αλγόριθμος Trust Region Optimization (TRPO) αναπτύχθηκε για να καλύψει σε μεγάλο βαθμό κυρίως τη σταθερότητα και την συγκλίση σε καλές λύσεις. Αυτά τα πλεονεκτήματα του TRPO επιτυγχάνονται μέσα από την εφαρμογή της μεθόδου trust-region η οποία και αναλύεται περαιτέρω πιο κάτω.

Trust Region Constraints και Line-search method

Υπάρχουν αρκετές θεμελιώδεις τεχνικές βελτιστοποίησης οι οποίες χρησιμοποιούνται επαναληπτικά για την βελτίωση της πολιτικής. Οι δυο πιο κύριες τεχνικές που κάνουν χρήση οι περισσότεροι αλγόριθμοι είναι:

1. Line-search method

   Η βασική ιδέα είναι η εύρεση μιας κατεύθυνσης (ανοδική ή καθοδική) στον χώρο της μεταβλητής με απώτερο στόχο να βρει την τιμή η οποία μεγιστοποιεί ή ελαχιστοποιεί το objective function. Οι μέθοδοι αναζήτησης γραμμής χρησιμοποιούνται σε πολλούς αλγορίθμους βελτιστοποίησης, όπως gradient descent, gradient descent.

2. Trust-region methods

   Η βασική ιδέα πίσω από τις μεθόδους Trust Region είναι ότι η αναζήτηση βέλτιστης λύσης περιορίζεται σε έναν συγκεκριμένο όγκο ή περιοχή (συνήθως έχουν ελλειψοειδές σχήμα), που ονομάζεται περιοχή εμπιστοσύνης (trust region). Μέσα από αυτό τον περιορισμό της αναζήτησης διασφαλίζεται ότι η βελτιστοποίηση θα συνεχιστεί σταδιακά ενώ ταυτόχρονα αποφεύγονται τα μεγάλα βήματα που θα μπορούσαν να οδηγήσουν σε αστάθεια.

Όπως σε κάθε αλγόριθμο PG έτσι και στον TRPO, στόχος είναι να βελτιστοποιηθεί μια πολιτική προκειμένου να μεγιστοποιηθεί η αναμενόμενη σωρευτική ανταμοιβή expected reward σε ένα περιβάλλον ενισχυτικής μάθησης. Η διαφορά σε σχέση με τους υπόλοιπους αλγορίθ-

μους PG εκτός του ότι κάνει χρήση της (trust region) είναι ότι η βελτιστοποίηση στην πολιτική πρέπει να γίνει με ένα ρυθμό ο οποίος να εμπίπτει μέσα στο (trust region) για να αποφευχθεί η αστάθεια. Για να διασφαλιστεί ότι η αλλαγή στην πολιτική σε κάθε βήμα εμπίπτει μέσα σε αυτές τις περιοχές γίνεται χρήση της KL Divergence. Η KL Divergence είναι ένα εργαλείο για τον προσδιορισμό της διαφοράς μεταξύ δύο πιθανοτικών κατανομών. Πιο συγκεκριμένα, αναλύει το πόσο διαφέρει η παλαιά πολιτική από νέα βελτιωμένη πολιτική (νέα κατανομή). Το Objective function είναι:

$$L(\partial_k, \partial) = \mathbb{E}\left[\frac{\pi_\partial(a|s)}{\pi_{\partial_k}(a|s)} A(s,a)\right] \pi_{\partial_k}(a|s) \tag{2.11}$$

όπου:

1. $L(\partial_k, \partial)$: Είναι η συνάρτηση κόστος που θέλουμε να μεγιστοποιησουμε με παραμετροποιημένη με ∂_k και∂.

2. Ε : Είναι ο τελεστής για την αναμενόμενη σωρετική ανταμοιβή (expectation operator).

3. $\pi_\partial(a|s)$: Η πολιτική π με παράμετρο θ, που περιγράφει την πιθανότητα να επιλεγεί μια ενέργεια a δεδομένου της κατάστασης s.

4. $\pi_{\partial_k}(a|s)$: Η βελτιωμένη πολιτική π με παράμετρο π_{∂_k}, που χρησιμοποιήται για την σύγκριση με την παλαιά πολιτική $\pi_\partial(a|s)$.

5. $A(s,a)$**: Είναι ο όρος που αναπαριστά τη συνάρτηση πλεονεκτήματος (advantage function), που συνήθως μετρά πόσο καλύτερη ή χειρότερη είναι μια ενέργεια a σε σχέση με το μέσο ή αναφερόμενο επίπεδο.**

Στην εξίσωση 2.11 προστίθετε ο πιο κάτω περιορισμός KL Divergence

$$\partial_{k+1} = \max L(\partial_k, \partial) \tag{2.12}$$

$$\text{subject to } \text{KLD}(\partial \| \partial_k) \leq \delta$$

όπου δ είναι ένα προκαθορισμένο κατώτατο όριο για την KL Divergence.

2.4.4 Proximal Policy Optimization (PPO)

Ο αλγόριθμος Proximal Policy Optimization (PPO) μπορεί να θεωρηθεί ως μία νέα βελτιωμένη έκδοση του TRPO και συνεπώς διέπεται από την ίδια νοοτροπία επίλυσης προβλημάτων η οποία είναι να επιτύχουν όσο το δυνατό μεγαλύτερο βήμα βελτίωσης της πολιτικής τους σε κάθε επανάληψη με την προϋπόθεση ότι δεν θα προκαλέσουν αστάθεια. Το καινοτόμο μέρος του PPO σε σχέση με τον TRPO είναι ότι προσπαθεί να λύσει ένα πρόβλημα μέσω μίας πρωτοβάθμιας εξίσωσης σε αντίθεση με τον TRPO ο οποίος κάνει χρήση μίας πολύπλοκης δεύτερης τάξης εξίσωσης. Συνεπώς οι μέθοδοι του PPO είναι απλούστεροι στην εφαρμογή ενώ ταυτόχρονα εμπειρικά φαίνεται να αποδίδουν τουλάχιστον το ίδιο καλά με το TRPO.

Υπάρχουν δύο κύριες παραλλαγές του αλγορίθμου PPO που διαφέρουν στον τρόπο που διαχειρίζονται την ενημέρωση της πολιτικής κατά την εκπαίδευση.:

1. PPO-Penalty

   Η προσέγγιση PPO-Penalty εισάγει ένα κόστος πέναλτι (penalties) πάνω στη συνάρτηση απώλειας για τις τροποποιήσεις που γίνονται στην πολιτική και είναι εκτός της trust region. Αυτό το κόστος πέναλτι αυξάνεται ανάλογα με το πόσο μεγάλη και εκτός απο την περιοχή εμπιστοσύνης είναι η ενημερώση της πολιτικής, προστατεύοντας έτσι την σταθερότητα της εκπαίδευσης. Ο μαθηματικός τύπος για την προσέγγιση PPO-Penalty είναι:

2. PPO-Clip

   Ο PPO-Clip εισάγει ένα νέο μηχανισμό σχετικά με τον τρόπο που διαχειρίζεται την ενημέρωση της πολιτικής, το "clipping". Το clipping προσδιορίζει ένα προκαθορισμένο εύρος (clipping range) το οποίο καθορίζει το πόσο μπορεί να αλλάξει μία πολιτική κατά την ενημέρωση της. Συγκεκριμένα ο PPO-CLip κατά τη διάρκεια της εκπαίδευσης, υπολογίζει τον λόγο ανανέωσης (ε -update ratio) της πολιτικής (δηλαδή το πόσο μεγάλη είναι η προτεινόμενη αλλαγή σε σχέση με την προηγούμενη πολιτική). Ακολούθως υπαρχουν δύο πιθανά σενάρια:

   (α΄) Εαν ο λόγος ανανέωσης είναι **εντός** του clip range τότε η ενημέρωση της πολιτικής γίνεται ως έχει.

   (β΄) Εαν ο λόγος ανανέωσης είναι **εκτός** του clip range τότε η ενημέρωση της πολιτικής περικόπτεται στα όρια του clip.

$$\nabla_{\text{Clip}}(x, \text{lower},) = \begin{cases} 1 & \text{if lower} \leq x \leq \text{upper} \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

Αυτή η προσέγγιση του PPO-Clip βοηθά στη διατήρηση της σταθερότητας της εκπαίδευσης, καθώς περιορίζει τις υπερβολικές αλλαγές στην πολιτική που μπορεί να οδηγήσουν σε αστάθεια. Επιτρέπει επίσης στον αλγόριθμο να εκπαιδεύει πολιτικές που βελτιώνουν την απόδοση στο περιβάλλον μάθησης, αλλά χωρίς να τροποποιεί αυθαίρετα την πολιτική σε έναν βαθμό που μπορεί να είναι ανεπιθύμητος.

Οι πιο πάνω περιορισμοί οι οποίοι εφαρμόζονται στην εξίσωσωση [46] διασφαλίζουν ότι αυξάνει το ότι δεν θα υπάρχει υπερβολική διαφοροποίηση της πολιτικής σε κάθε βήμα.

Εδώ, θα εστιάσουμε μόνο στο PPO-Clip (την κύρια παραλλαγή που χρησιμοποιείται στο OpenAI).

2.4.5 Deep Deterministic Policy Gradients (DDPG)

Ο αλγόριθμος Deep Deterministic Policy Gradient (DDPG) θεωρείται κατάλληλος για ντετερμινιστικές πολιτικές που εφαρμόζονται σε συνεχούς χώρους που επιστρέφουν πάντα μια μεμονωμένη ενέργεια δεδομένης μιας κατάστασης εισόδου. Επομένως ο αλγόριθμος DDPG είναι σχεδιασμένος ειδικά για την εκπαίδευση πολιτικών που αντιστοιχούν μια ενέργεια σε κάθε κατάσταση. Ένας τρόπος με τον οποίο ένα περιβάλλον συνεχούς χώρου μπορεί να γίνει διαχειρίσιμο μέσω στοχαστικής πολιτικής είναι ο συνδυασμός της μεθόδου ηθοποιού-κριτικού με τα βαθιά νευρωνικά δίκτυα που είναι και η βάση του DDPG.

Συγκρίνοντας τον DDPG με τους προηγούμενους αλγορίθμους που αναφέρθηκαν VPG, TRPO και PPO η κύρια διαφορά τους είναι ότι αυτοί κάνουν χρηση στοχαστικών πολιτικών $\pi : a \sim P(a \mid s)$ για να εξερευνήσουν και να συγκρίνουν ενέργειες. Αντίθετα η κατανομή $P_\vartheta(a \mid s)$ έχει διπλό ρόλο. Αρχικά μέσω αυτής γίνεται η επιλογή της κάθε ενέργειας (στοχαστικά) για μία συγκεκριμένη κατάσταση ενώ ταυτόχρονα βάση αυτής καθορίζεται η εκπαίδευση της πολιτικής μέσα απο τον υπολογίσμό των κλίσεων σε σχέση με την παράμετρο ϑ.

Σε περιβάλλοντα κυρίως του πραγματικού κόσμου (Αυτόνομη Οδήγηση, Βιοϊατρικές Εφαρμογές-δόσης φαρμάκων) όπου η τυχαιότητα θεωρείται ανεπιθύμητη ο DDPG είναι ιδανικός αφού μέσα από την ντετερμινιστική πολιτική που αναπτύσσει εξαλείφει αυτήν την τυχαιότητα, α-

ποδίδοντας απλούστερες και πιο προβλέψιμες πολιτικές.

Το DDPG χρησιμοποιεί τέσσερα νευρωνικά δίκτυα:

1. Actor Network (∂^{μ})

   Το Actor Network χρησιμοποιείται για την δημιουργία μίας ντετερμινιστικής πολιτικής $\mu(s)$, η οποία αντιστοιχίζει καταστάσεις με ενέργειες. Η ενημέρωση του Actor Network γίνεται μέσω του υπολογισμού της κλίσης της ντετερμινιστικής πολιτικής μέσω της εξίσωσης:

$$\nabla_{\partial^{\mu}} J(\partial^{\mu}) \approx \mathbb{E}_{s\sim\rho_{\beta}}[\nabla_{a} Q(s, a|\partial^{Q})|_{a=\mu(s)} \nabla_{\partial^{\mu}} \mu(s|\partial^{\mu})] \tag{2.14}$$

   όπου:

   ∂^{μ} αναπαριστά τις παραμέτρους του actor network.

   $J(\partial^{\mu})$ αναπαριστά την αναμενόμενη σωρετυτική ανταμοιβή της πολιτικής ϑ.

   ρ_{β} αναπαριστά την κατανομή πιθανοτήτων πάνω στο σύνολο των καταστάσεων βάσει της πολιτικής.

2. Critic Network (∂^{Q})

   Μέσω του Critic Network υπολογίζεται το Action value Function(Q(s, a). Η ενημέρωση του Critic Network γίνεται μέσω του υπολογισμού του σφάλματος χρονικής διαφοράς (TD):

$$\nabla_{\partial^{Q}} J(\partial^{Q}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[\nabla_{\partial^{Q}} Q(s, a|\partial^{Q})(Q(s, a|\partial^{Q}) - (r + \gamma \cdot Q(s', \mu'(s')|\partial^{Q'})) \right] \tag{2.15}$$

   όπου:

   ∂^{Q} αναπαριστά τις παραμέτρους του critic network.

   $\mathcal{D}$ αναπαριστά το replay buffer το οποίο περιέχει προηγούμενες εμπειρίες.

   μ' αναπαριστά την πολιτική του target actor network.

   $\partial^{Q'}$ αναπαριστά τις παραμέτρους του target network parameters.

3. Target Network

   Ο αλγόριθμος DDPG κάνει χρήση των δυο πιο κάτω Targets Networks για να επιτύχει σταθερότητα κατα την διαδικασία εκπαίδευσης.

   (α') Actor Target Network ($\partial^{\mu'}$)

$$\partial^{\mu'} \leftarrow \tau \cdot \partial^{\mu} + (1 - \tau) \cdot \partial^{\mu'} \tag{2.16}$$

   (β') Critic Target Network ($\partial^{\mu'}$)

$$\partial^{Q'} \leftarrow \tau \cdot \partial^{Q} + (1 - \tau) \cdot \partial^{Q'} \tag{2.17}$$

   όπου: το τ καθορίζει τον ρυθμό ενημερώσης των δικτύων.

Τέλος ένα ζήτημα που παρουσιάζεται μέσω του υπολογισμού της ντετερμινιστικής κλίσης της πολιτικής είναι η έλλειψη εξερεύνησης. Για την αντιμετώπιση αυτού του ζητήματος έχει εισαχθεί κατα την διαδικασία ενημέρωσης των παράμετρων της πολιτικής του critic θόρυβος εξερεύνησης(exploration noise):

$$\partial^{\mu} \leftarrow \partial^{\mu} + a \cdot \nabla_{\partial^{\mu}} J(\partial^{\mu}) + \epsilon \cdot \mathcal{N}(0, \sigma) \tag{2.18}$$

όπου:

a είναι ο ρυθμός εκμάθησης

ϵ είναι ο θόρυβος εξερεύνησης που λαμβάνεται δείγμα από μια κατανομή θορύβου συνήθως Gaussian με τυπική απόκλιση σ).

2.4.6 Twin Delayed DDPG (TD3)

Αν και ο αλγόριθμος DDPG σε αρκετές περιπτώσεις επιτύγχάνει εξαιρετικές απόδοσεις, συχνά υποφέρει από ζητήματα αστάθειας και χαμηλής απόδοσης. Ο κύριος λόγος αυτών των ζητημάτων είναι λόγω του ότι τα ΝΔ δεν προσεγγίζουν ομαλά την συνάρτηση βελτιστοποίησης με αποτέλεσμα να καταλήγει σε τοπικό ελάχιστο/μέγιστο. Ακόμη ένας λόγος για την κακή απόδοση του αλγορίθμου είναι ότι η συνάρτηση Q που μαθαίνει σταδιακά ο αλγόριθμος υπερεκτιμά τις τιμές Q. Για την αντιμετώπιση του προβλήματος της αστάθειας και της χαμηλής απόδοσης του αλγορίθμου DDPG αναπτύχθηκε ο αλγόριθμος Twin Delayed Deep Deterministic Policy Gradients (TD3) που θεωρείται μια βελτιωμένη εκδοχή του DDPG. Πιο συγκεκριμένα ο αλγόριθμος TD3 εμπεριέχει τρείς νέες μεθόδους:

1. Twin Q Learning

   Σε αντίθεση με τον DDPG ο TD3 κάνει χρήση δύο ανεξάρτητων δικτύων κριτή (critic networks, Q_{ϕ_1} και Q_{ϕ_2}) αντί για ένα. Επομένως το κάθε ένα από αυτά τα δίκτυα κάνει εκτίμηση της συνάρτησης Q και επιλέγεται η μικρότερη από τις δύο αυτές τιμές. Μέσω αυτής της διαδικασίας επιτυγχάνεται μείωση του προβλήματος της υπερεκτίμησης ενώ ταυτόχρονα βελτιώνει τη σταθερότητα της εκπαίδευσης.

2. Target Delay

   Ο TD3 ενημερώνει τα target networks (Actor Target Network ($\partial^{\mu'}$), Critic Target Network ($\partial^{\mu'}$)) λιγότερο συχνά από τα κύρια δίκτυα του κριτή-συνάρτησης Q(Critic Network (∂^{Q})) και της πολιτικής(Actor Network (∂^{μ})). Αυτή η καθυστέρηση ενημέρωσης βοηθά στη βελτίωση της σταθερότητας της εκπαίδευσης και αποφεύγει τον κίνδυνο της υπερεκτίμησης των ενεργειών.

3. Target Smoothing

   Ο TD3 προσθέτει ένα μικρό θόρυβο- σφάλμα (συνήθως σε Gaussian κατανομή) στον στόχο του κριτή (Critic Target), με αποτέλεσμα την μείωση της διακύμανσης του. Αυτή η προσθήκη σφάλματος γίνεται επαναληπτικά και η διακύμανση μειώνεται συνεχώς με αποτέλεσμα να επιτυγχάνεται μεγαλύτερη σταθερότητα, ενισχύοντας έτσι την απόδοση του αλγορίθμου.

2.4.7 Soft Actor-Critic (SAC)

Ο αλγόριθμος Soft Actor-Critic (SAC) μπορεί να θεωρηθεί ως ένας πρωτοποριακός αλγόριθμος ΕΜ λόγω της καινοτόμας εισαγωγής της εντροπίας σε αυτόν. Συγκεκριμένα η εκμάθηση μιας πολιτικής στο SAC δέν απαιτεί την επιστροφή μόνο της υψηλότερης δυνατής αναμενόμενης απόδοσης όπως ισχύει στους υπόλοιπους αλγόριθμους, αλλά μια μέγιστη αντιστάθμιση μεταξύ της αναμενόμενης απόδοσης και της εντροπίας.

Ο όρος εντροπία είναι μια ποσότητα που υποδικνύει πόσο τυχαία είναι μία μεταβλητή και ορίζεται με την πιο κάτω εξίσωση:

$$H(P) = -\mathbb{E}_{x\sim P}[-\log P(x)] \tag{2.19}$$

Ενσωματόνοντας την εντροπία στην ΕΜ για την εύρεση της βέλτιστικης πολιτικής καταλήγουμε στις πιο κάτω εξισώσεις:

1. Εξίσωση βέλτιστικης πολιτικής

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + aH\left(\pi(\cdot|s_t)\right)\right)\right] \tag{2.20}$$

   πού $a > 0$ είναι ο συντελεστής αντιστάθμισης.

2. State-value Function(V-function)

$$V^{\pi}(s) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + aH\left(\pi(\cdot|s_t)\right)\right) \middle| s_0 = s\right] \tag{2.21}$$

3. Action value Function(Q-Function

$$Q^{\pi}(s, a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + aH\left(\pi(\cdot|s_t)\right)\right) \middle| s_0 = s, a_0 = a\right] \tag{2.22}$$

4. Εξίσωση συσχέτισης των τιμών $V_{\pi}(s)$και $Q_{\pi}(s, a)$ και εντροπίας

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi}[Q^{\pi}(s, a)] + aH\left(\pi(\cdot|s)\right) \tag{2.23}$$

Η εισαγωγή της εντροπίας στις πιο πάνω εξισώσεις επηρεάζει άμεσα την σχέση μεταξύ Exploitation και Exploration (Εκμετάλλευσης-Εξερεύνησης). Η αύξηση της εντροπίας οδηγεί σε περισσότερη εξερεύνηση, η οποία μπορεί να επιταχύνει τη μάθηση αργότερα. Μπορεί επίσης να αποτρέψει την πρόωρη σύγκλιση της πολιτικής σε ένα κακό τοπικό βέλτιστο [77].

Ακόμη μία σημαντική μεθόδος που εφαρμόζεται στον SAC η οποία αναφέρθηκε προηγουμένως στον TD3 είναι η χρήση δύο critic networks-(Twin Q Learning). Η ταυτόχρονη διπλή εκτίμηση της Q-συνάρτησης (Q_{ϕ_1}, Q_{ϕ_2}) επιτρέπει στον SAC να εκτιμά την αναμενόμενη ανταμοιβή με μεγαλύτερη ακρίβεια. Αυτό συνεισφέρει στην αυξηση της σταθερότητας της εκπαίδευσης. Επιπλέον, ο SAC χρησιμοποιεί αλγόριθμους προσαρμογής για την ενημέρωση των δύο critic networks, επιτρέποντας τους να προσαρμόζεται δυναμικά στις διαφορετικές πολυπλοκότητες των περιβαλλόντων [77].

2.5 Αλγορίθμοι Βαθίας Ενισχυτικής Μάθησης (Deep Reinforcement Learning Algorithms)

2.5.1 Εισαγωγή

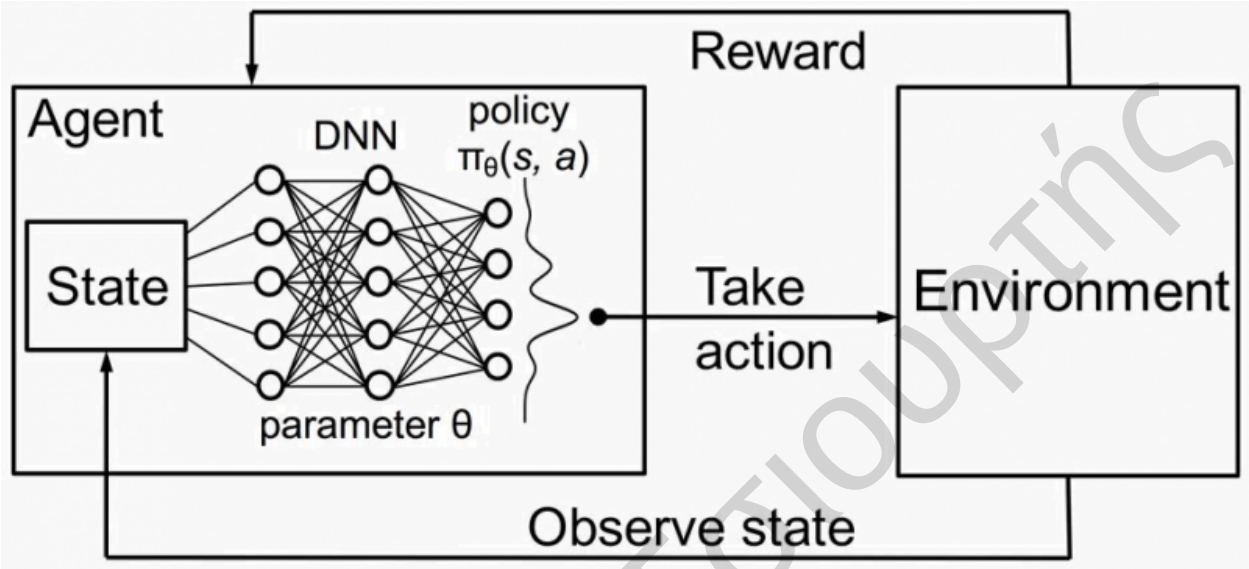
Όπως παρουσιάστηκε και αναλύθηκε στα προηγούμενα κεφάλαια η ΕΜ και η ΒΜ είναι δύο πεδία στην ΤΝ τα οποία έχουν παράξει αξιοσημείωτα και ανεξάρτητα επιτεύγματα τόσο ως προς την λήψη αποφάσεων όσο και προς στην επίλυση σύνθετων προβλημάτων. Πίο συγκεκριμένα η ΒΜ παρέχει την ικανότητα κατανόησης και εξαγωγής περίπλοκων μοτίβων μέσω τεράστιων όγκων δεδομένων, ενώ ταυτόχρονα η ΕΜ έχει επιτύχει την κατάκτηση της μάθησης μέσω της αλληλεπίδρασης με το περιβάλλον επιτρέποντας στους πράκτορες να κάνουν διαδοχικές λήψεις αποφάσεων με απώτερο στόχο την βέλτιστη ανταμοιβή.

Απο το 2015 και έπειτα ένα νέο πεδίο αναδύθηκε μέσω της θεμελιώδους εργασίας "Human-level control through deep reinforcement learning" των Volodymyr Mnih et al. (2015), η οποία συγχωνεύει την ΕΜ και την ΒΜ που οδήγησε στη δημιουργία της Βαθιάς Ενισχυτικής Μάθησης (ΒΕΜ) (Deep Reinforcement Learning-DRL).

Οι ερευνητές αναγνώρισαν ότι ο συνδυασμός της ικανότητας της ΕΜ να μαθαίνει από την αλληλεπίδραση και της ικανότητας της ΒΜ να χειρίζεται πολύπλοκα μοτίβα και την εξαγωγή χαρακτηριστικών θα μπορούσε ενδεχομένως να αντιμετωπίσει τις προκλήσεις που αντιμετώπι-

ζαν οι παραδοσιακές μέθοδοι ΕΜ σε υψηλές διαστάσεις και πολύπλοκα περιβάλλοντα.

Σε αυτό το κεφάλαιο θα γίνει μία εκτενής παρουσίαση τόσο του πρώτου αλγορίθμου ΒΕΜ αλλά και των όσων περιορισμών-προβλημάτων προέκυψαν σε αυτούς καθώς και των αλγορίθμων που επινοήθηκαν για την αντιμετώπιση τους.



Σχήμα 2.24: Συνδιασμός των βασικών στοιχείων που αποτελούν ένα μοντέλο ΒΕΜ [78]

2.5.2 Deep Q-Network (DQN)

Όπως αναφέρθηκε προηγουμένως ο αλγόριθμος Q-learning λειτουργεί πολύ καλά κάνοντας χρήση πινάκα για την αντιμετώπιση προβλημάτων με λίγες και απλές καταστάσεις. Ωστόσο όταν ο αλγόριθμος Q-learning εφαρμοστεί σε σενάρια πραγματικού κόσμου (περιβάλλοντα τα οποία έχουν πολλές δυνατότητες και αποτελέσματα - δηλαδή είναι μεγάλα και περίπλοκα) γίνεται γρήγορα αναποτελεσματικός αφού καθίσταται υπολογιστικά αδύνατη η δημιουργία ενός πίνακα.

Πρόσφατη έρευνα της DeepMind του 2013 στη τεχνητή νοημοσύνη [79]οδήγησε σε νέες τεχνικές βαθιάς ενισχυτικής μάθησης για την αντιμετώπιση αυτού του προβλήματος. Στο άρθρο το οποίο δημοσιεύτηκε [79] παρουσιάζεται για πρώτη φορά αλγόριθμος Deep Q-Network (DQN), ο οποίος συνδυάζει στοιχεία τόσο του αλγορίθμου Q-learning όσο και των νευρωνικών δικτύων έτσι ώστε ο πράκτορας να μάθει μία βέλτιστη πολιτική για τη λήψη αποφάσεων με απώτερο στόχο τη μεγιστοποίηση όσο το δυνατό περισσότερο γίνεται του σωρευτικού σήματος ανταμοιβής. Δεδομένου ότι τα νευρωνικά δίκτυα είναι εξαιρετικά στη μοντελοποίηση σύνθετων

συναρτήσεων, ο αλγόριθμος DQN κάνει χρήση μίας συνάρτησης Q (μέσω ενός νευρωνικού δικτύου) αντί για έναν πίνακα Q, η οποία επιτυγχάνει το ίδιο αποτέλεσμα αντιστοίχισης ζευγών καταστάσεων και ενεργειών σε μια τιμή Q.

Πιο συγκεκριμένα αυτή η συνάρτηση κάνει μία εκτίμηση των τιμών Q όλων των πιθανών ενεργειών που μπορούν να γίνουν από κάθε πιθανή κατάσταση. Η εκτίμηση βασίζεται στην συνεχή εκπαίδευση του δικτύου (μεταβολή παραμέτρων δικτύου - βάρη) έτσι ώστε να μπορεί να εξάγει αυτές τις τιμές Q. Στη συνέχεια γίνεται χρήση αυτών των εκτιμήσεων για να επιλεγεί η βέλτιστη ενέργεια που πρέπει να γίνει από κάθε κατάσταση.

Επίσης έχει παρατηρηθεί ότι το Q-learning μπορεί να υποφέρει από αστάθεια και απόκλιση όταν συνδυάζεται με μια μη γραμμική προσέγγιση συνάρτησης τιμής Q. Η αστάθεια στα νευρωνικά δίκτυα εμφανίζεται όταν η παρτίδα (batch) με την οποία εκπαιδεύεται το δίκτυο εμπεριέχει διαδοχικές ενέργειες οι οποίες συσχετίζονται σε μεγάλο βαθμό μεταξύ τους. Αυτό οδηγεί σε ένα πρόβλημα που ονομάζεται Καταστροφική Λήθη Catastrophic Forgetting, όπου το δίκτυο 'ξεχνά' πράγματα που είχε μάθει λίγο νωρίτερα. Η καλύτερη πρακτική για την εκπαίδευση του δικτύου είναι η παρτίδα να περιέχει δείγματα από τα δεδομένα εκπαίδευσης διαφορετικών χρονικών στιγμών (ανακατεμένα). Αυτό διασφαλίζει ότι υπάρχει αρκετή ποικιλομορφία στα δεδομένα εκπαίδευσης για να επιτρέψει στο δίκτυο να μάθει τέτοια βάρη τα οποία γενικεύονται καλά και μπορούν να χειριστούν μια σειρά από τιμές δεδομένων. Το Deep Q-Network [19] στοχεύει στο να βελτιώσει σημαντικά και να σταθεροποιήσει τη διαδικασία εκπαίδευσης της Q-learning με δύο καινοτόμους μηχανισμούς:

1. Περιοδική ενημέρωση του δικτύου Targer- periodically updated Targer
2. Επανάληψη εμπειρίας Experience Replay

Προτού αναλυθούν αυτοί οι δύο μηχανισμοί θα παρουσιαστεί η αρχιτεκτονική του DQN η οποία απαρτίζεται από:

1. Δύο νευρωνικά δίκτυα:
   - το δίκτυο Q
   - το δίκτυο Target
2. Το στοιχείο που ονομάζεται Επανάληψη εμπειρίας-Experience Replay

Το δίκτυο Q έχει εκπαιδευτεί να παράγει την τιμή Βέλτιστης Κατάστασης-Δράσης. Το δίκτυο

Χ μπορεί να θεωρηθεί ως μία τυπική αρχιτεκτονική νευρωνικών δικτύων (Πχ MLP,CNN,RNN). Το δίκτυο Target είναι πανομοιότυπο με το δίκτυο Q .

Το κάθε επεισόδιο e_t που εκτελείται απο τον πράκτορα αποθηκεύεται σε μία μνήμη επανάληψης $D_t = e_t. . . , e_t$ και αποτελείται από το σύνολο των ενεργειών που εκτελέσθηκαν για κάθε συγκεκριμένη κατάσταση καθώς και την ανταμοιβή και την επόμενη κατάσταση που προκύπτει. $e_t = S_t, A_t, R_t, S_{t+1}$.

Η βασική ιδέα της Επανάληψης εμπειρίας είναι ότι αφού αλληλεπιδράσει με το περιβάλλον και αποθηκεύσει τις εμπειρίες του πράκτορα (μορφή που αναφέρθηκε πιο πάνω), συνδυάζει αυτά τα αποθηκευμένα δεδομένα τυχαία μεταξύ τους για να δημιουργήσει παρτίδες οι οποίες περιέχουν παλαιότερη αλλά και νέα εμπειρία για την εκπαίδευση του δικτύου Q. Μέσα από αυτή την διαδικασία διασφαλίζεται ότι η παρτίδα περιέχει αρκετή ποικιλομορφία από παλαιότερα και νεότερα δείγματα για να επιτραπεί στο δίκτυο να ενημερώσει τα βάρη του έτσι ώστε να δημιουργήσει μία γενίκευση για να μπορεί να αντεπεξέλθει ορθά σε όλα τα σενάρια που θα απαιτείται να χειριστεί [80].

Η βασική Περιοδική ενημέρωση του δικτύου Targer κατά την διάρκεια της διαδικασίας εκπαίδευσης είναι μία από τις σημαντικότερες προσθήκες που έγιναν για την δημιουργία του αλγορίθμου DQN. Το πρόβλημα που εμφανίστηκε με την χρήση ενός μόνο δικτύου ήταν ότι η τιμές του δικτύου Q μετατοπίζονταν και άλλαζαν συνεχώς με αποτέλεσμα το δίκτυο να αποσταθεροποιείται πέφτοντας σε βρόχους ανάδρασης μεταξύ του στόχου και των εκτιμώμενων τιμών Q. Προκειμένου να μετριαστεί αυτός ο κίνδυνος, έγινε χρήση ενός δεύτερου δικτύου Targer του οποίου τα βάρη είναι σταθερά **(δεν εκπαιδεύεται)** και ενημερώνονται περιοδικά αλλά όχι μετά το πέρας της κάθε παρατήρησης. Αυτή η ενημέρωση γίνεται βάση των τιμών του δικτύου Q. Με αυτόν τον τρόπο η εκπαίδευση μπορεί να προχωρήσει με πιο σταθερό τρόπο. Αυτό το δεύτερο δίκτυο χρησιμοποιείται για τη δημιουργία των τιμών target-Q που θα χρησιμοποιηθούν για τον υπολογισμό της απώλειας Loos function για κάθε ενέργεια κατά τη διάρκεια της εκπαίδευσης.

Ο αλγόριθμος Deep Reinforcement Learning έχει επιφέρει εντυπωσιακή πρόοδο στην ΕΜ και γενικά στην τεχνητή νοημοσύνη τα τελευταία χρόνια, ξεπερνώντας ακόμη και την ανθρώπινη απόδοση σε τομείς στους οποίους μερικά χρόνια προηγουμένως φαινόταν ακατόρθωτο, όπως σε κάποια παιχίδια στο Atari 2600 [79], το Go [3] και το πόκερ. Αν και ο αλγόριθμος DQN έπαιξε καθοριστικό ρόλο στον τομέα της ΕΜ, με το πέρας του χρόνου έχουν αναγνωριστεί

αρκετοί περιορισμοί του. Βάση αυτών των περιορισμών έχουν προταθεί αρκετές νέες επεκτάσεις του αλγορίθμου για αντιμετώπιση τους. Ο κάθε αλγόριθμος που αναφέρεται πιο κάτω στοχεύει στην αντιμετώπιση συγκεκριμένων περιορισμών του αλγορίθμουDQN:

1. Διπλό Q-Learning (Double Q-Learning)
2. Επανάληψη εμπειρίας προτεραιότητας (Prioritized experience replay)
3. Αντιπαλικά δίκτυα (Dueling Networks)
4. Εκπαίδευση πολλαπλών βημάτων (Multi-step train)
5. Ενισχυτική Μάθηση Κατανομής (Distributional RL)
6. Αλγόριθμος Rainbow (Rainbow algorithm)

2.5.3 Double Q-Learning (DDQN)

Τον Σεπτέμβριο του 2015 στο άρθρο “Deep Reinforcement Learning with Double Q-learning” [81] εξηγεί πως η εφαρμογή του DQN σε στοχαστικά περιβάλλοντα και συγκεκριμένα ο αλγόριθμος Q-Learning δεν αποδίδει καλά. Αυτή η χαμηλή απόδοση αποδόθηκε στον περιορισμό της υπερεκτίμησης των τιμών ενέργειας(overestimation bias) που γίνεται χρησιμοποιώντας ένα μεμονωμένο δίκτυο Q με την χρήση του Max Q(s’,a) και που οδηγεί σε υπερβολικά αισιόδοξες και μη βέλτιστες αποφάσεις πολιτικής.

Στον παραδοσιακό αλγόριθμο Q, το ίδιο δίκτυο Q σε κάθε κατάσταση επιλέγει άπληστα την μεγαλύτερη τιμή της συνάρτησης Q(γmaxQ(s_{t+1},a)) που είναι διαθέσιμη για την κατάσταση s_{t+1} βλέποντας μυωπικά . Αυτό μπορεί να οδηγήσει σε υπερεκτίμηση των τιμών κάθε ενέργειας και ιδιαίτερα σε σενάρια όπου υπάρχουν θόρυβος ή ανακριβείς εκτιμήσεις καταστάσεων. Η υπερεκτίμηση των τιμών ενέργειας επηρεάζει τη διαδικασία εκμάθησης αναγκάζοντας τον πράκτορα ΕΜ να ευνοήσει ενέργειες που μπορεί να μην είναι πραγματικά βέλτιστες, εμποδίζοντας τελικά τον πράκτορα να λαμβάνει βέλτιστες αποφάσεις σε πολύπλοκα περιβάλλοντα.

Εδώ πρέπει να αναφερθεί ότι η πρώτη έκδοση του αλγόριθμου DDQN [82] που παρουστικε ως λύση του προβλήματος υπερεκτίμησης, εισήγαγε δύο διαφορετικά Q-network για την εκτίμηση των συναρτήσεων Q (δηλαδή εφάρμοσε δύο ανεξάρτητες συνάρτησης εκτίμησής Q^A, Q^B). Ενώ το 2015 παρουσιάζεται μία πιο βελτιωμένη έκδοση η οποία εισάγει το διαχωρισμό της

διαδικασίας εκτίμησης των συναρτήσεων Q από τη διαδικασία της επιλογής της βέλτιστης ενέργειας μέσω της χρήσης δύο διαφορετικών νευρωνικών δικτύων:

1. Q-network

   Το Q-network παίζει διπλό ρόλο:

   1.1 Χρησιμοποιείται για την εκτίμηση των τιμών Q για όλες τις πιθανές ενέργειες σε μια δεδομένη κατάσταση. Αυτό διακρίνεται απο την πίο κάτω εξίσωση Bellman αφού για τον υπολογισμό της τιμής Q για μια συγκεκριμένη ενέργεια σε μια δεδομένη κατάσταση γίνεται χρήση του Q-network:

$$Q(s,a) = \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma \max_{a'} Q(s',a')\right] \tag{2.24}$$

   1.2 Επιλέγει την καλύτερη ενέργεια ανάμεσα σε αυτές που είναι διαθέσιμες για την δεδομένη κατάσταση

2. Target neural network Κάνει αξιολόγιση της επιλογής ενέργειας που έκανε το Q-network

Μέσα απο την εισαγωγή αυτής της διαδικασίας έχει αποδειχθεί ότι μειώνονται οι υπερεκτιμήσεις, γεγονός που οδηγεί σε καλύτερες πολιτικές.

2.5.4 Dueling DQN

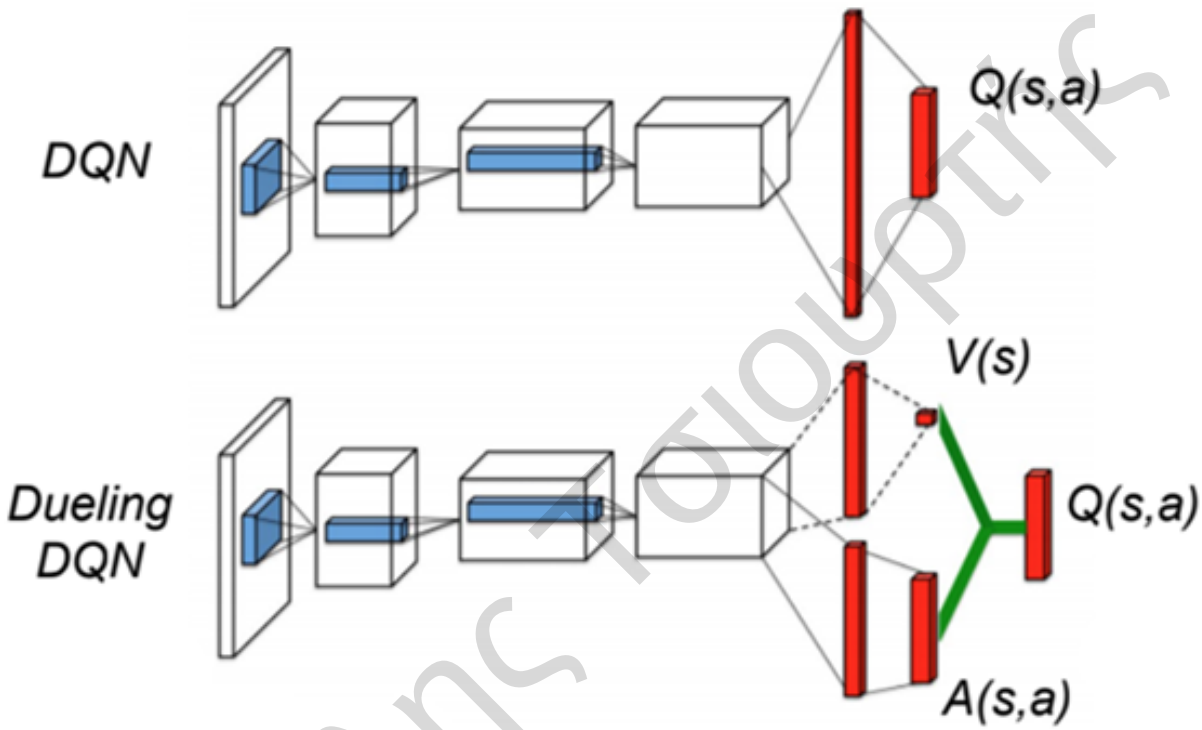
Όπως ο αλγόριθμος Double Q-Learning (DDQN) θεωρήται μία βελτιωμένη έκδοση του DQN, έτσι και ο Dueling DQN [83] έχει στόχο να βελτιώσει τον DQN. Η διαφορά μεταξύ του Double Q-Learning (DDQN) και του Dueling DQN είναι ότι ο πρώτος επικεντρώνεται στην βελτίωση του προβλήματος overestimation bias ενώ ο δεύτερος επικεντρώνεται στη βελτίωση της διαδικασίας αξιολόγησης των ενεργειών του αλγόριθμου DQN σε περιπλοκότερα περιβάλλοντα, στα οποία η εκτίμηση της αξίας είναι δύσκολη. Για να επιτύχει αυτή την βελτίωση ο Dueling DQN διαχωρίζει τη συνάρτηση Q σε δύο μέρη όπως διακρινεται απο την πίο κάτω εξίσωση:

$$Q(s,a) = V(s) + A(s,a) \tag{2.25}$$

Όπού:

$A(s,a)$ Συνάρτηση Πλεονεκτήματος (Advantage function) η οποία μας λέει πόσο καλύτερη είναι μια ενέργεια σε σύγκριση με το σύνολο των υπόλοιπων ενεργειών.

$V(s)$ Συνάρτηση Αξίας Κατάστασης (V-function) η οποία μας λέει πόση είναι η αναμενώμενη ανταμοιβή από την κατάσταση s.



Σχήμα 2.25: Στη πάνω μέρος απεικονίζεται η αρχιτεκτονική ενός κανονικού DQN δικτύου. Στο κάτω μέρος απεικονίζεται η αρχιτεκτονική ενός κανονικού Dueling DQN δικτύου. [83]

Όπως διακρίνεται και στο Σχήμα2.31 ο αλγόριθμος Dueling DQN προτείνει το διαχωρισμό του τελευταίου στρώματος του δικτύου σε δύο μέρη:

1. Το ένα μέρος θα είναι υπεύθυνο για την εκτίμηση της State-value Function(V-function)για την κατάσταση s ($V(s)$).
2. Το δεύτερο μέρος θα είναι υπεύθυνο για την εκτίμηση της συνάρτησης πλεονεκτήματος για κάθε ενέργεια a ($A(s,a)$).

Στο τέλος θα γίνεται συνδυάσμός των δύο αυτών αποτελεσμάτων σε μια ενιαία έξοδο, η οποία θα κάνει εκτιμήση της τιμής $Q(s,a)$.

Ωστόστο για την εκπαίδευση του ΝΔ δεν επαρκεί μόνο η τιμή της συνάρτησης Q, αφού

δεν μπορούμε να βρούμε τις συναρτήσεις $A(s,a)$ και $V(s)$. Αυτή η έλλειψη δυνατότητας εύρεσης του $V(s)$ και $A(s,a)$ δεδομένου του $Q(s,a)$ δημιουργεί πρόβλημα κατα το βήμα back propagation της διαδικασίας εκπαίδευσης και συνεπώς καταλήγουμε στο πρόβλημα identifiable.[1].

Για να ξεπεραστεί αυτό το πρόβλημα προτάθηκε αρχικά η πιο κάτω εξίσωση:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a' \in |A|} A(s,a';\theta,\alpha)\right) \tag{2.26}$$

ενώ μία δεύτερη παραλαγή είναι η ατικατασταση του όρου $\max_{a' \in |A|} A(s,a';\theta,\alpha))$ με το μέσο όρο της συνάρτησης πλεονεκτήματος όλων των καταστάσεων. Μέσω αυτής της αντικατάστασης η εξίσωση παρουσιάζεται ως:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha)\right) \tag{2.27}$$

2.5.5 Prioritized Experience Replay

Το 2105 παρουσιάστηκε μία νέα τεχνική η Prioritized Experience Replay(PER) απο τον Tom Schaul [84] και εφαρμόζεται κυρίως σε αλγορίθμους όπως το Deep Q-Network (DQN). Η κύρια ιδέα πίσω απο την ανάπτυξη αυτής της νέας τεχνικής είναι το ότι ορισμένα δεδομένα (εμπειρίες) που αποκτά ο πράκτορας είναι πιο σημαντικά από άλλα για την εκπαίδευσή, αλλά συνήθος συμβαίνουν λιγότερο συχνά.

Μέχρι τότε η δειγματοληψία για την παρτίδα (batch) γινόταν με ομοιόμορφη κατανομή (τα δεδομένα-εμπειρίες επιλέγονταν ισοπίθανα-τυχία μέσα απο το experience replay buffer) και συνεπώς οι πίο σημαντικές πληροφορίες εμφανίζονται λιγότερο συχνά απο τες μή τόσο σημαντικές.

Έχοντας στόχο την επίλυση αυτού του προβληματος, η μέθοδος PER αλλάζει την κατανομή δειγματοληψίας μέσω ενός κριτηρίου προτεραιότητας του κάθε (batch) εμπειριών που βρίσκεται μέσα στο experience replay buffer.

[1]Στους τομής της στατιστικής και της μηχανική μάθηση, το πρόβλημα της "identifiability"(ταυτοποιησιμότητα) αναφέρεται στην ικανότητα να μπορέσουμε να εκτιμήσουμε τις παραμέτρους ενός στατιστικού μοντέλου από τα δεδομένα που έχουμε στη διάθεσή μας. Το πρόβλημα της identifiability μπορεί να προκύψει όταν το μοντέλο είναι υπερπαραμετροποιημένο, δηλαδή περιέχει περισσότερες παραμέτρους από ό,τι μπορούμε να εκτιμήσουμε από τα δεδομένα [45].

Η πιο κάτω εξίσωση υπολογιζει τη πιθανότητα επιλογής για κάθε (batch) εμπειριών εκπαίδευσης:

$$P(i) = \frac{\frac{p(j)}{a}}{\sum_j \frac{p(j)}{a}} \tag{2.28}$$

Όπου:

α σταθερή παράμετρος που ρυθμίζει το βαθμό της προτεραιότητας

Το άθροισμα στον παρονομαστή είναι το συνολικό άθροισμα των προτεραιοτήτων επιλογής των (batch) εμπειριών.

Ενώ το $p(j)$ είναι η προτεραιότητα για το (batch) εμπειριών και η εξίσωση υπολογισμού του είναι:

$$p(i) = |\delta_i| + \epsilon \tag{2.29}$$

Όπου:

- $|\delta_i|$ είναι το σφάλμα πρόβλεψης (TD error) για το (batch) εμπειριών i.

  Το σφάλμα TD (Temporal Difference) υπολογίζεται για κάθε (batch) εμπειριών, και προσδιορίζει το πόσο καλά προβλέπει το ΝΔ το (reward) που λήφθηκε μετά από μια ενέργεια συγκρίνοντας το με το (expected return). Στην ουσιαα αντιπροσωπεύει τη διαφορά μεταξύ της εκτιμημένης ανανταμοιβής (expected return) και της πραγματικής ανατροφοδότησης (reward). Ο στόχος είναι να μειώσουμε αυτό το σφάλμα μέσω της εκπαίδευσης του μοντέλου.

  Η εξίσωση για το σφάλμα TD είναι:

$$\delta_i = r_i + \gamma \max_a Q(s_{i+1}, a; \theta^-) - Q(s_i, a_i; \theta) \tag{2.30}$$

όπου:

r_i είναι η ανταμοιβή που λήφθηκε μετά την ενέργεια a_i στην κατάσταση s_i.

γ είναι ο παράγοντας εκπτωτικής απόσβεσης.

$Q(s_{i+1}, a; \theta^-)$ είναι η εκτίμηση της αξίας Q της καλύτερης ενέργειας στην κατάσταση s_{i+1} με τα βάρη του προηγούμενου (σταθερά) δικτύου, γνωστού ως το ταργετ νετωορκ.

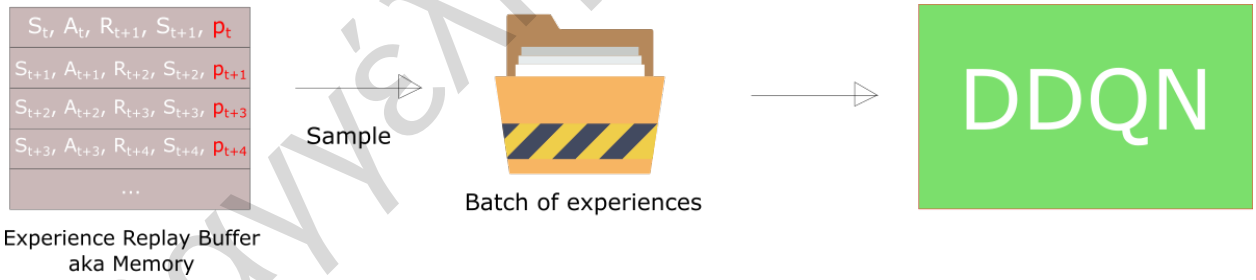
$Q(s_i, a_i; \theta)$ είναι η τρέχουσα εκτίμηση της αξίας X της ενέργειας a_i στην κατάσταση s_i με τα τρέχοντα βάρη του δικτύου.

- ϵ είναι μία μικρή θετική σταθερά για να αποφευχθεί η διαίρεση με μηδέν σε περίπτωση που το $|\delta_i|$ είναι ίσο με μηδέ.

δίνουμε αυτή την προτεραιότητα στην εμπειρία κάθε buffer επανάληψης.

Αλλά δεν μπορούμε να κάνουμε απλώς άπληστη ιεράρχηση, γιατί θα οδηγήσει στο να εκπαιδεύουμε πάντα τις ίδιες εμπειρίες (που έχουν μεγάλη προτεραιότητα) και, επομένως, να ταιριάζουν υπερβολικά.

Έτσι εισάγουμε τη στοχαστικότητα στην διαδικασία δειγματοληψίας (εισαγωγή πιθανότητας για την επιλογή δειγματος απο το Experience Replay Buffer Deep όπως διακρίνεται στο Σχήμα2.31.



Σχήμα 2.26: Εισαγωγή της στοχαστικότητας στην διαδικασία δειγματοληψίας(εισαγωγή πιθανότηταςγια την επιλογή δειγμάτων απο το Experience Replay Buffer Deep για το Q-Network (DQN) [85]

$$P(i) = \frac{|\delta_i|^a}{\sum_j P(j)^a} \tag{2.31}$$

2.5.6 Rainbow DQN

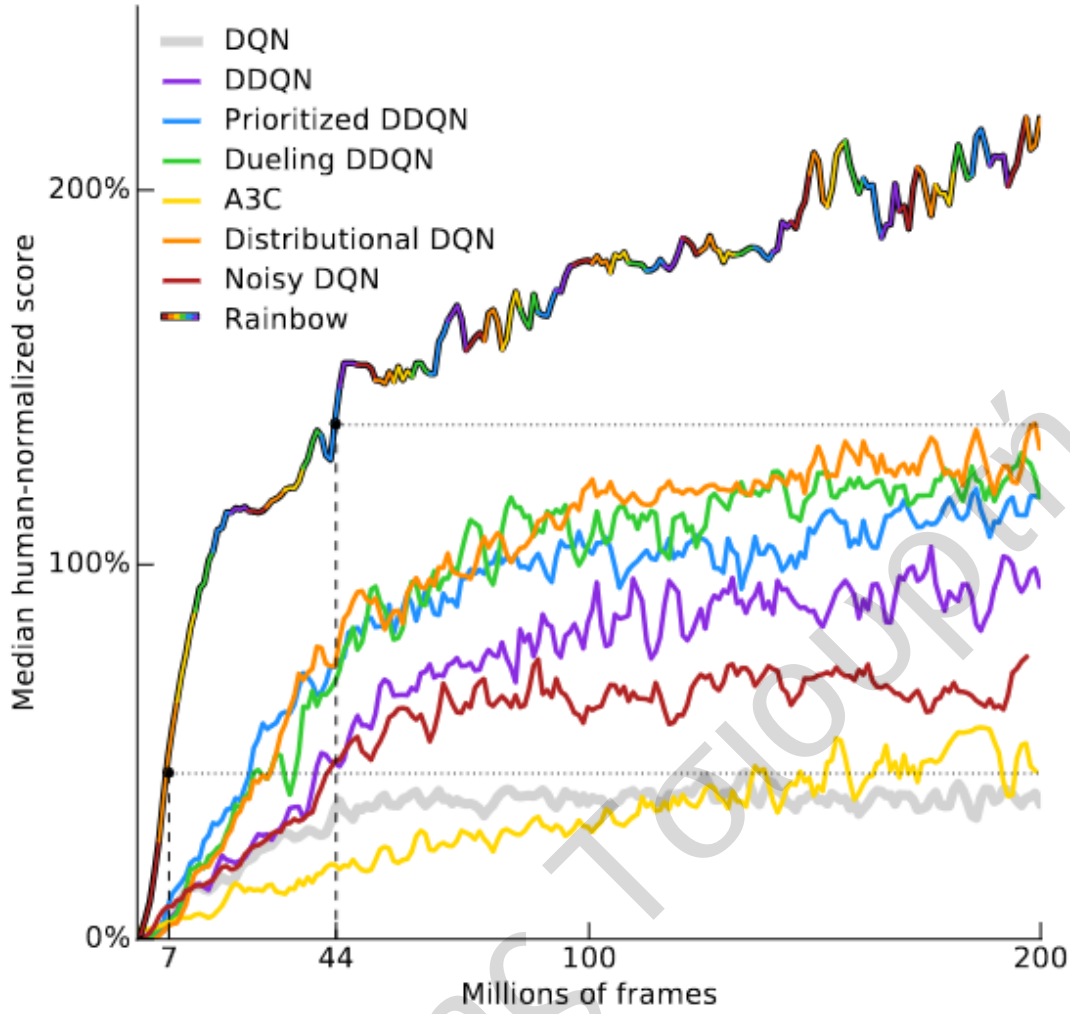
Όπως αναφέρθηκε και προηγουμένως η αλλαγή του τρόπου αντιμετώπισης των προβλημάτων της ΕΜ από έναν πίνακα Q (εμπεριέχει όλες τις τιμές όλων των ζευγών κατάστασης-δράσης) με ένα νευρωνικό δίκτυο DQN (θεωρητικά ικανό να μετατρέψει οποιαδήποτε κατάσταση εισόδου σε τιμή εξόδου ανά ενέργεια) μπορεί να αντιμετώπισε το πρόβλημα της υψηλής διαστάσιμότητας και των πολύπλοκων περιβαλλόντων [86]. Αυτή η νέα μέθοδος έφερε μαζί της διαφορετικά προβλήματα. Για την αντιμετώπιση αυτών των προβλημάτων έχουν προταθεί διάφορες τεχνικές και αλγορίθμοι όπου κάθε μία βελτιώνει τον αρχικό DQN αλγοριθμο ως πρός ένα συγκεκριμένο μειωνέκτημα του.

Το 2017 η Deepmind έχοντας ως κύριο ερώτημα "Εάν μια τεχνική που εφαρμόστηκε στον DQN μπορεί να βελτιώσει δραστικά την απόδοση, γιατί να μην συνδιαστούν όλες σε μία·" δημοσίευσε το άρθο Rainbow: Combining Improvements in Deep Reinforcement Learning [87]. Οι συγγραφείς αυτού του άρθου προσπάθησαν να δούν ποίες απο αυτές τις τεχνικές:

1. Double DQN
2. Prioritized Experience Replay
3. Dueling Network Architecture
4. Multi-step Learning
5. Distributional value estimation as in C51 Algorithm
6. Noisy Networks

είναι συμβατές μεταξύ τους αλλά ταυτόχρονα να παρέχουν και καλύτερη απόδοση.

Ο συνδιασμός όλων των πίο πάνω δοκιμάστηκε σε 57 περιβάλλοντα- παιχνίδια στο Atari 2600 αποδείκνύοντας ότι ο συνδυασμός όλων αυτών των τεχνικών επιφέρει απόδοση πολύ υψηλότερη απο την κάθε τεχνική ξεχωριστά.

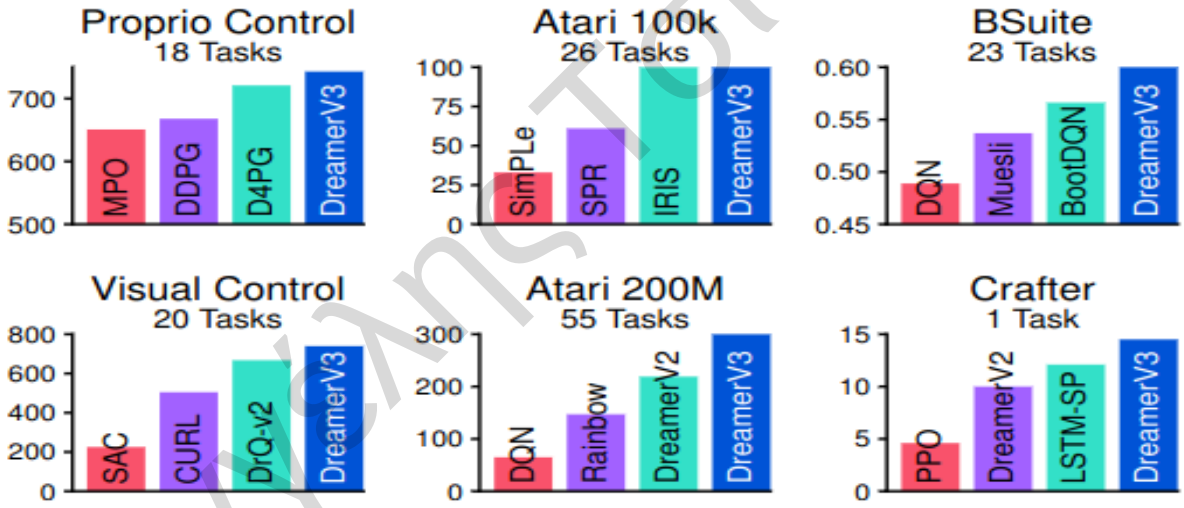


Σχήμα 2.27: Σε αυτό το σχήμα διακρίνεται η μέση απόδοση του κάθε αλγορίθμου βάση των 57 παιχνιδιών Atari σε σχέση με την μέση ανθρώπινη απόδοση (άξονας y) και των αριθμό των φραμες που έχουν χρησιμοποιηθεί για την εκπαίδευση του αλγόριθμο (άξονας x). Παρατηρείτε ότι μετά τα 7 εκατομμύρια frames ο αλγόριθμος Rainbow ξεπερνά οποιοδήποτε άλλο αλγόριθμο σε απόδοση αλλά και την μέση ανθρώπινη απόδοση. [87]

2.5.7 DreamerV3

2.5.7.1 Εισαγωγή

Το παρόν υποκεφάλαιο εξετάζει εκτενώς τον βασισμένο σε μοντέλα αλγόριθμο DreamerV3, έναν γενικό και επεκτάσιμο αλγόριθμο Ενισχυτικής Μάθησης που βασίζεται σε δύο κύριες μεθόδους, το World Model και το Actor-Critic, προκειμένου να επιτύχει εξαιρετικά αποτελέσματα σε ποικίλους τομείς. Παρέχεται λεπτομερής ανάλυση του τρόπου λειτουργίας του DreamerV3 με βάση αυτές τις δύο μεθόδους, ενώ παράλληλα εξετάζεται η απόδοσή του σε επτά βασικά είδη παιχνιδιών-benchmarks, καθένα από τα οποία χαρακτηρίζεται από διαφορετικό τύπο περιβάλλοντος. Επισημαίνεται ιδιαιτέρως η εξαιρετική απόδοση του αλγορίθμου στην πρόκληση του Minecraft, όπου το DreamerV3 καταφέρνει να συλλέξει διαμάντια χωρίς την παραμικρή ανθρώπινη παρέμβαση, κατοχυροντας την θέση του πρώτου αλγορίθμου που το επιτυγχάνει αυτό.



Σχήμα 2.28: Παρουσίαση σύγκρισης της απόδοσης ποικίλων αλγορίθμων με τον DreamerV3 σε έξι από τα επτά διαφορετικά Benchmarks που χρησιμοποιήθηκαν για την συνολική αξιολόγηση. [4]

2.5.7.2 Περιγραφή DreamerV3

Τον Ιανουάριο του 2023, δημοσιεύτηκε το άρθρο [4], στο οποίο παρουσιάζεται ο αλγόριθμος DreamerV3 βασισμένος σε μοντέλα. Ο εν λόγω αλγόριθμος θεωρείται καινοτόμος στον τομέα της ΕΜ. Όπως αναφέρθηκε προηγουμένως, η κύρια πρόκληση για τους αλγορίθμους στον τομέα της ΕΜ είναι η γενίκευση (Versatility) και η κλιμάκωση (Scalability) τους σε διάφορα προβλήματα και νέους τομείς. Για να επιτευχθεί αυτό, συνήθως απαιτείται εξειδικευμένη

γνώση και πολλοί υπολογιστικοί πόροι για να αξιολογηθεί κατά πόσο ο κάθε αλγόριθμος, ρυθμίζοντας τις υπερπαραμέτρους του, μπορεί να προσαρμοστεί και να επιλύσει διαφορετικά προβλήματα σε διαφορετικούς τομείς και διαστάσεις. Αυτά τα προβλήματα έρχεται να τα αντιμετωπίσει σε μεγάλο βαθμό ο αλγόριθμος DreamerV3. Εμφανίζεται να ξεπερνά τους φραγμούς της εξειδικευμένης γνώσης και της υπολογιστικής ισχύος παρέχοντας ένα σύνολο σταθερών υπερπαραμέτρων, κατακτώντας τη γενίκευση και την κλιμάκωση μέσω της ικανότητάς του να εφαρμόζεται άμεσα σε μια ποικιλία πρακτικών περιβαλλόντων και της υψηλής απόδοσής του σε αυτά, η οποία αυξάνεται όσο μεγαλώνει το μέγεθος του μοντέλου.

Name	Symbol	Value
General		
Replay capacity (FIFO)	—	10^6
Batch size	B	16
Batch length	T	64
Activation	—	LayerNorm + SiLU
World Model		
Number of latents	—	32
Classes per latent	—	32
Reconstruction loss scale	β_{pred}	1.0
Dynamics loss scale	β_{dyn}	0.5
Representation loss scale	β_{rep}	0.1
Learning rate	—	10^{-4}
Adam epsilon	ϵ	10^{-8}
Gradient clipping	—	1000
Actor Critic		
Imagination horizon	H	15
Discount horizon	$1/(1-\gamma)$	333
Return lambda	λ	0.95
Critic EMA decay	—	0.98
Critic EMA regularizer	—	1
Return normalization scale	S	$\text{Per}(R, 95) - \text{Per}(R, 5)$
Return normalization limit	L	1
Return normalization decay	—	0.99
Actor entropy scale	η	$3 \cdot 10^{-4}$
Learning rate	—	$3 \cdot 10^{-5}$
Adam epsilon	ϵ	10^{-5}
Gradient clipping	—	100

Σχήμα 2.29: Στον παραπάνω πίνακα διακρίνονται οι τιμές των υπερπαραμέτρων του αλγορίθμου DreamerV3, οι Οποίες χρησιμοποιήθηκαν σε όλες τις συγκρίσεις αξιολόγησης που διεξήχθησαν στο άρθρο του [4]

2.5.7.3 Ο Τρόπος Λειτουργίας του DreamerV3 με Βάση τις Μεθόδους World Model και Actor-Critic

Σε αυτή την υποενότητα θα παρουσιαστούν και θα εξηγηθούν οι δύο βασικές μεθόδοι, World Model και Actor-Critic, στις οποίες βασίζεται ο αλγόριθμος DreamerV3 για τη λειτουργία του και ακολούθος θα γίνει αναφορά το πως ενσωματώνονται σε αυτών.

World Model

Σύμφωνα με τον Jay Wright Forrester (πατέρα των δυναμικών συστημάτων) στο άρθρο [5] ο άνθρωπος αναπτύσσει ένα νοητικό μοντέλο του κόσμου που κατά κόρων είναι βασισμένο στις αισθήσεις του. Όλες οι αποφάσεις και οι ενέργειες που εκτελούμε βασίζονται σε αυτό το μοντέλο που το περιγράφει ως την εικόνα του κόσμου γύρω μας που κουβαλάμε στο κεφάλι μας.

Ο άνθρωπος καθημερινά βομβαρδίζεται συνεχώς από ένα τεράστιο όγκο δεδομώνων μέσω των αισθητήριων όργανων του. Για να γίνει εφικτή η διαχείριση αυτού του τεράστιου όγκου, ο ανθρώπινος εγκέφαλός μας μαθαίνει να αναπαριστά τη πληροφορία η οποία είναι ένας συνδυασμός της χωρικής και χρονικής αντίληψης όχι με επακριβής λεπτομέρεια αλλά μέσω μίας αφηρημένης αναπαράσταση. Επομένως είμαστε ικανή να παρατηρούμε το περιβάλλον και να θυμόμαστε μια αφηρημένη περιγραφή του και όχι την επακριβής αναπαράσταση [88] [89](δηλαδή με μειωμένη πληροφορία-κρατά την ουσία).

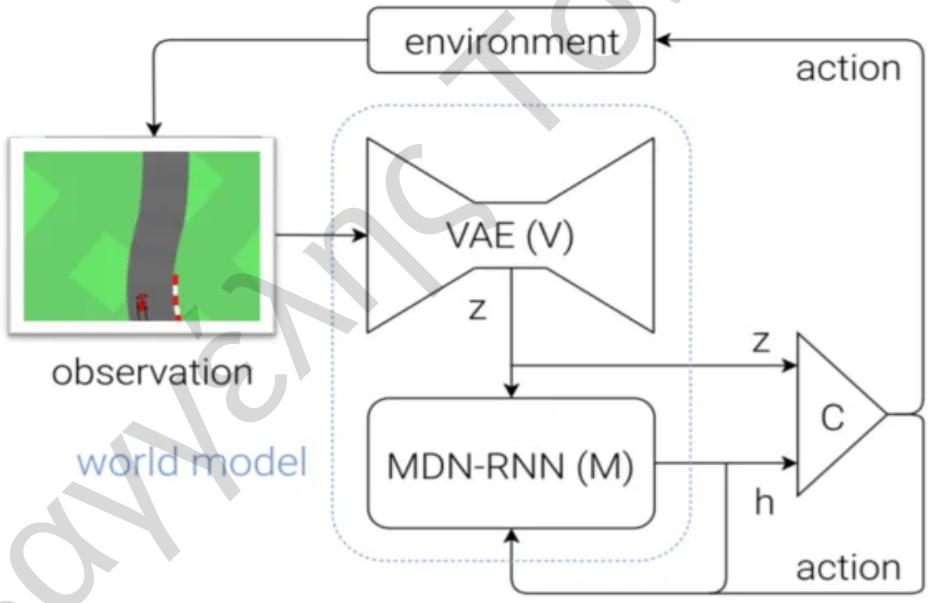
Σύμφωνα με τα αποτελέσματα των συγγραφέων των άρθρων Sensorimotor mismatch signals in primary visual cortex of the behaving mouse [90],A Sensorimotor Circuit in Mouse Cortex for Visual Flow Predictions [91] οι οποίοι παρατηρούσαν την δραστηριότητα και αλληλεπίδραση μεταξύ οπτικού και κινητικού φλοιού σε ποντίκια, διαπίστωσαν ότι η δραστηριότητά τους συσχετίστηκε ισχυρά με την κίνηση με την κίνηση και την προκύπτουσα οπτική ανατροφοδότηση ροής με τρόπο που εξαρτάται από την εμπειρία.

Με πιο απλά λόγια υπάρχουν ενδείξεις που υποδηλώνουν ότι ο εγκέφαλος μας, πιθανόν, κατανοεί την παρούσα στιγμή με βάση τις προβλέψεις των μελλοντικών αισθητηριακών δεδομένων, λαμβάνοντας υπόψη τις τρέχουσες κινήσεις μας , στηριζόμενος σε εσωτερικά μοντέλα. Αυτό επιτρέπει άμεσες αντιδράσεις και συμπεριφορές οι οποίες θεωρούνται ενστικτωδώς, δηλαδή χωρίς την ανάγκη συνειδητοποιημένου σχεδιασμού ενός σχεδίου στρατηγικής [5].

Στο άρθρο δίνει ως παράδειγμα για καλύτερη κατανόηση το παιχνίδι του μπέιζμπολ.

Ένας παίκτης μπέιζμπολ πρέπει να αποφασίσει πώς θα χτυπήσει την εισερχόμενη μπάλα(ταχύτητα πέραν των 160 χιλιομέτρων την ώρα) σε λιγότερο χρόνο από ό,τι χρειάζεται το οπτικό σήμα να φτάσει στον εγκέφαλο(120ms) και να εκδοθούν εντολές στους μύες. Η ικανότητά μας να προβλέπουμε ενστικτωδώς την πορεία της μπάλας μας επιτρέπει χτυπούν το ρόπαλο τη σωστή στιγμή και στη σωτή τοποθεσία σύμφωνα με τις προβλέψεις των εσωτερικών τους μοντέλων [92], χωρίς συνειδητή προσπάθεια για σχεδιασμό [93]. Οι επαγγελματίες παίκτες αντιδρούν αυτόματα, ακολουθώντας τις προβλέψεις των εσωτερικών τους μοντέλων, επιτρέποντας τους να δράσουν γρήγορα χωρίς συνειδητό σχεδιασμό [5].

Στη συνέχεια το άρθρο παρουσιάζει ένα σχετικά απλό μοντέλο που είναι εμπνευσμένο από το γνωστικό σύστημα του ανθρώπου που αναφέρθηκε πιο πάνω. Συγκεκριμένα το μοντέλο μπορεί να θεωρηθεί ως αλγόριθμος ανάδρασης διαχωρίζοντας τον πράκτορα σε δύο ΝΔ και συγκεκριμένα σε ένα μεγάλο που είναι για το World model και ένα μικρότερο μοντέλο που αντιστοιχεί σε έναν controler.



Σχήμα 2.30: Στο σχήμα απεικονίζεται η αρχιτεκτονική της μεθόδου World Model [5]

Το world model αποτελείται από τρία βασικά μέρη: το Vision (V), το Memory (M), και το Controller (C). Κάθε μέρος εκπαιδεύεται ξεχωριστά, συνδυάζοντας τεχνικές ενίσχυσης μάθησης με generative models. Το κείμενο που ακολουθεί παρέχει λεπτομερείς πληροφορίες για τη λειτουργία κάθε μέρους του μοντέλου.

1. Vision (V):

   - Χρησιμοποιείται ένα Convolutional Variational Autoencoder (VAE).
   - Λαμβάνει μια παρατήρηση από το περιβάλλον, η οποία είναι μια εικόνα 2D-3D με διαστάσεις 64x64x3 (πλάτος, μήκος, RGB βάθος).
   - Κωδικοποιεί την παρατήρηση σε έναν latent vector z με μέγεθος 32 (δηλαδή συμπιέζει την παρατήρηση που λαμβάνει απο το περιβάλλον την χρονική στιγμή t σε ένα χαμηλών διαστάσεων latent διάνυσμα Z_t). Αυτή η συμπιεσμένη αναπαράσταση μπορεί να χρησιμοποιηθεί για την ανακατασκευή της αρχικής εικόνας

2. Memory (M):

   Ο ρόλος του μοντέλου M είναι να προβλέπει το μέλλον, λειτουργεί ως προγνωστικό μοντέλο των μελλοντικών διανυσμάτων z που αναμένεται να παράγει το V.

   Το μοντέλο M είναι βασικά ένα Long Short-Term Memory (LSTM) με ένα Mixture Density Network (MDN).

   Συγκεκριμένα παίρνει ως εισόδους:

   - latent vector z από το Vision
   - την προηγούμενη ενέργεια a που επιλέγεται από τον controler
   - και την προηγούμενη κρυφή κατάσταση h του ίδιου.

3. Controller (C):

   Εκπαιδεύεται έτσι ώστε να δίνει στην έξοδο του μια ενέργεια για κάθε χρονική στιγμή συνδιάζοντας το latent vector z το οποίο προέρχεταιαπό το Vision component και το h vector από το Memory component.

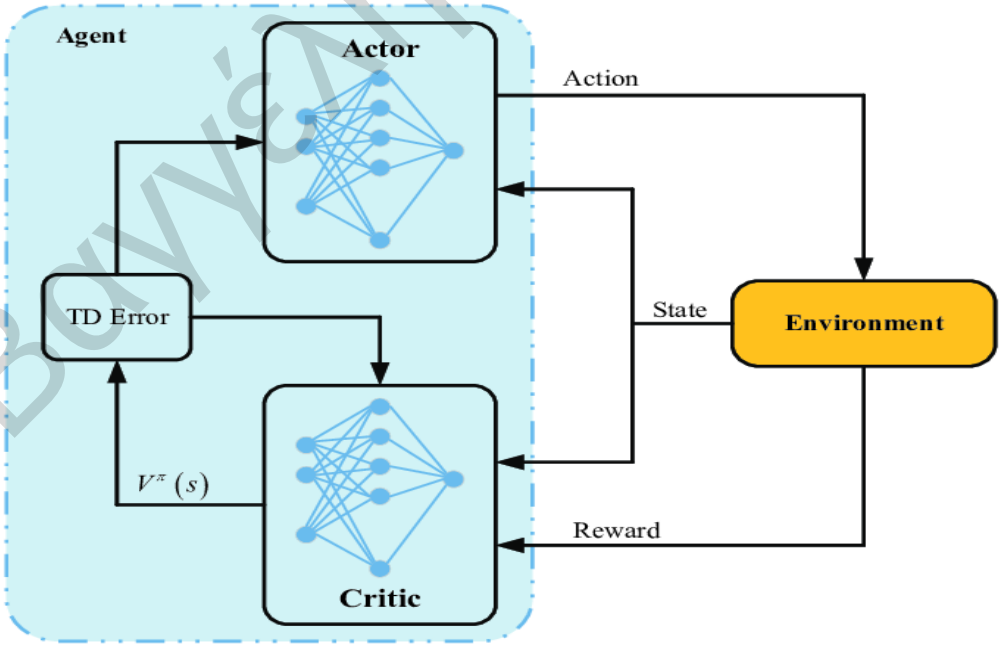
Actor Critic:

Μία σχετικά νέα μέθοδος στον τομέα της ΕΜ είναι η μέθοδος "Actor-Critic" , όπου ο στόχος είναι να μάθει ένα σύστημα να πραγματοποιεί ακολουθίες ενεργειών σε ένα περιβάλλον με σκοπό τη μεγιστοποίηση μιας ανταμοιβής [10].

Η μέθοδος Actor-Critic αποτελεί μια τεχνική βασισμένη σε Temporal Difference (TD) και αποτελείται από δύο ξεχωριστά δίκτυα νευρώνων. Το πρώτο δίκτυο, που είναι ο Actor, αντιπροσωπεύει τον πράκτορα που λαμβάνει αποφάσεις και εκτελεί ενέργειες σε ένα περιβάλλον. Αντί να επικεντρώνεται στον υπολογισμό μιας ακριβούς συνάρτησης αξίας, ο Actor επιδιώκει να βελτιστοποιήσει απευθείας την πολιτική, δηλαδή τον τρόπο με τον οποίο επιλέγονται οι ενέργειες [10]. Από την άλλη, ο Critic λειτουργεί ως εκτιμητής της ποιότητας των ενεργειών που αναλαμβάνει ο Actor. Αναπτύσσει μια συνάρτηση αξίας που εκτιμά πόσο καλές είναι οι ενέργειες που επιλέγονται. Αυτή η πληροφορία χρησιμοποιείται για τη βελτίωση της πολιτικής του Actor με βάση την εκτίμηση της συνάρτησης αξίας [10].

Εξίσωση του Actor-Critic για τον υπολογισμό του gradient J(ϑ) που είανι η μέση τιμή των m τροχιών:

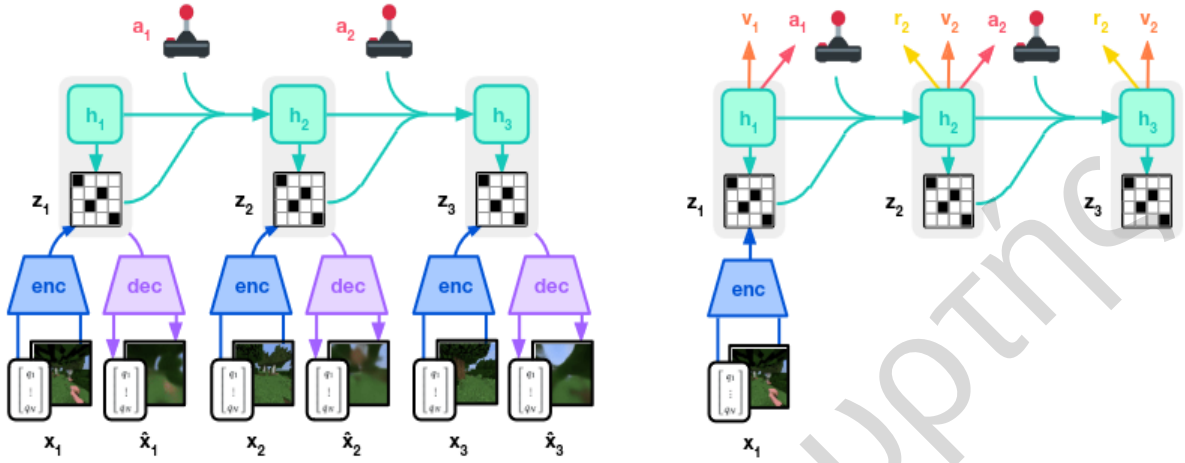
$$\nabla_{\vartheta} J(\vartheta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T} \nabla_{\vartheta} \log \pi_{\vartheta}(a_t \mid s_t)(Q_{t,a_t} - V_{\varphi}(s_t)) \tag{2.32}$$



Σχήμα 2.31: Στο σχήμα απεικονίζεται η αρχιτεκτονική της μεθόδου Actor-Critic στην ΒΕΜ [94]

Τρόπος Λειτουργίας του DreamerV3

Ο αλγόριθμος DreamerV3 αποτελείται από τρία βασικά νευρικά δίκτυα: το (world model), το (critic) και το (actor) όπως φένεται στο Σχήμα2.32



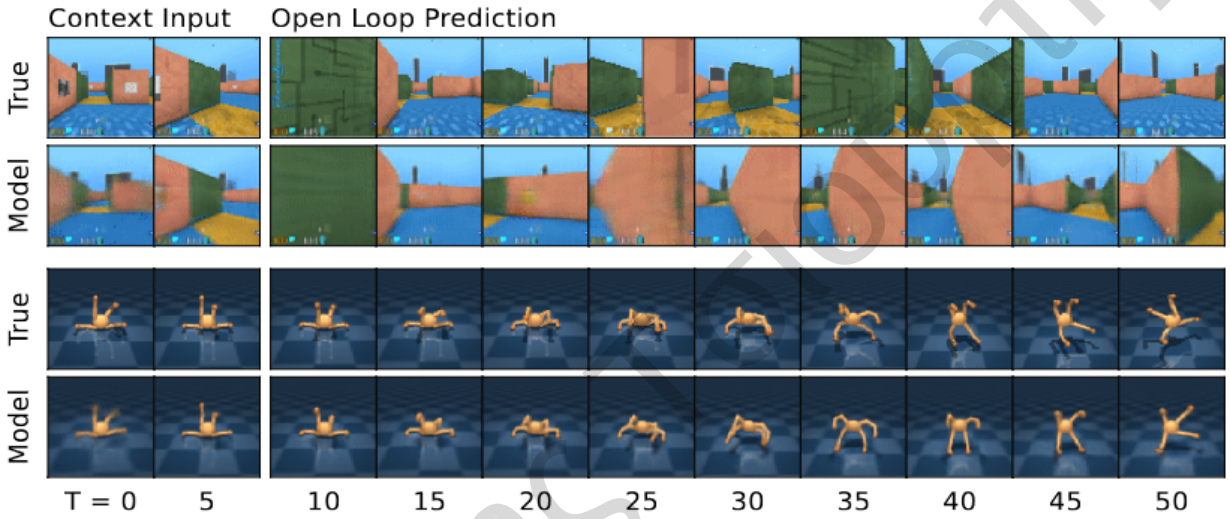
Σχήμα 2.32: Το Σχήμα παρουσιάζει τη διαδικασία εκπαίδευσης του DreamerV3. Το (world model) κωδικοποιεί τις εισόδους σε ένα χαμηλών διαστάσεων latent διάνυσμα Z_t που προβλέπεται από ένα μοντέλο ακολουθίας προηγούμενη κρυφή κατάσταση h_t δεδομένων των ενεργειών a_t. Ο Actor και ο Critic μαθαίνουν από τροχιές των παραγώμενων αναπαραστάσεων που προβλέπονται από το world model [4]

Το world model λειτουργεί αρχικά ως κωδικοποιητής παρατηρήσεων τις οποίες λαμβάνει από το περιβάλλον στη χρονική στιγμή t, μετατρέποντας τις σε ένα χαμηλών διαστάσεων latent διάνυσμα Z_t σε συνδιασμό με την προηγούμενη κρυφή κατάσταση h_t. Εδώ πρέπει να αναφερθέι ότι το world model μέσω ενός αποκωδικοποιητή κάνοντας χρήση της συμπιεσμένης αναπαράστασης Z_t επιτυγχάνει την ανακατασκευή της αρχικής εικόνας, όπως φαίνεται στο Σχήμα 2.32 (a). Το βασικό έργο του world model είναι να προβλέπει τις μελλοντικές εικόνες-αναπαραστάσεις h_{t+1}, μερικά παραδείγματα εμφανίζονται στο Σχήμα 2.33. Για να επιτύχει αυτήν την πρόβλεψη, το world model χρησιμοποιεί το latent διάνυσμα σε συνδυασμό με την προηγούμενη κρυφή κατάσταση h_t από το Recurrent State-Space Model - RSSM και μια πιθανή ενέργεια a_t.

Το πρώτο δίκτυο, που είναι ο Actor, αναπαριστά τον πράκτορα που λαμβάνει αποφάσεις και εκτελεί ενέργειες σε ένα περιβάλλον, λαμβάνοντας ως είσοδο τις μελλοντικές εικόνες-αναπαραστάσεις που προβλέπονται από το World Model. Κατά τη διάρκεια της εκπαίδευσης, ο Ηθοποιός μαθαίνει να επιλέγει ενέργειες που μεγιστοποιούν την αναμενόμενη ανταμοιβή r_i μέσω της βελτίωσης της πολιτικής του.

Ο Actor είναι υπεύθυνος για τον υπολογισμό της αναμενόμενης ανταμοιβής (expected return) r_t για κάθε κατάσταση του μοντέλου. Ο Actor επίσης υπολογίζει τη συνάρτηση αξίας $V_\pi(s)$ που χρησιμοποιείται για να αξιολογήσει το πόσο καλές είναι οι ενέργειες που επιλέγονται από τον Actor και συμβάλλει στην καθοδήγηση του Actor προς ενέργειες που προκαλούν υψηλή ανταμοιβή.

Σημειώνεται ότι κατά τη διάρκεια της εκπαίδευσής τους, δεν υπάρχει κοινός υπολογισμός των κλίσεων μεταξύ τους. Αυτό σημαίνει ότι κάθε ένα από αυτά τα δίκτυα μαθαίνει ανεξάρτητα, βασιζόμενο σε εμπειρίες που του παρέχονται.



Σχήμα 2.33: Απόδοση της πρόβλεψης πολλαπλών μελλοντικών βημάτων στα περιβάλλοντα DMLab και Control Suite. χρησιμοποιώντας αρχικά 5 στιγμιότυπα από κάθε περιβάλλον, το World Model κατορθώνει να προβλέψει 45 βήματα μπροστά στο χρόνο, λαμβάνοντας υπόψη την ακολουθία ενεργειών χωρίς πρόσβαση σε ενδιάμεσες εικόνες. Αυτή η εξαιρετική ικανότητα πρόβλεψης πολλαπλών βημάτων αναδεικνύει τη δύναμη του DreamerV3 στην αντιμετώπιση διαφορετικών προκλήσεων σε ποικίλα περιβάλλοντα [4]. Στην πάνω γραμμή Κάθε συστοιχίας εικόνων, η πρώτη γραμμή απεικονίζει την πραγματική ροή της χρονοσειράς των βημάτων μέσω εικόνων, ενώ η κάθε κάτω γραμμή απεικονίζει την προβλεπόμενη χρονοσειρά των βημάτων μέσω εικόνων.

2.5.7.4 Αξιολόγηση του DreamerV3 σε Επτά Κύρια benchmarks

Για την επαλήθευση των ιδιοτήτων γενικότητας και κλιμάκωσης του αλγορίθμου DreamerV3, καθώς και για την αξιολόγηση της απόδοσής του σε σύγκριση με άλλους εξειδικευμένους αλγορίθμους, υποβλήθηκε σε εμπειρική αξιολόγηση. Η αξιολόγηση διεξήχθη σε επτά benchmarks που περιλαμβάνουν συνεχείς (continuous) και διακριτές (discrete) ενέργειες, οπτικές (visual) αλλά και χαμηλές διαστάσεις (low-dimensional) εισόδου, πυκνές (dense) και αραιές (sparse) ανταμοιβές, διαφορετικές κλίμακες ανταμοιβής, 2D και 3D περιβάλλοντα.

Παρακάτω γίνεται αναφορά στα 7 benchmarks που χρησιμοποιήθηκαν στο άρθρο Mastering Diverse Domains through World Models [4]. Οι γραφικές παραστάσεις της απόδοσης καθώς και των γραφημάτων απεικονίζονται στα appendix του άρθου του DreamerV3 [4].

Τα επτά benchmarks είναι τα ακόλουθα:

1. **Proprio Control Suite (18 Continuous Control Tasks, 500K Steps):**
   - **Χαρακτηριστικά:**
     (α΄) 18 συνεχή περιβάλλοντα με χαμηλές διαστάσεις εισόδου.
     (β΄) Εκτέλεση πειραμάτων σε ένα σύνολο 500 χιλιάδων βημάτων.
   - **Επίδοση:** Ο αλγόριθμος DreamerV3 κατακτά νέο ρεκόρ στο benchmark, ξεπερνώντας D4PG, DMPO, και MPO.
2. **Visual Control Suite (20 Tasks, 1M Steps):**
   - **Χαρακτηριστικά:**
     (α΄) 20 συνεχή περιβάλλοντα με υψηλές διαστάσεις εισόδου.
     (β΄) Εκτέλεση πειραμάτων σε ένα σύνολο 1 εκατομμυρίου βημάτων.
   - **Επίδοση:** Ο αλγόριθμος DreamerV3 κατακτά νέο ρεκόρ στο benchmark, ξεπερνώντας DDrQ-v2, CURL.
3. **Atari 100k (26 Games, 400K Steps):**
   - **Χαρακτηριστικά:**
     (α΄) 26 παιχνίδια Atari
     (β΄) Εκτέλεση πειραμάτων σε ένα σύνολο 400 χιλίαδων βημάτων.

- **Επίδοση:** Ο αλγόριθμος DreamerV3 καταφέρνει να ξεπεράσει όλους τους άλλους αλγορίθμους με τους οποίους συγκρίθηκε, αλλά δεν καταφέρνει να ξεπεράσει τον EfficientZero [95], ο οποίος θεωρείται η πιο αποδοτική λύση στο εν λόγω πρόβλημα.

4. **Atari 200M (55 Games, 200M Steps):**

   - **Χαρακτηριστικά:**

     (α') 55 παιχνίδια Atari

     (β') Εκτέλεση πειραμάτων σε ένα σύνολο 200 εκατομμυρίων βημάτων.

   - **Επίδοση:** Ο αλγόριθμος DreamerV3 κατακτά νέο ρεκόρ, ξεπερνώντας τον DreamerV2 και τους αλγόριθμους Rainbow και IQN.

5. **BSuite (23 Environments, 468 Configurations):**

   - **Χαρακτηριστικά:**

     (α') 23 περιβάλλοντα

   - **Επίδοση:** Ο αλγόριθμος DreamerV3 κατακτά νέο state-of-the-art, ξεπερνώντας Bootstrap DQN και Muesli και Rainbow, θεωρούμενοι ως οι προηγούμενοι πρωτοπόροι στον τομέα.

6. **Crafter (Procedurally Generated Survival Environment):**

   - **Χαρακτηριστικά:**

     (α') Παιχνίδι επιβίωσης με γραφικά 2D όπου η γωνία της κάμερας δείχνει τον παίκτη και το περιβάλλον γύρω του από ψηλά, και οι ενέργειες σε αυτό είναι διακριτικές.

     (β') Εκτέλεση πειραμάτων σε ένα σύνολο 1 εκατομμυρίου βημάτων.

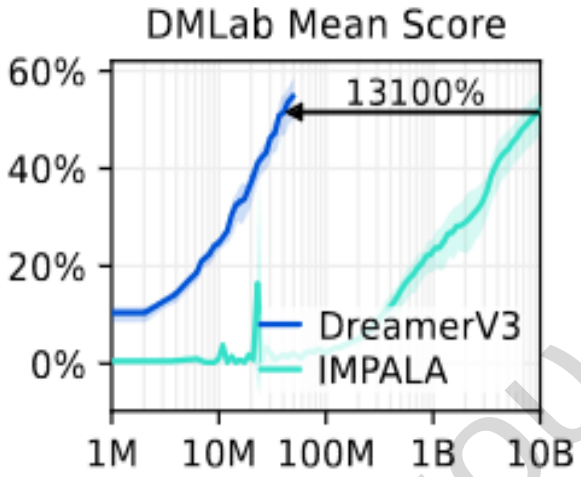
   - **Επίδοση:** Ο αλγόριθμος DreamerV3 κατακτά νέο state-of-the-art, ξεπερνώντας PPO με την αρχιτεκτονική LSTM-SPCNN, OC-SA, DreamerV2 και Rainbow.

7. **DMLab (8 Tasks, 50M Steps):**

   - **Χαρακτηριστικά:**

     (α') 8 προκλητικά περιβάλλοντα σε 3D

- **Επίδοση:** Ο αλγόριθμος DreamerV3 κατακτά νέο state-of-the-art, ξεπερνώντας την τελική απόδοση του αλγορίθμου IMPALA. Η αύξηση της απόδοσης ξεπερνά το 13000% σε σύγκριση με τον IMPALA σε 10 δισεκατομμύρια βήματα, όπως φαίνεται στο Σχήμα 2.34.



Σχήμα 2.34: Απο το γράφιμα διακρίνεται η απόδοση του αγλορίθμου DreamerV3 σε σχέση με τον IMPALA αλγόριθμο για το περιβάλλον DMLab [4]

2.5.7.5 Επιτυχία στην Πρόκληση Συλλογής Διαμαντίων στο Παιχνίδι Minecraft:

Ένα από τα πιο διαδεδομένα βιντεοπαιχνίδια στον κόσμο θεωρείται το Minecraft [96], με πάνω από 141 εκατομμύρια ενεργούς παίκτες παγκοσμίως και καθημερινή συμμετοχή περίπου 1 έως 1,5 εκατομμυρίου παικτών [97]. Το περιβάλλον του Minecraft είναι τρισδιάστατο με πολλά διαφορετικά σενάρια.

Στο παιχνίδι Minecraft, μία από τις πολλές προκλήσεις που αντιμετωπίζει ο παίκτης είναι η εύρεση και συλλογή ενός διαμαντιού. Η εύρεση του διαμαντιού θεωρείται δύσκολη λόγω του ότι απαιτεί μια μακροσκελή και χρονοβόρα διαδικασία, αποτελούμενη από 12 διαφορετικές υποδιαδικασίες. Αυτές περιλαμβάνουν την εύρεση και περισυλλογή πόρων, όπως ξύλο και πέτρα, και τη δημιουργία εργαλείων με τη χρήση των περισυλεγμένων πόρων. Η δυσκολία προκύπτει από το γεγονός ότι δεν μπορείς να φτάσεις στο σημείο όπου βρίσκονται τα διαμάντια χωρίς τα συγκεκριμένα εργαλεία, και το ίδιο ισχύει για κάθε εργαλείο που απαιτεί άλλους πόρους για την κατασκευή του. Αυτή η διαδικασία έχει θεωρηθεί ως μια πρόκληση στον τομέα της τεχνητής νοημοσύνης, και ειδικότερα στον τομέα της ΕΜ. Επίσης, αξίζει να σημειωθεί ότι τα διαμάντια μπορούν να βρεθούν και σπάζοντας θησαυροφυλάκια, αλλά η

εύρεση διαμαντιών με αυτόν τον τρόπο επιτυγχάνει μέγιστο σκορ για το επεισόδιο και, ως εκ τούτου, αυτή η μέθοδος δεν συμπεριλαμβάνεται στα στατιστικά που παρουσιάζονται στο DreamerV3.

Ένα σημαντικό επίτευγμα του αλγορίθμου DreamerV3 αποτελεί η επιτυχής αντιμετώπιση της πρόκλησης της δημιουργίας διαμαντιών στο Minecraft. Σε αντίθεση με προηγούμενες προσεγγίσεις που εξαρτώνταν από ανθρώπινα δεδομένα για την εκπαίδευση των αλγορίθμων, ο DreamerV3 καταφέρνει να συλλέξει διαμάντια χωρίς καμία υπόδειξη. Μάθαινε όλες τις υποδιαδικασίες ξεχωριστά και, τελικά, επιτυγχάνει τη συλλογή ενός διαμαντιού.

Ακολούθως, παρουσιάζεται αναλυτικά η διαδικασία που ακολούθησε ο DreamerV3 για να επιτύχει τη συλλογή διαμαντιού:

1. **Υπερπαραμετροποίηση**

Αρχικά, οι υπερπαραμέτροι του DreamerV3 τέθηκαν σταθερές, όπως εφαρμόστηκε και στα άλλα βενσημαρκς.

2. **Εκπαίδευση dreamerv3 σε κάθε υποδιαδικασία**

Η εφαρμογή του DreamerV3 ξεκίνησε για κάθε μία από τις υποδιαδικασίες ξεχωριστά, για 100 εκατομμύρια βήματα στην κάθε μία, όπως φαίνεται στο σχήμα 2.28.

3. **Αριθμός Seeds**

Για κάθε μία από αυτές τις υποδιαδικασίες, εφαρμόστηκαν 40 seeds για να επιτευχθεί γενίκευση του αλγορίθμου και σε διάφορες συνθήκες.

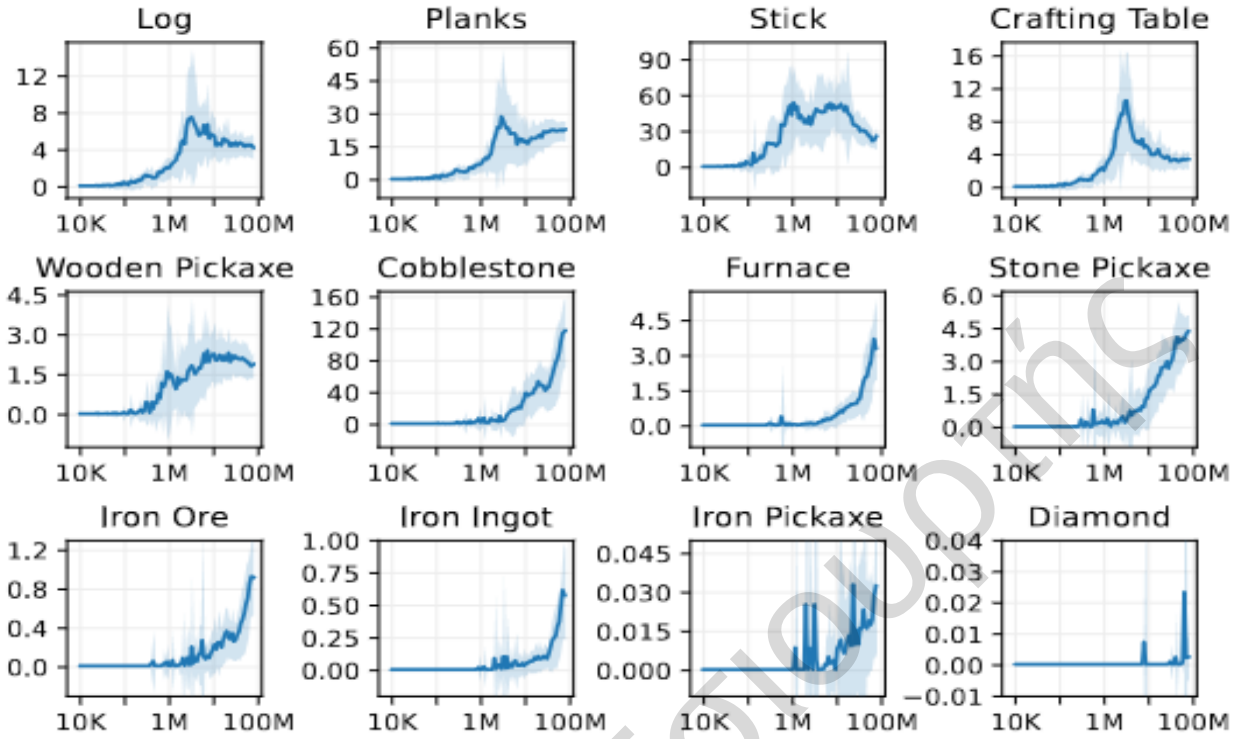
4. **Επιτάχυνση της εκπαίδευσης**

Για επιτάχυνση της εκπαίδευσης, έγινε εφαρμογή επιτάχυνσης στο παιχνίδι στα 20 Hz (20 αλλαγές ή κινήσεις στο παιχνίδι ανά δευτερόλεπτο).

5. **Αποτελέσαμτα**

Το πρώτο διαμάντι επετεύχθη στα 29,3 εκατομμύρια βήματα(17 μέρες παιχνιδιού - 5,6 ώρες), με τη συχνότητα αυτή να αυξάνεται καθώς προχωρά η εκπαίδευση. Συνολικά, από τα 40 συνολικά seeds, ο DreamerV3 κατόρθωσε να συλλέξει τουλάχιστον ένα διαμάντι σε 24 από αυτά, με τον μέγιστο αριθμό διαμαντιών σε ένα seed να είναι 6. Επιπλέον, ο μεσαίος αριθμός των βημάτων μέχρι τη συλλογή του πρώτου διαμαντιού ήταν 74 εκατομμύρια, αντιστοιχώντας

σε 42 ημέρες παιχνιδιού (13.86 ώρες) στα 20 Hz.



Σχήμα 2.35: Απο το γράφημα διακρίνονται τα Score για κάθε υποδιαδικασία ξεχωριστά, σε σχέση με τον αριθμό των βημάτων, λαμβάνοντας υπόψη και την διακύμανση [4].

2.6 Ενισχυτική Μάθηση και Ηλεκτρονικά Παιχνίδια (Reinforcement Learning and Electronic Games)

2.6.1 Εισαγωγή

Τα λεκτρονικά παιχνίδια (ΗΠ), πέρα από την αναμφισβήτητη ψυχαγωγική τους αξία, αποτελούν μια διανοητική πρόκληση για τον άνθρωπο. Η αντίδραση σε αυτή την πρόκληση οδήγησε τον άνθρωπο στην αναζήτηση μέσων που θα του επέτρεπαν να επιτύχει όσο το δυνατόν υψηλότερες επιδόσεις σε αυτά τα παιχνίδια. Αυτή η ανάγκη δεν χρειάστηκε πολύ για να οδηγήσει την επιστημονική κοινότητα να αναγνωρίσει τα βιντεοπαιχνίδια ως ένα από τα κυριότερα πεδία δοκιμών στην τεχνητή νοημοσύνη και, συγκεκριμένα, στην ΕΜ. Τα ΗΠ, λόγω της φύσης τους, προσφέρουν ένα δυναμικό και προκλητικό περιβάλλον, μέσα από το οποίο οι πράκτορες της ΕΜ μπορούν να μαθαίνουν και να βελτιστοποιούν σταδιακά την απόδοσή τους. Μέχρι σήμερα, έχουν εφαρμοστεί διάφοροι αλγόριθμοι ΕΜ σε κλασικά επιτραπέζια παιχνίδια, όπως το σκάκι και το Go, καθώς και σε σύγχρονα ΗΠ όπως το Dota 2 και το Minecraft. Μέσω αυτής της αλληλεπίδρασης πράκτορα με ΗΠ μέσα στο πλαίσιο της ΕΜ, οι ερευνητές αναγκάστηκαν να αναπτύξουν καινοτόμους αλγόριθμους και τεχνικές ΕΜ για να επιτύχουν όσο το δυνατόν υψηλότερες επιδόσεις. Την τελευταία δεκαετία, έχουν διοργανωθεί πολλοί διαγωνισμοί εφαρμογής αλγορίθμων ΕΜ σε ΗΠ, όπως το OpenAI Gym και το Dota 2 OpenAI Five. Αυτοί οι διαγωνισμοί αναγκάζουν τους ερευνητές να βελτιώνουν τους πράκτορες τους και να ξεπερνούν τα όρια τους, με αποτέλεσμα να προκύπτει αξιόλογη πρόοδος στον τομέα της ΕΜ. Στην παρούσα ενότητα γίνεται μια αναφορά στην χρονική εξέλιξη της ΕΜ μέσα από την παρουσίαση των κυριότερων εφαρμογών αλγορίθμων τόσο σε επιτραπέζια παιχνίδια όσο και σε ΗΠ, με στόχο την επίτευξη υψηλών αποδόσεων.

2.6.2 Ντάμα-Checkers

Η δημιουργία του «Checkers Player» από τον Arthur Samuel's [98] το 1959 μπορεί να θεωρηθεί ως ο πρόδρομος όλων των αλγορίθμων ΜΜ στα επιτραπέζια παιχνίδια αφού χαρακτηρίστηκε μια από τις πρώτες και πιο σημαντικές εφαρμογές. Το πρωτοφανές για την τότε εποχή χαρακτηριστικό του «Checkers Player» ήταν η ικανότητά του να μαθαίνει από τα δικά του λάθη. Το πρόγραμμα συμμετείχε σε μια διαδικασία να ανταγωνίζεται τον ίδιο του τον εαυτό για να βελτιώσει την πολιτική του, επιτυγχάνοντας πολύ ψηλά επίπεδα απόδοσης σε

εξαιρετικά σύντομο χρονικό διάστημα (μεταξύ 8 με 10 ώρες παιχνιδιού) παρέχοντας του μόνο τους κανόνες του παιχνιδιού [98].

2.6.3 Σκάκι-Chess

Ένα από τα πρώτα παιχνίδια το οποίο θεωρήτω ως ένα αρκετά προκλητικό benchmarks είναι το σκάκι. Ο λόγος που είχε καθιερωθεί ως ένα από τα πιο προκλητικά περιβάλλοντα είναι αρχικά λόγω των ιδιοτήτων του ίδιου του παιχνιδιού αλλά και άλλων παραγόντων όπως για παράδειγμα:

1. Απλοί κανόνες οι οποίοι μπορούν εύκολα να μοντελοποιηθούν μαθηματικά
2. Ο συνδυασμός του αριθμού των πιθανών θέσεων και κινήσεων οδηγούν σε εκθετική αύξηση του αριθμού των πιθανών σεναρίων με αποτέλεσμα να δημιουργείτε μία τεράστια πολυπλοκότητα και πρόκληση για τους αλγόριθμους του ΤΝ.
3. Ο μεγάλος όγκος ψηλής ποιότητας διαθέσιμων δεδομένων για το σκάκι από πολλούς παίκτες grandmaster, μπορούσε να χρησιμοποιηθεί για την εκπαίδευση και αξιολόγηση αλγορίθμων ΤΝ.

Το 1997 διεξάχθηκε ένας αγώνας σκακιού μεταξύ του Deep Blue(υπολογιστής για σκάκι) της IBM και του τότε παγκόσμιου πρωταθλητή Kasparov [1] ο οποίος θεωρείτε ορόσημο για την ιστορία της τεχνητής νοημοσύνης. Ο συγκεκριμένος αγώνας αποτελείτο από έξι παρτίδες σκακιού και στον οποίο ο Deep Blue κέρδισε τον αγώνα. Το γεγονός αυτό απέδειξε για το τι είναι ικανοί να επιτύχουν οι αλγόριθμοι στο πεδίο της ΤΝ αλλά ταυτόχρονα πυροδότησε την ανάγκη για ανάπτυξη ακόμη πιο βέλτιστων και αποδοτικών αλγορίθμων σε πολύπλοκα επιτραπέζια παιχνίδια.

2.6.4 Atari Games

Το 2013 η εταιρεία DeepMind παρουσίασε για την τότε χρονική στιγμή τον πρωτοποριακό αλγόριθμό Deep Q-Network (DQN) [79] (Ο DQN παρουσιάστηκε και αναλύθηκε εκτενώς στο Κεφάλαιο 2.5.2). Ο αλγόριθμος DQN έδειξε ένα νέο τρόπο προσέγγισης στους πράκτορες ως προς το πώς να μαθαίνουν να παίζουν και να βελτιώνονται απευθείας από εισόδους ακατέργαστων pixel, χωρίς να χρειάζονται χειροκίνητες λειτουργίες ή γνώσεις για τον συγκεκριμένο τομέα.

Η εφαρμογή του DQN παρείχε ένα ευέλικτο και γενικεύσιμο πλαίσιο για την εκπαίδευση πρακτόρων οι οποίοι κατάφεραν να διαπρέψουν σε ένα ποικίλο σύνολο παιχνιδιών του Atari 2600(πχ. Breakout, Pong, Space Invaders) ξεπερνώντας τις υψηλότερες ανθρώπινες επιδόσεις. Αυτό έφερε στην επιφάνεια τις δυνατότητες της ΕΜ και των βαθιών νευρωνικών δικτύων στην αντιμετώπιση σύνθετων εργασιών.

Ο DQN της DeepMind πυροδότησε μία νέα εξελικτική πρόοδο στην κοινότητα της τεχνητής νοημοσύνης. Η παρουσίαση της νέας αυτής μεθόδου επέφερε τόσο άμεσες αλλά και ριζικές αλλαγές στην ΕΜ με επακόλουθο την εξοικείωση με πολύπλοκους τομείς που προηγουμένως θεωρείτο ακατόρθωτο.

2.6.5 Go (Ιαπωνέζικο Επιτραπέζιο)

Το Go είναι ένα επιτραπέζιο παιχνίδι που παίζεται μεταξυ δύο παικτών. Το GO αποτελείται από ένα ταμπλό($19x19$) και ο στόχος του παιχνιδιού είναι να καταλάβουμε όσο το δυνατό πιο μεγάλη περιοχή στο ταμπλό περιβάλλοντας τις πέτρες του αντιπάλου μας εντός αυτής. Κάποιος βλέποντας τους απλούς κανόνες που το διέπουν υποθέτει ότι είναι ένα απλό παιχνίδι. Στην πραγματικότητα είναι ένα πολύπλοκο και βαθυστόχαστο παιχνίδι με ένα τεράστιο χώρο αναζήτησης. Αυτά τα χαρακτηριστικά καθιστούν δύσκολη την αξιολόγηση των θέσεων και των κινήσεων στο ταμπλό [99]

Το AlphaGo έγινε ο πρώτος αλγόριθμος που νίκησε έναν παγκόσμιο πρωταθλητή στο παιχνίδι του Go [3]. Το AlphaGo προβλέπει όλες τις πιθανές τροχιές που μπορούν να προκύψουν χρησιμοποιώντας μία συγκεκριμένη διάταξη των πετρών στο ταμπλό. Η πρόβλεψή των τροχιών γίνεται μέσω του συνδυασμού ενός Monte Carlo tree search (MCTS) και βαθιών νευρωνικών δικτύων. Τα νευρωνικά δίκτυα τα οποία χρησιμοποιουνται εκπαιδεύονται με εποπτευόμενη μάθηση από παιχνίδια-κινήσεις έμπειρων παικτών και με ΕΜ παίζοντας με τον ίδιο τον εαυτό του [3]. Μέσω αυτού μπορούμε να πούμε ότι το AlphaGo γίνεται δάσκαλος του εαυτού του. Αφού γίνει η πρόβλεψη των τροχιών γίνεται αξιολόγηση της κάθε μίας και επιλέγεται αυτή που επιφέρει το μεγαλύτερο όφελος μακροπρόθεσμα.

Τον Οκτωβρίο του 2015 το πρόγραμμά AlphaGo νίκησε τον Ευρωπαίο πρωταθλητή του Go Fan Hui με 5 προς 0. Αυτή ήταν η πρώτη φορά που ένα πρόγραμμα υπολογιστή νίκησε έναν επαγγελματία παίκτη χωρίς να χάσει σε ούτε ένα παιχνίδι.

Το 2017 η Deepmind παρουσίασε την νέα έκδοση του AlphaGo , το AlphaGo Zero. Η νέα

αυτή έκδοση κατάφερε να επιφέρει πιο υψηλή απόδοση μέσω της διαδικασίας αυτοεκπαίδευσης. Το αποκορύφωμα αυτού του αλγορίθμου ήταν η νίκη του προς τον τότε παγκόσμιο πρωταθλητή Lee Sedol το 2017. Το AlphaZero, είναι ένα πιο γενικευμένο σύστημα που έχει επιτύχει επιδόσεις καλύτερες από τον άνθρωπο και στο σκάκι και το σόγκι.

2.6.6 Dota 2

Με την επιτυχή δημιουργία αλγορίθμων για παιχνίδια (πχ. σκάκι,Go) που αποδίδουν καλύτερα από τις κορυφαίες ανθρώπινες επιδόσεις και θεωρούνται πρόκληση για την ΕΜ ως πριν από λίγα χρόνια, η επιστημονική κοινότητα στράφηκε σε πιο περίπλοκα παιχνίδια. Το κύριο χαρακτηριστικό αυτών των παιχνιδίων είναι η αποτύπωση της αταξίας [2] και της συνεχής δυναμικής τους. Ένα τέτοιο παιχνίδι θεωρείται και το Dota 2. Το Dota 2 είναι ένα παιχνίδι στρατηγικής που παίζεται μεταξύ δύο ομάδων των πέντε παικτών, με κάθε παίκτη να ελέγχει έναν χαρακτήρα που ονομάζεται «ήρωας»

Θεωρείτε μία πρόκληση για τον τομέα της ΕΜ λόγω της φύσης του. Τα ζητήματα τα οποία το καθιστούν τόσο προκλητικό εκτός από τους περίπλοκους κανόνες του είναι [100]:

1. Long time horizons

   Σε αντίθεση με το σκάκι και το Go τα οποία συνήθως τελειώνουν μέσα σε 40 και 150 κινήσεις αντίστοιχα, το Dota 2 παράγει 30 frames per second για μέσο χρόνο 45 λεπτά το οποίο αυξάνει την πολυπλοκότητα. Επομένως λόγω του long time horizons ο πράκτορας θα πρέπει να λαμβάνει υπόψη όχι μόνο τις άμεσες ανταμοιβές αλλά και τις πιθανές ανταμοιβές στο μακρινό μέλλον.

2. Partially-observed state

   Ο κάθε παίκτης είναι ικανός να παρατηρεί το τι συμβαίνει στο περιβάλλον μόνο για μία συγκεκριμένη περιοχή γύρω του. Ο υπόλοιπος χάρτης καλύπτεται από ομίχλη κρύβοντας τους εχθρούς και τις στρατηγικές τους. Επομένως η αλγόριθμοι που θα εφαρμοστούν στο παιχνίδι θα χρειαστεί να εξαγάγουν συμπεράσματα με βάση ελλιπή δεδομένα. Αντίθετα το σκάκι όσο και το Go είναι παιχνίδια πλήρους πληροφόρησης.

3. High-dimensional, continuous action space

   Στο Dota 2, κάθε ήρωας έχει την δυνατότητα να εκτελέσει διάφορες ενέργειες που

[2] Ένα σύστυμα-περιβάλλον είναι άτακτο όταν υπάρχουν πολλοί τρόποι με τους οποίους μπορούν να ανακατανεμηθούν τα συστατικά που το απαρτίζουν σε διαφορετικές μικροκαταστάσεις χωρίς να αλλάζουν σημαντικά οι ιδιότητες του.

στοχεύουν είτε σε άλλο ήρωα είτε σε μια γεωγραφική τοποθεσία στο χάρτη. Η OpenAi έχει καταγράψει όλες τις πιθανές ενέργειες του κάθε ήρωα σε ένα διακριτό σύνολο, που σημαίνει ότι ο χώρος ενεργειών χωρίζεται σε ένα πεπερασμένο σύνολο διακριτών ενεργειών. Υπάρχουν περίπου 170.000 πιθανές ενέργειες για κάθε ήρωα, ωστόσο δεν είναι όλες οι πιθανές ενέργειες δυνατό να επιλεχθούν σε οποιαδήποτε δεδομένη. Έτσι, στην πράξη, υπάρχουν περίπου 1.000 έγκυρες ενέργειες από τις οποίες μπορεί να επιλέξει ένας ήρωας κατά τη διάρκεια κάθε παιχνιδιού. Ο μέσος αριθμός ενεργειών στο σκάκι είναι 35 και στο Go 250.

4. High-dimensional, continuous observation space

   Η πολυπλοκότητα και η ποσότητα πληροφοριών αναπαράστασης μεταξύ του Dota 2 και των παραδοσιακών επιτραπέζιων παιχνιδιών όπως το σκάκι και το Go διαφέρει σε πολύ μεγάλο βαθμό. Το κάθε στιγμιότυπο στο σκάκι αντιπροσωπεύεται μέσα από 70 τιμές (διαστάσεις σκακιέρας 8x8,τα 6 είδη πιονιών σε κάποια από αυτά, χρώμα τεμαχίου) ενώ ένα στιγμιότυπο Go χρειάζεται περίπου 400 τιμές για να αναπαρασταθεί (ένας πίνακας 19x19, 2 τύποι πετρών). Για την αναπαράσταση ενός στιγμιότυπου του Dota 2 χρειάζεται περίπου 20.000 τιμές (χάρτης, δέκα ήρωες, κτήρια, NPC non-player character κτλ).

Ο οργανισμός OpenAI για να αντιμετωπίσει αυτές τις προκλήσεις του Dota 2, ξεκίνησε μία νέα ερευνητική ομάδα την OpenAI Five. Ο στόχος της ήταν να αναπτυχθούν πράκτορες τεχνητής νοημοσύνης ικανοί να ανταγωνίζονται τους καλύτερους επαγγελματίες παίκτες στο παιχνίδι. Η OpenAI χρησιμοποίησε την ΕΜ ως βασικό εργαλείο για την εκπαίδευση των πρακτόρων της στο περιβάλλον Dota 2.

Το έργο Dota 2 του OpenAI σημείωσε αξιοσημείωτη επιτυχία αρχίζοντας από το 2017 και δημιούργησαν ένα πράκτορα ο οποίος κατάφερε να κερδίζει τους κορυφαίους επαγγελματίες του κόσμου σε αγώνες ένα εναντίον ενός του Dota 2 [101]. Ο πράκτορας αρχικά εκπαιδεύτηκε παίζοντας μόνος του στο περιβάλλον. Μέσα από τα αποτέλεσμα αποδείχθηκε ότι η ατομική εκπαίδευση(διάστημα ενός μηνός) του πράκτορα(παίζοντας μόνος του) εκτόξευσε την απόδοση του σε επίπεδα πολύ πιο ψηλά απο εκείνα των κορυφαίων παικτών [102]. Συνεχίζοντας η OpenAI Five κατάφερε μέσω μίας ομάδας πέντε νευρωνικών δικτύων να νικάει ερασιτεχνικές ανθρώπινες ομάδες πέντε ατόμων στο Dota 2 [103]. Το 2018 ο αλγόριθμος της OpenAI Five κέρδισε μια ομάδα πέντε επαγγελματιών παικτών Dota 2 μπροστά σε ζωντανό κοινό

των 100.000 τηλεθεατών [104]. Τέλος το 2019 η OpenAI Five κέρδισε τους παγκόσμιους πρωταθλητές και συγκεκριμένα κερδίζοντας δύο παιχνίδια συνεχόμενα [2].

2.6.7 Obstacle Tower

Όπως αναφέρθηκε και προηγουμένως τις τελευταίες δεκαετίες οι αλγόριθμοι στο πεδίο της ΕΜ έχουν αρχίσει να εμφανίζουν εντυπωσιακές επιδόσεις ακόμα καλύτερες και από αυτές που επιτυγχάνουν οι άνθρωποι αλλά σε συγκεκριμένα προβλήματα ο κάθε ένας. Αυτό έχει οδηγήσει την κοινότητα της τεχνητής νοημοσύνης στην αναζήτηση εύρεσης νέων περιβάλλοντων τα οποία θα μπορούν να αποτυπώσουν καλύτερα την πολυπλοκότητα και την ποικιλομορφία σεναρίων που μοιάζουν πολύ με προκλήσεις του πραγματικού κόσμου(πχ πλοήγηση, λήψη αποφάσεων, στρατηγική και η αλληλεπίδραση με δυναμικά και απρόβλεπτα περιβάλλοντα) για σύγκριση και αξιολόγηση κυρίως της γενίκευσης των αλγορίθμων της ΕΜ.

Τον Ιούλιο του 2019 η Unity παρουσιάζει το βιντεοπαιχνίδι Obstacle Tower που σχεδιάστηκε ειδικά ως ερευνητική πλατφόρμα για την ΤΝ. Ο απώτερος στόχος του Obstacle Tower είναι η δοκιμή και η αξιολόγηση των δυνατοτήτων των πρακτόρων ΤΝ να επιδεικνύουν δεξιότητες επίλυσης πολύπλοκων προβλημάτων/περιβαλλόντων, να προσαρμόζονται σε μεταβαλλόμενα περιβάλλοντα και να επιδεικνύουν δυνατότητες γενίκευσης. Το Obstacle Tower είναι ένα περίπλοκο, δυναμικό τρισδιάστατο περιβάλλον βιντεοπαιχνιδιού που αποτελείται από 100 ορόφους, όπου ο κάθε όροφος παρουσιάζει ένα ευρύ φάσμα προκλήσεων (πχ. πλοήγηση σε πλατφόρμες με διαφορετικά ύψη, κινούμενες πλατφόρμες, στενά περάσματα και εμπόδια που απαιτούν ακριβή άλματα, αποφυγή εχθρών κτλ.) [6].

Σε κάθε επεισόδιο παρατηρείτε αλλαγή στην διάταξη των ορόφων του Obstacle Tower, διασφαλίζοντας έτσι την ικανότητα προσαρμογής και την καλύτερη απόδοση του πράκτορα.Η πολυπλοκότητα και η μεταβλητότητα του περιβάλλοντος Obstacle Tower το καθιστούν ιδανικό εργαλείο δοκιμών για την αξιολόγηση της προσαρμοστικότητας και των δυνατοτήτων γενίκευσης των αλγόριθμων ΕΜ. Αυτό το τυποποιημένο και προκλητικό περιβάλλον επιτρέπει δίκαιες συγκρίσεις και αποκαλύπτει πληροφορίες σχετικά με τα δυνατά και τα αδύνατα σημεία των διαφόρων αλγόριθμων [105] [106] [107] .

Το Obstacle Tower έχει δύο ρυθμίσεις συνάρτησης ανταμοιβής: sparse και dense. Στην ρύθμιση sparse, παρέχεται μια θετική ανταμοιβή +1 μόνο όταν ο πράκτορας ολοκληρώσει έναν όροφο του πύργου. Στην ρύθμιση dense εκτός από την +1 ανταμοιβή που λαμβάνει ο πράκτορας με την ολοκλήρωση του ορόφου λαμβάνει επίσης μία θετική ανταμοιβή +0.1 σε τρις περιπτώσεις:

1. Όταν πάει αρκετά κοντά στην πόρτα και αυτή ανοίξει
2. Όταν επίλυση ένα puzzle
3. Όταν πάρει ένα κλειδί.

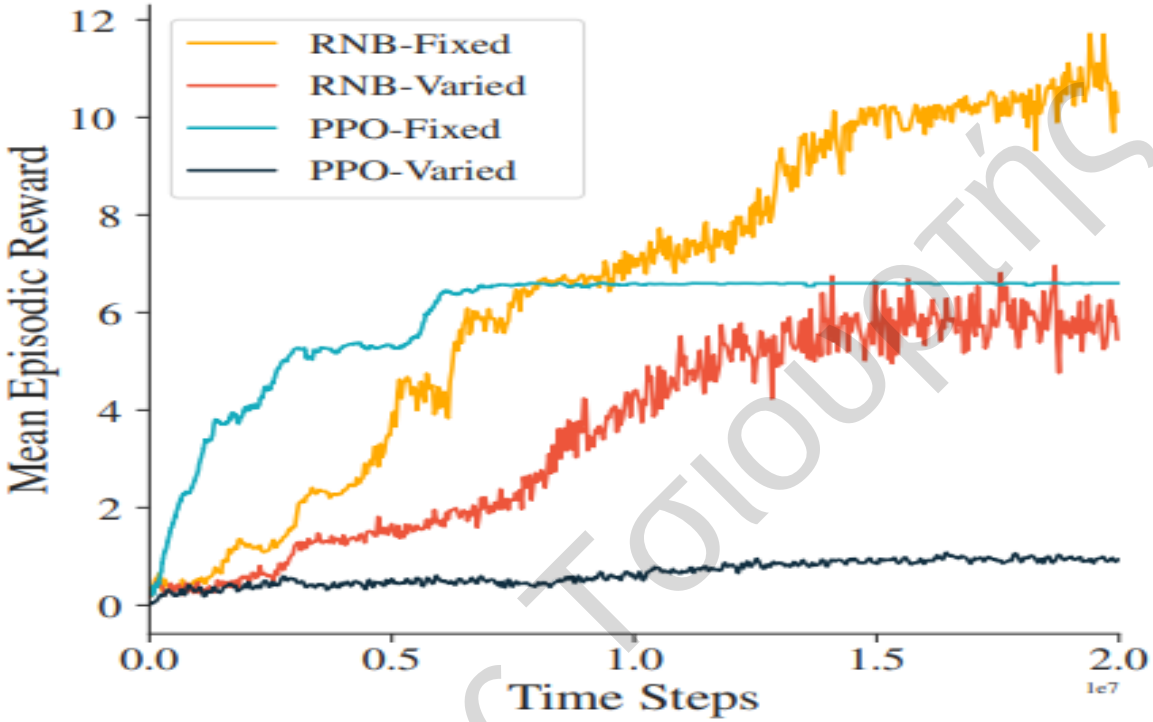
Δεδομένου του ότι ο πράκτορας λαμβάνει επιβράβευση σε πολύ μεμονωμένες περιπτώσεις λόγο της φύσεις του παιχνιδιού, το άρθρο Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning [6] προτείνει τους ερευνητές να ενσωματώσουν επιπλέον μηχανισμούς για την ενίσχυση του σήματος ανταμοιβής.

Στο ίδιο άρθρο με την παρουσίαση του παιχνιδιού Obstacle Tower είχε γίνει και εφαρμογή και αξιολόγηση των αποτελεσμάτων των τότε πιο σύγχρονων αλγορίθμων ΒΕΜ. Συγκεκριμένα, εφαρμόστηκαν οι αλγόριθμοι Proximal Policy Optimization (PPO) [108] καθώς ο αλγόριθμος Rainbow [87].

Για κάθε έναν από τους δύο παραπάνω αλγορίθμους, εκτελέστηκαν δύο διαφορετικά πειράματα. Αρχικά, εφάρμοσαν τον κάθε αλγόριθμο με σταθερή τιμή στην παράμετρο "Tower-seed", με αποτέλεσμα ο πράκτορας σε κάθε επεισόδιο να αλληλεπιδρά με την ίδια διαμόρφωση του χώρου. Στη δεύτερη πειραματική διάταξη, τέθηκε η τιμή της παραμέτρου "Tower-seed" σε τυχαία τιμή, με αποτέλεσμα ο πράκτορας σε κάθε νέο επεισόδιο να αλληλεπιδρά με μια διαφορετική διαμόρφωση του χώρου. Επίσης, κατά τη διάρκεια των πειραμάτων, χρησιμοποίησαν την παράμετρο "dense reward", καθώς η χρήση της "sparse dense" δεν οδήγησε σε αξιόλογα αποτελέσματα. Οι υπερπαράμετροι που εφάρμοσαν ήταν οι ίδιες με αυτές που χρησιμοποίησαν οι αλγόριθμοι για τα "Atari benchmarks".

Και οι δύο αλγόριθμοι εκπαιδεύτηκαν στο περιβάλλον για 20 εκατομμύρια βήματα. Στο Σχήμα 2.36 παρουσιάζονται τα γραφήματα της μέσης ανταμοιβής κατά την εκπαίδευση των δύο αλγορίθμων τόσο για το σταθερό περιβάλλον (Fixed Environment) όσο και για το μεταβαλλόμενο περιβάλλον (Varied Environment). Από την ανάλυση των αποτελεσμάτων προκύπτει ότι οι πράκτορες, και στις δύο πειραματικές διατάξεις, δεν ξεπερνούν τους 10 ορόφους, με

τις επιδόσεις τους σε σταθερό περιβάλλον να είναι αισθητά ανώτερες. Αυτό είναι αναμενόμενο, καθώς οι πράκτορες στη σταθερή συνθήκη πρέπει να μάθουν να επιλύουν μόνο μία συγκεκριμένη διαμόρφωση του χώρου. Τέλος, ο αλγόριθμος Rainbow επιτυγχάνει καλύτερα αποτελέσματα και στα δύο σενάρια σε σχέση με τον PPO αλγόριθμο.



Σχήμα 2.36: Μέσες ανταμοιβές ανά επεισόδιο κατά τη διάρκεια της εκπαίδευσης των δυο πρακτόρων, χρησιμοποιώντας τους αλγορίθμους OpenAI Baseline PPO (PPO) και Dopamine Rainbow (RNB) σε σταθερό και μεταβαλλόμενο περιβάλλον εκπαίδευσης. [6].

Challenge Winners			
Place Name	Username	Average floors	Average reward
1st Alex Nichol	unixpickle	19.4	35.86
2nd Compscience.org	giadefa	16	28.7
3rd Songbin Choi	sungbinchoi	13.2	23.2

Πίνακας 2.1: Πίνακας με τους τρεις πρώτους νικητές του Obstacle Tower μαζί με τον μέσο όρο ορόφων και μέση ανταμοιβή [7].

Μέχρι στιγμής, οι καλύτεροι αλγόριθμοι που έχουν εφαρμοστεί στο περιβάλλον Obstacle Tower έχουν επιτύχει να επιλύσουν πάνω από 10 ορόφους, με τον καλύτερο αλγόριθμο να επιλύει κατά μέσο όρο 20 ορόφους, αποδοση που είναι παρόμοια με την απόδοση έμπειρων ανθρώπων [7].

Στον πίνακα 2.1 παρουσιάζονται οι αποδόσεις των τριών αλγορίθμων, οι οποίοι επέτυχαν τις υψηλότερες επιδόσεις στο παιχνίδι.

Σε αυτό το σημείο θα αναλυθεί ο τρόπος με τον οποίο αντιμετώπισε την πρόκληση του Obstacle Tower ο νικητής του διαγωνισμού, ο Αλεχ.

Ο Άλεξ αρχικά προσπάθησε να εφαρμόσει κάποιες θεμελειώδεις αλγορίθμους βελτιστοποίησης ΕΜ για την εκπαίδευσης μίας πολιτικής όπως

- PPO με εικόνες εισόδου (5x5) [108]
- CMA-ES(Covarinace matrix adaptation evalution trategy) [20]
- CEM(Cross-entropy method) [109]

Ωστόσο καμία απο αυτές δεν ξεπερνούσε τον 5ο όροφο. Ωστόσο μέσα από αυτές τις δοκιμές συνειδητοποίησε ότι ο PPO αλγόριθμος παρείχε τα καλύτερα αποτελέσματα θέτοντας τον ως την βασική επιλογή του. Εδω πρέπει να αναφερθεί ότι το μεγαλύτερο πρόβλημα που θεωρούσε αρχικά ο Άλεξ ότι θα αντιμετώπιζε θα ήταν η γενίκευση του αλγορίθμου. Αυτή η εντύπωση του καταρήφθηκε κατα την προσπάθεια του να συντονίσει(tuning) τον PPO, αφού συνηδητοποίησε ότι με το να θέσει 100 διαφορετικά seeds η γενίκευση είχε επιτευχθεί.

Η πρώτη υλοποίηση του Άλεξ βασίστηκε σε μία παραλλαγή του PPO στο anyrl-py [110] και κατάφερε να φτάσει μέχρι τον 10 όροφο. Σε αυτό το σημείο αντιμετώπιζε ένα πρόβλημα Exploration αφού ο agent δεν κατάφερνε να επιλύσει το Sokoban puzzle το οποίο πρωτοεμφανιζόταν στον 10ο όροφο. Σε αυτό το σημείο αντιλήφθηκε ότι οι παραδοσιακές μεθόδοι(πχ. curiosity-driven Exploration, Go-explore) δεν θα επέφεραν αποτέλεσμα αφού είχε εφαρμο-

στέι μόνο σε δισδιάστατα περιβάλλοντα και όχι τόσο προκλητικά περιβάλλοντα. Έτσι θεώρησε ότι υπήρχαν δύο πιθανοί τρόποι αντιμετώπισης αυτού του προβλήματος:

- Evolutionary algorithms (Explore in aprameter space rather than algorithm space)
- Human demostration(HD)

Το ότι θα επέλεγε τοHD ήταν αναπόφευκτα λόγω έλλειψης υπολογιστή ισχύος(1 Workplace-> 1 GPU). Ακολούθησε με το να καταγράψει τον ίδιο τον εατό του να παίζει έτσι ώστε μέσω supervised learning να προσαρμόσει την πολιτική του πράκτορα του. Δοκιμάζοντας τον πράκτορα του μετά από αυτη την εκπαίδευση πάλι δεν ήταν εφικτό να επιλύσει το Sokoban puzzle. Τότε δοκίμασε κάποιες άλλες προσεγγίσεις όπως imitation learning alorithms(eg. GAIL) αλλά ούτε αυτοί κατάφεραν να επιλύσουν το Sokoban puzzle.

Σε αυτό το σημείο συνειδητοποίησε ότι το πρόβλημα ήταν θέμα μνήμης και το αντιμετώπισε μέσω χρήσης ενός state representation το οποίο θα μπορούσε να πάει πίσω μέχρι και 50 timesteps, σε αντίθεση με τα RNN τα οποία παίρνουν τεράστιο αριθμό δειγμάτων αλλά εμπεριέχουν και μεγάλο πλήθος αχρίαστων πληροφοριών. Τότε ο agent άρχισε να σπρώχνει το κουτί χωρίς όμως να ολοκληρώνει το όλο σενάριο του ορόφου. Εδω ο Άλεξ εισήγαγε ένα classifier για την αναγνώριση των αντικειμένων (κουτιά, πόρτες κ.λπ.). απο τον agent και ενσωμάτωσε το output του classifier στο state representation. Με αυτόν τον τρόπο κατάφερε να επιλύσει το Sukoban Puzzle.

Αρχίζοντας απο τον 1ο όροφο όμως όταν έφτανε στον 10ο, ο πράκτορας ξεχνούσε πώς να σπρώξει το κουτί και αυτό οφείλετε στο entropy bonus, το οποίο ενθαρύνει τον πράκτορα να πάρει τυχαίες κινήσεις όσο το δυνατόν περισσότερο και συνεπώς έτσι καταστρέφει την pre-trained συμπεριφορά που είχε μάθει προηγουμένως. Τέλος πρόσθεσε το prierarch το οποίο είναι εναλλακτική προσέγγιση για την ιεραρχία στην ενισχυμένη μάθηση, χωρίς την ανάγκη να καθοριστούν αυστηρά όρια δράσης ή διακριτικά επίπεδα ιεραρχίας [111]. Συνεπώς βάση όλων των πιο πάνω ο Άλεχ στην λύση του είχε 2 διαφορετικούς πράκτορες. Ο πρώτος έλυσε τους ορόφους 1-9 και ο δεύτερος έλυσε τους ορόφους 10 και μετά. Ο δεύτερο πράκτορας εκπαιδεύτηκε με αρχικούς ορόφους που δειγματολήφθηκαν τυχαία μεταξύ 10 και 15. Αυτό τον ανάγκασε να μάθει να λύνει αμέσως το παζλ του Sokoban, αντί να τελειοποιεί πρώτα τους ορόφους 1-9. Εφορμώντας όλα τα πιο πάνω κατάφερε να φτάσει κατά μέσω όρο 19.4 ορόφους και με μέση ανταμοιβή 35,86.

Κεφάλαιο 3

Μεθοδολογία (Methodology)

3.1 Εισαγωγή

Το παρόν κεφάλαιο αποσκοπεί στην εκτενή αξιολόγηση της αποδοτικότητας του αλγορίθμου DreamerV3 στο απαιτητικό περιβάλλον του παιχνιδιού Obstacle Tower. Μέσα από τη διεξαγωγή πειραματικών διατάξεων, προσπαθούμε να κατανοήσουμε την επίδραση των παραμέτρων τόσο του ίδιου του αλγορίθμου Dreamerv3, όπως το Training Ratio και το Model Size, όσο και του παιχνιδιού Obstacle Tower, όπως το Tower Seed, το Dense Reward κ.ά.

Αρχικά, παρουσιάζουμε τις βασικές παραμέτρους και χαρακτηριστικά του DreamerV3, καθώς και του παιχνιδιού Obstacle Tower. Στη συνέχεια, εξετάζουμε διάφορες παραμετρικές διαμορφώσεις μέσω πειραματικών σεναρίων, αναδεικνύοντας την επίδραση του Training Ratio και του Model Size στη διαδικασία εκπαίδευσης του DreamerV3. Μέσα από λεπτομερή ανάλυση αυτών των πειραματικών διατάξεων, αναδεικνύουμε τις πιθανές αλληλεπιδράσεις μεταξύ των παραμέτρων, καθώς και τη σημασία της σωστής επιλογής παραμετρικών τιμών για την επίτευξη υψηλής απόδοσης του μοντέλου στο πλαίσιο του Obstacle Tower. Επιπλέον, διερευνούμε τον τρόπο με τον οποίο οι αλλαγές στις παράμετρες του Obstacle Tower επηρεάζουν την απόδοση του DreamerV3.

3.2 Παραμέτροι Αλγορίθμοι DreamerV3

Οι δύο παράμετροι οι οποίες μεταβάλλονται και επηρεάζουν άμεσα την εκπαιδευτική απόδοση του αλγόριθμου DreamerV3 είναι:

- **Training ratio-Αναλογία εκπαίδευσης**

  Η αναλογία εκπαίδευσης(ΑΕ) αναφέρεται στην αναλογία μεταξύ του αριθμού των βημάτων που έκανε ο πράκτορας στο περιβάλλον κατά τη διάρκεια της εκπαίδευσης πριν από την ενημέρωση (update) που έκανε για την δυναμική του μοντέλου. Πιο συγκεκριμένα ο αλγόριθμος DreamerV3 αλληλεπιδρά επαναληπτικά με το περιβάλλον συλλέγοντας δεδομένα και μέσω αυτών εκπαιδεύει-ενημερώνει το μοντέλο του για την δυναμική του περιβάλλοντος. Η αναλογία εκπαίδευσης καθορίζει μετά από πόσα βήματα του πράκτορα μέσα στο περιβάλλον γίνεται η ενημέρωση του μοντέλου. Αυτή η αναλογία ελέγχει την ισορροπία μεταξύ εξερεύνησης (περιβαλλοντική αλληλεπίδραση) και μάθησης (ενημέρωση μοντέλου) στη διαδικασία εκπαίδευσης. Συγκεκριμένα στο άρθρο [4] έχουν χρησιμοποιηθεί 7 διαφορετικές αναλογίες εκπαίδευσης (1 , 2, 4, 8, 16, 32, 64).

- **Model size- Μέγεθος μοντέλου**.

  Ο όρος μέγεθος του μοντέλου (ΜΜ) αναφαίρετε κυρίως στον αριθμό των παραμέτρων (βάρη (weight), bias) που αποτελούν ένα νευρωνικό δίκτυο (ΝΔ). Οι παράμετροι είναι τιμές οι οποίες μαθαίνονται από το μοντέλο μέσω της διαδικασίας εκπαίδευσης με απώτερο στόχο την προσέγγιση της επιθυμητής συνάρτησης ή πολιτικής. Ο αριθμός των παραμέτρων μπορεί να ποικίλλει σημαντικά και εξαρτάται κατά κόρων στην πολυπλοκότητα του προβλήματος στην αρχιτεκτονική του ΝΔ(αριθμός κρυφών επιπέδων(Hidden layer, αριθμός νευρώνων σε κάθε κρυφό επίπεδο,) και το είδος του ΝΔ (PerceptronConvolutional neural network,Multilayer perceptron,Recurrent neural network κτλ.). Μεγαλύτερα και πιο σύνθετα προβλήματα απαιτούν συχνά μεγαλύτερα μοντέλα με μεγαλύτερο αριθμό παραμέτρων για την αποτελεσματική καταγραφή σύνθετων μοτίβων και αναπαραστάσεις δεδομένων με την προϋπόθεση ότι υποστηρίζονται από υπολογιστικούς πόρους. Εδώ πρέπει να σημειωθεί ότι τα μεγαλύτερα μοντέλα από μόνα τους δεν είναι επαρκής για την καλή απόδοση δύσκολων προβλημάτων, αλλά πρέπει να συνδυάζονται και από καλής ποιότητας δεδομένα εισόδου-εκπαίδευσης, κατάλληλο αλγόριθμο μάθησης(Backprobagation, Adam, AdaBoost κτλ.), κατάλλη-

λες υπερπαραμέτρους, ισορροπία μεταξύ εξερεύνησης(exploration) και εκμετάλλευσης(exploitation).

Στον αλγόριθμο DreamerV3 το ΜΜ μπορεί να διαφέρει ανάλογα με τη συγκεκριμένη υλοποίηση και την πολυπλοκότητα του προβλήματος που αντιμετωπίζεται. Όπως διακρίνεται στο Σχήμα3.1, από το άρθρο [4] έχουν χρησιμοποιηθεί 4 διαφορετικά ΜΜ (XS, S, M, L, XL) τα οποία αποτελούνται από 4 NN(GRU recurrent units,CNN multiplier, Dense hidden units, MLP layers) με το κάθε μέγεθος να διαφέρει βάση των παραμέτρων που περιέχουν τα NN τους. Μέσα από τα διεξαγωγή των πειραμάτων στο άρθρο [4] έχει αποδειχθεί ότι τα μεγαλύτερα μοντέλα επιτυγχάνουν όχι μόνο υψηλότερη τελική απόδοση αλλά και υψηλότερη απόδοση δεδομένων.

Dimension	**XS**	**S**	**M**	**L**	**XL**
GRU recurrent units	256	512	1024	2048	4096
CNN multiplier	24	32	48	64	96
Dense hidden units	256	512	640	768	1024
MLP layers	1	2	3	4	5
Parameters	8M	18M	37M	77M	200M

Σχήμα 3.1: Πίνακας απεικόνισης της αρχιτεκτονικής του κάθε Model Size που παρέχεται απο τον DreamerV3. Digital image, 10 Jan 2023, [4]

Οι δύο πιο πάνω παράγοντες (ΜΜ, AE) δεν αναφέρεται μέσα στο άρθρο [4] να συσχετίζονται άμεσα, αλλά και οι δύο επηρεάζουν τη διαδικασία μάθησης και την απόδοση. Η επιλογή του μεγέθους του μοντέλου και της αναλογίας εκπαίδευσης στο DreamerV3 βασίζεται τυπικά σε εμπειρικούς πειραματισμούς και σε συγκεκριμένους τομείς. Το μέγεθος του μοντέλου επηρεάζει κυρίως την πολυπλοκότητα και την ικανότητα μάθησης του νευρωνικού δικτύου που χρησιμοποιεί ο DreamerV3, ενώ η αναλογία εκπαίδευσης μπορεί να επηρεάσει την αντιστάθμιση εξερεύνησης-εκμετάλλευσης και την ταχύτητα σύγκλισης της μαθησιακής διαδικασίας.

3.3 Παράμετροι Παιχνιδίου Obstacle Tower

Στο πιο κατω πίνακα3.1 παρουσιάζονται όλες οι διαθέσιμες παράμετροι του Obstacle Tower προκειμένου να προσαρμοστεί η δυσκολία και το περιεχόμενο του περιβάλλοντος του.

Parameter	**Value range**	**Effect**
train-mode	(0-**1**)	Whether to run the environment in inference (0) or training (1) mode.
tower-seed	(**-1** - 99999)	Sets the seed used to generate the tower. -1 corresponds to a random tower on every reset() call.
starting-floor	(**0**,99)	Sets the starting floor for the agent on reset().
total-floors	(1,**100**)	Sets the maximum number of possible floors in the tower.
dense-reward	(0,**1**)	Whether to use the sparse (0) or dense (1) reward function.
lighting-type	(0,**1**,2)	Whether to use no realtime light (0), a single realtime light with minimal color variations (1), or a realtime light with large color variations (2).
visual-theme	(0,**1**,2)	Whether to use only the default-theme (0), the normal ordering or themes (1), or a random theme every floor (2).
agent-perspective	(0,**1**)	Whether to use first-person (0) or third-person (1) perspective for the agent.
allowed-rooms	(0,1,**2**)	Whether to use only normal rooms (0), normal and key rooms (1), or normal, key, and puzzle rooms (2).
allowed-modules	(0,1,**2**)	Whether to fill rooms with no modules (0), only easy modules (1), or the full range of modules (2).
allowed-floors	(0,1,**2**)	Whether to include only straightforward floor layouts (0), layouts that include branching (1), or layouts that include branching and circling (2).
default-theme	(**0**,1,2,3,4)	Whether to set the default theme to Ancient (0), Moorish (1), Industrial (2), Modern (3), or Future (4).

Πίνακας 3.1: Λίστα Παραμέτρων, μαζί με το εύρος τιμών και σε τι αντιστοιχούν. Οι προεπιλεγμένες Τιμές επισημαίνονται με έντονη γραφή [112].

3.4 Πειραματικές Διατάξεις

3.4.1 Πειραματικές Διατάξεις Τύπου Α

Όλες οι πιο κάτω πειραματικές διατάξεις εκτελέστηκαν με ρυθμό ανανέωσης 20 Hz (20 αλλαγές ή κινήσεις στο παιχνίδι κατά μέσο όρο κάθε δευτερόλεπτο). Αυτή η αύξηση του ρυθμού επιλέχθηκε κυρίως για την επιταχυνση της εκπαίδευσης του πράκτορα μας.

Πειραματική Διάταξη Α.1

Η διεξαγωγή των πειραμάτων Α.1.1, Α.1.2, Α.1.3 έχουν διπλό στόχο. Αρχικά την επαλήθευση του γεγονότος ότι τα μεγαλύτερα μοντέλα επιτυγχάνουν υψηλότερη τελική απόδοση (higher final performance) [1]αλλά και υψηλότερη απόδοση δεδομένων (Higher data-efficiency)[2]και κατά δεύτερον ότι η υψηλότερη αναλογία εκπαίδευσης επιφέρει σημαντικά βελτιωμένη απόδοση δεδομένων. Για την διερεύνηση των πιο πάνω οι παράμετροι του παιχνιδιού Obstacle Tower θα τεθούν ως οι προεπιλεγμένες τιμές (Πίνακας 3.1), ενώ όσον αφορά τις παραμέτρους του αλγόριθμου DreamerV3 (Training Ratio, Model size) θα μεταβάλλονται όπως διακρίνεται στις αντίστοιχες στήλες για τα πειράματα 1.1, 1.2, 1.3 του Πίνακα 3.2.

Ο λόγος που επιλέχθηκαν οι συγκεκριμένες τιμές για τις παραμέτρους του αλγόριθμου DreamerV3 πηγάζουν από την Σχήμα 3.2 του άρθρου [4].

Για την παράμετρο model size διακρίνεται ξεκάθαρα ότι το XS μέγεθος μοντέλου πάντοτε προσφέρει την μικρότερη απόδοση σε αντίθεση με τα μεγέθη L, XL τα οποία προσφέρουν την μεγαλύτερη απόδοση. Αν και τις πλείστες φορές το XL μοντέλο μεγέθους επιφέρει υψηλότερη απόδοση από το L μοντέλο μεγέθους, στο παιχνίδι Breakout το L μοντέλο μεγέθους καταφέρνει να ξεπεράσει την απόδοση του XL μοντέλου απόδοσης.

Βάσει των παραπάνω συγκεκριμένων πειραμάτων, προσδοκούμε να επιβεβαιώσουμε ότι τα μικρότερα μοντέλα παρουσιάζουν χαμηλότερη απόδοση δεδομένων και χαμηλότερη τελική τιμή. Επιπλέον, σχετικά με το μοντέλο XL μεγέθους, θα αξιολογήσουμε εάν επιτυγχάνει

[1]Η υψηλότερη τελική απόδοση-higher final performance: Αναφέρεται στη συνολική αποτελεσματικότητα ή ακρίβεια ενός αλγορίθμου εκμάθησης αφού έχει εκπαιδευτεί σε επαρκή ποσότητα δεδομένων. Μέσω μίας υψηλότερη τελικής απόδοσης υποδηλώνει ότι ο αλγόριθμος επιτυγχάνει καλύτερα αποτελέσματα ή μεγαλύτερη ακρίβεια σε ένα δεδομένο πρόβλημα.

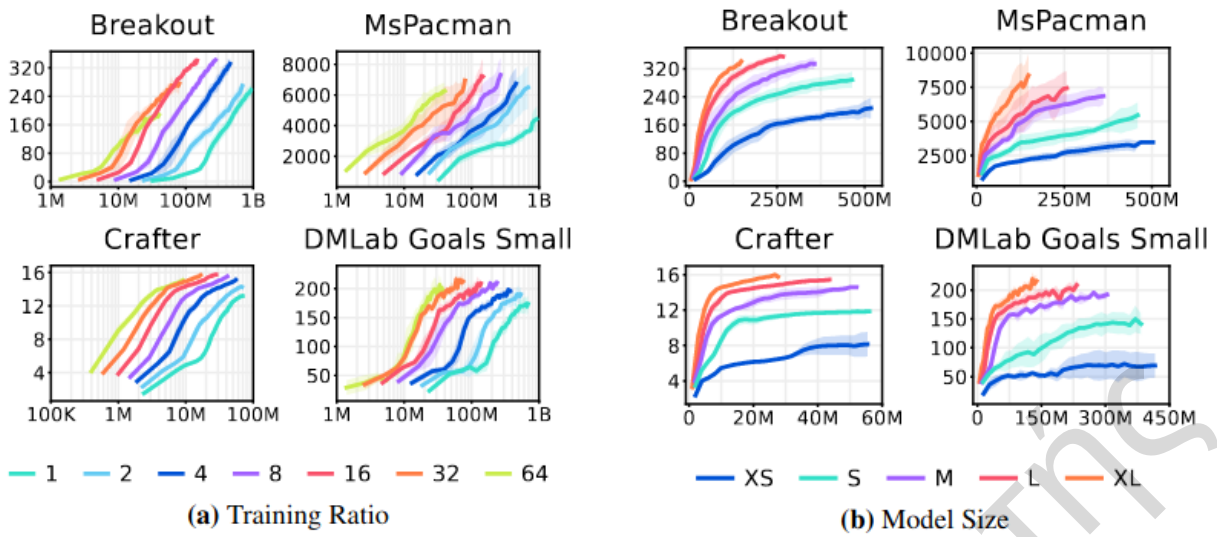
[2]Η υψηλότερη αποδοτικότητα δεδομένων αναφέρεται στην ικανότητα ενός αλγορίθμου μάθησης να επιτυγχάνει καλή απόδοση με μικρότερο όγκο δεδομένων εκπαίδευσης στην περίπτωση του DreamerV3 λιγότερα βήματα αλληλεπίδρασης στο περιβάλλον. Η υψηλότερη απόδοση δεδομένων σημαίνει ότι ο αλγόριθμος απαιτεί λιγότερα παραδείγματα εκπαίδευσης ή αλληλεπιδράσεις με το περιβάλλον για να φτάσει σε ένα ικανοποιητικό επίπεδο απόδοσης. Συνοψίζοντας η υψηλότερη απόδοση δεδομένων υποδεικνύει ότι ο αλγόριθμος μπορεί να μάθει αποτελεσματικά και να γενικεύσει καλά από έναν περιορισμένο όγκο δεδομένων.

Experiment .1				
Dreamer V3 Parameters		Experiment .1.1	Experiment .1.2	Experiment .1.3
Model Size		S-M-XL	S-M-XL	S-M-XL
Training Ratio		16	32	64
Obstacle Tower Parameters				
Train-mode		1	1	1
Tower-seed		-1	-1	-1
Starting-floor		0	0	0
Total-floors		100	100	100
Dense-reward		1	1	1
Lighting-type		1	1	1
Visual-theme		1	1	1
Agent-perspective		0	0	0
Allowed-rooms		2	2	2
Allowed-modules		2	2	2
Allowed-floors		2	2	2

Πίνακας 3.2: Λίστα παραμέτρων που επιλέχθηκαν για την πειραματική διάταξη A.1.1, A.1.2, A.1.3 [112]

πάντοτε υψηλότερη τιμή και υψηλότερη απόδοση δεδομένων σε σύγκριση με τα υπόλοιπα μεγέθη.

Κάθε πείραμα θα διεξαχθεί με διαφορετικές τιμές Training Ratio , με σκοπό να εξεταστεί εάν με την αύξηση της τιμής του Training Ratio αυξάνεται ο αριθμός των βημάτων-αλληλεπιδράσεων που απαιτούνται για την εμφάνιση αύξησης στο σκορ, δηλαδή εάν υπάρχει μείωση στην αποδοτικότητα δεδομένων καθώς αυξάνεται το Training Ratio. Ταυτόχρονα, προσδοκάται να επιβεβαιωθεί εάν το Training Ratio=16 φαίνεται να επιφέρει, σε ένα ενδιάμεσο χρονικό διάστημα, μία από τις υψηλότερες αποδόσεις.



Σχήμα 3.2: Τα Σχήματα στα δεξιά (a) Training Ratio παρουσιάζουν την απόδοση που προέκυψε σε κάθε παιχνίδι σε σχέση με τα βήματα του περιβάλλοντος για διαφορετικές τιμές της μεταβλητής Training Ratio. Τα σχήματα στα δεξιά (b) Model Size παρουσιάζουν την απόδοση που προέκυψε σε κάθε παιχνίδι σε σχέση με τα βήματα του περιβάλλοντος για διαφορετικά Model Size (Αναλύθηκαν στον Σχήμα3.1) που κυμαίνονται από 8 εκατομμύρια έως 200 εκατομμύρια παραμέτρους (Digital image, 10 Jan 2023 [4])

.

Πειραματική Διάταξη Α.2

Αναμένοτας ότι από την πειρατική διάταξη 1 θα καταλήξουμε στο ότι το XL μοντέλο και η τιμή 16 στο Training ratio επιφέρουν την υψηλότερη τελική απόδοση και αποδοτικότητα των δεδομένων , θα κρατήσουμε αυτές τις δύο τιμές σταθερές και θα μεταβάλουμε την παράμετρο του Obstacle Tower Agent-perspective. Μέσω της μεταβολής του Agent-perspective από την τιμή 0 στο 1 ο πράκτορας θα παρατηρεί το περιβάλλον όχι μέσα από την δική του οπτική γωνία αλλά ενός τρίτου προσώπου.

Όσον αφορά τις τιμές των παραμέτρων του DreamerV3 Training Ratio και Model Size θα εξαρτηθούν από τα αποτελέσματα της πειραματικής διάταξης Α.1. Οι υπόλοιπες παράμετροι του Obstacle Tower θα παραμείνουν στις προκαθορισμένες τους τιμές.

Experiment 2	
Dreamer V3 Parameters	Experiment 2.1
Model Size	depends on experiment 1
Training Ratio	depends on experiment 1
Obstacle Tower Parameters	
Train-mode	1
Tower-seed	-1
Starting-floor	0
Total-floors	100
Dense-reward	1
Lighting-type	1
Visual-theme	1
Agent-perspective	1
Allowed-rooms	2
Allowed-modules	2
Allowed-floors	2

Πίνακας 3.3: Λίστα παραμέτρων που επιλέχθηκαν για την πειραματική διάταξη A.2 [112]

Πειραματική Διάταξη A.3

Σύμφωνα με το άρθρο "Competing in the Obstacle Tower Challenge" [7] από τον Alex Nichol, ο οποίος προσπάθησε να εκπαιδεύσει τον πράκτορά του να ξεπερνά το Sokoban Puzzle στο επίπεδο 10, χρησιμοποίησε τη μέθοδο της behavior cloning. Κατά τη διάρκεια της εκπαίδευσης, εισήγαγε ανθρώπινες δείξεις (human demonstrations) που προέρχονταν από τον ίδιο ως είσοδο, προκειμένου να καταλάβει τον τρόπο αντιμετώπισης του πράκτορα στην πρόκληση. Ενδιαφέρον παρατηρείται στο γεγονός ότι μείωσε ελαφρώς το χώρο δράσης (action space) κατά τη διάρκεια του παιχνιδιού στο επίπεδο 10. Η απόφασή του αυτή βασίστηκε στην επισήμανση πως αυτή η μείωση του χώρου δράσης τον έκανε πιο άνετο και αποδοτικό στο παιχνίδι. Έτσι, θα ήταν ενδιαφέρον να δούμε πώς αντιδρά ο πράκτορας όταν έχει στη διάθεση του έναν περιορισμένο χώρο δράσης σε αντίθεση όταν έχει ένα πλήρη.

Το action space στο Obstacle Tower είναι multi-discrete διάνυσμα και συγκεκριμένα θεωρείτε ως ένα υποσύνολο μικρότερων διακριτικών χώρων δράσης, των οποίων η ένωση αντιστοιχεί σε μια μόνο ενέργεια κάθε φορά στο περιβάλλον. Αυτά τα υποσύνολα-υποχώροι είναι οι εξής:

- κίνηση προς τα εμπρός(1)/προς τα πίσω(2)/καμία κίνηση(0)
- αριστερά(1)/δεξιά(2)/καμία κίνηση(0)
- καμία κίνηση(0)/άλμα(1)

- περιστροφή της κάμερας προς τα δεξιά(1)/προς τα αριστερά(2)/καμία κίνηση(0)

Επομένως το διάνυσμα για το action space που προκύπτει είναι το εξής: [3,3,2,3], ενώ όταν μετατραπεί σε ένα μόνο discrete διάνυσμα είναι 54 θέσεις.

Εάν αφαιρέσουμε όλους τους πιθανούς συνδυασμούς κινήσεων που περιλαμβάνουν τη δεξιά, αριστερά και προς τα πίσω κατεύθυνση από το action space, ο αριθμός των δυνατών κινήσεων μειώνεται από 54 σε 11.

Experiment 3	
Dreamer V3 Parameters	Experiment 3.1
Model Size	depends on experiment 1
Training Ratio	depends on experiment 1
Obstacle Tower Parameters	
Train-mode	1
Tower-seed	-1
Starting-floor	0
Total-floors	100
Dense-reward	1
Lighting-type	1
Visual-theme	1
Agent-perspective	depends on experiment 2
Allowed-rooms	2
Allowed-modules	2
Allowed-floors	2

Πίνακας 3.4: Λίστα παραμέτρων που επιλέχθηκαν για την πειραματική διάταξη A.3 [112]

Όσον αφορά τις τιμές των παραμέτρων του DreamerV3 Training Ratio και Model Size, καθώς και την τιμή του Obstacle Tower agent-perspective, αυτές θα εξαρτηθούν από τα αποτελέσματα των πειραματικών διατάξεων A.1 και A.2. Οι υπόλοιπες παράμετροι του Obstacle Tower θα παραμείνουν στις προκαθορισμένες τους τιμές.

Πειραματική Διάταξη Α.4

Στο άρθρο [4], πραγματοποιείται μια σύγκριση της απόδοσης του αλγορίθμου DreamerV3 σε διάφορα παιχνίδια, δοκιμάζοντας διάφορες τιμές στις παραμέτρους Model Size (Small, Medium, Large, XLarge) και Training Ratio (1, 2, 4, 8, 16, 32, 64), όπως παρουσιάζεται στο Σχήμα 3.2. Η ίδια διαδικασία θα εφαρμοστεί και για το παιχνίδι Obstacle Tower στην πειραματική διάταξη Α.1.

Επειδή οι τιμές που μπορεί να λάβει η παράμετρος Model Size είναι συγκεκριμένες και έχουν εξεταστεί, αντίθετα με το Training Ratio, το οποίο εφαρμόζει πολύ μεγάλες τιμές όπως 512 και 1024 σε κάποια παιχνίδια (πληροφορία που δεν αναφέρεται στο άρθρο), θα ήταν ενδιαφέρον να εξεταστεί πώς αντιδρά το σύστημα στο περιβάλλον Obstacle Tower, έχοντας το Training Ratio ρυθμισμένο σε πολύ μεγάλες τιμές, ειδικά στην περίπτωση της τιμής 1024.

Experiment 4	
Dreamer V3 Parameters	Experiment 4.1
Model Size	depends on experiment 1
Training Ratio	1024
Obstacle Tower Parameters	
Train-mode	1
Tower-seed	-1
Starting-floor	0
Total-floors	100
Dense-reward	1
Lighting-type	1
Visual-theme	1
Agent-perspective	depends on experiment 2
Allowed-rooms	2
Allowed-modules	2
Allowed-floors	2

Πίνακας 3.5: Λίστα παραμέτρων που επιλέχθηκαν για την πειραματική διάταξη Α.4 [112]

Όσον αφορά τη τιμή της παραμέτρου του DreamerV3 Model Size, καθώς και την τιμή του Obstacle Tower agent-perspective, αυτές θα εξαρτηθούν από τα αποτελέσματα των πειραματικών διατάξεων Α.1 και Α.2. Οι υπόλοιπες παράμετροι του Obstacle Tower θα παραμείνουν στις προκαθορισμένες τους τιμές.

3.4.2 Πειραματική Διατάξη Τύπου Β

Αφού ολοκληρωθούν όλες οι παραπάνω πειραματικές διατάξεις και, μέσω των αποτελεσμάτων, επιτεύχθηκε μια πιο σαφής κατανόηση του τρόπου λειτουργίας και της απόδοσης του αλγορίθμου DreamerV3, θα ήταν επιθυμητό να προβούμε σε μια αντίστοιχη πειραματική διάταξη και για το παιχνίδι Minecraft. Ειδικότερα, με βάση τις παραμέτρους που παρουσιάζονται στον πίνακα 3.1, διακρίνουμε 9 βασικές υποδιαδικασίες (αντίστοιχες των 12 στο Minecraft) βάση τριών παραμέτρων. Αυτές οι υποδιαδικασίες παρουσιάζονται παρακάτω:

1. Allowed-rooms
    - Normal room
    - Key room
    - Puzzle room
2. Allowed-floor
    - Straightforward floor
    - Branching floor
    - Circling floor
3. Allowed-modules
    - No modules
    - Easy modules (πλατφόρμες, ανυψωμένα μέρη)
    - Full range of modules (εχθροί, πριόνια, πολύ γρήγορες πλατφόρμες κ.ά.)

Για την υλοποίηση αυτού, απαιτείται λεπτομερής παρατήρηση για τον καθορισμό σε ποιον όροφο εμφανίζεται κάθε υποδιαδικασία. Στη συνέχεια, ο πράκτορας θα πρέπει να εκπαιδευτεί ξεκινώντας από τον συγκεκριμένο όροφο όπου εμφανίζεται η κάθε υποδιαδικασία, χρησιμοποιώντας την παράμετρο starting-floor και τελειώνοντας στο επιθυμητό όροφο μέσω της παραμέτρου total-floors, εκπαιδεύοντάς τον για έναν μεγάλο αριθμό βημάτων με διάφορα seeds. Για παράδειγμα, το puzzle εμφανίζεται για πρώτη φορά στον 10ο - 15ο όροφο και είναι το κυρίαρχο θέμα σε αυτούς τους ορόφους.

Για να γίνει όμως αυτό, απαιτούνται πολλοί υπολογιστικοί πόροι και πολύς χρόνος εκπα-

ίδευσης του πράκτορα, εάν λάβουμε υπόψη ότι για το Minecraft χρειάστηκαν 17 μέρες εκπαίδευσης σε XL Model size.

3.5 Αναμενόμενα Αποτελέσματα από τις πιο Πάνω Πειρματικές Διατάξεις

Μέσω των παραπάνω πειραματικών διατάξεων, εκτός από την επαλήθευση του ότι:

1. Όσο το μέγεθος του μοντέλου αυξάνεται, τόσο θα αυξάνεται η τελική απόδοση και η αποδοτικότητα των δεδομένων.
2. Το Training Ratio=16 επιφέρει την υψηλότερη τελική απόδοση.

Επιπλέον, υπάρχει η προσδοκία ότι τουλάχιστον ένα αποτέλεσμα των πιο πάνω πειραματικών διατάξεων θα υπερβαίνειτην καλύτερη απόδοση που επιτεύχθηκε από τον πράκτορα του Alex Nichol (Μέσος όρος ορόφων = 19.4, Μέσο Reward = 35.86) [7]. Για να το επιτύχουμε αυτό θα πρέπει να λάβουμε τα εξής αποτελέσματα:

1. Μέσος όρος αναμενόμενου Reward-Average Expected reward > 35.86
2. Μέσος όρος αναμενόμενων ορόφων-Average Expected floors > 19.4

Κεφάλαιο 4

Αποτελέσματα Πειραμάτων (Results)

4.1 Μετρικές Επίδοσης

Για την αξιολόγηση του αλγορίθμου DreamerV3 στο Obstacle Tower, για κάθε μία από τις πειραματικές διατάξεις που αναφέρθηκαν στο προηγούμενο κεφάλαιο, χρησιμοποιήθηκαν γραφικές παραστάσεις που βασίζονταν στον κινητό μέσο όρο ΚΜΟ (moving average). Αυτές οι παραστάσεις δείχνουν τη συσσωρευτική ανταμοιβή (reward) που κατάφερε ο πράκτορας σε σχέση με τα βήματα/επεισόδια που εκτέλεσε στο περιβάλλον.

Πριν προχωρήσουμε στην παρουσίαση των γραφικών, θα γίνει μία μικρή αναφορά στους ορισμούς των μετρικών στις οποίες βασίζονται οι γραφικές παραστάσεις μας.

- **Βήμα (Step):** Κάθε ενέργεια που εκτελεί ο πράκτορας στο περιβάλλον θεωρείται ένα βήμα.
- **Συσσωρευτική Ανταμοιβή -Σκορ (reward):** Ανταμοιβή-σκορ που λαμβάνει ο πράκτορας όταν καταφέρει να εκτελέσει μια διαδικασία που τον φέρνει πιο κοντά στο να βγει από τον συγκεκριμένο όροφο στον οποίο βρισκόταν (π.χ. πλησιάζει την πόρτα και αυτή άνοιγε (+0.1), περνά από την πόρτα δωματίου (+1), αποκτά ένα κλειδί κ.λπ.).
- **Επεισόδιο (Episode):** Όταν ο πράκτορας ολοκληρώνει μια σειρά δράσεων μέχρι τον τερματισμό του παιχνιδιού (π.χ. να εξαντλήθει ο διαθέσιμος χρόνο ή να πέσει σε γκρεμό), θεωρείται ένα επεισόδιο.

4.2 Έλεγχος Αλγορίθμου DreamerV3 Μέσω Επαλήθευσης Δεδομένων στο Παιχνίδι Crafter

Προτού προχωρήσουμε στην υλοποίηση των πειραματικών διατάξεων του αλγορίθμου DreamerV3 στο παιχνίδι Obstacle Tower, θεωρήσαμε σκόπιμο να αναπαράγουμε ορισμένα από τα δεδομένα που παρέχονται στο άρθρο [4], προκειμένου να επαληθεύσουμε την εγκυρότητά τους και την αποδοτικότητα του DreamerV3. Το παιχνίδι που επιλέχθηκε ήταν το Crafter. Στο άρθρο [4] αναφέρεται ότι όλα τα παιχνίδια εκπαιδεύτηκαν σε μια κάρτα γραφικών Nvidia V100 GPU (όπως φαίνεται στο σχήμα 4.1, με 32 GB μνήμη).

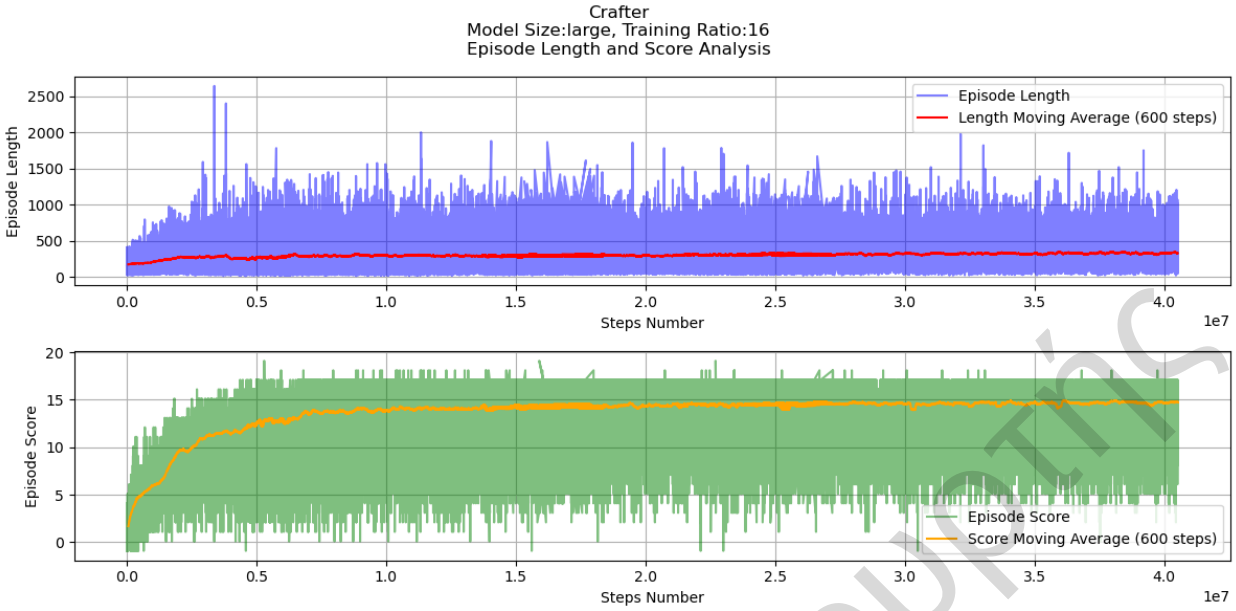
SPECIFICATIONS

	Tesla V100 PCIe	Tesla V100 SXM2
GPU Architecture	NVIDIA Volta	
NVIDIA Tensor Cores	640	
NVIDIA CUDA® Cores	5,120	
Double-Precision Performance	7 TFLOPS	7.8 TFLOPS
Single-Precision Performance	14 TFLOPS	15.7 TFLOPS
Tensor Performance	112 TFLOPS	125 TFLOPS
GPU Memory	32GB /16GB HBM2	
Memory Bandwidth	900GB/sec	
ECC	Yes	
Interconnect Bandwidth	32GB/sec	300GB/sec
System Interface	PCIe Gen3	NVIDIA NVLink
Form Factor	PCIe Full Height/Length	SXM2
Max Power Comsumption	250 W	300 W
Thermal Solution	Passive	
Compute APIs	CUDA, DirectCompute, OpenCL™, OpenACC	

Σχήμα 4.1: Χαρακτηριστικά κάρτας γραφικών μοντέλου Nvidia V100 [113]

Επιλέξαμε να αναπαράγουμε τα δεδομένα με Train Ratio: 16 και Model size: Medium για λόγους κυρίως εξοικονόμησης χρόνου και λόγω του ότι η δική μας κάρτα γραφικών (NVIDIA GeForce RTX 2080 GPU) [114] διαθέτει 12 GB μνήμης, κάτι που διαδραμάτισε καθοριστικό ρόλο για την μετέπειτα εκτέλεση των πειραμάτων.

Τα αποτελέσματα που πήραμε διακρίνονται στο παρακάτω Σχήμα 4.2.



Σχήμα 4.2: Αποτελέσματα αλγορίθμου DreamerV3 στο παιχνίδι Crafter.Στη πάνω γραφική παράσταση απεικονίζεται ο αριθμός των βημάτων που εκτελέστηκαν σε κάθε επεισόδιο σε σχέση με τον συνολικό αριθμό βημάτων, ενώ αντίστοιχα στην κάτω γραφική παράσταση απεικονίζεται το score σε σχέση με τον αριθμό των βημάτων

Συγκρίνοντας αυτά τα αποτελέσματα με το γράφημα 3.2 από το άρθρο του dreamerv3, επαληθεύονται τα δεδομένα.

4.3 Αποτελέσματα

Είναι σημαντικό να σημειωθεί ότι μέχρι τα 10 εκατομμύρια βήματα, κατά τη διάρκεια των πειραματικών διατάξεων, οι υψηλότερες τελικής απόδοσης και οι υψηλότερη τιμή του moving Average Rewards της κάθε μιας από αυτές είχαν σχεδόν αμελητέα διαφορά, με εξαίρεση την πειραματική διάταξη Α.4. Ως επέκταση αυτού, θεωρείται ότι έχουν ίδια τιμή.

4.3.1 Πειραματικές Διατάξεις Τύπου Α

Σε αυτό το υποκαιφάλαιο θα παρουσιαστούν αναλυτικά όλα τα αποτολέσματα απο τις πειραματικές διατάξεις του κεφαλαιου Μεθοδολογία (Methodology)3.

Αποτελέσματα Πειραματικής Διάταξης Α.1

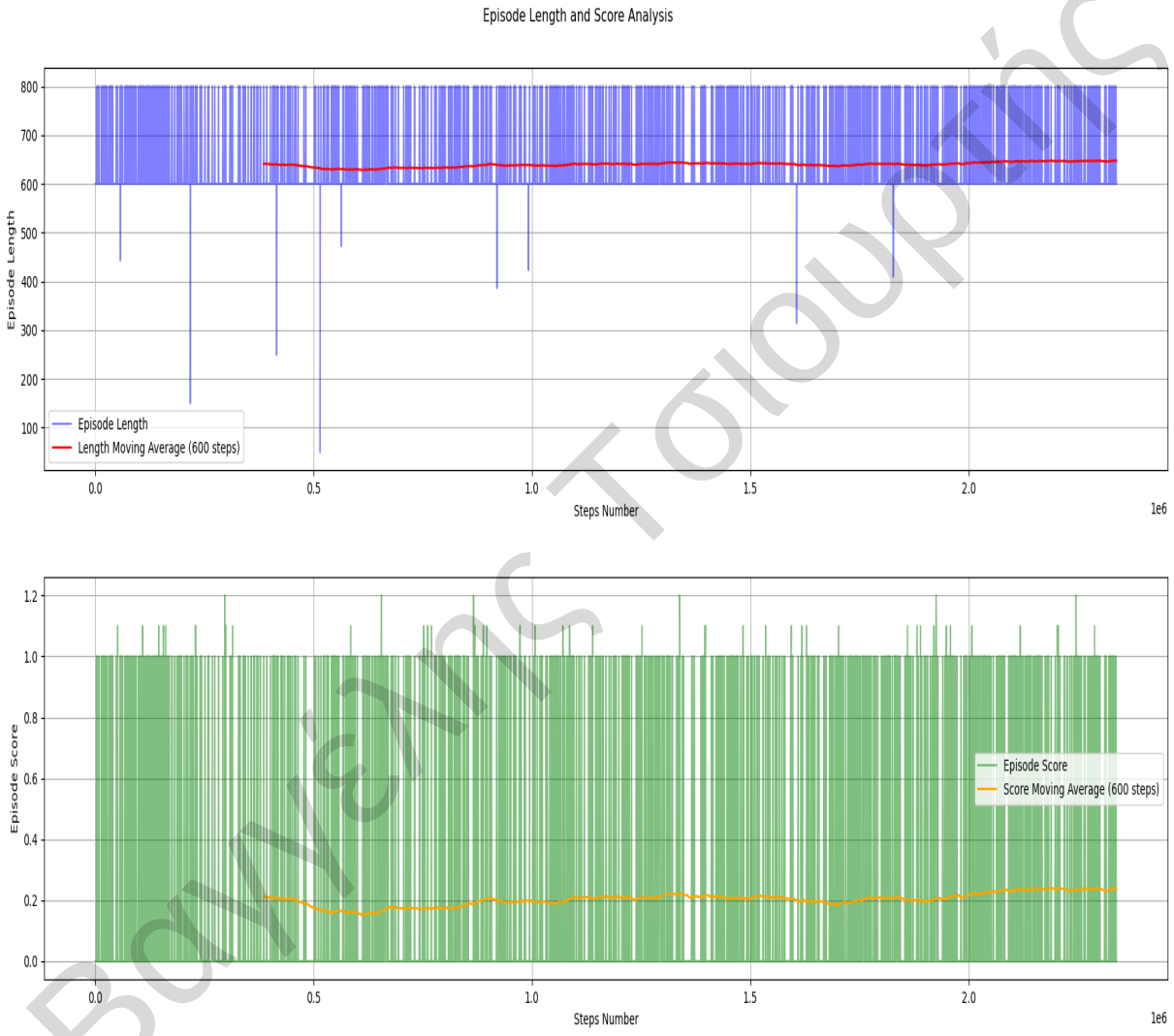
Αρχίζοντας την υλοποίηση των πειραματικών διατάξεων 1.1 - 1.3 αντιμετωπίσαμε τις πιο κάτω περιορισμούς και προκλήσεις:

1. Αρχικά, για το retro mode, το Obstacle Tower παρέχει οπτική παρατήρηση (visual observation) διαστάσεων 84x84. Λόγω της μνήμης 12 GB της GPU που χρησιμοποιήσαμε, δεν ήταν δυνατό να εκτελέσουμε οποιοδήποτε πείραμα. Ως αποτέλεσμα, μειώσαμε τις διαστάσεις της οπτικής παρατήρησης σε 64x64.

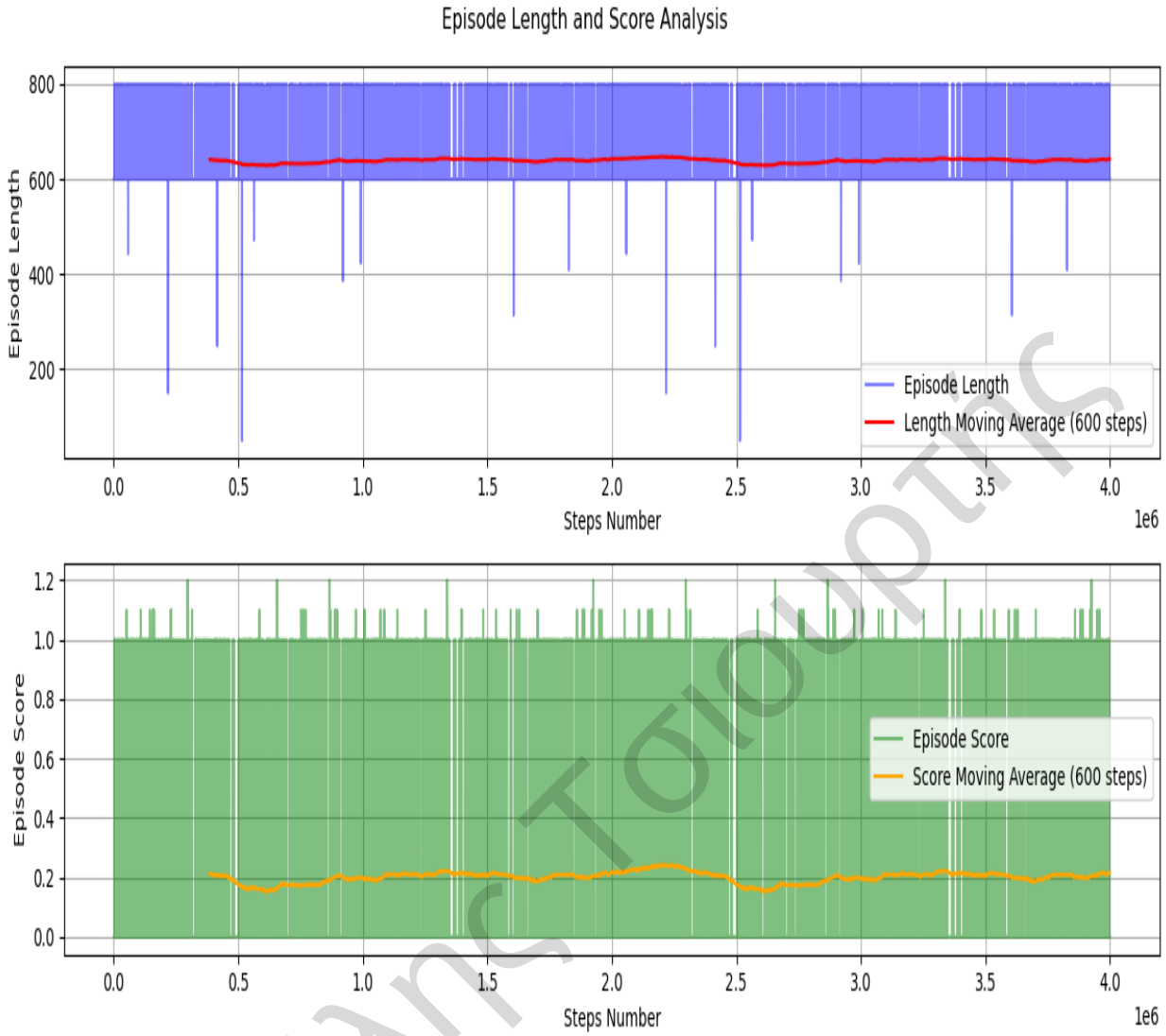
2. Κατά την προσπάθειά μας να εκτελέσουμε τον DreamerV3 με μέγεθος μοντέλου (Model Size XL), το ανώτατο όριο που καταφέρναμε να φτάσουμε ήταν περίπου 2,4 εκατομμύρια βήματα, χωρίς να παρατηρήσουμε σημαντική βελτίωση στην εκπαίδευση4.3. Αλλάζοντας την παράμετρο Model Size σε Medium ή Small, παρατηρήσαμε αύξηση των βημάτων πέραν των 2,5 εκατομμυρίων και συγκεκριμένα είχε φτάσει μέχρι περίου 7 εκατομμύρια βηματα, αλλά και πάλι χωρίς σημαντική βελτίωση στην εκπαίδευση του πράκτορα με αποτέλεσμα να το διακόψουμε4.4. Έτσι, δοκιμάζοντας διάφορες αλλαγές τόσο στις παραμέτρους του DreamerV3 (π.χ., rssm(deter, units), encoder, decoder (mlp layers, mlp units, cnn depth)) όσο και στις παραμέτρους του Obstacle Tower (π.χ., train-mode, tower-seed, dense-reward), συνειδητοποιήσαμε ότι η επιλογή σταθερού seed οδηγεί σε βελτίωση του DreamerV3. Ως αποτέλεσμα, θέσαμε τη μεταβλητή tower-seed=1.

3. Ακόμη και με σταθερό seed, ο πράκτορας δεν κατάφερε να βελτιωθεί στο XL Model Size λόγω των περιορισμένων βημάτων που μπορούσε να κάνει λόγω της RAM.

4. Έχοντας αφιερώσει αρκετό χρόνο στα παραπάνω και λόγο περιορισμένης υπολογιστικής ισχύος, αναγκαστήκαμε να μειώσουμε τον αριθμό των πειραμάτων από 9 σε 6. Επιπλέον, πραγματοποιήσαμε αλλαγές στη μεταβλητή tower-seed, την οποία θέσαμε ίση με 1, αντίθετα με τον προηγούμενο προκαθορισμό όπου παίρνει τιμή -1, προκειμένου να διασφαλίσουμε ότι σε κάθε νέο επεισόδιο ο πράκτορας αντιμετωπίζει το ίδιο περιβάλλον. Οι νέες 4 πειραματικές διατάξεις διακρίνονται στον νέο πίνακα 4.1.



Σχήμα 4.3: Η πανω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας χ και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size=Xlarge, Training Ratio=16 και tower-seed=-1



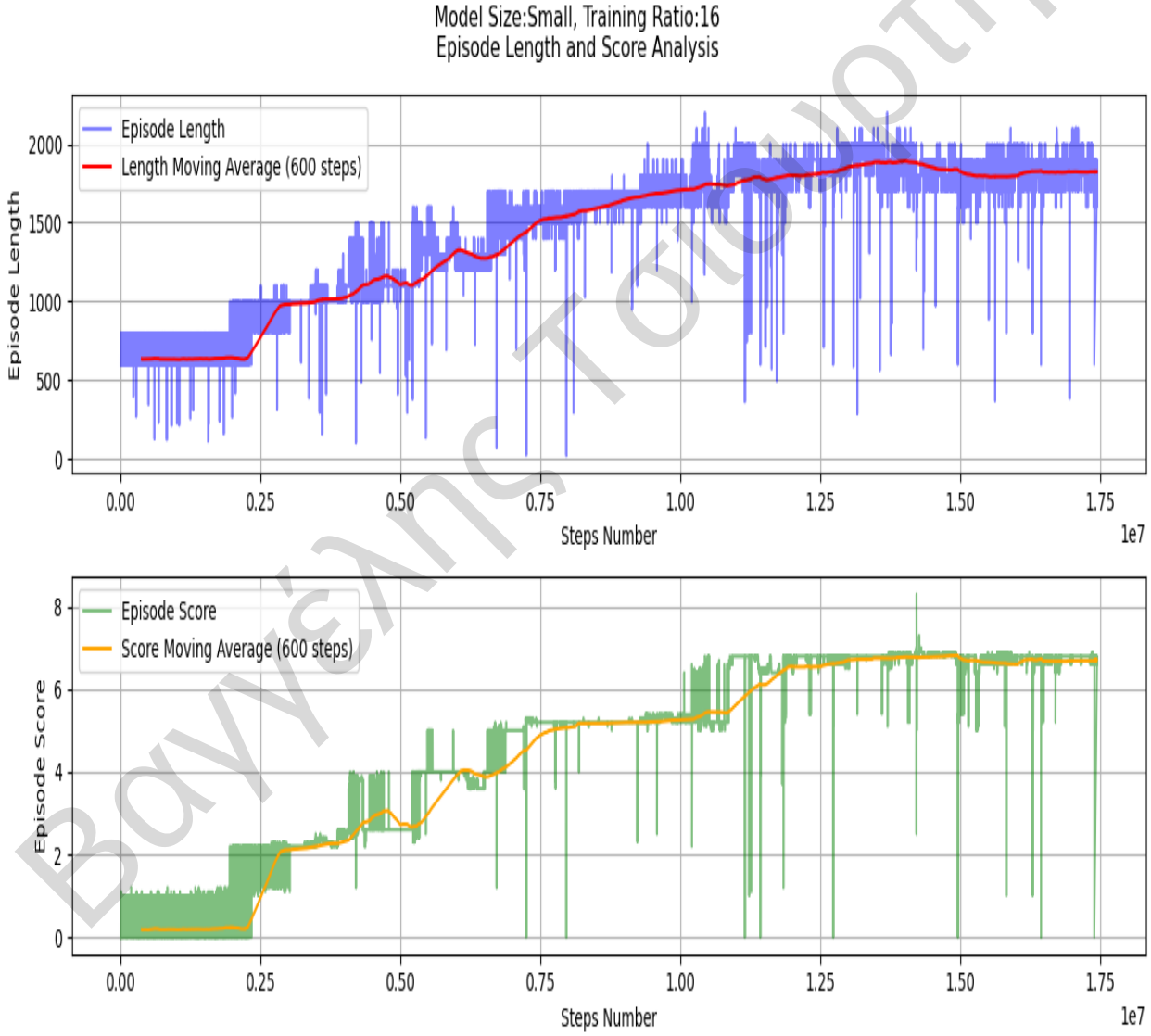
Σχήμα 4.4: Η πανω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας χ και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size=Medium, Training Ratio=16 και tower-seed=-1

Experiment 1			
Dreamer V3 Parameters		Experiment A.1.1	Experiment A.1.2
Model Size		S-M-L	S-M-L
Training Ratio		16	64
Obstacle Tower Parameters			
Train-mode		1	1
Tower-seed		1	1
Starting-floor		0	0
Total-floors		100	100
Dense-reward		1	1
Lighting-type		1	1
Visual-theme		1	1
Agent-perspective		0	0
Allowed-rooms		2	2
Allowed-modules		2	2
Allowed-floors		2	2

Πίνακας 4.1: Νέα λίστα παραμέτρων που επιλέχθηκαν για την πειραματική διάταξη 1 λόγο των προκλήσεων που προέκυψαν

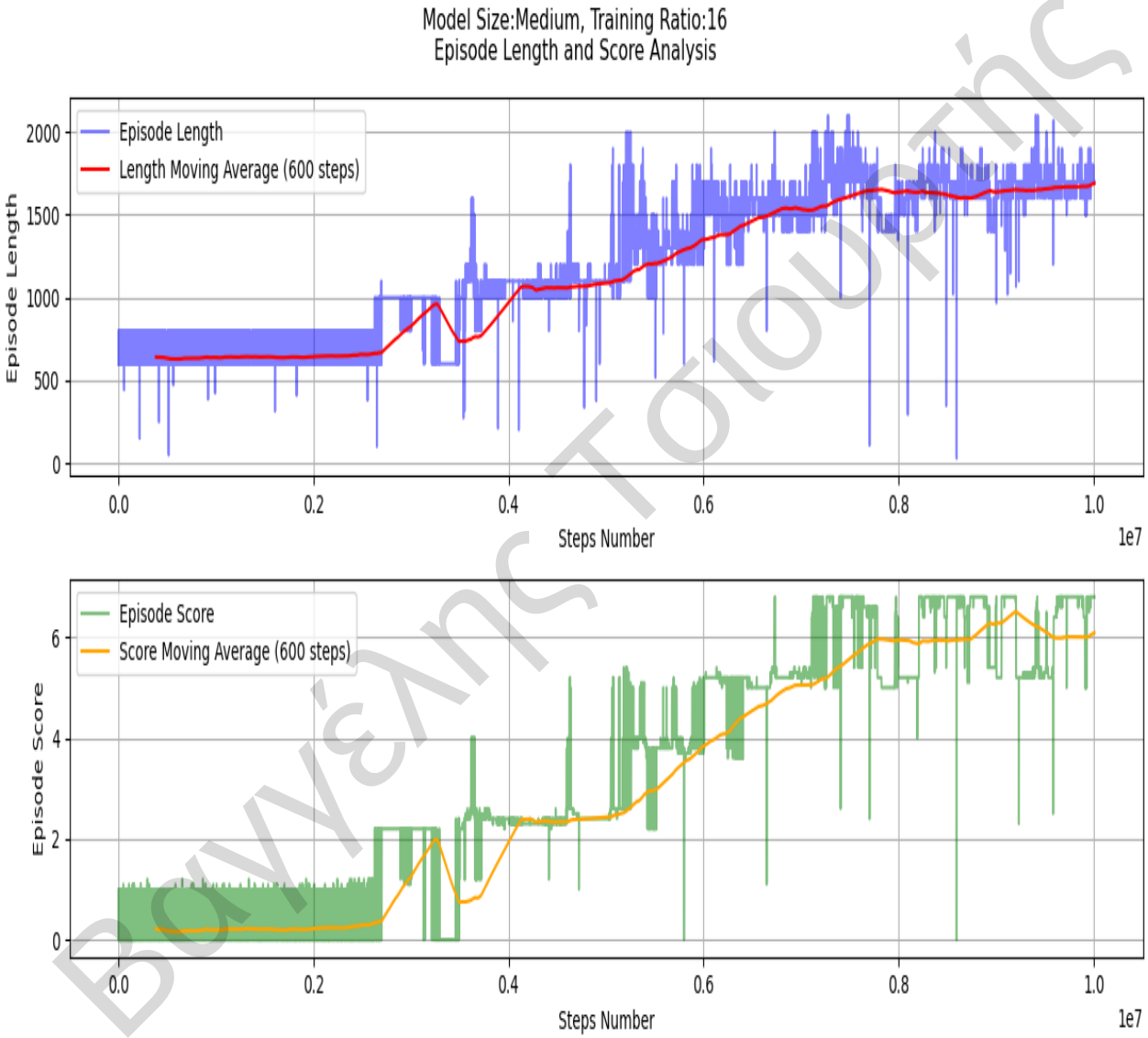
Παρουσίαση των αποτελεσμάτων της νέας πειραματικής διάταξης A.1.1 που απεικονίζετε στον Πίνακα4.1

Το Σχήμα.4.5 παρουσιάζει την εξέλιξη του πράκτορα στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Η πάνω γραφική απεικονίζει τον κινούμενο μέσο όρο της διάρκειας των επεισοδίων σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X αντιπροσωπεύει το σύνολο των βημάτων που εκτελέστηκαν στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Τα χαρακτηριστικά του μοντέλου είναι Model Size= Small και Training Ratio=16.



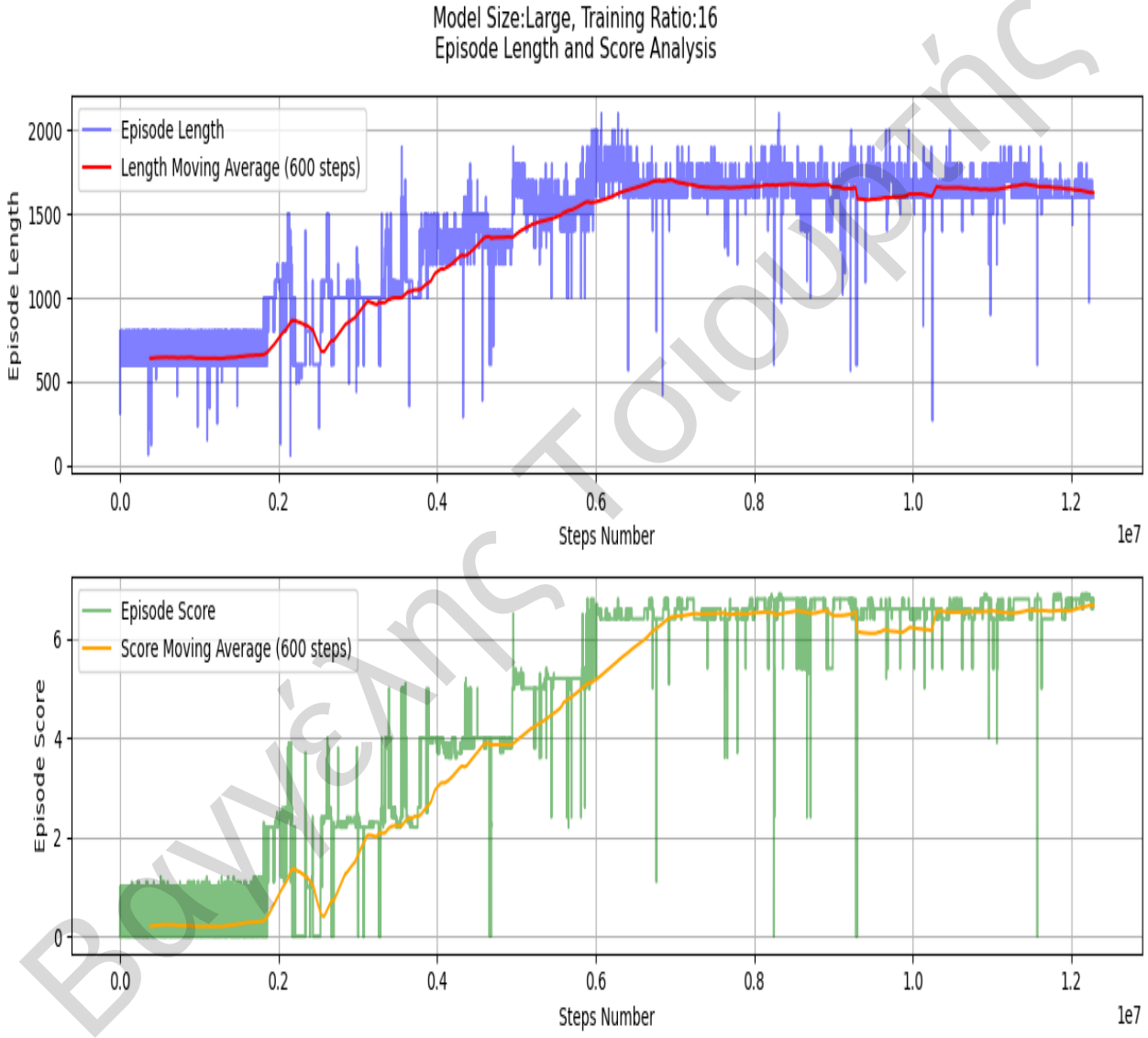
Σχήμα 4.5: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Small και Training Ratio=16

Το Σχήμα.4.6 παρουσιάζει την εξέλιξη του πράκτορα στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Η πάνω γραφική απεικονίζει τον κινούμενο μέσο όρο της διάρκειας των επεισοδίων σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X αντιπροσωπεύει το σύνολο των βημάτων που εκτελέστηκαν στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Τα χαρακτηριστικά του μοντέλου είναι Model Size= Medium και Training Ratio=16.



Σχήμα 4.6: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Medium και Training Ratio=16

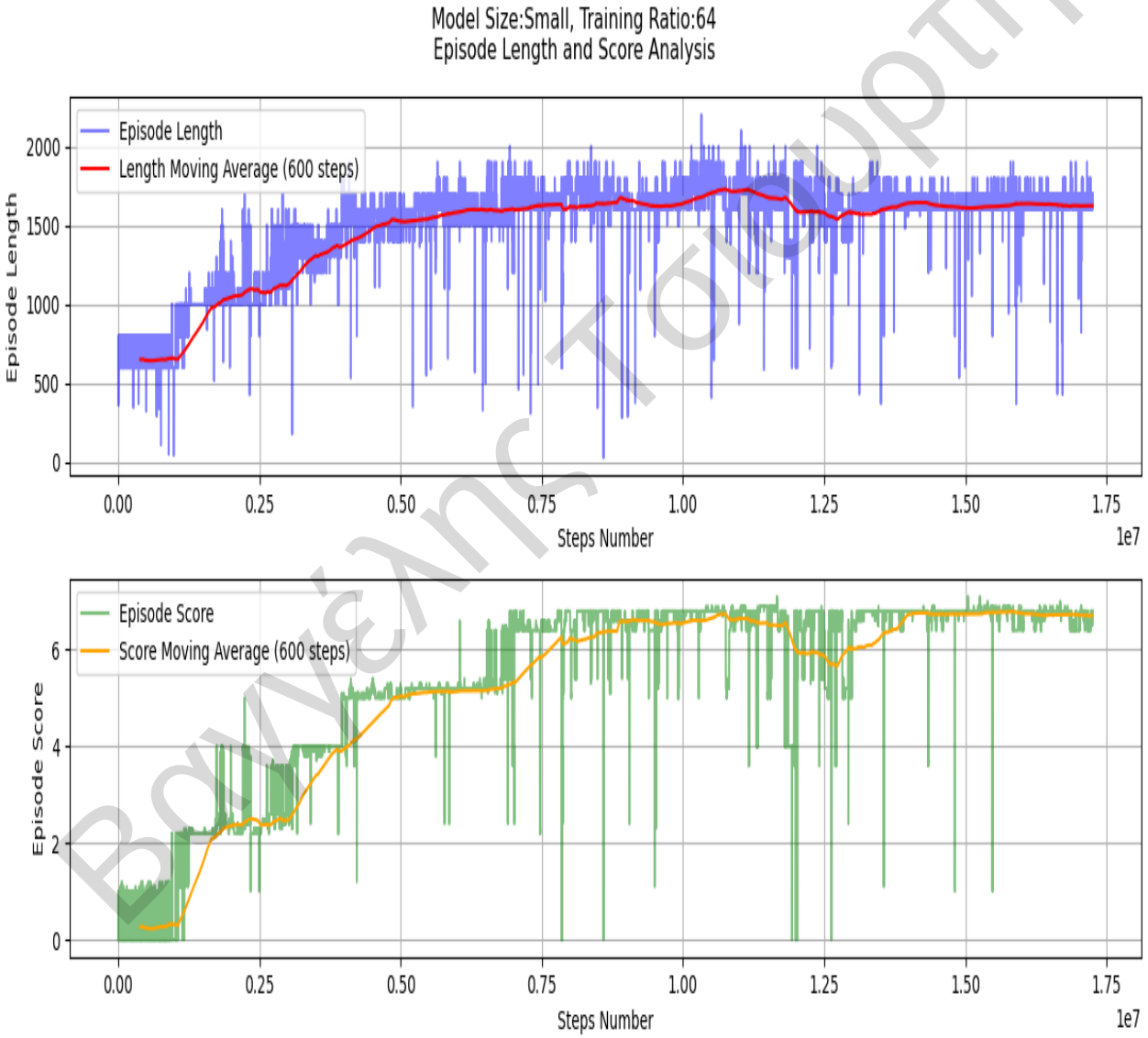
Το Σχήμα.4.7 παρουσιάζει την εξέλιξη του πράκτορα στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Η πάνω γραφική απεικονίζει τον κινούμενο μέσο όρο της διάρκειας των επεισοδίων σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X αντιπροσωπεύει το σύνολο των βημάτων που εκτελέστηκαν στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Τα χαρακτηριστικά του μοντέλου είναι Model Size= Large και Training Ratio=16.



Σχήμα 4.7: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Large και Training Ratio=16

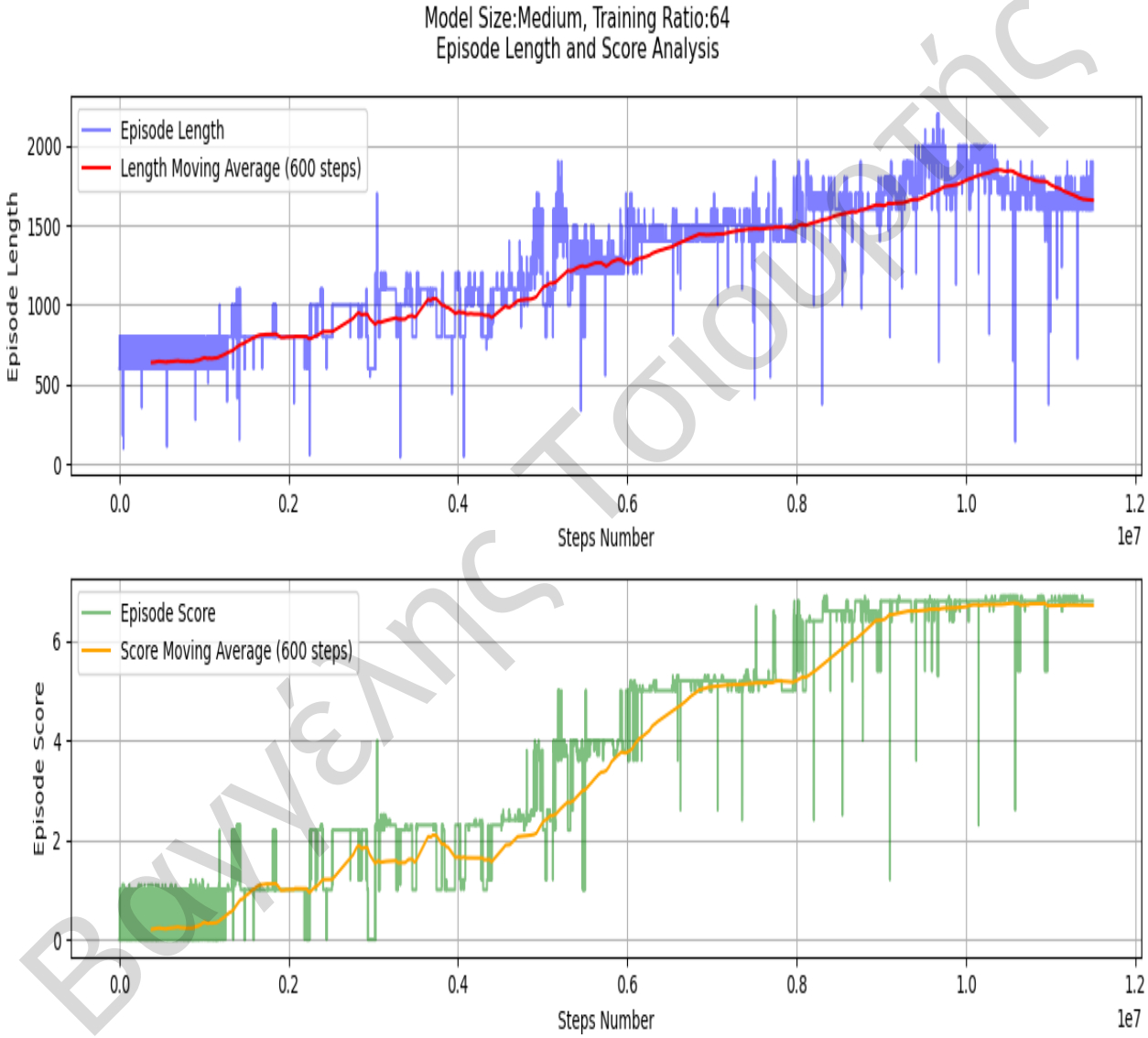
Παρουσίαση των αποτελεσμάτων της νέας πειραματικής διάταξης A.1.2 που απεικονίζετε στον Πίνακα4.1

Το Σχήμα.4.8 παρουσιάζει την εξέλιξη του πράκτορα στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Η πάνω γραφική απεικονίζει τον κινούμενο μέσο όρο της διάρκειας των επεισοδίων σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X αντιπροσωπεύει το σύνολο των βημάτων που εκτελέστηκαν στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Τα χαρακτηριστικά του μοντέλου είναι Model Size= Small και Training Ratio=64.



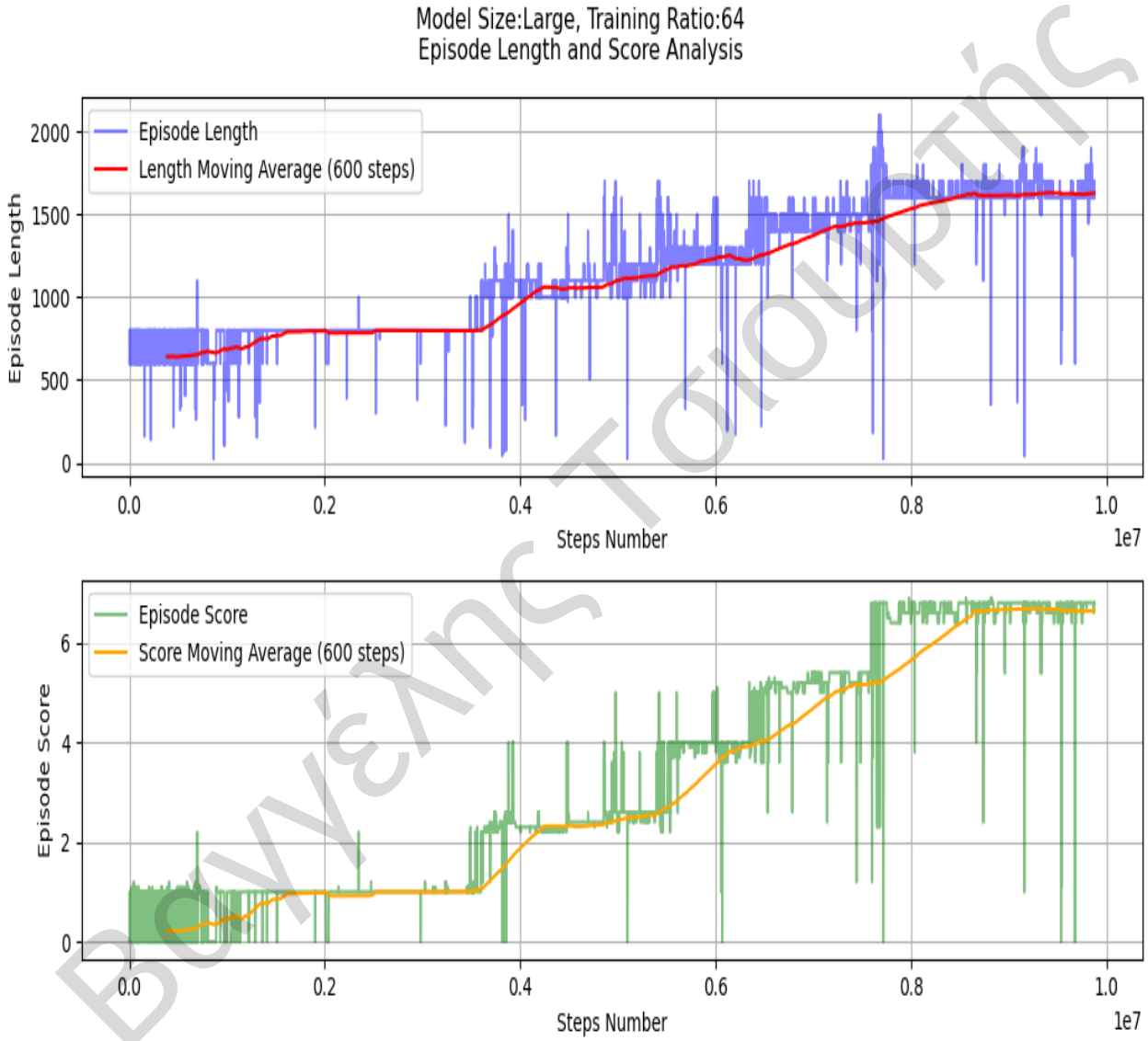
Σχήμα 4.8: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Small και Training Ratio=64

Το Σχήμα.4.9 παρουσιάζει την εξέλιξη του πράκτορα στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Η πάνω γραφική απεικονίζει τον κινούμενο μέσο όρο της διάρκειας των επεισοδίων σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X αντιπροσωπεύει το σύνολο των βημάτων που εκτελέστηκαν στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Τα χαρακτηριστικά του μοντέλου είναι Model Size= Medium και Training Ratio=64.



Σχήμα 4.9: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Medium και Training Ratio=64

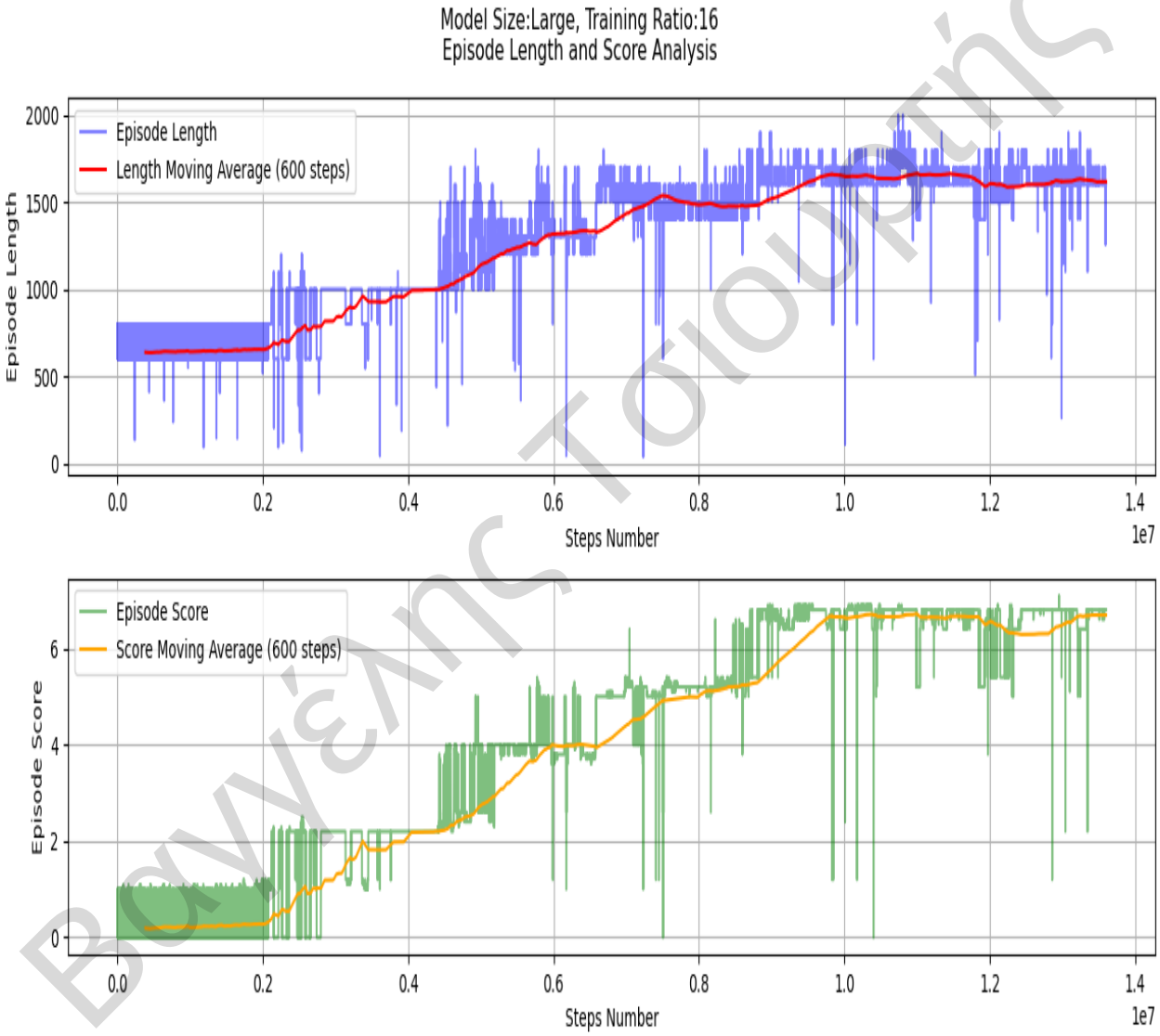
Το Σχήμα.4.10 παρουσιάζει την εξέλιξη του πράκτορα στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Η πάνω γραφική απεικονίζει τον κινούμενο μέσο όρο της διάρκειας των επεισοδίων σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X αντιπροσωπεύει το σύνολο των βημάτων που εκτελέστηκαν στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσης. Τα χαρακτηριστικά του μοντέλου είναι Model Size= Large και Training Ratio=64.



Σχήμα 4.10: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας X και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτελεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Large και Training Ratio=64

Αποτελέσματα Πειραματικής Διάταξης Α.2

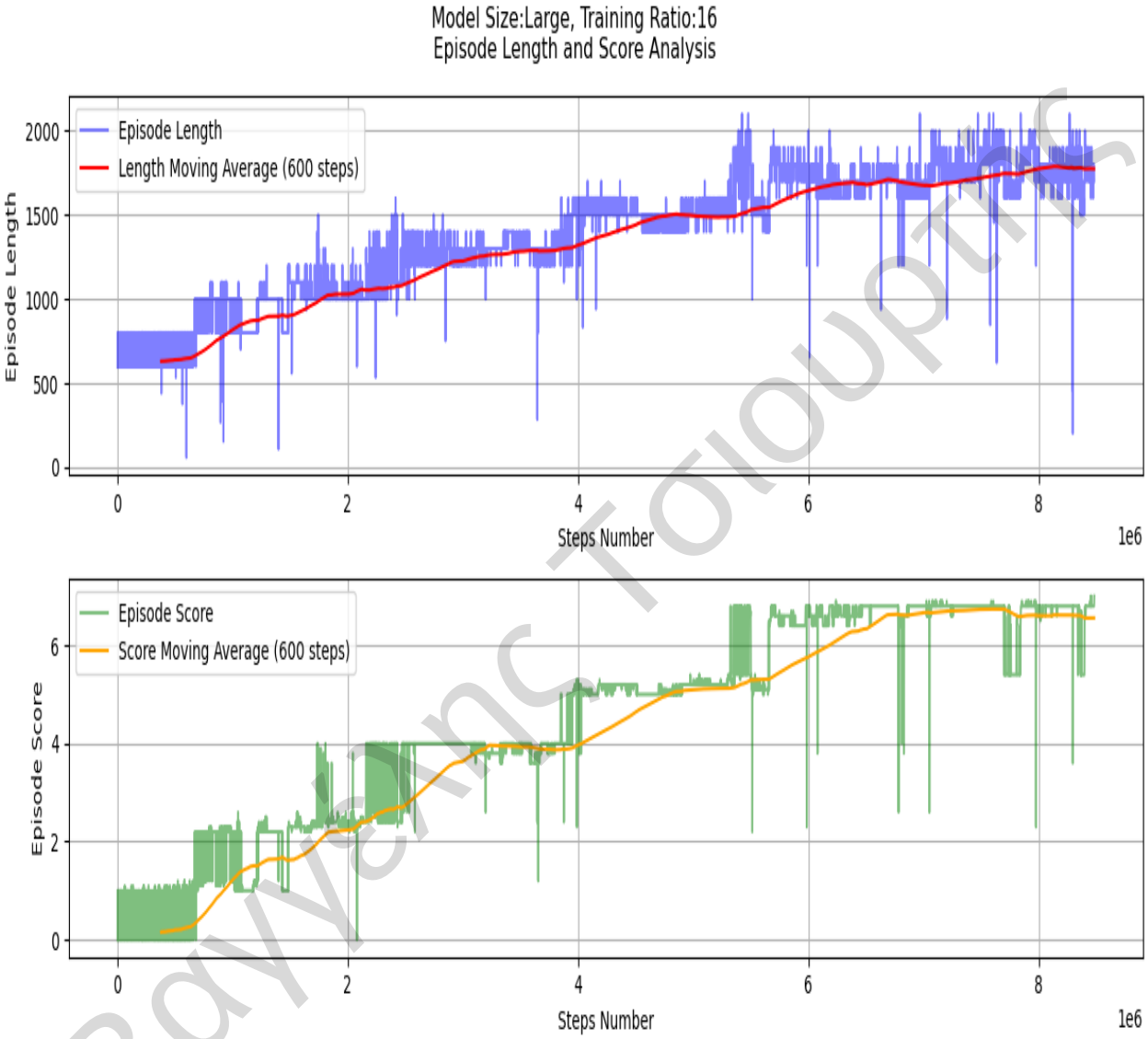
Το Σχήμα4.11 παρουσίαζει τα αποτελεμάτα της περιαματικής διάταξης Α.2 στην οποία έγινε μεταβολή της παραμέτρου του παιχνιδίου Obstacle Tower (Agent-perspective) από την τιμή 0 στο 1, με αποτέλεμα ο πράκτορας να παρατηρεί το περιβάλλον όχι μέσα από την δική του οπτική γωνία όπως συνέβη στην πειραματική διάταξη τύπου Α αλλά ενός τρίτου προσώπου.



Σχήμα 4.11: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας Χ και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτέλεσε στο περιβάλλον Obstacle Tower ο πράκτορας Κατα την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Large, Training Ratio=16, και η παράμετρος Agent-Perspective=1

Αποτελέσματα Πειραματικής Διάταξης Α.3

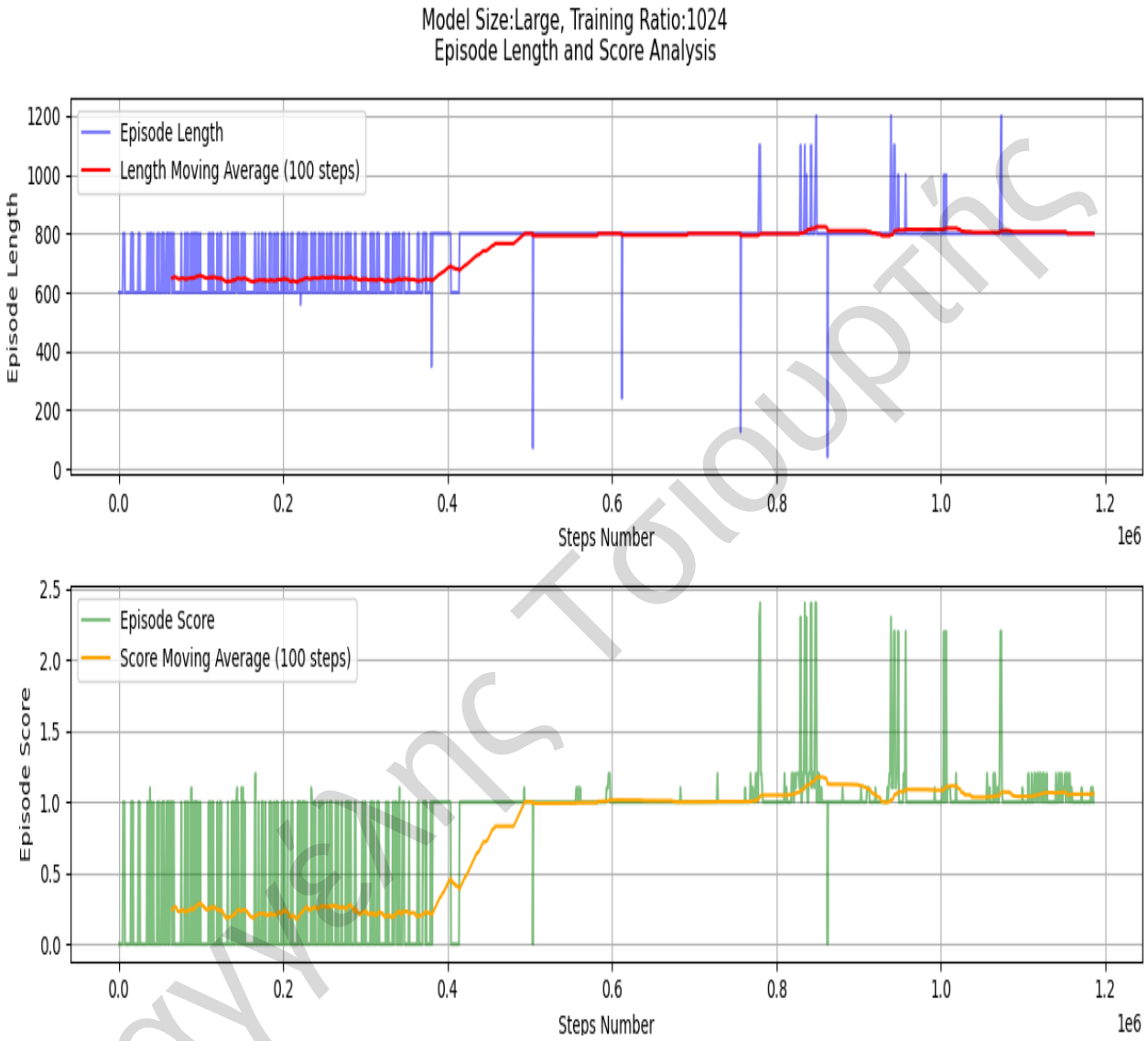
Το Σχήμα4.12 παρουσίαζει τα αποτελεσμάτα της πειραματικής διάταξης Α.3, όπου πραγματοποιήθηκε μετάβαση του Action Space από 54 σε 11 κινήσεις, με την ανάκληση της δυνατότητας εκτέλεσης κινήσεων προς τα δεξιά, αριστερά και πίσω από τον πράκτορα.



Σχήμα 4.12: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας Χ και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτέλεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Large, Training Ratio=16, και η παράμετρος Agent-Perspective=0

Αποτελέσματα Πειραματικής Διάταξης Α.4

Το Σχήμα4.13 παρουσίαζει τα αποτελεσμάτα της πειραματικής διάταξης Α.4, όπου πραγματοποιήθηκε μεταβολή της τιμής του Training Ratio σε 1024.



Σχήμα 4.13: Η πάνω γραφική παρουσιάζει τον κινούμενο μέσο όρο της διάρκειας του επεισοδίου σε βήματα, ενώ η κάτω γραφική απεικονίζει τον κινούμενο μέσο όρο του σκορ του πράκτορα. Ο άξονας Χ και στις δύο γραφικές απεικονίζει το σύνολο των βημάτων που εκτέλεσε στο περιβάλλον Obstacle Tower ο πράκτορας κατά την διάρκεια της εκπαίδευση του. Οι παράμετροι του μοντέλου είναι Model Size= Large, Training Ratio=1024, και η παράμετρος Agent-Perspective=0

Κεφάλαιο 5

Συζήτηση (Discussion)

5.1 Εισαγωγή

Σκοπός αυτής της ενότητας είναι αρχικά η ερμηνεία των αποτελεσμάτων των πειραματιών διατάξεων που παρουσιάστηκαν στο κεφάλαιο Αποτελέσματα Πειραμάτων και αποτελεί τον πυλώνα αξιολόγησης της μεταπτυχιακής εργασιας. Ακολούθος θα γίνει μια αναφορά στους περιορισμούς και τις προκλήσεις που αντιμετοπίστηκαν κατα την διάρκεια αυτής της μεταπτυχιακής εργασίας.

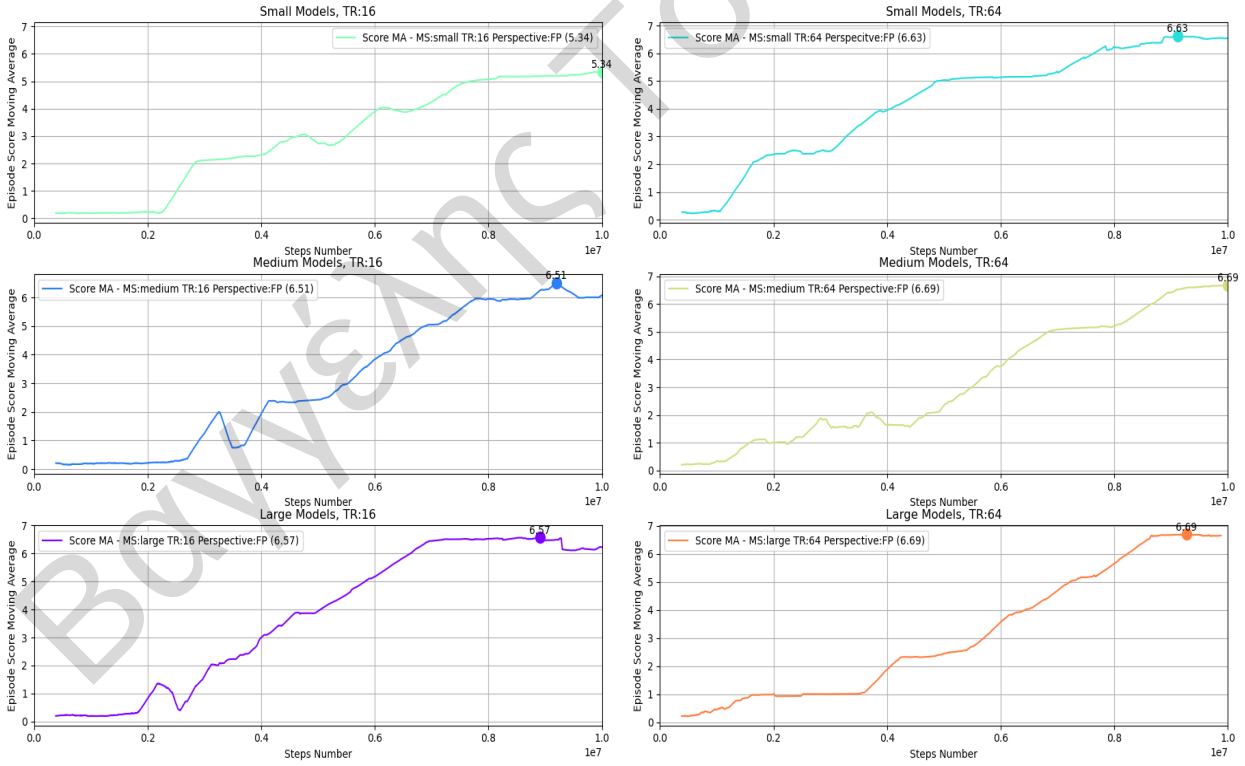
5.2 Ερμηνεία των Αποτελεσμάτων

Αυτή η ενότητα ερμηνεύει τα αποτελέσματα που προήλθαν από το Κεφάλαιο 4, λαμβάνοντας υπόψη πόσο καλά απέδωσε ο αλγόριθμος DreamerV3 στο Obstacle Tower. Συγκρίνει τα αποτελέσματα με τις αρχικές προσδοκίες και συζητά οποιεσδήποτε έκπληξεις ή αποκλίσεις από τα αναμενόμενα αποτελέσματα.

Ερμηνεία Αποτελεσμάτων Περιματικής Διάταξης A.1

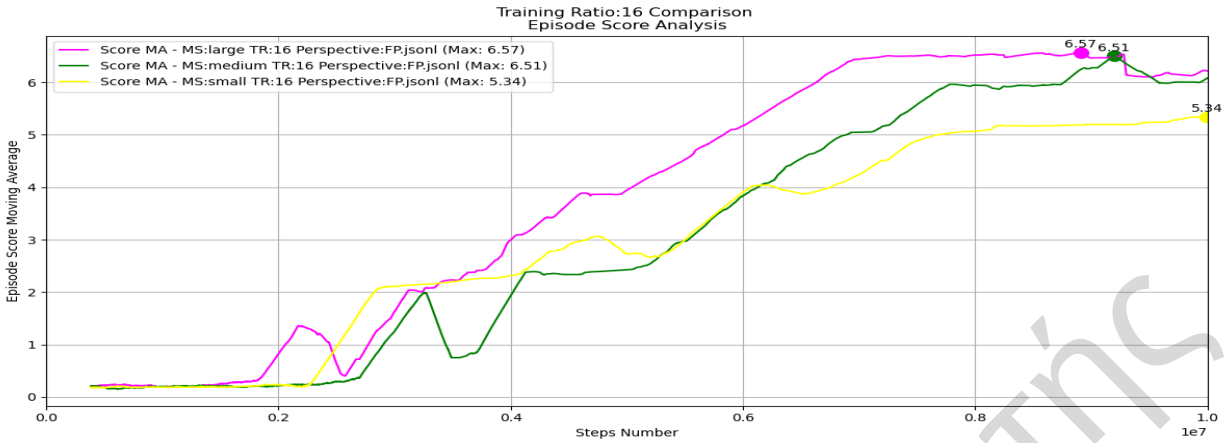
Για τη σύγκριση των αποτελεσμάτων της πειραματικής διάταξης A.1, πραγματοποιήθηκαν δύο βασικές κατηγορίες σύγκρισης, με βάση τη γραφική παράσταση του κινούμενου μέσου όρου του σκορ ως προς τον άξονα X, που αντιπροσωπεύει το σύνολο των βημάτων που εκτέλεσε ο πράκτορας κατά τη διάρκεια της εκπαίδευσής του. Οι δύο κύριες κατηγορίες είναι το "Model size" και το "Training Ratio". Για το "Model size", πραγματοποιήθηκαν τρεις βασικές

συγκρίσεις για τα μεγέθη "small", "medium" και "large" του μοντέλου. Επιπλέον, για το "Training Ratio", υπήρχαν δύο βασικές συγκρίσεις, μία για το ¨Training Ratio=16" και μία για το "Training Ratio=64". Ο κύριος στόχος ήταν να διαπιστωθεί ποιο μέγεθος μοντέλου και ποιο training ratio οδηγούν σε υψηλότερη τελική απόδοση, εκφραζόμενη μέσω του υψηλότερου σκορ, καθώς και στην αποδοτικότητα των δεδομένων αλλά και το εάν αλληλεπιδρούν μεταξύτους αυτές οι παραμέτροι. Αυτή η αξιολόγηση επικεντρώνεται στην επαλήθευση των αναμενομένων αποτελεσμάτων. Εδώ πρέπει να σημειωθεί ότι όλες οι παρακάτω συγκρίσεις αφορούν τα πρώτα 10 εκατομμύρια βήματα που πραγματοποίησε ο πράκτορας στο περιβάλλον Obstacle Tower κατά τη διάρκεια της εκπαίδευσής του. Αυτό συνέβη κυρίως λόγω έλλειψης υπολογιστικών πόρων και της χρονοβόρας εκτέλεσης κάθε πειραματικής διάταξης. Παράλληλα, αυτές οι συγκρίσεις παρείχαν μια σχετικά ξεκάθαρη εικόνα για την εξαγωγή συμπερασμάτων.Οι γραφικές παραστάσεις που αναπαριστούν τα αποτελέσματα της πειραματικής διάταξης A, και επάνω στις οποίες θα βασιστεί η ερμηνεία, παρουσιάζονται στο Σχήμα 5.1



Σχήμα 5.1: Παρουσίαση όλων των αποτελεσμάτων του κινούμενου μέσου όρου του σκορ της Πειραματικής Διάταξης A σε Σχέση με το Σύνολο των Βημάτων που Εκτέλεσε ο Πράκτορας Κατά τη Διάρκεια της Εκπαίδευσής του, Παρουσιαζόμενο σε Διάφορα Διαγράμματα. Στη Δεξιά Στήλη Εμφανίζονται τα Σκορ που Πέτυχε κάθε Μοντέλο (Small, Medium, Large) με Training Ratio:16, ενώ στην Αριστερή Στήλη Εμφανίζονται τα Αντίστοιχα Σκορ για το Training Ratio:64.

Κατηγορία Σύγκρισης "Training Ratio"



Σχήμα 5.2: Παρουσίαση των Αποτελεσμάτων του Κινούμενου Μέσου Όρου του Σκορ που Πέτυχε το Κάθε Μοντέλο (Small, Medium, Large) με Training Ratio:16 της Πειραματικής Διάταξης Α σε Σχέση με το Σύνολο των Βημάτων που Εκτέλεσε ο Πράκτορας Κατά τη Διάρκεια της Εκπαίδευσής του.

Από το Σχήμα 5.2, όπου παρουσιάζονται οι γραφικές απεικονίσεις του κινούμενου μέσου όρου του σκορ σε σχέση με τον συνολικό αριθμό βημάτων που εκτέλεσε ο πράκτορας στο περιβάλλον, για τα μοντέλα μεγέθους Small, Medium, Large με Training Ratio:16, παρατηρούμε τα εξής:

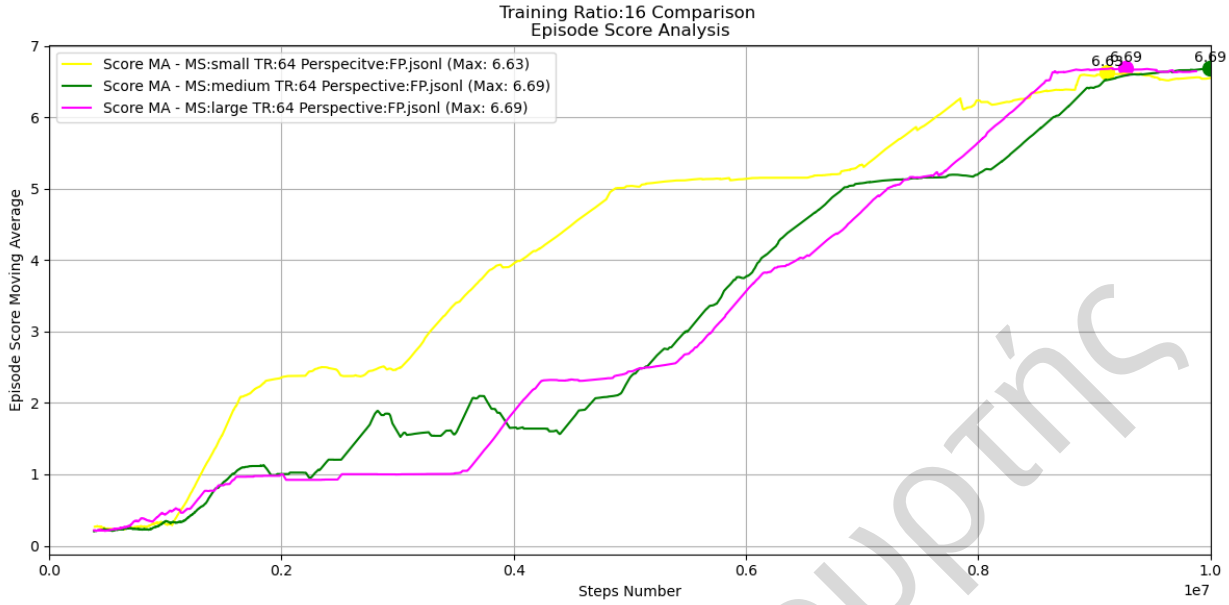
1.Το μοντέλο μεγέθους Large εμφανίζει υψηλότερη ταχύτητα μάθησης και βελτίωσης του σκορ σε σύγκριση με τα άλλα δύο μοντέλα, καταφέρνοντας να επιτύχει υψηλότερη αποδοτικότητα δεδομένων. Συγκεκριμένα, φτάνει στην υψηλότερη τελική απόδοση, περίπου μετά από 7 εκατομμύρια βήματα, σε αντίθεση με τα μοντέλα Small και Medium, τα οποία είτε δεν καταφέρνουν να φτάσουν καθόλου σε αυτήν την απόδοση είτε καθυστερούν πολύ.

2. Όσον αφορά την υψηλότερη τελική απόδοση, τα μοντέλα μεγέθους Medium και Large παρουσιάζουν σχεδόν ίδιες τιμές, δηλαδή 6.51 και 6.57 αντίστοιχα. Αντίθετα, το μοντέλο μεγέθους Small υστερεί σημαντικά, επιτυγχάνοντας ελάχιστα μικρότερο σκορ των 5.34.

Συμπέρασμα:

1. Συγκρίνοντας τα αποτελέσματα μας με το Σχήμα 3.2, επαληθεύεται ότι όσο πιο μεγάλο το μοντέλο, τόσο πιο υψηλή αποδοτικότητα δεδομένων και υψηλότερη τελική απόδοση επιτυγχάνεται, αν και η διαφορά μεταξύ του μοντέλου Medium και Large ως προς την είναι υψηλότερη τελική απόδοση είναι αρκετά κοντινή.

2. Για την επίτευξη των 10 εκατομμυρίων βημάτων απαιτήθηκαν περίπου 48 ώρες.



Σχήμα 5.3: Παρουσίαση των αποτελεσμάτων του κινούμενου μέσου όρου του σκορ που πέτυχε το κάθε μοντέλο (Small, Medium, Large) με Training Ratio:64 της πειραματικής διάταξης A σε σχέση με το σύνολο των βημάτων που εκτέλεσε ο πράκτορας κατά τη διάρκεια της εκπαίδευσής του.

Από το Σχήμα 5.3, όπου παρουσιάζονται οι γραφικές παραστάσεις του κινούμενου μέσου όρου του σκορ σε σχέση με τον συνολικό αριθμό βημάτων που εκτέλεσε ο πράκτορας στο περιβάλλον, για τα μοντέλα μεγέθους Small, Medium, Large με Training Ratio:64, παρατηρούμε τα εξής:

1. Το μοντέλο μεγέθους Small εμφανίζει αρχικά υψηλότερη ταχύτητα μάθησης και βελτίωσης του σκορ σε σύγκριση με τα άλλα δύο μοντέλα, επιτυγχάνοντας υψηλότερη αποδοτικότητα δεδομένων. Πιο συγκεκριμένα, πριν από τα 6 εκατομμύρια βήματα, αυτή η διαφορά είναι αισθητά πιο έντονη.

2. Όσον αφορά την υψηλότερη τελική απόδοση, τα μοντέλα μεγέθους Medium και Large έχουν την ίδια τιμή, συγκεκριμένα 6.69, ενώ το μοντέλο Medium επιτυγχάνει ελάχιστα μικρότερο σκορ, που ανέρχεται σε 6.63.

Συμπέρασμα:

1. Συγκρίνοντας τα αποτελέσματα μας με το Σχήμα3.2, ερχόμαστε σε αντίθεση με το γεγονός ότι όσο πιο μεγάλο το μοντέλο, τόσο υψηλότερη αποδοτικότητα δεδομένων επιτυγχάνεται. Όσον αφορά την υψηλότερη τελική απόδοση, δεν μπορούμε να πούμε ότι όσο μεγαλύτερο το

μέγεθος του μοντέλου, τόσο υψηλότερη είναι η τιμή αυτή, αφού τα τρία μεγέθη έχουν αν όχι ίδια, τουλάχιστον παρόμοια τελική τιμή.

2. Επίσης, σημειώνεται ότι για την επίτευξη των 10 εκατομμυρίων βημάτων σε κάθε πείραμα απαιτήθηκε περίπου 60 ώρες.

Σύγκριση μεταξύ των Σχημάτων 5.2 και 5.3:

1. Ξεκινώντας από την υψηλότερη τελική απόδοση, παρατηρούμε ότι με τη μετάβαση από το Training Ratio 16 στο 64, επιτυγχάνεται βελτίωση σε όλα τα μοντέλα. Συγκεκριμένα, τα μοντέλα Medium και Large παρουσιάζουν αύξηση της τάξης του 0.1 με 0.2, ενώ το μοντέλο Small σημειώνει σημαντική ανάπτυξη, από 5.34 σε 6.63.

2. Όσον αφορά την απόδοση δεδομένων, παρατηρήθηκε ότι με τη μετάβαση από το Training Ratio 16 στο 64, η αποδοτικότητα των μοντέλων εξελίσσεται ανάλογα με το μέγεθός τους. Πιο συγκεκριμένα, στα μεγαλύτερα μοντέλα μειώνεται η αποδοτικότητα δεδομένων, ενώ στα μικρότερα αυξάνεται η αποδοτικότητα δεδομένων. Αυτό φαίνεται σαφώς στη σύγκριση των γραφικών αναπαραστάσεων για τα Training Ratios 16 και 64. Στο γράφημα με Training Ratio 16, το μοντέλο Large εμφανίζει τη μεγαλύτερη αποδοτικότητα, το Medium τη μεσαία, και το Small τη χαμηλότερη. Στην πειραματική διάταξη με Training Ratio 64, παρατηρείται αντιστροφή της τάσης. Το Small μοντέλο εμφανίζει σημαντική αύξηση στην αποδοτικότητα δεδομένων, ξεπερνώντας τα άλλα δύο, το Large μοντέλο μειώνει αισθητά την απόδοση του επιφέροντας επιφέροντας την μικρότερη απόδοση δεδομένων, ενώ το Medium μοντέλο παρουσιάζει ελάχιστη μείωση στην αποδοτικότητα δεδομένων, παραμένοντας η ενδιάμεση απόδοση.

3. Συγκρίνοντας τη γραφική αναπαράσταση του μοντέλου μεγέθους Large με Training Ratio 16 και τη γραφική αναπαράσταση του μοντέλου μεγέθους Small με Training Ratio 64, παρατηρούμε ότι, ενώ αρχικά το Small επιδεικνύει πιο γρήγορο ρυθμό μάθησης, μετά από τα 3 εκατομμύρια βήματα το Large κατορθώνει να διατηρήσει έναν υψηλότερο ρυθμό μάθησης μέχρι το τέλος. Παρότι το Small size μοντέλο επιτυγχάνει την υψηλότερη τελική απόδοση, η διαφορά μεταξύ τους μπορεί να θεωρηθεί αμελητέα.

Γενικά Συμπέρασμα για την Κατηγορία Σύγκρισης "Training Ratio":

Για την απόδοτικότητα δεδομένων:

1. Υψηλότερες τιμές στο Training Ratio αυξάνουν την αποδοτικότητα δεδομένων για μικρότερα μοντέλα, ενώ χαμηλότερες δεν το επιτυγχάνουν αυτό.

2. Υψηλότερες τιμές στο Training Ratio μειώνουν την αποδοτικότητα δεδομένων στα μεγαλύτερα μοντέλα, ενώ χαμηλότερες τιμές παρέχουν μια αύξηση στην απόδοτικότητα δεδομένων.

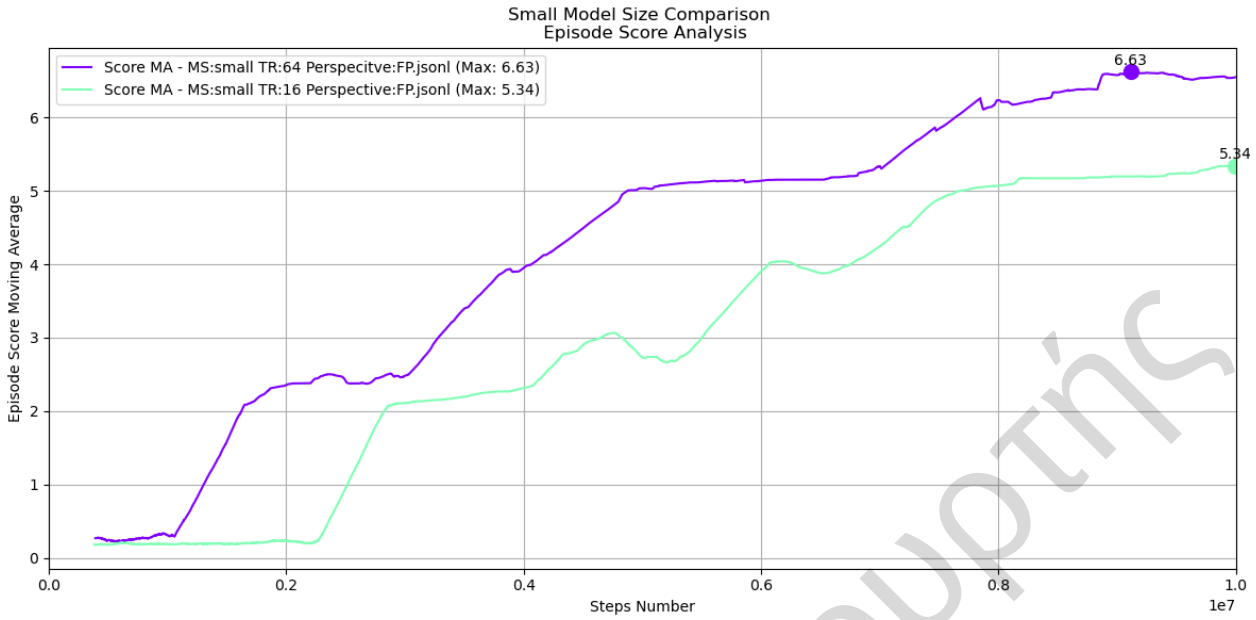
Για την τελικη απόδοση:

1. Υψηλότερες τιμές στο Training Ratio αυξάνουν την τελική απόδοση για όλα τα μεγέθη μοντέλων, αλλά η αύξηση είναι πιο έντονη για τα μικρότερα. Αντίστροφα, χαμηλότερες τιμές μειώνουν την τελική απόδοση για όλα τα μεγέθη μοντέλων.

Ο βέλτιστος συνδυασμός Model Size και Training Ratio:

Συγκρίνοντας τις δύο καλυτερες αποδόσεις, δηλαδή το Large (TR:16) με το Small (TR:64), το μεγαλύτερο διατηρεί υψηλότερη αποδοτικότητα δεδομενων μετά από τα 3 εκατομμύρια βήματα. Παρότι το μικρότερο μοντέλο φτάνει σε υψηλότερη τελική απόδοση, η διαφορά μεταξύ τους θεωρείται αμελητέα. Επιπλέον, γενικά το Training Ratio:16 απαιτεί λιγότερο χρόνο εκτέλεσης για την επίτευξη των 10 εκατομμυρίων βημάτων (48 ώρες) σε σχέση με το Training Ratio:64 (60 ώρες).

Κατηγορία Σύγκρισης "Model Size"



Σχήμα 5.4: Παρουσίαση των αποτελεσμάτων του κινούμενου μέσου όρου του σκορ που πέτυχε το Small μοντέλο με Training Ratio:16, 64 της πειραματικής διάταξης A σε σχέση με το σύνολο των βημάτων που εκτέλεσε ο πράκτορας Κατά τη διάρκεια της εκπαίδευσής του.

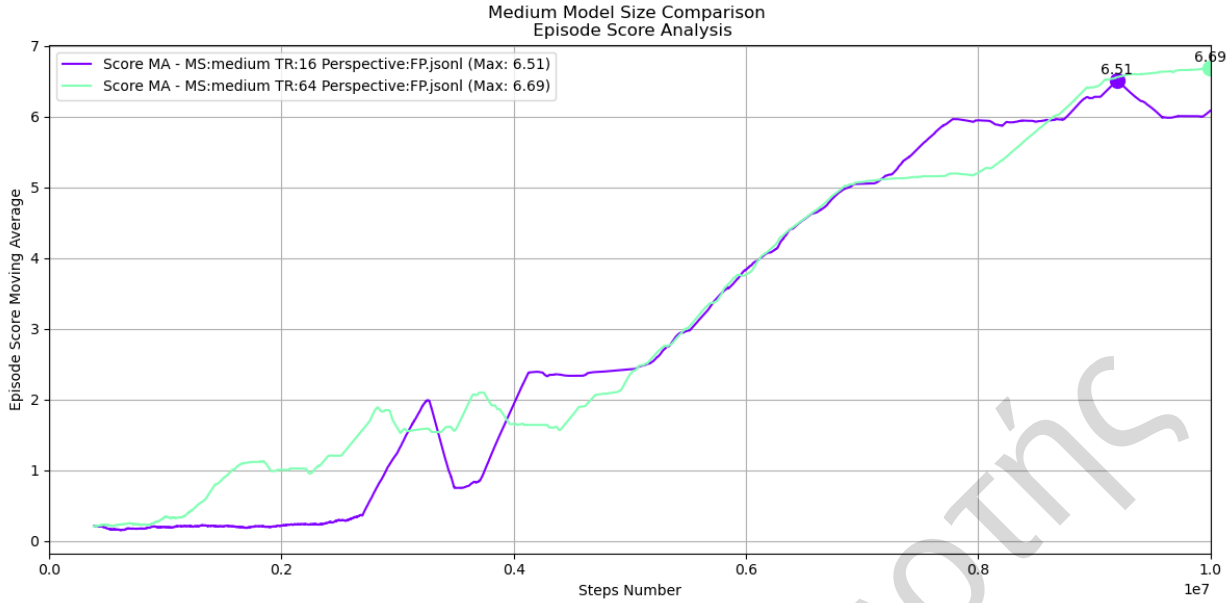
Από το Σχήμα 5.4, όπου παρουσιάζονται οι γραφικές απεικονίσεις του κινούμενου μέσου όρου του σκορ σε σχέση με τον συνολικό αριθμό βημάτων που εκτέλεσε ο πράκτορας στο περιβάλλον, για το μοντέλο μεγέθους Small με Training Ratio:16 και Training Ratio:64, παρατηρούμε τα εξής:

1. Το μοντέλο με Training Ratio:64 επιδεικνύει σημαντικά υψηλότερη ταχύτητα εκπαίδευσης, δηλαδή υψηλότερη αποδοτικότητα δεδομένων και βελτίωσης σκορ σε σύγκριση με το μοντέλο με Training Ratio:16. Συγκεκριμένα, φτάνει στην υψηλότερη τελική απόδοση, περίπου στα 6.63 σκορ στα 9 εκατομμύρια βήματα, ενώ το μοντέλο με Training Ratio:16 φτάνει μόνο τα 5.34.

Συμπέρασμα:

1. Παρατηρούμε ότι το μοντέλο με Training Ratio:64 επιφέρει τόσο υψηλότερη αποδοτικότητα δεδομένων όσο και υψηλότερη τελική απόδοση.

2. Το χρονικό διάστημα που απαιτείται για την επίτευξη των 10 εκατομμυρίων βημάτων διαφέρει, καθώς το μοντέλο με Training Ratio:16 απαιτεί περίπου 48 ώρες, ενώ το μοντέλο με Training Ratio:64 απαιτεί περίπου 60 ώρες.



Σχήμα 5.5: Παρουσίαση των αποτελεσμάτων του κινούμενου μέσου όρου του σκορ που πέτυχε το Medium μοντέλο με Training Ratio:16, 64 της πειραματικής διάταξης Α σε σχέση με το σύνολο των βημάτων που εκτέλεσε ο πράκτορας κατά τη διάρκεια της εκπαίδευσής του.

Από το Σχήμα 5.5, όπου παρουσιάζονται οι γραφικές απεικονίσεις του κινούμενου μέσου όρου του σκορ σε σχέση με τον συνολικό αριθμό βημάτων που εκτέλεσε ο πράκτορας στο περιβάλλον, για το μοντέλο μεγέθους Medium με Training Ratio:16 και Training Ratio:64, παρατηρούμε τα εξής:

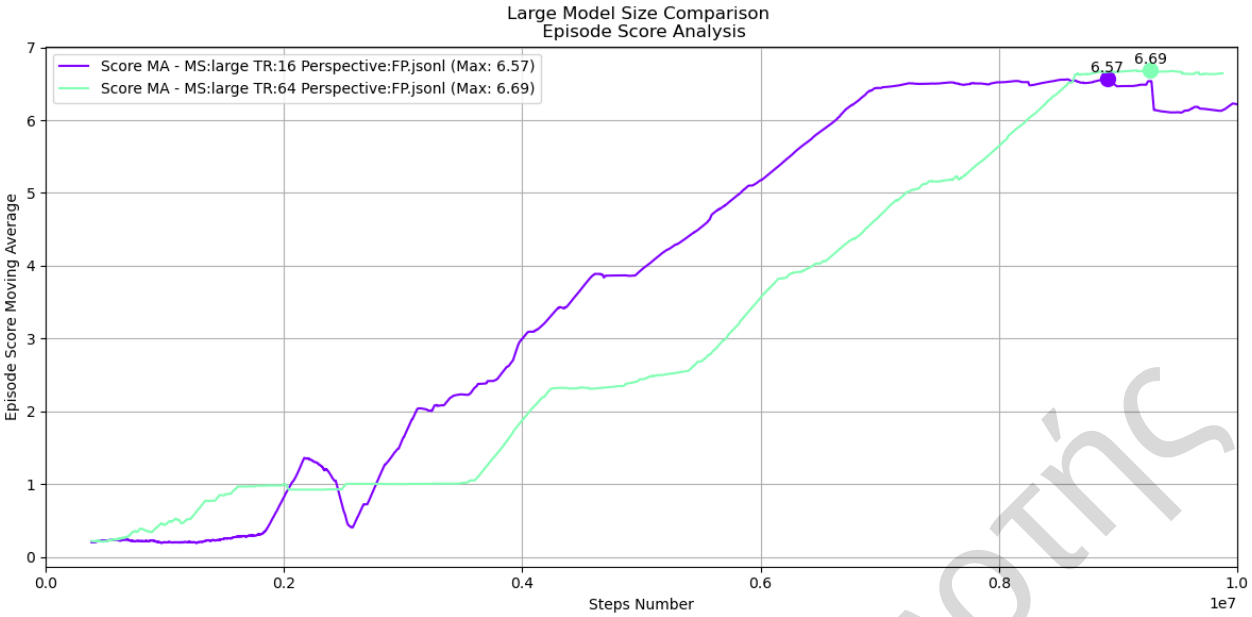
1. Το μοντέλο με Training Ratio:64 επιτυγχάνει ελάχιστα υψηλότερη τελική απόδοση, και συγκεκριμένα 6.69 σκορ στα 10 εκατομμύρια βήματα, ενώ το μοντέλο με Training Ratio:16 φτάνει τα 6.51 γύρω στα 9.5 εκατομμύρια βήματα.

2. Όσον αφορά την υψηλότερη αποδοτικότητα δεδομένων, δεν διακρίνεται κάποια ουσιαστική διαφορά μεταξύ των δύο.

Συμπέρασμα:

1. Παρατηρούμε ότι τα δύο μοντέλα με διαφορετικές τιμές Training Ratio δεν εμφανίζουν κάποια ουσιαστική διαφορά μεταξύ τους.

2. Το χρονικό διάστημα που απαιτείται για την επίτευξη των 10 εκατομμυρίων βημάτων διαφέρει, καθώς το μοντέλο με Training Ratio:16 απαιτεί περίπου 48 ώρες, ενώ το μοντέλο με Training Ratio:64 απαιτεί περίπου 60 ώρες.



Σχήμα 5.6: Παρουσίαση των αποτελεσμάτων του κινούμενου μέσου όρου του σκορ που πέτυχε το Large μοντέλο με Training Ratio:16, 64 της πειραματικής διάταξης A σε σχέση με το σύνολο των βημάτων που εκτέλεσε ο πράκτορας κατά τη διάρκεια της εκπαίδευσής του.

Από το Σχήμα 5.6, όπου παρουσιάζονται οι γραφικές απεικονίσεις του κινούμενου μέσου όρου του σκορ σε σχέση με τον συνολικό αριθμό βημάτων που εκτέλεσε ο πράκτορας στο περιβάλλον, για το μοντέλο μεγέθους Large με Training Ratio:16 και Training Ratio:64, παρατηρούμε τα εξής:

1. Το μοντέλο με Training Ratio:16 επιτυγχάνει ελάχιστα υψηλότερη τελική απόδοση, και συγκεκριμένα 6.69 σκορ γύρω στα 9.5 εκατομμύρια βήματα, ενώ το μοντέλο με Training Ratio:64 φτάνει τα 6.57 γύρω στα 9.5 εκατομμύρια βήματα.

2. Όσον αφορά την υψηλότερη αποδοτικότητα δεδομένων, το μοντέλο με Training Ratio:64 κατορθώνει να επιτύχει ξεκάθαρα υψηλότερη απόδοση δεδομένων, καθώς ήδη από τα 7 εκατομμύρια βήματα παρουσιάζει σχεδόν την υψηλότερη τελική απόδοση, ενώ το μοντέλο με Training Ratio:16 το πετυχαίνει κοντά στα 9 εκατομμύρια βήματα.

Συμπέρασμα:

1. Παρατηρούμε ότι τα δύο μοντέλα με διαφορετικές τιμές TR δεν εμφανίζουν κάποια ουσιαστική διαφορά μεταξύ τους ως προς την υψηλότερη τελική απόδοση, ενώ ως προς υψηλότερη αποδοτικότητα δεδομένων το TR:16 επιτύχει ξεκάθαρα υψηλότερη απόδοση.

2. Το χρονικό διάστημα που απαιτείται για την επίτευξη των 10 εκατομμυρίων βημάτων διαφέρει, καθώς το μοντέλο με TR:16 απαιτεί περίπου 48 ώρες, ενώ το TR:64 περίπου 60.

Γενικά Συμπέρασμα για την Κατηγορία Σύγκρισης "Model Size"

1.Συγκρίνοντας τα μοντέλα Small, Medium, Large όσον αφορά την υψηλότερη τελική απόδοση και την υψηλότερη αποδοτικότητα δεδομένων με τη μεταβολή του Training Ratio από 16 σε 64, παρατηρούμε τα εξής:

1. Στο μοντέλο μεγέθους Small με Training Ratio=64, παρατηρείται υψηλότερη τελική απόδοση και υψηλότερη αποδοτικότητα δεδομένων.

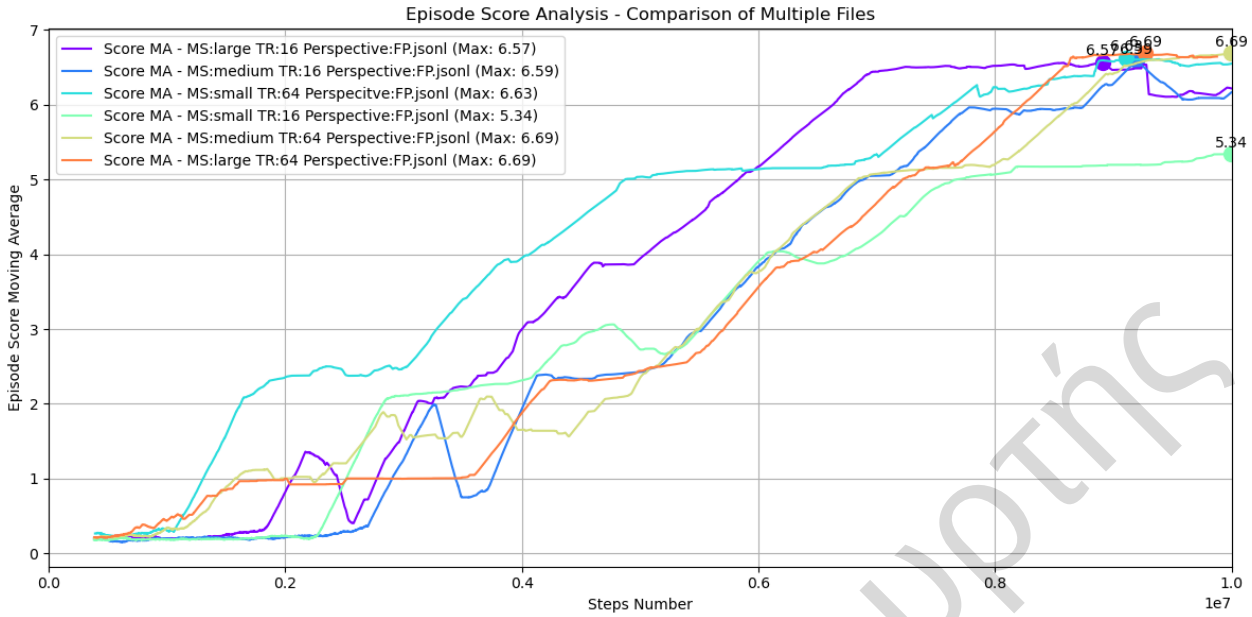
2. Στο μοντέλο μεγέθους Medium, δεν υπάρχει καμία ουσιαστική διαφορά μεταξύ του Training Ratio 16 και 64 όσον αφορά την υψηλότερη τελική απόδοση και υψηλότερη αποδοτικότητα δεδομένων.

3. Στο μεγάλο μοντέλο Large με Training Ratio=16, παρατηρείται υψηλότερη αποδοτικότητα δεδομένων, ενώ η υψηλότερη τελική απόδοση παραμένει σχεδόν η ίδια σε και τις δύο περιπτώσεις.

4. Βάση των πιο πάνω επαληθεύεται το συμπέρασμα από την κατηγορία σύγκρισης "Training Ratio" ότι όσο αυξάνεται η τιμή του Training Ratio σε μεγάλα μοντέλα, τόσο μειώνεται η αποδοτικότητα των δεδομένων και το αντίστροφο. Επιπλέον, παρατηρείται αύξηση στην υψηλότερη τελική απόδοση σε όλα τα μεγέθη μοντέλων, ειδικά στα μικρότερα.

5. Για την επίτευξη των 10 εκατομμυρίων βημάτων σε κάθε πείραμα, παρατηρείται ότι με τον ρυθμό εκπαίδευσης TR:64 απαιτήθηκαν περίπου 60 ώρες. Αντίστοιχα, με τον ρυθμό εκπαίδευσης TR:16, η επίτευξη των 10 εκατομμυρίων βημάτων απαιτούσε περίπου 48 ώρες. Επομένως, το Training Ratio=16 στο μοντέλο Large εκτός του ότι μας επιφέρει υψηλότερη αποδοτικότητα δεδομένων απαιτεί και λιγότερο χρόνο εκτέλεσης.

Γενικά Συμπεράσαμτα για την Πειραματική Διάταξη Α.1



Σχήμα 5.7: Παρουσίαση όλων των αποτελεσμάτων του κινούμενου μέσου όρου του σκορ της πειραματικής διάταξης Α σε σχέση με το σύνολο των βημάτων που εκτέλεσε ο πράκτορας κατά τη διάρκεια της εκπαίδευσής του στο ίδιο διάγραμμα.

Απο τα αποτελέσματα της πειραματικής διάταξης Α.1 που λάβαμε επαληθεύσαμε ότι:

1. Υψηλότερες τιμές του Training Ratio υποστηρίζουν τη βελτίωση της τελικής απόδοσης και της αποδοτικότητας δεδομένων για τα μικρότερα μοντέλα (Model size). Ωστόσο, καθώς αυξάνεται το μέγεθος του μοντέλου, η αύξηση και των δύο μειώνεται αισθητά.

2. Υψηλότερες τιμές του Training Ratio επηρεάζουν αρνητικά την αποδοτικότητα δεδομένων των μεγαλύτερων μοντέλων (Model size). Κατά τη μείωση του Training Ratio, παρατηρείται αύξηση της αποδοτικότητας δεδομένων. Όσον αφορά την τελική απόδοση, παρατηρείται ελάχιστη αύξηση όσο αυξάνεται η τιμή του Training Ratio.

3. Συγκεκριμένα, από τα αποτελέσματά μας προέκυψε ότι το μοντέλο μεγέθους Large με TR:16 και το Small με TR:64 επιδεικνύουν τις υψηλότερες αποδοτικότητες δεδομένων, καθώς και μία από τις υψηλότερες τελικές αποδόσεις.

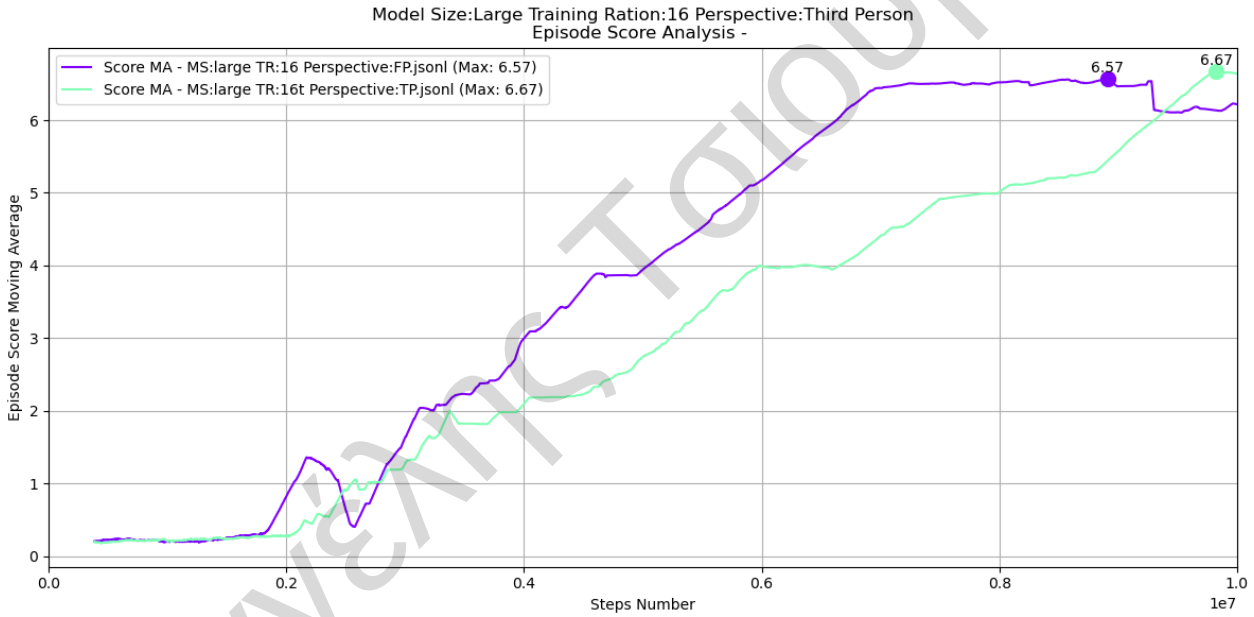
4. Το Training Ratio=16, για την επίτευξη των 10 εκατομμυρίων βημάτων, απαιτεί περίπου 48 ώρες, ενώ το Training Ratio=64 περίπου 60 ώρες.

Συνεπώς, ο συνδυασμός του μοντέλου Large με το Training Ratio=16, εκτός από την υψηλότερη αποδοτικότητα δεδομένων, επιφέρει και μία από τις υψηλότερες τελικές τιμές, ενώ απαιτεί και λιγότερο χρόνο εκτέλεσης.

Ερμηνεία Αποτελεσμάτων Περιματικής Διάταξης Α.2

Βασιζόμενοι στο συμπέρασμα ότι ο συνδυασμός του μοντέλου Large με το Training Ratio=16 οδηγεί σε υψηλότερη αποδοτικότητα δεδομένων και μία από τις υψηλότερες τελικές τιμές, ενώ απαιτεί λιγότερο χρόνο εκτέλεσης, εφαρμόστηκε στην πειραματική διάταξη Α.2 με την μόνη διαφορά ότι σε αυτή τη διάταξη, το Agent-perspective τροποποιήθηκε από 0 σε 1, επιφέροντας την παρατήρηση του περιβάλλοντος από την οπτική γωνία ενός τρίτου προσώπου αντί από τη δική του.

Για την ανάλυση των αποτελεσμάτων της πειραματικής διάταξης Α.2, προβαίνουμε σε σύγκριση της απόδοσης του κινούμενου μέσου όρου του σκορ στην Α.2 με εκείνη της Α.1, χρησιμοποιώντας το μοντέλο Large με το Training Ratio=16.



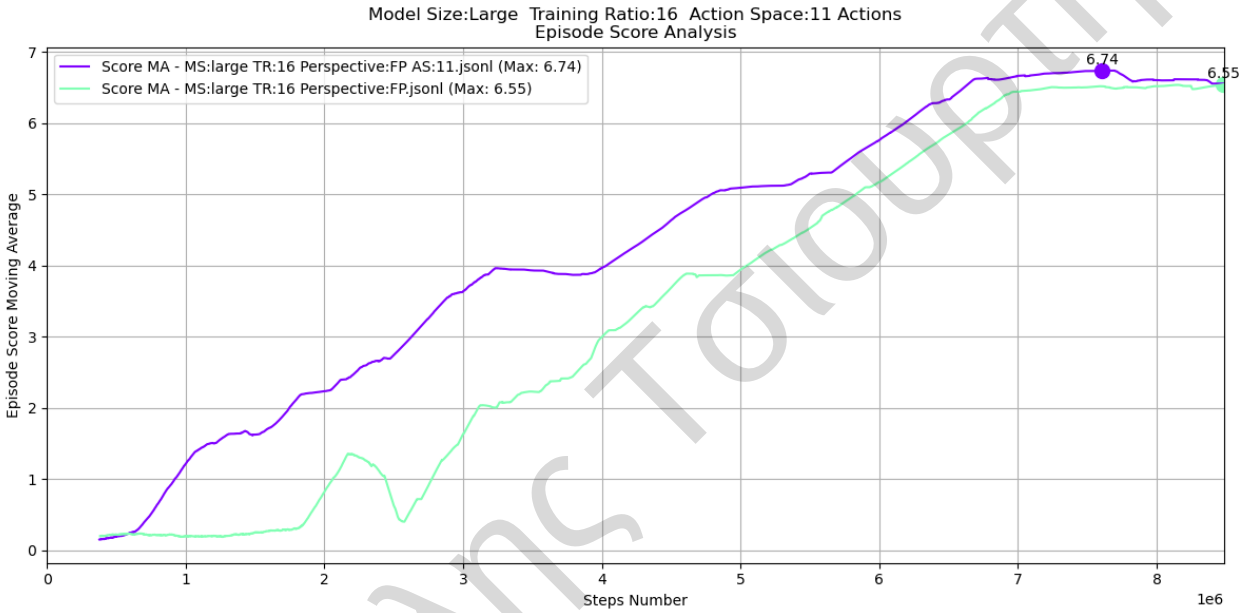
Σχήμα 5.8: Παρουσίαση αποτελεσμάτων του κινούμενου μέσου όρου σκορ για τις Διατάξεις Α.2 με τιμή παραμέτρου Agent-perspective=0 και Α.1 με τιμή παραμέτρου Agent-perspective=1, χρησιμοποιώντας το μοντέλο Large με Training Ratio=16, σε σχέση με τον συνολικό αριθμό των βημάτων που εκτελέστηκαν από τον πράκτορα κατά τη διάρκεια της εκπαίδευσής του.

Γενικά Συμπεράσαμτα για την Πειραματική Διάταξη Α.2

Από τα αποτελέσματα της πειραματικής διάταξης Α.2 που απεικονίζονται στο Σχήμα5.8, προκύπτει ότι με τη ρύθμιση της παραμέτρου Agent-perspective σε 1, υπάρχει μια μείωση στην αποδοτικότητα δεδομένων και ελάχιστη αύξηση στην τελική τιμή, η οποία δεν θεωρείται σημαντική. Έτσι, ο συνδυασμός του μοντέλου Large με το Training Ratio=16 και Agent-perspective=0 επιφέρει τα πιο επιθυμητά αποτελέσματα μέχρι στιγμής.

Ερμηνεία Αποτελεσμάτων Πειραματικής Διάταξης Α.3

Βασιζόμενοι στο συμπέρασμα από τις πειραματικές διατάξεις Α.1 και Α.2, που δείχνουν ότι ο συνδυασμός του μοντέλου Large με το Training Ratio=16 και Agent-perspective=0 παρέχει τα πιο επιθυμητά αποτελέσματα, εφαρμόστηκε στην πειραματική διάταξη Α.3. Η μόνη διαφορά σε σχέση με τις προηγούμενες διατάξεις είναι ότι σε αυτήν αφαιρέθηκαν όλοι οι πιθανοί συνδυασμοί κινήσεων που περιλαμβάνουν τη δεξιά, αριστερά και προς τα πίσω κατεύθυνση από το action space, μειώνοντας τον αριθμό των δυνατών κινήσεων από 54 σε 11.



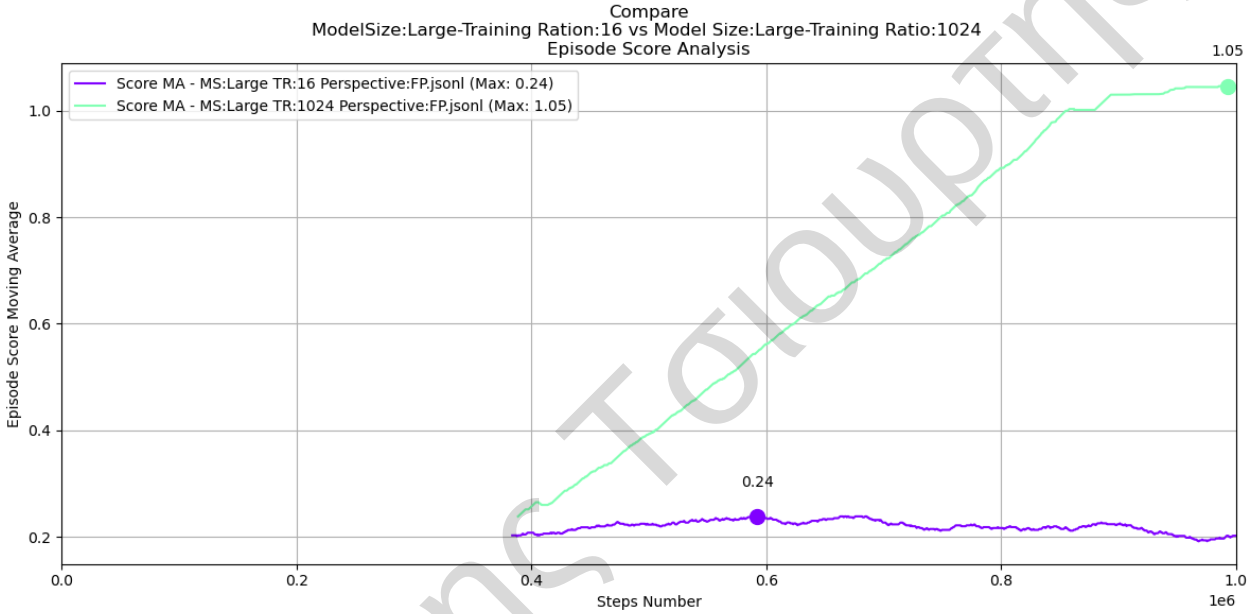
Σχήμα 5.9: Παρουσίαση αποτελεσμάτων του κινούμενου μέσου όρου σκορ για τις διατάξεις Α.3 με Action Space=11 και Α.1 με Action Space=54, χρησιμοποιώντας το μοντέλο Large με Training Ratio=16, σε σχέση με τον συνολικό αριθμό των βημάτων που εκτελέστηκαν από τον πράκτορα κατά τη διάρκεια της εκπαίδευσής του.

Γενικά Συμπεράσαμτα για την Πειραματική Διάταξη Α.3

Από τα αποτελέσματα της πειραματικής διάταξης Α.3 που απεικονίζονται στο Σχήμα5.9, προκύπτει ότι η μείωση του αριθμού των δυνατών κινήσεων στο action space από 54 σε 11 οδηγεί σε μια αισθητή αύξηση της αποδοτικότητας δεδομένων και σε ελάχιστη αύξηση της τελικής τιμής, η οποία δεν θεωρείται σημαντική. Έτσι, ο συνδυασμός του μοντέλου Large με το Training Ratio=16, Agent-perspective=0 και το μειωμένο action space παράγει τα πιο επιθυμητά αποτελέσματα μέχρι στιγμής.

Ερμηνεία Αποτελεσμάτων Περιματικής Διάταξης Α.4

Βασιζόμενοι στο συμπέρασμα που προκύπτει από τις πειραματικές διατάξεις Α.1 και Α.2, όπου καταδεικνύεται ότι ο βέλτιστος συνδυασμός μοντέλου Large, Training Ratio=16 και Agent-perspective=0 επιφέρει από τα πιο επιθυμητά αποτελέσματα, αποφασίσαμε να εφαρμόσουμε αυτόν τον συνδυασμό στην πειραματική διάταξη Α.4. Σημειώνεται πως η μόνη διαφορά σε σχέση με τις προηγούμενες διατάξεις είναι η χρήση της τιμής 1024 για το Training Ratio.



Σχήμα 5.10: Παρουσίαση αποτελεσμάτων του κινούμενου μέσου όρου σκορ για τις διατάξεις Α.3 με τιμή Training Ratio=1024 και Α.1 τιμή Training Ratio=16 , χρησιμοποιώντας το μοντέλο Large με Training Ratio=16, σε σχέση με τον συνολικό αριθμό των βημάτων που εκτελέστηκαν από τον πράκτορα κατά τη διάρκεια της εκπαίδευσής του.

Γενικά Συμπεράσαμτα για την Πειραματική Διάταξη Α.4

Τα αποτελέσματα της πειραματικής διάταξης Α.4, που φαίνονται στο Σχήμα 5.10, αποκαλύπτουν ότι η σημαντική αύξηση του Training Ratio οδηγεί όχι μόνο σε σημαντική ενίσχυση της αποδοτικότητας δεδομένων, αλλά και σε αυξημένη τελική τιμή. Αυτή η αύξηση θεωρείται σημαντική, που μας φέρνει ενάντια ως προς τα προηγούμενα συμπεράσματά μας. Σημειώνεται ότι στο Σχήμα 5.10, ο αριθμός των βημάτων φτάνει έως ένα εκατομμύριο, διότι η χρήση τόσο υψηλού Training Ratio απαιτούσε περισσότερες από 72 ώρες για την εκτέλεσή τους, καθιστώντας σχεδόν αδύνατη τη χρήση του.

Γενικά Συμπεράσαμτα Απο τις Πειραματικές Διατάξεις

Με την ολοκλήρωση των πειραματικών διατάξεων και την ανάλυση των συλλεγόμενων δεδομένων, καταλήγουμε στα ακόλουθα συμπεράσματα:

1. Μέγεθος Μοντέλου με Training Ratio:

1. Ο συνδυασμός μεγαλύτερων μοντέλων (Model Size) με χαμηλές τιμές Training Ratio επιδεικνύει υψηλότερη αποδοτικότητα δεδομένων. Αντίθετα, ο συνδυασμός μεγαλύτερων μοντέλων (Model Size) με υψηλές τιμές Training Ratio οδηγεί σε χαμηλή αποδοτικότητα δεδομένων. Σε και τις δύο περιπτώσεις, η τελική απόδοση είναι από τις υψηλότερες, με τη δεύτερη περίπτωση να δείχνει ελάχιστα υψηλότερη.

2. Ο συνδυασμός μικρότερων μοντέλων (Model Size) με μεγάλες τιμές Training Ratio επιδεικνύει υψηλότερη αποδοτικότητα δεδομένων. Αντίθετα, ο συνδυασμός μικρότερων μοντέλων (Model Size) με μικρές τιμές Training Ratio οδηγεί σε χαμηλή αποδοτικότητα δεδομένων. Στην πρώτη περίπτωση του συνδυασμού μικρότερων μοντέλων με μεγάλες τιμές Training Ratio, η τελική απόδοση αυξάνεται αισθητά σε αντίθεση με την δεύτερη περίπτωση.

2. Agent Perspective:

Η ρύθμιση της παραμέτρου Agent-perspective σε Third Person (1) αντί First Person (0) οδηγεί σε μείωση της αποδοτικότητας δεδομένων, χωρίς σημαντική αύξηση της τελικής τιμής.

3. Συνδυασμός Παραμέτρων:

Ο συνδυασμός του μοντέλου Large με το Training Ratio=16 και Agent-perspective=0 επιφέρει τα πιο επιθυμητά αποτελέσματα.

4. Αριθμός Κινήσεων στο Action Space:

Η μείωση του αριθμού των δυνατών κινήσεων στο action space από 54 σε 11 οδηγεί σε αύξηση της αποδοτικότητας δεδομένων.

5. Χρόνος Εκπαίδευσης:

Η αύξηση του Training Ratio συνεπάγεται αυξημένο χρόνο επίτευξης αριθμού βημάτων στο περιβάλλον.

6. Σχολιασμός του Average Reward και του Μέσου Όρου Ορόφων

Δεν ήταν δυνατόν να επιτευχθεί τουλάχιστον μία από τις πειραματικές διατάξεις αποτέλεσμα με μέσο Reward > 35.86 και μέσο όρο ορόφων (Average Floors) > 19.4. Οι μέγιστες τιμές που καταγράφηκαν προήλθαν από την πειραματική διάταξη A.3. Αυτή η διάταξη χρησιμοποιεί ένα μοντέλο μεγέθους Large, Training Ratio=16, Agent-perspective=0, και μειωμένο action space=11, επιτυγχάνοντας (Average Floors)=6 όροφο με Average Reward=6.74.

5.3 Περιορισμοί και Προκλήσεις

Η παρούσα ενότητα παρέχει μια διαφανή συζήτηση σχετικά με τους περιορισμούς που αντιμετωπίστηκαν στη μεταπτυχιακή εργασία. Αυτό περιλαμβάνει πτυχές όπως η διαθεσιμότητα δεδομένων, οι υπολογιστικοί πόροι, και άλλοι περιορισμοί που ενδέχεται να επηρεάσουν τη γενίκευση των αποτελεσμάτων της μελέτης.

Προκλήσεις:

Η ενσωμάτωση του αλγορίθμου DreamerV3 στο περιβάλλον του παιχνιδιού Obstacle Tower δεν αποδείχθηκε ιδιαίτερα εύκολη, καθώς οι προκλήσεις που αναδύθηκαν σε κάθε βήμα της διαδικασίας υπογράμμισαν τη σημασία της σχολαστικής προσέγγισης και της αφιέρωσης επαρκούς χρόνου σε κάθε μία από αυτές. Οι κύριες προκλήσεις που προέκυψαν είναι οι εξής:

1. **Συμβατότητα Εκδόσεων και Πακέτων Python:**

   Η ανάγκη να επιλεγεί το κατάλληλο Python version==3.8.2 και οι συμβατές εκδόσεις πακέτων για τον αλγόριθμο DreamerV3, που δημοσιεύτηκε το 2023, με πιο πρόσφατα πακέτα από το Obstacle Tower του 2019, απαιτούσε προσεκτική διαχείριση.

2. **Έλλειπής Πληροφόριση για τον Κώδικα DreamerV3 (Όχι Documentetion):**

   Η έλλειψη επαρκούς πληροφόρισης για το κώδικα του αλγορίθμου DreamerV3 μας έθεσε αντιμέτωπους με πολλά ερωτήματα και προβλήματα. Για την αντιμετώπιση αυτών, απαιτήθηκε σημαντικός χρόνος για την κατανόηση του κώδικα, προκειμένου να επιτευχθεί η ορθή ενσωμάτωση και προσαρμογή στο περιβάλλον του Obstacle Tower.

3. **Αντιμετώπιση Σφαλμάτων (Bugs)**:

   Συγκεκριμένα, ο κώδικας του Dreamerv3 δεν υποστήριζε παιχνίδια με διαφορετικές διαστάσεις του visual observation εκτός από 64x64. Επιπλέον, ο αλγόριθμος Dreamerv3 δεν υποστηρίζει περιβάλλοντα που δεν είναι της μορφής **retro**. Έγινε μια προσπάθεια για τροποποίηση του κώδικα για να αντιμετωπιστούν αυτά τα προβλήματα, ωστόσο μόνο το πρώτο πρόβλημα έγινε κατορθωτό να επιλυθεί.

   Η ανίχνευση και επίλυση αυτών των σφαλμάτων κατά τη διάρκεια της υλοποίησης απαιτούσε πρόσθετο χρόνο και προσοχή και αποτέλεσε σημαντικό κομμάτι της διαδικασίας.

4. **High-Performance Computing (HPC)**

   Επιπλέον, η προσπάθεια εκτέλεσης πειραμάτων στο High-Performance Computing (HPC) με τον DreamerV3 στο παιχνίδι Obstacle Tower δεν επετεύχθη, ενώ η εκτέλεση πειραμάτων του DreamerV3 στο παιχνίδι Crafter κατορθώθηκε. Αυτό υπογραμμίζει ξανά τη σημασία της συμβατότητας των εκδόσεων των πακέτων. Μια πιθανή λύση για αυτό θεωρείται μέσω του Dockerfile.

Περιορισμοί: Εκτός από τις προκλήσεις που αντιμετωπίστηκαν, οι περιορισμοί που προέκυψαν σε αυτήν τη μεταπτυχιακή εργασία ήταν αρκετοί, κυρίως λόγω της φύσης του πεδίου της Μηχανικής Μάθησης και συγκεκριμένα της BEM.

1. **Περιορισμός Μνήμης Κάρτας Γραφικών (GPU Memory Constraint)**

   Ξεκινώντας από την αρχή, παρά τις προσπάθειες τροποποίησης του κώδικα του Dreamerv3 για να προσαρμοστεί σε περιβάλλοντα με διαστάσεις διαφορετικές από τις προεπιλεγμένες των 64x64, όπως στο Obstacle Tower με διαστάσεις 84x84, η προσπάθεια απέτυχε λόγω έλλειψης επαρκούς διαθέσιμης μνήμης στην κάρτα γραφικών. Ως αποτέλεσμα, ήμασταν υποχρεωμένοι να μειώσουμε τις διαστάσεις της visual observation σε 64x64, συμπράττοντας με την αναγκαία απώλεια πληροφοριών.

2. **Περιορισμός Εκτέλεσης Πειραμάτων Λόγο Έλληψης Υπολογιστικών Πόρων**

   Όπως αναφέρθηκε και προιγουμένος δέν κατέστη εφικτό να εκτελέσουμε πειραματικές διατάξεις στο HPC μεταξύ του Dreamerv3 και του Obstacle Tower με αποτέλεσμα να περιοριστούμε στην εκτέλεση των πειραματικών διατάξεων μας σε ένα μόνο υπολογιστή. Ως επακόλουθο αυτό μας έθεσε αντιμέτωπους με αρκετούς περιορισμούς :

   1. Δεν ήταν εφικτό να τρέξουμε παράλληλα πολλαπλές πειραματικές διατάξεις και επομένος εκτελέστηκαν σειριακά η μία μετά την αλλη σε έναν υπολογιστή.

   2. Η RAM στον υπολογιστή έχοντας την παράμετρο tower-seed=-1 δεν ήταν επαρκής για της πειραμτικές μας διατάξεις καθώς γέμιζε σε πρόιμο στάδιο σε σχεέση με τον αριθμών των βημάτων που εκτελούσε μέχρι εκείνη την χρονική στιγμή ο πράκτορας και πρίν κάν δούμε κάποια ουσιαστική βελτίωση του στην εκπαίδευση (2,5 εκατομμύρια βήματα για το model size= XL και μέχρι 7 εκατομμύρια βηματα για το Medium).

   Έτσι, δοκιμάζοντας διάφορες αλλαγές τόσο στις παραμέτρους του DreamerV3 (π.χ.,

rssm(deter, units), encoder, decoder (mlp layers, mlp units, cnn depth)) όσο και στις παραμέτρους του Obstacle Tower (π.χ., train-mode, tower-seed, dense-reward), συνειδητοποιήσαμε ότι η επιλογή σταθερού seed οδηγεί σε βελτίωση του DreamerV3. Ως αποτέλεσμα, θέσαμε τη μεταβλητή tower-seed=1.

3. **Χρονοβόρα Εκτέλεση Πειραματικών Διατάξεων**

Παρόλο που αυξήσαμε τον ρυθμό εκπαίδευσης του πράκτορα στα 20 Hz, κάθε πειραματική διάταξη αποδείχθηκε σημαντικά χρονοβόρα (48 έως 72 ώρες κάθε μία για 10-17 εκατομμύρια βήματα). Αυτό οδήγησε σε σημαντικές καθυστερήσεις στην παραγωγή αποτελεσμάτων.

Αυτή η καθυστέρηση είχε αρνητική επίδραση στην αποδοτικότητα της έρευνας, καθώς δεν υπήρχε δυνατότητα τροποποίησης των παραμέτρων κατά τη διάρκεια της εκτέλεσης των πειραμάτων. Επίσης, δεν ήταν δυνατή η αξιολόγηση του κώδικα με τις συγκεκριμένες παραμέτρους πριν από ένα μεγάλο αριθμό βημάτων, που στα αρχικά πειράματα ήταν ακόμη άγνωστος.

Έτσι, προτού καταλήξουμε στην κατάλληλη παραμετροποίηση (όπως το break=20, seed=1, κλπ.) και όχι σε προσωρινές ρυθμίσεις, απαιτήθηκε πολύς χρόνος.

Κεφάλαιο 6

Ανασκόπηση και Μελλοντική Εργασία (Conclusion and Future Work)

6.1 Εισαγωγή

Με την ολοκλήρωση των πειραματικών διατάξεων και την ανάλυση των συλλεγόντων δεδομένων μέσω αυτών, η μεταπτυχιακή εργασία ολοκληρώθηκε. Σκοπός του τελευταίου κεφαλαίου είναι η καταγραφή του γενικού αποτελέσματος και των συμπερασμάτων της μεταπτυχιακής. Επίσης, θα γίνει ανασκόπηση της όλης έρευνας που έγινε μέχρι τώρα και θα προταθούν μελλοντικές προοπτικές.

6.2 Ανασκόπηση και Συμπεράσματα

Αυτή η μεταπτυχιακή μελέτη ανέδειξε τη σημαντική συνεισφορά του αλγορίθμου Dreamer V3 στην αντιμετώπιση των πολύπλοκων προκλήσεων του περιβάλλοντος Obstacle Tower. Η ομαλή ενσωμάτωση του αλγορίθμου στο περιβάλλον παιχνιδιού Obstacle Tower επιτεύχθηκε μέσω της εκμετάλλευσης των ιδιοτήτων του όσον αφορά την προσαρμοστικότητα και την επεκτασιμότητα του. Διεξάχθηκαν εννέα πειραματικές διατάξεις μεταβάλλοντας τις παραμέτρους του αλγορίθμου Dreamer V3: το Training Ratio σε 16 και 64, το Model Size σε Small, Medium και Large. Επίσης, μεταβλήθηκαν οι παράμετροι του Obstacle Tower Agent Perspective σε First και Third person και το Action Space από 54 κινήσεις σε 11.

Η ανάλυση των αποτελεσμάτων που προέκυψαν από τις πειραματικές διατάξεις βασίστηκε στα εξής κριτήρια: υψηλότερη τελική απόδοση, αποδοτικότητα των δεδομένων, μέσος αριθμός ορόφων και μέσο reward. Ταυτόχρονα εξετάστηκε εάν ο Dreamer V3 έχει τη δυνατότητα να υπερβεί την απόδοση άλλων κορυφαίων αλγορίθμων στο περιβάλλον Obstacle Tower.

Τα αποτελέσματά παρουσιάζουν μια ξεκάθαρη εικόνα της βελτίωσης τόσο του μέσου αριθμού ορόφων όσο και του μέσου reward που επιτυγχάνει ο αλγόριθμος Dreamer V3 κατά την διάρκεια της αλληλεπίδρασης του μέσα στο περιβάλλον.

Μέσα από τη σύγκριση των αποτελεσμάτων των πειραματικών διατάξεων προκύπτει το συμπεράσματα ότι η χρήση μεγαλύτερων μοντέλων και του Training Ratio=16 αποτελεί βέλτιστη πρακτική για την απόδοση του αλγορίθμου σε πολύπλοκα περιβάλλοντα. Ωστόσο, αυτές οι παράμετροι δεν οδηγούν απαραίτητα σε υψηλότερη τελική απόδοση, όπως αναμενόταν. Η παράμετρος Agent-perspective ως First person καθώς και η μείωση του Action space από 54 κινήσεις σε 11 βελτιώνουν αισθητά την αποδοτικότητα δεδομένων και ελαφρώς την τελική απόδοση, υπογραμμίζοντας τη σημασία της παραμετροποίησης για την αποτελεσματικότητα του αλγορίθμου.

Ο συνδυασμός ενός μοντέλο Large, Training Ratio=16, Agent perspective στο First Person, και μείωση του Action space σε 11 επιφέρουν την υψηλότερη αποδοτικότητα δεδομένων και τελική τιμή κατορθώνοντας να φτάσουν μέχρι τον 6ο όροφο με Average Reward=6.74.

Όσον αφορά τον δεύτερο στόχο, δεν κατέστη δυνατό να ξεπεραστούν οι επιδόσεις των προηγούμενων κορυφαίων αλγορίθμων, με μέσο όρο ορόφων=19.4 και μέσο Reward=35.86. Αυτό οφείλεται κυρίως στον περιορισμένο χρόνο εκπαίδευσης του αλγορίθμου, δεδομένου ότι πα-

ραχωρήθηκαν μόνο 10 εκατομμύρια βήματα εκπαίδευσης, τα οποία, τελικά, αποδείχθηκαν λίγα.

Συνολικά, η έρευνά μας παρέχει σημαντικές ενδείξεις σχετικά με τη δυνατότητα επίτευξης υψηλότερων αποδόσεων από τον αλγόριθμο Dreamer V3 στο παιχνίδι Obstacle Tower. Αυτό προϋποθέτει είτε επιπλέον εκπαίδευση του αλγορίθμου σε αυτό το περιβάλλον, δηλαδή τον αλγόριθμο Dreamer V3 να αλληλεπιδρά με το περιβάλλον Obstacle Tower για περισσότερα από δέκα εκατομμύρια βήματα, όπως πραγματοποιήθηκε στη μεταπτυχιακή εργασία, είτε την εκπαίδευση του σε κάθε πολύπλοκη πρόκληση-υποδιαδικασία του περιβάλλοντος Obstacle Tower ξεχωριστά και στη συνέχεια σε όλο το περιβάλλον συνολικά.

6.3 Μελλοντικές Προοπτικές-Επεκτασιμότητα

Στο πλαίσιο της περαιτέρω ανάπτυξης της έρευνας, εξετάζουμε μελλοντικές προοπτικές και δυνατότητες επεκτασιμότητας. Αναζητούμε την επιβεβαίωση των δεδομένων και την ενσωμάτωση διακύμανσης με αυξημένο αριθμό επαναλήψεων, ενώ εξετάζουμε επίσης τη χρήση μεγάλου μοντέλου και την επίδραση διαφορετικών παραμέτρων στην απόδοση. Παράλληλα, εξετάζουμε τη δυνατότητα απόκτησης υποδομών όπως High-Performance Computing και γραφικών καρτών με μεγαλύτερη μνήμη για την ενσωμάτωση περισσότερων παραμέτρων και τη βελτίωση της συνολικής απόδοσης. Αυτές οι επεκτάσιμες προσεγγίσεις αντανακλούν τη δέσμευσή μας στη συνεχή εξέλιξη της έρευνας στην εμβαθυμένη μηχανική μάθηση στον τομέα των παιχνιδιών.

Πιο συγκεκριμένα, όσον αφορά τις πειραματικές διατάξεις, παραθέτουμε αναλυτικά τις διάφορες προσεγγίσεις που θα μπορούσαν να προσφέρουν μια εκτενέστερη κατανόηση της απόδοσης του αλγορίθμου Dreamerv3 στο παιχνίδι Obstacle Tower.

1. Αρχικά, θα ήταν καλό να εκτελούνται οι πειραματικές διατάξεις 1.1, 1.2, 1.3 σε έναν ικανοποιητικό αριθμό επαναλήψεων (π.χ., 10 φορές το καθένα), ώστε να επαληθευτούν τα δεδομένα και να ενσωματωθεί και η διακύμανση σε αυτά. Για να επιτευχθεί αυτό, θα απαιτηθεί η χρήση High-Performance Computing (HPC), επιτρέποντας την ταυτόχρονη εκτέλεση πολλών πειραμάτων.

2. Υλοποίηση πειραματικών διατάξεων χρησιμοποιώντας ένα μοντέλο μεγάλου μεγέθους (Model Size XL, TR=16, 32, 64). Για αυτή την πειραματική διάταξη θα απαιτηθούν

μεγάλες ποσότητες RAM.

3. Λαμβάνοντας υπόψη την πειραματική διάταξη που θα επιφέρει την υψηλότερη απόδοση απο τα πιο πάνω, θα ήταν επιθυμητό να εκτελεστεί χρησιμοποιώντας ένα μεγάλο αριθμό seeds (π.χ., 40) και έναν εκτεταμένο αριθμό βημάτων εκπαίδευσης (100 εκατομμύρια). Ο κύριος στόχος είναι να επαληθευτεί η γενικότητα και η απόδοση του DreamerV3. Σημειώστε ότι αυτή η πειραματική διάταξη θα απαιτήσει επίσης σημαντικούς πόρους RAM.

4. Υλοποίηση της πειραματικής διάταξης τύπου Β.

5. Απόκτηση γραφικής κάρτας με μεγαλύτερη μνήμη για να ενσωματωθεί το Obstacle Tower με διαστάσεις 84x84. Είναι δύσκολο να προβλέψουμε το κατα πόσο αυτή η αλλαγή θα επηρεάσει την επίδοση θετικά, αλλά είναι μια προσπάθεια που αξίζει να εξεταστεί.

Bibliography

[1] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[2] O. OpenAI Five, "Dota 2 with large scale deep reinforcement learning6," Dec 2019. [Online]. Available: https://openai.com/research/dota-2-with-large-scale-deep-reinforcement-learning

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[4] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," *arXiv preprint arXiv:2301.04104*, 2023.

[5] D. Ha and J. Schmidhuber, "World models," Mar 2018. [Online]. Available: https://worldmodels.github.io/

[6] A. Juliani, A. Khalifa, V.-P. Berges, J. Harper, E. Teng, H. Henry, A. Crespi, J. Togelius, and D. Lange, "Obstacle tower: A generalization challenge in vision, control, and planning," *arXiv preprint arXiv:1902.01378*, 2019.

[7] A. Juliani and J. Shih, "Announcing the obstacle tower challenge winners and open source release," Aug 2019. [Online]. Available: https://blog.unity.com/news/announcing-obstacle-tower-challenge-winners-and-open-source-release

[8] Α. Γεωργούλη, "Τεχνητή νοημοσύνη," 2016.

[9] Sep 2018. [Online]. Available: https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[12] Z. SALLOUM, "Reinforcement learning policy for developers," Dec 2021. [Online]. Available: https://medium.com/towards-data-science/revisiting-policy-in-reinforcement-learning-for-developers-43cd2b713182

[13] L. Weng, "A (long) peek into reinforcement learning," Feb 2018. [Online]. Available: https://lilianweng.github.io/posts/2018-02-19-rl-overview/

[14] J. Rocca, "The exploration-exploitation trade-off: Intuitions and strategies," May 2021. [Online]. Available: https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbe1e82

[15] S. Bhatt, "5 things you need to know about reinforcement learning," Mar 2018. [Online]. Available: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

[16] C. Szepesv´ari, Jun 2009. [Online]. Available: https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf

[17] M. Bhowmik, "Deep recurrent q-network," Jan 2019. [Online]. Available: https://marl-ieee-nitk.github.io/deep-reinforcement-learning/2019/01/06/DRQN.html

[18] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] C. t. W. projects, "Partially observable markov decision process," Mar 2023. [Online]. Available: https://en-m-wikipedia-org.translate.goog/wiki/Partially_observable_Markov_decision_process?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=el

[21] B. Roy, "Introduction to reinforcement learning," Feb 2023. [Online]. Available: https://baijayanta.medium.com/start-now-reinforcement-learning-a7450c2e67c1

[22] L. Owen, "Bird's-eye view of reinforcement learning algorithms taxonomy," Oct 2020. [Online]. Available: https://towardsdatascience.com/birds-eye-view-of-reinforcement-learning-algorithms-landscape-2aba7840211c

[23] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[24] "On-policy v/s off-policy learning," Jul 2020. [Online]. Available: https://towardsdatascience.com/on-policy-v-s-off-policy-learning-75089916bc2f

[25] Z. SALLOUM, "How to think in dynamic programming," May 2017. [Online]. Available: https://zsalloum.medium.com/how-to-think-in-dynamic-programming-3f6804a79429

[26] B. Roy, "Monte carlo learning," Feb 2023. [Online]. Available: https://towardsdatascience.com/monte-carlo-learning-b83f75233f92

[27] Z. SALLOUM, "Monte carlo in reinforcement learning, the easy way," Dec 2021. [Online]. Available: https://zsalloum.medium.com/monte-carlo-in-reinforcement-learning-the-easy-way-564c53010511

[28] ——, "Function approximation in reinforcement learning," Dec 2021. [Online]. Available: https://towardsdatascience.com/function-approximation-in-reinforcement-learning-85a4864d566

[29] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017.

[30] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[31] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International conference on machine learning*. PMLR, 2019, pp. 2555–2565.

[32] M. Hutsebaut-Buysse, K. Mets, and S. Latré, "Hierarchical reinforcement learning: A survey and open research challenges," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 172–221, 2022.

[33] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.

[34] Y. Flet-Berliac, "The promise of hierarchical reinforcement learning," Dec 2021. [Online]. Available: https://thegradient.pub/the-promise-of-hierarchical-reinforcement-learning/

[35] J. J. Ribas-Fernandes, A. Solway, C. Diuk, J. T. McGuire, A. G. Barto, Y. Niv, and M. M. Botvinick, "A neural signature of hierarchical reinforcement learning," *Neuron*, vol. 71, no. 2, pp. 370–379, 2011.

[36] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete event dynamic systems*, vol. 13, no. 1-2, pp. 41–77, 2003.

[37] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[38] I. Goodfellow *et al.*, "Deep learning-ian goodfellow, yoshua bengio, aaron courville," *Adapt. Comput. Mach. Learn.*, 2016.

[39] M. M. Willcock, "Mythological paradeigma in the iliad," *The Classical Quarterly*, vol. 14, no. 2, pp. 141–154, 1964.

[40] Y. Frumer, "The short, strange life of the first friendly robot," May 2020. [Online]. Available: https://spectrum.ieee.org/the-short-strange-life-of-the-first-friendly-robot#toggle-gdpr

[41] E. C. Berkeley, "Giant brains; or, machines that think.." 1949.

[42] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, 10 1950. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433

[43] T. M. Mitchell, "Does machine learning really work?" *AI magazine*, vol. 18, no. 3, pp. 11–11, 1997.

[44] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[45] Jul 2023. [Online]. Available: https://en.wikipedia.org/wiki/Reticular_theory

[46] Α. Γεωργούλη, "Τεχνητή νοημοσύνη," 2016.

[47] Feb 2022. [Online]. Available: https://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CE%AC%CE%BE%CE%BF%CE%BD%CE%B1%CF%82

[48] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[49] A. Huxley, "Hodgkin and the action potential 1935–1952," *The Journal of physiology*, vol. 538, no. Pt 1, p. 2, 2002.

[50] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[51] C. Meunier and I. Segev, "Playing the devil's advocate: is the hodgkin–huxley model useful?" *Trends in neurosciences*, vol. 25, no. 11, pp. 558–563, 2002.

[52] A. F. Strassberg and L. J. DeFelice, "Limitations of the hodgkin-huxley formalism: Effects of single channel kinetics on transmembrane voltage dynamics," *Neural computation*, vol. 5, no. 6, pp. 843–855, 1993.

[53] E. Schneidman, B. Freedman, and I. Segev, "Ion channel stochasticity may be critical in determining the reliability and precision of spike timing," *Neural computation*, vol. 10, no. 7, pp. 1679–1703, 1998.

[54] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological cybernetics*, vol. 95, no. 1, pp. 1–19, 2006.

[55] W. Gerstner, "1.3 Integrate-And-Fire Models | Neuronal Dynamics online book — neuronaldynamics.epfl.ch," https://neuronaldynamics.epfl.ch/online/Ch1.S3.html.

[56] A. Delorme, L. Perrinet, and S. J. Thorpe, "Networks of integrate-and-fire neurons using rank order coding b: Spike timing dependent plasticity and emergence of orientation selectivity," *Neurocomputing*, vol. 38, pp. 539–545, 2001.

[57] "File:Leaky Integrate-and-Fire model neuron (schematic).jpg - Wikimedia Commons — commons.wikimedia.org," https://commons.wikimedia.org/wiki/File:Leaky_Integrate-and-Fire_model_neuron_(schematic).jpg.

[58] R. Pramoditha, "The concept of artificial neurons (perceptrons) in neural networks," Dec 2021. [Online]. Available: https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc

[59] M. Minsky and S. Papert, "An introduction to computational geometry," *Cambridge tiass., HIT*, vol. 479, no. 480, p. 104, 1969.

[60] ύ, "έ έ ί perceptron." [Online]. Available: https://www.cs.uoi.gr/~arly/courses/nn/slides/K2.pdf

[61] W. Tong, L. Li, X. Zhou, A. Hamilton, and K. Zhang, "Deep learning pm 2.5 concentrations with bidirectional lstm rnn," *Air Quality, Atmosphere & Health*, vol. 12, pp. 411–423, 2019.

[62] J. Guliyev, "Gradient descent:," Jan 2021. [Online]. Available: https://medium.com/analytics-vidhya/gradient-descent-b0dc1af33517

[63] V. Vryniotis, "Machine learning blog amp; software development news," Oct 2013. [Online]. Available: http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/

[64] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.

[65] S. Saha, "A comprehensive guide to convolutional neural networks-the eli5 way," Nov 2022. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[66] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[67] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[68] C. Tan, M. Šarlija, and N. Kasabov, "Spiking neural networks: Background, recent development and the neucube architecture," *Neural Processing Letters*, vol. 52, no. 2, pp. 1675–1701, 2020.

[69] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, p. 774, 2018.

[70] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[71] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[72] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[73] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[74] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[75] F. Yu, "Felix yu," Oct 2016. [Online]. Available: https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html

[76] K. Doshi, "Reinforcement learning explained visually (part 5): Deep q networks, step-by-step," Feb 2021. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-explained-visually-part-5-deep-q-networks-step-by-step-5a5317197f

[77] "Actor-critic - spinning up documentation." [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/sac.html

[78] V. V. PV, "Deep reinforcement learning: Value functions, dqn, actor-critic method, backpropagation through..." Aug 2020. [Online]. Available: https://medium.com/@vishnuvijayanpv/deep-reinforcement-learning-value-functions-dqn-actor-critic-method-backpropagation-throu

[79] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[80] A. Juliani, "Simple reinforcement learning with tensorflow part 4: Deep q-networks and beyond," Jul 2017. [Online]. Available: https://awjuliani.medium.com/simple-reinforcement-learning-with-tensorflow-part-4-deep-q-networks-and-beyond-8438a3e

[81] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015.

[82] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, 2010.

[83] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2016.

[84] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[85] T. Simonini, "Improvements in deep q learning: Dueling double dqn, prioritized experience replay, and fixed..."

Mar 2023. [Online]. Available: https://medium.com/free-code-camp/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-5

[86] W. v. Heeswijk, "Rainbow dqn-the best reinforcement learning has to offer?" Dec 2022. [Online]. Available: https://towardsdatascience.com/rainbow-dqn-the-best-reinforcement-learning-has-to-offer-166cb8ed2f86

[87] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," 2017.

[88] L. Chang and D. Y. Tsao, "The code for facial identity in the primate brain," *Cell*, vol. 169, no. 6, pp. 1013–1028, 2017.

[89] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried, "Invariant visual representation by single neurons in the human brain," *Nature*, vol. 435, no. 7045, pp. 1102–1107, 2005.

[90] G. B. Keller, T. Bonhoeffer, and M. Hübener, "Sensorimotor mismatch signals in primary visual cortex of the behaving mouse," *Neuron*, vol. 74, no. 5, pp. 809–815, 2012.

[91] M. Leinweber, D. R. Ward, J. M. Sobczak, A. Attinger, and G. B. Keller, "A sensorimotor circuit in mouse cortex for visual flow predictions," *Neuron*, vol. 95, no. 6, pp. 1420–1432, 2017.

[92] G. W. Maus, J. Fischer, and D. Whitney, "Motion-dependent representation of space in area mt+," *Neuron*, vol. 78, no. 3, pp. 554–562, 2013.

[93] D. Mobbs, C. C. Hagan, T. Dalgleish, B. Silston, and C. Prévost, "The ecology of human fear: survival optimization and the nervous system," *Frontiers in neuroscience*, vol. 9, p. 55, 2015.

[94] H. T. H. Giang, T. N. K. Hoan, P. D. Thanh, and I. Koo, "Hybrid noma/oma-based dynamic power allocation scheme using deep reinforcement learning in 5g networks," *Applied Sciences*, vol. 10, no. 12, p. 4236, 2020.

[95] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, "Mastering atari games with limited data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25 476–25 488, 2021.

[96] Oct 2023. [Online]. Available: https://www.minecraft.net/en-us/about-minecraft

[97] Jul 2023. [Online]. Available: https://www.searchlogistics.com/learn/statistics/minecraft-user-statistics/

[98] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[99] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and et al., "Mastering the game of go with deep neural networks and tree search," Jan 2016. [Online]. Available: https://www.nature.com/articles/nature16961

[100] Aug 2017. [Online]. Available: https://openai.com/research/more-on-dota-2

[101] O. OpenAI FIve, Aug 2017. [Online]. Available: https://openai.com/research/more-on-dota-

[102] ——, Aug 2017. [Online]. Available: https://openai.com/research/more-on-dota-2

[103] O. OpenAI Five, "Openai five benchmark: Results.4," Aug 2018. [Online]. Available: https://openai.com/research/openai-five-benchmark-results

[104] ——, "The international 2018: Results.5," Aug 2018. [Online]. Available: https://openai.com/research/the-international-2018-results

[105] M. Pleines, J. Jitsev, M. Preuss, and F. Zimmer, "Obstacle tower without human demonstrations: How far a deep feed-forward network goes with reinforcement learning," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 447–454.

[106] J. Booth, "Ppo dash: Improving generalization in deep reinforcement learning," *arXiv preprint arXiv:1907.06704*, 2019.

[107] J. Kuypers, "Exploring evolution strategies for reinforcement learning in the obstacle tower environment," Ph.D. dissertation, Universidade NOVA de Lisboa (Portugal), 2021.

[108] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[109] C. t. W. projects, "Cross-entropy method," Sep 2023. [Online]. Available: https://en-m-wikipedia-org.translate.goog/wiki/Cross-entropy_method?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=el

[110] Unixpickle, "Unixpickle/anyrl-py: A reinforcement learning framework," Feb 2017. [Online]. Available: https://github-com.translate.goog/unixpickle/anyrl-py?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=el

[111] A. unixpickle, "Prierarchy: Implicit hierarchies," Feb 2020. [Online]. Available: https://blog.aqnichol.com/2019/04/03/prierarchy-implicit-hierarchies/

[112] awjuliani and MarcoMeter, "Obstacle tower reset parameters," Jun 2020. [Online]. Available: https://github.com/Unity-Technologies/obstacle-tower-env/blob/master/reset-parameters.md

[113] [Online]. Available: https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf

[114] Oct 2023. [Online]. Available: https://www.techpowerup.com/gpu-specs/geforce-rtx-2080-ti-12-gb.c3938