ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σκακιστική μηχανή μέσω ενισχυτικής μάθησης

Ορέστης Δ. Γοργογιάννης
Ελευθέριος Η. Ζιώρης

Επιβλέπων: **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2021

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Reinforcement learning chess engine

Orestis D. Gorgogiannis
Eleftherios E. Zioris

Supervisor: **Panagiotis Stamatopoulos,** Assistant Professor

ATHENS

SEPTEMBER 2021

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σκακιστική μηχανή μέσω ενισχυτικής μάθησης

Ορέστης Δ. Γοργογιάννης
S.N.: 1115201700024
Ελευθέριος Η. Ζιώρης
S.N.: 1115201500043

ΕΠΙΒΛΕΠΩΝ: Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής

BSc THESIS

Reinforcement learning chess engine

Orestis D. Gorgogiannis
S.N.: 1115201700024
Eleftherios E. Zioris
S.N.: 1115201500043

SUPERVISOR: **Panagiotis Stamatopoulos,** Assistant Professor

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, η άνοδος της ενισχυτικής μάθησης την έχει εγκαθιδρύσει ώς την πιο προηγμένη μέθοδο στον τομέα των μηχανών παιχνιδιών βασισμένων σε νευρωνικά δίκτυα. Χάρη σε αυτήν, η εταιρία Deep Mind, δημιούργησε το AlphaGo και κατάφερε επίδοση πολύ μεγαλύτερη από ό,τι θεωρούνταν ότι είναι πιθανό για την εποχή. Μετά από λίγο καιρό, το βελτιωμένο AlphaGo Zero κατάφερε να ξεπεράσει ακόμα και αυτά τα αποτελέσματα, χρησιμοποιώντας μηδενική προϋπάρχουσα γνώση. Από τότε, οι μελέτες στην τεχνητή νοημοσύνη συγκεντρώθηκαν περισσότερο στην προσέγγιση της ενισχυτικής μάθησης. Σε αυτήν την εργασία επιχειρούμε να δημιουργήσουμε ένα μοντέλο μηχανικής μάθησης το οποίο μαθαίνει να παίζει σκάκι με μηδενική προϋπάρχουσα γνώση, βασισμένο στον αλγόριθμο εκπαίδευσης του AlphaGo Zero. Η διαφορές στα παιχνίδια σκάκι και Go δημιουργούν την ανάγκη για έναν πιο γενικευμένο αλγόριθμο, ο οποίος δεν εκμεταλλεύεται τις ιδιότητες των παιχνιδιών αυτών. Λαμβάνουμε επίσης υπ' όψιν τα όρια στα μηχανήματα που έχουμε στη διάθεσή μας και προσπαθούμε να επιτύχουμε όσο το δυνατόν καλύτερα αποτελέσματα σε περιορισμένο χρόνο και μνήμη κατά την εκπαίδευση. Με τα απολύτως απαραίτητα εργαλεία, το μοντέλο μας αναπτύσσει μια σταθερή διαδικασία εκπαίδευσης και σταδιακά βελτιώνει τον τρόπο παιξίματος, νικώντας τις προηγούμενες φάσεις του σε βάθος χρόνου.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μηχανική Μάθηση, Τεχνητή Νοημοσύνη, Ενισχυτική Μάθηση, Θεωρία παιγνίων

ABSTRACT

In recent years, reinforcement learning has risen to become the state of the art method in neural network based game engines. With the creation of AlphaGo, DeepMind achieved results that were thought to be unreachable in the near future. Then, AlphaGo Zero broke even that barrier, surpassing every other game engine using no prior domain knowledge. Since then, Artificial Intelligence studies have steered towards the pure reinforcement learning approach. In this paper, we attempt to create a model that learns to play chess without prior knowledge, using a generalized training procedure based on AlphaGo Zero. The difference in game rules between chess and Go create the need for a more general purpose algorithm, free from limitations the rules each different game demands. Modifications were also necessary to overcome the hardware limitations we face. Using the bare essentials for machine learning model training, our model is able to achieve a steady learning process, slowly getting better and beating its previous versions with each iteraion.

SUBJECT AREA: Machine Learning

KEYWORDS: Machine Learning, Artificial Intelligence, Reinforcement Learning, Game Theory

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα θέλαμε να ευχαριστήσουμε θερμά όλους τους φίλους, την οικογένεια, τους συγγενείς, τον επιβλέποντα καθηγητή Σταματόπουλο Παναγιώτη, αλλά και γενικότερα, όσους μας στήριξαν κατά τη εκπόνησή της.

CONTENTS

LIST OF FIGURES

ΠΡΟΛΟΓΟΣ

Η πτυχιακή αυτή εργασία πραγματοποιήθηκε στην Αθήνα, το ακαδημαϊκό έτος 2020-2021.

Σε αυτή την πτυχιακή εργασία γίνεται εκτεταμένη αναφορά στο παιχνίδι σκάκι, καθώς και σε νευρωνικά δίκτυα. Κάποιες βασικές πληροφορίες εξηγούνται παρακάτω, όμως συνίσταται ο αναγνώστης/στρια να έχει κάποια επιφανειακή επίγνωση των κανόνων του παιχνιδιού, καθώς και της λειτουργίας των νευρωνικών δικτύων.

1. ΕΙΣΑΓΩΓΗ

1.1 Η ιστορία του παιχνιδιού

Το σκάκι είναι από τα αρχαιότερα παιχνίδια στην ιστορία, σημειώνοντας ιστορία σχεδόν 1500 χρόνων. Τον 7ο αιώνα στην Ινδία παιζόταν μια πρώιμη μορφή του παιχνιδιού που ονομάζεται Chaturanga. Όμως η πραγματική προέλευση του παιχνιδιού είναι ακόμα άγνωστη. Στη σημερινή εποχή, το σκάκι έχει εξελιχθεί σε ένα από τα δημοφιλέστερα παιχνίδια στον κόσμο, με πάνω από 800 εκατομμύρια παίκτες. Ένα τόσο ευρέως διαδεδομένο παιχνίδι, είναι φυσικό να εξελίσσεται διαρκώς, με νέες στρατηγικές και τρόπους παιξίματος να δημιουργούνται διαρκώς. Στην πορεία αυτή είναι καθοριστική η εξέλιξη της τεχνητής νοημοσύνης, καθώς ένας υπολογιστής είναι εδώ και δεκαετίες, ικανός να ξεπεράσει κατά πολύ τις ανθρώπινες δυνατότητες στο παιχνίδι αυτό.

Η πρώτη φορά που ένας υπολογιστής ξεπέρασε τον άνθρωπο στο σκάκι, ήταν το 1997. Στη Νέα Υόρκη των ΗΠΑ, ο τότε παγκόσμιος πρωταθήτής στο σκάκι, Γκάρι Κασπάροβ (Garry Kasparov), χάνει από τον υπολογιστή της IBM που ονομαζόταν Deep Blue, σημειώνοντας την πρώτη νίκη του υπολογιστή επί του ικανότερου τότε παίκτη. Από τότε, νέα μοντέλα και μέθοδοι μηχανικής μάθησης έχουν εξελιχθεί ακόμα περισσότερο, ξεπερνώντας ολοκληρωτικά την ανθρώπινη νοημοσύνη. Στη σημερινή εποχή, κανένας άνθρωπος δεν μπορεί να συναγωνιστεί τους υπολογιστές στο παιχνίδι αυτό.

Επανάσταση έφερε επίσης η Deep Mind το 2017, με το μοντέλο AlphaZero. Βασισμένοι στο μοντέλο AlphaGo Zero [33], που έναν χρόνο πριν είχε ξεπεράσει εντελώς κάθε προηγούμενο μοντέλο στο παιχνίδι Go, το Alpha Zero με μόνο 9 ώρες εκπαίδευση και χωρίς να χρησιμοποιήσει βάση δεδομένων με παιχνίδια από έμπειρους παίκτες καταφέρνει να ξεπεράσει το μέχρι τότε επικρατέστερο μοντέλο, Stockfish 8, νικώντας το σε ένα σετ 100 παιχνιδιών με τελικό σκορ 28 νίκες, 0 ήττες και 72 ισοπαλίες. Η νίκη αυτή συνέβαλε στην αναγνώριση της ενισχυτικής μάθησης ως την επικρατέστερη μέθοδο όσον αφορά την απόδοση σε τέτοιου είδους παιχνίδια.

1.2 Επιδόσεις μηχανών στο σκάκι

Η ομάδα Deep Blue, αφού το μοντέλο της έχασε στην πρώτη αναμέτρηση με τον Garry Kasparov το 1996, γνώριζε ότι υπήρχαν ορισμένες ελλείψεις στο μοντέλο της, που έπρεπε να ξεπεραστούν, όπως τα κενά στη γνώση του παιχνιδιού και την ταχύτητα υπολογισμού. Έτσι οι Campbell, Hoane και Hsu σχεδίασαν ένα νέο και σημαντικά βελτιωμένο τσιπ με τις εξής αλλαγές:

1. Είχε μια πλήρως επανασχεδιασμένη λειτουργία αξιολόγησης, από «περίπου 6.400 χαρακτηριστικά σε πάνω από 8.000»
2. πρόσθεσε «ανίχνευση επανάληψης υλικού», που περιλάμβανε έναν αριθμό κατηγοριών για την δημιουργία εξειδικευμένων κινήσεων (π.χ. για τη δημιουργία όλων των κινήσεων που επιτίθενται στα κομμάτια του αντιπάλου)
3. Αύξησε την ταχύτητα αναζήτησης σε πάνω από 2 εκατομμύρια καταστάσεις ανά δευτερόλεπτο.

Το Deep Blue (1997) ήταν ένα μαζικά παράλληλο σύστημα, σχεδιασμένο για τη διεξαγωγή αναζητήσεων στο σκάκι. Η κατανομη της αρχιτεκτονικής του αποτελείται από 30 κόμβους (30 επεξεργαστές, ένας ανά κόμβο), τον υπολογιστή IBM RS/6000 SP, ο οποίος πραγματοποιούσε τη λήψη αποφάσεων υψηλού επιπέδου και 480 τσιπ για σκάκι (το καθένα μια μηχανή αναζήτησης σκακιού), με 16 τσιπ ανά επεξεργαστή SP. Τα 480 τσιπ λειτουργούσαν παράλληλα για να πραγματοποιούν

- βαθιές αναζητήσεις
- παραγωγή κινήσεων
- αξιολόγηση θέσης με πάνω από 8000 χαρακτηριστικά αξιολόγησης

Στη συνέχεια σχεδίασαν, δοκίμασαν και συντόνισαν τη νέα συνάρτηση αξιολόγησης.

Οι μηχανές σκακιού πλέον αποδίδουν σε υπεράνθρωπο επίπεδο, δηλαδή έχουν ξεπεράσει κάθε άνθρωπο όσον αφορά τις δυνατότητες στο παιχνίδι αυτό. Διαφορετικές μηχανές σκακιού που νίκησαν πρωταθλητές:

- Υπερυπολογιστής (1997): Deep Blue vs. Garry Kasparov (3.5–2.5)
- Προσωπικός υπολογιστής (2006): Deep Fritz vs. Vladimir Kramnik (4–2)
- Κινητό τηλέφωνο (2009): Το Pocket Fritz 4 νίκησε το τουρνουά Copa Mercosur Grandmaster, σημειώνοντας 9 νίκες και μία ισοπαλία (τελικό σκορ 9.5 στα 10).
- Υπολογιστής νικάει άνθρωπο με υπολογιστή (2017): Η μηχανή σκακιού Zor κέρδισε το τουρνουά freestyle Ultimate Challenge (το freestyle ή Advanced Chess είναι μια παραλλαγή, στην οποία επιτρέπονται επίσης ομάδες συμβούλων πχ παίκτες σε συνεργασία με υπολογιστές). Ο καλύτερος υπολογιστής σε συνεργασία με ανθρώπινη βοήθεια ήρθε στην 3η θέση.

2. BACKGROUND

2.1 Οι κανόνες του παιχνιδιού

Το σκάκι είναι ένα παινίδι τέλειας πληροφορίας μεταξύ δύο παικτών. Παίζεται σε ταμπλό 8x8 με τις στήλες του να ονομάζονται files και τις γραμμές να ονομάζονται ranks. Οι γραμμές είναι αριθμημένες από το 1 έως το 8 και οι στήλες είναι αριθμημένες με τους λατινικούς χαρακτήρες a-h. Χωρίζουν το ταμπλό σε 8x8 = 64 τετράγωνα των οποίων το χρώμα εναλάσσεται μεταξύ άσπρο και μαύρο σε κάθε γραμμή και στήλη. Σκοπός του παιχνιδιού είναι να απειλήσεις τον βασιλιά του αντιπάλου με τέτοιον τρόπο ώστε να μην μπορεί να ξεφύγει.

Οι παίκτες έχουν πιόνια που ξεχωρίζουν από το χρώμα τους, άσπρο και μαύρο. Τα άσπρα πιόνια τοποθετούνται στις γραμμές 1 και 2 και τα μαύρα στις γραμμές 7 και 8 κατά την εκκίνηση του παιχνιδιού. Υπάρχουν 6 διαφορετικά είδη από πιόνια. Ο βασιλιάς, η βασίλισσα, ο πύργος, ο αξιωματικός, το άλογο και ο στρατιώτης ή πιόνι (pawn). Στην εκκίνηση, στις γραμμές 1 και 8 αντίστοιχα τοποθετούνται με την εξής σειρά τα πιόνια: πύργος, άλογο, αξιωματικός, βασίλισσα, βασιλιάς, αξιωματικός, άλογο, πύργος με σειρά από τα αριστερά προς τα δεξιά. Στις γραμμές 2 και 7 τοποθετούνται 8 στρατιώτες για τον κάθε παίκτη αντίστοιχα. Την πρώτη κίνηση την έχει ο παίκτης με τα λευκά.

Στο σκάκι, υπάρχουν 6 διαφορετικά είδη από πιόνια. Το κάθε ένα έχει δικό του τρόπο κίνησης στο ταμπλό.

- **Ο στρατιώτης** κινείται προς τα εμπρός ένα τετράγωνο κάθε φορά. Υπάρχουν δύο εξαιρέσεις σε αυτόν τον κανόνα:
    1. Στην πρώτη κίνηση, μπορεί να προχωρήσει ένα ή δύο τετράγωνα.
    2. Όταν ένας στρατιώτης απειλεί ένα εχθρικό πίονι τότε μπορεί να κινηθεί ένα τετράγωνο διαγώνια μπροστά. Όταν φτάνει στην άλλη πλευρά του ταμπλό μπορεί να μετατραπεί σε οποιοδήποτε πίονι εκτός από τον βασιλία.
- **Το άλογο** είναι το μόνο πιόνι που μπορεί να περάσει πάνω από ένα άλλο πιόνι. Τα άλογα κινούνται τρία τετράγωνα τη φορά: δύο τετράγωνα προς τα εμπρός ή προς τα πίσω, στη συνέχεια ένα τετράγωνο προς τα αριστερά ή προς τα δεξιά ή δύο τετράγωνα προς το αριστερά ή δεξιά, μετά ένα τετράγωνο προς τα εμπρός ή προς τα πίσω. Η κίνηση μοιάζει με το γράμμα L. Το άλογο πάντα καταλήγει σε ένα τετράγωνο με διαφορετικό χρώμα από το οποίο ξεκίνησε.
- **Ο αξιωματικός** κινείται διαγώνια όσα ανοιχτά τετράγωνα επιθυμεί ο παίκτης. Ο αξιωματικός πρέπει να παραμείνει στο ίδιο χρώμα τετραγώνου με το οποίο ξεκίνησε το παιχνίδι.
- **Ο πύργος** κινείται σε ευθεία γραμμή, οριζόντια ή κάθετα όσα ανοιχτά τετράγωνα επιθυμεί ο παίκτης.
- **Η βασίλισσα** είναι το πιο ισχυρό από τα πιόνια του παιχνιδιού. Η Βασίλισσα κινείται προς οποιαδήποτε κατεύθυνση (οριζόντια, κάθετα ή διαγώνια) οποιονδήποτε αριθμό τετραγώνων επιθυμεί ο παίκτης.

- **Ο βασιλίας** είναι το πιο σημαντικό πιόνι στο ταμπλό, γιατί αν παγιδευτεί, ο παίκτης χάνει το παιχνίδι. Κινείται ένα τετράγωνο προς οποιαδήποτε κατεύθυνση, αρκεί να μην μπαίνει σε κατάσταση ρουά.

Πέρα από τις επιτρεπόμενες κινήσεις, υπάρχουν οι εξής κανόνες:

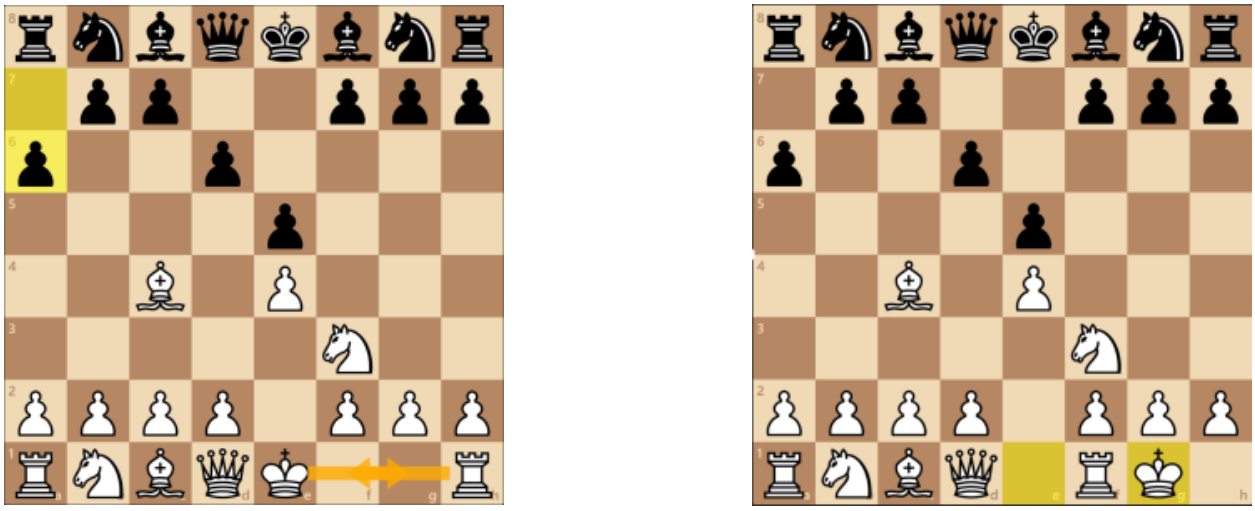
- **Αιχμαλωσία**: Όταν ένα από τα πιόνια ενός παίκτη κινείται και καταλήγει στη θέση που υπάρχει πιόνι ενός άλλου παίκτη, το αιχμλωτίζει και το αφαιρεί από το ταμπλό.
- **Ρουά**: Ένας παίκτης είναι σε ρουά (ή check), αν ένα από τα πιόνια του αντιπάλου είναι σε θέση να αιχμαλωτίσει τον βασιλιά του στην επόμενη κίνηση. Σε αυτήν την περίπτωση υπάρχουν μόνο 3 επιλογές για τον παίκτη που βρίσκεται σε ρουά:
    1. Να μετακινήσει τον βασιλιά του ώστε να μην απειλείται πια
    2. Να μετακινήσει κάποιο άλλο πιόνι ανάμεσα στο βασιλιά και το αντίπαλο πιόνι για να εμποδίσει την απειλή
    3. Να αιχμαλωτίσει το αντίπαλο πιόνι
- **Ρουά Ματ**: Όταν καμμία από τις 3 παραπάνω επιλογές δεν είναι εφικτή, ο παίκτης βρίσκεται σε Ρουά Ματ και χάνει το παιχνίδι.
- **Αιχμαλωσία "en passant"**: Όταν ένα πιόνι κινείται δύο θέσεις και βρεθεί δίπλα από ένα αντίπαλο πιόνι, ο αντίπαλος μπορεί να αιχμαλωτίσει το πιόνι αυτό "en passant" (γαλλικά για: Κατα τη διέλευση). Στην επόμενη κίνηση και μόνο τότε, το αντίπαλο πιόνι μπορεί να μετακινηθεί διαγώνια, προς το τετράγωνο πίσω από το πιόνι που μετακινήθηκε δύο τετράγωνα και να το αιχμαλώτίσει.

Βήμα 1 **Βήμα 2** **Βήμα 3**

Σχήμα 2.1: En Passant

- **Ροκέ**: Το ροκέ (ή castling) είναι μια ειδική κίνηση που μπορεί να πραγματοποιήσει ο βασιλιάς με οποιονδήποτε από τους 2 πύργους. Είναι η μόνη κίνηση στην οποία κινούνται 2 πιόνια ταυτόχρονα. Για να πραγματοποιηθεί το ροκέ ο πύργος κινείται δίπλα στον βασιλιά και έπειτα, ο βασιλιάς περνάει από πάνω και καταλήγει δίπλα στον πύργο από την άλλη πλευρά του. Για να γίνει αυτή η κίνηση πρέπει:

  1. Να είναι η πρώτη κίνηση και του βασιλιά και του πύργου
  2. Να μην υπάρχουν πιόνια μεταξύ τους
  3. Να μην απειλείται ο βασιλιάς πριν ή μετά το ροκέ
  4. Να μην απειλείται ο βασιλιάς σε κανένα από τα ενδιάμεσα τετράγωνα από τα οποία περνάει κατά το ροκέ



Βήμα 1 **Βήμα 2**

Σχήμα 2.2: Castling

- **Νίκη**: Η νίκη στο σκάκι πραγματοποιείται με ρουά ματ στον αντίπαλο βασιλιά, ή αν ο αντίπαλος παίκτης παραιτηθεί. Πολλά παιχνίδια καταλήγουν σε ισοπαλία. Ισοπαλία υπάρχει όταν:

  – Κανένας παίκτης δεν μπορεί να φέρει τον αντίπαλο σε θέση ρουά ματ
  – Οποιαδήποτε κίνηση οδηγεί σε ρουά
  – Γίνει τριπλή επανάληψη μίας ή περισσότερων κινήσεων χωρίς να αλλάξει κάτι στο ταμπλό
  – Γίνουν 50 κινήσεις χωρίς κάποια κίνηση στρατιώτη ή αιχμαλωσία και από τους 2 παίκτες.

2.2 Νευρωνικά Δίκτυα

Τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks ή ANNs) [31] είναι συστήματα υπολογισμού εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα, τα οποία αποτελούν τον εγκέφαλο ζώων. Ένα ANN ανάγεται σε ένα σύνολο από συνδεδεμένες μονάδες ή κόμβους που ονομάζονται νευρώνες. Κάθε σύνδεση μεταφέρει ένα σήμα (σε μορφή αριθμητικής τιμής) από έναν νευρώνα στον άλλον. Ο νευρώνας που δέχεται το σήμα, το επεξεργάζεται

και στη συνέχεια το προωθεί στους επόμενους. Η επεξεργασία έχει τη μορφή μιας μη γραμμικής συνάρτησης, η οποία ονομάζεται συνάρτηση ενεργοποίησης (activation function). Η συνάρτηση ενεργοποίησης δέχεται το άθροισμα των σημάτων από τις συνδέσεις του νευρώνα και προωθεί το αποτέλεσμα που παράγει. Οι συνδέσεις αυτές ονομάζονται ακμές (edges), όπως οι συνδέσεις γράφων. Οι νευρώνες και οι ακμές συνήθως έχουν βάρη, τα οποία μεταβάλλονται κατά την εκμάθηση. Οι νευρώνες ταξινομούνται σε επίπεδα ή στρώματα (layers). Ο αριθμός επιπέδων ενος δικτύου και το πλήθος νευρώνων σε κάθε επίπεδο καθορίζουν την πολυπλοκότητα του δικτύου.

Η απλούστερη μορφή τεχνητού νευρωνικού δικτύου είναι ένα δίκτυο με μια σειρά από δεδομένα εισόδου και έναν μοναδικό νευρώνα. Το δίκτυο αυτό ονομάζεται Perceptron (Frank Rosenblatt, 1957) [39] και αποκαλείται και Threshold Logic Unit (TLU).

Υπάρχουν πολλά διαφορετικά είδη Νευρωνικών Δικτύων. Τα πιο σημαντικά είναι τα εξής:

1. Δίκτυα Εμπρόσθιας Τροφοδοσίας
2. Συνελικτικά Δίκτυα
3. Επαναλαμβανόμενα δίκτυα

2.2.1 Δίκτυα Εμπρόσθιας Τροφοδοσίας

Η πιο απλή και συνηθισμένη μορφή δικτύων είναι τα δίκτυα εμπρόσθιας τροφοδοσίας (Feed Forward Neural Networks). Αποτελούνται από πολλά perceptrons στοιχισμένα σε επίπεδα. Γι αυτό αποκαλούνται και πολυεπίπεδα perceptrons [10]. Σε ένα τέτοιο δίκτυο, τα δεδομένα ακολουθούν μια συγκεκριμένη ροή και οι συνδέσεις μεταξύ των νευρώνων έχουν τη μορφή άκυκλου, κατευθυνόμενου γραφήματος. Κάθε κόμβος λαμβάνει σήματα από το 'από πάνω' και προωθεί το σήμα στο 'από κάτω' επίπεδο, αφού το επεξεργαστεί.

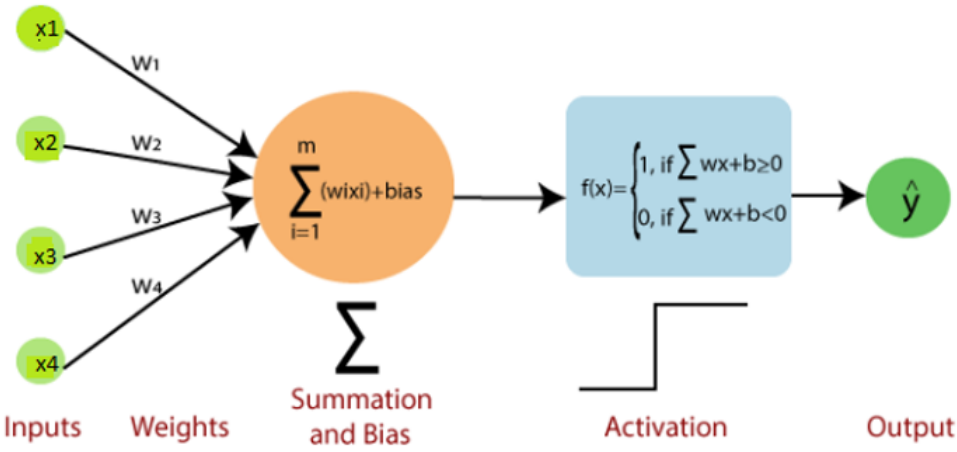
Τα δίκτυα εμπρόσθιας τροφοδοσίας χωρίζονται σε επίπεδα που μπορούν να κατηγοριοποιηθούν σε 3 υποκατηγορίες. Το πρώτο επίπεδο ονομάζεται επίπεδο εισόδου και από εκεί διέρχονται τα δεδομένα στην αρχική μορφή τους. Το τελευταίο επίπεδο ονομάζεται επίπεδο εξόδου και παράγει το αποτέλεσμα των υπολογισμών του δικτύου, όπως μια πρόβλεψη, κατηγοριοποίηση κ.α. Όλα τα ενδιάμεσα επίπεδα ανάγονται στο κρυφό (hidden) επίπεδο. Αν τα επίπεδα αυτά είναι πάνω από ένα, το δίκτυο ονομάζεται βαθύ νευρωνικό δίκτυο (deep neural network) [30].

Όταν όλοι οι νευρώνες ενός επιπέδου του δικτύου συνδέονται με κάθε νευρώνα του προηγούμενου επιπέδου, το επίπεδο αυτό ονομάζεται πλήρως συνδεδεμένο επίπεδο ή πυκνό επίπεδο (dense layer).

2.2.2 Συνελικτικά Δίκτυα

Τα Συνελικτικά Δίκτυα (Convolutional Neural Networks, CNNs) προέκυψαν από μελέτες στον οπτικό φλοιό του εγκεφάλου και ξεκίνησαν να χρησιμοποιούνται τη δεκαετία του 80 κυρίως στην αναγνώριση εικόνας (Fukushima, 1982) [9]. Τα δίκτυα αυτά βασίζονται σε ένα διαφορετικό είδος επιπέδων, τα συνελικτικά επίπεδα.

Οι νευρώνες ενός συνελικτικού επιπέδου δε συνδέονται με κάθε τιμή του προηγούμενου, αλλά μόνο με συγκεκριμένα τμήματα. Αυτά τα τμήματα ονομάζονται δεκτικά πεδία



Σχήμα 2.3: Απλό perceptron με συνάρτηση ενεργοποίησης την step function

(receptive fields) των νευρώνων. Χάρη σε αυτά, το δίκτυο έχει τη δυνατότητα να επικεντρώνεται σε χαμηλού επιπέδου χαρακτηριστικά στα πρώτα στρώματα και να τα συνδυάζει σε χαρακτηριστικά μεγαλύτερου επιπέδου στα επόμενα.

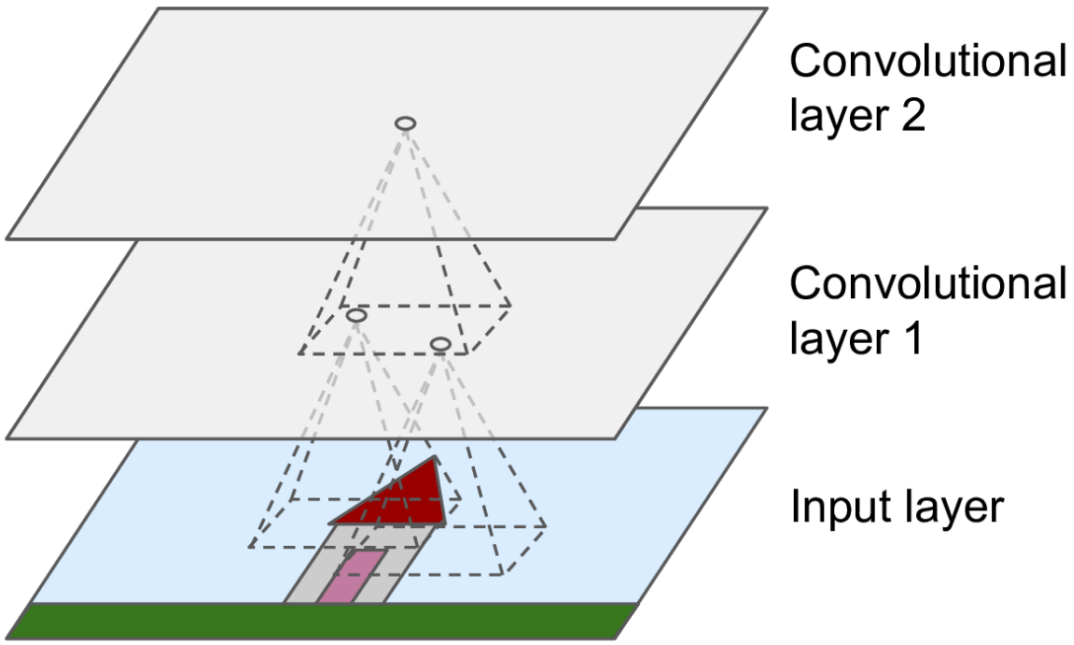
Τα βάρη σε ένα συνελικτικό δίκτυο αναπαρίστανται ως μικροί, πολυδιάστατοι πίνακες μεγέθους ίσου με το δεκτικό πεδίο τους, που ονομάζονται φίλτρα (filters). Οι διαστάσεις των φίλτρων εξαρτώνται από τις διαστάσεις των δεδομένων που επεξεργάζονται τα δίκτυα. Για παράδειγμα, ένα συνελικτικό δίκτυο για την αναγνώριση εικόνων θα έχει δισδιάστατα φίλτρα, ενώ ένα που σκοπός του είναι η τρισδιάστατη απεικόνιση θα έχει φίλτρα με τρεις διαστάσεις.

Από τις σημαντικότερες εφαρμογές των συνελικτικών δικτύων είναι η κατηγοριοποίηση εικόνων. Ένα δισδιάστατο συνελικτικό δίκτυο έχει τη δυνατότητα να αντιλαμβάνεται τις διαφορές μεταξύ εικόνων, λαμβάνοντας σαν είσοδο το περιεχόμενο των pixel της εικόνας. Η δυνατότητα αυτή μιμείται την ανθρώπινη όραση (στις 2 διαστάσεις) και χρησιμοποιείται σε πληθώρα εφαρμογών. Μία από αυτές είναι η απόδοση σε παιχνίδια που παίζονται πάνω σε ταμπλό. Αρκετά συχνά είναι δυνατόν να διαχειριστεί κανείς το ταμπλό ενός παιχνιδιού σαν μια εικόνα, κωδικοποιώντας κάθε θέση του ώστε να αναπαριστά την κατάσταση του παιχνιδιού στη θέση αυτή. Για παράδειγμα, κάθε τετράγωνο στο σκάκι μπορεί να κωδικοποιηθεί ώστε να δείχνει το είδος και χρώμα του πιονιού που βρίσκεται εκεί. Έτσι, τα συνελικτικά δίκτυα είναι κατάλληλα για την δημιουργία μοντέλου για παιχνίδια σαν και αυτό.

2.2.3 Επαναλαμβανόμενα δίκτυα

Ένα άλλο είδος δικτύου, το οποίο χρησιμοποιείται για την επίλυση πιο πολύπλοκων προβλημάτων, είναι το επαναλαμβανόμενο δίκτυο (recurrent network) [19]. Σε ένα τέτοιο δίκτυο, οι έξοδοι ανακατευθύνονται στις εισόδους, δημιουργώντας ένα δυναμικό σύστημα το οποίο μπορεί να φτάσει σε μια κατάσταση ηρεμίας ή να παρουσιάσει αναταράξεις.

Η μεγαλύτερη διαφορά του από τα προηγούμενα είδη δικτύων είναι ότι δεν πρόκειται για γραμμική ροή δεδομένων. Η είσοδος χωρίζεται σε ξεχωριστά τμήματα εισόδου και κάθε στρώμα του δικτύου δέχεται ένα από αυτά τα τμήματα, μαζί με την έξοδό όλων των προηγούμενων στρωμάτων. Η προσέγγιση αυτή έχει οδηγήσει στην επίλυση ακόμα πιο πολύπλοκων προβλημάτων σε διάφορους τομείς, με σημαντικότερη βελτίωση απόδοσης αυτήν στον τομέα της επεξεργασίας φυσικής γλώσσας.



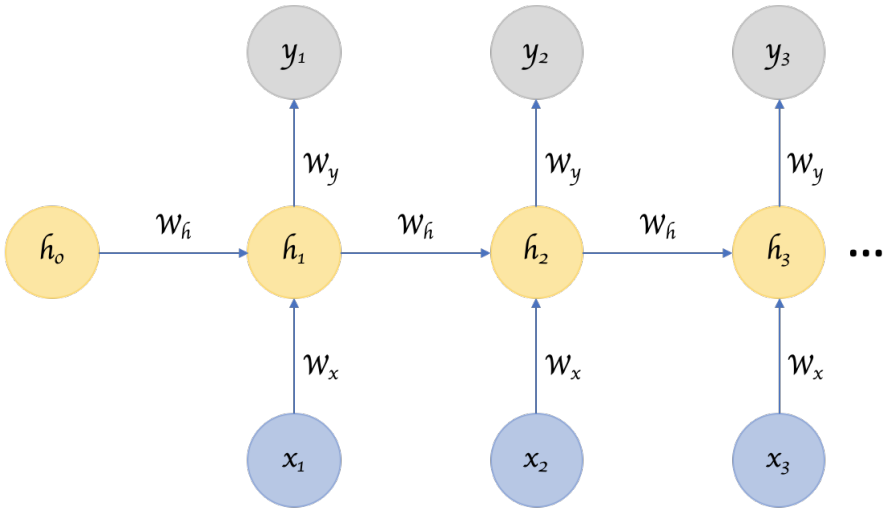
Σχήμα 2.4: Οπτικοποίηση συνελικτικού δικτύου

2.2.4 Οπισθοδιάδοση

Η εκπαίδευση ενός τεχνητού νευρωνικού δικτύου βασίζεται στη μέθοδο της οπισθοδιάδοσης (Backpropagation) [12]. Τα βάρη αρχικοποιούνται με κάποιες αυθαίρετες τιμές. Στη συνέχεια, το δίκτυο τροφοδοτείται με στοιχεία εισόδου και παράγει κάποια έξοδο βασισμένη σε αυτά τα βάρη. Η έξοδος αυτή αξιολογείται με κάποιον τρόπο, συνήθως με σύγκριση με την επιθυμητή έξοδο του δικτύου όταν δέχεται τη συγκεκριμένη είσοδο. Μετριέται η απόσταση της επιθυμητής εξόδου με την πραγματική και παράγεται μια τιμή σφάλματος. Έπειτα, το σφάλμα αυτό διαδίδεται από το στρώμα εξόδου προς τα πίσω, στα προηγούμενα επίπεδα, αλλάζοντας τα βάρη σύμφωνα με κάποια μέθοδο ώστε να μειωθεί το σφάλμα στις επόμενες προσπάθειες.

Η διαδικασία αυτή πραγματοποιείται για όλο το σύνολο δεδομένων εκπαίδευσης. Αυτό ονομάζεται μία εποχή (epoch). Η διαδικασία εκπαιδευσης έχει διάρκεια από μία έως και τεράστιο αριθμό εποχών, ανάλογα με την πολυπλοκότητα του προβλήματος.

Η πολιτική που ακολουθείται για την μεταβολή τιμών των βαρών του δικτύου κατά την οπισθοδιάδοση διαφέρει από δίκτυο σε δίκτυο. Στη συγκεκριμένη εργασία χρησιμοποιείται η μέθοδος Gradient Descent [5]. Πιο συγκεκριμένα, μια παραλλαγή της που επιφέρει γρηγορότερη σύγκλιση, η Stohastic Gradient Descent [6].



Σχήμα 2.5: Επαναλαμβανόμενο δίκτυο

2.2.5 Εκμάθηση Χρονικής Διαφοράς

Η Εκμάθηση Χρονικής Διαφοράς (Temporal Difference Learning) [38] βασίζεται στη διαφορά μεταξύ χρονικά διαδοχικών προβλέψεων. Με άλλα λόγια, ο σκοπός της εκπαίδευσης είναι να μεγιστοποιήσει την ομοιότητα μεταξύ της τρέχουσας πρόβλεψης στην τωρινή είσοδο με την επόμενη πρόβλεψη στο επόμενο χρονικό βήμα. Ίσως η πιο διαδεδομένη μέθοδος εκμάθησης χρονικής διαφοράς είναι η TD(lambda):

Η TD(lambda), όπως εφαρμόζεται σε ένα απλό multi-layered perceptron, ανανεώνει τα βάρη του δικτύου σε κάθε χρονικό βήμα ως εξής:

$$w_{t+1} - w_t = a(Y_{t+1} - Y_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w Y_k$$

Όπου a είναι ο βαθμός μάθησης (Learning rate), μια μικρή, σταθερή τιμή που αποφασίζει κατά πόσο επηρεάζει η διαφορά των αποτελεσμάτων την αλλαγή στα βάρη. W είναι το διάνυσμα με τα βάρη που παραμετροποιούν το δίκτυο τη χρονική στιγμή t και $t+1$. $\nabla_w Y_k$ είναι η παράγωγος της εξόδου του δικτύου ως προς τα βάρη. Η παραπάνω συνάρτηση προκύπτει από εκπαίδευση με μόνο ένα αντικείμενο εισόδου. Σε περίπτωση πολλαπλών αντικειμένων ανά εποχή, η συνολική αλλαγή των w δίνεται από το άθροισμα των αλλαγών από κάθε αντικείμενο ξεχωριστά.

Η ποσότητα λ είναι μια ευρετική παράμετρος που ελέγχει το χρονικό ποσοστό κατά το οποίο το σφάλμα που καταμετρήθηκε σε μια χρονική στιγμή, διαδίδεται προς τα πίσω για να διορθώσει προηγούμενες εκτιμήσεις. Με λ κοντά στο 0, δεν υπάρχει σχεδόν καθόλου ανατροφοδότηση πέρα από την συγκεκριμένη χρονική στιγμή. Με λ κοντά στο 1, η κάθε πρόβλεψη ανατροφοδοτείται στα προηγούμενα χρονικά επίπεδα με σχεδόν μηδαμινή εξασθένηση, επηρεάζοντας όλες τις προηγούμενες προβλέψεις.

2.3 Μέθοδοι μάθησης

Όπως αναλύθηκε στην προηγούμενη ενότητα, τα νευρωνικά δίκτυα αποτελούνται από ένα σύνολο παραμέτρων, που ονομάζονται βάρη. Τα βάρη αυτά αρχικοποιούνται συνήθως με τυχαίες τιμες και σιγά σιγά μεταβάλλονται για να εκπληρώσουν το στόχο που θέτει ο προγραμματιστής. Η διαδικασία αυτή ονομάζεται εκπαίδευση ή εκμάθηση. Υπάρχουν 3 βασικές τεχνικές εκπαίδευσης ενός δικτύου με διαφορετικές δυνατότητες, κάθε μία από τις οποίες έχει διαφορετικές εφαρμογές.

2.3.1 Μάθηση χωρίς επίβλεψη

Η μη επιτηρούμενη μάθηση ή μάθηση χωρίς επίβλεψη [3] είναι μια τεχνική μηχανικής μάθησης στην οποία οι χρήστες δεν χρειάζεται να επιβλέπουν το μοντέλο. Αντ 'αυτού, επιτρέπει στο μοντέλο να δουλεύει μόνο του για να ανακαλύπτει μοτίβα και πληροφορίες που προηγουμένως δεν είχαν εντοπιστεί. Ασχολείται κυρίως με δεδομένα στα οποία δεν έχουν δοθεί τίτλοι κατηγοριών από τον άνθρωπο. Η μάθηση χωρίς επίβλεψη χρησιμοποιείται ώστε να βρει κάποιο μοτίβο μεταξύ των δεδομένων, που δεν είναι εύκολο να παρατηρηθέι και φυσικά στοχεύει στην κατηγοριοποίησή τους.

Μερικά παραδείγματα αλγορίθμων μάθησης χωρίς επίβλεψη είναι:

1. Hierarchical clustering [25]
2. K-means clustering [16]
3. Principal Component Analysis [1]
4. Singular Value Decomposition [24]
5. Independent Component Analysis [8]

2.3.2 Μάθηση με επίβλεψη

Στην Μάθηση με επίβλεψη (supervised learning) [11] τα δεδομένα εκπαίδευσης που δέχεται ο αλγόριθμος συμπεριλαμβάνουν τις επιθυμητές λύσεις, που ονομάζονται labels. Τα πιο συνηθισμένα παραδείγματα Supervised Learning είναι η κατηγοριοποίηση (classification) και η πρόβλεψη (regression). Ένα παράδειγμα κατηγοριοποίησης είναι το spam detector, που κατηγοριοποιεί e-mails, κατατάσσοντάς τα σε ενοχλητικά ή χρήσιμα. Ένα παράδειγμα regression είναι η πρόβλεψη τιμής ενός αυτοκινήτου. Κατά την εκπαίδευση σε αυτό το παράδειγμα, δίνονται στο μοντέλο πολλά αυτοκίνητα με τις πραγματικές τιμές τους και τις προβλέψεις των τιμών τους.

Μερικά παραδείγματα αλγορίθμων μάθησης με επίβλεψη είναι:

1. k-Nearest Neighbors [26]
2. Linear Regression [21]
3. Logistic Regression [15]
4. Support Vector Machines (SVMs) [40]

5. Decision Trees and Random Forests [2]
6. Neural networks [31]

2.3.3 Ενισχυτική Μάθηση

Η ενισχυτική μάθηση (Reinforcement Learning, RL) [36] εξετάζει τη συνεχή αλληλεπίδραση υπολογιστικών μονάδων (agents) με το περιβάλλον τους.

Η ιδέα της ενισχυτικής μάθησης είναι εμπνευσμένη απο την ψυχολογία της συμπεριφοράς (behavior psychology). Έχει ως αφετηρία εργασίες όπως του Pavlov [18] και στην διαπίστωση ότι η μάθηση ωθείται από τη διαφορά μεταξύ της εκτιμώμενης και της λαμβανόμενης ανταμοιβής (Rescorla-Wagner) [20]. Ο Sutton [37] περιγράφει την ιδέα οτι μια υπολογιστική μονάδα μπορεί να μάθει πώς να ενεργεί για την πραγματοποίηση κάποιου σκοπού απο την αλληλεπίδραση με το περιβάλλον της, όπως ακριβώς και μια βιολογική μοναδα.

Η θεμελίωση της ενισχυτικής μάθησης και οι επακόλουθες μεγάλες επιτυχίες της προκύπτουν και απο τους ισχυρούς δεσμούς της με τον βέλτιστο έλεγχο (Optimal control), την επιχειρησιακή έρευνα (Operations Research), τις προσεγγιστικές δομές (στοχαστική προσέγγιση, νευρωνικά δίκτυα), την εξομείωση και τη θεωρία παιγνίων.

Το σύστημα εκμάθησης, που ονομάζεται μονάδα (ή πράκτορας) μπορεί να παρατηρήσει το περιβάλλον, να επιλέξει ενέργειες και να επιβραβευθεί για τις επιτυχημένες. Αντίστοιχα με τις λανθασμένες ενέργειες δέχεται αρνητικές επιβραβεύσεις, δηλαδή ποινές. Πρέπει να μάθει από μόνο του ποια είναι η κατάλληλη στρατηγική, που ονομάζεται πολιτική (policy) για να πάρει τις περισσότερες επιβραβεύσεις σε βάθος χρόνου. Μια πολιτική καθορίζει ποιά ενέργεια θα ακολουθήσει ο πράκτορας όταν βρίσκεται σε κάποια κατάσταση. Πιο συγκεκριμένα:

- **Μονάδα:** Σε κάθε χρονική στιγμή t κάθε μονάδα k παρατηρεί την κατάσταση του περιβάλλοντος x_t με μια μέτρηση y_t^k συνοδευόμενη από κάποιον θόρυβο. Με βάση την εικόνα που διαμορφώνει για τη κατάσταση του περιβάλλοντος, τη στιγμή t (αξιοποιώντας μετρήσεις και ανταμοιβές), η μονάδα k επιλέγει ενέργεια a_t^k και την εκτελεί.
- **Περιβάλλον:** Ως συνέπεια των ενεργειών των μονάδων, το περιβάλλον αλλάζει και μεταβαίνει στοχαστικά σε νέα κατάσταση. Ταυτόχρονα παράγει νέα μέτρηση y_{t+1}^k και ανταμοιβή r_t^k για κάθε μοναδα k. Ο κύκλος αυτός επαναλαμβάνεται συνεχώς επιτρέποντας στις μονάδες να μαθαίνουν πως να ενεργούν ώστε να βελτιώνουν το συνολικό μακροπρόθεσμο όφελος.

2.3.4 Αξιοσημείωτα κατορθώματα της ενισχυτικής μάθησης

Η ενισχυτική μάθηση διαφέρει από τις κλασσικές μεθόδους στο ότι δε βασίζεται σε προϋπάρχουσα γνώση και αντ' αυτού αφήνει την υπολογιστική μονάδα να σχεδιάσει και να διδάξει στον εαυτό της τη λύση στο πρόβλημα που της τέθηκε, λαμβάνοντας μόνο επιβραβεύσεις ή ποινές σε κάθε βήμα της εκμάθησης, ανάλογα με τις σωστές ή λάθος επιλογές της. Η στρατηγική είναι λιγότερο διαισθητική από άλλες μεθόδους όπως αυτές της μάθησης με επίβλεψη. Όμως η ενισχυτική μάθηση έχει επιτύχει σημαντικά κατορθώματα και σε πολλές περιπτώσεις, καλύτερες επιδόσεις από κάθε άλλη μέθοδο. Μερικά από τα επιτεύγματα της ενισχυτικής μάθησης είναι τα εξής:

1. Tesauro 1995. Το TD-Gammon είναι εν απρόγραμμα που παίζει τάβλι και δημιουργήθηκε το 1992 από τον Gerald Tesauro στο IBM's Thomas J. Watson Research Center [38]. Το όνομά του προέρχεται από την μέθοδο εκμάθησης χρονικής διαφοράς (συγκεκριμένα TD-lambda) που είναι και η βασική μέθοδος για την εκπαίδευση του προγράμματος.Το πρόγραμμα κατάφερε να νικήσει όλες τις προηγούμενες εκδόσεις καθώς και τους πρωταθλήτες με βασική καινοτομία ότι ανέπτυξε στρατηγηκές από μόνο του αρκετά μακριά από τις ανθρώπινες καθώς και αναβάθμισε το επίπεδο των ανθρωπίνων γνώσεων ως προς το παιχνίδι.

2. Mnih et al., 2015 Επιτυγχάνουν επιδόσεις ανώτερες του ανθρώπου με τη χρήση Reinforcement Learning και νευρωνικών δικτύων (Deep RL) στα παιχνίδια του Atari. Με την αξιοσημείωτη επίδοση ότι σε σύγκριση 49 παιχνιδίων μεταξύ ανθρώπων και προγράμματος, το πρόγραμμα πέτυχε σε πάνω από τα μισά παιχνίδια βαθμολογία μεγαλύτερη από εκείνη των ανθρώπων κατά 75%.

3. Silver et al., 2016 προγραμματα που νικουν πρωταθλητές στο παιγνίδι Go. Αυτό επιτέυχθηκε με συνδυασμό ενισχυτικής και επιβλεπόμενης μάθησης.

4. Moravski et al., 2017 προγραμματα νικούν πρωταθλητές στο πόκερ.

5. Silver et al., 2017 πρόγραμμα που ξεπερνά κάθε προηγούμενο μοντέλο για το παιχνίδι Go, με χρήση μόνο ενισχυτικής μάθησης.

2.4 Θεωρία παιγνίων

Η θεωρία παιγνίων είναι ένας επιστημονικός κλάδος που μελετά στρατηγικές για την επιλογή της καλύτερης δυνατής επιλογής σε παιχνίδια. Έχει τεράστιο θεωρητικό υπόβαθρο και συμβάλλει σημαντικά στη μεθοδολογία όλων των αλγορίθμων με εφαρμογές σε παιχνίδια.

2.4.1 Παιχνίδια μηδενικού αθροίσματος τέλειας πληροφορίας

Στη θεωρία παιγνίων, παιχνίδι μηδενικού αθροίσματος (zero-sum game) [22] ονομάζεται η μαθηματική αναπαράσταση μιας κατάστασης στην οποία κάθε επιβράβευση ή ποινή κάποιου συμμετέχοντος ισσοροπείται από τις ποινές και επιβραβεύσεις των υπόλοιπων συμμετεχόντων. Αυτό σημαίνει ότι αθροίζοντας τις επιβραβεύσεις και ποινές όλων των συμμετεχόντων, το αποτέλεσμα που προκύπτει θα είναι πάντα μηδενικό.

Επίσης, ένα παιχνίδι λέμε ότι έχει τέλεια πληροφορία (perfect information) [28] όταν κάθε παίκτης σε κάθε μια από τις κινήσεις του, έχει πλήρη γνώση των γεγονότων που έχουν ήδη λάβει μέρος, ακόμα και της αρχικής κατάστασης του παιχνιδιού. Δεν υπάρχει δηλαδή καμία πληροφορία την οποία εκείνος δεν γνωρίζει όταν αποφασίζει την επόμενη κίνησή του.

2.4.2 Markov Games

Τα παιχνίδια Markov [17] είναι μια ιδιαίτερη υποκατηγορία των παιχνιδιών τέλειας πληροφορίας. Στα παιχνίδια αυτά, υπάρχει ένας χώρος καταστάσεων S (οι καταστάσεις συμπεριλαμβάνουν ποιος παίκτης παίζει), ένας χώρος ενεργειών/κινήσεων A, που ορίζει

τις νόμιμες κινήσεις σε κάθε κατάσταση s του S, μια συνάρτηση διαδοχής $f(s, a, r)$, που καθορίζει την κατάσταση που προκύπτει αν επιλεγεί η ενέργεια a στην κατάσταση s με τυχαιότητα r (πχ ζάρια), και μια συνάρτηση επιβράβευσης $r_i(s)$ που ορίζει την επιβράβευση του παίκτη i στην κατάσταση s.

Στα παιχνίδια μηδενικού αθροίσματος, έχουμε:

$$r_1(s) = -r_2(s) = r(s)$$

με ντετερμινιστικές διαδοχές:

$$f(s, a, r) = f(s, a)$$

και μηδενικές επιβραβεύσεις πέρα από το τερματικό χρονικό βήμα.

2.4.3 Παράγοντας Διακλάδωσης

Στη θεωρία παιγνίων, μια σημαντική τιμή είναι αυτή του παράγοντα διακλάδωσης (branching factor). Η τιμή αυτή δηλώνει τον αριθμό παιδιών που προέρχονται από έναν κόμβο σε κάποιο δέντρο παιχνιδιού. Σε περιπτώσεις που ο αριθμός παιδιών διαφέρει, χρησιμοποιείται ο μέσος παράγοντας διακλάδωσης.

Η αύξηση της τιμής του συνεπάγεται αύξηση της πολυπλοκότητας του παιχνιδιού, καθώς παράγονται περισσότερα υποδέντρα σε κάθε επίπεδο. Το Go έχει σημαντικά μεγαλύτερο παράγοντα διακλάδωσης από παιχνίδια, όπως το σκάκι. Γι αυτό, η απόδοση του AlphaGo Zero θεωρείται τόσο μεγάλη επιτυχία της ενισχυτικής μάθησης.

2.5 Αλγόριθμοι αναζήτησης σε δέντρα

Οι αλγόριθμοι αναζήτησης σε δέντρα χρησιμοποιούν κατά κύριο λόγο δενδρικές δομές είτε δυαδικές είτε με περισσότερες διακλαδώσεις ανάλογα με την φύση και τις ανάγκες του προβλήματος. Τέτοια δέντρα ονομάζονται δενδρα παιχνιδιού. Σε ένα δέντρο παιχνιδιού οι κόμβοι είναι οι καταστάσεις και οι ακμές είναι οι κινήσεις των παικτών. Κάθε κόμβος έχει επάνω του μία ή περισσότερες τιμές σύμφωνα με την οποία κρίνεται η σημασία του και βάσει αυτής αξιολογείται η αξία της συγκεκριμένης κατάστασης (κόμβου).

2.5.1 Minimax

Ο Minimax [35] είναι ένα είδος αλγορίθμου backtracking (οπισθοδρόμησης) που χρησιμοποιείται στη λήψη αποφάσεων και στη θεωρία παιγνίων για να βρει τη βέλτιστη κίνηση για έναν παίκτη, υποθέτοντας ότι ο αντίπαλος παίζει επίσης με τον καλύτερο τρόπο. Χρησιμοποιείται ευρέως σε παιχνίδια δύο παικτών εναλασσόμενης σειράς, όπως τρίλιζα, τάβλι, Mancala, σκάκι κ.α.

Στο Minimax οι δύο παίκτες ονομάζονται μεγιστοποιητής και ελαχιστοποιητής. Ο μεγιστοποιητής προσπαθεί να πάρει την υψηλότερη δυνατή βαθμολογία, ενώ ο ελαχιστοποιητής προσπαθεί να κάνει το αντίθετο και να πάρει τη χαμηλότερη δυνατή βαθμολογία.

Κάθε κατάσταση του δέντρου έχει μια τιμή. Κάθε επίθεδο του δέντρου θεωρείται εναλλάξ μεγιστοποιητής ή ελαχιστοποιητής, ξεκινώντας συνήθως από τον μεγιστοποιητή (ρίζα δέντρου). Το δέντρο διατρέχεται προς τα κάτω και το κάθε επίπεδο επιλέγει την καλύτερη τιμή με βάση την λειτουργία του. Στην συνέχεια διατρέχονται πίσω οι κόμβοι με οπισθοδρόμηση και κάθε κόμβος γνωρίζει την καλύτερη επιλογή για εκείνον. Σε μια δεδομένη κατάσταση εάν ο μεγιστοποιητής έχει το πάνω χέρι τότε, το σκορ του πίνακα θα έχει την τάση να είναι κάποια θετική τιμή. Εάν ο ελαχιστοποιητής έχει το πάνω χέρι σε αυτήν την κατάσταση του πίνακα τότε θα έχει την τάση να έχει κάποια αρνητική τιμή. Οι τιμές του πίνακα υπολογίζονται από ορισμένες ευρετικές μεθόδους που είναι μοναδικές για κάθε τύπο παιχνιδιού. Δεδομένου ότι αυτός είναι ένας αλγόριθμος βασισμένος στο backtracking, δοκιμάζει όλες τις πιθανές κινήσεις, στη συνέχεια κάνει backtrack και παίρνει μια απόφαση.

```
function minimax(node,depth,maximizingPlayer) {

    if depth == 0 or node is a terminal node then
        return static evaluation of node


    if maximizingPlayer {    // for maximizing Player
        maxEva = -infinity
        for each child of node {
            eva = minimax(child,depth-1,false)
            maxEva = max(maxEva, eva)
        }
        return maxEva
    }
    else{                    // for minimizer player
        minEva = +infinity
        for each child of node {
            eva = minimax(child,depth-1,true)
            minEva = min(minEva, eva)
        }
        return minEva
    }
}
```

Σχήμα 2.6: Ψευδοκώδικας για τον αλγόριθμο Minimax

2.5.2 Alpha Beta Pruning

Το πρόβλημα με τον αλγόριθμο minimax είναι ότι ο αριθμός των καταστάσεων που πρέπει να εξετάσουμε είναι εκθετικός ως προς το βάθος του δέντρου παιχνιδιού. Δεν μπορούμε να αποφύγουμε τον εκθέτη αυτόν, αλλά μπορούμε να τον μειώσουμε περίπου στο μισό με το να μην εξετάζουμε όλους τους κόμβους του δέντρου. Ο αλγόριθμος που το επιτυγχάνει αυτό λέγεται κλάδεμα άλφα-βήτα (alpha-beta pruning) [29].

Σε ένα δέντρο minimax τα επίπεδα εναλλάσσονται ανάμεσα σε ελαχιστοποιητές και μεγιστοποιητές. Η ιδέα του alpha-beta pruning βασίζεται στην εξής περίπτωση: αν ένας μεγιστοποιητής επιλέξει μια τιμή συνεχίζει να διατρέχει το δέντρο με την ελπίδα να βρει παρακάτω μια μεγαλύτερη τιμή απο αυτήν που επέλεξε. Όμως στην συνέχεια ακολουθεί η σειρά ενός ελαχιστοποιητή, ο οποίος θα επιλέξει την ελάχιστη τιμή για αυτόν. Συνεπώς, αν ο ελαχιστοποιητής βρει μία τιμή μικρότερη από αυτήν που έχει επιλέξει ο παραπάνω μεγιστοποιητής τότε όλο το από κάτω υποδέντρο θα παρακαμφθεί. Αυτό συμβαίνει με την απλή λογική ότι αφου ο ελαχιστοπιητής βρήκε μια μικρότερη τιμή από αυτή του μεγιστοποιητή, δεν έχει νόημα η συνέχεια καθώς ο ελαχιστοποιητής είτε θα επιλέξει στην συνέχεια κάτι ακόμα μικρότερο είτε αυτό που επέλεξε στην αρχή. Και στις δύο όμως περιπτώσεις

```
function minimax(node,depth,maximizingPlayer) {

    if depth == 0 or node is a terminal node then
        return static evaluation of node


    if maximizingPlayer {    // for maximizing Player
        maxEva = -infinity
        for each child of node {
            eva = minimax(child,depth-1,alpha,beta,false)
            maxEva = max(maxEva, eva)
            alpha = max(alpha,maxEva)
            if beta <= alpha
                break
        }
        return maxEva
    }
    else{                    // for minimizer player
        minEva = +infinity
        for each child of node {
            eva = minimax(child,depth-1,alpha,beta,true)
            minEva = min(minEva, eva)
            beta = min(beta, eva)
            if beta <= alpha
                break
        }
        return minEva
    }
}
```

Σχήμα 2.7: Ψευδοκώδικας για τον αλγόριθμο Alpha Beta pruning

ο από πανω μεγιστοποιητής θα απορρίψει και τις δύο τιμές καθώς έχει επιλέξει ήδη την μέγιστη.

2.5.3 Monte Carlo

Η αναζήτηση Monte Carlo σε δέντρο (Monte Carlo Tree Search, MCTS) [7] είναι μια μέθοδος επιλογής του καλύτερου κάθε φορά αποτελέσματος (best-first search), καθοδηγούμενη από προσομοιώσεις Monte-Carlo. Μια προσομοίωση Monte-Carlo χρησιμοποιείται για να μοντελοποιήσουμε την πιθανότητα από διαφορετικές εκδοχές κάποιας διαδικασίας στην οποία υπάρχει κάποιας μορφής τυχαιότητα. Βασίζεται στην τυχαία δειγματοληψία για να αποκτήσει αριθμητικές τιμές για την κάθε εκδοχή και η βασική του ιδέα είναι να χρησιμοποιηθεί η τυχαιότητα για να επιλυθούν ντετερμινιστικά προβλήματα.

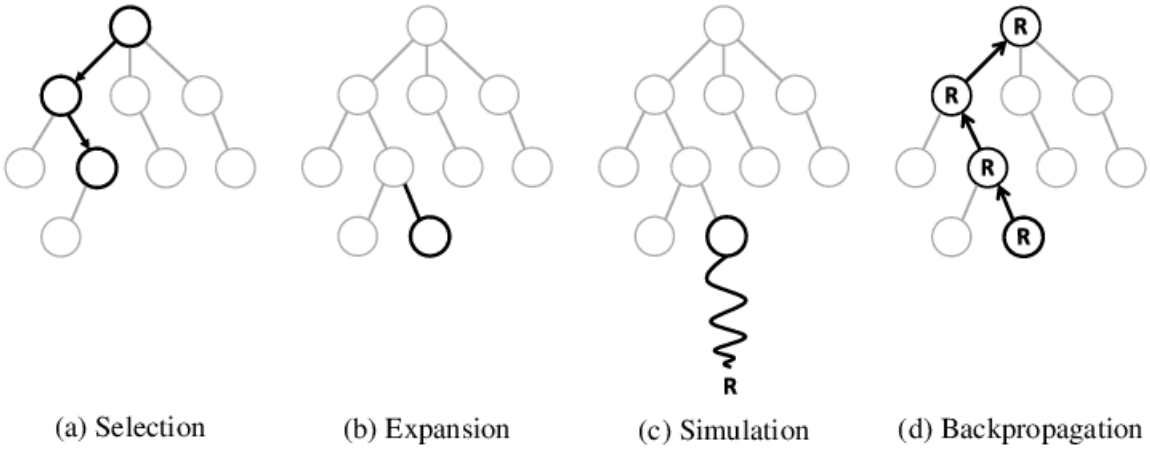
Συγκεκριμένα, οι Monte-Carlo προσομοιώσεις πραγματοποιούν συνεχόμενη δειγματοληψία για να προσεγγίσουν τις στατιστικές ιδιότητες κάποιου φαινομένου. Για παράδειγμα, το πρόβλημα της πιθανότητας ένα δίκαιο νόμισμα να έρθει κορώνα ή γράμματα με τη μέθοδο Monte-Carlo λύνεται ως εξής: Παράγουμε τυχαίες τιμές στο $[0, 1]$ και μετράμε τις μικρότερες ή ίσες του 0.50 ως γράμματα και τις υπόλοιπες ως κορώνα. Αυτή είναι η προσομοίωση Monte-Carlo του πειράματος της συνεχόμενης ρίψης νομίσματος.

Η αναζήτηση Monte Carlo σε δέντρο ακολουθεί την ίδια λογική και αποτελείται από 2 κύρια στοιχεία. Ένα σχετικά αβαθές δέντρο αναζήτησης και βαθιές προσομοιώσεις. Ξεκινά από μια κατάσταση, την οποία επεκτείνει προς κάθε κατεύθυνση, αλλά σε πολύ μικρό βάθος με ντετερμινιστικό τρόπο, δημιουργώντας έτσι το δέντρο αναζήτησης. Στη συνέχεια, επαναλαμβάνονται τέσσερα βήματα σε κάθε επανάληψη του αλγορίθμου.

```
DATA: root_node
Result: best_move

while(has_time)do
{
    current_node <- root_node

    /*traverse the tree */
    while(current_node is in Tree)do
    {
        last_node <- current_node
        current_node <- Select(current_node)
    }

    /* a node is added */
    last_node <- Expand(last_node)

    /* a simulated game is played */
    R <- Simulate_game(last_node)

    /* the result is backpropagated */
    current_node <- last_node
    while(current_node is in Tree)
    {
        Backpropagate(current_node, R)
        current_node <- current_node.parent
    }
}
return best_move = argmax(root_node)
```

Σχήμα 2.8: Ψευδοκώδικας αναζήτησης Monte Carlo

1. **Επιλογή**: Στο βήμα αυτό διασχίζεται το δέντρο ακολουθώντας τη βέλτιστη διαδρομή μέχρι να βρεθεί κόμβος-φύλλο.
2. **Επέκταση**: Στη συνέχεια προστίθεται ένας κόμβος στο δέντρο ως απόγονος του πρώην τελικού κόμβου.
3. **Προσομοίωση**: Η τιμή του νέου κόμβου υπολογίζεται πιθανοτικά με την ως τώρα πολιτική που ακολουθείται
4. **Οπισθοδιάδοση**: Τέλος, το αποτέλεσμα διαδίδεται στα παραπάνω στρώματα, έως και τη ρίζα του δέντρου, τροποποιώντας τις τιμές του μονοπατιού.



Σχήμα 2.9: Αναζήτηση Monte Carlo σε δέντρο

Το MCTS μπορεί επίσης να θεωρηθεί ως μια μορφή μάθησης με στόχο τη βελτίωση απόδοσης των παρτίδων εξάσκησης. Οι κόμβοι του δέντρου αναζήτησης περιέχουν τη συνάρτηση τιμής για τις θέσεις που προσπελαύνονται κατά τη διάρκεια της αναζήτησης. Κατά τη μέθοδο αυτή, οι τιμές ενημερώνονται σε κάθε προσομοίωση, βελτιώνοντας την παρούσα πολιτική.

3. ΣΧΕΤΙΚΗ ΕΡΕΥΝΑ

Στον χώρο της τεχνητής νοημοσύνης χρησιμοποιούνται συχνά μέθοδοι ενισχυτικής μάθησης. Αν και τα περισσότερα μοντέλα τη συνδυάζουν με επιβλεπόμενη μάθηση, η χρήση μεθόδων που δε βασίζεται σε γνώση που προέρχεται από τον άνθρωπο προσδίδει νέες δυνατότητες στα μοντέλα. Το μεγαλύτερο πλεονέκτημα των τεχνικών αυτών είναι η ελευθερία που δίνεται στα δίκτυα να αναπτύξουν στρατηγικές που δεν επηρεάζονται από ανθρώπινες προδιαθέσεις. Έτσι, μεγάλος αριθμός από μοντέλα μηχανικής μάθησης επωφελούνται από την ενισχυτική μάθηση και πετυχαίνουν καλύτερες αποδόσεις. Μερικά από τα μοντέλα αυτά είναι τα εξής:

3.1 Giraffe

Το Giraffe είναι μια μηχανή για σκάκι που βασίζεται κυρίως στην ενισχυτική μάθηση και χρησιμοποιεί ελάχιστη προϋπάρχουσα γνώση. Ο στόχος αυτής της προσσέγγισης είναι να πραγματοποιήσει εξαγωγή χαρακτηριστικών υψηλού επιπέδου με την χρήση νευρωνικού δικτύου χωρίς να βασίζεται στην δημιουργικότητα του κατασκευαστή της. Επίσης, πραγματοποιεί αναγνώριση προτύπων και καταφέρνει να κατανοεί μοτίβα σε παρόμοιες καταστάσεις του ταμπλό.

3.1.1 Η μέθοδος του Giraffe

Οι αλγόριθμοι που χρησιμοποιούνται είναι ήδη αποδεκτοί στο χώρο της μηχανικής μάθησης και βασίζονται στον TD-Leaf. Ο TD-Leaf είναι ο συνδυασμός του αλγορίθμου temporal-difference learning (TDL) με τον τρόπο που εφαρμόζει ο minimax την εύρεση κόμβων (ελαχίστου - μεγίστου). Η προσέγγιση αυτή επιτρέπει πιο γρήγορη εκμάθηση, εκμεταλλευόμενη την κλίση (gradient) του αλγορίθμου minimax. Δηλαδή χρησιμοποιεί την συνάρτηση αξιολόγησης που καλείται στους κόμβους - φύλλα του αλγορίθμου. Πιο συγκεκριμένα, για κάθε επανάληψη εκπαίδευσης επιλέγονται 256 θέσεις από το σύνολο εκπαίδευσης και αφήνεται το μοντέλο να παίξει με τον εαυτό του για 12 ακόμα κινήσεις. Στην συνέχεια ελέγχεται πόσο άλλαξε η βαθμολογία των κινήσεων και υπολογίζεται το σφάλμα του συνόλου των κινήσεων με βάση το πόσο απέχει το υπάρχον ταμπλό από το αρχικό. Δηλαδή ανατίθενται βάρη στην κάθε κίνηση ανάλογα την "απόσταση" από την αρχική θέση. Το σφάλμα κλιμακώνεται με ρυθμό λ^m, όπου λ είναι σταθερό στο 0,7 και το m δηλώνει τον αριθμό των κινήσεων από την αρχική κατάσταση. Τα βάρη του δικτύου ενημερώνονται με την εξής διαδικασία:

$$w = w + a \sum_{t=1}^{N-1} \nabla J(x_t, w)[\sum_{j=t}^{N-1} \lambda^{j-t} d_t]$$

όπου w είναι το σύνολο βαρών του μοντέλου, α είναι ο ρυθμός εκμάθησης (ορίζεται στο 1 στην περίπτωσή μας), $\nabla J(x_t, w)$ είναι η κλίση του μοντέλου σε κάθε χρονικό σημείο και d_t είναι οι διαφορές από κάθε χρονικό σημείο d_t στο επόμενο.

Σε κάποιες κινήσεις παρατηρείται αλλαγή στη βαθμολογία (χρονική ασυνέπια - temporal inconsistency). Το πρόβλημα οφείλεται στο ότι όλες οι κινήσεις προκαλούν αλλαγή από την αρχική θέση αλλά η βαθμολογία αλλάζει σε συγκεκριμένες κινήσεις, με αποτέλεσμα

να μην γίνεται αντιληπτό ποια ακριβώς κίνηση προκάλεσε την ασυνέπεια. Έτσι, σε κάθε περίπτωση θεωρείται ότι η κίνηση πριν την αλλαγή προκάλεσε την διαφοροποιήση βαθμολογίας. Αυτό έχει ως αποτέλεσμα το δίκτυο να ανακαλύπτει ακολουθίες κινήσεων με μακροπρόθεσμη συνέπεια, δίνοντας όμως προτεραιότητα στις βραχυπρόθεσμες.

3.1.2 Τα αποτελεσματα του Giraffe

Η αξιολόγηση και ο έλεγχος της μηχανής Giraffe επικεντρώθηκαν στη δυνατότητα του μοντέλου να αντιλαμβμάνεται τις στρατηγικές και την σημασία των κινήσεων πάνω στο ταμπλό. Στο μοντέλο δόθηκε μια συλλογή από 1500 θέσεις, οι οποιές χωρίζονταν σε 15 κατηγορίες με 100 θέσεις η κάθε μια. Για παράδειγμα, κάποιες από τις κατηγορίες είναι το πώς αντιλαμβάνεται τις ελεύθερες γραμμές και το πώς διαχειρίζεται τον έλεγχο του κέντρου του ταμπλό. Ουσιαστικά, η κάθε κίνηση είχε μία λίστα από κινήσεις-λύσεις, μαζί με κάποια τιμή αξιολόγησης από το 0-10 (μηδέν η χειρότερη και δέκα η καλύτερη κίνηση). Οπότε, το μοντέλο είχε να επιλέξει κάποια κίνηση-λύση και στο τέλος θα αξιολογούνταν από την τελική συνολική βαθμολογία. Σε αυτό το σημείο να διευκρινιστεί ότι η συλλογή με κινήσεις που δίνονταν στο μοντέλο δεν είχε κάποια σχέση με τα σύνολα αρχικής εκπαίδευσης. Επιπλέον η αξιολόγηση επικεντρώθηκε στην ταχύτητα του μοντέλου καθώς και στο πόσους κόμβους μπορούσε να προσπελάσει και σε πόσο χρονικό διάστημα. Είναι έυκολο να παρατηρηθεί ότι λόγω της έλλειψης υπολογιστικής δύναμης η αξιολόγηση επικεντρώνεται σε στοιχεία που δείχνουν ότι το μοντέλο εκπαιδεύεται και αντιλαμβάνεται καταστάσεις αλλά σε καμία περίπτωση η αξιολόγηση του δεν σχετίζεται με την ολοκλήρωση μιας παρτίδας σε υπεράνθωπο επίπεδο όπως επιτεύχθη λίγα χρόνια αργότερα από επόμενα μοντέλα.

3.2 AlphaGo

Τη μεγάλη αλλαγή στην κοινή γνώμη για τις μεθόδους Ενισχυτικής Μάθησης στον τομέα των παιχνιδιών μηδενικού αθροίσματος τέλειας πληροφορίας έφερε η μηχανή του παιχνιδιού Go, AlphaGo [32]. Ο μεγάλος παράγοντας διακλάδωσης του παιχνιδιού και η δυσκολία στην αξιολόγηση της κάθε κατάστασης το καθιστούσε την μεγαλύτερη πρόκληση στον τομέα της τεχνητής νοημοσύνης. Όμως το 2016, οι Silver et al. εφάρμοσαν μια νέα, επαναστατική μέθοδο και χάρη σε αυτήν κατάφεραν να επιτύχουν επιδόσεις που μέχρι τότε φάνταζαν εξωπραγματικές.

Το AlphaGo βασίζεται στην εκπαίδευση 'δικτύων εκτίμησης' (value networks) για την αξιολόγηση μιας κατάστασης και 'δικτύων πολιτικής' (policy networks) για την επιλογή των κινήσεων. Τα βαθιά αυτά νευρωνικά δίκτυα εκπαιδεύονται με χρήση ενός συνδυασμού παιχνιδιών από ειδικούς και ενισχυτικής μάθησης βασισμένης σε παραγόμενες παρτίδες που παίζει το μοντέλο με τον εαυτό του. Χρησιμοποιώντας Αναζήτηση Δέντρου Monte Carlo, προσομοιώνει χιλιάδες παρτίδες οι οποίες χρησιμοποιούνται για την εκπαίδευση των δικτύων. Χάρη στη μέθοδο αυτή, εκπαιδεύτηκε ένα μοντέλο το οποίο κατάφερε να νικήσει κατηγορηματικά έναν επαγγελματία παίκτη Go, επίτευγμα που θεωρούνταν μια δεκαετία μπροστά από την εποχή του.

3.2.1 Η μέθοδος του AlphaGo

Η πρώτη φάση της εκπαίδευσής του βασίζεται στην εκμάθηση με επίβλεψη. Ένα βαθύ συνελικτικό νευρωνικό δίκτυο χρησιμοποιείται με είσοδο την πλήρη κατάσταση του ταμπλό

και εκπαιδεύεται με ζευγάρια ταμπλό-κινήσεων που παράγονται με τυχαία δειγματοληψία από παιχνίδια ειδικών. Η μέθοδος της Στοχαστικής Βαθμιαίας Ανάβασης, ή Stochastic Gradient Ascent (SGA) χρησιμοποιείται για να συγκλίνει η πρόβλεψη του δικτύου με την πραγματική ανθρώπινη κίνηση. Το δίκτυο αυτό είχε 13 επίπεδα και εκπαιδεύτηκε με 30 εκατομμύρια ταμπλό. Τελικά πέτυχε 57% ακρίβεια στην πρόβλεψη των κινήσεων.

Η δεύτερη φάση της εκπαίδευσης βασίζεται στην ενισχυτική μάθηση. Τα βάρη του δικτύου αρχικοποιούνται στις τιμές των ήδη εκπαιδευμένων παραμέτρων της πρώτης φάσης. Έπειτα, παίζονται παρτίδες με τυχαίες πολιτικές από προηγούμενες χρονικές στιγμές της εκπαίδευσης. Η ποικιλία αντιπάλων συμβάλλει στην αποφυγή της υπερεκπαίδευσης. Τα αποτελέσματα λαμβάνονται από μια συνάρτηση επιβράβευσης η οποία επιστρέφει 0 για κάθε μη τερματική κατάσταση, 1 για νίκη και -1 για ήττα. Η μέθοδος SGA χρησιμοποιείται για να μεγιστοποιηθεί το προβλεπόμενο αποτέλεσμα.

Η τρίτη και τελευταία φάση της εκπαίδευσης είναι η εκτίμηση των καταστάσεων. Στόχος είναι η εκτίμηση μιας συνάρτησης που προβλέπει την έκβαση των παιχνιδιών που παίζονται χρησιμοποιώντας την ήδη εκπαιδευμένη πολιτική και από τους 2 παίκτες. Το δίκτυο που εκπαιδεύεται είναι παρόμοιας αρχικτεκτονικής με το δίκτυο πολιτικής, με την ειδοποιό διαφορά ότι το δίκτυο εκτίμησης παράγει μια μοναδική πρόβλεψη και όχι διαμοιρασμό πιθανότητας για τις διαφορετικές εκβάσεις. Για την εκπαίδευσή του χρησιμοποιείται Στοχαστική Βαθμιαία Κατάβαση, ή Stochastic Gradient Descent (SGD) για να μειωθεί το μέσο τετραγωνικό σφάλμα, ή Mean Squared Error (MSE) μεταξύ της προβλεπόμενης έκβασης και του πραγματικού αποτελέσματος των παρτίδων.

Ο αλγόριθμος AlphaGo συνδυάζει τα δύο αυτά δίκτυα σε έναν αλγόριθμο MCTS και καταφέρνει να ξεπεράσει κάθε προηγούμενο πρόγραμμα Go, αλλά και να νικήσει έναν επαγγελματία άνθρωπο.

3.3 AlphaGo Zero

Το 2017, οι Silver et al. [33] εκπαίδευσαν ίσως το πιο προηγμένο μοντέλο ενισχυτικής μάθησης. Το βελτιωμένο μοντέλο, AlphaGo Zero, επιδεικνύει εξαιρετική επίδοση έναντι όλων των προηγούμενων εκδόσεων του, σημειώνοντας σκορ 100-0 εναντίον της πιο βελτιωμένης έκδοσης του προηγούμενου AlphaGo. Αυτό το κατάφερε με χρήση αποκλειστικά ενισχυτικής μάθησης και μηδενικής προϋπάρχουσας γνώσης.

3.3.1 Η μέθοδος του AlphaGo Zero

Η μέθοδος της επαναληπτικής πολιτικής (iteration policy) [27] είναι ένας αλγόριθμος που γεννά ακολουθίες βελτιωμένων πολιτικών και χωρίζεται σε δύο βασικές κατηγορίες:

1. Την εκτίμηση μιας συνάρτησης τιμής για την δεδομένη πολιτική
2. Την βελτίωση της πολιτικής

Μία απλή προσέγγιση της πρώτης μεθόδου είναι η εκτίμηση μιας συνάρτησης τιμής από το αποτέλεσμα των δειγματοληπτημένων τροχιών (sampled trajctories) [4, 34]. Μία απλή προσέγγιση της δεύτερης μεθόδου είναι η άπληστη επιλογή κινήσεων με κριτήριο μία συνάρτησης τιμής [14].

3.3.2 Η λειτουργία της ενισχυτικής μάθησης

Η κατηγοριοποίηση βασισμένη στην ενισχυτική μάθηση (classification-based) βελτιώνει την πολιτική χρησιμοποιώντας μία απλή αναζήτηση Monte Carlo. Πολλές αναδιπλώσεις εκτελούνται για την κάθε κίνηση και στο τέλος η κίνηση που έδωσε την μεγαλύτερη κατά μέσο όρο τιμή χαρακτηρίζεται ως θετικό - κατάλληλο παράδειγμα εκμάθησης και τα υπόλοιπα ως αρνητικά. Πρακτικά, η πολιτική έχει ως στόχο να κατηγοριοποιήσει τις κινήσεις ως θετικές ή αρνητικές. Μία πιο οργανωμένη και πρόσφατη μέθοδος για την επίτευξη του στόχου αυτού είναι η classification-based modified policy iteration (CBMPI), η οποία εκτελει περικομένες αναδιπλώσεις. Παρ΄ όλα αυτά η τεχνική αυτή οριοθετήθηκε από απλές αναδιπλώσεις και γραμμικούς εκτιμητές χρησιμοποιώντας χαρακτηριστικά επιλεγμένα απευθείας από επιστήμονες μετά από πειραματισμούς.

Ο αλγόριθμος self-play του AlphaGo Zero χρησιμοποιεί το MCTS (Monte Carlo tree search) για τις δύο πολιτικές, την βελτίωση και την αξιολόγηση πολιτικής. Εκτελείται ένα MCTS βάσει των συστάσεων αυτής της πολιτικής και στη συνέχεια προβάλλει την πολιτική αναζήτησης στο χώρο λειτουργίας του νευρωνικού δικτύου. Η αξιολόγηση εφαρμόζεται στην πολιτική αναζήτησης, δηλαδή τα αποτελέσματα των παιχνιδίων με τον εαυτό του προβάλλονται και πάλι στο νευρωνικό δίκτυο μέσω της εκπαίδευσης των υπερπαραμέτρων ώστε το αποτέλεσμα των αξιολογήσεων να ταιριάζει με τις πιθανότητες αναζήτησης και το αποτέλεσμα του παιχνιδιού.

Στην προσέγγιση αλγορίθμου του Go χρησιμοποιήθηκαν δυο τεχνικές όσο αφορά την ενισχυτική μάθηση. Η εκμάθηση χρονικής διαφοράς και η αναζήτηση Monte Carlo σε δέντρα.

3.3.3 Η εκμάθηση του μοντέλου

Η εκμάθηση του AlphaGo Zero αποτελείται από τρία κύρια στοιχεία, τα οποία εκτελούνται παράλληλα και ασύγχρονα. Οι παράμετροι του νευρωνικού δικτυου u_i βελτιστοποιούνται συνεχώς από τα δεδομένα που παράγονται μέσω του παιξίματος με τον εαυτό του. Οι AlphaGo Zero παίκτες au_i αξιολογούνται συνεχώς και ο παίκτης με την καλύτερη απόδοση μέχρι στιγμής, ο au_k, χρησιμοποιείται για τη δημιουργία νέων δεδομένων για την εκμάθηση του επόμενου επιπέδου.

3.3.4 Αξιολόγηση

Για να διασφαλιστεί ότι δημιουργείται πάντα η καλύτερη ποιότητα δεδομένων, αξιολογείται το κάθε νέο σημείο ελέγχου νευρωνικού δικτύου έναντι του τρέχοντος καλύτερου δικτύου fu_i πριν από τη χρήση για τη δημιουργία δεδομένων. Το νευρωνικό δίκτυο fu_i αξιολογείται από την απόδοση του σε μια αναζήτηση MCTS au_i, η οποία χρησιμοποιεί το fu_i για να αξιολογήσει τις θέσεις των φύλλων καθώς και τις επικρατέστερες πιθανότητες. Κάθε αξιολόγηση αποτελείται από 400 παιχνίδια, χρησιμοποιώντας ένα MCTS με 1.600 προσομοιώσεις για την επιλογη κάθε κίνησης με μια ελάχιστη θερμοκρασία (temperature) $t \to 0$ (δηλαδή, επιλέγει αποφασιστικά την κίνηση με τον μέγιστο αριθμό επισκέψεων. Αυτό σημαίνει ότι δίνουμε βάση στην πιθανότερη-συχνότερη κίνηση). Εάν ο νέος παίκτης κερδίσει με περιθώριο μεγαλύτερο του 55% (για να αποφευχθεί η επιλογή noise alone), τότε γίνεται ο καλύτερος παίκτης au_k, και στη συνέχεια χρησιμοποιείται για τη δημιουργία παιχνιδιών με τον εαυτό του, και γίνεται η βάση για τις επόμενες συγκρίσεις.

3.4 Παραλλαγές του AlphaGo Zero

Η επικράτηση των μεθόδων του AlphaGo Zero ως την πιο προηγμένη προσέγγιση εκμάθησης μηχανών για παιχνίδια, επέφερε άνοδο στη δημοτικότητα των μεθόδων αυτών. Πολλοί επιχείρησαν να αναπαράγουν, να βελτιώσουν και να επεκτείνουν τη μέθοδο των Silver et al., με διάφορους βαθμούς επιτυχίας. Η δυσκολία εντοπίζεται κυρίως στην υπολογιστική δύναμη που είχε η ομάδα της DeepMind στη διάθεσή της κατά την εκπαίδευση του επαναστατικού τους μοντέλου. Χωρίς ανάλογο υλικό, το να ξεπεραστούν οι αποδόσεις του AlphaGo Zero καθίσταται μάλλον αδύνατο.

Η δυσκολία αυτή όμως δεν εμπόδισε νέες τεχνικές, βασισμένες στο AlphaGo Zero να δημιουργηθούν και να δοκιμαστούν σε παιχνίδια παρόμοια με το Go, αν και υπολογιστικά ευκολότερα. Πλήθος δοκιμών σε διάφορα παιχνίδια απέδειξε ότι οι μέθοδοι των Silver et al. επιτυγχάνουν ικανοποιητικές αποδόσεις και εκτός του Go. Για παράδειγμα, στην ιστοσελίδα του πανεπιστημίου Stanford, υπάρχει μελέτη με τελείως ανοιχτό κώδικα που χρησιμοποιεί την μέθοδο των αναδιπλώσεων Monte Carlo του AlphaGo Zero για να επιτύχει υπεράνθρωπη επίδοση στα παιχνίδια othello, connect 4 και άλλα [23].

Η ύπαρξη repositories όπως αυτό δηλώνει τη σημασία της ενισχυτικής μάθησης και του AlphaGo Zero στον τομέα της τεχνητής νοημοσύνης. Η προσέγγιση που ακολουθήθηκε για το παιχνίδι Go επεκτείνεται και μεταβάλλεται συνεχώς για να χρησιμοποιηθεί και σε άλλα παιχνίδια και αργότερα σε άλλα, γενικότερα προβλήματα που ίσως μπορούν να επιλυθούν με την ενισχυτική μάθηση. Η ομάδα της DeepMind αργότερα παρουσίασε έναν αλγόριθμο που γενικεύει την προηγούμενή τους δουλειά και μπορεί να χρησιμοποιηθεί με εξίσου καλά αποτελέσματα σε ένα σύνολο παιχνιδιών.

3.5 AlphaZero

Ακολουθώντας την επιτυχία του AlphaGo και AlphaGo Zero, οι David Silver et al. δημιούργησαν ένα μοντέλο κατάλληλο για μεγαλύτερο εύρος παιχνιδιών. Ο αλγόριθμος AlphaZero είναι μια γενικευμένη μορφή του AlphaGo Zero, με τη βοήθεια του οποίου επιτεύχθηκαν υπεράνθρωπες επιδόσεις στα παιχνίδια σκάκι, Go και shogi, επίσης γνωστό ως ιαπωνικό σκάκι. Οι επιδόσεις αυτές χρειάστηκαν μόνο 24 ώρες εκμάθησης για να πραγματοποιηθούν. Το AlphaZero επίσης νίκησε από ένα τελευταίας τεχνολογίας πρόγραμμα σε καθένα από τα παιχνίδια αυτά.

3.5.1 Η μέθοδος του AlphaZero

Το AlphaZero λειτουργεί με παρόμοια μέθοδο με το AlphaGo. Η εκμάθηση πραγματοποιείται μέσω παιχνιδιών με τον εαυτό του. Στη συνέχεια, τα αποτελέσματα των παιχνιδιών αποτελούν νέα δεδομένα για τροφοδοσία στο νευρωνικό δίκτυο το οποίο παράγει την επόμενη γενιά μοντέλου. Κάθε νέο μοντέλο αξιολογείται παίζοντας παρτίδες με το μέχρι τώρα καλύτερο και αν το ποσοστό νίκης του ξεπερνάει το 55%, τότε το νέο μοντέλο παίρνει τη θέση του καλύτερου. Τα νέα δεδομένα παράγονται πάντα από το πιο αποδοτικό μοντέλο μέχρι εκείνη τη χρονική στιγμη. Για την ανανέωση του μοντέλου χρησιμοποιείται Stohastic Gradient Decent (SGD) στην εξής συνάρτηση λάθους:

$$l = (z - v)^2 - \pi^T \log p + c||\theta||^2$$

Η συνάρτηση αυτή ανανεώνει τις παραμέτρους του δικτύου με στόχο τη μείωση του σφάλματος μεταξύ της πρόβλεψης του αποτελέσματος v, με το πραγματικό αποτέλεσμα z, καθώς και τη μεγιστοποίηση της ομοιότητας του διανύσματος πολιτικής p με τις πιθανότητες της αναζήτησης Monte Carlo π. Το c είναι ο βαθμός μάθησης με τον οποίο ανανεώνονται οι παράμετροι θ του δικτύου.

3.5.2 Διαφορές με το AphaGo Zero

Οι επιπλέον δυσκολίες του AlphaZero σε σχέση με το AlphaGo Zero προέρχονται από τις ειδοποιούς διαφορές μεταξύ των παιχνιδιών. Σε αντίθεση με το Go, το σκάκι και το shogi έχουν κανόνες που βασίζονται στην θέση που έχουν τα πιόνια πάνω στο ταμπλό. Για παράδειγμα, στο σκάκι ένα στρατιωτάκι μπορεί να μετακινηθεί ένα ή δύο τετράγωνα αν βρίσκεται στην αρχική του θέση. Τέτοιου είδους κανόνες αφαιρούν την ιδιότητα της συμμετρίας, η οποία συναντάται στο Go. Αυτό αυξάνει σημαντικά το μέγεθος του συνόλου κινήσεων, αφού η κάθε κίνηση εξαρτάται από την αρχική και την τελική θέση του πιονιού. Επίσης, ενώ στο AlphaGo, το ταμπλό μπορεί να περαστεί σαν είσοδος στο νευρωνικό δίκτυο πολλαπλές φορές, λαμβάνοντας υπόψιν τη συμμετρία του παιχνιδιού, κάτι τέτοιο στο σκάκι και το shogi είναι αδύνατον. Έτσι, η αρχιτεκτονική του AlphaZero διαφέρει από αυτή του AlphaGo, με στόχο μια γενικότερη στρατηγική εκμάθησης για μεγαλύτερο εύρος παιχνιδιών.

Οι διαφορές μεταξύ AlphaGo και AlphaZero επεκτείνονται και στη διαδικασία της εκπαίδευσης. Το AlphaZero ανανεώνει το δίκτυο συνεχόμενα, χωρίς να περιμένει να ολοκληρωθεί μια εποχή εκμάθησης, προσθέτοντας δυναμικότητα στην εκπαίδευση του μοντέλου. Στα παραδείγματα εκπαίδευσης επίσης προστίθεται η ισοπαλία σαν πιθανή έκβαση, αφου σε αντίθεση με το Go, τα άλλα δύο παιχνίδια μπορούν να καταλήξουν σε ισοπαλία. Μάλιστα, η κοινή γνώμη είναι ότι η βέλτιστη λύση για το σκάκι (αν αυτή ποτέ βρεθεί), είναι ισοπαλία.

3.5.3 Τα αποτελέσματα του AlphaZero

Το AlphaZero εκπαιδεύτηκε για 700.000 βήματα ομαδοποιώντας τα δεδομένα σε πακέτα 4096 παραδειγμάτων. Για την εκπαίδευση χρησιμοποιήθηκαν 5000 μονάδες επεξεργασίας τενσόρων (TPUs) πρώτης γενιάς για την παραγωγή των παιχνιδιών και 64 TPUs δεύτερης γενιάς για να εκπαιδευτεί το νευρωνικό δίκτυο.

Με το υλικό αυτό, το AlphaZero κατάφερε να ξεπεράσει τους προκατόχους του στα τρία προαναφερθέντα παιχνίδια. Στο σκάκι, μέσα σε μόνο 4 ώρες εκμάθησης (300.000 βήματα), έφτασε σε δυναμικότητα το Stockfish, το καλύτερο ανοιχτού λογισμικού μοντέλο το οποίο ήταν πρωταθλητής κόσμου το 2016 μεταξύ των μηχανών. Στο shogi, το AlphaZero ξεπέρασε τη μηχανή πρωταθλητή του 2017, το Elmo, με μόνο 2 ώρες εκμάθησης (110.000 βήματα). Τέλος, στο Go ξεπέρασε εύκολα το AlphaGo Lee μέσα σε 8 ώρες (165.000 βήματα) και νίκησε ακόμα και την τότε πιο προηγμένη μηχανή, AlphaGo Zero με επαρκή εκμάθηση 700.000 βημάτων. Η απόδοσή του στο τέλος ήταν παρόμοια με αυτή του AlphaGo Zero, το οποίο όμως είχε εκπαιδευτεί σε βάθος τριών ημερών.

3.6 Η ενισχυτική μάθηση σε παιχνίδια μη τέλειας πληροφορίας

Οι ωφέλειες της ενισχυτικής μάθησης επεκτείνονται και σε παιχνίδια χωρίς τέλεια πληροφορία (imperfect information games). Η πρόσθετη δυσκολία των παιχνιδιών αυτών πηγάζει από το γεγονός ότι ο κάθε παίκτης βασίζεται κατά ένα ποσοστό στην τύχη για την επιλογή της στρατηγικής του, αφού δεν γνωρίζει όλες τις πληροφορίες που μπορούν να επηρεάσουν την κρίση του κατά την λήψη μίας απόφασης. Για ένα μοντέλο μηχανικής μάθησης, η δυσκολία αυτή ανάγεται σε δυσκολία της εκπαίδευσής του. Το μοντέλο χρειάζεται να αναπτύξει μια στρατηγική που επιφέρει ικανοποιητικά αποτελέσματα, χωρίς όμως να έχει στη διάθεσή του κάθε πληροφορία για τις κινήσεις του αντιπάλου του.

Είναι προφανές ότι χωρίς τέλεια πληροφορία, πάντα θα υπάρχει κάποια αβεβαιότητα ως προς την εκτίμηση της ορθότητας μιας ενέργειας έναντι κάποιας άλλης. Το 2016, οι Johannes Heinrich και David Silver, δημιούργησαν έναν αλγόριθμο [13] που αποδίδει στα ίδια επίπεδα με τα πιο προηγμένα προγράμματα σε παραλλαγές του poker. Ο αλγόριθμος αυτός βασίζεται εξ ολοκλήρου σε ενισχυτική μάθηση και δε χρησιμοποιεί καθόλου προϋπάρχουσα γνώση.

3.6.1 Background

Αρχικά, ας αναλύσουμε μερικές βασικές έννοιες:

- on-policy: Είναι η κατάσταση κατά την οποία ο πράκτορας μαθαίνει από την τρέχουσα πολιτική που ακολουθεί.
- off-policy: Είναι η κατάσταση κατά την οποία ο πράκτορας μαθαίνει από άλλες πολιτικές, για παράδειγμα την προηγούμενη
- Nash equilibrium: Η ισορροπία Nash στην θεωρία παιγνίων αποτελεί ένα σύνολο λύσεων για μη συνεργατικά παιχνίδια. Υποθέτουμε ότι ο κάθε παίκτης γνωρίζει τις στρατηγικές ισορροπίας που θα ακολουθήσουν οι άλλοι παίκτες και δεν οφελείται σε τίποτα να αλλάξει μόνο την δική του στρατηγική. Αν κάποιος παίκτης δεν έχει κέρδος από την αλλαγή στρατηγικής του, ενώ όλοι οι άλλοι παίκτες διατηρούν τις δικές τους σταθερές, τότε το σετ των στρατηγικών και των αποτελεσμάτων τους βρίσκεται σε ισορροπία Nash.
- Extensive-form games: Είναι ένα μοντέλο από συνεχείς εξελισσόμενες αλληλεπιδράσεις μεταξύ πολλαπλών παικτών.

Τα παιχνίδια στα οποία ο παίκτης δεν έχει τέλεια πληροφορία, δηλαδή δεν γνωρίζει όλες τις κινήσεις που έγιναν ή τις δυνατές κινήσεις του αντιπάλου αλλά γνωρίζει πληροφρία μόνο για την δική του κατάσταση ονομάζονται imperfect information games. Για παράδειγμα, στο poker ο κάθε παίχτης γνωρίζει τις κάρτες του και όχι των υπολοίπων. Σε τέτοιου τύπου παιχνίδια θεωρείται ότι ο κάθε παίκτης προσπαθεί να αυξήσει το κέρδος του σε σχέση με τους υπολοίπους.

3.6.2 NSFP

Ο αλγόριθμος, ο οποίος πέτυχε υπεράνθρωπο αποτέλεσμα, ονομάζεται Neural Fictitious Self-Play (NFSP) [13]. Πιο συγκεκριμένα ο NFSP συνδυάζει τον αλγόριθμο Fictitious Self-Play (FSP) με ένα νευρωνικό δίκτυο ως συνάρτηση εκτίμησης.

Το Fictitious play είναι ένα θεωρητικό μοντέλο παιχνιδιού που μαθαίνει παίζοντας με τον εαυτό του. Στο μοντέλο αυτό, οι παίκτες επιλέγουν την καλύτερη αντίδραση σε σχέση με την κατά μέσο όρο συμπεριφορά των αντιπάλων τους. Σε συγκεκριμένες κατηγορίες παιχνιδιών η προηγούμενη μέση στρατηγική συγκλίνει στην ισορροπία Nash. Το 2006 έγινε αναφορά στο γενικευμένο αποδυναμωμένο fictitious play, το οποίο είχε παρόμοιες εγγυήσεις σύγκλισης με τον προκάτοχο του, με την διαφόρα ότι επέτρεπε την εκτίμηση των καλύτερων αντιδράσεων των παικτών, καθώς και την διαταραχή των ενημερώσεων της μέσης στρατηγικής. Οι αλλαγές ταίριαξαν τον αλγόριθμο με την μηχανική μάθηση. Το 2015 έγινε εισαγωγή μιας νέας έννοιας, με όνομα Full-Width Extensive-Form Fictitious Play (XFP), η οποία επιτρέπει στους παίκτες να ενημερώσουν την στρατηγική τους συμπεριφορά (Extensive-Form), σε γραμμικό χώρο και χρόνο. Την ίδια χρονιά αναπτύχθηκε ο αλγόριθμος Fictitious Self-Play (FSP). Ο αλγόριθμος αυτός αντικαθιστά τον υπολογισμό της καλύτερης ενέργειας και την ενημέρωση της μέσης στρατηγικής με ενισχυτική και με επίβλεψη μάθηση αντίστοιχα.

Οι πληροφορίες εκμάθησης αποθηκεύονται σε δύο ξεχωριστές μνήμες. Στην M_{RL}, ή αλλιώς reinforcement learning memory, αποθηκεύεται η εμπειρία που έχει ο κάθε πράκτορας, ενώ στην M_{SL} (supervised learning memory) αποθηκεύεται η στρατηγική συμπεριφορά του ίδιου του πράκτορα.

3.6.3 Περιγραφή του αλγορίθμου

Στο σημείο αυτό είμαστε σε θέση να εξετάσουμε τον αλγόριθμο NFSP. Αρχικά, ο πράκτορας εκπαιδεύει ένα νευρωνικό δίκτυο, ώστε να προβλέψει τιμές ενεργειών μέσω της μνήμης M_{RL} χρησιμοποιώντας off-policy ενισχυτική μάθηση. Το νευρωνικό δίκτυο οριστικοποιεί την προσέγγιση της στρατηγικής του πράκτορα. Αυτό συμβαίνει διαλέγοντας μία τυχαία ενέργεια με κάποια πιθανότητα και από την άλλη επιλέγει την κίνηση που θα μεγιστοποιήσει τις τιμές ενεργειών που έχει προβλέψει. Στην συνέχεια ο πράκτορας εκπαιδεύει ένα ξεχωριστό δίκτυο, το οποίο προσπαθεί να μιμηθεί την προηγούμενή του καλύτερη συμπεριφορά χρησιμοποιώντας μάθηση με επίβλεψη μέσω της μνήμης M_{SL}. Το τελευταίο δίκτυο συνδυάζει κινήσεις και καταστάσεις και δημιουργεί την κατά μέσο όρο στρατηγική του πράκτορα.

Τέλος για να γίνει ο αλγόριθμος πιο σταθερός χρησιμοποιεί δύο μεθόδους: Πρώτον, την μέθοδο της δειγματοληψίας από κάποια μνήμη, ώστε να αποφύγει το πρόβλημα του window artifact. Δεύτερον, χρησιμοποιεί δυναμική πρόληψη που επιτρέπει σε κάθε πράκτορα να δειγματοληπτεί την δική του καλύτερη συμπεριφορά αλλά και πιο αποτελεσματικά τις αλλαγές στην συμπεριφορά των αντιπάλων του.

3.6.4 Συμπεράσματα για τον NSFP

Ο αλγόρθιμος NSFP είναι ο πρώτος αλγόριθμος που χρησιμοποιεί εξ ολοκλήρου βαθιά νευρωνικά δίκτυα για την εκμάθησή του με στόχο την ισορροπία Nash στα παιχνίδια μη τέλειας πληροφορίας, εκμεταλλευόμενος παρτίδες με τον εαυτό του. Είναι επίσης η πρώτη μέθοδος που αποδείχτηκε ότι έχει δυνατότητες κλιμάκωσης, δίχως να χρησιμοποιεί προϋπάρχουσα γνώση. Πειραματικά φάνηκε ότι ο NSFP συγκλίνει σε προσσεγγιστική ισορροπία Nash σε μικρές παρτίδες πόκερ, ενώ διάφορες άλλες προηγμένες στρατηγικές αποτυγχάνουν. Ο NSFP είναι σε θέση να ανταγωνίζεται τα πιο εξελιγμένα υπεράνθρωπα προγράμματα σε παιχνίδια μη τέλειας πληροφορίας χρησιμοποιώντας αποκλειστικά ενισχυτική μάθηση, αποδεικνύοντας την αποτελεσματικότητά της και σε αυτόν τον τομέα.

4. ΤΟ ΜΟΝΤΕΛΟ ΜΑΣ

Η προσέγγιση που ακολουθήθηκε είναι βασισμένη στα μοντέλα των AlphaGo Zero και AlphaZero. Ορισμένες αλλαγές κρίθηκαν αναγκαίες λόγω των υλικών περιορισμών που υπήρχαν, όμως ο αλγόριθμος παραμένει πιστός στις μεθόδους των μοντέλων αυτών, ακολουθώντας παρόμοιες διαδικασίες.

Η υλοποίηση βασίζεται στην εκμάθηση μέσω παιχνιδιών του μοντέλου με τον εαυτό του (self-play). Η εκτέλεση των παιχνιδιών γίνεται με τη μέθοδο του Monte Carlo tree search (MCTS), το οποίο με πολλαπλές προσομοιώσεις δημιουργεί δέντρα κινήσεων και καταστάσεων ώστε να μπορεί να αξιολογεί όσο το δυνατόν καλύτερα κάποια κατάσταση (ταμπλό του σκακιού) και την κίνηση που προκύπτει πιο συχνά από την κατάσταση αυτή. Όταν ολοκληρωθούν τα self-play, το πρόγραμμα δημιουργεί μια λίστα με παραδείγματα εκπαίδευσης, τα οποία τροφοδοτούν το δίκτυο. Από αυτά τα παραδείγματα, το δίκτυο εκπαιδεύεται ώστε να υπολογίζει την πολιτική p_i, που είναι η πιθανότητα κάθε κίνησης σε κάθε κατάσταση, αλλά και την αξιολόγηση v της κάθε κατάστασης (πόσο πιθανό είναι να οδηγήσει στη νίκη). Όταν το δίκτυο τελειώσει τις εποχές εκπαίδευσής, αρχίζει να παίζει παιχνίδια με μία προηγούμενη έκδοσή του, με στόχο την αξιολόγηση της πιο πρόσφατης περιόδου της διαδικασίας της εκπαίδευσης. Σε κάθε αξιολόγηση παρατηρείται ένα ποσοστό νικών σε σχέση με την προηγούμενη έκδοση ώστε να διαπιστωθεί εάν η νέα περίοδος εκπαίδευσης δημιούργησε αναβαθμισμένο ή υποβαθμισμένο μοντέλο. Σε κάθε περίπτωση το δίκτυο διατηρεί την φαινομενικά πιο ισχυρή έκδοση με βάση το ποσοστό νικών.

4.1 Μοντελοποίηση

Η μοντελοποιήση του παιχνιδιού έχει δύο επίπεδα. Το πρώτο επίπεδο κωδικοποιεί το ταμπλό και τις κινήσεις των πιονιών πάνω σε αυτό, ενώ το δεύτερο διαχειρίζεται τους πιο σύνθετους κανόνες του παιχνιδιού. Το πρώτο επίπεδο ονομαζόμενο Board είναι υπεύθυνο για την κωδικοποίηση του ταμπλό όσον αφορα τις συντεταγμένες και τις κινήσεις του κάθε διαφορετικού πιονιού. Πιο συγκεκριμένα, το κάθε πιόνι αντιστοιχείται σε έναν θετικό αριθμό από το 1 έως το 6 για τον παίκτη με τα λευκά και από το -1 έως το -6 για τον παίκτη με τα μαύρα αντίστοιχα. Το ταμπλό κωδικοποιείται ως ένας δισδιάστατος πίνακας και για τις συντεταγμένες υπάρχει άμεση σχέση μεταξύ γραμμών-στηλών, που στη γλώσσα του σκακιού κωδικοποιούνται στα ζέυγη [(a-h),(1-8)], με το κάθε γράμμα να δηλώνει μια στήλη από αριστερά προς τα δεξιά και τον κάθε αριθμό μια γραμμή από κάτω προς τα πάνω. Οι συντεταγμένες λαμβάνονται από την οπτική του παίκτη με τα λευκά. Επιπλέον, σε αυτό το επίπεδο γίνεται η εύρεση του συνόλου επιτρεπτών κινήσεων για κάθε πιόνι, καθώς και η εφαρμογή κάποιας κίνησης όταν αυτή έχει επιλεγεί.

Το δεύτερο επίπεδο ονομάζεται Game και κωδικοποιεί όλες τις επιπλέον αναγκαίες πληροφορίες και κανόνες που είναι απαραίτητα για την διεξαγωγή της παρτίδας. Περιέχει πληροφοριες όπως την κατεύθυνση των πιονιών, καθώς αυτή αλλάζει ανάλογα με το χρώμα του τωρινού παίκτη, τον έλεγχο μεμονωμένων κινήσεων που εκτελούνται υπό προϋποθέσεις (en passant, castling), καθώς και πληροφορίες για τον έλεγχο ισοπαλιών όπως αν υπάρχουν 50 κινήσεις χωρίς κάποια πρόοδο ή αν έχει εμφανιστεί το ίδιο ταμπλό τρείς φορές. Η οντότητα Game είναι αναγκαία και επαρκής για να κωδικοποιήσει εξ ολοκλήρου μια παρτίδα σκάκι, καθώς η ίδια περιέχει μια οντότητα Board και λειτουργεί ως διεπαφή για αυτήν. Το Game είναι η δομή που μοντελοποιεί πλήρως το παιχνίδι.

4.2 Monte Carlo Tree Search

Το Monte Carlo Tree Search έχει ως στόχο να βοηθήσει το νευρωνικό δίκτυο κατά την περίοδο εκμάθησης στο να εκτιμήσει και να δημιουργήσει μία πολιτική από μία δοσμένη κατάσταση s προς την επόμενη κίνηση. Ουσιαστικά δημιουργεί ένα δέντρο κινήσεων, το οποίο στην αρχή είναι άδειο και στην ρίζα έχει την αρχική κατάσταση (αρχικό ταμπλό). Κάθε φορά προχωράμε σε μία επόμενη κατάσταση. Αν αυτή η κατάσταση έιναι τερματική, αποθηκεύεται η έκβαση της παρτίδας και διαδίδεται προς τα πίσω, αλλαγμένη από τη σκοπιά του κάθε παίκτη. Αν φτάσει σε κόμβο φύλλο, επιλέγει την επόμενη κατάσταση βαθμολογώντας την μέσω του δικτύου και τέλος επιστρέφει την προβλεπόμενη έκβαση από τη σκοπιά του κάθε παίκτη στα ανώτερα επίπεδα του δέντρου, ανάλογα με το μονοπάτι που ακολούθησε. Στην περίπτωση του σκακιού ελέγχει και αν ένα παιχνίδι καταλήγει σε ισοπαλία, στην οποία περίπτωση ενημερώνει αντίστοιχα τον κόμβο χωρίς να τον εισάγει στους τερματικούς. Αυτό συμβαίνει γιατί αν η παρτίδα τερματίσει με ισοπαλία σε μια κατάσταση, δεν είναι αναγκαίο ότι η κατάσταση αυτή θα τερματίζει πάντα σε ισοπαλία.

Ας αναλύσουμε τον αλγόριθμο με περισσότερη λεπτομέρια. Για κάθε κατάσταση s, ή ζεύγος κατάστασης s και ενέργειας a, αποθηκεύονται τα εξής:

- $Q(s, a)$: Η προβλεπόμενη επιβράβευση για την κίνηση στην κατάσταση s
- $N(s, a)$: Οι φορές στις οποίες από την κατάσταση s επιλέξαμε την κίνηση
- N_s : Οι φορές που επισκεφτήκαμε την κατάσταση s
- $P(s)$: Η πολιτική που επιστρέφει το δίκτυο
- E_s : Το πραγματικό αποτέλεσμα της κατάστασης s
- V_s : Οι επιτρεπόμενες κινήσεις από την κατάσταση s
- c_{puct} : Υπερπαράμετρος που ελέγχει το βαθμό εξερεύνησης των κινήσεων

Με τις παραπάνω τιμές μπορούμε να υπολογίσουμε το άνω όριο πιθανότητας

$$U(s, a) = Q(s, a) + c_{puct} P(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)}$$

το οποίο αποτελεί την τιμή 'εμπιστοσύνης' που δείχνει ο αλγόριθμος στην κίνηση a, όταν βρίσκεται στην την κατάσταση s.

Μια απλή εκτέλεση του αλγορίθμου λειτουργεί ως εξής. Αρχικά υπολογίζει και επιλέγει μία κίνηση a που μεγιστοποιεί το άνω όριο πιθανότητας $U(s, a)$ για την κατάσταση s και παράγει μια νέα κατάσταση s'. Αν η s' υπάρχει στο δέντρο, καλεί αναδρομικά την συνάρτηση για την s'. Σε αντίθετη περίπτωση προσθέτει ένα νέο κόμβο στο δέντρο και μέσω του δικτύου υπολογίζει την πολιτική $P(s')$ και την εκτίμηση $v(s')$ της κατάστασης. Επίσης, αρχικοποιεί τα $Q(s', a)$ και $N(s', a)$. Τέλος ακολουθεί το δέντρο προς τα άνω στρώματα και μέσω των $v(s')$ ενημερώνει όλες τις προηγούμενες $Q(s, a)$. Αντίθετα, αν η κίνηση είναι τερματική επιστρέφει την αντίστοιχη τιμή που δηλώνει το αποτέλεσμα τις παρτίδας.

4.3 Δίκτυο

Το έργο των AlphaGo, AlphaGo Zero και AlphaZero απέδειξε ότι για την εκπαίδευση ενός μοντέλου για παιχνίδι μηδενικού αθροίσματος και τέλειας πληροφορίας, υπάρχουν πολλαπλές αρχιτεκτονικές που επιτυγχάνουν ικανοποιητικές αποδόσεις. Η προσέγγιση που ακολουθήσαμε εμείς είναι η χρήση ενός συνελικτικού δικτύου (convolutional network) το οποίο δημιουργεί την πολιτική και αξιολογεί τις καταστάσεις ταυτόχρονα. Τα στρώματα συνέλιξης χρησιμοποιούνται εναλλάξ με κανονικοποίηση πακέτου (Batch normalization) και τελικά καταλήγουν σε πλήρως συνδεδεμένα στρώματα, τα οποία παράγουν το διάνυσμα p της πολιτικής και την πρόβλεψη v της έκβασης της παρτίδας.

4.3.1 Δεδομένα εισόδου και εξόδου

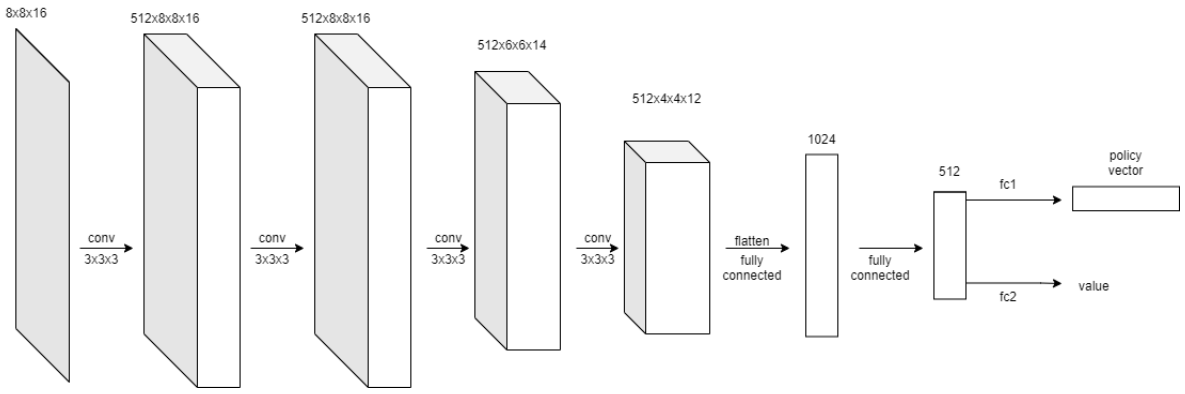
Σαν είσοδο για το δίκτυο δίνονται πίνακες που αντιπροσωπεύουν τις καταστάσεις του ταμπλό. Η μορφή της εισόδου είναι διαστάσεων $8 \times 8 \times 16$. Οι πρώτες 2 διαστάσεις είναι οι συντεταγμένες x, y της κάθε θέσης. Για κάθε θέση στη σκακιέρα δίνονται πληροφορίες που αντιπροσωπεύουν τα εξής:

- Τα πρώτα 12 κελιά είναι μια one-hot αναπαράσταση του πιονιού και χρώματος αυτού. Υπάρχουν 6 διαφορετικά είδη πιονιού στο σκάκι για κάθε παίκτη, οπότε συνολικά χρειάζονται 12, από τα οποία ένα είναι 1 και όλα τα άλλα 0. Αν δεν υπάρχει 1, το τετράγωνο αυτό στη σκακιέρα είναι κενό.
- Το χρώμα του παίκτη που έχει την επόμενη κίνηση
- Έναν μετρητή που δείχνει τις συνολικές κινήσεις της παρτίδας
- Έναν μετρητή για τις κινήσεις που έχουν γίνει χωρίς να υπάρχει πρόοδος
- Έναν μετρητή για τις φορές που έχει εμφανιστεί η συγκεκριμένη κατάσταση στην παρτίδα. Στις 3 επαναλήψεις, το παιχνίδι λήγει σε ισοπαλία.

Το AlphaZero στην αρχιτεκτονική του χρησιμοποιεί επίσης ένα ιστορικό κινήσεων, δίνοντας στο δίκτυο μερικές ακόμα δωδεκάδες κελιά για κάθε θέση, τα οποία αναπαριστούν το είδος και χρώμα πιονιού που υπήρχε εκεί σε προηγούμενες χρονικές στιγμές. Η διαδικασία αυτή απορρίφθηκε καθώς η αποθήκευση ιστορικού ήταν αρκετά ακριβή σε μνήμη και η επεξεργασία του για την δημιουργία των δεδομένων επιβάρυνε αρκετά την υπολογιστική δύναμη της μοναδικής GPU που ήταν διαθέσιμη στην εκπαίδευση του μοντέλου.

4.3.2 Αρχιτεκτονική

Τα δεδομένα στην παραπάνω μορφή ανακατεύονται με τυχαίο τρόπο και χωρίζονται σε πακέτα μεγέθους 64 παραδειγμάτων. Τα πακέτα αυτά περνούν από 4 συνελικτικά στρώματα τριών διαστάσεων με μέγεθος kernel 3x3x3. Έπειτα, περνούν μέσα από 2 πλήρως συνδεδεμένα στρώματα που καταλήγουν σε ένα διάνυσμα μεγέθους 512. Τέλος, υπάρχουν 2 ξεχωριστά πλήρως συνδεδεμένα στρώματα τα οποία παράγουν το διάνυσμα πολιτικής και την πρόβλεψη έκβασης από κάθε κατάσταση. Η πρόβλεψη ορίζεται 1 για νίκη, -1 για ήττα και 0 για ισοπαλία.



Σχήμα 4.1: Η αρχιτεκτονική του δικτύου

Το διάνυσμα πολιτικής έχει μέγεθος 64 * 64. Η κάθε κίνηση κωδικοποιείται με την αρχική και την τελική θέση του πιονιού. Στο τέλος του δικτύου, οι πιθανότητες για κάθε κίνηση παράγονται από λογαριθμική softmax συνάρτηση που εφαρμόζεται στο τελευταίο πλήρως συνδεδεμένο επίπεδο. Αντίστοιχα, η πρόβλεψη νίκης παράγεται από συνάρτηση tanh στο δικό του τελευταίο επίπεδο. Τελικά, το νευρωνικό δίκτυο με είσοδο μια κατάσταση του ταμπλό, παράγει μια πολιτική για την κατάσταση αυτή και μια εκτίμηση του αποτελέσματος του παιχνιδιού για τον συγκεκριμένο παίκτη.

4.4 Εκπαίδευση Μοντέλου

Το μοντέλο αρχικά παίζει μια σειρά από παρτίδες με τον εαυτό του. Αυτό πρακτικά σημαίνει ότι ο ίδιος ακριβώς αλγόριθμος χρησιμοποιείται για την επιλογή κινήσεων από την πλευρά και του άσπρου και του μαύρου. Για κάθε κίνηση χρησιμοποιείται MCTS για την επιλογή της καλύτερης έκβασης σύμφωνα με την πολιτική του δικτύου. Εφ' όσον αρχικά το δίκτυο έχει τυχαία αρχικοποιημένες παραμέτρους, οι κινήσεις επιλέγονται τυχαία κατά την πρώτη επανάληψη της εκπαιδευτικής διαδικασίας. Κατά την διεκπεραίωση της κάθε παρτίδας, κάθε κατάσταση του ταμπλό αποθηκεύεται, μαζί με την πολιτική για την κατάσταση αυτή και τον παίκτη που έχει σειρά.

Στο τέλος κάθε παιχνιδιού, η έκβαση προσκολλάται σε κάθε μία από τις καταστάσεις που έχουν αποθηκευτεί, αλλαγμένη για να αντιπροσωπεύει την νίκη ή την ήττα του παίκτη που έκανε κίνηση στην συγκεκριμένη κατάσταση. Η κατάσταση αυτή τελικά έρχεται στη μορφή που περιγράφεται στην αρχιτεκτονική του δικτύου, με σκοπό να χρησιμοποιηθεί ως παράδειγμα για την εκπαίδευσή του.

Μετά από συγκεκριμένο αριθμό παιχνιδιών, τα παραδείγματα ανακατεύονται, χωρίζονται σε πακέτα μεγέθους που ορίζεται από τις υπερπαραμέτρους του δικτύου και ξεκινάει η εκπαίδευση του νευρωνικού δικτύου. Τα παραδείγματα αρχικά περνούν μέσα από το δίκτυο, παράγοντας μια πολιτική και μια προβλεπόμενη κατάληξη για το καθένα. Οι τιμές αυτές, που παράγονται από το δίκτυο, συγκρίνονται με τις πραγματικές τιμές πολιτκής και έκβασης του παιχνιδιού, όπως έγιναν πραγματικά στο παιχνίδι. Για τη σύγκριση χρησιμοποιείται η εξής συνάρτηση λάθους:

$$l = (z - v)^2 - \pi^T p + c||\theta||^2$$

η οποία είναι παρόμοια με αυτήν που χρησιμοποιείται στον αλγόριθμο AlphaZero. Η συνάρτηση επιχειρεί να μειώσει την τετραγωνική διαφορά μεταξύ της προβλεπόμενης έκβα-

σης και της πραγματικής, και να μεγιστοποιήσει την ομοιότητα μεταξύ των δύο διανυσμάτων πολιτικής. Για να το πετύχει αυτό χρησιμοποιείται η μέθοδος βελτιστοποίησης SGD.

Η εκμάθηση διαρκεί 10 εποχές, κατά τις οποίες όλα τα παραδείγματα διέρχονται από το δίκτυο κάθε φορά. Στο τέλος της διαδικασίας, η νέα έκδοση του μοντέλου παίζει μια σειρά παιχνιδιών με την παλαιότερη έκδοσή του, για να αξιολογηθεί το αν η εκπαίδευση αυτή επέφερε τελικά βελτίωση στον τρόπο παιχνιδιού του ή όχι.

4.5 Αξιολόγηση Μοντέλου

Η τελευταία εκδοχή της μηχανής παίζει μια σειρά παιχνιδιών με την προγενέστερη έκδοση του εαυτού της. Αυτό γίνεται αρχικοποιώντας τους δύο παίκτες με δίκτυα που βρίσκονται σε διαφορετική φάση της εκπαίδευσης. Οι πολιτικές και προβλέψεις έκβασης παιχνιδιών που παράγουν τα δύο αυτά δίκτυα θα είναι προφανώς διαφορετικά μεταξύ τους σε κάποιο βαθμό. Τα μισά από τα παιχνίδια παίζονται από την πλευρά του άσπρου και τα υπόλοιπα από την πλευρά του μαύρου. Από τα παιχνίδια αυτά, αν οι νίκες του νεότερου μοντέλου είναι περισσότερες από τις ήττες (με ποσοστό μεγαλύτερο του 55%, μη μετρώντας τις ισοπαλίες), τότε το μοντέλο αυτό υιοθετείται ως καλύτερο και θα είναι το επόμενο σημείο αναφοράς για τις μεταγενέστερες επαναλήψεις της εκπαιδευτικής διαδικασίας.

Ο αριθμός των παιχνιδιών που χρησιμοποιήσαμε είναι 40. Η χρήση περισσότερων δεν βελτίωνε την διαδικασία της αξιολόγησης και αντ' αυτού έκανε τη διαδικασία της εκπαίδευσης ακόμα πιο αργή. Οι ισοπαλίες δεν λαμβάνονται υπ' όψιν στην καταμέτρηση της απόδοσης του μοντέλου, καθώς τα περισσότερα παιχνίδια στο σκάκι μεταξύ αντιπάλων παρόμοιας δυναμικότητας καταλήγουν σε ισοπαλία και θα ήταν αδύνατον οι νίκες του μοντέλου να υπερτερούν των ισοπαλιών σε βάθος χρόνου.

Χάρη στην διαδικασία της αξιολόγησης, είναι σίγουρο ότι το μοντέλο που επιλέγεται είναι πάντα το πιο βελτιωμένο, με αποκλειστικό κριτήριο τη νίκη περισσότερων παιχνιδιών. Η φάση αυτή είναι αναγκαία, καθώς το δίκτυο χειρίζεται καταστάσεις και όχι ολόκληρες παρτίδες. Λόγω της τυχαιότητας των κινήσεων στα αρχικά στάδια, είναι σημαντικό η εκμάθηση να μην έχει σαν στόχο την ομοιότητα στις κινήσεις αυτές, αλλά μόνο τη νίκη.

5. ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΕΙΡΑΜΑΤΩΝ

Χρησιμοποιήσαμε μια σειρά από πειράματα με σκοπό την εκπαίδευση του μοντέλου για να παράγει όσο το δυνατόν καλύτερες αποδόσεις σε μικρότερο χρονικό διάστημα. Το υλικό που είχαμε στη διάθεσή μας ήταν αρκετά περιοριστικό, αλλά ικανοποιητικό για να εξάγουμε σημαντικά συμπεράσματα. Συγκεκριμένα, τα πειράματα όλα πραγματοποιήθηκαν στο Google Colab με τα εξής:

- GPU: T4 ή P100
- RAM: 25GB

Δοκιμάστηκαν διάφορες τεχνικές κατά την εκπαίδευση και καταγράφηκαν τα διαφορετικά αποτελέσματα που παρήγαγε η κάθε μία. Σε κάθε πείραμα, οι υπερπαράμετροι ήταν:

- learning rate: 0.001
- dropout rate: 0.3
- batch size: 64

οι οποίες προέκυψαν μετά από σύντομο πειραματικό fine tuning, εξετάζοντας την ταχύτητα εκπαίδευσης και την απόδοση του μοντέλου σε 1-2 εποχές.

Τέλος, κατά τη φάση της εκπαίδευσης, αρχικά σε κάθε επανάληψη χρησιμοποιήσαμε 30 παιχνίδια self-play, 40 προσομοιώσεις Monte Carlo ανά κίνηση και 40 παιχνίδια αξιολόγησης μοντέλου.

Ο αριθμός των

5.1 Δισδιάστατη είσοδος

Η αρχική μας προσέγγιση διαφέρει από αυτήν που εξηγήθηκε στο προηγούμενο κεφάλαιο. Αρχικά επιχειρήσαμε να χρησιμοποιήσουμε ένα δισδιάστατο συνελικτικό δίκτυο, δίνοντας σαν είσοδο μόνο την εικόνα του ταμπλό σαν δισδιάστατο πίνακα.

Το πείραμα αυτό είχε σαν στόχο την παρατήρηση της διαδικασίας της εκπαίδευσης με τα απολύτως απαραίτητα σαν είσοδο. Η ταχύτητα των υπολογισμών των δισδιάστατων πινάκων σε σχέση με τους τρισδιάστατους ήταν ένα βασικό προτέρημα της μεθόδου αυτής. Το δίκτυο θα ήταν σε θέση να πραματοποιήσει τις εποχές εκπαίδευσης σε πολύ μικρότερο χρόνο με τη μέθοδο αυτή, παρακάμπτοντας το σχετικά αργό υλικό που είχαμε στη διάθεσή μας. Μία παρτίδα self-play είχε διάρκεια λιγότερο από 1 λεπτό με τη μέθοδο αυτή, κάτι που επιτάχυνε αρκετά την εκπαίδευση. Σκοπός μας ήταν να μελετήσουμε τον αλγόριθμό μας σε βάθος χρόνου χάρη σε αυτήν την επιτάχυνση, καθώς οι υπόλοιπες προσεγγίσεις μας θα χρειάζονταν πολλαπλάσιο χρόνο για παρόμοιο αριθμό εποχών εκπαίδευσης.

Το αποτέλεσμα της μεθόδου αυτής δεν ήταν θετικό. Ενώ οι εποχές εκπαίδευσης πράγματι χρειάζονταν πολύ μικρότερο χρόνο, οι πληροφορίες που τροφοδοτούνταν στο δίκτυο ήταν ανεπαρκείς. Μετά την πρώτη επανάληψη της συνολικής διαδικασίας της εκπαίδευσης, η νέα πολιτική δεν λειτουργούσε όπως περιμέναμε. Ενώ το μοντέλο είχε πράγματι

ανανεωθεί και ακολουθούσε κάποια στρατηγική στο παιχνίδι του, δεν είχε πληροφορίες για τους ειδικούς κανόνες ισοπαλίας στο σκάκι. Χάρη στην εκτίμηση του δικτύου, κάποιες καταστάσεις του ταμπλό θεωρούνταν προτιμότερες από άλλες και ο 'βελτιωμένος' παίκτης προσπαθούσε να φτάσει στις καταστάσεις αυτές. Αυτό όμως σήμαινε ότι οι ισοπαλίες λόγω τριπλής επανάληψης γίνονταν πιο συχνές. Μετά από μερικές ακόμα ολοκληρωμένες εποχές εκπαίδευσης, και οι δύο παίκτες έιχαν το πρόβλημα αυτό, με αποτέλεσμα τα παινχίδια να λήγουν σε ισοπαλία σε πολύ λίγες κινήσεις, μην αφήνοντας το δίκτυο να αποκτήσει νέες γνώσεις εξερευνώντας περισσότερες καταστάσεις.

5.2 Τρισδιάστατη είσοδος

Από το προηγούμενο πείραμα ήταν φανερό ότι χρειαζόταν να τροφοδοτούμε περισσότερες πληροφορίες στο δίκτυο. Έτσι, αλλάξαμε την αρχιτεκτονική του στη μορφή που αναφέρουμε παραπάνω και χρησιμοποιήσαμε την επιπλέον διάσταση για να συμπεριλάβουμε παραπάνω πληροφορίες για την κάθε κατάσταση στα δεδομένα εισόδου.

5.2.1 Βελτίωση απόδοσης

Η διαδικασία αυτή είχε ως αποτέλεσμα την μείωση της ταχύτητας εκπαίδευσης, χάρη στην αυξημένη υπολογιστική ανάγκη των τριών διαστάσεων του δικτύου. Επίσης ανέβηκε η ανάγκη για περισσότερη μνήμη, λόγω του αυξημένου μεγέθους δεδομένων. Με τις νέες αυτές πληροφορίες όμως με τις οποίες τροφοδοτούνταν το δίκτυο, οι νέες πολιτικές θα άλλαζαν για να μην λήγει το παιχνίδι με τον κανόνα των επαναλήψεων τόσο συχνά.

Πράγματι, η νέα διαδικασία της εκπαίδευσης ξεκίνησε με καλούς ρυθμούς. Μετά από τις περισσότερες επαναλήψεις εκπαίδευσης, το ανανεωμένο μοντέλο νικούσε την προηγούμενη καλύτερη έκδοση κατά τη φάση της αξιολόγησης. Από κάποιο σημείο και έπειτα, τα αποτελέσματα είχαν τη μορφή 5-10% νίκη, 90-95% ισοπαλία για το νέο μοντέλο, χωρίς να χάνει καθόλου από το παλιό. Η διαδικασία αυτή φαινόταν ότι συγκλίνει προς μια πολιτική που ευνοεί τη νίκη. Γρήγορα όμως δημιουργήθηκε νέο πρόβλημα.

5.2.2 Υπερεκπαίδευση

Η νέα εκδοχή του μοντέλου άρχιζε σιγά σιγά να νικάει όλο και λιγότερο. Σύντομα, τόσο στη διαδικασία των παιχνιδιών με τον εαυτό του όσο και στη φάση αξιολόγησης εναντίον προηγούμενων εκδόσεων, οι παρτίδες άρχισαν να καταλήγουν αποκλειστικά σε ισοπαλία. Το αποτέλεσμα αυτό δεν σταματούσε σε βάθος χρόνου και τα νέα μοντέλα συνέχιζαν να απορρίπτονται, αφού στην αξιολόγησή τους σημείωναν 100% ισοπαλίες.

Τα αποτελέσματα αυτά μας οδήγησαν στο συμπέρασμα ότι το μοντέλο υπερεκπαιδεύεται στην νέα του πολιτική. Αυτό σημαίνει ότι αλλάζει υπερβολικά ριζικά τον τρόπο παιχνιδιού του στα αρχικά στάδια, κάνοντας μεγάλες αλλαγές στην πολιτική και στη συνέχεια αδυνατεί να ξεφύγει από τις αλλαγές αυτές, αφού πλέον σε όλα τα παιχνίδια με τον εαυτό του ακολουθείται η ίδια στρατηγική. Αν για παράδειγμα στις αρχικές φάσεις της εκπαίδευσης, που είναι οι πιο κρίσιμες για τη διαμόρφωση της πολιτικής, τύχει να νικηθούν πολλά παιχνίδια που ξεκινούν με την συχνή κίνηση 1e4, το δίκτυο στα αργότερα στάδια αναπτύσσει πολιτική που βασίζεται σχεδόν ολοκληρωτικά στην κίνηση αυτή για να ξεκινάει το παιχνίδι και αγνοεί άλλες εξίσου διαδεδομένες αρχικές κινήσεις όπως 1d4 ή 1Nf3.

Σημειώνουμε ότι η υπερεκπαίδευση συμβαίνει στα αρχικά στάδια της εκπαίδευσης και γι αυτό θεωρείται ότι απορρίπτονται αποδοτικές κινήσεις. Είναι λογικό οι ισοπαλίες να αυξάνονται όσο το μοντέλο μαθαίνει να αποδίδει καλύτερα, αλλά είναι προφανές ότι με τόσο λίγη εκπαίδευση (το πολύ 5 εποχές) είναι αδύνατον να αυξάνει την επίδοσή του σε τέτοιο βαθμό που δεν μπορεί να βελτιωθεί άλλο.

5.3 Αλλαγές για την αποφυγή της υπερεκπαίδευσης

Έχοντας ως στόχο την τη μείωση της υπερεκπαίδευσης κατά την περίοδο της εκπαίδευσης αρχικά επιχειρήσαμε μια σειρά από δοκιμές. Αφού εξοικονομήσαμε επιπλέον χώρο στη μνήμη μέσω συγκεκριμένων τεχνικών βελτιστοποίησης, πειραματιστήκαμε με τον αριθμό των παιχνιδιών του προγράμματος με τον εαυτό του, των προσομοιώσεων Monte Carlo και, σε μικρότερο βαθμό, των παρτίδων αξιολόγησης. Η ιδέα μας ήταν ότι αυξάνοντας τον αριθμό των προσομοιώσεων και μεγαλώνοντας έτσι το δέντρο αναζήτησης, θα είχε ως αποτέλεσμα μια πιο ευρεία αναζήτηση στις αρχικές φάσεις και έτσι θα επιβραδυνόταν η σύγκλιση σε μια στρατηγική. Η αύξηση των παιχνιδιών έγινε με σκοπό να αναγνωριστεί πιο εύκολα η αύξηση αυτή στο εύρος των κινήσεων που εξετάζονταν, αν αυτή υπήρχε.

Πράγματι, αφού πειραματιστήκαμε με τις παραπάνω παραμέτρους, σημειώθηκε μια μικρή βελτίωση. Το μοντέλο είχε λίγο πιο αργή σύγκλιση σε συγκεκριμένη πολιτική. Αυτό όμως δεν ήταν αρκετό για να αποτρέψει την υπερεκπαίδευση και μετά από κάποιο σημείο, κάθε παρτίδα εκπαίδευσης ή αξιολόγησης κατέληγε σε ισοπαλία. Η αύξηση των προσομοιώσεων συντέλεσε επίσης στη ραγδαία επιβράδυνση της εκπαίδευσης, γεγονός που μας δυσκόλεψε στην διεξαγωγή ικανοποιητικού αριθμού πειραμάτων. Οι αριθμοί ανέβηκαν μέχρι και 80 προσομοιώσεις ανά κίνηση και 50 παιχνίδια του προγράμματος με τον εαυτό του. Τα παιχνίδια αξιολόγησης δεν βοήθησαν καθόλου και απλά επιβράδυναν τη διαδικασία, οπότε μετά από λίγες δοκιμές αφέθηκαν στον αρχικό αριθμό των 40.

Ήταν σαφές ότι το πρόγραμμα χρειαζόταν μια επιπλέον αλλαγή, πιο δραστική από μικρές αλλαγές σε τιμές των παραμέτρων του. Αυτό που χρειαζόμασταν ήταν μια μέθοδο που να επέτρεπε στο μοντέλο να ξεπερνούσε τους περιορισμούς της πολιτικής του και να να του παρείχε τη δυνατότητα να εξερευνήσει νέα μονοπάτια στο δέντρο αναζήτησης.

5.4 Προσθήκη τυχαιότητας μέσω θορύβου

Μια αρκετά διαδεδομένη μέθοδος που προσθέτει τυχαιότητα σε κάποιο σύστημα είναι η προσθήκη θορύβου στις αρχικές πιθανότητες. Προσθέτοντας μια μικρή δόση από θόρυβο στις πολιτικές που επιστρέφει το δίκτυο, αυξάνεται η πιθανότητα να ακολουθούν διαφορετικές κινήσεις κατά την εκπαίδευση του μοντέλου. Έτσι, κατά τη δημιουργία των κόμβων φύλλων στο δέντρο αναζήτησης Monte Carlo, προσθέσαμε θόρυβο φτιαγμένο από δειγματοληψία της κατανομής Dirichlet. Ο θόρυβος αυτός προστίθεται στις αρχικές πιθανότητες που παράγει το δίκτυο και αποθηκεύεται στους κόμβους του δέντρου. Τα νέα διανύσματα κανονικοποιούνται μετά την προσθήκη για να διατηρηθεί η βασική ιδιότητα ενός πιθανοτικού διανύσματος πολιτικής.

Η τεχνική αυτή βελτίωσε αρκετά το πρόγραμμα σχετικά με το πρόβλημα της υπερεκπαίδευσης. Οι αρχικές εποχές της εκπαίδευσης παρουσιάζουν μεγαλύτερη ποικιλία αποτελεσμάτων χάρη στο πιο ευρύ δέντρο αναζήτησης. Στη συνέχεια, το μοντέλο αρχίζει να σχηματίζει την πολιτική που θα χρησιμοποιήσει ως βάση για τις μετέπειτα εποχές εκπαί-

δευσης. Η σταθεροποίηση γίνεται ακόμα πιο αργά, χάρη στον θόρυβο. Κάποια στιγμή, η πλειονότητα των παρτίδων αρχίζει να καταλήγει ξανά σε ισοπαλίες. Αυτό δείχνει σημάδια υπερεκπαίδευσης. Χάρη όμως στη νέα μέθοδο, σε κάποια από τις μετέπειτα εποχές, το μοντέλο εξερευνά νέες κινήσεις και αργά ή γρήγορα καταφέρνει να ξεπεράσει την απόδοση του μέχρι τότε καλύτερου και να το αντικαταστήσει.

Από το γεγονός αυτό καταλαβαίνουμε ότι η εισαγωγή θορύβου στο σύστημα συντελεί στη συνέχεια της διαδικασίας εκπαίδευσης. Μετά από κάποιο σημείο, η εκπαίδευση επιβραδύνεται σημαντικά, όπως και είναι το φυσιολογικό για ένα μοντέλο που συνεχώς βελτιώνεται. Όμως δεν παρατηρήσαμε στασιμότητα στην εκπαίδευση αν και η ταχύτητά της μειώνεται σημαντικά. Με λίγο ακόμα πειραματισμό με τις υπερπαραμέτρους του δικτύου, επιτύχαμε την πιο αποδοτική έκδοση του προγράμματος. Σε βάθος χρόνου πιστεύουμε ότι το μοντέλο θα μπορέσει να επιτύχει υπεράνθρωπες επιδόσες, αυτό όμως απαιτεί χρόνο και υλικές υποδομές που κατά τη διάρκεια των πειραμάτων μας είναι απαγορευτικά.

5.5 Χρόνοι πειραμάτων

Κατά την πραγματοποίηση των διαφορετικών δοκιμών, η φάση της εκπαίδευσης επιταχυνόταν ή επιβραδυνόταν ανάλογα με τις αλλαγές στον αλγόριθμο. Συγκεκριμένα, μια ολοκληρωμένη επανάληψη της εκπαίδευσης, που περιλαμβάνει τις παρτίδες εκμάθησης, την εκπαίδευση και αναβάθμιση του δικτύου και τις παρτίδες αξιολόγησης κατά μέσο όρο διαρκούσε:

- Δισδιάστατη είσοδος: 2 ώρες
- Τρισδιάστατη είσοδος με 25 προσομοιώσεις Monte Carlo: 2,5 ώρες
- Τρισδιάστατη είσοδος με 40 προσομοιώσεις Monte Carlo: 3 ώρες
- Τρισδιάστατη είσοδος με 80 προσομοιώσεις Monte Carlo: 5,5 ώρες
- Τρισδιάστατη είσοδος με 80 προσομοιώσεις Monte Carlo και προσθήκη θορύβου: 6 ώρες

Οι χρόνοι αυτοί είναι προσεγγιστικοί και κάθε φορά που το μοντέλο πλησίαζε την υπερεκπαίδευση, μειώνονταν αρκετά, έως και κάτω από το μισό, αφού τα παιχνίδια μειώνονταν σε διάρκεια κατά μέσο όρο. Είναι προφανές ότι οι χρόνοι εκπαίδευσης αποτελούσαν εμπόδιο στην ομαλή διένεξη των πειραμάτων, αφού το περιβάλλον που είχαμε στη διάθεσή μας, το Google Colab, σταματούσε τη λειτουργία του προγράμματος το πολύ μέσα σε 24 ώρες.

6. ΤΕΛΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

6.1 Τελικά συμπεράσματα

Το θέμα της Πτυχιακής Εργασίας ήταν να δημιουργηθεί ένα μοντέλο που παίζει σκάκι με μηδενική προϋπάρχουσα γνώση, αποκλειστικά με τη χρήση της ενισχυτικής μάθησης. Ανώτατος στόχος ήταν να επιτχύχει το μοντέλο μας επιδόσεις συγκρίσιμες με αυτές των μοντέλων της DeepMind. Κατανοήσαμε κατά την εκπόνηση της εργασίας ότι κάτι τέτοιο θα ήταν μάλλον αδύνατο, δεδομένου των υλικοτεχνικών υποδομών που είχαμε στη διάθεσή μας, αλλά και τις ελλείψεις στη γνώση μας σε λεπτομέρειες βελτιστοποίησης της εκπαιδευτικής διαδικασίας. Ένας πιο ρεαλιστικός στόχος είναι η δημιουργία ενός αλγορίθμου εκμάθησης ο οποίος με επαρκή χρόνο θα μπορούσε να αγγίξει υπεράνθρωπη επίδοση.

Οι προσεγγίσεις μας έδιναν προτεραιότητα στην ύπαρξη μιας συνεχόμενης πορείας κατά την εκπαίδευση και όχι στα ραγδαία αρχικά άλματα στην απόδοση, πιστεύοντας ότι σε βάθος χρόνου αυτό θα οδηγούσε σε ευνοϊκότερα αποτελέσματα. Καταλήξαμε στο ότι μια πιο αργή διαδικασία πράγματι ωφελεί το μοντέλο στη συνολική του πορεία και αν μας το επέτρεπαν τα χρονικά περιθώρια, το αυτό μπορεί να ανταπεξέλθει σε αρκετά υψηλό ανταγωνισμό, αν και ίσως όχι υπεράνθρωπο.

Η εργασία αυτή μας δίδαξε την αποτελεσματικότητα της ενισχυτικής μάθησης και μας βοήθησε να κατανοήσουμε καλύτερα αυτόν τον κλάδο της τεχνητής νοημοσύνης. Εμβαθύναμε επίσης στα νευρωνικά δίκτυα και τις αναζητήσεις Monte Carlo, που είναι και από τις πιο προηγμένες μεθόδους αναζήτησης βασισμένες στην τυχαιοκρατική επιλογή. Η ιδέα ότι μπορεί να μελετηθεί επαρκώς μια διαδικασία με χρήση επαρκών προσομοιώσεων στις διάφορες εκβάσεις της ανοίγει νέους δρόμους σε πολλούς επιστημονικούς τομείς. Με τον συνδυασμό των μεθόδων αυτών, είναι πράγματι εφικτό να φτάσει ένα μοντέλο υπεράνθρωπη επίδοση, δίχως επίβλεψη από τον άνθρωπο κατά τη διάρκεια των επιλογών του.

6.2 Προτάσεις βελτίωσης και επέκτασης του μοντέλου

Κατά τη εκπόνηση της Πτυχιακής Εργασίας εκτελέστηκαν μια σειρά από πειράματα, το κάθε ένα από τα οποία οδήγησε σε προσαρμογές με σκοπό τη βελτίωση του αλγορίθμου. Στο μέλλον, σκοπεύουμε να συνεχίσουμε τις δοκιμές και να πειραματιστούμε με διαφορετικές πτυχές του αλγορίθμου για να επιτύχουμε ακόμα καλύτερα αποτελέσματα:

- Το μοντέλο μας θα μπορούσε να επιτύχει πολύ μεγαλύτερες αποδόσεις αν είχαμε πρόσβαση σε ταχύτερο υλικό που υποστηρίζει παραλληλία υπολογισμών. Με καλύτερες GPU θα μπορούσαμε να χρησιμοποιήσουμε μεγαλύτερους αριθμούς προσομοιώσεων Monte Carlo και παιχνιδών self-play. Αυτό θα συνέβαλε στην εκπαίδευση του μοντέλου και στην πιο ορθή λειτουργία του αλγορίθμου. Επίσης, με πάνω από 1 GPU στη διάθεσή μας θα μπορούσαμε να παραλληλοποιήσουμε τις διαδικασίες των προσομοιώσεων και τον παιχνιδιών εκπαίδευσης, αφήνοντάς τες να λειτουργούν ταυτόχρονα και έτσι θα μειωθεί δραστικά ο χρόνος που χρειάζεται για μια συνολική εποχή εκπαίδευσης, παρ' όλο το επιπρόσθετο φορτίο λόγω περισσότερων παιχνιδιών. Με τις αλλαγές αυτές θα μπορούν να γίνουν περισσότερες δοκιμές και να εξεταστούν πειραματικά διάφορες βελτιστοποιήσεις, οι οποίες κατά τη συγγραφή της εργασίας είναι απαγορευτικές λόγω των χρονικών περιορισμών που έχουν τεθεί.

- Η μοντελοποίηση του παιχνιδιού επιδέχεται επίσης πλήθος βελτιώσεων. Σκοπός μας για το μέλλον είναι να πειραματιστούμε με διαφορετικούς τρόπους απεικόνισης του ταμπλό στο νευρωνικό δίκτυο. Υπάρχουν περισσότερα δεδομένα που μπορούμε να εισάγουμε στην αρχιτεκτόνική μας όπως ιστορικό κινήσεων και μερικές ακόμα μετρικές, οι οποίες θα έδιναν παραπάνω πληροφορίες στο μοντέλο για την κάθε κατάσταση και θα του επέτρεπαν πιθανώς να το αξιολογήσει καλύτερα. Επίσης, χάρη στην αρχιτεκτονική του προγράμματος, θα μπορούσε εύκολα να χρησιμοποιηθεί ένα διαφορετικό παιχνίδι στο μέλλον. Η μοντελοποίησή του με τις κατάλληλες συναρτήσεις που χρησιμοποιούνται στο σκάκι θα επέτρεπαν στο δίκτυο να εκπαιδευτεί και για άλλα παιχνίδια, αξιολογώντας έτσι την γενικότερη ισχύ του αλγορίθμου.
- Στην εργασία αυτή πειραματιστήκαμε με διάφορες μορφές συνελικτικού δικτύου. Ο λόγος για την επιλογή αυτή ήταν η καταλληλότητα των δικτύων αυτών σε προβλήματα αναγνώρισης εικόνας σε 2 ή 3 διαστάσεις. Οι μοντελοποιήσεις του ταμπλό που χρησιμοποιήσαμε είχαν μορφή που εμφάνιζε αρκετά κοινά χαρακτηριστικά με εικόνες, οπότε αποφασίσαμε να επικεντρωθούμε σε τέτοια δίκτυα. Παρ' όλα αυτά θα είχε ενδιαφέρον η χρήση διαφορετικών ειδών νευρωνικών δικτύων και η καταγραφή της διαφορετικής επίδοσης σε κάθε ένα από αυτά. Μοντέλα που χρησιμοποιούν επαναλαμβανόμενα δίκτυα, ή συνδυασμούς τους με συνελικτικά θα μπορούσαν να ξεπεράσουν ίσως τις επιδόσεις του παρόντος δικτύου.

ΣΥΝΤΟΜΕΥΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ANNs	Artificial Neural Networks
TLU	Threshold Logic Unit
CNNs	Convolutional Neural Networks
TDL	Temporal Difference Learning
MCTS	Monte Carlo Tree Search
SGA	Stochastic Gradient Ascent
SGD	Stochastic Gradient Descent
MSE	Mean Squared Error
CBMPI	Classification-Based Modified Policy Iteration
NFSP	Neural Fictitious Self-Play
FSP	Fictitious Self-Play
XFP	Full-Width Extensive-Form Fictitious Play
M_{RL}	Reinforcement Learning Memory
M_{SL}	Supervised Learning Memory

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Hervé Abdi and Lynne J. Williams. Principal component analysis. volume 2, pages 433--459, 2010.

[2] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272, 2012.

[3] Horace B Barlow. Unsupervised learning. volume 1, pages 295--311. MIT Press, 1989.

[4] Andrew G Barto and Michael Duff. Monte carlo matrix inversion and reinforcement learning. *Advances in Neural Information Processing Systems*, pages 687--687, 1994.

[5] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Gilbert Lechevallier, Yvesand Saporta, editor, *Proceedings of COMPSTAT'2010*, pages 177--186, Heidelberg, 2010. Physica-Verlag HD.

[6] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421--436. Springer, 2012.

[7] Guillaume Maurice Jean-Bernard Chaslot Chaslot. *Monte-carlo tree search*. Maastricht University, 2010.

[8] Pierre Comon. Independent Component Analysis. In J-L.Lacoume, editor, *Higher-Order Statistics*, pages 29--38. Elsevier, June 1992.

[9] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In Shun-ichi Amari and Michael A. Arbib, editors, *Competition and Cooperation in Neural Nets*, pages 267--285, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[10] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of supervised learning. In *The elements of statistical learning*, pages 9--41. Springer, 2009.

[12] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65--93. Elsevier, 1992.

[13] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[14] Ronald A Howard. Dynamic programming and markov processes. 1960.

[15] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[16] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. volume 36, pages 451--461, 2003. Biometrics.

[17] Michael L Littman. Value-function reinforcement learning in markov games. *Cognitive systems research*, 2(1):55--66, 2001.

[18] Stephen Maren. Neurobiology of pavlovian fear conditioning. *Annual review of neuroscience*, 24(1):897--931, 2001.

[19] Larry R Medsker and LC Jain. Recurrent neural networks. pages 11--20.

[20] Ralph R Miller, Robert C Barnet, and Nicholas J Grahame. Assessment of the rescorla-wagner model. *Psychological bulletin*, 117(3):363, 1995.

[21] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[22] Hervé Moulin and J-P Vial. Strategically zero-sum games: the class of games whose completely mixed equilibria cannot be improved upon. *International Journal of Game Theory*, 7(3-4):201--221, 1978.

[23] Surag Nair. Simple, generic AlphaGo Zero, 12 2017.

[24] C.C. Paige. The general linear model and the generalized singular value decomposition. volume 70, pages 269--284, 1985.

[25] S. Patel, S. Sihmar, and A. Jatain. A study of hierarchical clustering algorithms. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 537--541, 2015.

[26] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[27] Martin L. Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. volume 24, pages 1127--1137, 1978.

[28] Robert W Rosenthal. Games of perfect information, predatory pricing and the chain-store paradox. *Journal of Economic theory*, 25(1):92--100, 1981.

[29] Abdallah Saffidine, Hilmar Finnsson, and Michael Buro. Alpha-beta pruning for games with simultaneous moves. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[30] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85--117, 2015.

[31] Subana Shanmuganathan. *Artificial Neural Network Modelling: An Introduction*, pages 1--14. Springer International Publishing, Cham, 2016.

[32] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484--489, 2016.

[33] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354--359, 2017.

[34] Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1):123--158, 1996.

[35] Glenn Strong. The minimax algorithm. *Trinity College Dublin*, 2011.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*.

[37] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[38] Gerald Tesauro. Practical issues in temporal difference learning. *Machine learning*, 8(3):257--277, 1992.

[39] C. Van Der Malsburg. Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. In Günther Palm and Ad Aertsen, editors, *Brain Theory*, pages 245--248, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[40] Lipo Wang. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.