

الجمهورية العربية السورية
المعهد العالي للعلوم التطبيقية والتكنولوجيا
قسم الاتصالات
العام الدراسي 2022/2023

مشروع تخرج

أعد لنيل درجة الإجازة في هندسة المعلومات – هندسة البرمجيات والذكاء الصناعي



بناء نظام ذكي للعب الشطرنج

تقديم

أحمد ملهم قطان

إشراف

د. رياض سنبل

م. سيما أدهم

8/16/2023

الإهداء

إلى سر وجودي في هذا الكون، إلى من عطر لي عبق الحياة، إلى من أفتخر بكوني قطعة منها

أمي الحبيبة

إلى الحارس الذي أطاب لي غمض العين، إلى المنارة التي تضيء دربي

أبي الغالي

إلى عدتي في شدي، إلى الذين كانوا سنداً اتكأت عليهم في لحظات ضعفي، إلى من كان نجاحي بسببهم ولأجلهم

إخوتي الأعزاء

إلى من كان ملجأ دوماً، إلى أخوتي الذين لم تلدهم أمي

حسن رشيد وإلاء ميا

إلى خليفتي الذي أدعو الله أن يكلل دربه بالنجاح (الذي لا تحلو السهرة إلا بوجوده معنا)

محمد صالح التركي

إلى من كانوا بجواري في السراء والضراء، إلى الذين عشت معهم أجمل سنين عمري

علي خضر وعبد الله الشيخ حسين ومحمود حداد ومجد إبراهيم

إلى الذين لا يكلّون من تشجيعي على المثابرة والاستمرار

غفار حيدر ويزن حمود وأحمد علي وخضر خليل ومناف زلف

إلى كل من وقف بجاني يوماً

إلى جميع أصدقائي وزملائي الذين شاركوني مقاعد الدراسة منذ بداية طريق العلم

كلمة شكر

أتقدم بالشكر الجزيل للدكتور رياض سنبل على إشرافه المتميز ومتابعته لكل خطوة من تفاصيل هذا العمل من أجل نجاح هذا المشروع، وتقديمه لي الكثير من خبراته العلمية والعملية.

كما أتوجه بجزيل الشكر إلى م. سيما الأدهم الذي لم يكن لمشروعي هذا أن يرى النور لولا متابعته الدائمة لسير العمل، ومساعدتي في الكثير من الأمور التخصصية الصعبة.

لا أنسى أبداً الدكتور مصطفى الدقاق الذي أعطاني خبرة كبيرة العام الماضي في تنظيم أمور العمل بمشروعي العلمي، إضافةً إلى جهوده المبذولة في متابعتنا ورغبته في إتمام مشاريعنا على أكمل وجه.

الخلاصة

الشطرنج هو لعبة استراتيجية للاعبين تُلعب على لوح شطرنج، وهو لوح للعبة يحتوي على 64 مربعاً مرتبة في شبكة 8×8 . لقد قامت التطورات التكنولوجية الحالية بتغيير طريقة التي نلعب بها الشطرنج، حيث أصبحنا نلعب الشطرنج وندرس تقنياتها ونقوم بتحليل اللعبة. التنبؤ بالخطوة التالية في لعبة الشطرنج ليس مهمة سهلة، حيث أن معدل الفروع في الشطرنج وسطياً هو 35. وبالتالي، يجب البحث عن خوارزميات جديدة لإنشاء محرك لعبة مثالي. في هذا البحث، تمت محاولة دمج الشبكات العصبية التلافيفية (CNN) وخوارزمية بحث شجرة Alpha-Beta لإنشاء ذكاء اصطناعي هجين ماهر في لعب الشطرنج. يوفر جوهر الشطرنج، المليء بالتفكير الاستراتيجي واتخاذ القرارات المعقدة، اختباراً مثالياً لدمج التقنيات المتطورة كالشبكات العصبونية مع مبادئ نظرية اللعبة الكلاسيكية كشجرة البحث. يتضمن هذا النهج مجموعات بيانات منسقة ومفلترة بدقة تخضع لعملية تحويل لإنشاء مصفوفات لتكون دخلاً مناسباً للشبكات العصبونية. تم إنشاء سبع شبكات CNN يمكن تقسيمهم إلى مجموعتين، الأولى تهدف إلى تحديد القطعة المراد تحريكها وتحتوي على شبكة وحيدة، والمجموعة الثانية تهدف لتحديد الخلية المراد تحريك القطعة إليها وتضم 6 شبكات كل منها مسؤول عن نوع محدد من القطع. بعد التدريب، يتم استخدام دمج مخرجات الشبكة لتحقيق أكثر الحركات الواعدة. يؤكد هذا البحث على القوة من دمج كل من التعلم العميق والتفكير الاستراتيجي في صياغة الذكاء الاصطناعي المتفوق في الشطرنج، تم التدريب على مجموعتين من البيانات، الأولى ذات الـ 20 ألف لعبة والثانية تحتوي على 140 ألف لغز شطرنجي، وعند تجريب النظام ضد Stockfish 1000-1500 Elo نجد أنه فاز بـ 7 ألعاب وخسر 3 وتعادل 2 من أصل 12 لعبة ونقدر الـ Elo للنموذج الهجين بـ 1700 Elo.

Abstract

Chess is a strategy game played by two players on a chessboard, which is a game board with 64 squares arranged in an 8x8 grid. Recent technological developments have changed how we play and study chess. Predicting the next move in chess is not an easy task, as, on average, there are 35 possible moves to consider. This makes it important to find new methods to create a great computer chess player. In this research, we tried combining Convolutional Neural Networks (CNNs) with the Alpha-Beta tree search algorithm to make a clever hybrid artificial intelligence for playing chess. Chess is a perfect game to use these two things together because it needs both smart thinking and understanding patterns. We do this by picking and changing groups of data that the computer can learn from. We make seven groups of computer programs to help us choose the best moves. Some of these groups help us choose which piece to move, and others help us decide where to move that piece. After training these programs, we put together what they say to find the best moves. This research shows that combining these two methods makes a better chess computer. We trained our programs using two sets of examples: one with 20,000 chess games and another with 140,000 chess puzzles. When we tested Our chess program against Stockfish, a strong chess engine, it won 7 games, lost 3, and tied 2 out of 12 games. We estimate that our hybrid model has a skill level (Elo) of around 1700. In conclusion, putting together Convolutional Neural Networks and the Alpha-Beta tree search makes a strong chess computer. This research doesn't just help chess; it also shows how different ways of thinking can work together to make computers smarter in many areas.

المحتويات

1- الفصل الأول	1
الإطار العام للمشروع	1
1.1- مقدمة	2
2.1- هدف المشروع	1
3.1- متطلبات المشروع	2
1.3.1- المتطلبات الوظيفية	2
2.3.1- المتطلبات غير الوظيفية	2
4.1- الخطة الزمنية	2
2- الفصل الثاني	3
الدراسة النظرية والمرجعية	3
1.2- الشطرنج (Chess)	4
1.1.2- تجهيز الرقعة	4
2.1.2- حركة القطع	5
3.1.2- الكش والكش مات والموت خنقاً	7
2.2- محركات الشطرنج (Chess Engines)	8
3.2- محركات الشطرنج التقليدية	9
1.3.2- خوارزميات تقليل الحد الأعظمي لمقدار الخسارة Minimax	9
2.3.2- تابع التقييم (Evaluation Function)	10
3.3.2- تقليم ألفا-بيتا (Alpha-Beta Pruning)	11
4.3.2- الأعمال السابقة	13
4.2- الشبكات العصبونية التلافيفية (Convolutional Neural Networks)	14
1.4.2- بنية الشبكات التلافيفية	14
1.1.4.2- الطبقات التلافيفية (Conv Layers)	14
2.1.4.2- طبقات الانتخاب (Pooling Layers)	15
3.1.4.2- طبقات الإهمال (Dropout)	15
4.1.4.2- توابع التنشيط (Activation Function)	16
5.1.4.2- الطبقات المتصلة كلياً Fully connected layers	18
6.1.4.2- توابع التقييم	19
7.1.4.2- توابع الأمثلة أو التحسين (Optimizers)	22
2.4.2- التنبؤ بالحركة	22
3.4.2- تمثيل المدخلات	23

23	4.4.2- هندسة الشبكات
23	5.4.2- الإخراج والتدريب
24	6.4.2- مزايا التنبؤ بالحركة CNNs
24	7.4.2- القيود والاعتبارات
24	8.4.2- الأعمال السابقة
25	5.2- التعلم المعزز العميق (Deep Reinforcement Learning)
25	1.5.2- عميل الشطرنج في DRL
25	2.5.2- آلية اللعب الذاتي:
26	3.5.2- نجاحات محركات الشطرنج القائمة على DRL (AlphaZero)
26	4.5.2- القيود والاعتبارات
26	5.5.2- الأعمال السابقة
27	6.2- شجرة بحث مونت كارلو (Monte Carlo Tree Search)
28	1.6.2- الاختيار (Selection)
28	2.6.2- التوسع (Expansion)
28	3.6.2- المحاكاة (Simulation)
28	4.6.2- الانتشار العكسي (Backpropagation)
29	5.6.2- الأعمال السابقة
30	7.2- شبكات الذاكرة طويلة المدى (LSTM)
30	1.7.2- الأعمال السابقة
32	8.2- المحركات Leela Chess Zero, AlphaZero, Stockfish
32	1.8.2- Stockfish
33	2.8.2- AlphaZero
33	3.8.2- Leela Chess Zero (Lc0)
34	9.2- مجموعة البيانات (Datasets)
34	1.9.2- مجموعة بيانات ألعاب الشطرنج
34	1.1.9.2- محتوى مجموعة البيانات
36	2.9.2- مجموعة بيانات ألغاز الشطرنج
36	1.2.9.2- محتوى مجموعة البيانات:

3- الفصل الثالث 37

الحل المقترح 37

37	1.3- مقدمة
38	2.3- الخط الرئيسي لنهج العمل
39	3.3- فلتر البيانات
39	1.3.3- فلتر بيانات الألعاب
40	2.3.3- فلتر بيانات الألغاز
40	4.3- تجهيز ومعالجة البيانات

41	5.3- هيكلية النموذج
42	1.5.3- هيكلية الشبكات
44	2.5.3- تدريب الشبكات
51	3.5.3- إعادة تدريب الشبكات (Fine Tune)
59	6.3- شجرة البحث
59	1.6.3- بناء الشجرة
59	2.6.3- تابع التقييم
60	3.6.3- تحسينات شجرة البحث
62	4- الفصل الرابع
62	الاختبارات والنتائج
62	1.4- اختبار النظام عن طريق تحليل الألعاب الفردية
62	1.1.4- النموذج المدرب فقط
63	2.1.4- النموذج المعاد تدريبه من دون تجميد طبقات التلافيفية
63	3.1.4- النموذج المعاد تدريبه مع تجميد الطبقات التلافيفية
63	4.1.4- النموذج المعاد تدريبه من دون تجميد مع شجرة البحث
67	5- الفصل الخامس
67	التنجز العملي
68	1.5- الهيكل العام للمشروع
68	2.5- بيانات العمل المستخدمة
68	1.2.5- بيئة Google.Colab
71	2.2.5- بيئة Jupyter Notebook
71	3.2.5- بيئة Visual Studio Code
72	3.5- المكتبات المستخدمة
72	1.1.3.5- مكتبة chess and chess.pgn
72	2.3.5- مكتبة Tensorflow
73	3.3.5- مكتبة Flask
73	4.3.5- مكتبة subprocess
73	5.3.5- مكتبات NumPy, Panda, Matplotlib, Seaborn
74	6.3.5- مكتبة Chessboard.js
74	7.3.5- مكتبة Chess.js
74	8.3.5- تابع التقييم لـ Stockfish
75	4.5- بناء واجهة برمجة التطبيقات API للتفاعل مع النظام
76	5.5- بناء واجهات المستخدم للتفاعل واللعب مع النظام
77	6- الفصل السادس

77	الخاتمة
77	1.6- الصعوبات
77	2.6- الافاق المستقبلية
78	3.6- الخاتمة

قائمة الأشكال

- شكل 1: شكل الرقعة الابتدائي 4
- شكل 2: حركة البيادق وطريقة الأسر بها 5
- شكل 3: الأسر بالتجاوز En Passant 6
- شكل 4: حركة كل من الرخ، الأسقف، الفارس، والمملكة 6
- شكل 5: مثال على شجرة البحث minimax 10
- شكل 6: مثال عن تقليم ألفا بيتا في شجرة ال minimax 13
- شكل 7: هيكل الشبكة العصبونية 14
- شكل 8: طريقة عمل طبقات انتخاب أعظمي 15
- شكل 9: منحنى ومعادلة تابع Sigmoid 16
- شكل 10: منحنى تابع ReLU 17
- شكل 11: منحنى تابع Leaky ReLU 18
- شكل 12: المسافة بين القيمة الحقيقية T والقيمة المتوقعة S 20
- شكل 13 : مصفوفة الارتباك Confusion Matrix 21
- شكل 14 : المساحة تحت المنحنى AUC 22
- شكل 15: تحليل موضع ما لرقعة شطرنج لقنوات كدخل للشبكة العصبونية 23
- شكل 16: نموذج التعلم العميق الأساسي 26
- شكل 17: خطوات تطبيق خوارزمية MCTS 29
- شكل 18: Leela Chess Zero VS AlphaZero VS Stockfish 34
- شكل 19: مثال عن شكل بيانات لعبة مرمزة في ملف PGN 35
- شكل 20: لعبة كاملة مرمزة بترميز SAN 36
- شكل 21: نفس اللعبة في الشكل (20) السابق ولكن بترميز UCI 36
- شكل 22: مثال عن FEN 36
- شكل 23: الخطة الرئيسية للحل 39
- شكل 24: تحويل الرقعة إلى مصفوفات 41
- شكل 25: هيكلية الشبكات العصبونية التلافيفية CNNs architecture 44
- شكل 26: نتيجة تدريب الشبكة الأولى منتقي القطعة Piece Selector 45

- شكل 27: توزيع الدقة على كافة القطع باستخدام الشبكة الأولى المدربة باستخدام المحسن Adam 45
- شكل 28: تغير الدقة بدلالة ال K للشبكة الأولى المدربة باستخدام المحسن Adam 46
- شكل 29: نتيجة تدريب الشبكة الأولى منتقي القطعة Piece Selector باستخدام المحسن RMSProp 47
- شكل 30: تغير الدقة بدلالة k وذلك للشبكة الأولى المدربة باستخدام المحسن RMSProp 47
- شكل 31: نتيجة تدريب شبكة تحريك البيدق 48
- شكل 32: نتيجة تدريب شبكة تحريك الملكة 48
- شكل 33: نتيجة تدريب شبكة تحريك الأسقف 48
- شكل 34: نتيجة تدريب شبكة تحريك الفارس 49
- شكل 35: نتيجة تدريب شبكة محرك الرخ 49
- شكل 36: نتيجة تدريب شبكة محرك الملك 49
- شكل 37: تغير الدقة بتغير K للنموذج صاحب الشبكة الأولى المدربة باستخدام Adam 50
- شكل 38: تغير الدقة بتغير K للنموذج صاحب الشبكة الأولى المدربة باستخدام RMSProp 50
- شكل 39: نتيجة إعادة تدريب الشبكة الأولى منتقي القطعة Piece Selector المدربة سابقا باستخدام المحسن Adam 52
- شكل 40: نتيجة إعادة تدريب الشبكة الأولى المدربة سابقا باستخدام المحسن RMSProp باستخدام المحسن Adam 52
- شكل 41: توزيع الدقة على كافة القطع بالشبكة الأولى المدربة باستخدام المحسن Adam والمعاد تدريبها باستخدام نفس المحسن مع وبلا تجميد 53
- شكل 42: توزيع الدقة على كافة القطع بالشبكة الأولى المدربة باستخدام المحسن RMSProp والمعاد تدريبها بالمحسن Adam وبدون تجميد 53
- شكل 43: تغير الدقة بدلالة k للشبكة المدربة باستخدام المحسن Adam والمعاد تدريبها باستخدام المحسن Adam 54
- شكل 44: تغير الدقة بدلالة k للشبكة المدربة باستخدام المحسن Adam والمعاد تدريبها باستخدام المحسن Adam 54
- شكل 45: نتيجة إعادة تدريب شبكة محرك البيدق 55
- شكل 46: نتيجة إعادة تدريب شبكة محرك الملكة 55
- شكل 47: نتيجة إعادة تدريب شبكة محرك الأسقف 56
- شكل 48: نتيجة إعادة تدريب شبكة محرك الفارس 56
- شكل 49: نتيجة إعادة تدريب شبكة محرك الرخ 57
- شكل 50: نتيجة إعادة تدريب شبكة محرك الملك 57
- شكل 51: تغير الدقة بتغير K مع وبدون تجميد للنموذج صاحب الشبكة الأولى المدربة باستخدام Adam 58
- شكل 52: تغير الدقة بتغير K مع وبدون تجميد للنموذج صاحب الشبكة الأولى المدربة باستخدام RMSProp 58

- شكل 53: تحليل اللعبة ضد النموذج المدرب فقط وهو يلعب باللون الأسود..... 62
- شكل 54: تحليل اللعبة ضد النموذج المعاد تدريبه على الألغاز ومن دون تحميد الطبقات التلافيفية وهو يلعب باللون الأبيض..... 63
- شكل 55: تحليل اللعبة ضد النموذج المعاد تدريبه على الألغاز مع تحميد الطبقات التلافيفية باللون الأبيض..... 63
- شكل 56: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأسود..... 64
- شكل 57: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأبيض..... 64
- شكل 58: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأسود..... 65
- شكل 59: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأبيض..... 65
- شكل 60: نتيجة تحليل اللعبة بين Stockfish 1500 Elo بالأسود والنموذج مع الشجرة الديناميكية بالأبيض..... 66
- شكل 61: الهيكل العام للمشروع..... 68
- شكل 62: شكل واجهات بيئة عمل Google.Colab..... 70
- شكل 63: مثال عن استخدام الـ API باستخدام Postman..... 75
- شكل 64 : شكل واجهة المستخدم..... 76

الاختصارات

CNN	Convolutional Neural Network	الشبكة العصبونية التلافيفية
ReLU	Rectified Linear Unit	الوحدة الخطية المصححة
AUC	Area Under the Curve	المساحة تحت المنحني
RMSProp	Root Mean Square Propagation	جذر متوسط التربيع للانتشار
Adam	Adaptive Moment Estimation	تقدير اللحظة التكيفية
DRL	Deep Reinforcement Learning	التعلم المعزز العميق
MCTS	Monte Carlo Tree Search	شجرة بحث مونت كارلو
UCB	Upper Confidence Bound	الحد الأعلى للثقة
UCT	Upper Confidence bound applied to Trees	الحد الأعلى للثقة المطبق على الأشجار
MCC	Monte Carlo Chess	تطبيق مونت كارلو في الشطرنج
LSTM	Long Short-Term Memory	شبكات الذاكرة طويلة المدى
TCEC	Top Chess Engine Championship	البطولة العليا لمحركات الشطرنج
CCRL	Computer Chess Rating Lists	قوائم تصنيف الحواسيب الشطرنجية
NNUE	Efficiently Updatable Neural Network	الشبكة العصبية القابلة للتحديث
GPUs	Graphics Processing Units	وحدات المعالجة الرسومية
TPUs	Tensor Processing Units	وحدات المعالجة التنسورية
Lc0	Leela Chess Zero	محرك شطرنج ليلا صفر
PGN	Portable Game Notation	تدوين اللعبة المحمولة
CSV	Comma-Separated Values	قيم مفصولة بفواصل
SAN	Simple Algebraic Notation	تدوين جبري بسيط
UCI	Universal Chess Interface	واجهة الشطرنج العالمية
FEN	Forsyth-Edwards Notation	تدوين فورسيث إدواردز
API	Application Program Interface	واجهة برمجة التطبيقات
IDE	Integrated development environment	بيئة تطوير متكاملة

HTML	Hypertext Markup Language	لغة ترميز النصوص التشعبية
CSS	Cascading Style Sheets	اوراق النمط المتعاقب
SGD	Stochastic Gradient Descent	الانحدار العشوائي
CP	Centi Pawn	واحدة: جزء من المئة من البيدق
JSON	Java Script Object Notation	تدوين كائنات جافا سكريبت

الرموز

ELO:

1-الفصل الأول

الإطار العام للمشروع

نبيّن في هذا الفصل مقدمة عن المشروع والتعريف به وبأهدافه، متطلبات المشروع الوظيفية والغير الوظيفية، الخطة الزمنية، مع توضيح الهيكل العام للمشروع.

1.1- مقدمة

لقد استحوذ الشطرنج، وهو رمز خالد للتفكير الاستراتيجي والبراعة الفكرية، على مدى قرون على خيال اللاعبين والمفكرين. إلى جانب جاذبيتها التقليدية، كان الشطرنج بمثابة معيار دائم لتقدم الذكاء الاصطناعي. والجدير بالذكر أن اللحظات المحورية مثل المباراة الأسطورية بين Deep Blue من IBM وسيد الشطرنج الكبير Grand Master Garry Kasparov أكدت الخطوات الكبيرة التي تم إحرازها في قدرة الذكاء الاصطناعي على فك رموز تعقيدات هذه اللعبة المعقدة.

التحديات التي يمثلها الشطرنج متعددة الأوجه وهائلة. مع وجود عدد هائل من المواقف المحتملة والتفاعل المعقد للأنماط، تتطلب اللعبة ليس فقط فهما عميقا للعمق الاستراتيجي ولكن أيضا القدرة على التكيف مع سيناريوهات لا تعد ولا تحصى. كشفت المحاولات التاريخية لبناء أنظمة لعب الشطرنج، بدءا من المحركات القائمة على القواعد إلى خوارزمية minimax الكلاسيكية، عن قيود في التقاط الأنماط الدقيقة المتأصلة في ديناميكيات اللعبة.

بث ظهور الشبكات العصبية حياة جديدة في البحث عن أنظمة لعب الشطرنج الذكية. قدمت الشبكات العصبية التلافيفية (CNNs)، المشهورة بكفاءتها في التعرف على الأنماط، وسيلة محيرة لمعالجة تعقيدات لعبة الشطرنج. ومع ذلك، على الرغم من قدراتها، واجهت شبكات CNN وحدها تحديا بسبب الحاجة إلى التبصر الاستراتيجي والتطلع إلى الأمام اللذين يحددان طريقة لعب الشطرنج.

يتبنى هذا المشروع نهجا مبتكرا يسد الفجوة بين التعلم العميق ونظرية الألعاب الكلاسيكية. من خلال دمج الشبكات العصبية التلافيفية مع خوارزمية البحث الشجري Alpha-Beta، يسعى هذا البحث لريادة نموذج يتجاوز قيود الأساليب السابقة. يستفيد هذا الاندماج من براعة شبكات CNN في استخراج الأنماط المكانية المعقدة مع تسخير الرؤية الاستراتيجية لـ Alpha-Beta للقيام بعملية صنع القرار.

تنتقل الرحلة القادمة عبر مجموعات البيانات المنسقة بدقة، وإنشاء شبكات عصبية متخصصة، وتنسيق بحث شجرة Alpha-Beta. من خلال هذا التزاوج المتناغم بين تقنيات التعلم العميق المعاصرة ونظرية اللعبة الكلاسيكية، لا يهدف البحث إلى رفع قدرات الذكاء الاصطناعي في الشطرنج فحسب، بل يهدف أيضا إلى المساهمة في المشهد الأوسع لاستراتيجية الذكاء الاصطناعي.

بينما نبدأ في هذا الاستكشاف لدمج الشبكات العصبية مع البصيرة الاستراتيجية الكلاسيكية، نتعمق في عالم تصبح فيه رقعة الشطرنج لوحة للابتكار، ويرتبط التعرف على الأنماط مع الفطنة الاستراتيجية، وإمكانيات الأنظمة الذكية لإتقان تعقيدات صنع القرار الاستراتيجي تنمو أكثر عمقا من أي وقت مضى.

الشطرنج هي لعبة تتطلب الكثير من الإبداع والتفكير المتطور لدرجة أنه كان يُنظر إليها ذات مرة على أنها شيء لن تستطيع أجهزة الكمبيوتر القيام به على الإطلاق. تم إدراجها في كثير من الأحيان إلى جانب أنشطة مثل كتابة الشعر والرسم، كأمثلة على المهام التي لا يمكن أن يؤديها إلا البشر. بينما ظلت كتابة الشعر صعبة للغاية على أجهزة الكمبيوتر حتى يومنا هذا، فقد حقق الباحثون نجاحاً أكبر بكثير في بناء أجهزة حاسوبية للعب الشطرنج.

الشطرنج ليست فقط واحدة من أقدم وأشهر ألعاب الطاولة في العالم، بل هي أيضاً أرض خصبة للخوارزميات المعقدة، ومؤخراً، التعلم الآلي. تعتبر لعبة الشطرنج نظرياً لعبة حتمية، بمعنى أنه لا توجد معلومات مخفية عن أي من اللاعبين وكل مركز يحتوي على مجموعة قابلة للحساب من الحركات المحتملة. إنها ليست لعبة "محولة"، مما يعني أنه لا يمكن دائماً التنبؤ بنتيجة أي موقف بشكل صحيح، بل يتم تقديرها فقط. لأن عامل التفرع في الشطرنج يتراوح من 35 إلى 38 حركة، مما يعني أنه في كل وضع متوسط 35 إلى 38 إجراء ممكن، هذا التقدير يتطلب عددًا هائلاً من الحسابات.

على مدار تاريخ علوم الحاسوب بأكمله، حاول الباحثون باستمرار إيجاد طرق أفضل لحساب ما إذا كان الموضع يعتبر فائزاً أو خاسراً. أشهر مثال على ذلك هو محرك StockFish، والذي يستخدم خوارزمية minimax مع تقليص alpha-beta لحساب أفضل حركة. في الآونة الأخيرة، طور الباحثون في Google DeepMind خوارزمية جديدة تسمى AlphaZero. تم إنشاء AlphaZero باستخدام التعلم المعزز العميق، من خلال لعب ملايين الألعاب ضد نفسها واستخدام تلك الألعاب كبيانات تدريب للشبكة العصبية.

2.1- هدف المشروع

يهدف هذا المشروع إلى بناء نظام ذكي قادر على التنبؤ بالحركة الأفضل لموضع معين على رقعة الشطرنج، ويمكن اللعب ألعاب كاملة ضده، باستخدام تقنيات الذكاء الاصطناعي.

3.1- متطلبات المشروع

1.3.1- المتطلبات الوظيفية

- إجراء دراسة مرجعية كافية.
- واجهة ويب تتيح للمستخدم اللعب ضد النظام مع تحديد اللون المراد اللعب به.
- إجراء تقييم للنظام.

2.3.1- المتطلبات غير الوظيفية

- إعطاء نتائج التنبؤ في وقت قصير بين 15 و 25 ثانية.
- إعطاء نتائج تنبؤ بدقة مقبولة.

4.1- الخطة الزمنية

اسم المهمة	عدد الأيام
دراسة عامة حول المشروع	6
تحديد متطلبات المشروع	2
الدراسة النظرية	10
الدراسة المرجعية	40
تصميم النظام	30
اختبارات	15
تصميم الموقع	7
كتابة التقرير	10

2-الفصل الثاني

الدراسة النظرية والمرجعية

يهدف هذا الفصل إلى تقديم إطار نظري وخلفية معرفية للقارئ حول المفاهيم والتقنيات المتعلقة بالحل المقترح. من خلال هذا الفصل، سيتم توضيح النظريات والمفاهيم التي يعتمد عليها الحل، بالإضافة إلى استعراض الدراسات والأبحاث السابقة ذات الصلة في هذا المجال.

1.2- الشطرنج (Chess)

الشطرنج هي لعبة لوحية استراتيجية بين لاعبين. يتم لعبها على رقعة مربعة مكونة من 64 مربعاً متناوبة باللونين الأسود والأبيض مرتبة على شكل شبكة 8*8 حيث تسمى الخطوط العمودية أعمدة والخطوط الأفقية صفوف والخطوط المائلة أوتار، مع ستة عشر قطعة لكل لاعب، أحد اللاعبين يلعب بالقطع البيضاء والآخر بالقطع السوداء. هناك ستة أنواع من القطع في الشطرنج: Pawns (جنود)، رخ Rooks (قلاع)، فرسان Knights (أحصنة)، أساقفة Bishops (فيلة)، ملكات Queens (وزراء)، والملوك Kings. يحصل كل لاعب على ثمانية بيادق ورخان وفارسان واثنان من الأساقفة، وملكة وملك واحد. كل قطعة من هذه القطع لها شكل محدد من الحركات التي تستطيع الحركة بها على الرقعة، حيث يتم اللعب بالتناوب ومن يمتلك القطع البيضاء يبدأ أولاً.

يمكن أسر Capture قطع الخصم، وبذلك يتم إزالة القطعة التي تم أسرها من الرقعة، ووضع قطعة المنافس في موضعها، الهدف من اللعبة هو كشف ملك Checkmate للخصم. [8]

1.1.2- تجهيز الرقعة

في بداية لعبة الشطرنج، توضع القطع على الرقعة كما في الشكل (1):



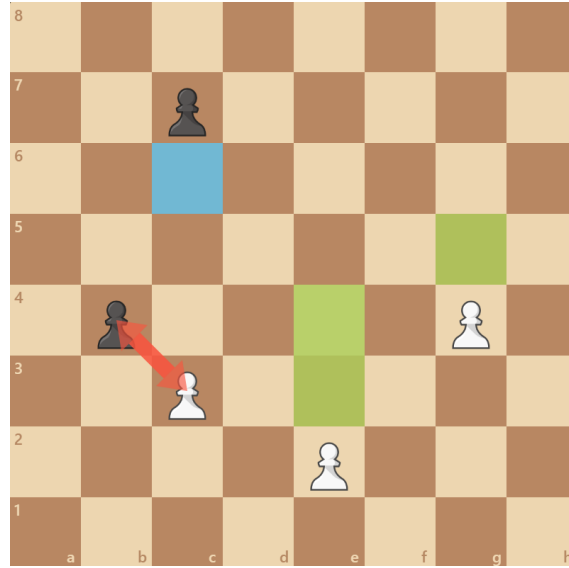
شكل 1: شكل الرقعة الابتدائي

من أسفل اليسار: رخ، فارس، أسقف، ملكة، ملك، أسقف، فارس، رخ. الصف الثاني مملوء بـ 8 بيادق. يضع الخصم نفس القطع على الجانب الآخر من اللوحة بشكل متناظر أفقياً.

2.1.2- حركة القطع

يتحرك كل نوع من الأنواع الستة المختلفة من القطع بشكل مختلف. لا يمكن للقطع التحرك عبر القطع الأخرى (فقط الفارس يمكنه القفز فوق القطع الأخرى)، ولا يمكنها أبداً الانتقال إلى مربع يحتوي قطعة صديقة. ومع ذلك، يمكن تحريكها لتحل محل قطعة الخصم التي يتم أسرها بعد ذلك. يتم نقل القطع بشكل عام إلى مواضع حيث يمكنها أسر قطع أخرى (عن طريق التحرك إلى مربعهم ثم استبدالهم)، أو الدفاع عن قطعهم في حالة أسرها، أو التحكم في المربعات المهمة في اللعبة.

عادة ما تتحرك البيادق للأمام بمقدار مربع واحد فقط في كل مرة، ولكن إذا كان البيدق في موضعه الأولي (في الصف الثاني للأبيض وفي الصف السابع للأسود)، فيمكن للاعب أن يختار تحريكه مربعين للأمام أو مربع واحد للأمام. لأسر القطع، يجب أن يتحرك البيدق قطرياً مربعاً واحداً. ويمكن القيام بذلك فقط عندما يتم أسر قطعة للخصم موضوعة قطرياً للأمام من البيدق. عندما يتحرك أو يأسر البيدق ويصل إلى الجانب الآخر من الرقعة (الصف الثامن بالنسبة للأبيض والأول بالنسبة للأسود)، يجب الترقية Promotion بحيث يختار ترقية البيدق إلى ملكة، أو رخ، أو أسقف، أو فارس، أي يتم استبدال البيدق بالقطعة المختارة.



شكل 2: حركة البيادق وطريقة الأسر بها

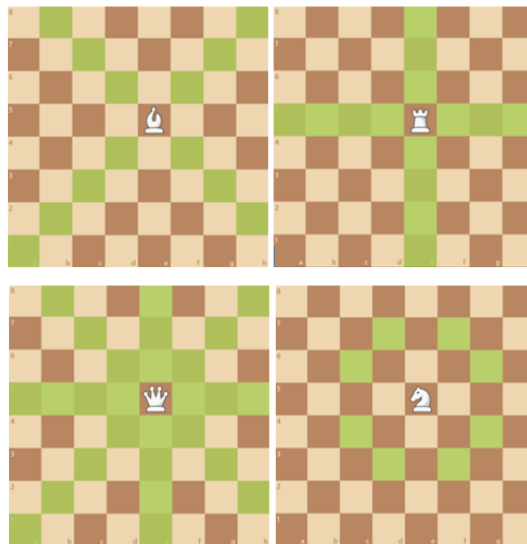
الأسر بالتجاوز "En Passant" هي حركة بيدق خاصة لا تكون ممكنة إلا عندما يقوم اللاعب بتحريك بيدق مربعين للأمام. إذا كان البيدق مجاوراً لأحد بيادق الخصم، فإنه يمكن للخصم أن يختار أسر هذا البيدق كما لو أنه تحرك مربعاً واحداً للأمام،

وهذه الحركة يمكن القيام بها فقط في ذلك الدور ولا يمكن القيام بها في أي دور لاحق، أي يخسر الخصم حق الأسر بالتجاوز إذا لم يقوم بالأسر في هذا الدور.



شكل 3: الأسر بالتجاوز En Passant

يستطيع الرخ التحرك أي عدد من المربعات عمودياً أو أفقياً، في حالة عدم وجود قطع تحجب طريقه. يمكن للأسقف أن يفعل الشيء نفسه، لكن بشكل مائل (وترياً). الملكة هي مزيج من الرخ والأسقف: تستطيع تتحرك عمودياً وقطرياً وأفقياً.



شكل 4: حركة كل من الرخ، الأسقف، الفارس، والملكة

يتحرك الفارس على شكل حرف L تتكون حركة الفارس من تحريك مربع واحد عمودياً واثنين أفقياً، أو العكس. الفارس هو قطعة الشطرنج الوحيدة التي يمكنها القفز فوق القطع الأخرى. يمكن للملك أن يتحرك تمامًا مثل الملكة، ولكن يتحرك مربع واحد فقط في كل مرة ولا يمكنه أبداً التحرك إلى خلية مهاجمة من قبل الخصم. يمكن للملك أيضاً التبييت وهي حركة خاصة تضع الملك في أمان بتحريكه مربعين باتجاه أحد الرخ، وتحريك الرخ إلى المربع الذي عبره الملك.

3.1.2- الكشف والكش مات والموت خنقاً

تهديد الملك تسمى كش Check. عندما يحدث هذا، يجب على الخصم إما تحريك الملك بعيداً عن التهديد، أو إيقاف الهجوم عن طريق أسر القطعة المهاجمة أو صد الهجوم بإحدى القطع. عندما لا يستطيع اللاعب إيقاف التهديد، يخسر اللاعب اللعبة. هذا يسمى بالكش مات Checkmate. لا يمكن للاعب أن ينهي حركته أبداً إذا كان ملكه قيد التهديد. ستكون هذه خطوة غير قانونية Illegal Move. عندما يحين دور اللاعب ولكن لا توجد حركات قانونية متاحة، ولا يكون الملك في حالة تهديد، تنهى المباراة بالتعادل. وهذا ما يسمى الموت خنقاً Stalemate [8].

2.2- محركات الشطرنج (Chess Engines)

محرك الشطرنج هو برنامج حاسوبي يقوم بتحليل المواقف Positions في الشطرنج، ويقوم بإنشاء قائمة بالحركات التي يعتبرها الأقوى. بالنظر إلى أي موضع شطرنج، سيقدر المحرك الفائز في هذا الموضع بناءً على قوة التحركات المستقبلية المحتملة حتى عمق معين. غالباً ما يتم تحديد قوة محرك الشطرنج من خلال عدد المواضع، سواء من حيث العمق أو العرض، التي يمكن للمحرك تقييمها.

لنفترض موضع معين، نريد أن نجد حركة ما من كافة الحركات القانونية في هذا الموضع، التي سوف تزيد من فرصتنا في الفوز باللعبة. هذا يعني أنه يجب أن يكون لدينا نموذج للخصم لاختيار الحركات. في لعبة الشطرنج عالية المستوى، عادة ما نصمم الخصم ليكون مثلنا أي نفترض أنهم يختارون الحركات تماماً كما نفعل نحن، ولا يخطئون أبداً.

قد لا يكون هذا الافتراض صالحاً في المباريات ضد لاعبين أضعف بكثير، حيث، على سبيل المثال، قد يكون لدينا حركتان محتملتان A و B، حيث تؤدي الحركة A إلى وضع أفضل لنا ما لم يرى الخصم رداً معقداً للغاية ويلعبه (هذا رأينا ولكن نعتقد أن الخصم سوف يغفل بسبب عدم الكفاءة النسبية). لكن من الخطورة الاعتماد على ارتكاب الخصوم للأخطاء (أو ما يبدو أنه أخطاء من وجهة نظرنا)، لذا فإن جميع محركات الشطرنج تقريباً تطبق النموذج الذي يفترض خصوماً مثاليين.

هناك صيغة بديلة للتقييم تستخدم في كثير من الأحيان في الممارسة. بالنظر إلى موضع الشطرنج، نريد تخصيص درجة لها تتوافق مع فرصة الفوز للجانب صاحب الدور للتحرك. الشرط الوحيد لهذه النتيجة هو أنها يجب أن تزداد بشكل رتيب فيما يتعلق بفرصة الفوز. في هذه الحالة، نظرًا لأن لعبة الشطرنج عبارة عن لعبة محصلتها صفر، فإن النتيجة الخاصة بأحد الجانبين هي ببساطة النتيجة السلبية للجانب المقابل، لأي موضع معين.

3.2- محركات الشطرنج التقليدية

1.3.2- خوارزميات تقليل الحد الأعظمي لمقدار الخسارة Minimax

تعد خوارزمية minimax عامة حيث يمكن استخدامها في العديد من التطبيقات، بدءاً من الذكاء الاصطناعي لنظرية القرار ونظرية الألعاب. تحاول الخوارزمية تقليل الحد الأقصى لمقدار الخسارة. في لعبة الشطرنج، هذا يعني أن المحرك يحاول تقليل احتمالية حدوث أسوأ السيناريوهات، أي أن الخصم يقوم بكش مات اللاعب. بدلاً من ذلك، بالنسبة للألعاب التي يحتاج فيها اللاعب إلى تعظيم قيمة الموضع، فإن تسمى الخوارزمية maximin: تعظيم الحد الأدنى من الكسب.

تعمل هذه الخوارزمية من الناحية النظرية، وكذلك في الممارسة العملية للألعاب الأبسط مثل tic-tac-toe (حجم شجرة اللعبة على الأكثر 9! أو 362880). ومع ذلك، نظرًا لأنها تبحث في شجرة اللعبة بأكملها، فمن غير العملي تطبيقها على ألعاب مثل الشطرنج، حيث يبلغ متوسط عامل التفرع (عرض الشجرة) حوالي 35، ويبلغ متوسط طول اللعبة حوالي 80 حركة. يقدر حجم شجرة البحث عن لعبة الشطرنج بحوالي 10^{123} (10^{46} دون التكرار) وهو أعلى بكثير من مائة مرتبة مما هو ممكن حسابه باستخدام أجهزة الكمبيوتر الحديثة. لذلك، يجب أن يقرر محرك الشطرنج أي أجزاء من شجرة اللعبة يجب استكشافها. الطريقة الأكثر شيوعاً هي البحث ذو العمق الثابت، حيث نحد من المدى الذي سننظر فيه إلى الأمام، وعندما نكون في نهاية الشجرة الفرعية التي نريد البحث فيها، فإننا نستخدم تابع تقييم ثابت الذي يعين قيمة إلى الموضع من خلال تحليله بشكل ثابت (دون النظر إلى الأمام).

يتم تقييم موضع تلك العقدة الورقة ويتم إرجاع قيمتها للأعلى إلى العقدة الأصلية. تنظر العقدة الأب إلى جميع قيم الأبطال وتختار القيمة الأعلى بينها عندما يكون دور الأبيض، والقيمة الأدنى عندما يكون دور الأسود. يتكرر هذا حتى تتلقى العقدة الجذرية قيمة قوة الموضع الحالي.

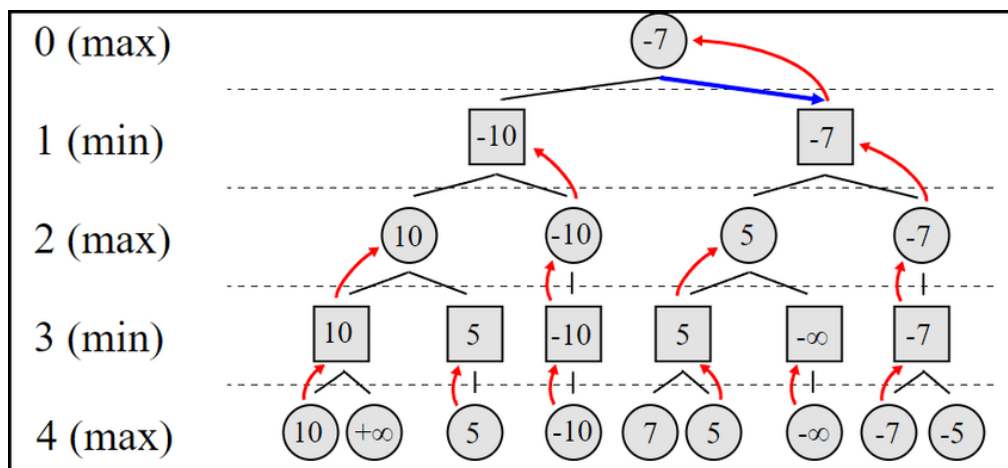
في هذه الخوارزمية قمنا بقطع جميع الفروع على نفس المسافة من الجذر (تسمى الأفق). هذا أمر خطير لأنه، على سبيل المثال، إذا حدث في فرع واحد أن تكون الخطوة الأخيرة هي QxP (الملكة تأسر بيدق)، فقد نحرز هذا الموضع كموقع جيد جداً، لأننا فزنا للتو بيدق. ومع ذلك، ما لا نراه هو أن البيدق يتم الدفاع عنه بواسطة بيدق آخر، وسيأخذ الخصم ملكتنا في الخطوة التالية. هذا في الواقع وضع سيء للغاية لنكون فيه.

لا يمكن حل هذه المشكلة ببساطة عن طريق زيادة حد العمق، لأنه بغض النظر عن مدى عمق البحث، سيكون هناك دائماً أفق. الحل هو البحث الهادئ. الفكرة هي أنه بمجرد أن نصل إلى الأوراق، بدلاً من استدعاء تابع التقييم، ندخل في وضع بحث خاص يوسع فقط أنواعاً معينة من الحركات، ونستدعي تابع التقييم فقط عندما نصل إلى موقف مستقر نسبياً.

أنواع الحركات التي يجب تضمينها في هذا البحث هي مسألة نقاش. بحيث يتفق مؤلفي المحرك على أنه يجب تضمين الأسر الفائر (التقاط قطعة ذات قيمة أعلى بقطعة أقل قيمة). هناك مجموعة حالات أخرى يمكن تضمينها والتي هي حالات الأسر الأخرى، الترقية، الكش، البيدق في الصف السابع (الصف الأخير قبل الترقية).

هناك مقايضة يجب إجراؤها هنا إذا قمنا بتضمين عدد كبير جداً من الحركات، فإن عمليات البحث الخاص سوف تصبح كبيرة جداً، ولن نتمكن من البحث في العديد من الطبقات في البحث العادي. إذا قمنا بتضمين عدد قليل جداً من التحركات، فقد نعاني من أشكال مخفضة لتأثير الأفق. [4]

```
function minimax(node, depth, maximizingPlayer) is
  if depth = 0 or node is a terminal node then
    return the heuristic value of node
  if maximizingPlayer then
    value := - inf
    for each child of node do
      value := max(value, minimax(child, depth - 1, FALSE))
    return value
  else (* minimizing player *)
    value := + inf
    for each child of node do
      value := min(value, minimax(child, depth - 1, TRUE))
    return value
```



شكل 5: مثال على شجرة البحث minimax

2.3.2- تابع التقييم (Evaluation Function)

يتم تقدير قيمة العقد الأوراق بواسطة تابع تقييم مكتوب خصيصاً للعبة heuristics. يمكن أن يختلف هذا التابع من محرك إلى محرك، وعادة ما تتم كتابتها بمساعدة خبراء الشطرنج، أحد توابع التقييم البسيطة للغاية هي ببساطة جمع قطع من كلا الجانبين، كل منها مضروب بثابت.

يتكون تابع التقييم في Stockfish من 9 أجزاء:

1. قيمة القطع: كل قطعة على الرقعة تحصل على قيمة لوجودها. يتم أيضا أخذ تأثيرات أخرى في الاعتبار (على سبيل المثال، يمنع كلا الأسقفين قيمة أعلى من ضعف القيمة المادية للأسقف الواحد)
2. جدول خلايا-قطع: تحصل كل قطعة على مكافأة أو عقوبة بناءً على مكان وجودها، بغض النظر عن مكان القطع الأخرى. يشجع مصطلح التقييم هذا المحرك على تطوير البيادق وتطوير الفرسان والأساقفة على سبيل المثال.
3. هيكل البيادق: يتم فحص الموضع بحثاً عن البيادق السالكة والمعزولة والمتعارضة والخلفية وغير المدعومة وذات الرافعة. يتم تعيين المكافآت والعقوبات لكل ميزة.
4. تقييم خاص بالقطعة: يتم تقييم كل قطعة على حدة، باستخدام ميزات خاصة بالقطعة. على سبيل المثال، يحصل الأساقفة والفرسان على مكافأة لكونهم في مركز Outpost.
5. الرخ تحصل على قيمة أعلى لكونها في عمود مفتوح أو عمود شبه مفتوح أو نفس صف بيادق الخصم.
6. قابلية الحركة: تحصل القطع على قيم إضافية مقابل عدد الحركات الممكنة لديها. في تطبيق Stockfish، لا يتم احتساب المربعات التي تسيطر عليها قطع الخصم ذات القيمة الأقل. على سبيل المثال، بالنسبة لحركة الأسقف، لا يتم تضمين المربعات التي يتم التحكم فيها بواسطة بيادق العدو، وبالنسبة لحركة الملكة، لا يتم تضمين المربعات التي يتحكم فيها الأساقفة أو الفرسان أو الرخ. كل نوع قطعة لديه مجموعة مختلفة من قيم قابلية الحركة.
7. أمان الملك: يتم منح المكافآت والعقوبات بناءً على عدد المهاجمين وقرهم، واكتمال ملجأ البيادق، وحقوق التثبيت.
8. التهديد: يتم منح الجزاءات للقطع غير المحمية ويتم منح القطع المحمية مكافآت حسب نوع القطعة ونوع المدافع.
9. المساحة: يتم منح المكافآت لوجود مربعات فارغة آمنة على جانب اللاعب.
10. التعادل: غالباً ما تؤدي تركيبات مادية معينة إلى التعادل، لذلك في هذه الحالات، يتم تحجيم التقييم باتجاه الصفر.

تابع التقييم هذا معقد للغاية مع الكثير من المعرفة المبرمجة يدوياً. لا تحتوي معظم الحركات على وظائف تقييم واسعة النطاق تقريباً، لأنه من الصعب ضبط مثل هذا العدد الكبير من المعاملات يدوياً. [4]

3.3.2- تقليل ألفا-بيتا (Alpha-Beta Pruning)

نظراً لأن عدد العقد اللازمة للحصول على تقييم جيد لقوة الموضع مرتفع جداً، يجب تحسين الخوارزمية. يهدف تقليل Alpha-beta إلى تقليل عدد العقد التي يجب استكشافها بواسطة minimax. يتم ذلك عن طريق قطع الفروع في شجرة البحث التي تؤدي إلى نتائج أسوأ وذلك بدون تقديم أي استدلالات وبدون فقدان أي معلومات، يمكن تحسين هذه الخوارزمية عن طريق إدخال نافذة لكل استدعاء. إذا كانت النتيجة الحقيقية أقل من الحد الأدنى، فهذا يعني أن اللاعب لديه بالفعل حركة أفضل،

وبالتالي لا يهتم بالقيمة الدقيقة لهذه العقدة (فقط لأنها أقل من الحد الأدنى). بالمقابل، إذا كانت النتيجة الحقيقية أعلى من الحد الأعلى، فهذا يعني أن الخصم لن يحاول لعب هذه الحركة لأن لديه حركة أفضل فأيضاً لا نهتم بالقيمة الدقيقة لهذه العقدة. قد يبدو من غير البديهي في البداية أن يكون لدينا حد علوي، ولكن السبب في ذلك هو أن لعبة الشطرنج ذات مجموع صفري، والحد الأعلى والأدنى لكل طبقة يتم التبديل بينهما كل مستوى من الشجرة.

في خوارزمية minimax، يكون ترتيب الحركات (الترتيب الذي نستكشف العقد به) غير مهم لأنه سوف يتم زيارة جميع العقد مرة واحدة بالضبط. من خلال تقليل $\alpha\beta$ ، يستكشف فقط العقد التي يمكن أن تكون مفيدة، والتوقف عن البحث عند عقدة بمجرد أن نثبت أن النتيجة ستكون خارج النافذة. هذا يعني، إذا تم توسيع أفضل العقد أولاً، فلا حاجة لفحص العديد من العقد.

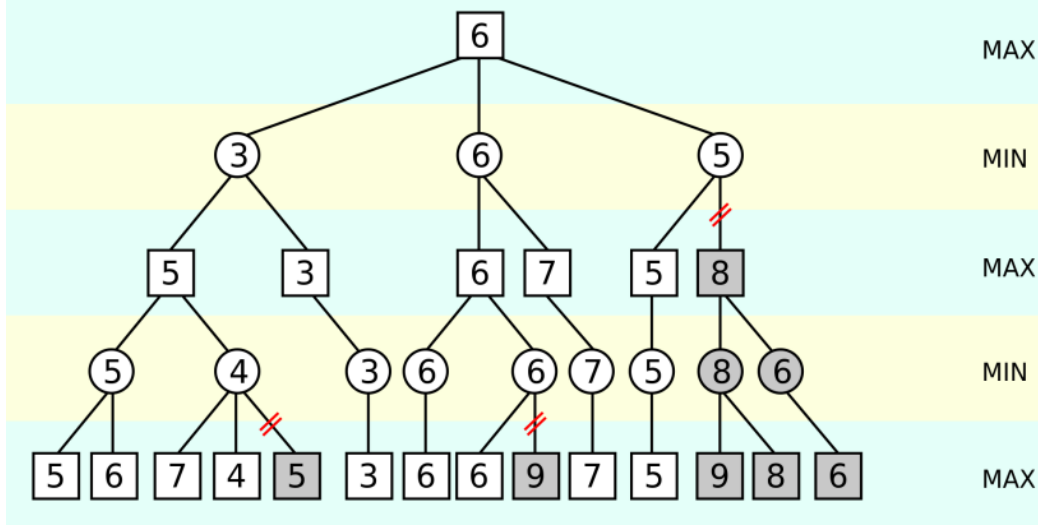
في أسوأ الحالات (عكس الترتيب الأمثل)، يتدهور تعقيد $\alpha\beta$ ليصل لتعقيد minimax. من ناحية أخرى، فقد ثبت أنه في أفضل الحالات، حيث يتم البحث دائماً عن أفضل الحركات أولاً، تقلل $\alpha\beta$ عامل التفرع الفعال إلى الجذر التربيعي للقيمة الأصلية، مما يسمح بالبحث عن ضعف المسافة في نفس الوقت.

من الواضح أنه من المستحيل دائماً الحصول على ترتيب مثالي لأنه إذا كانت هناك طريقة لإنشاء ترتيب مثالي لأي موضع معين، فلن تكون هناك حاجة للبحث على الإطلاق. ومع ذلك، من خلال التأكد من أن ترتيب الحركات قريب من المستوى الأمثل في معظم الحالات باستخدام الاستدلال، يمكننا جعل البحث أكثر كفاءة.

في محركات الشطرنج التقليدية، هناك عدد قليل من الاستدلالات الشائعة لتحسين ترتيب الحركات:

- إذا تم البحث سابقاً في نفس الموضع، فالحركة التي انتهت بها الأمر أن تكون الأفضل في البحث السابق من المحتمل أن تبقى الحركة الأفضل في البحث الحالي (حتى لو تم إجراء البحث السابق على عمق ضحل).
- إذا ثبت أن الحركة جيدة في العقدة الشقيقة، فهناك فرصة جيدة لتكون جيدة للعقدة قيد الدراسة أيضاً.
- عادةً ما يكون أسر القطع عالية القيمة بقطع منخفضة القيمة أمراً جيداً، وعادة ما يكون أسر القطع منخفضة القيمة المحمية بقطع عالية القيمة أمراً سيئاً.
- عادة ما تكون الترقية إلى ملكة جيدة. [1]

لنفترض أننا نلعب بالقطع البيضاء. نريد تقليل الحد الأقصى للخسارة، مما يعني أننا نريد التأكد من أن قيمة الأسود منخفضة قدر الإمكان. يفترض minimax أن الخصم سيلعب أفضل حركة ممكنة. إذا أدت إحدى حركات الأبيض الممكنة إلى موضع يحصل فيه الأسود على ميزة كبيرة، فسيؤدي ذلك إلى القضاء على هذا الفرع من شجرة البحث والعكس بالعكس. نتيجة لذلك، يتم تقليل عدد العقد التي يجب استكشافها بشكل كبير.



شكل 6: مثال عن تقليل ألفا بيتا في شجرة ال minimax

4.3.2- الأعمال السابقة

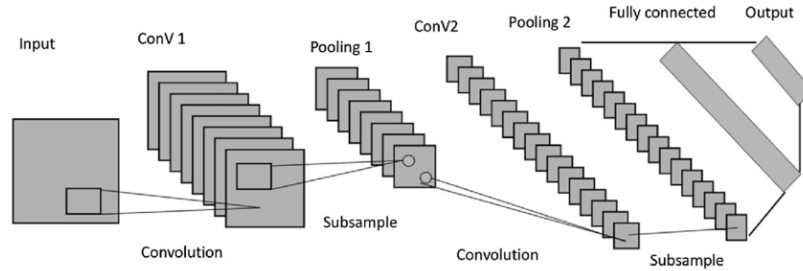
كان استخدام تقليل ألفا بيتا في برمجة الشطرنج تقدما محوريا، حيث عزز بشكل كبير كفاءة خوارزميات البحث. أسست المساهمات التاريخية، بما في ذلك الورقة التأسيسية لـ Claude Shannon's، الأساس لتطبيق هذه التقنية. نفذت محركات الشطرنج المبكرة البارزة مثل "MacHack VI" و "Cray Blitz" تقليل ألفا بيتا لتحقيق أعماق بحث أعمق. تميزت التطورات في الكفاءة بدمج تقنيات مثل جداول التحويل والنوافذ الأصغرية. أبرزت النجاحات البارزة، مثل Deep Blue لشركة IBM، فعالية الخوارزمية. تساهم الورقة البحثية "تحليل محرك الشطرنج استناداً إلى تقليل ألفا بيتا، معزز بالتعميق التكراري" في هذا الإرث من خلال توفير رؤى حول تحليل الأداء والتحسينات التي تم تحقيقها من خلال التعميق التكراري.[1]

4.2- الشبكات العصبونية التلافيفية (Convolutional Neural Networks)

الشبكات العصبونية التلافيفية (CNNs) هي فئة من الشبكات العصبونية العميقة المصممة خصيصًا لمعالجة البيانات الشبيهة بالشبكة، مثل الصور ومقاطع الفيديو. لقد أثبتت فعاليتها العالية في مهام رؤية الكمبيوتر نظرًا لقدرتهم على التعلم التلقائي واستخراج الميزات الهرمية من البيانات الأولية. يتمثل الابتكار الرئيسي لـ CNN في استخدام الطبقات التلافيفية التي تطبق المرشحات (المعروفة أيضًا باسم النواة) على بيانات الإدخال، مما يمكنها من التقاط الأنماط والهياكل المحلية.

1.4.2- بنية الشبكات التلافيفية

يتكون هذا النوع من الشبكات من عدد من الطبقات التي تساعد في عمليات استخراج السمات من الصورة وتحسينها أو اهمالها، وسنأتي في الأقسام اللاحقة إلى عمل أشهر أنواع طبقات هذه الشبكة، كما يمكننا أن نرى الشكل (7) مثالاً عن هيكل الشبكات العصبونية التلافيفية.



شكل 7: هيكل الشبكة العصبونية

1.1.4.2- الطبقات التلافيفية (Conv Layers)

تعتبر هذه الطبقات من الطبقات الأساسية في الشبكات التلافيفية والتي تقوم بغالبية عمليات الحساب التي تتم على هذه الشبكة، حيث أنها تقوم بإيجاد جداء النقطة Dot Product أو جداء التلاق بين مصفوفتين، بحيث تكون المصفوفة الأولى عبارة عن مجموعة من القيم التي يتم تعلمها خلال عملية التدريب وتسمى النواة Kernel، والمصفوفة الثانية تكون عبارة عن قطعة من الصورة (مجموعة البكسلات)، تكون أبعاد النواة عادة أعداد فردية وأصغر بكثير من أبعاد الصورة مثلاً (1، 3، 5، 7) للطول والعرض.

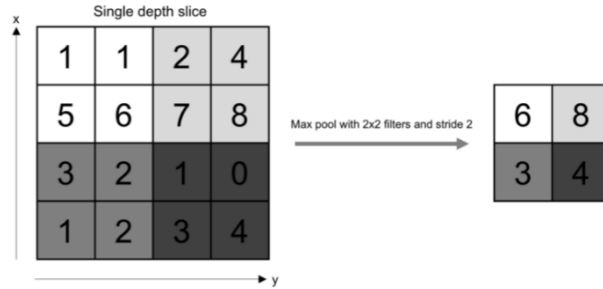
خلال عملية العبور الأمامي للشبكة، نقوم بزلق Slide النواة على طول وعرض صورة الدخل مما ينتج لنا تمثيل لتلك المنطقة من الصورة كمصفوفة ببعدين ويسمى هذا التمثيل بخريطة التنشيط Activation Map أو خريطة السمات Feature Map التي تعطي استجابة النواة عند كل بكسل من الصورة، ونسعى بمقدار انزلاق النواة بالشريط Stride، وكما يجب أن نقوم بمعالجة

حالة البكسلات الطرفية التي سيكون عندها بعض قيم النواة خارج الصورة، وتتم معالجة هذه الحالة بإضافة حشوة Padding لأطراف الصورة وتكون هذه الحشوة إما بكسلات سوداء أو بيضاء أو نقوم بتكرير البكسلات الطرفية.

2.1.4.2- طبقات الانتخاب (Pooling Layers)

تقوم هذه الطبقات باستبدال خرج الشبكة بمواقع معينة بملخص إحصائي مشتق من قيم الخرج المجاورة، كأن نقوم باستبدال مجموعة قيم متجاورة بأكبر أو أصغر قيمة بينهم، مما يساعد بتخفيف الحجم المكاني (أبعاد الخرج) للخرج السابق، بالتالي تقليل حجم الحسابات والأوزان المطلوبة، كما وإن عملية الانتخاب هذه تتم على كل شريحة Slice من التمثيل (الخرج السابق) بشكل منفصل.

كما ويوجد عدد كبير من التوابع التي يمكن استخدامها كتوابع انتخاب، مثلاً متوسط المجاورات، أو المتوسط المثلث بالاعتماد على البعد عن البكسل المركزي، وتعتبر طريقة القيمة الكبرى من أكثر الطرق المستخدمة في هذا النوع من الطبقات وتعمل هذه الطريقة على أن تقوم باستبدال كل مجموعة متجاورة من البكسلات بأعلى قيمة بينها، ويمكننا أن نرى مثلاً على طريقة Max Pooling في الشكل (8).



شكل 8: طريقة عمل طبقات انتخاب أعظمي

بالتالي فإن هذه الطبقة أيضاً تساهم في تخفيف أبعاد خريطة السمات، وإذا استخدمنا تابع Max بالتالي فإننا نأخذ أكبر القيم ضمن الدخل ومنه فإننا نقوم بعملية تحسين Enhance للسمات المهمة التي تعبر عن التفاصيل ونحمل السمات غير الأساسية كالحلفية.

3.1.4.2- طبقات الإهمال (Dropout)

من المشاكل الشائعة جداً عند القيام ببناء شبكة عصبية وتدريبها أن يكون النموذج الناتج يعمل بشكل مقالي على بيانات من نفس طبيعة بيانات التدريب، ولكن إذا أضفنا بعض الضجيج إلى هذه البيانات فسينخفض أداء هذا النموذج، ومن هنا جاء مصطلح Overfit الذي يعني أن يكون النموذج مرتبط ارتباطاً وثيقاً ببيانات الدخل، وإن أي تغيير بجودة البيانات أو أبعادها

فسوف تتأثر النتائج بشكل كبير، ومنه فإن هذه الطبقات تفيد في تفادي حدوث حالات Overfitting أثناء عملية تدريب الشبكة، حيث أننا ضمن هذا النوع من الطبقات نقوم بالاستغناء عن مجموعة عشوائية من قيم خرج الطبقة السابقة، مما يجعل عملية التدريب تحتوي على ضجيج مما يزيد من دقة أدائها على البيانات الجديدة.

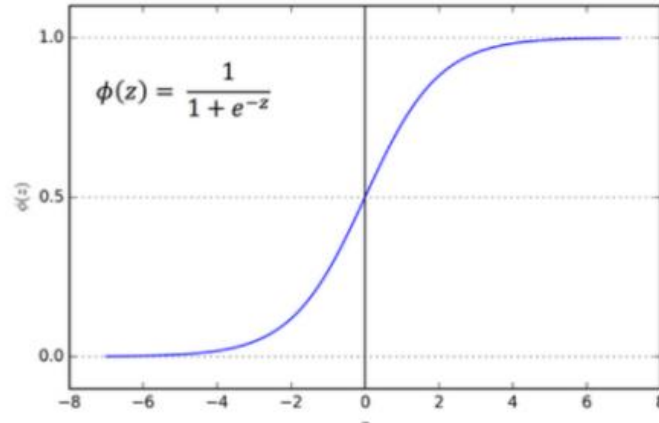
4.1.4.2- توابع التنشيط (Activation Function)

تعتبر هذه التوابع من التوابع الهامة جداً في بنية الشبكات التلافيفية وخصوصاً في الطبقات المتصلة كلياً، التي سنأتي على ذكرها في الفقرة التالية، وتأتي أهميتها من كونها التوابع التي تحدد كيفية نقل قيم شعاع السمات من طبقة إلى أخرى كما وتحدد طبيعة قيم هذا الشعاع والمجال الذي تتوزع عليه القيم، وعادةً نستخدم توابع خاصة تتميز بسمات معينة، من أهمها:

غير الخطية، حيث أنه يجب على تابع التنشيط ألا يكون خطياً وذلك لكي يتمكن النموذج من تعلم وتنفيذ مهام معقدة، حيث أن هذه التوابع تسمح بتكديس طبقات متعددة من العصبونات التي تشكل شبكة عصبية، ولو تخيلنا كون توابع التنشيط خطية بالتالي فيمكننا أن نقوم بتجميع هذه التوابع الخطية لتشكيل تابع خطي جديد بالتالي تصبح واحدة، ومنه لو أردنا شبكة عصبية بعدة طبقات قادرة على تعلم مجموعة معطيات معقدة لا يمكن التنبؤ بها فلا بد لنا أن نستخدم توابع غير خطية.

ومن أشهر توابع التنشيط غير الخطية، نذكر:

- Sigmoid: يأخذ هذا التابع منحني كما في الشكل (9)، والسبب الرئيسي في استخدام هذا التابع هو أنه يأخذ قيمه في المجال $[0,1]$ ، بالتالي فهو يستخدم بشكل خاص عندما يكون خرج الشبكة هو احتمال انتماء الدخل إلى صفوف الخرج، حيث أن احتمال أي شيء لا يمكن أن يكون خارج هذا المجال.



شكل 9: منحني ومعادلة تابع Sigmoid

وتعطى معادلة هذا التابع كما يلي:

$$S(z) = \frac{1}{1+e^{-z}} \quad (1)$$

- Softmax: يعتبر هذا التابع مشابهاً لتابع Sigmoid حيث أنه يأخذ قيمه أيضاً ضمن المجال $[0,1]$ ، ولكن الاختلاف هو أن قيم تابع Sigmoid يمكن أن يكون مجموعها أكبر من واحد، حيث أنه يقوم بمعالجة احتمال انتماء الدخل إلى صفوف الخرج بشكل منفصل، على عكس تابع Softmax الذي يكون مجموع قيمه هو 1، بالتالي فهو يعطي احتمال انتماء الدخل إلى صفوف الخرج ضمن نفس التوزع لأنه يأخذ بعين الاعتبار نتائج باقي الصفوف، وتعطى عبارة تابع Softmax بالعبارة التالية:

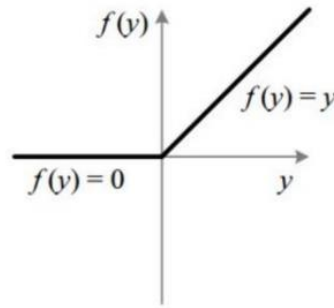
$$S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2)$$

حيث أن، z هو شعاع الدخل، و K هو عدد الصفوف المستخدمة.

- TanH: أو ما يسمى بـ hyperbolic tangent، يشبه هذا التابع بشكل كبير تابع Sigmoid، ولكنه يأخذ قيمه في المجال $[-1,1]$.
- ReLU أو Rectified Linear Unit، إنّ هذا التابع هو أشهر توابع التنشيط وأكثرها انتشاراً، وذلك لأنه يستخدم في غالبية الشبكات العصبية التلافيفية وشبكات التعلم العميق، وتعطى معادلته كما يلي:

$$ReLU(z) = \max(0, z) \quad (3)$$

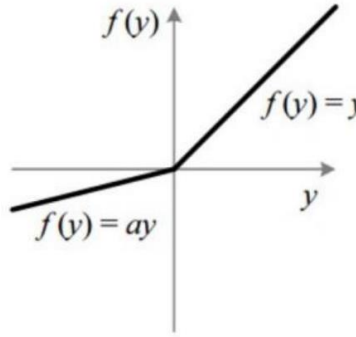
والسبب الأساسي في كثرة استخدام هذا التابع، هو سرعته حيث أنّه يقوم بوضع جميع القيم السالبة إلى الصفر، بالتالي تخفيف حجم الحساب بشكل كبير، ويمكننا أن نرى شكل هذا التابع في الشكل (10).



شكل 10: منحني تابع ReLU

إن هذا التابع يقوم بوضع جميع القيم السالبة إلى الصفر، بالتالي فإذا كان الدخل مثلاً يحوي على قيم سالبة فإنها تصبح صفرية مباشرة مما يعيق عملية التعلم. من هذه القيم التي قد تكون دلالة معينة مرتبطة بالمعطيات.

- Leaky ReLU: يعتبر هذا التابع كتطوير للتابع السابق، حيث أنه يخفف أثر القيم السالبة ولا يعدمها مباشرةً، وبعد عدد من عمليات العبور ستتناقص القيمة المطلقة لهذه القيم لتتقارب من الصفر لو كانت فعلاً لا تحمل أي دلالة، ويمكننا أن نرى شكل هذا التابع في الشكل (11).



شكل 11: منحني تابع Leaky ReLU

5.1.4.2- الطبقات المتصلة كلياً Fully connected layers

تتموضع هذه الطبقات عادةً في نهاية معمارية الشبكة، بعد آخر طبقة انتخاب Pooling أو Conv، ويكون دخل هذه الطبقات هو خريطة السمات الناتجة عن الطبقة السابقة بعد أن نقوم بعملية تسطيح flatten لها، حيث أنّ هذه العملية تعبر عن تحويل خريطة السمات من ثلاث أبعاد إلى شعاع واحد يحوي كافة قيم خريطة السمات.

إنّ العمليات الحسابية التي تتم على الشعاع السابق هي نفس العمليات المستخدمة في الشبكات العصبية الصناعية ANN، والتي تمثّل بالعبرة التالية:

$$g(Wx + b) \quad (4)$$

حيث أن:

x ، هو شعاع الدخل ذو الأبعاد $[P_1, 1]$ ، حيث أنّ P_1 هو عدد عصبونات الطبقات السابقة.

W ، هي مصفوفة الأوزان ذات الأبعاد $[P_1, N_1]$ ، حيث أنّ N_1 هو عدد عصبونات الطبقة الحالية.

b ، هو شعاع الانحياز bias ذو الأبعاد $[P_1, 1]$.

g ، هو تابع التنشيط، والذي يكون عادةً هو تابع وحدة التصحيح الخطية ReLU.

وبعد عبور كافة الطبقات المتصلة كلياً، يوجد عادةً طبقة أخيرة تستخدم تابع تنشيط Softmax، الذي يقوم بإيجاد احتمال انتماء الدخل إلى كل صف من صفوف الخرج.

6.1.4.2- توابع التقييم

في سياق تدريب الشبكة العصبية الاصطناعية وبهدف تقليل الخطأ بين خرج الشبكة والخرج المتوقع يتم استخدام توابع الخسارة.

- متوسط الخطأ المطلق Mean Absolute Error

تابع يستخدم لنماذج الانحدار. ويمثل متوسط مجموع الاختلافات المطلقة بين القيم الحقيقية y_i والقيم المتوقعة من قبل النموذج \hat{y}_i كلما كان أصغر كان النموذج أفضل. ويحسب بالعلاقة

$$MAE = \frac{1}{N} \sum_i |y_i - \hat{y}_i| \quad (5)$$

- متوسط الخطأ التربيعي Mean Square Error

تابع يستخدم لنماذج الانحدار. ويمثل متوسط مجموع مربع الاختلافات بين القيم الحقيقية y_i والقيم المتوقعة قبل النموذج \hat{y}_i وكلما كان أصغر كان النموذج أفضل. ويحسب بالعلاقة

$$MSE = \frac{1}{N} \sum_i (y_i - \hat{y}_i)^2 \quad (6)$$

- الجذر التربيعي لمتوسط الخطأ Root Mean Square Error

تابع يستخدم لنماذج الانحدار كلما كان أصغر كان النموذج أفضل. ويحسب بجذر العلاقة السابقة

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_i (y_i - \hat{y}_i)^2} \quad (7)$$

- متوسط نسبة الخطأ المطلق Mean Absolute Percent Error

تابع يستخدم لنماذج الانحدار، ويمثل متوسط مجموع نسب الاختلافات المطلقة بين القيم الحقيقية والقيم المتوقعة من قبل النموذج كلما كان أصغر كان النموذج أفضل. ويحسب بالعلاقة

$$MAPE = \frac{1}{N} \sum_i \frac{|y_i - \hat{y}_i|}{y_i} \quad (8)$$

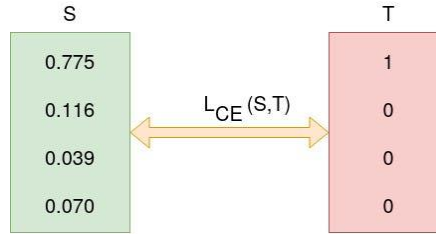
- تابع logcosh

تابع يقوم بحساب الخطأ بحسب العلاقة

$$\logcosh = \sum_i \log (\cosh(y_i - \hat{y}_i)) \quad (9)$$

- معيار Binary Cross Entropy

تقوم Softmax بتحويل الدخل إلى احتمال انتماءه إلى صفوف الخرج. الغرض من الانتروبي المتقاطع هو أخذ احتمالات الإخراج (P) وقياس المسافة مع القيم الحقيقية، الهدف هو جعل القيمة المتوقعة أقرب ما يمكن إلى القيمة الحقيقية، كلما كانت المسافة أصغر كان النموذج أفضل.



شكل 12: المسافة بين القيمة الحقيقية **T** والقيمة المتوقعة **S**

ويعطى بالعلاقة

$$H_p = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \quad (10)$$

حيث $p(y_i)$ هي احتمال التوقع أي هي نفسها \hat{y}_i القيم المتوقعة قبل النموذج و y_i هي القيمة الحقيقية.

- معيار الدقة Accuracy

معيار الدقة يستخدم في نماذج التصنيف، ويعبر عن نسبة التوقعات الصحيحة إلى عدد العينات المدخلة للمصنف تتراوح قيمته بين 0 و 1 كلما كانت القيم قريبة من 1 كان النموذج أفضل. ويعطى بالعلاقة

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad (11)$$

لفهم العلاقة السابقة ليكن لدينا مجموعة بيانات لصنفين سالب وموجب أو 0 و 1، بتدريب نموذج تصنيف على بيانات الصنفين للتنبؤ بالبيانات المدخلة تنتمي لأي من الصنفين لدينا مصفوفة الارتباك Confusion Matrix. تعرض المصفوفة الحقيقية على العمود الأيسر من المصفوفة ويمثل السطر الأول التصنيفات التي يتوقعها النموذج وتمثل الرموز.

- TP: True Positive عدد البيانات التي أصاب النموذج بتوقع الصنف الموجب لها.
- TN: True Negative عدد البيانات التي أصاب النموذج بتوقع الصنف السالب لها.
- FP: False Positive عدد البيانات التي صنفها النموذج موجبة وهي بالحقيقة سالبة.
- FN: False Negative عدد البيانات التي صنفها النموذج سالبة وهي بالحقيقة موجبة.

		Predicted 0	Predicted 1
Actual 0		TN	FP
Actual 1		FN	TP

شكل 13 : مصفوفة الارتباك Confusion Matrix

- معيار الاسترجاع Recall

ويعرف بمعيار الحساسية ويعبر عن نسبة القيم الموجبة التي استطاع النموذج توقعها بشكل صحيح على جميع القيم الموجبة تتراوح قيمته بين 0 و 1 كلما كانت القيمة قريبة من 1 كان النموذج أفضل. ويعطى بالعلاقة

$$Recall = \frac{TP}{TP+FN} \quad (12)$$

- معيار الضبط Precision

ويعرف بمعيار النوعية ويعبر عن نسبة القيم الموجبة استطاع النموذج توقعها بشكل صحيح على مجمل العينات التي قال عنها المصنف بأنها موجبة تتراوح قيمته بين 0 و 1 كلما كانت القيمة قريبة من 1 كان النموذج أفضل، ويعطى بالعلاقة

$$Precision = \frac{TP}{TP+FP} \quad (13)$$

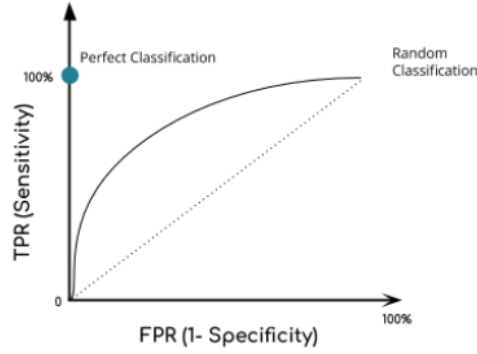
- معيار F1-Score

لأهمية المعيارين السابقين كان لا بد من معيار موازنة بين المعيارين السابقين وهو عبارة عن المتوسط التجانسي لكل من معيار الضبط والاسترجاع تتراوح قيمته بين 0 و 1 كلما كانت القيمة قريبة من 1 كان النموذج أفضل، ويعطى بالعلاقة

$$F_1 Score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 \times precision \times recall}{recall + precision} \quad (14)$$

- معيار المساحة تحت المنحني (AUC) Area Under the Curve

معيار يقيس قدرة النموذج على التمييز بين الأصناف تتراوح قيمته بين 0 و 1، وكلما ارتفعت قيمته كان أداء النموذج أفضل في التمييز بين الفئات يتم رسم معيار الحساسية مقابل معيار النوعية (تم ذكرهم سابقا) لاعتبارات تصنيف مختلفة



شكل 14 : المساحة تحت المنحني AUC

7.1.4.2- توابع الأمثلة أو التحسين (Optimizers)

توابع الأمثلة عبارة عن خوارزميات مستخدمة لتغيير سمات الشبكة العصبية مثل الأوزان ومعدل التعلم من أجل تقليل قيمة تابع الخسارة. هناك عدة خوارزميات نذكر منها:

RMSProp (Root Mean Square Propagation) هي خوارزمية تحسين تستخدم بشكل شائع في تدريب الشبكات العصبية. إنها تتكيف مع معدل التعلم لكل معامل بناءً على مقادير تدرجاتها التاريخية. يساعد RMSProp على تثبيت التدريب عن طريق ضبط التحديثات لكل معامل على حدة، مما يمنع التغيرات الشديدة في معدل التعلم.

Adam (Adaptive Moment Estimation) مقدر اللحظة التكيفية هو خوارزمية تحسين أخرى تجمع بين فوائد كل من RMSProp والزخم Momentum. يحافظ على معدلات تعلم تكيفية منفصلة لكل معلمة ويتضمن الزخم لتحمل تأثير التدرجات السابقة. غالبًا ما يُظهر Adam تقاربًا أسرع وأداءً قويًا في مهام مختلفة مقارنةً بـ RMSProp، وذلك بفضل التحديثات التكيفية والقائمة على الزخم.

2.4.2- التنبؤ بالحركة

يتضمن التنبؤ بالحركة باستخدام الشبكات العصبية التلافيفية (CNNs) تدريب شبكة للتنبؤ بالخطوة التالية الأكثر احتمالية بالنظر إلى الحالة الحالية للرقعة. تقوم CNN بمعالجة موضع الرقعة المدخلة وتنتج مخرجات تصنف الحركات المحتملة بناءً على احتمالاتها. يعزز هذا النهج قدرة CNN على التقاط الأنماط والعلاقات المكانية على رقعة الشطرنج.

3.4.2- تمثيل المدخلات

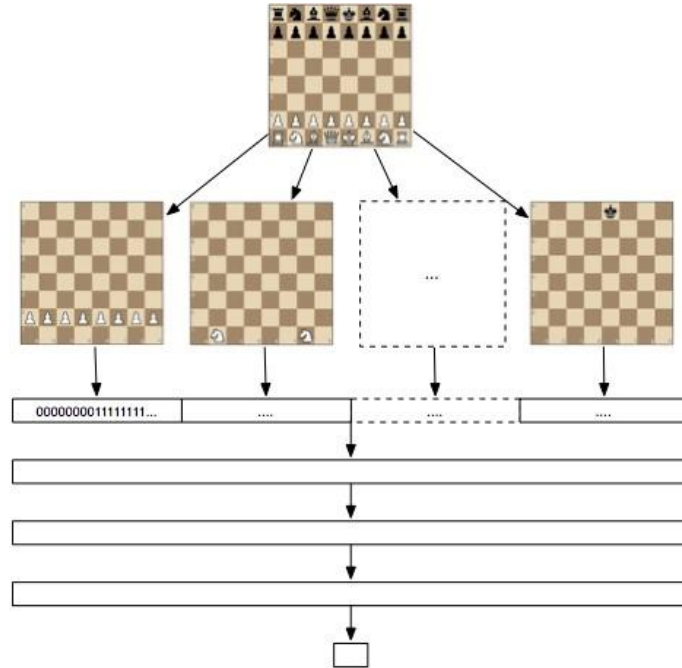
دخل CNN هو تمثيل لموضع رقعة الشطرنج الحالي. يتم ترميز كل مربع على الرقعة بمعلومات حول القطعة المحتلة وسماتها (اللون، النوع). يشكل هذا إدخالاً شبيهاً بالصورة حيث يتوافق كل مربع مع بكسل، وتمثل القنوات المختلفة جوانب مختلفة من القطع (مثل اللون والنوع).

4.4.2- هندسة الشبكات

تتكون بنية CNN من طبقات تلافيفية تليها طبقات متصلة بالكامل. تطبق الطبقات التلافيفية المرشحات على تمثيل المدخلات، وتكشف عن الأنماط والسمات المحلية. الطبقات المجمعة تهدف لتصغير خرائط المعالم، مما يقلل الأبعاد المكانية مع الاحتفاظ بالمعلومات المهمة. تقوم الطبقات المتصلة بالكامل بتجميع الميزات التي تم تعلمها وإجراء تنبؤات.

5.4.2- الإخراج والتدريب

خرج الـ CNN هو توزيع احتمالي على جميع الحركات الممكنة في الموضع المحدد. ترتبط كل حركة باحتمالية تشير إلى ثقة الشبكة في أن هذه الخطوة هي الخيار الأفضل. أثناء التدريب، يتم تغذية الشبكة بمجموعة بيانات لمواقع الشطرنج والتحركات المثلى المقابلة لها، تتم مقارنة تنبؤات الشبكة بالحركات الفعلية، ويتم استخدام تابع الخسارة لقياس خطأ التنبؤ. يتم تحديث معاملات الشبكة من خلال الانتشار العكسي backpropagation ودرجة الانحدار gradient descent لتقليل الخسارة.



شكل 15: تحليل موضع ما لرقعة شطرنج لقنوات كدخل للشبكة العصبونية

6.4.2- مزايا التنبؤ بالحركة CNNs

- التعرف على الأنماط: تتفوق شبكات CNN في التعرف على الأنماط المكانية، وهو أمر بالغ الأهمية لتحديد السمات والعلاقات المهمة على رقعة الشطرنج التي تؤثر على اختيار الحركة.
- سياق الالتقاط: يمكن لشبكات CNN التقاط كل من السياق المحلي والعام للموضع، مما يسمح لها بالنظر في ترتيب القطع والتهديدات المحتملة والفرص التكتيكية.
- التعلم الفعال: من خلال التعلم من مجموعة بيانات كبيرة لمواقف الشطرنج والتحركات المثلّية، يمكن لشبكات CNN تعميم الأنماط والاستراتيجيات الشائعة، وتمكينها من إجراء تنبؤات عن التحركات الجيدة.

7.4.2- القيود والاعتبارات

بينما تقدم CNN مزايا كبيرة للتنبؤ بالحركة في لعبة الشطرنج، إلا أنها قد تواجه بعض المشاكل في المواقف التي تتطلب فهمًا استراتيجيًا عميقًا أو تخطيطًا طويل المدى. قد يعانون أيضًا من المواقف التي تنطوي على تضحيات أو تحركات غير بديهية تتحدى الأنماط المحلية.

8.4.2- الأعمال السابقة

في مجال التنبؤ بحركة الشطرنج، استفادت الأبحاث الحديثة من الشبكات العصبية التلافيفية (CNNs) لتعزيز دقة تنبؤات الحركة. استخدمت الدراسة التي تحمل عنوان "التنبؤ بالحركات في الشطرنج باستخدام الشبكات العصبية التلافيفية" بنية CNN ثلاثية الطبقات للتنبؤ بالحركة في لعبة الشطرنج. تم تأطير المهمة على أنها مشكلة تصنيف من جزأين تتضمن اختيار القطعة واختيار الحركة. سجل محدد القطع CNN قطعًا بيضاء للحركة، بينما أنتج محدد الحركة CNN نتائج للوجهات المحتملة. من خلال القيام بذلك، تم تقليل تعقيد صفوف الحركات في الشطرنج بشكل كبير. تم تدريب الشبكات على مجموعة بيانات 20000 لعبة تضم أكثر من 245000 حركة من قبل لاعبين مع تصنيف ELO أعلى من 2000. والجدير بالذكر أن القطع التي صنعت حركات محلية أظهرت أداء أفضل، مما سلط الضوء على فعالية الطبقات التلافيفية في التعرف على الأنماط المحلية. [2]

وبالمثل، بحثت الورقة البحثية "توقع تحركات الشطرنج باستخدام الشبكات العصبية للتعلم العميق" التأزر بين التعلم العميق والتنبؤ بحركة الشطرنج. اقترح البحث استخدام شبكات CNN جنبًا إلى جنب مع خوارزمية Minimax التقليدية للتعرف على الأنماط والتكتيكات في لعبة الشطرنج. من خلال التدريب على مجموعة بيانات من 1.5 مليون حالة لوحة، مثلثة في بعد $8 \times 8 \times 14$ ، أظهر نموذج CNN دقة تقييم للوحة بنسبة 39.16٪. على الرغم من هذه الدقة الواعدة ضد محرك الشطرنج Stockfish، أقرت الدراسة أن تقييم النموذج نسبي بين حالات الرقعة المختلفة، مما يعكس الطبيعة المعقدة لاستراتيجيات

الشطرنج. يؤكد دمج شبكات CNN في تعلم الشطرنج على قدرتها على التعرف على الأنماط المعقدة والعمليات المنطقية، مما يوفر وسيلة جديدة لتحقيق المزيد من اللعب الاستراتيجي. [3]

5.2- التعلم المعزز العميق (Deep Reinforcement Learning)

التعلم المعزز العميق DRL هو نهج التعلم الآلي الذي يجمع بين الشبكات العصبية العميقة ومبادئ التعلم المعزز. يشمل التعلم المعزز على تدريب العميل Agent لتعلم الإجراءات المثلى من خلال التفاعل مع البيئة، وتعظيم الربح المكتسب. اكتسب DRL اهتمامًا كبيرًا لقدرته على تحقيق أداء رائع في المهام المعقدة من خلال التعلم من التجربة والخطأ.

1.5.2- عميل الشطرنج في DRL

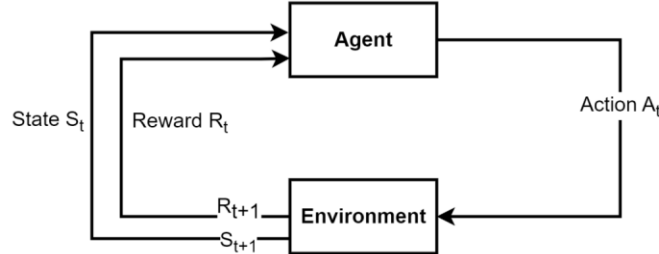
تم تطبيق DRL لتدريب عملاء الشطرنج من خلال السماح لهم بتعلم الاستراتيجيات والتكتيكات من خلال اللعب الذاتي وإشارات التعزيز. يتضمن تطبيق DRL على الشطرنج المكونات التالية:

- العميل: عميل الشطرنج يتفاعل مع رقعة الشطرنج كبيئة. يتخذ الإجراءات (الحركات) ويتلقى ردود على شكل مكافآت أو عقوبات ويتم ذلك بعد انتهاء لعبة كاملة.
- تمثيل الحالة: يتم تمثيل الحالة الحالية للرقعة الشطرنج كمدخلات لشبكة عصبية عميقة. يرصد هذا التمثيل مواقف القطع وأنواعها وسياق اللعبة.
- مساحة العمل: تتكون مساحة العمل من جميع الحركات القانونية الممكنة في الموضع الحالي. يختار العميل الحركة بناءً على السياسة التي تعلمتها الشبكة العصبية.
- شبكة السياسة: شبكة السياسة عبارة عن شبكة عصبية عميقة تأخذ الحالة الحالية كمدخلات وتخرج توزيعًا احتماليًا على التحركات المحتملة. يتعلم كيفية التنبؤ بالحركات التي تؤدي إلى نتائج إيجابية.
- شبكة القيمة: بالإضافة إلى شبكة السياسة، يتم استخدام شبكة القيمة لتقدير النتيجة المحتملة للعبة من موقع معين. هذا يساعد العميل على تقييم مدى الرغبة في المواقف المختلفة. [10]

2.5.2- آلية اللعب الذاتي:

- تعد آلية اللعب الذاتي إحدى السمات الرئيسية للعبة DRL في الشطرنج. يلعب العميل أعباء ضد نفسه، ويستكشف استراتيجيات مختلفة ويتعلم من تجاربه الخاصة. تتضمن العملية:
- الاستكشاف: يستكشف العميل الحركات المختلفة من خلال أخذ عينات للحركات من توزيع مخرجات شبكة سياسته.

- محاكاة اللعبة: يقوم العميل بمحاكاة الألعاب عن طريق إجراء الحركات وتسجيل المواضع والمكافآت الناتجة على اللوحة.
- تجربة إعادة التشغيل: يخزن العميل خبراته (الموضع، الحركة، المكافأة) في ذاكرة مؤقتة ويعيد تشغيلها أثناء التدريب. هذا يقلل من الارتباط بين التجارب المتتالية ويساعد على استقرار التعلم.



شكل 16: نموذج التعلم العميق الأساسي

3.5.2- نجاحات محركات الشطرنج القائمة على DRL (AlphaZero)

AlphaZero هو محرك الشطرنج الرائد القائم على DRL والذي طورته DeepMind، حقق نجاحا ملحوظا في لعب الشطرنج والألعاب الأخرى. أظهر AlphaZero قوة DRL في إتقان الألعاب المعقدة بمعرفة بشرية محدودة. تشمل إنجازاته:

- التعلم من الصفر: يتعلم فقط من اللعب الذاتي، دون أي بيانات مسبقة من صنع الإنسان أو الاستدلال.
- أداء خارق: حقق أداء خارقا، حيث هزم باستمرار أفضل محركات الشطرنج وأبطال العالم.
- التعميم: التعلم من اللعب الذاتي لـ AlphaZero يمكنه من تعميم الاستراتيجيات والتكيف مع أساليب اللعب المختلفة.
- نقل التعلم: تم تطبيق هندسة ومبادئ AlphaZero على مجالات أخرى، مما يعرض إمكانياته خارج الشطرنج.

4.5.2- القيود والاعتبارات

على الرغم من نجاحاتها، تتطلب محركات الشطرنج القائمة على DRL موارد حسابية مكثفة ووقت تدريب. قد يواجهون صعوبة في التعامل مع جداول النهايات وبعض المواقف الإستراتيجية المعقدة حيث تكون المعرفة البشرية الخبرة أمراً بالغ الأهمية.

5.5.2- الأعمال السابقة

في عالم التعلم المعزز والتعلم المعزز العميق في لعبة الشطرنج، ظهرت العديد من الأوراق البحثية المؤثرة لإعادة تشكيل مشهد اللعب واتخاذ القرار الاستراتيجي الذي يحركه الذكاء الاصطناعي.

تقدم الورقة "Giraffe: استخدام التعلم المعزز العميق للعب الشطرنج" محرك الشطرنج Giraffe، وهو تطبيق رائع للعب الذاتي لاكتساب المعرفة الخاصة بالمجال. على عكس الطرق السابقة التي كانت تؤدي في المقام الأول ضبط المعاملات على وظائف

التقييم المصنوعة يدويا، تستخدم Giraffe التعلم التعزيزي العميق لاستخراج الميزات تلقائيا والتعرف على الأنماط. تتنافس وظيفة التقييم المدربة في Giraffe على محركات الشطرنج الحديثة، وتحقق أداءً مشابهاً مع الاعتماد على الحد الأدنى من المدخلات المصنوعة يدويا. [4]

يقدم "إتقان الشطرنج والشوجي من خلال اللعب الذاتي باستخدام خوارزمية التعلم التعزيزي العامة" خوارزمية AlphaZero، التي حققت أداءً خارقاً في الشطرنج والشوجي والغو من خلال اللعب الذاتي. من خلال البدء من اللعب العشوائي والتعلم من قواعد اللعبة فقط، تجاوز AlphaZero برامج بطل العالم في هذه الألعاب في غضون 24 ساعة. تعرض الورقة قدرة AlphaZero الاستثنائية على التكيف وقدرتها على إتقان الألعاب المعقدة دون الاعتماد على المعرفة الخاصة بالمجال أو الميزات التي صنعها الإنسان. [5]

توضح هذه الأوراق التأثير الرائد للتعلم المعزز وخوارزميات التعلم المعزز العميق في الشطرنج والألعاب الإستراتيجية الأخرى. لم تحقق هذه الأساليب أداءً خارقاً فحسب، بل أظهرت أيضاً القدرة على تعلم الأنماط والاستراتيجيات المعقدة من خلال اللعب الذاتي، مما أحدث ثورة في تطوير محركات الألعاب التي تعتمد على الذكاء الاصطناعي.

6.2- شجرة بحث مونت كارلو (Monte Carlo Tree Search)

MCTS عبارة عن خوارزمية بحث إرشادية تُستخدم على نطاق واسع في ألعاب الذكاء الاصطناعي لاتخاذ قرارات مستنيرة من خلال محاكاة واستكشاف مسارات اللعبة المحتملة. وتكون فعالة بشكل خاص في المجالات ذات مساحات البحث الكبيرة والمعقدة، مثل الشطرنج. تتكون MCTS من أربع خطوات رئيسية:

أكبر مشكلة في خوارزميات minimax التي تستخدم حد العمق هي الاعتماد على تابع التقييم. إذا قام تابع التقييم بإعطاء تقديرات غير صحيحة أو دون المستوى الأمثل، فإن الخوارزمية ستقترح حركات سيئة. يحاول مطورو محركات الشطرنج المعاصرة مثل Stockfish باستمرار تحسين هذه الوظيفة.

يمكن أن يؤدي استخدام تقليد ألفا بيتا أيضاً إلى حدوث بعض المشكلات. لنفترض أنه بإمكان اللاعب التضحية بقطعة ما للحصول على ميزة كبيرة لاحقاً في اللعبة. قد تقطع الخوارزمية الفرع ولا تستكشف هذا الخط الفائق أبداً، لأنها تعتبر التضحية مركزاً خاسراً. يعد البحث في شجرة مونت كارلو (MCTS) خوارزمية بحث يمكن استخدامها للتخفيف من حدة هذه المشكلات. تقرب MCTS قيمة الموضع عن طريق إنشاء شجرة بحث باستخدام الاستكشاف العشوائي لأكثر الحركات الواعدة.

لإنشاء شجرة البحث هذه لموضع معين، ستقوم MCTS بتشغيل الخوارزمية التالية مئات المرات، وتتألف من أربع خطوات. كل تنفيذ للخوارزمية يسمى محاكاة MCTS. يجب التنويه على عدم الخلط بين هذا وبين الخطوة الثالثة من الخوارزمية، والتي تسمى أيضاً المحاكاة.

1.6.2- الاختيار (Selection)

بدءاً من العقدة الجذرية Root Node، يتم تحديد عقدة فرعية بناءً على صيغة معينة عادة ما يتم وضعها حسب المسألة. تستخدم معظم تطبيقات MCTS بعض المتغيرات من الحد الأعلى للثقة Upper Confidence Bound (UCB) الاستمرار في تحديد العقد حتى يتم الوصول إلى عقدة لم تتم زيارتها مسبقاً (غير موسعة). نسميها بالعقدة الورقية. إذا كانت العقدة الجذرية هي عقدة ورقية، تنتقل على الفور إلى الخطوة التالية.

2.6.2- التوسع (Expansion)

إذا كانت العقدة المختارة هي عقدة نهائية (تنتهي اللعبة)، فيتم المتابعة إلى خطوة الانتشار العكسي (الخطوة الرابعة). عندما لا تكون العقدة نهائية، فيتم إنشاء عقدة فرعية لكل حركة محتملة من هذا الموضع.

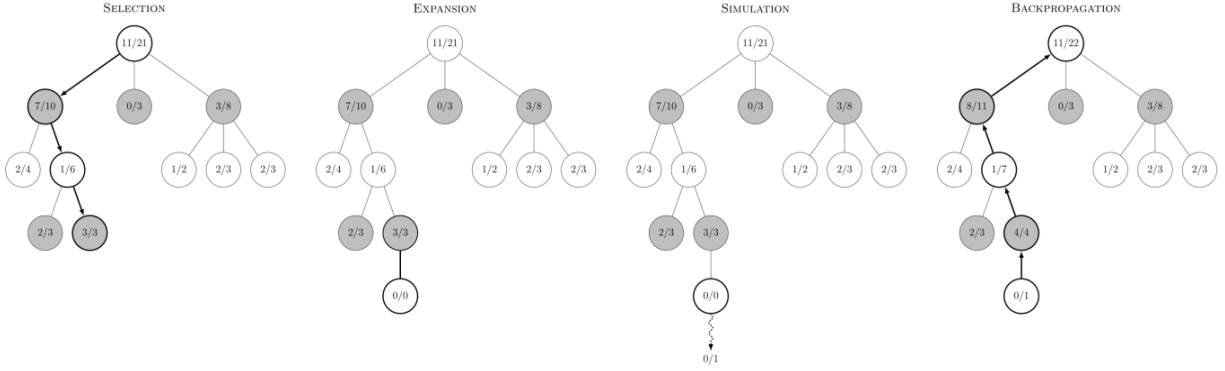
3.6.2- المحاكاة (Simulation)

في هذه المرحلة يتم اختيار عقدة فرعية عشوائية تم توسيعها في الخطوة السابقة. باختيار الحركات العشوائية فقط، ثم نقوم بمحاكاة بقية اللعبة من موضع تلك العقدة الفرعية.

4.6.2- الانتشار العكسي (Backpropagation)

يتم إعادة نتيجة المحاكاة لأعلى الشجرة. وتقوم كل عقدة بتعقب عدد المرات التي تمت زيارتها، وعدد المرات التي أدت فيها إلى الفوز.

بالنسبة للشطرنج، هذه الخوارزمية غير فعالة للغاية، بسبب ضرورتها محاكاة لعبة شطرنج كاملة في الخطوة الثالثة من كل محاكاة. لحساب قيمة مركز واحد فقط، يجب أن يكون هناك المئات من هذه المحاكاة للحصول على تقدير جيد. لذلك، يجب اختيار صيغة الاختيار بعناية لأنه من المهم اختيار العقد بطريقة متوازنة بين الاستكشاف والاستغلال.



شكل 17: خطوات تطبيق خوارزمية MCTS

5.6.2- الأعمال السابقة

في مجال تطبيق Monte Carlo Tree Search (MCTS) المطبق على لعبة الشطرنج، ساهمت الأوراق البحثية التالية في تقديم رؤى وتطورات قيمة:

تقدم الورقة البحثية "مونت كارلو الشطرنج" MCC، وهو محرك شطرنج قائم على UCT مصمم لتقييم أداء MCTS في لعبة الشطرنج. من خلال التعديلات الإستراتيجية التي تعزز دقة المحاكاة، حقق تطبيق MCC الأساسي تحسين Elo بنحو 864. على الرغم من هذا التحسين، لا يزال MCC يواجه تحديات في التنافس مع برامج الشطرنج المبنية على Minimax وتجنب مصائد البحث. يلقي هذا العمل الضوء على التعقيدات والتحسينات المحتملة لـ MCTS للعب الشطرنج. [6]

يتناول "العب الشطرنج على مستوى وأسلوب مرغوب فيه للإنسان" تكامل MCTS واتخاذ القرارات الشبيهة بالإنسان لتصميم عملاء الشطرنج الذي يتماشى مع تفضيلات الإنسان. مع الاعتراف بأن اللاعبين البشريين يفضلون الخصوم الذين لديهم أساليب لعب متوافقة، تقترح الورقة نهجاً مبتكراً. من خلال دمج نظرية صنع القرار لدى لاعبي الشطرنج مع خوارزميات MCTS الحديثة، يحقق الوكلاء الذين تم إنشاؤهم من خلال هذه الطريقة تقييمات Elo المرغوبة. تستخدم الدراسة كلاً من التحليلات الكمية والنوعية لإثبات فعالية النهج. في اختبار مستوحى من Turing، يقوم الوكلاء بأداء مستوى لا يمكن تمييزه عن اللاعبين البشريين، مما يبرز قدرة MCTS على سد الفجوة بين أنماط لعب الإنسان والعميل. [7]

تسلط هذه الأوراق بشكل جماعي الضوء على تطبيق MCTS في سياق الشطرنج. يستكشفون التحديات والتحسينات وإمكانية مواءمة وكلاء الشطرنج المدفوع بالذكاء الاصطناعي مع تفضيلات اللعب البشري، مما يساهم في التطور المستمر لأسلوب اللعب المدعوم بالذكاء الاصطناعي واتخاذ القرارات الاستراتيجية في الشطرنج.

7.2- شبكات الذاكرة طويلة المدى (LSTM)

شبكات الذاكرة طويلة المدى (LSTM) هي نوع متخصص من الشبكات العصبية المتكررة (RNN) المصممة لنمذجة والتقاط التبعيات المتسلسلة في البيانات. على عكس الشبكات العصبية التقليدية ذات التغذية الأمامية، فإن LSTMs مزودة بخلايا ذاكرة يمكنها تخزين المعلومات واستردادها على فترات زمنية متفاوتة. تجعل هذه البنية الفريدة من LSTMs فعالة بشكل خاص للمهام التي تتضمن تسلسلا، مثل معالجة اللغة الطبيعية، وتحليل السلاسل الزمنية، وتحليل تسلسل حركات الشطرنج بشكل مناسب.

يمكن تطبيق LSTMs في مجال الشطرنج من خلال توظيفها لتحليل تسلسل الحركات التي تتم أثناء الألعاب. من خلال ترميز تسلسلات الحركة كتسلسلات إدخال لشبكة LSTM، بحيث يمكن للنموذج معرفة العلاقات المعقدة بين الحركات المختلفة وتحديد الأنماط والتكتيكات والفروق الإستراتيجية الدقيقة. إن قدرة الشبكة على تذكر واستخدام المعلومات من التحركات السابقة، حتى مع تقدم التسلسل، تمكنها من التنبؤ بالحركات المثلى التي تتوافق مع الاستراتيجيات الناجحة.

تتمثل إحدى المزايا المميزة لـ LSTM في قدرتها على التقاط السياق والمعلومات الإستراتيجية من سلسلة من التحركات. في سياق الشطرنج، هذا يعني أن LSTM لا يمكنها فقط تحليل التحركات الفردية ولكن أيضًا تحديد الاستراتيجيات الشاملة التي تتكشف على عدة أدوار. تمكن هذه القدرة الشبكة من فهم التطور الاستراتيجي للعبة، والتنبؤ بنوايا اللاعب المحتملة، وتكييف تنبؤات حركتها وفقا لذلك. من خلال ترميز كل من السياق المحلي والعام، تقدم LSTM منظورا دقيقا لمواضع الشطرنج ومساراتها المحتملة.

1.7.2- الأعمال السابقة

في مجال تطبيق شبكات الذاكرة طويلة المدى (LSTM) في لعبة الشطرنج، تم إحراز تقدم كبير، كما يتضح من الأوراق البحثية التالية:

في دراسة بعنوان "الشطرنج كبيانات متسلسلة في توقع نتيجة مباراة الشطرنج باستخدام التعلم العميق مع تمثيلات رقعة الشطرنج المختلفة"، استكشف الباحثون مفهوم التعامل مع حركات الشطرنج على أنها بيانات متسلسلة. من خلال تصميم نموذج باستخدام طبقات LSTM حصريا، كانوا يهدفون إلى الكشف عن الطبيعة المتسلسلة للحركات في مباريات الشطرنج. أثبتت النتائج من نموذجهم صحة هذا النهج، وسلطت الضوء على قدرته على التقاط أنماط متسلسلة. قدم الباحثون أيضًا بنية نموذجية جديدة، تجمع بين طبقات LSTM وأنواع بيانات متعددة، بما في ذلك حركات الشطرنج والبيانات الوصفية للعبة. حقق هذا النموذج الهجين دقة تصنيف ملحوظة تقترب من 69%. ومن المثير للاهتمام أن الورقة غامر في إجراء تحليل مقارن لتمثيلات رقعة الشطرنج، موضحة أنه على الرغم من أن مدخلات الصور النقطية تحمل معلومات نظرية أقل من المدخلات الجبرية، إلا أنها قدمت نتائج أفضل في سياق تدريب الشبكة العصبية. [14]

مساهمة أخرى ذات صلة هي العمل الذي يحمل عنوان "مشاهدة نموذج تعلم لغة الشطرنج"، والذي درس كيف تكتسب نماذج اللغة القائمة على المحولات قواعد الشطرنج من سجلات اللعبة النصية. بحث الباحثون في العلاقة بين قدرة النموذج، وحجم بيانات التدريب، ونجاح تعلم نموذج اللغة، باستخدام مقاييس خاصة بالشطرنج. أظهرت النتائج التي توصلوا إليها أن دمج عدد أكبر من الألعاب التدريبية أدى إلى تحسن كبير في نتائج التعلم خلال نفس فترة التدريب. والجدير بالذكر أنه على الرغم من أن حجم النموذج لم يظهر تأثيراً واضحاً، فقد كشف هذا التحقيق عن تناقض بين مقاييس تقييم نموذج اللغة التقليدية، مثل الدقة التنبؤية والارتباك، ونجاح التعلم الخاص بالشطرنج. بالإضافة إلى ذلك، ألقت الدراسة الضوء على كيفية تخزين النماذج المدربة للمعلومات حول حالة اللوحة ضمن عمليات تنشيط مجموعة الخلايا العصبية وكيف يؤثر تسلسل الحركات السابقة على توليد الحركات الجديدة. [13]

تؤكد الأوراق البحثية هذه مجتمعة على تطبيق شبكات LSTM في فك رموز الطبيعة المتسلسلة لتحركات الشطرنج واكتساب قواعد الشطرنج بواسطة نماذج اللغة. من خلال هذه المساهمات، تم تطوير إمكانات النماذج المستندة إلى LSTM لالتقاط وتعلم أنماط متسلسلة معقدة في لعبة الشطرنج بشكل كبير.

8.2- المحركات Leela Chess Zero, AlphaZero, Stockfish

في عالم محركات الشطرنج المتقدمة، يكشف هذا القسم عن ثلاثة لاعبين محوريين أعادوا تعريف مشهد اللعب الذي يحركه الذكاء الاصطناعي: Leela Chess Zero و AlphaZero و Stockfish. تمثل هذه المحركات ذروة التقنيات المتطورة، من التعلم المعزز العميق إلى خوارزميات البحث التقليدية. يجلب كل محرك منهجاً فريداً للشطرنج، حيث يعرض اندماج الاستراتيجيات التي صنعها الإنسان والتعلم المستقل، مما يؤدي إلى حقبة جديدة من اتخاذ القرارات الإستراتيجية في اللعبة.

1.8.2 Stockfish

هو عبارة عن أقوى محرك شطرنج متاح للجمهور منذ فترة طويلة. فهو محرك مجاني المصدر يتم تطويره حالياً بواسطة المجتمع بأكمله. يعتمد Stockfish على محرك الشطرنج الذي أنشأه تورند رومستاد عام 2004 والذي تم تطويره من قبل ماركو كوستالبا عام 2008. ومن المؤسسين أيضاً جونا كيسكي وغاري لينسكوت.

إنه ليس من أقوى محركات الشطرنج المتاحة فقط بل وأيضاً من أسهلها للوصول والحصول عليه أيضاً. إنه متاح بسهولة على العديد من أنظمة التشغيل الأساسية، بما في ذلك Windows و Mac و Linux و Android و iOS.

إن إنجازاته مثيرة للإعجاب أكثر من إنجازات أي محرك شطرنج آخر. لقد فاز بثماني بطولات لأفضل محركات الشطرنج (TCEC) حتى عام 2020. حيث سيطر أيضاً على بطولة chess.com لشطرنج الحواسيب والمحركات منذ عام 2018، حيث فاز في أول ست بطولات.

أثبت Stockfish نفسه بقوة باعتباره أقوى محرك شطرنج في العالم قبل عام 2017، ولهذا السبب اهتز عالم الشطرنج من صميمه عندما خسر مباراة أمام محرك الشبكة العصبية والذي يسمى AlphaZero، أدت خسارته أمام AlphaZero إلى تطوير محركات أخرى تستخدم الشبكات العصبية (أبرزها LeelaZero).

على الرغم من احتفاظ Stockfish بموقعه على رأس قائمة محركات الشطرنج، إلا أن محركات التي تستخدم الشبكات العصبية كانت تقترب أكثر فأكثر من قوته. في أيلول 2020، تم إصدار Stockfish12، وتم الإعلان عن أن Stockfish قد استوعب مشروع Stockfish + NNUE (الشبكة العصبية القابلة للتحديث Efficiently updatable neural network). حيث تم تحسين قدرته على فهم المواضع الشطرنجية.

اعتباراً من تشرين الأول 2020، يعد المحرك الأعلى تقييماً وفقاً لقائمة تصنيف الكمبيوترات الشطرنجية (CCRL) مع تصنيف 3514، حيث إنه المحرك الوحيد الذي حصل على تصنيف أعلى من 3500. ووفقاً لقائمة تصنيف جمعية الحواسيب السويدية للشطرنج في تموز 2020 (SSDF)، فإن Stockfish9 كان في المرتبة الثالثة، و Stockfish10 في المرتبة الثانية، وجاء

وStockfish11 في المرتبة الأولى مع تصنيف 3558. حيث أن احتلاله للمراكز الثلاثة الأولى بإصدارات مختلفة أمر مثير للإعجاب.

AlphaZero -2.8.2

AlphaZero هو محرك شطرنج تم تطويره بواسطة شركة DeepMind المتخصصة بمجال الأبحاث والذكاء الاصطناعي والتي استحوذت عليها جوجل. إنه برنامج حاسوبي وصل إلى مستوى لعب لا يمكن تصوره وقام بتدريب شبكاته العصبية باستخدام التعلم المعزز واللعب الذاتي فقط. بمعنى آخر، تم إعطاؤه قواعد اللعبة فقط ثم لعب ضد نفسه عدة ملايين من المرات (44 مليون مباراة في الساعات التسع الأولى، وفقًا للشركة). [12]

يستخدم AlphaZero شبكاته العصبية لإجراء تقييمات متقدمة للغاية للمواقف الشطرنجية، مما يلغي الحاجة إلى تحليل أكثر من 70 مليون موقف شطرنجي في الثانية (كما يفعل Stockfish). وفقًا للشركة المطورة، فقد وصل إلى المعايير اللازمة لهزيمة Stockfish في غضون أربع ساعات فقط.

يتم تشغيل AlphaZero على أجهزة مخصصة يشار إليها باسم كمبيوترات جوجل العملاقة على الرغم من أن الشركة المطورة قد أوضحت منذ ذلك الحين أنه يعمل على أربع وحدات معالجة Tensor Processing Units (TPUs) في مبارياته.

للأسف، محرك الشطرنج هذا غير متاح للاستخدام من قبل الجمهور بأي شكل من الأشكال. حيث أدت نتائج التحديات المذهلة بينه وبين Stockfish إلى إنشاء العديد من محركات وبرامج الشطرنج مفتوحة المصدر وتستخدم تقنية الشبكات العصبية. [11]

(Lc0) Leela Chess Zero -3.8.2

معروف أيضًا باسم Lc0 وLCZero وLeela هو محرك شطرنج مفتوح المصدر يعتمد على الشبكة العصبية (NN). تم الإعلان عن مشروع Lc0 في أوائل عام 2018، وقاد جاري Gary Linscott (مطور Stockfish) التطوير. نظرًا لطبيعته المجانية والمفتوحة المصدر، يمكن تشغيله على العديد من الأنظمة الأساسية، بما في ذلك Windows وMac وLinux وAndroid وUbuntu. إن Lc0 هو أقوى محرك NN متاح للجمهور.

Lc0 مستوحى بقوة من مشروع AlphaZero التابع لشركة DeepMind وتعلم اللعبة بنفس الطريقة. على عكس محركات الشطرنج التقليدية، لم تُمنح Leela سوى قواعد لعبة الشطرنج وأصبحت قوية بشكل لا يصدق باستخدام التعلم المعزز من اللعب الذاتي المتكرر اعتبارًا من عام 2020 لعبت أكثر من 300 مليون مباراة ضد نفسها.

وفقًا لقائمة تصنيف شطرنج الحواسيب لشهر أيلول 2020 (CCRL)، فإن Leela هي ثاني أعلى محرك شطرنج تصنيفًا في العالم مع تصنيف 3462، خلف Stockfish بقليل.

Leela مكتوب بلغة C++، وقد نجح في اللعب بمستوى مشابه لأفضل إصدار حالي من Stockfish. نظرًا لأن Ic0 مشروع مدفوع بالجمع، يمكن للمتطوعين المساعدة في إنشاء ألعاب تدريبية من خلال اللعب الذاتي باستخدام أجهزة الكمبيوتر الخاصة بهم. هذا جعل من الممكن إضافة الملايين من ألعاب الشطرنج في شبكته. [9]



شكل 18: Leela Chess Zero VS AlphaZero VS Stockfish

9.2- مجموعة البيانات (Datasets)

لدينا مجموعتين من البيانات، الأولى تحتوي ألعاب شطرنج كاملة على شكل ملف Portable Game Notation PGN والمجموعة الثانية تحتوي على أغاز شطرنج مع حلولها وبعض التفاصيل عنها على شكل ملف Comma-Separated Values CSV.

1.9.2- مجموعة بيانات ألعاب الشطرنج

تحتوي المجموعة على ألعاب شطرنج كاملة لأشخاص الـ ELO الخاص بهم مرتفع نسبيًا بحيث يمكن اعتماد الألعاب هذه لتدريب نظام قادر على لعب الشطرنج على مستوى متقدم، حجم مجموعة البيانات تقريبًا 33 ميغابايت وهو قليل نسبيًا مقارنة بحجم مجموعات البيانات الأخرى ولكن بما أنه على شكل PGN فهو يحوي مجموعة الحركات لكل لعبة على شكل رموز تدل على نوع القطعة والخلية التي تحركت إليها سنشرح بتفصيل أكبر عنها في الأقسام التالية، وعدد الألعاب في هذه المجموعة تقريبًا 40 ألف لعبة. [16]

1.1.9.2- محتوى مجموعة البيانات

تحتوي المجموعة على العديد من الألعاب، كل لعبة مكتوبة على شكل PGN بحيث نجد أولاً معلومات عن اللعبة ومن ثم الحركات لكامل اللعبة، أهم المعلومات عن اللعبة مثل اسم اللاعبين والـ ELO الخاص بكل منهما ونظام الوقت والتاريخ وعدد الحركات والنتيجة ورمز الافتتاحية، تليها مجموعة من الحركات مرقمة كحركات كاملة (هي حركتين واحدة للأبيض والأخرى للأسود)، على شكل تدوين جبري Simple Algebraic Notation SAN والذي هو على الشكل التالي:

كل حركة تبدأ بحرف يعبر عن اسم القطعة (Q, K, N, B, R) ويلها محرفين رمز العمود ورقم السطر الخانة الذي سوف تتحرك إليه هذه القطعة، في حالة القطعة كانت عبارة عن بيدق فلا داعي لوضع P وتحمل، في حالة كانت الحركة فيها أسر توضع x بين المحرف الأول والمحرفين اللاحقين، أما في حالة كانت القطعة التي تأسر هي بيدق فيوضع محرف يمثل رمز العمود الذي كان به البيدق قبل الحركة، في حالة كان هناك تضارب بين قطعتين من نفس النوع وبإمكان كليهما الذهاب لنفس الخانة فيوضع محرف إضافي بين المحرف الأول والمحرفين اللاحقين للتفريق وتوضيح أي قطعة تتحرك فمثلا إن كان هناك رُتْخَان يستطيعان الذهاب لنفس الخانة إن كانا على نفس السطر يوضع محرف رمز العمود الذي جاءت منه القطعة والعكس بالعكس، وفي حالة الترقية يوضع رمز = ومحرف يليه يعبر عن نوع القطعة المراد الترقية إليها، في حالة الكش يوضع + بعد الحركة وفي حالة الكش مات يوضع # بعد الحركة، في حالة التبييت على جناح الملك نرمز لها بالرمز o-o والتبييت على جناح الوزير نرمز لها على الشكل o-o-o.

كما نجد سابقا أنه ليست جميع الحركات يمكن أن تكون صحيحة وقانونية، وفي حال كانت كذلك يبقى هناك مشكلة معرفة من أين أتت هذه القطعة قبل الحركة، في حالة عدم متابعة اللعبة من بدايتها نجد هناك غموض واضح ولا يمكن معرفته إلا بتطبيق كافة الحركات من بداية اللعبة حتى اللحظة الحالية، ولتخطي ذلك هناك ترميز آخر غير متعلق بالقطعة وهو ترميز UCI وهو عبارة عن 4 أو 5 محارف تقسم على الشكل، أول محرفين (رمز العمود ورقم السطر) يعبران عن الموقع السابق للقطعة التي تتحرك والمحرفين التاليين يعبران عن موقع القطعة بعد الحركة والمحرف الخامس يمكن أن يكون موجود في حالة ترقية ويعبر عن رمز القطعة المراد الترقية لها، لا يوجد ترميز للأسر، وترميز التبييت هو حركة الملك من موقعة السابق إلى موقعه الجديد (الحالة الوحيدة التي يمكن للملك التحرك خطوتين بها).

```
[Event "FICS rated lightning game"]
[Site "FICS freechess.org"]
[FICSGamesDBGameNo "371408277"]
[White "theblob"]
[Black "IFDThor"]
[WhiteElo "1846"]
[BlackElo "2160"]
[BlackIsComp "Yes"]
[TimeControl "60+0"]
[Date "2015.02.01"]
[Time "02:49:00"]
[WhiteClock "0:01:00.000"]
[BlackClock "0:01:00.000"]
[ECO "E50"]
[PlyCount "64"]
[Result "0-1"]
```

1. d4 Nf6 2. c4 e6 3. Nc3 Bb4 4. Nf3 b6 5. e3 O-O 6. Bd3 Nc6 7. O-O

شكل 19: مثال عن شكل بيانات لعبة مرمزة في ملف PGN

1. d4 Nf6 2. c4 e6 3. Nc3 Bb4 4. Nf3 b6 5. Bf4 O-O 6. e3 Ne4
7. Bd3 Nxc3 8. bxc3 Bxc3+ 9. Nd2 Bxa1 10. Qxa1 Nc6 11. O-O d6 12. d5 e5
13. dxc6 exf4 14. exf4 Re8 15. Ne4 Qh4 16. g3 Qg4 17. f3 Qh5 18. Ng5 h6
19. Ne4 f5 20. Nc3 a5 21. Nd5 Ra7 22. Qd4 Qh3 23. a4 Kh7 24. Bc2 Re2
25. Rf2 Re1+ 26. Rf1 Qxf1# {White checkmated} 0-1

شكل 20: لعبة كاملة مرمزة بترميز SAN

1. d2d4 g8f6 2. c2c4 e7e6 3. b1c3 f8b4 4. g1g3 b7b6 5. c1f4 e8g8 6. e2e3 f6e4
7. f1d3 e4c3 8. b2c3 b4c3 9. f3d2 c3a1 10. d1a1 b8c6 11. e1g1 d7d6 12. d4d5 e6e5
13. d5c6 e5f4 14. e3f4 f8e8 15. d2e4 d8h4 16. g2g3 h4g4 17. f2f3 g4h5 18. e4g5 h7h6
19. g5e4 f7f5 20. e4c3 a7a5 21. c3d5 a8a7 22. a1d1 h5h3 23. a2a4 g8h7 24. d3c2 e8e2
25. f1f2 e2e1 26. f2f1 h3f1

شكل 21: نفس اللعبة في الشكل (20) السابق ولكن بترميز UCI

كل لعبة تحتوي على مجموعة حركات وكل حركة تقابل رقعة معينة لذلك كل ثنائية رقعة حركة هي دخل للشبكة العصبونية للتدريب لذلك فإن حجم مجموعة البيانات هو كبير جداً لأن متوسط طول لعبة الشطرنج يكون 80 حركة (40 حركة كاملة) أي كل لعبة لدينا 80 ثنائية للتدريب.

2.9.2- مجموعة بيانات ألغاز الشطرنج

تحتوي المجموعة على ألغاز شطرنج محملة من موقع Lichess من تبلغ حجم المجموعة 160 ميغابايت مضغوطة بالصيغة (.csv.zst). بحيث تحتوي على 3.3 مليون لغز مع حلولها تم فلترة هذه الألغاز على معايير معينة والإبقاء على 140 ألف لغز فقط. [15]

1.2.9.2- محتوى مجموعة البيانات:

تحتوي المجموعة على العديد من الألغاز في ملف csv، كل لغز يحتوي على مُعرّف، وتدوين يحتوي على كافة تفاصيل الموضع الحالي للرقعة الخاصة باللغز ويسمى Forsyth-Edwards Notation (FEN)، ويحتوي على الحركات المطلوبة لحل اللغز، وتصنيف للغز حسب مستواه ELO، وتشتت التصنيف، وشعبية اللغز على الموقع يتراوح بين 100 و -100، وعدد مرات لعب اللغز، وتوصيف للغز Theme، ورابط اللغز.

يبدأ اللغز بعد تجهيز الرقعة بالتدوين FEN ومن ثم لعب أول حركة من حركات الحل على الرقعة.

NR5r/3P1rpk/2n2P1p/pnP1Qq2/P1RPbP1N/BP1B1pKP/p1pp3p/b7 w - - 0 1

شكل 22: مثال عن FEN

3-الفصل الثالث

الحل المقترح

في هذا الفصل سيتم عرض نظرة شاملة حول الطريقة المقترحة لتحسين أداء برامج الشطرنج باستخدام التقنيات الذكاء الاصطناعي. سيتم تقديم تفاصيل عن الخطوات والمفاهيم التي يستند إليها الحل.

1.3- مقدمة

تأتي الابتكارات والتطورات في عالم التكنولوجيا لتحديث ثورات في مجموعة متنوعة من المجالات، ومن بين تلك المجالات تبرز ألعاب الشطرنج والذكاء الاصطناعي. تعد ألعاب الشطرنج من أقدم ألعاب الاستراتيجية التي لا تزال تحظى بشعبية واسعة حتى اليوم. تحتضن هذه اللعبة تعقيدات استراتيجية تتطلب فهماً عميقاً للتحركات والخطط الممكنة، مما يمثل تحدياً مثيراً لعشاق هذا العالم.

مع التطور السريع للذكاء الاصطناعي، ظهرت فرص جديدة لاستخدامه في مجال الشطرنج. إذ يمكن للخوارزميات والنماذج العصبية أن تقدم تحسينات كبيرة في قدرة الحواسيب على فهم الأوضاع الشطرنجية واتخاذ القرارات الأمثل. هذا ما يجسده الحل المقترح، الذي يقوم على استخدام تقنيات التعلم العميق لتحسين أداء البرامج الشطرنجية.

سيتم في هذا النص مناقشة الحل المقترح والطريقة التي تعتمد عليها البرامج الشطرنجية القائمة على الذكاء الاصطناعي لاتخاذ القرارات والتحسينات التي يمكن أن تقدمها في هذا السياق. سنلقي الضوء على الخوارزميات والمراحل التي يتضمنها الحل، مشددين على الفوائد المحتملة لاستخدام هذا النهج في تطوير أنظمة شطرنج أكثر قوة وفعالية. سنستعرض معاً التفاصيل العميقة لهذا النهج وكيفية تطبيقه العملي في سياق لعبة الشطرنج.

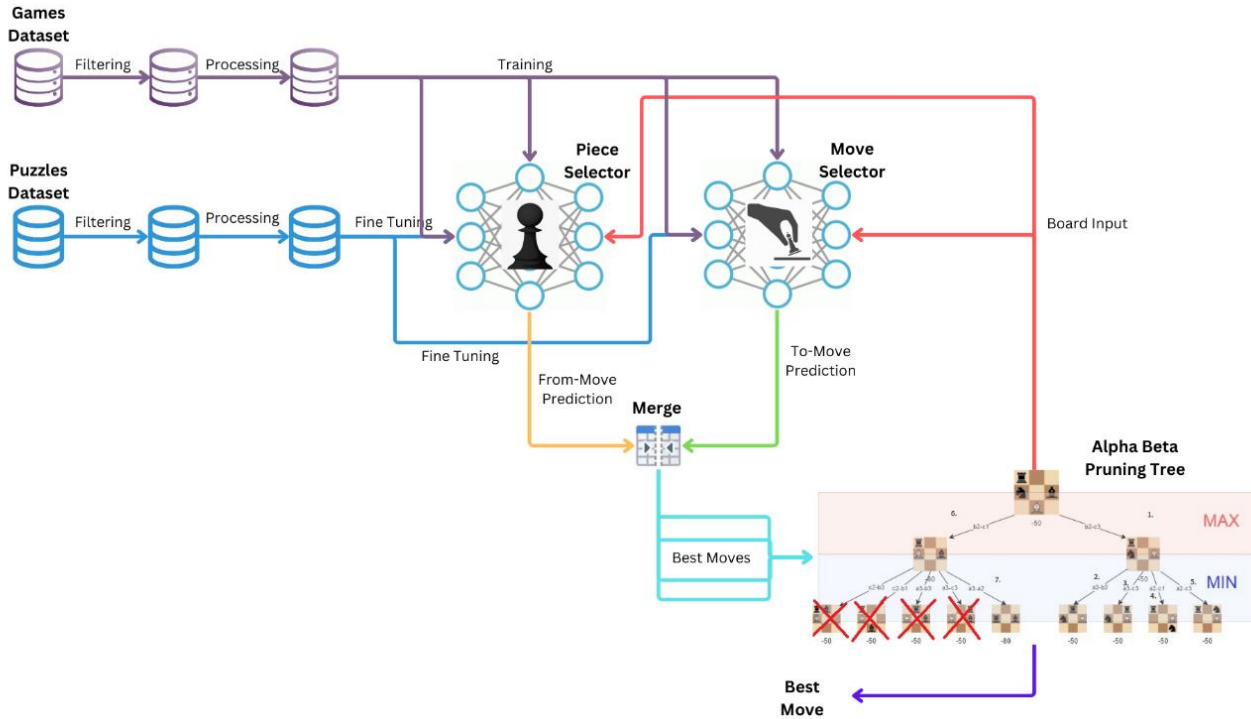
2.3- الخط الرئيسي لنهج العمل

أولاً تم تنقية وفلترة مجموعات البيانات والانتقاء منها على أسس معينة تساعد في تعلم أفضل للنظام، ثم تم إعادة صياغة وتشكيل هذه البيانات بمعالجتها وتحويل الحركات الموجودة فيها لتشكيل الرقع وتحويلها إلى مصفوفات مناسبة لتكون كدخل لتدريب الشبكات العصبونية، ثم تم بناء 7 شبكات عصبونية تلافيفية، الهدف منها هو استخراج الحركة الأفضل من رقعة معينة، تهدف الشبكة الأولى إلى تحديد القطعة المراد تحريكها والشبكات الـ 6 الأخريات تهدف إلى تحديد الخلية المراد تحريك تلك القطعة إليها، حيث تكون كل شبكة مسؤولة عن نوع معين من القطع.

بعد التدريب يتم استخلاص أفضل الحركات عن طريق دمج خرج كل من الشبكات السابقة، حيث كل شبكة تعطي توزيع احتمالي على كافة خلايا الشبكة، بعد اقتصاص الخلايا الغير صالحة بتصغير احتمالاتها، يتم ضرب الاحتمالين خرج الشبكة الأولى مع خرج أحد الشبكات الثانية حسب القطعة، والثنائيات (الخلية من الشبكة الأولى، الخلية من الشبكة الثانية) التي تعطي أكبر احتمالاً تكون من أفضل الحركات المقترحة بالنسبة للشبكات.

يتم استخلاص عدد معين من تلك الحركات الأفضل ولمعرفة الحركة الأفضل بينها نقوم بالاستفادة من شجرة تقليم ألفا بيتا، بحيث كل عقدة تمثل الموضع الحالي للرقعة، وتمثل الوصلات كافة الحركات الممكنة لهذا الموضع، ولكن نكتفي بأخذ الحركات الأفضل المقترحة من قبل النموذج السابق ليحدد عامل التفرع لهذه الشجرة، يتم البحث في تلك الشجرة إلى عمق معين عن الموضع الأفضل حسب تابع تقييم معين، ثم تحديد الوصلة الأولى التي أوصلت إلى هذه العقدة الطرفية وهي تكون الحركة الأفضل، يتم تحديد معامل التفرع وعمق الشجرة عن طريق التجريب لمعرفة القيم التي تكون الدقة فيها كافية بشكل مناسب باستهلاك غير كبير نسبياً بالنسبة للزمن.

تم اعتماد محرك الشطرنج Stockfish للقيام بعملية التقييم السابقة للعقد بحيث يعطى حداً معيناً للزمن للقيام بهذا التقييم، بازدياد الزمن تزداد دقة التقييم وبها تزداد دقة تحديد الحركة الأفضل ولذلك تم اعتبار هذا الحد من المعاملات التي يتم بها المفاضلة بين الزمن والدقة.



شكل 23: الخطة الرئيسية للحل

3.3- فلتر البيانات

تم القيام بالعديد من التنقية للبيانات للحصول على بيانات ذات جودة عالية للتدريب والاختبار، من حيث تقييم اللاعبين لبيانات الألعاب ومن حيث تقييم وجودة اللغز لبيانات الألغاز.

1.3.3- فلتر بيانات الألعاب

تمت أولاً فلتر البيانات على ELO للاعبين بحيث إبقاء الألعاب التي يكون الـ ELO لكلا اللاعبين أعلى من 2000 لضمان جودة الألعاب التي سيتم معالجتها، ثم تم سحب الحركات المقابلة للطرف الرابع في حالة كانت اللعبة منتهية بفوز أحدهما على الآخر، وسحب حركات كلا اللاعبين في حالة التعادل.

تم تطبيق كافة الحركات المسحوبة وغير المسحوبة للعبة على رقعة لتشكيل الثنائيات (رقعة قبل الحركة، حركة) بحيث يتم الحفاظ على صحة الرقعة، ثم الإبقاء فقط على الحركات المسحوبة.

تم الاحتفاظ بـ 20000 لعبة من كافة البيانات بحيث يتم فرزها إلى قسمين القسم الأول يحتوي على 10000 لعبة يحتوي على كافة الثنائيات المسحوبة (رقعة قبل الحركة، حركة) وبلغ عددها حوالي الـ 500 ألف ثنائية والقسم الثاني يحتوي على الـ

10000 لعبة أخرى بحيث يتم تقسيمها إلى 6 أقسام كل قسم يحتوي على حركات قطعة معينة حسب نوعها، ويتم أيضا الاحتفاظ بالثنائيات (رقعة قبل الحركة، حركة) التي بلغ عددها حوالي الـ 500 ألف ثنائية أيضا، ثم تخزين كافة الأقسام الـ 7 للبيانات على الـ Google Drive.

تم الاحتفاظ بـ 3000 لعبة إضافية للقيام بعمليات الاختبار.

2.3.3- فلتر بيانات الألغاز

بعد دراسة الرؤى حول المزايا الخاصة بمجموعة البيانات هذه تمت عملية الفلتر على الأجزاء التالية:

أولا تم الإبقاء على الألغاز التي تحتوي عدد حركات أصغر من 7 حركات.

ثم تم الإبقاء على الألغاز التي يتراوح التصنيف الخاص بها بين 800 والـ 1600 ELO.

وعلى الألغاز التي تحمل شعبية أكبر من 85.

وأخيرا على الألغاز التي تم لعبها أكثر من 5 آلاف مرة.

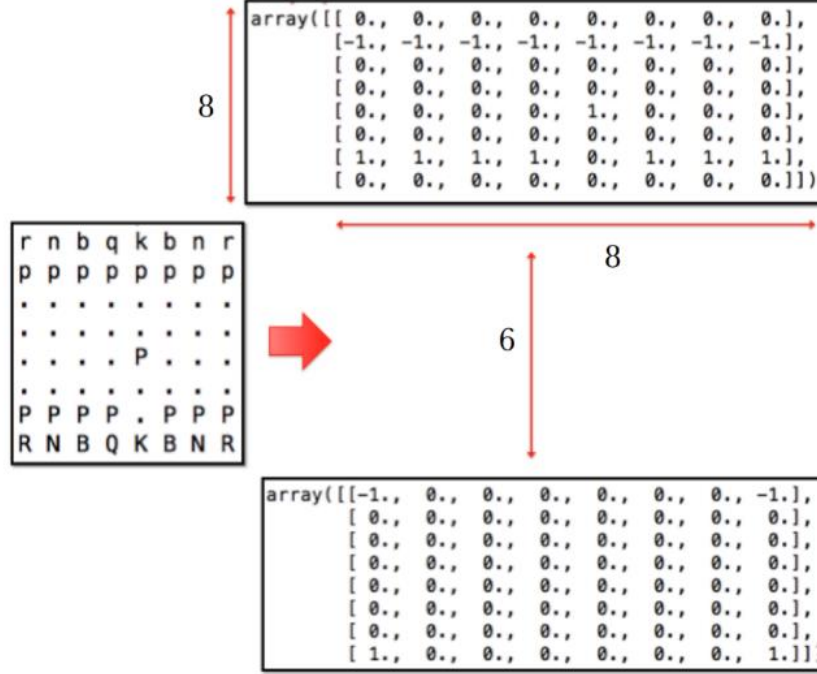
بعد عمليات الفلتر السابقة بقي حوالي 140 ألف لغز، تم فرزها بنفس الطريقة السابقة إلى ثنائيات (رقعة قبل الحركة، حركة) حيث بلغ عدد الثنائيات حوالي 500 ألف ثنائية، وتخزينها على Google Drive.

4.3- تجهيز ومعالجة البيانات

تم تحويل كافة أنواع مجموعات البيانات السابقة (كافة الثنائيات) إلى شكل بحيث يناسب دخل وخرج النموذج، بحيث أولا تم تحويل الحركات من ترميز SAN إلى ترميز UCI، ثم تم تحويل كافة الرقع والحركات بحيث يجب أن يلعب الأبيض، أي تم قلب الرقعة شاقوليا مع قلب ألوانها وقلب إحداثيات الحركة في حال كان الحركة للاعب صاحب اللون الأسود، ثم تحويل الرقعة من FEN إلى غرض Object من الصف Board من مكتبة python-chess، وتحويل هذه الرقعة إلى مصفوفة من الشكل $(6 \times 8 \times 8)$ بحيث 8×8 هي أبعاد الرقعة والـ 6 هي عدد القنوات وكل قناة تحتوي على مواقع القطع من نوع معين لكل من الأبيض مرمزة بـ 1 وقطع الأسود مرمزة بـ -1.

ثانيا تم تحويل كافة الحركات من UCI إلى جزئين منفصلين بحيث الجزء الأول يحتوي ترميز موقع الخلية التي كانت بها القطعة قبل الحركة والجزء الثاني يحتوي على ترميز موقع الخلية التي تحركت إليها القطعة، وتم تحويل هذين الجزئين إلى شعاعين بحيث كل شعاع له طول 64 (حجم الرقعة) يحمل القيمة 1 عند الفهرس المقابل للخلية و 0 في باقي عناصر الشعاع (one-hot encode).

تم تخزين كل من مصفوفات الرقع وأشعة الحركات في Google Drive ليتم إعادة استخدامها لاحقا في التدريب. عند التدريب تم تقسيم البيانات إلى جزئين الجزء الأول حوالي 80% من البيانات للتدريب، و 20% للاختبار.



شكل 24: تحويل الرقعة إلى مصفوفات

5.3- هيكلية النموذج

تم أولاً بناء نموذج يأخذ كمدخل الرقعة على شكل (8×8×12) حيث تم فصل القنوات إلى 12 بدلا من 6 بحيث يتم توزيع قطع اللاعبين على قنوات مختلفة وبحيث يكون خرج النموذج هو شعاع ذي الطول 64×64 والذي يمثل الحركة كاملة (من – إلى) ولكن أعطى هذا النموذج نتائج منخفضة الدقة جدا وذلك بسبب صعوبة المهمة الموكلة إلى النموذج وهي إيجاد الصف المناسب من أصل 4096 صف، حتى مع توسيع حجم البيانات المدرب عليها سيزداد سوء النموذج فقط، والنموذج كان في حالة إفراط في الملائمة Overfitting وذلك لأن النموذج معقد جدا، أو المهمة الموكلة له معقدة جدا، وهذا يدل على ضعف قدرة النموذج على التعميم.

لذلك تم تقسيم المهمة إلى جزئين، الجزء الأول هو تدريب شبكة CNN على التنبؤ بالخلية التي يجب نقل القطعة منها، وهو يحسد فكرة الهروب عندما تتعرض قطعة للهجوم أو يحتاج الملك للتحرك. دخل الشبكة هو تمثيل للرقعة على الشكل (8×8×6) حيث يتم تمثيل قطع اللاعب وقطع الخصم من نفس النوع على نفس القناة ب 1 للاعب و -1 للخصم، ويكون خرج الشبكة هو توزيع احتمالي على خانات الرقعة كاملة وتعبر عن مدى رغبة الشبكة في تحريك القطعة بعيدا عن هذه الخلية (أي اختيار

القطعة التي سوف تتحرك)، مع تصفير كافة الاحتمالات للخلايا التي لا تحتوي على قطع للاعب. أما الجزء الثاني فهو تدريب 6 شبكات أخريات CNNs كل شبكة مسؤولة عن قطعة من قطع الشطرنج ويكون لها نفس دخل الشبكة الأولى أما خرج كل شبكة منها فهو توزيع احتمالي على كافة خلايا الرقعة بحيث تعبر عن مدى رغبة كل شبكة في وضع القطعة المحركة إلى تلك الخلية مع تصفير كافة الاحتمالات للخلايا التي تشكل حركة غير مسموحة.

بهذا نكون قد توصلنا لنموذج يعطي الحركة الأفضل من خلال دمج كل من الشبكة الأولى سنسميها منتقي القطعة Piece Selector مع كافة الشبكات الـ 6 المتبقية سنسميها منتقي الحركة Move Selector. ويمكن الحصول على الحركة الأفضل من خلال ضرب قيم احتمالات محددة القطعة مع أعلى احتمال لمنتقي الحركة المقابلة للقطعة المراد تحريكها من منتقي القطعة ثم أخذ الحركة التي تعطي أكبر احتمال من نواتج الضرب السابقة للحصول على إحداثيات خليتين الأولى هي الخلية التي تحوي القطعة التي نريد نقلها من الرقعة والثانية هي الخلية التي نريد وضع تلك القطعة بها.

ويمكن بهذه الطريقة التأكد من أن النموذج يعطي فقط الحركات المسموحة وذلك بتصفير كافة احتمالات الخلايا التي لا تحوي على قطع للاعب وتصفير احتمالات كافة الخلايا التي لا يمكن وضع القطعة بها.

بهذا يصبح لدينا كل شبكة CNN تتوقع من عدد صفوف يبلغ 64 فقط (جذر تربيعي للنموذج السابق!) مقابل تكلفة مضاعفة في وقت التدريب. ومن المثير للاهتمام، على الرغم من ذلك، أن هذه الطريقة (النموذج الحالي) تلتقط الكثير من الحدس البشري وراء التفكير في لعبة الشطرنج، ففي بعض الأحيان يتم اتخاذ خطوة بهدف حماية قطعة تتعرض للهجوم (شبكة منتقي القطع تنتج احتمالات عالية) وفي أوقات أخرى بهدف رؤية ميزة موضوعية (شبكة منتقي الحركة تنتج احتمالات عالية). الجانب السلبي لهذه الطريقة هو أن هناك مجموعات من الحركات المحددة للغاية لم يتم تعلمها بين كلتا الشبكتين معا، على الرغم من أنه تم اعتبار وجود تمثيلات كافية للتعلم في كل شبكة على حدة، فإن تعلم الشبكتين معا سيكون صعبا ولكن سيعطي نتائج أفضل.

1.5.3- هيكلية الشبكات

تأخذ جميع الشبكات السبعة دخل الرقعة الممثل بالفقرة (تجهيز ومعالجة البيانات) وخرج كل شبكة هو توزيع احتمالي على 8×8 شعاع يمثل قيمة كل خلية.

تم استخدام 3 طبقات من الشبكات العصبونية التلافيفية CNN من الشكل [conv-relu] مع حجم نواة مساوي لـ 3×3 وخرج كل طبقة على الشكل (96، 256، 384)، تليها طبقة Flatten، ومن ثم طبقة كاملة الترابط Fully Connected ذات الحجم 64 معه تابع تفعيل softmax ليكون الخرج على شكل توزيع احتمالي على 64 خلية، عدد معاملات الشبكة تقريبا يساوي 2.7 مليون معامل.

لم يتم التدريب على شبكات أكبر (خاصة تلك التي تحتوي على طبقات تلافيفية أكثر) بحيث هذا يعقد النموذج بشكل أكبر وغير متناسب مع شكل البيانات، لأنه سيكون هناك عدد كبير جدا من المعاملات في الشبكة وشكل البيانات متناثرة Sparse التي يصعب التدريب عليها في الطبقات الكثيرة.

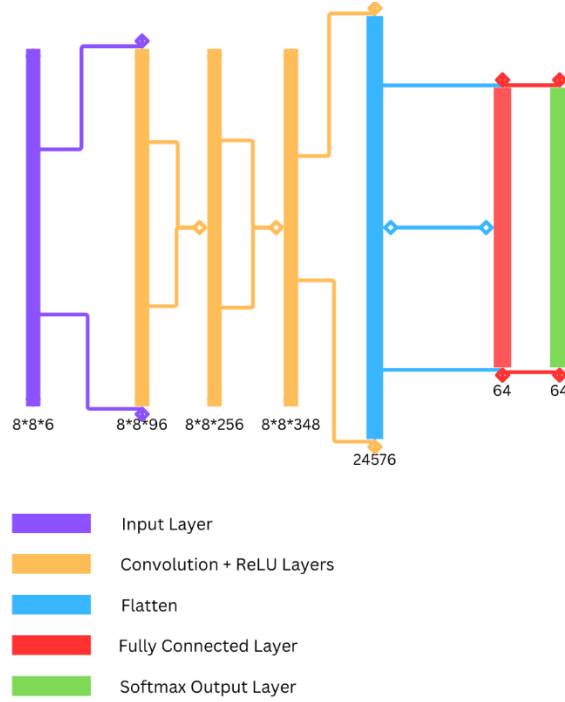
لم يتم استخدام طبقات التجميع Pooling للحفاظ على أكبر قدر من البيانات ممكن أثناء التدريب. التجميع لتعلم التحويلات ليس له أية صلة لأن أي تحويل في رقعة الشطرنج له تأثير كبير على نتيجة الرقعة بالكامل.

بشكل حاسم، كان لابد من تهيئة الأوزان إلى قيم منخفضة جدا لمطابقة القيم الصغيرة للبيانات المكونة من -1 و 0 و 1 في طبقة الدخل. عند التدريب مع أوزان تهيئة عالية، لم يكن لبيانات الدخل أي تأثير على توزيع الاحتمالي للصف النهائي مع تأثيرها الإجمالي على الانتشار الأمامي المنخفض بسبب الأوزان العالية. تم تحديد الأوزان لتكون من مرتبة 10^7 .

لم يتم استخدام أي من التسوية Regularization. حيث أن تجانس المعاملات لا يبدو أنه قابل للتطبيق على هذه المهمة لأن الشطرنج يعرض إنتروبيا أكثر من التعرف على الصور.

كما هو الحال في التسوية، تم اعتبار التسرب Dropout غير متوافق بشكل جيد مع هذه المهمة، صورة الرقعة صغيرة بما يكفي بحيث يجب أن تتفاعل جميع الميزات مع بعضها البعض على مستوى معين، بحيث أنه في حال استبعاد بعض العقد المنشطة سيؤدي إلى القضاء على الأنماط الحاسمة في الانتشار الأمامي. أيضا، نظرا لأن البيانات متناثرة، فإن التسرب يزيل أجزاء من البيانات في التدريب التي تشتت الحاجة إليها في هذه المهمة.

تم استخدام تابع الخسارة softmax بحيث يمكن تفسير التحركات على أنها تتم باستخدام احتمال بدلاً من درجة بمقياس تعسفي. هذا مهم بشكل خاص عند دمج منتقي القطع مع منتقي الحركة لا يمكن دمج مقياسين تعسفيتين معا بأي طريقة ذات معنى. تعتبر الاحتمالات كمخرجات مفيدة أيضا في تفسير الحركات الثانية والثالثة الأفضل لمراقبة الاستراتيجيات الأخرى المقصودة للخوارزمية.

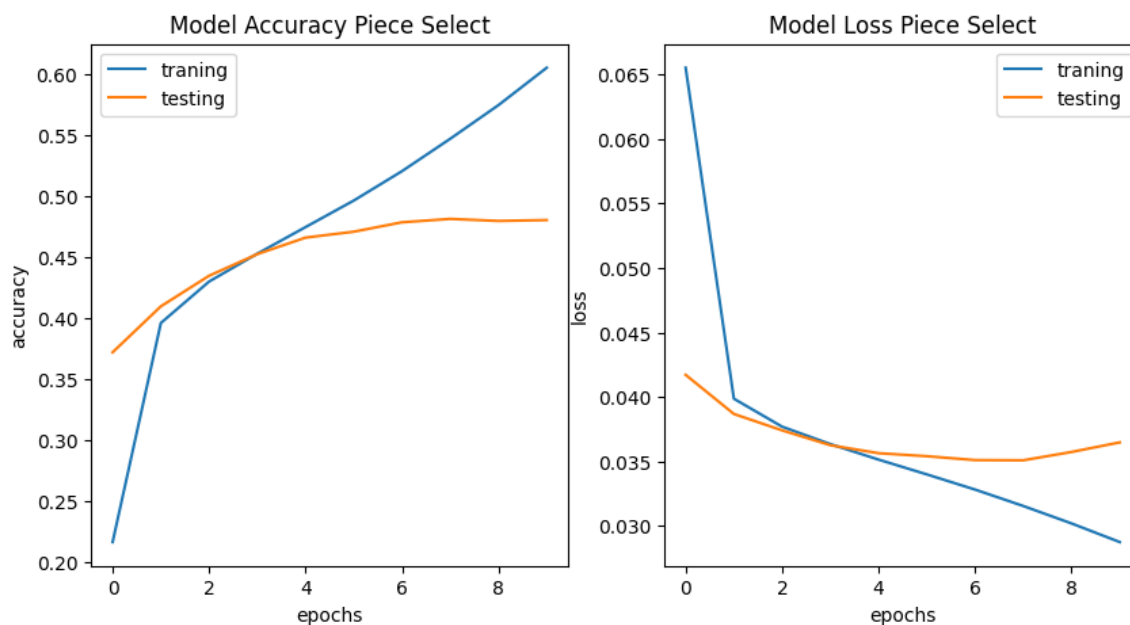


شكل 25: هيكلية الشبكات العصبونية التلافيفية CNNs architecture

2.5.3- تدريب الشبكات

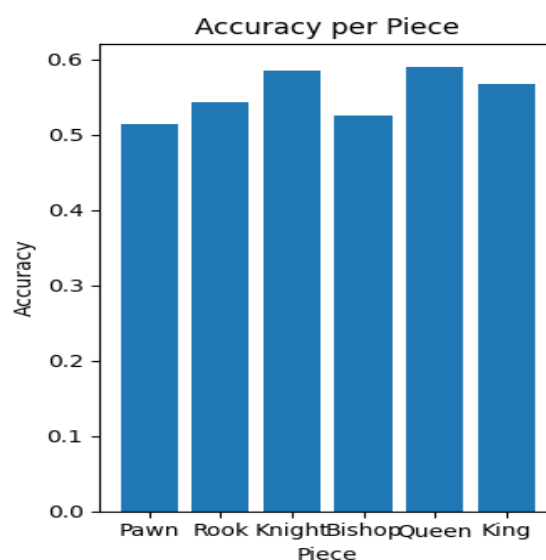
تم تدريب جميع الشبكات باستخدام وحدة المعالجة الرسومية GPU المدعومة من بيئة Google.Colab، حيث تم تدريب كل شبكة على البيانات للألعاب المخزنة في Google Drive المناسبة للشبكة. تم تدريب الشبكة الأولى Piece Selector على البيانات بالمعاملات التالية:

عدد دورات Epochs مساويا لـ 10، ومعدل تعلم Learning rate مساويا لـ 0.0015 وحجم الحزمة Batch Size مساويا لـ 500 وباستخدام المعيار Criterion الانتروبي الثنائي Binary Cross Entropy فكانت النتائج كما يلي:



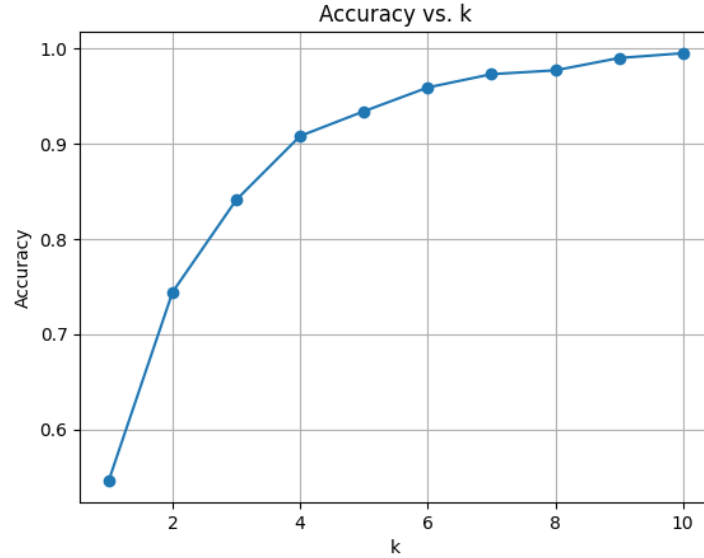
شكل 26: نتيجة تدريب الشبكة الأولى منتقي القطعة **Piece Selector**

نجد أن الدقة مساويا لـ 48% وهو رقم جيد بالنسبة للشبكة الأولى التي تحمل على عاتقها أصعب مهمة في هذا النهج. وكان توزيع الدقة على كافة القطع بالشبكة الأولى:



شكل 27: توزيع الدقة على كافة القطع باستخدام الشبكة الأولى المدربة باستخدام المحسن **Adam**

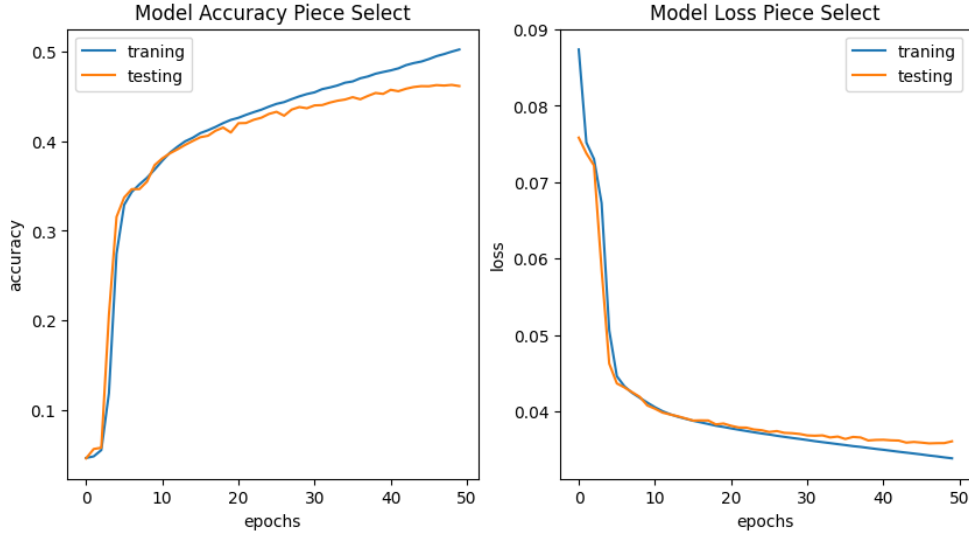
تم حساب دقة الشبكة في حالة كانت اختيار القطعة الأفضل موجودة ضمن أعلى K احتمالا للقطع التي تنبأت بها الشبكة منتقية القطع فكان تغير الدقة بتغير K يعطى كالآتي:



شكل 28: تغير الدقة بدلالة الـ K للشبكة الأولى المدربة باستخدام المحسن Adam

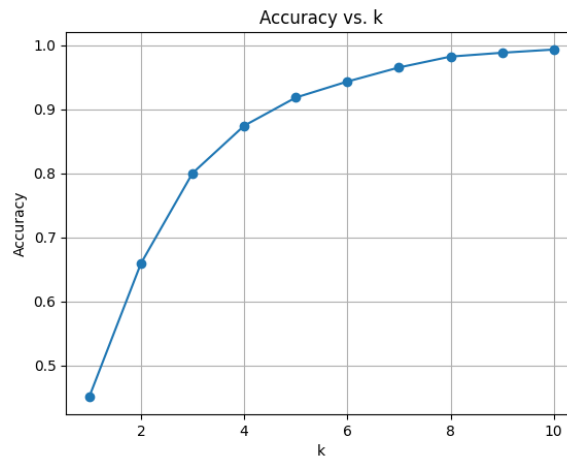
نلاحظ من الشكل (28) أن دقة الشبكة الأولى في أن تتنبأ بأفضل قطعة للحركة هي 54 ولكنها تزداد بشكل سريع جدا مع ازدياد k بحيث أنه عند k مساويا لـ 6 فإن دقة الشبكة تصل لـ 95%.

ونلاحظ من الشكل (26) وجود حالة إفراط في الملائمة Overfitting في التدريب ولذلك تم استبدال المحسن Adam بالمحسن RMSProp للتأكيد على مفهوم "الثقة" في التدريب. نظرا لأن قوة تحديث RMSProp تتأثر بمتوسط تشغيل لأحجام التدرجات الحديثة، فإنها تسمح بالحركات المتكررة للتأثير على النموذج بقوة أكبر. يشجع هذا أيضا على التوزيع النهائي للنتائج للحصول على انحراف معياري أعلى مما يعكس ثقة أكبر في بضع حركات، وهو أمر جيد (يجب تصفية الحركات التي تتم بشكل أقل اتساقا بين اللاعبين - الخصوصيات - ويكون لها تأثير أقل على التدريب)، فكانت نتائج التدريب بعد عدد دورات Epochs مساويا لـ 50 كما في الشكل (29):



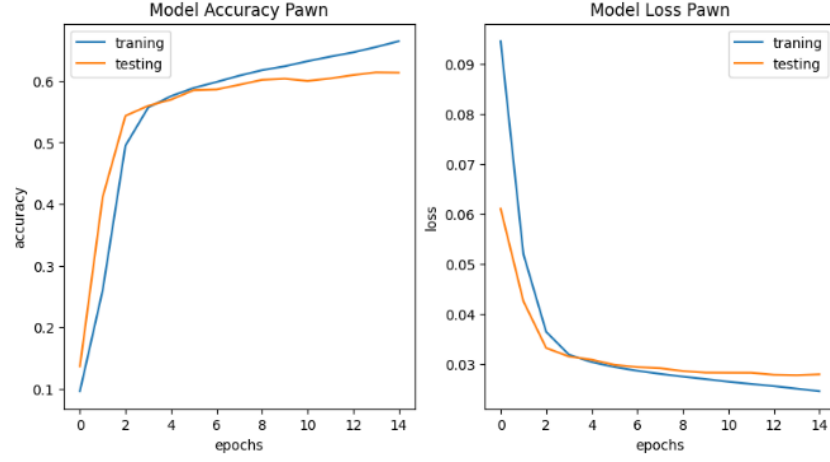
شكل 29: نتيجة تدريب الشبكة الأولى منتقي القطعة **Piece Selector** باستخدام المحسن **RMSProp**

نلاحظ أنه تم تقليل الإفراط في الملائمة **Overfitting** بشكل كبير ولكن لم نصل إلى الدقة التي تم الوصول إليها باستخدام المحسن **Adam**، على الرغم من أن المحسن **RMSProp** أكثر استقراراً وممانعاً للإفراط في الملائمة. تم حساب دقة الشبكة المدربة باستخدام **RMSProp** في حالة كانت اختيار القطعة الأفضل موجودة ضمن أعلى K احتمالاً للقطع التي تنبأت بها الشبكة منتقية القطع فكان تغير الدقة بتغير K يعطى كالآتي

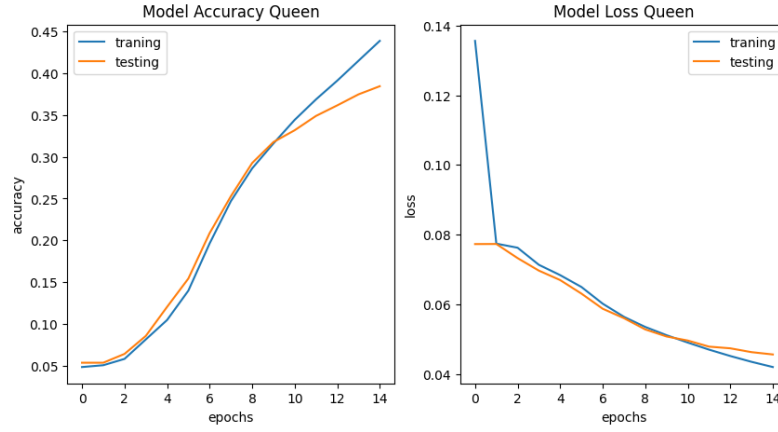


شكل 30: تغير الدقة بدلالة k وذلك للشبكة الأولى المدربة باستخدام المحسن **RMSProp**

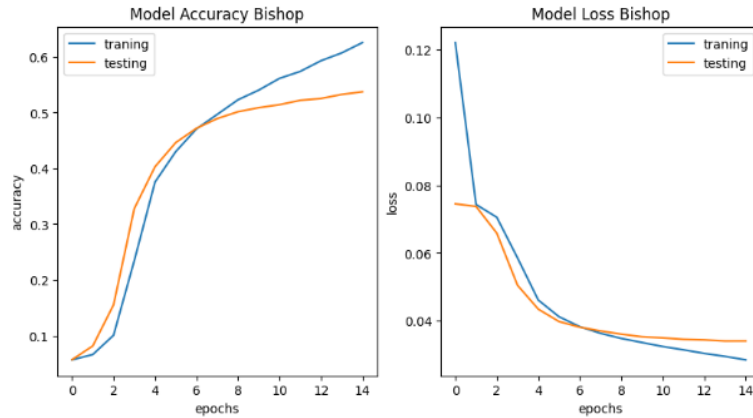
وكانت نتائج تدريب لبقية الشبكات منتقي الحركة بعد عدد دورات Epochs مساويا لـ 15:



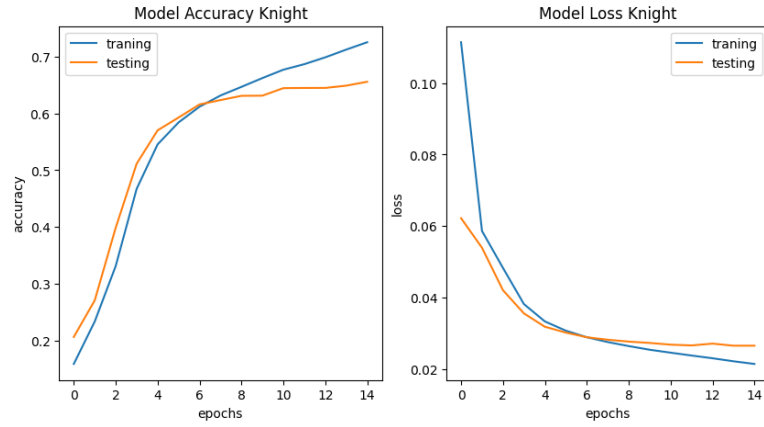
شكل 31: نتيجة تدريب شبكة تحريك البندق



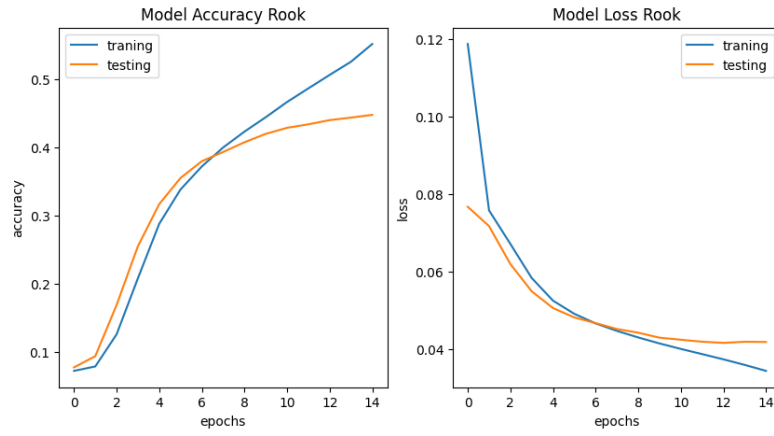
شكل 32: نتيجة تدريب شبكة تحريك الملكة



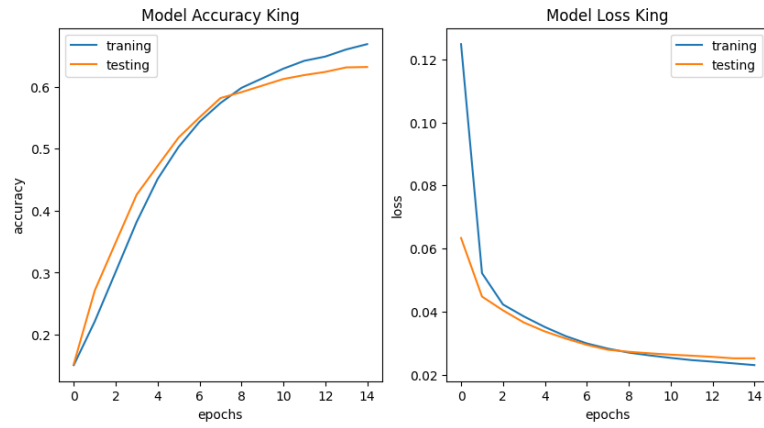
شكل 33: نتيجة تدريب شبكة تحريك الأسقف



شكل 34: نتيجة تدريب شبكة تحريك الفارس

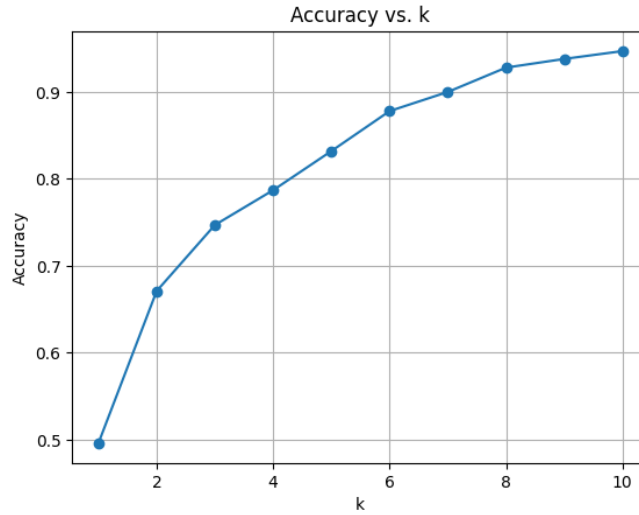


شكل 35: نتيجة تدريب شبكة محرك الرخ



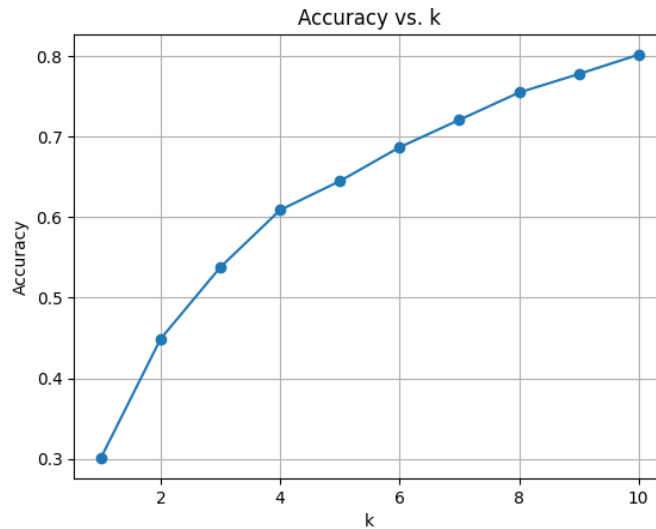
شكل 36: نتيجة تدريب شبكة محرك الملك

تم حساب دقة النموذج في حالة كانت الحركة الأفضل موجودة ضمن أعلى K حركة احتمالاً تنبأ بها النموذج صاحب الشبكة الأولى المدربة باستخدام Adam فكان تغير الدقة بتغير K يعطى كالآتي:



شكل 37: تغير الدقة بتغير K للنموذج صاحب الشبكة الأولى المدربة باستخدام Adam

وتم حساب دقة النموذج في حالة كانت الحركة الأفضل موجودة ضمن أعلى K حركة احتمالاً تنبأ بها النموذج صاحب الشبكة الأولى المدربة باستخدام RMSProp فكان تغير الدقة بتغير K يعطى كالآتي:



شكل 38: تغير الدقة بتغير K للنموذج صاحب الشبكة الأولى المدربة باستخدام RMSProp

نلاحظ أنه باستخدام المحسن Adam لتدريب الشبكة الأولى أعطى نتائج أدق على مستوى النموذج ككل من استخدام المحسن RMSProp على الرغم من أن المحسن RMSProp كان أكثر استقراراً.

3.5.3- إعادة تدريب الشبكات (Fine Tune)

عند اختبار النموذج تبين ضعف أدائه في المواضيع ما بعد الافتتاحيات في اللعبة، وذلك مجموعة البيانات تشارك الكثير من المواضيع في الافتتاحيات، بينما تتفرع إلى مواضيع كثيرة مع تقدم اللعبة ولذلك لا يوجد ما يكفي من المواضيع ليتم التعلم منها، لذلك تم إغناء الشبكة ببيانات الألغاز وذلك ليتعلم النموذج حل الألغاز، وذلك لأن مواضيعها تكثر في متوسط اللعبة وفي الختاميات، تم تدريب جميع الشبكات الأخرى على بيانات الألغاز مع بيانات الألعاب بنسبة 60% للألغاز وذلك لعدم نسيان ما قد تم تعلمه من مواضيع الألعاب.

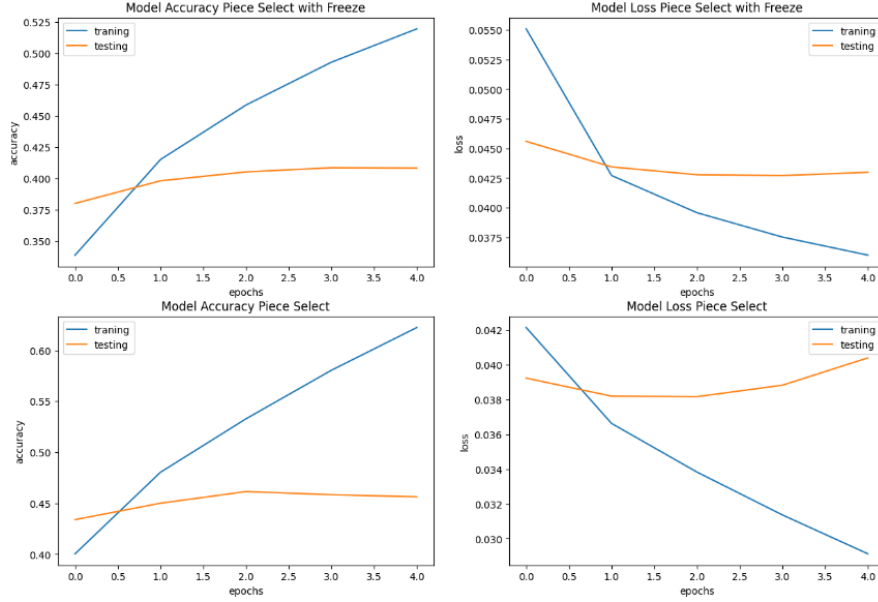
تم اختبار حالتين في إعادة التدريب وذلك كالتالي:

الحالة الأولى مع تجميد معاملات أول 3 طبقات في الشبكات واستبدال الطبقة الأخيرة كاملة الترابط بطبقة جديدة كاملة الترابط (أي إعادة تهيئة الأوزان لهذه الطبقة) والحالة الثانية من دون أي تجميد أو تعديل.

تمت إعادة تدريب كل من الشبكات الـ 7 على البيانات الخاصة بالألغاز المحفوظة على Google Drive ولكن كانت نتائج إعادة التدريب سيئة جداً بحيث تزداد دقة النموذج ككل في نهايات الشطرنج Ending ولكن تنخفض دقة النموذج بشكل دراماتيكي في الافتتاحيات Opening.

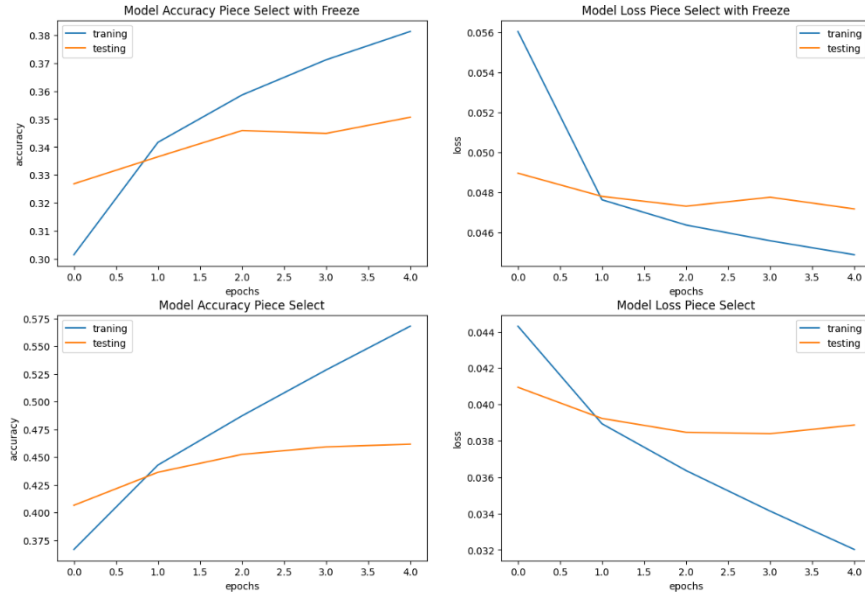
لذلك تمت إعادة تدريب كل من الشبكات الـ 7 على مزيج من بيانات الألغاز وبيانات الألعاب وذلك بنسبة 40% للألعاب لكيلا ينسى النموذج ما تعلمه من البيانات السابقة:

تمت إعادة التدريب باستخدام معاملات التدريب ذاتها ولكن مع تغيير عدد دورات Epochs ليصبح مساوياً لـ 5 للشبكة الأولى منتقي القطع المدربة سابقاً باستخدام المحسن Adam باستخدام ذات المحسن، فكانت النتائج كالآتي:



شكل 39: نتيجة إعادة تدريب الشبكة الأولى منتقي القطعة **Piece Selector** المدربة سابقا باستخدام المحسن **Adam**

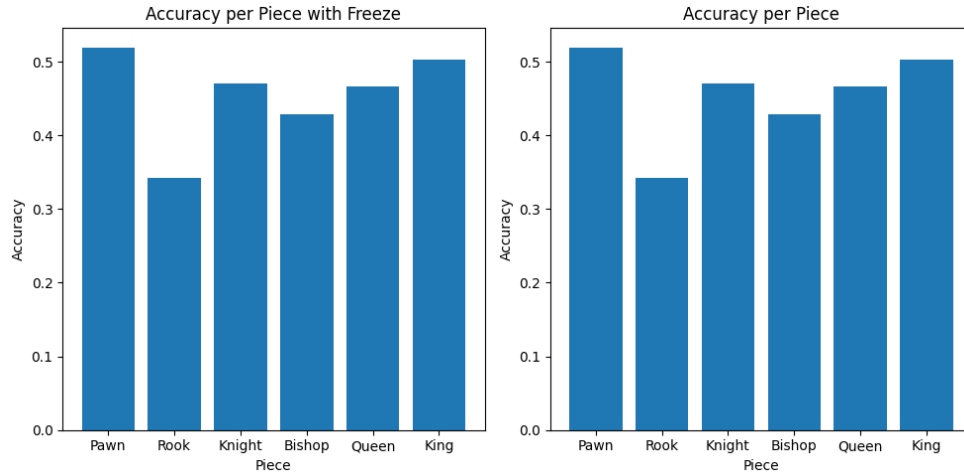
وعند إعادة تدريب الشبكة الأولى المدربة باستخدام المحسن **RMSProp** وذلك باستخدام المحسن **Adam** بعدد دورات Epochs مساويا لـ 5 كانت النتائج:



شكل 40: نتيجة إعادة تدريب الشبكة الأولى المدربة سابقا باستخدام المحسن **RMSProp** باستخدام المحسن **Adam**

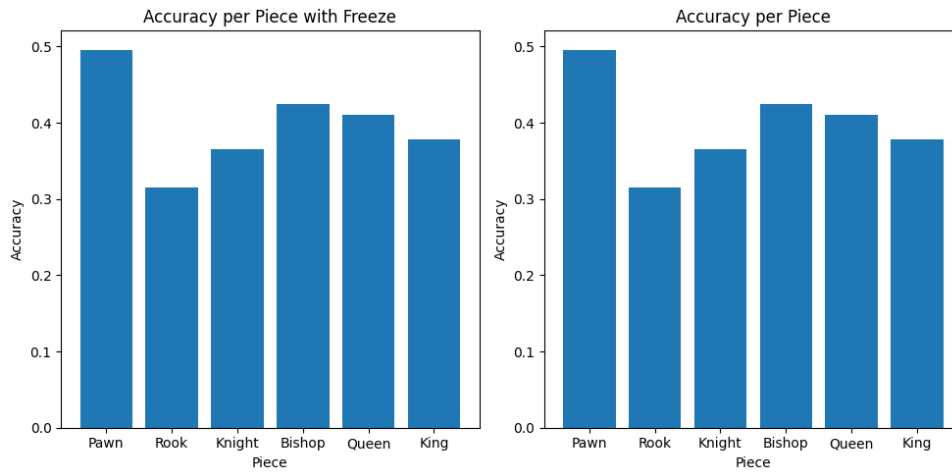
إن إعادة تدريب الشبكة الأولى المدربة سابقا باستخدام المحسن **RMSProp** بذات المحسن أعطى نتائج سيئة للغاية.

وكانت توزيع الدقة على كافة القطع بالشبكة الأولى المدربة باستخدام المحسن Adam:



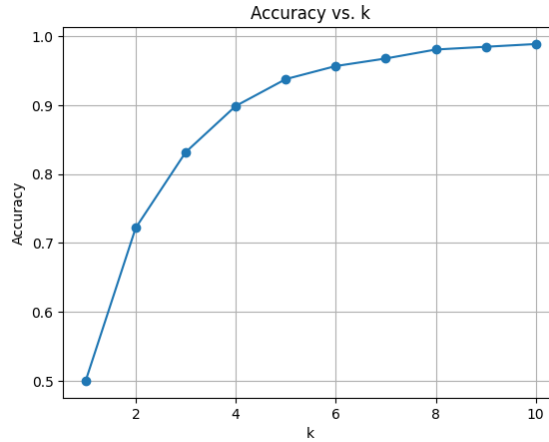
شكل 41: توزيع الدقة على كافة القطع بالشبكة الأولى المدربة باستخدام المحسن Adam والمعاد تدريبها باستخدام نفس المحسن مع وبلا تجميد

أما توزيع الدقة على كافة القطع بالشبكة الأولى المدربة باستخدام المحسن RMSProp:



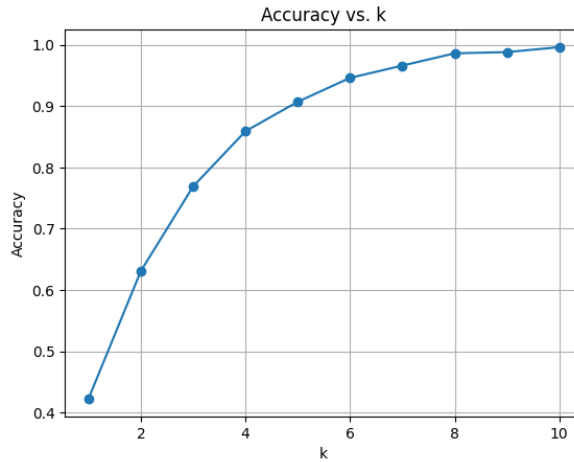
شكل 42: توزيع الدقة على كافة القطع بالشبكة الأولى المدربة باستخدام المحسن RMSProp والمعاد تدريبها بالمحسن Adam مع وبدون تجميد

تم حساب دقة الشبكة المدربة باستخدام Adam في حالة كانت اختيار القطعة الأفضل موجودة ضمن أعلى K احتمالا للقطع التي تنبأت بها الشبكة منتقية القطع المعاد تدريبها فكان تغير الدقة بتغير K يعطى كالآتي



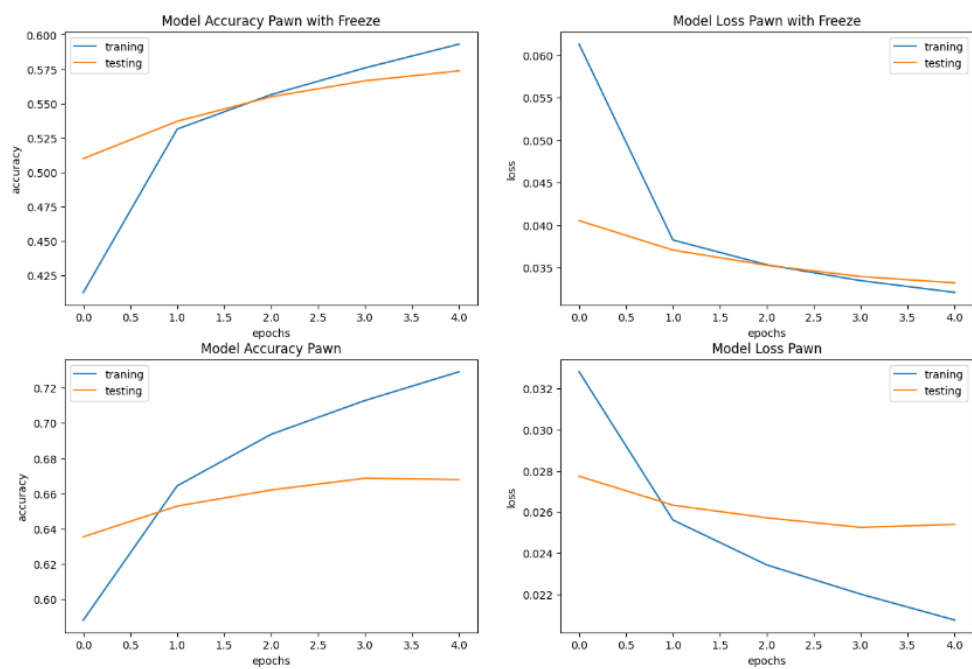
شكل 43: تغير الدقة بدلالة k للشبكة المدربة باستخدام المحسن Adam والمعاد تدريبها باستخدام المحسن Adam

وتم حساب دقة الشبكة المدربة باستخدام RMSProp في حالة كانت اختيار القطعة الأفضل موجودة ضمن أعلى K احتمالا للقطع التي تنبأت بها الشبكة منتقية القطع المعاد تدريبها باستخدام المحسن Adam فكان تغير الدقة بتغير K يعطى كالآتي

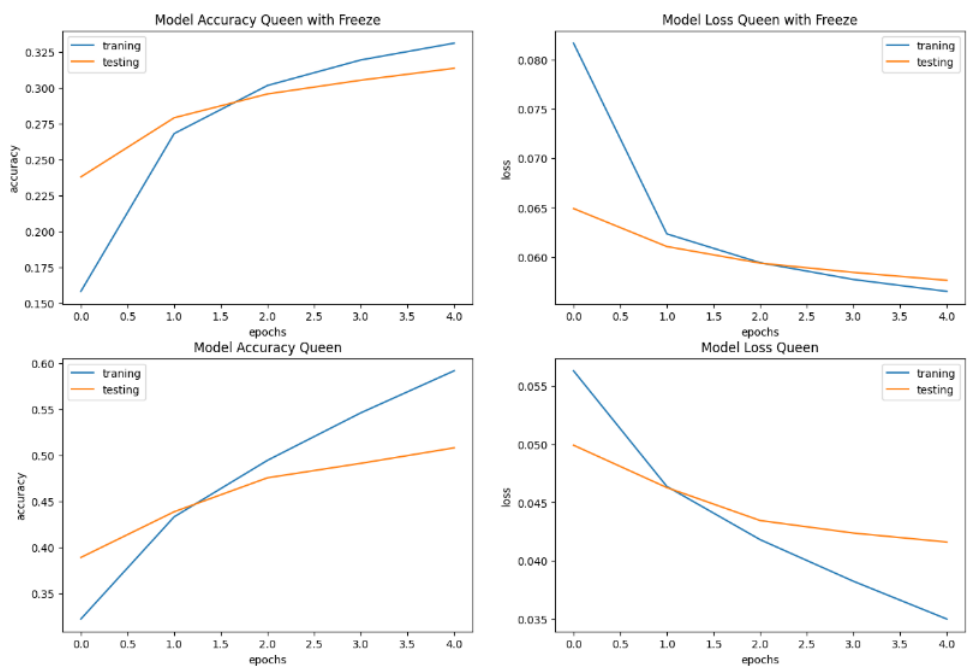


شكل 44: تغير الدقة بدلالة k للشبكة المدربة باستخدام المحسن Adam والمعاد تدريبها باستخدام المحسن Adam

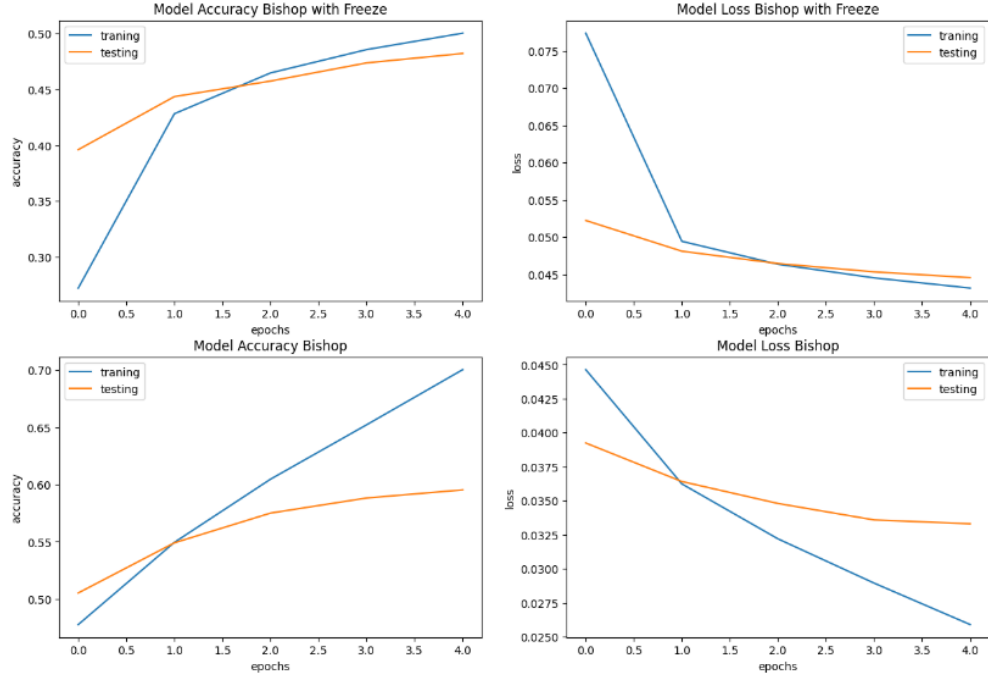
تم إعادة تدريب الشبكات الأخرى منتقية الحركة باستخدام المحسن Adam ولعدد دورات Epochs مساويا لـ 5 فكانت نتائج التدريب:



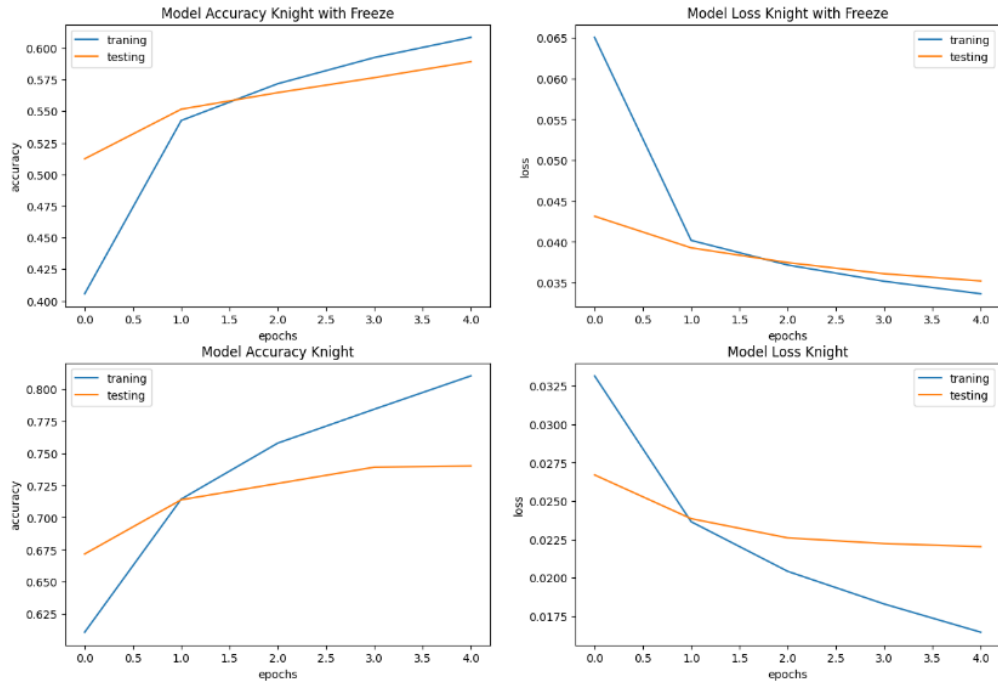
شكل 45: نتيجة إعادة تدريب شبكة محرك البندق



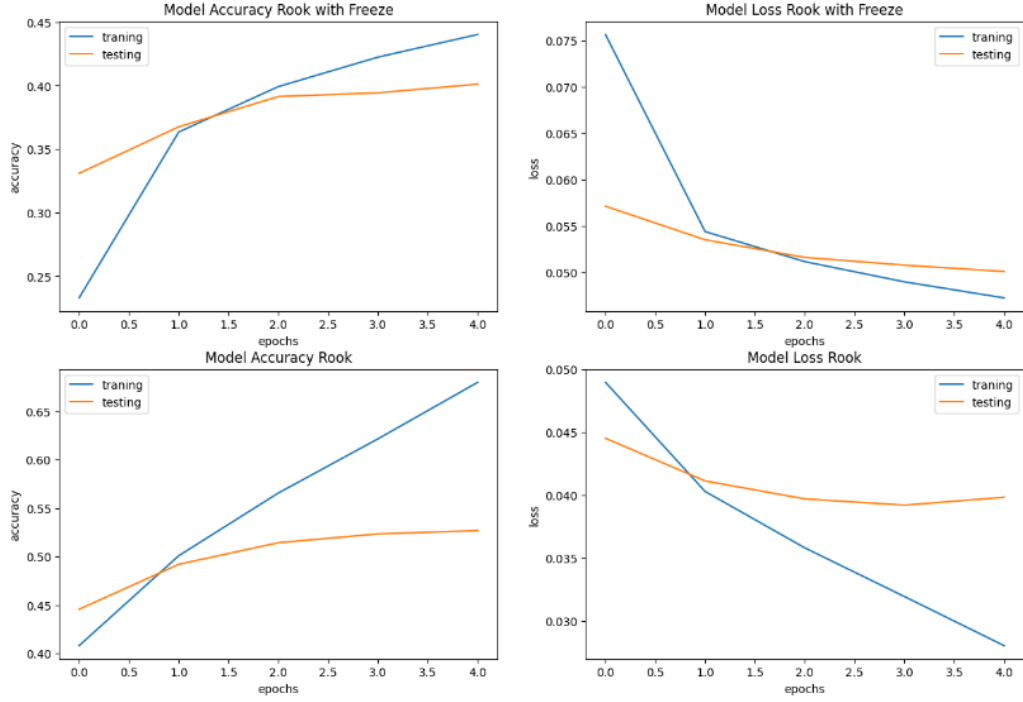
شكل 46: نتيجة إعادة تدريب شبكة محرك الملكة



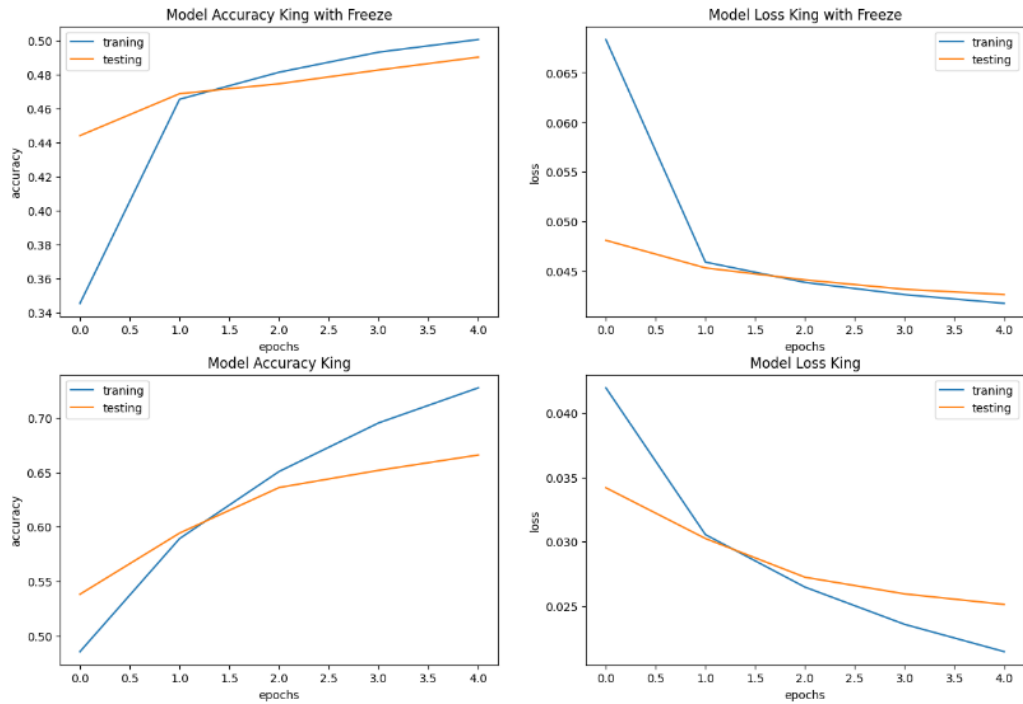
شكل 47: نتيجة إعادة تدريب شبكة محرك الأسقف



شكل 48: نتيجة إعادة تدريب شبكة محرك الفارس

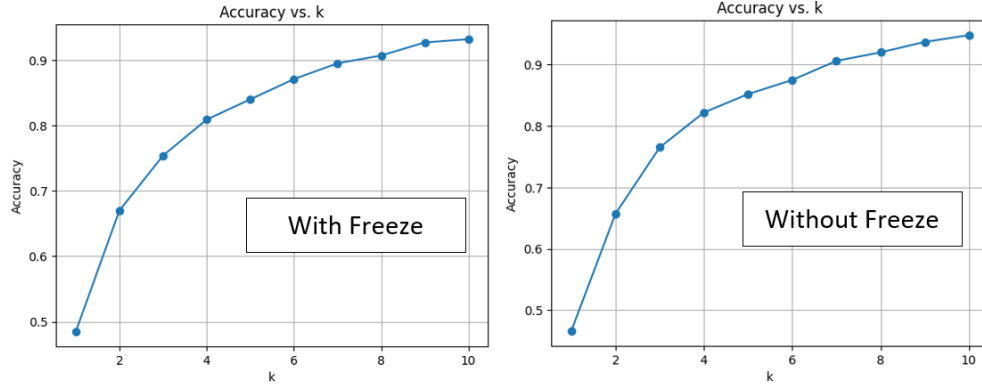


شكل 49: نتيجة إعادة تدريب شبكة محرك الرخ



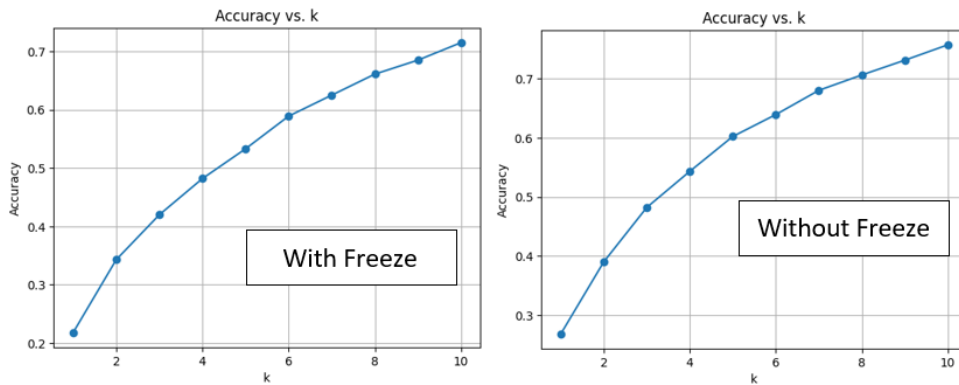
شكل 50: نتيجة إعادة تدريب شبكة محرك الملك

تم حساب دقة النموذج صاحب الشبكة الأولى المدربة باستخدام Adam في حالة كانت الحركة الأفضل موجودة ضمن أعلى K حركة احتمالا تنبأ بها النموذج فكان تغير الدقة بتغير K يعطى كالآتي:



شكل 51: تغير الدقة بتغير K مع وبدون تجميد للنموذج صاحب الشبكة الأولى المدربة باستخدام Adam

كما تم حساب دقة النموذج صاحب الشبكة الأولى المدربة باستخدام RMSProp في حالة كانت الحركة الأفضل موجودة ضمن أعلى K حركة احتمالا تنبأ بها النموذج فكان تغير الدقة بتغير K يعطى كالآتي:



شكل 52: تغير الدقة بتغير K مع وبدون تجميد للنموذج صاحب الشبكة الأولى المدربة باستخدام RMSProp

نلاحظ أنه في حالة تجميد معاملات الطبقات التلافيفية لم ينفذ في تحسيت النموذج وإنما أثر سلبا على دقته. لقد استغرق تدريب كل شبكة حوالي الـ 15 دقيقة كما استغرق إعادة التدريب كل شبكة حوالي 10 دقائق.

6.3- شجرة البحث

من المخططات السابقة لتغير الدقة بدلالة k نجد أن الدقة تتحسن بشكل جيد مع ازدياد k ولكن لا نعلم ما هي الحركة الأفضل بين الـ k حركة المقترحة من قبل النموذج لذلك لمعرفة الحركة الأفضل فيمكننا بناء شجرة بحث minimax لإيجاد الحركة التي تعطي أفضل تقييم للموضع ولكن مع ازدياد k يزداد تفرع الشجرة Branching Factor وهذا ما يزيد من المدة الزمنية اللازمة للبحث عن الحركة الأفضل وهنا نجد المفاضلة بين الدقة والزمن، لذلك يمكن اتخاذ k معينة بحيث لا نخسر الكثير من الدقة بالإضافة إلى عدم ازدياد المدة الزمنية اللازمة بشكل كبير. تم تثبيت k على القيمة 4 والعمق مساوياً للـ 4 بعد اختبارات معينة لمراعاة الوقت وتم اختيار النموذج المعاد تدريبه بلا تجميد ليكون النموذج الذي يتنبأ بالـ k حركة لموضع ما ليستخدم مع شجرة البحث.

1.6.3- بناء الشجرة

تم بناء شجرة بحث تتبع خوارزمية تقليل alpha-beta بحيث يكون الـ k هو معامل الـ branching factor للشجرة وعمق الشجرة يمكن تثبيته عند حد معين حسب الزمن اللازم والدقة المطلوبة، أولاً ابتداء من الجذر الذي يحتوي على الرقعة التي نريد معرفة أفضل حركة موافقة للموضع الحالي، نستدعي التابع الذي يتنبأ بأفضل k حركة من النموذج السابق ثم يتم بناء العقد الأبناء عن طريق تطبيق الحركات k على الرقعة لنحصل على k رقعة ثم نبدأ عملية البحث العودي لإيجاد الحركة التي تعطي أفضل تقييم، مع تقليل أجزاء الشجرة التي لا يمكن الوصول إليها بسبب منطق اللعبة ذات المجموع الصفري (كلا اللاعبين يلعب بأفضل الطريقة ويحاول الفوز، وتقدم طرف من أطراف اللعبة بأفضلية بمقدار معين يعني أن الطرف الآخر متأخر بنفس المقدار ولكن بإشارة سالبة)، تم إضافة بعض التحسينات كترتيب العقد المراد زيارتها والتخبة.

2.6.3- تابع التقييم

يمكن بناء استدلالية معينة بحيث تعطي تقييم للموضع الحالي للرقعة بغض النظر عن أي بحث في الشجرة يمكن اتباع بعض أساسيات تقييم موضع رقعة الشطرنج لبناء هكذا تابع بحيث يعتمد على مجموعة عوامل، كعدد كل قطعة مضروبة بثقل للقطعة، هيكل البيادق (ويتضمن السلاسل البيدقية والبيادق المعزولة والجزر البيدقية والبيادق الخلفية والبيادق السالكة والبيادق المزدوجة)، أمان الملك، تطوير القطع، التحكم والسيطرة على منتصف الرقعة، المساحة الشاغرة، نشاط القطع، وأمور أخرى كالحفاظ على كلا الأسقفين ووجود الرخ على الصف السابع (أي صف بيادق الخصم) أو على الأعمدة المفتوحة أو نصف المفتوحة.

ولكن لبناء مثل هذا التابع يجب التقدير الخارق لقيمة كل معامل وتثقله لحساب تقييم الموضع وأي خطأ حسابي صغير في التقييم يؤدي إلى خطأ فادح في توقع الحركة الأفضل للنظام، لذلك يمكن اللجوء لتدريب نموذج للقيام بمهمة تقييم الموضع.

ولكن أيضا لبناء مثل هكذا نموذج فيجب أن يكون لدينا بيانات عن مواضع منمطة بالتقييم الخاص بكل موضع والبيانات التي لدينا لا تحتوي على تقييم وتنميطها على محرك شطرنج يكلف الكثير من الوقت، لذلك بدلا من تدريب نموذج للقيام بمهمة التقييم سوف نستعين بنماذج مدربة مسبقا وهي عبارة محرك الشطرنج نفسه بحيث أنه مدرب على مواضع شطرنجية كثيرة ودقته في التقييم أعلى بكثير من التابع الاستدلالي التي نريد بناءه لذلك سوف نستخدم محرك الشطرنج Stockfish بحيث يعطينا فقط تقييم الموضع لكل عقدة بالشجرة.

يمكن التحكم بحد الزمن اللازم لإيجاد التقييم للموضع وذلك لتسريع التقييم ولكن يتم ذلك على حساب الزمن، فعند زيادة الزمن تزداد دقة تقييم الموضع وكذلك دقة الحل الناتج أي الحركة الأفضل، فيمكن المفاضلة بين الزمن والدقة عن طريق التحكم بهذا الحد.

3.6.3- تحسينات شجرة البحث

تم تسريع البحث عن طريق ترتيب العقد الأولاد عند توليدها عند كل عقدة حسب تقييم العقدة باستخدام تابع التقييم، وذلك لأن تابع التقييم لا يستهلك الكثير من الوقت مقارنة بتابع معرفة k حركة المتنبئة من قبل النموذج أي توليد أولاد العقدة، وبهذا الترتيب نضمن أنه إذا كان يمكن أن يحدث تقليل في هذه العقدة سوف يحدث في العقد البدائية، وهو ما يسرع عملية البحث ككل.

تم تسريع البحث في حال لعب لعبة كاملة ضد خصم معين وذلك لأن النظام ككل سوف يختار الحركة التي تعطي أفضل تقييم من الشجرة وبهذا فرع من أفرع الشجرة قد تمت دراسته مسبقا في الحركة السابقة وفي حال لعب الخصم لأفضل حركة له فهي أيضا موجودة ضمن نفس فرع الشجرة السابق وهو أيضا تمت دراسته سابقا، أما في حال لم يلعب الخصم أفضل حركة له فلا يتم التسريع بأفضل ما يمكن ولكن بما أن الخصم لم يلعب أفضل حركة فهذا يعني أن الأفضلية في صالح النظام، فلا مشكلة من حساب هذا الفرع من الشجرة من جديد.

يمكن التسريع السابق بكل بساطة بتخبيئة الحركات k المتنبئة من قبل النموذج في الذاكرة بحيث نقوم بتطبيق تابع Hash على التدوينة FEN التي تعبر عن الموضع في العقدة ونخزن ال k حركة في الذاكرة عند هذا ال hashed FEN بحيث يتم التحقق من وجوده في الذاكرة قبل استدعاء تابع التنبؤ الخاص بالنموذج، ويمكن تفريغ تلك الذاكرة بعد تجاوزها حد معين عن طريق حذف عدد معين من ال hashed FEN المستخدمة سابقا.

تم تسريع الحل العام بقصر نتيجة الشبكة الأولى على عدد k من القطع فقط وذلك على حساب دقة تنبؤ النموذج، بحيث يتم اختيار k قطعة التي تحمل الاحتمالات الأعلى ضمن باقي القطع.

أخيرا عند البحث يتم بناء جميع الأشجار من العمق 1 إلى العمق k بحيث يتم وضع حد للوقت يمكن تحديده من قبل المطور في حال تجاوز وقت البحث في الشجرة هذا الحد فيتم إعادة آخر حركة أعطت أعلى تقييم في العمق السابق للعمق الحالي الذي يتم البحث به.

4- الفصل الرابع

الاختبارات والنتائج

يتحدث هذا الفصل عن مجموعة الاختبارات التي تم إنجازها على النظام لاختبار دقته وأدائه.

1.4- اختبار النظام عن طريق تحليل الألعاب الفردية

تم اللعب ضد النماذج السابقة جميعاً، بما في ذلك النموذج المدرب فقط باستخدام المحسن Adam (فقرة 2.5.3)، والنموذج المعاد تدريبه من دون تجميد والنموذج المعاد تدريبه مع تجميد (الفقرة 3.5.3)، والنموذج المعاد تدريبه من دون تجميد مع استخدام الشجرة.

1.1.4- النموذج المدرب فقط

تم اللعب ضده باللون الأبيض فقد خسر وكان تحليل اللعبة في الشكل (53)



شكل 53: تحليل اللعبة ضد النموذج المدرب فقط وهو يلعب باللون الأسود

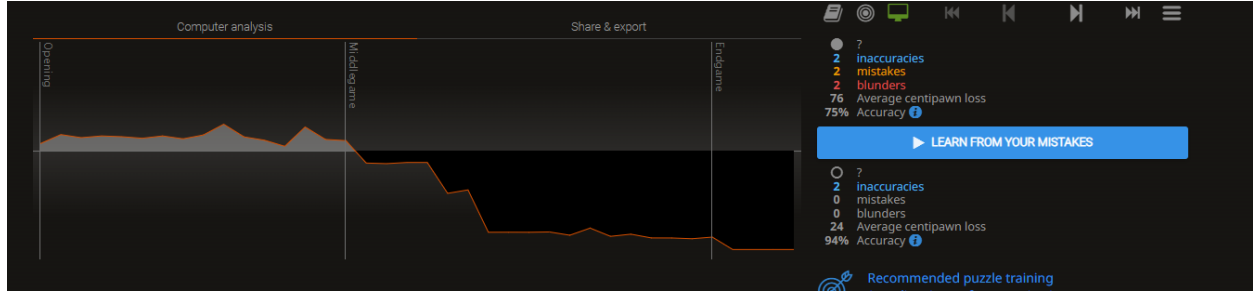
نجد من الشكل السابق أنه يلعب بدقة 41% فقط كما أنه سيء جداً بعد أن تنتهي الافتتاحيات.

وذلك لأن النموذج مدرب بشكل كبير على الافتتاحيات لأنها تشترك بجميع الألعاب أما وسط اللعبة والنهايات فمن الصعب جداً الوصول لنفس الموضع في لعبتين مختلفتين.

لذلك تم تدريبه على الألغاز لتعزيز هذا الجانب في النموذج.

2.1.4- النموذج المعاد تدريبه من دون تجميد الطبقات التلافيفية

تم اللعب ضده باللون الأسود فخسر وكانت نتيجة تحليل اللعبة في الشكل (54)

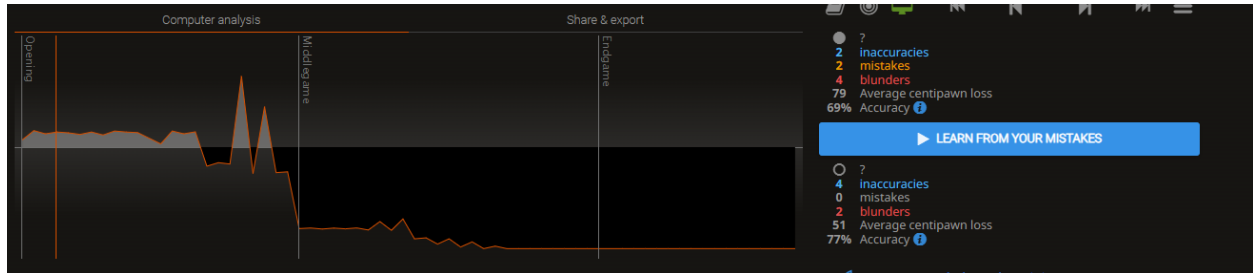


شكل 54: تحليل اللعبة ضد النموذج المعاد تدريبه على الألغاز ومن دون تجميد الطبقات التلافيفية وهو يلعب باللون الأبيض

نجد من التحليل السابق أن النموذج تحسنت دقته عن اللعبة السابقة فقد أصبحت 75% ولكنه ما زال يرتكب الكثير من الأخطاء ولم يتحسن أدائه بكلتا مرحلتين وسط اللعبة والنهايات بشكل كبير.

3.1.4- النموذج المعاد تدريبه مع تجميد الطبقات التلافيفية

تم اللعب ضده باللون الأسود أيضا فكانت نتيجة تحليل اللعبة في الشكل (55)

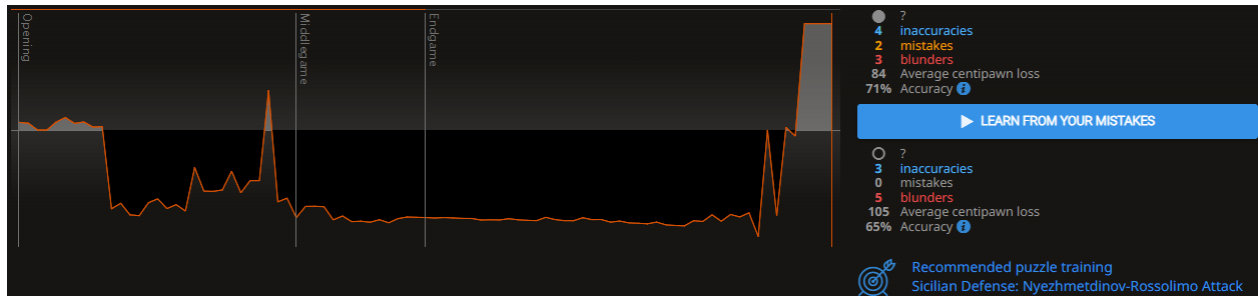


شكل 55: تحليل اللعبة ضد النموذج المعاد تدريبه على الألغاز مع تجميد الطبقات التلافيفية باللون الأبيض

نجد من التحليل السابق أن دقة النموذج المعاد تدريبه مع تجميد أسوأ من الدقة التي لعب بها النموذج من دون تجميد وذلك يدعم ما رأيناه في القسم السابق حيث أن التجميد لم يفيد في تحسين النموذج، وله الكثير من الأخطاء، ولم يتحسن أدائه بكلتا مرحلتين وسط اللعبة والنهايات بشكل كبير.

4.1.4- النموذج المعاد تدريبه من دون تجميد مع شجرة البحث

تم أولا اللعب ضده باستخدام شجرة بحث بعمق 4 وبعامل تفرع 4 حيث استغرقت اللعبة وقتا طويلا ولكن لعب بدقة عالية جدا، تم اللعب ضده باللون الأبيض

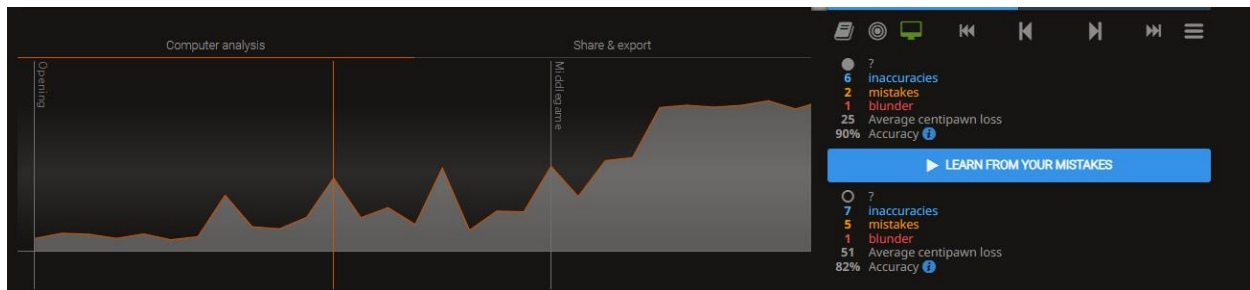


شكل 56: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأسود

نجد من التحليل السابق أنه كان مستحوذاً على التقدم في اللعبة بكاملها من الافتتاحية إلى وسط اللعبة وصولاً إلى النهايات ولكن في النهايات قام النموذج بتحريك الملك فقط لـ 3 خطوات متتالية مع أن الحركة الأفضل كانت بتحريك الرخ وليس الملك، وينتج ذلك الخطأ عن الشبكة الأولى في النموذج، حيث تم الاطلاع ودراسة جميع الحركات المقترحة من قبل النموذج وجميعها تقضي تحريك الملك، أي أن الشبكة الأولى انحازت إلى رفع احتمالية تحريك الملك بشكل كبير في النهايات.

وعلى الرغم من ذلك فقد لعب النموذج بدقة 65%.

أما اللعبة التي لعب بها باللون الأبيض فقد أبحر الجميع بفوزه الأول 😊



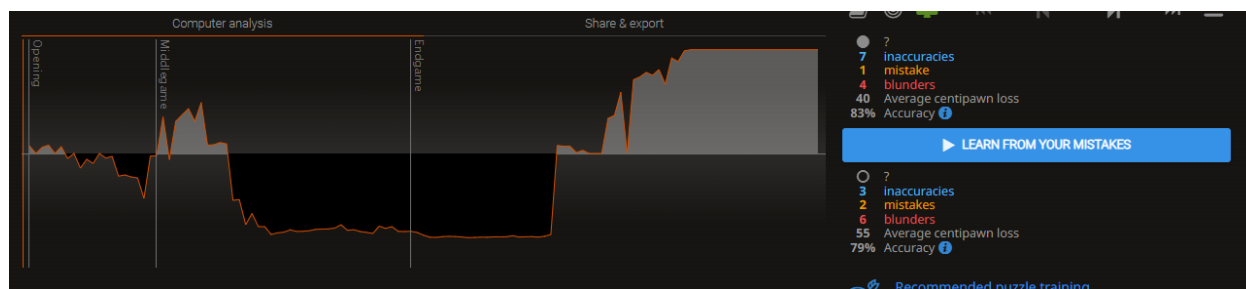
شكل 57: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأبيض

نجد من التحليل السابق أنه كان يستحوذ على الأفضلية في كامل اللعبة، كما أنه لعب بدقة 90% في هذه اللعبة.

وبهذا نجد أن شجرة البحث Alpha Beta قد حسنت كثيراً من دقة النموذج.

تم أيضا اللعب ضده باستخدام شجرة بحث بعمق وعامل تفرع ديناميكيين بحيث يبدأ أولا بعمق 1 وعامل تفرع 1 في أول 5 تحركات في اللعبة ثم ثانيا ينتقل إلى حالة عمق 2 وعامل تفرع 2 ويتم اختيار أفضل 6 قطع اقترحت تحريكهم الشبكة الأولى وحد زمني للتقييم 0.1 ثانية في الـ 10 حركات اللاحقة، ثم ينتقل إلى عمق 3 وعامل تفرع 3 ويتم اختيار أفضل 6 قطع اقترحت من قبل الشبكة الأولى وحد زمني 0.1 حتى يتبقى لطرف النموذج 5 قطع كبيرة (القطع الكبيرة هي جميع القطع ما عدا الملك والبيادق) حينها ينتقل للمرحلة الأخيرة وهي عمق 4 وعامل تفرع 4 ويتم اختيار أفضل 3 قطع اقترحت تحريكهم الشبكة الأولى وحد زمني 0.1 ثانية، حيث تستغرق الحركة من الحالة الأولى حوالي الثانية الواحدة وتستغرق الحركة من الحالة الثانية حوالي 5 ثواني وتستغرق الحركة من الحالة الثالثة من 10 إلى 25 ثانية تتغير بحسب عدد القطع وتعقيد الموضع وأخيرا تستغرق الحركة من الحالة الأخيرة في حالة عدد القطع القليل 20 إلى 30 ثانية، وعندها تستغرق اللعبة بشكل عام حوالي 15 دقيقة، وهو تسريع كبير مقارنة بالحالة السابقة ولكن على حساب الدقة بحيث انخفضت دقة النموذج بشكل عام.

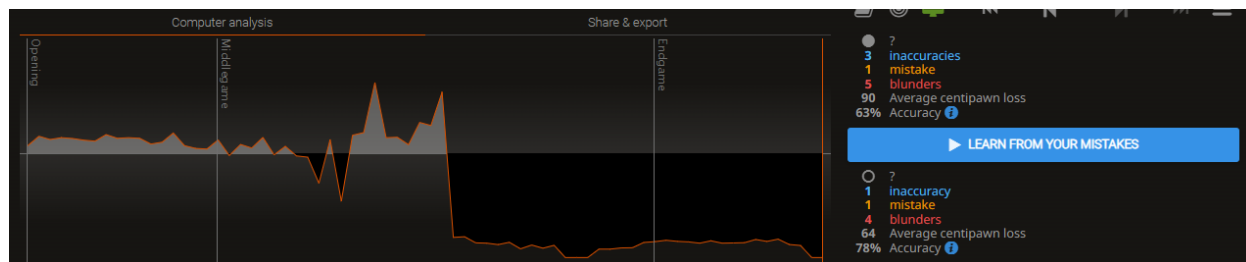
تم أولا اللعب ضده باللون الأبيض



شكل 58: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأسود

نجد أنه كان يملك الأفضلية في وسط اللعبة ولكنها خسرها بعد الدخول في نهاية اللعبة ولذلك نستنتج أنه بالإمكان زيادة العمق ومعامل التفرع للحصول على دقة أكبر في اللعب، ونجد هنا أنه لعب بدقة 79%.

تم ثانيا اللعب ضده باللون الأسود



شكل 59: نتيجة تحليل اللعبة ضد النموذج مع الشجرة باللون الأبيض

نجد من التحليل السابق أنه كان يستحوذ على الأفضلية في وسط اللعبة ولكنها يخسر في النهايات، كما أنه لعب بدقة 63% في هذه اللعبة.

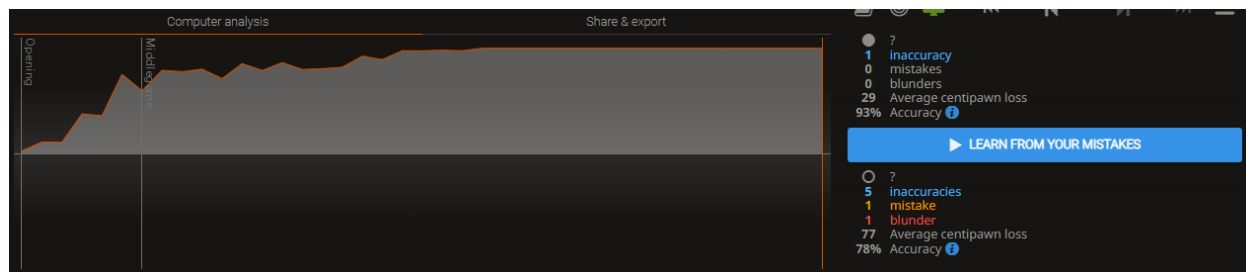
وبهذا نجد أن شجرة البحث Alpha Beta قد حسنت كثيرا من دقة النموذج.

أخيرا تم تجربة النموذج ضد Stockfish ولكن بتحديد قوة Stockfish لـ 1000, 1500 ELO فكانت نتيجة العشر ألعاب هي:

مستوى Stockfish	عدد الألعاب	فوز Stockfish	فوز النموذج	التعادل	نسبة اللاعبين فوق الـ ELO المختار
1000	6	1	4	1	49.95%
1500	6	1	3	2	24.92%

حسب موقع [Chess Rating Percentile Calculator & Distribution Graph \(chessgrandmonkey.com\)](http://chessgrandmonkey.com)

وهذه تحليل إحدى الألعاب بين Stockfish والنموذج المدرب باستخدام المحسن Adam والمعاد تدريبه من دون تجميد الطبقات التلافيفية باستخدام المحسن ذاته مع استخدام شجرة البحث تقليم ألفا بيتا Alpha-Beta Pruning ذات العمق والعامل التفرع المتغير ديناميكيا مع حالة اللعبة، حيث أن Stockfish يلعب بالأسود بتقييم 1500 Elo ونموذج يلعب باللون الأبيض.



شكل 60: نتيجة تحليل اللعبة بين Stockfish 1500 Elo بالأسود والنموذج مع الشجرة الديناميكية بالأبيض

نجد أن النموذج قد لعب بدقة 93% بـ 1 Inaccuracy.

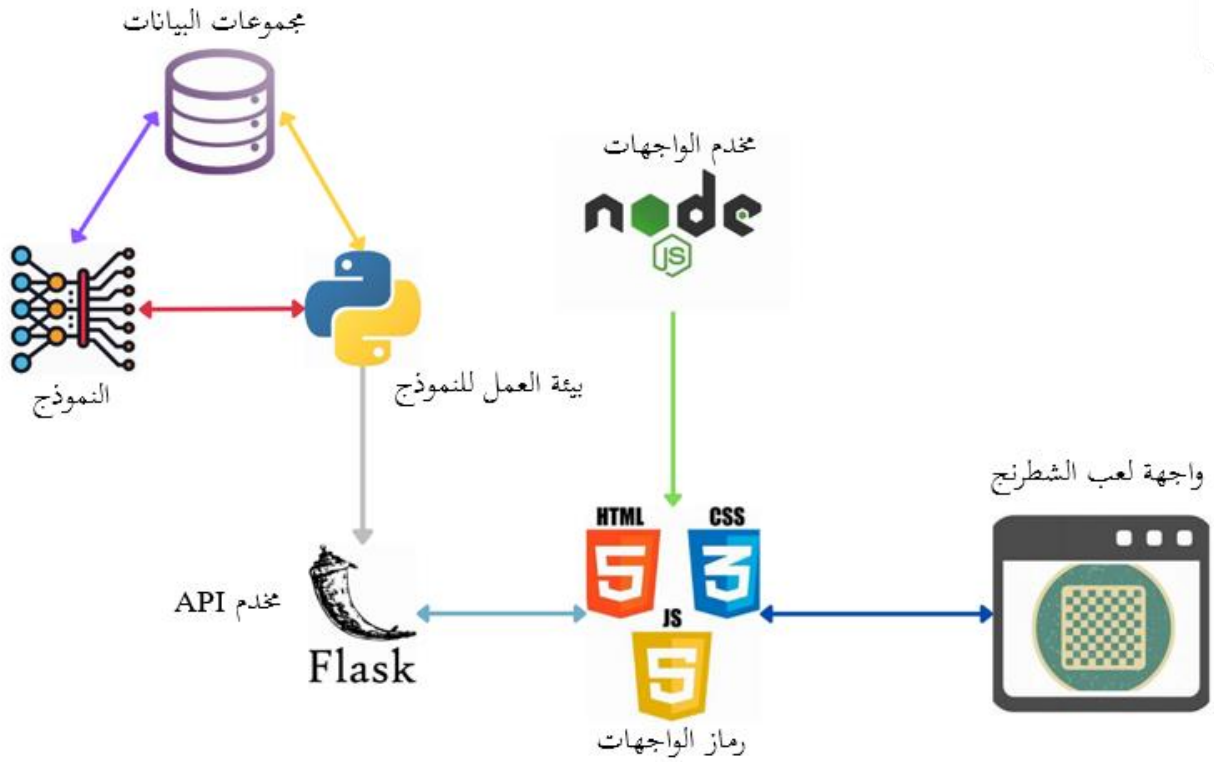
ومن الجدول السابق نؤمن قوة النموذج المطور بـ 1700 Elo.

5-الفصل الخامس

التنفيذ العملي

يتحدث هذا الفصل عن هيكلية المشروع العامة، مع ذكر بيئات العمل والمكاتب المستخدم وتنفيذ واجهات المشروع وواجهة برمجة التطبيقات.

1.5- الهيكل العام للمشروع



شكل 61: الهيكل العام للمشروع

2.5- بيانات العمل المستخدمة

بغية تنفيذ النظام المقترح تمت الاستعانة بعدد من البرمجيات مفتوحة المصدر التي تساعد في عملية تدريب محرك الشطرنج من مرحلة تجهيز المعطيات وصولاً إلى النموذج النهائي.

1.2.5- بيئة Google.Colab

تعتبر منصة Google.Colab من أهم وأشهر المنصات التي تتيح إمكانية بناء مشاريع ذكاء صناعي وتعلم الآلة، حيث أنها تقوم بتقديم إمكانية الاستفادة من وحدة معالجة رسومية GPU مجانية، ويمكن استخدام هذه المنصة لكتابة لغة رماز بلغة بايثون Python ضمن مفكرة Notebook ضمن المتصفح بشكل تفاعلي Interactive وتتيح إمكانية حفظ ونشر هذه المفكرة عن طريق مشاركة رابط فقط.

تستعمل هذه المنصة بشكل أساسي لإنجاز تدريب نماذج تعلم الآلة والتعلم العميق، حيث أن عملية بناء نماذج الشبكات العصبية تحتاج إلى كمية هائلة من جدا من العمليات الحسابية على مجموعة بيانات التدريب بغرض ضبط أوزان الشبكة، وعوضا عن استهلاك موارد الحواسيب الشخصية التي قد لا تكون ملائمة، فإن المطور يمكن أن يختار وضع استخدام وحدة المعالجة الرسومية GPU المجانية واستخدمها لتدريب الشبكة المطلوبة.

ومن أهم ميزات هذه المنصة أنها تتيح إمكانية وصل (ربط) مع قرص سحابة غوغل Google Drive، بالتالي فإنه يمكننا الوصول بشكل مباشر وفوري إلى كافة البيانات المخزنة على قرص غوغل واستخدامها، أو حتى تخزين النتائج التي تم التوصل لها بعد تنفيذ الرماز المطلوب على السحابة بشكل آني، ويتم وصل قرص السحابة عن طريق التعليمات البسيطة التالية:

```
from google.colab import drive
drive.mount('/content/drive')
```

إن طريقة عمل هذه المنصة تقوم على أن يقوم المستخدم بإنشاء مفكرة جديدة، وعندها ستقوم المنصة بإنشاء نسخة من نظام تشغيل Linux مسبقة بإشارة تعجب "!", وتتيح أيضا إمكانية الانتقال ضمن مجلدات النظام عن طريق استخدام الرمز "%" قبل كل تعليمة "cd"، ويتم تدمير هذه النسخة من النظام بعد انقطاع التفاعل مع المفكرة لبضع ساعات.

وتتميز أيضا بسهولة تحميل واستخدام واستدعاء المكاتب باستخدام تعليمة بسيطة مثال:

```
%pip install python-chess
```

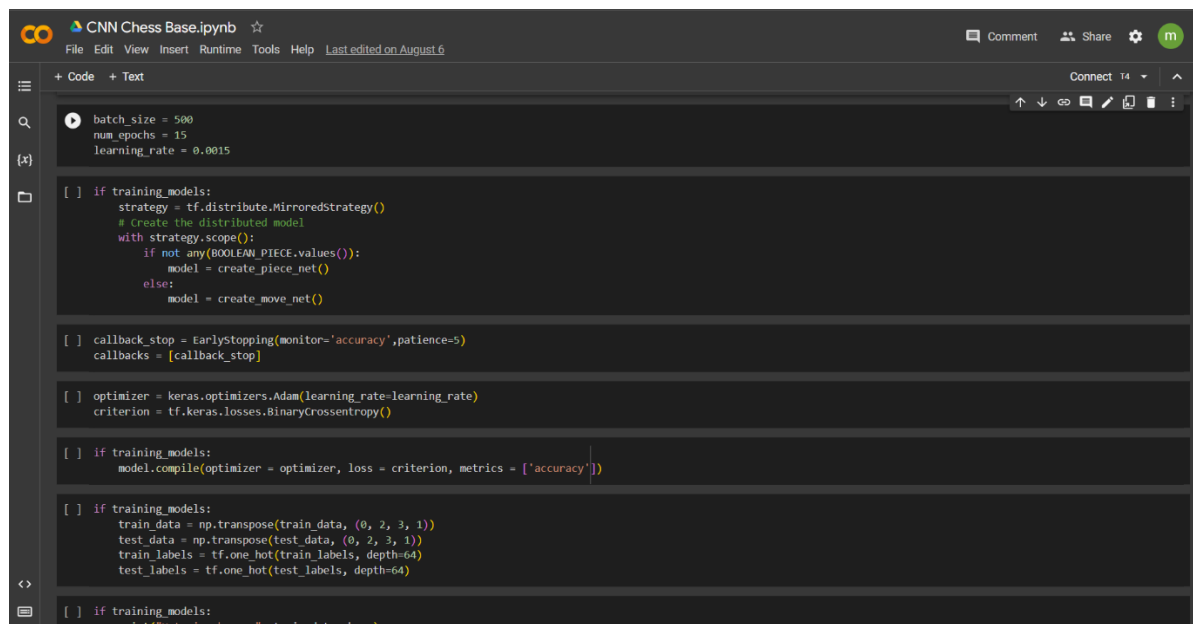
حيث إن نتيجة تنفيذ هذه التعليمة هو تحميل أحدث إصدار من مكتبة "python-chess" وضبط المسارات اللازمة بشكل تلقائي، والأهم من ذلك أن سرعة التحميل مستقلة عن سرعة اتصال الانترنت عند المستخدم، وبالتالي فإن استخدام هذه المنصة هو حل مثالي للمطورين الذين ليس لديهم سرعة اتصال عالية بالشبكة.

كما ويمكن تحميل أي مكتبة بالإصدار الذي يريده المطور عن طريق تحديد رقم الإصدار، مثال:

```
%pip install python-chess==1.1.0
```

بالإضافة إلى إمكانية إظهار رسوم توضيحية بشكل تفاعلي ضمن المتصفح، مثل الرسومات التي تقدمها مكتبة Matplotlib.

ويبين الشكل التالي شكل واجهة بيئة Google.Colab



```
batch_size = 500
num_epochs = 15
learning_rate = 0.0015

[ ] if training_models:
    strategy = tf.distribute.MirroredStrategy()
    # Create the distributed model
    with strategy.scope():
        if not any(BOOLEAN_PIECE.values()):
            model = create_piece_net()
        else:
            model = create_move_net()

[ ] callback_stop = EarlyStopping(monitor='accuracy',patience=5)
    callbacks = [callback_stop]

[ ] optimizer = keras.optimizers.Adam(learning_rate=learning_rate)
    criterion = tf.keras.losses.Binarycrossentropy()

[ ] if training_models:
    model.compile(optimizer = optimizer, loss = criterion, metrics = ['accuracy'])

[ ] if training_models:
    train_data = np.transpose(train_data, (0, 2, 3, 1))
    test_data = np.transpose(test_data, (0, 2, 3, 1))
    train_labels = tf.one_hot(train_labels, depth=64)
    test_labels = tf.one_hot(test_labels, depth=64)

[ ] if training_models:
    print("train shape: ", train_data.shape)
```

شكل 62: شكل واجهات بيئة عمل Google.Colab

كما سمحت القدرات التعاونية لـ Google Colab بالاستفادة من الوصول وتعديل الرماز وتشغيله على عدة حسابات، بحيث أنه تم العمل على الرماز من أكثر من حساب Google، وذلك بسبب محدودية Google.colab لاستخدام GPU لعدد ساعات محددة في اليوم.

2.2.5 - بيئة Jupyter Notebook

تعتبر بيئة العمل Jupyter Notebook من أهم وأشهر البيئات التي تتيح إمكانية بناء رماز بلغة Python ضمن مفكرة Notebook ضمن المتصفح بشكل تفاعلي Interactive وتتيح إمكانية حفظ ونشر هذه المفكرة عن طريق، فمن السهل استخدامه حيث يمكن تشغيل خلية تلو الأخرى للحصول على فهم أفضل لما يفعله الكود، ومن السهل جدا استضافة مخدم عليها على عكس Google.Colab الذي لا يدعم خاصية استضافة مخدم.

تم إنشاء مخدم باستخدام مكتبة Flask لإنشاء واجهة برمجة التطبيقات API الذي يمكن المستخدمين من التواصل والاستفادة من النظام الذكي الخاص بنا.

ومن أهم ميزت هذه المنصة أنها تتيح إمكانية وصل (ربط) مع قرص سحابة غوغل Google Drive، بالتالي فإنه يمكننا الوصول بشكل مباشر وفوري إلى كافة البيانات المخزنة على قرص غوغل واستخدامها، حيث يتم ربط قرص سحابة غوغل مع الجهاز نفسه بحيث يظهر على جهاز الحاسوب الخاص، وذلك عن طريق تنزيل برمجية Google Drive على جهاز الحاسوب ثم تركيب قرص السحابة على أحد أقراص نظام الحاسب، ومن ثم تطبيق التعليمات التالية لتستطيع بيئة العمل Jupyter الوصول لـ Google Drive:

```
mklink /J "C:\Users\ahmadalkattan\Google Drive" "G:/"
```

بشرط أن يكون قرص سحابة غوغل مركب على القرص G.

تم استخدام الخاصية السابقة لسحب آخر نسخة من النماذج المرفوعة على قرص سحابة غوغل عن طريق Google.Colab ليتم استخدامها ضمن API.

3.2.5 - بيئة Visual Studio Code

تم استغلال إمكانيات Visual Studio Code كبيئة تطوير، حيث كان IDE بمثابة منصة قوية لإنشاء مخدم Node.js، وهو أمر محوري في تسهيل واجهة برمجة التطبيقات (API) المصممة لتقديم الصفحة الوب ديناميكيا. كان تكامل مكتبة chess.js مع المخدم، التي هي وحدة Node.js، لا غنى عنه لتنفيذ منطق الشطرنج المعقد بدقة وكفاءة.

في الواجهة الأمامية، تم تنسيق مزيج متناغم من HTML و CSS و JavaScript لإنشاء صفحة وب للعب الشطرنج ضد النظام الذكي. لعبت مكتبة رقعة الشطرنج دورا محوريا في هذا الأمر، حيث سهّلت إنشاء واجهة بديهية وجذابة بصريا للتفاعلات المتعلقة بالشطرنج. مجموعة ميزات Visual Studio Code، بما في ذلك الإكمال التلقائي للكود، ووظائف التصحيح، ومجموعة متنوعة من الإضافات، عجلت بشكل كبير من دورة حياة التطوير، مما أدى إلى مشروع منظم بدقة ويعمل بكامل طاقته.

3.5- المكاتب المستخدمة

تم إثراء بيئة التطوير من خلال تكامل الحزم والمكاتب المختلفة، كل منها ساهم في جوانب مميزة لنظام الشطرنج الذكي:

1.1.3.5 مكتبة chess and chess.pgn

كانت هذه الحزم بمثابة العمود الفقري لبيئة الشطرنج. حيث وفرت هذه الحزم بيئة الشطرنج الأساسية اللازمة لإدارة ومعالجة وتحليل ألعاب الشطرنج المخزنة بتنسيق (PGN) Portable Game Notation دون عناء. وقد لعبت قدراتهم دوراً حيوياً في العديد من الجوانب الرئيسية للمشروع، بما في ذلك إعداد البيانات، والتدريب على النموذج، والمساعدة في اتخاذ القرار داخل شجرة اللعبة.

مكنت مكتبة الشطرنج من تحميل ملفات PGN بكفاءة والتي تحتوي على العديد من ألعاب الشطرنج. حيث تمكن اجتياز الألعاب بسلاسة واستخراج الحركات الفردية والحصول على مواضع اللوحة المقابلة. يسهل هذا المعالجة المسبقة للبيانات، حيث تم تحويل الحركات الأولية إلى مدخلات منظمة مناسبة لتدريب النموذج. من خلال التفاعل مع مكتبة chess.pgn، تم الوصول إلى البيانات الوصفية والتعليقات التوضيحية المرتبطة بالألعاب، مما سمح بتصفية واختيار ألعاب عالية الجودة للتدريب والتحقق من صحتها.

كما سمحت بإنشاء مواضع الشطرنج ومعالجتها وتحليلها داخل عُقد شجرة اللعبة. وتطبيق التحركات واستكشاف الاختلافات وتقييم حالات الرقعة لاتخاذ قرارات حاسمة أثناء عملية البحث.

2.3.5 مكتبة Tensorflow

سهلت Tensorflow التصميم السلس وتكوين بنى الشبكات العصبية المعقدة، المصممة لتعقيدات التنبؤ بحركة الشطرنج. من خلال توافره عالية المستوى وسهولة الاستخدام، كما مكنت من تحديد الطبقات وإنشاء الاتصالات وتخصيص توابع التنشيط بسهولة. كما ساعد التجريد المقدم من tensorflow للتركيز على تصميم بنية النموذج دون الخوض في تفاصيل التنفيذ منخفضة المستوى.

بالاستفادة من خوارزميات التحسين المتقدمة ووظائف الخسارة في Tensorflow، تم ضبط شبكات CNN للتعلم من بيانات موضع الشطرنج. حيث تساعد المكتبة بتحسين معاملات النموذج باستخدام مُحسِّنات مختلف، مثل Adam أو Stochastic Gradient Descent (SGD)، مما يضمن التقارب الفعال وتعزيز الدقة التنبؤية للنموذج.

بمجرد تدريب شبكات CNN، وفرت Tensorflow آليات لتصدير وحفظ أوزان النموذج وبنياته. مكن ذلك من إعادة استخدام النماذج المدربة ونشرها عبر بيئات مختلفة، مما ضمن إمكانية الوصول والتطبيق العملي لنظام الذكاء الاصطناعي.

3.3.5- مكتبة Flask

ساعدت المكتبة في دمج نموذج الشطرنج بسلاسة مع العديد من التطبيقات والمنصات. من خلال إنشاء واجهة برمجة تطبيقات API سهلة الاستخدام، حيث تم تحويل عملية اتخاذ القرار الاستراتيجي للذكاء الاصطناعي إلى خدمة يمكن الوصول إليها واستخدامها من قبل المطورين وحتى دمجها في واجهات العرض Front-end.

ممكن تصميم Flask المعياري من تغليف منطق الذكاء الاصطناعي بطريقة قائمة بذاتها. سهلت هذه الوحدة النمطية صيانة الكود والتحديثات والتحسينات. حيث تسمح للمطورين بدمج قدرات النموذج في تطبيقاتهم بسهولة دون الخوض في التفاصيل المعقدة لتنفيذ النموذج.

4.3.5- مكتبة subprocess

لعبت مكتبة العمليات الفرعية دورا مهما من خلال تمكين التفاعل السلس مع الأدوات الخارجية، وعلى الأخص محرك الشطرنج Stockfish. سهّل هذا التفاعل عملية تحقيق صرامة وساهم في تعزيز دقة الذكاء الاصطناعي وقوته، من خلال الاستفادة من قدرات التقييم المتقدمة ل Stockfish، تم الاستعانة بها من أجل البحث عن الحركة الأفضل التي تعطي تقييما أفضل في الشجرة.

5.3.5- مكاتب NumPy, Panda, Matplotlib, Seaborn

باستخدام Numpy، تم تسخير قدراته العددية لإجراء عمليات حسابية مبسطة على مجموعات البيانات الكبيرة. من ناحية أخرى، مكنت Panda من إدارة وتنظيم هياكل البيانات المعقدة، بما في ذلك التمثيلات الجدولية لحركات الشطرنج والمواضع والبيانات الوصفية للألغاز الشطرنجية. سهلت هذه الحزم بشكل جماعي إعداد وتنظيم مجموعة البيانات لتدريب النموذج.

يعد استخراج رؤى ذات مغزى من البيانات أمرا بالغ الأهمية في تطوير الذكاء الاصطناعي. ساعدت seaborn و matplotlib.pyplot في فهم الأنماط المعقدة والاتجاهات ومقاييس الأداء الناشئة عن تدريب النموذج والبيانات. تم إنشاء رسوم بيانية ومخططات إعلامية تصور تقدم التدريب ومنحنيات الخسارة واتجاهات الدقة على مر التدريب. لم تساعد هذه التصورات في مراقبة تقارب النماذج فحسب، بل قدمت أيضا رؤى لا تقدر بثمن في مجالات التحسين المحتملة.

6.3.5- مكتبة Chessboard.js

هي مكتبة لرقعة الشطرنج مكتوبة بلغة JavaScript. مصممة لتكون "بمجرد لوحة" وتدعم واجهة برمجة تطبيقات API قوية بحيث يمكن استخدامها بطرق مختلفة، من خلال الاستفادة من مكتبة Chessboard، تم تمثيل رقعة الشطرنج بصرياً وعرض تقدم اللعبة. عزز هذا التمثيل المرئي تفاعل المستخدم وسهل تتبع الحركات بسهولة، مما مكن اللاعبين من اللعب ضد نموذج الذكاء الاصطناعي.

7.3.5- مكتبة Chess.js

هي مكتبة شطرنج من نوع TypeScript تُستخدم لمعالجة المنطق للعبة الشطرنج كالتحقق من صحة حركة، ومعالجة الحركات المسموحة، والكشف عن حالات الكش / الكش مات / الموت خنقا، حيث ساعدت المستخدم بالالتزام بقواعد اللعبة.

8.3.5- تابع التقييم لـ Stockfish

تم تنزيل برنامج Stockfish على Google Drive بحيث يمكن تشغيله ضمن بيئة الـ Linux في Google.Colab والذي يبلغ من الحجم 40 ميغابايت، حيث تم تشغيله على بيئة Google.Colab عن طريق التعليمة التالية:

```
!chmod 777 "/content/drive/MyDrive/Chess_data/models/stockfish-ubuntu-x86-64-avx2"
```

بحيث تقوم التعليمة السابقة بإعطاء المستخدم (Colab) الصلاحيات اللازمة لتشغيل البرنامج ثم يمكن تحميله ضمن بيئة Python عن طريق التعليمة التالية:

```
with chess.engine.SimpleEngine.popen_uci(stockfish_path) as engine:
```

ويمكن تقييم موضع معين عن طريق التعليمة التالية:

```
info = engine.analyse(board, chess.engine.Limit(time=time_limit,depth=depth),game=object())  
evaluation = info["score"].relative
```

وهنا يكون تقييم الموضع تعطى بالشكلين التاليين:

Cp (+20) : وهي تعني التقييم موجب أي التقييم منحاز لطرف اللاعب الذي له الدور في اللعب إن كانت القيمة سالبة يكون التقييم منحاز للاعب الآخر وقيمة الانحياز تقاس بوحدة الـ CentiPawn أي كل 100 منها تعادل التقدم ببندق واحد.

(+4) Mate : وهي تعني التقييم موجب والطرف الذي له الدور في اللعب لديه سلسلة من أربع حركات كاملة في هذا المثال (أي 8 نصف حركة) بحيث بعد أن يتم لعب هذه السلسلة يحصل على كش مات على الخصم، ولا يمكن للخصم الهرب من الكش مات حتى وإن لعب أفضل حركة في الموضع الحالي، أما إن لم يلعب أفضل حركة في الموضع الحال يمكن أن تقصر طول السلسلة.

في حالة الشجرة لدينا تعيد العقد الأوراق تقييم التابع في حال كان تقييم الموضع يعيد الحالة الأولى مضروباً بإشارة (-) في حالة كان اللاعب الذي لديه الدور في اللعب صاحب اللون الأسود، أو تعيد العقدة الورقة لانهائية في حالة كانت الحالة الثانية مع الضرب بإشارة (-) في حال كان دور الأسود في اللعب.

4.5- بناء واجهة برمجة التطبيقات API للتفاعل مع النظام

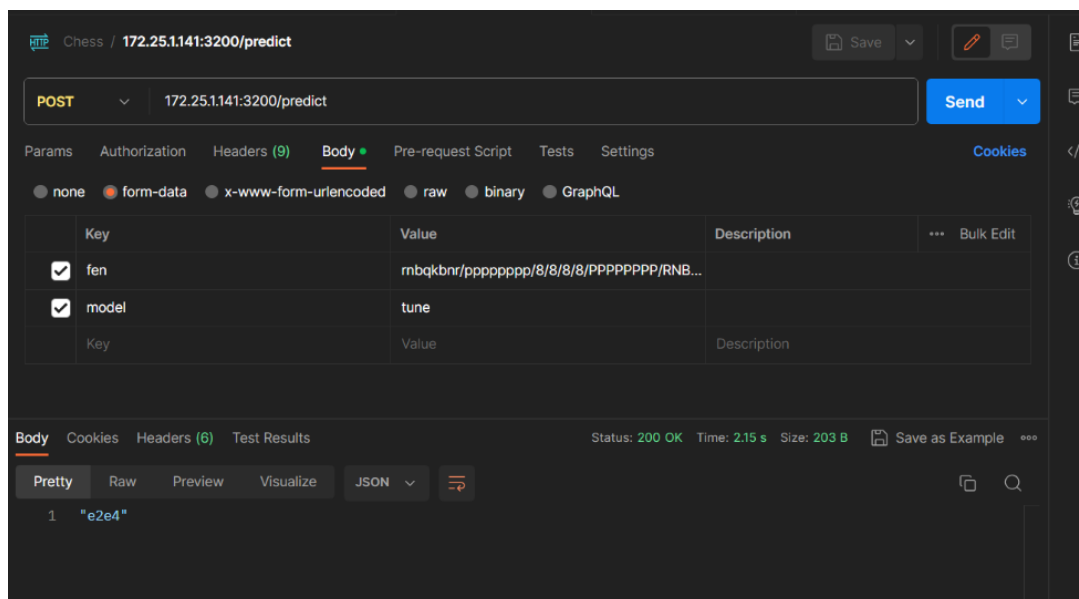
تم بناء واجهة برمجة التطبيقات API باستخدام مكتبة FLASK بلغة Python بحيث يحتوي على تابع Post وحيد /predict/ يحتوي على المعاملات التالية التي يجب إرسالها مع الطلب:

Fen : تدوينة الموضع الشطرنجي المراد توقع أفضل حركة له.

Model : وهو عبارة عن اسم النموذج المراد استخدامه.

وبعيد الطلب ملف json يحتوي على ترميز الحركة الأفضل للموضع المراد.

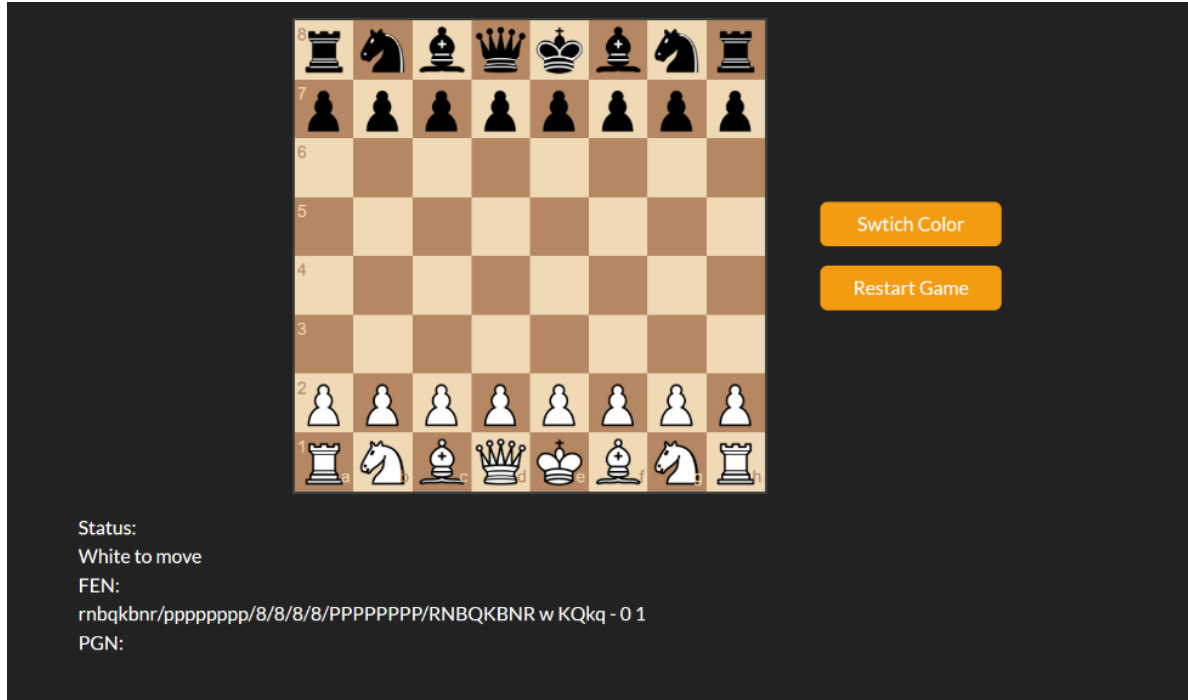
يقوم التابع باستدعاء التابع الذي يقوم بإنشاء شجرة والبحث فيها لإيجاد الحركة الأفضل للموضع المعطى كجذر للشجرة.



شكل 63: مثال عن استخدام ال API باستخدام Postman

5.5- بناء واجهات المستخدم للتفاعل واللعب مع النظام

يحتوي التطبيق على واجهة استخدام وحيدة وهي عبارة عن واجهة اللعب ضد النظام الذكي يوجد خيارات لتحديد الصعوبة المرغوبة للعب وتحديد اللون المراد اللعب به، تظهر على الواجهة ترميز الرقعة الحالية وحالة الرقعة مع كامل الحركات التي تم لعبها في اللعبة الحالية للتمكن من حفظ اللعبة لإجراء الدراسة عليها باستخدام مواقع التحليل.



شكل 64 : شكل واجهة المستخدم

6-الفصل السادس

الخاتمة

يتحدث هذا الفصل عن بعض الصعوبات التي واجهت المشروع وبعض الافاق المستقبلية للمشروع

1.6- الصعوبات

من أكثر الصعوبات والتحديات التي واجهت المشروع هي الذاكرة العشوائية المجانية في Google Colab صغيرة نسبيا بحيث لا يمكن الاستفادة من كامل مجموعات البيانات، حيث أدى ذلك إلى إدخال عملية ال Fine Tuning للنماذج عوضا عن التدريب على كامل البيانات (ألعاب + ألغاز)، كما أدى الوصول المجاني المحدود إلى معالجات Google Colab الرسومية GPU إلى مشاكل في التبديل بين حسابات Google ونقل الملفات مرارا وتكرارا في Google Drive للتمكن من استكمال العمل.

إيجاد طريقة واضحة لتقييم النموذج، كانت من أحد الصعوبات، وذلك بسبب عدم القدرة على الحكم على أن حركة معينة صحيحة وحركة أخرى خاطئة ليس صحيحا بشكل عام، وذلك لأن من أهم تحديات هذه المسألة هو التعامل مع طبيعة اللعبة ذات المتغيرات الكبيرة والاحتمالات المتعددة، مما يجعل الحركات التي يمكن أن تبدو خاطئة في مرحلة معينة قد تتحول إلى خيارات صحيحة في مراحل لاحقة، والعكس صحيح أيضا.

2.6- الافاق المستقبلية

مع استمرار تطور مجال الذكاء الاصطناعي، فإن النهج الهجين للجمع بين الشبكات العصبية التلافيفية (CNNs) وخوارزمية بحث شجرة Alpha-Beta يبشر بالخير. يمكن أن يستكشف البحث المستقبلي مزيدا من التحسينات على بنية CNN، مثل دمج آليات الانتباه Attention Mechanisms أو تقنيات التعلم المعززة Reinforcement Learning للتكيف مع سيناريوهات اللعب الديناميكية. بالإضافة إلى ذلك، قد يؤدي البحث عن طرق لتحسين تكامل CNNs و Alpha-Beta في سيناريوهات اتخاذ القرار في الوقت الفعلي، والتي من المحتمل أن تستفيد من مسرعات الأجهزة، إلى زيادة كفاءة أنظمة الذكاء الاصطناعي للعب الشطرنج، كما ينصح بالتعمق في فكرة إدخال تقييم الموضع إلى مدخلات الشبكة كأحد الميزات وأيضا يمكن التنويه إلى إمكانية تعزيز الميزات المدخلة إلى الشبكة العصبية التي تمثل موضع اللعبة من الناحية الاستراتيجية (أمان الملك،

المساحة الحرة، تطوير القطع، حرية تحرك القطع، هيكلية البيادق، إلخ). علاوة على ذلك، يمكن توسيع المعرفة المكتسبة من هذا البحث لتشمل الألعاب الأخرى المستندة إلى الإستراتيجية ومجالات صنع القرار المعقدة، مما يُظهر التأثير الدائم لهذا الدمج المبتكر.

3.6- الخاتمة

في الختام، فإن دمج الشبكات العصبية التلافيفية (CNNs) مع خوارزمية بحث شجرة Alpha-Beta قد أظهر إمكاناته في صياغة لاعب ذكاء اصطناعي ماهر في لعبة الشطرنج. التعقيد الاستراتيجي الذي يتسم به لعبة الشطرنج، مع التركيز على اتخاذ قرارات معقدة وتفكير مستقبلي، قد قدم مجالاً خصباً لدمج التقنيات المتقدمة مع مبادئ نظرية اللعب الكلاسيكية.

تجسد هذه الأبحاث نتائج مرحلية ناجحة تظهر تكامل التعلم العميق والتفكير الاستراتيجي، مما أسفر عن نظام ذكاء اصطناعي يظهر فهمًا لافئاً لديناميكيات لعبة الشطرنج. تنظيم مجموعات البيانات بعناية وتحويلها، جنباً إلى جنب مع بناء شبكات CNN المتداخلة، يؤكد على أهمية النهج متعدد التخصصات. من خلال فهم عناصر محددة من لعبة الشطرنج، يمكن للذكاء الاصطناعي المهجين التنقل بفعالية في شجرة القرار المعقدة للحركات الممكنة.

تبرز النتائج المقدمة فعالية النموذج المقترح، حيث حقق النظام دقة مذهلة في الحدود الزمنية المرنة والمقيدة على حد سواء. بالإضافة إلى ذلك، قدرة النموذج على التنافس مع محرك الشطرنج Stockfish وتحقيق نتائج واعدة تعزز قوته. تقدير مستوى مهارة النموذج (Elo) بحوالي 1700 يعزز من قدرته التنافسية في عالم الذكاء الاصطناعي للشطرنج.

بالنسبة لما وراء لعبة الشطرنج، تسلط هذه الأبحاث الضوء على الآثار الأوسع لتجميع التقنيات المتقدمة مع النظريات التقليدية. العلاقة التكاملية بين التعلم العميق والتفكير الاستراتيجي، كما تم تجسيدها في نموذج الذكاء الاصطناعي المهجين هذا، تحمل وعداً لتعزيز أنظمة الذكاء الاصطناعي في مجموعة متنوعة من المجالات.

في حين قدمت هذه الأبحاث تقدماً كبيراً في تطوير إمكانيات لاعبي الشطرنج الاصطناعي، هناك مجالات للاستكشاف المستمر. الجهود المستمرة في تحسين أداء النموذج، وتوسيع مجموعات البيانات، واستكشاف خوارزميات جديدة يمكن أن تسهم في تطور مستمر للعبة الذكية.

في ختام الأمر، تعتبر هذه الأبحاث بمثابة شهادة على إمكانية تجميع منهجيات متنوعة لإنشاء لاعب شطرنج اصطناعي متفوق. مع استمرار تطور التكنولوجيا، ستؤثر الدروس المستفادة من هذا الجهد بلا شك على تطوير أنظمة الذكاء الاصطناعي المستقبلية.

المراجع

- [1] : Parashar, A., Jha, A., & Kumar, M. (2022). Analyzing a Chess Engine Based on Alpha–Beta Pruning, Enhanced with Iterative Deepening. In *springer* (pp. 691–700).
- [2] : Oshri, B., & Khandwala, N. (2016). Predicting moves in chess using convolutional neural networks. ConvChess. pdf.
- [3] : Panchal, H., Mishra, S., & Shrivastava, V. (2021, October). Chess moves prediction using deep learning neural networks. In 2021 International Conference on Advances in Computing and Communications (ICACC) (pp. 1-6). IEEE.
- [4] : Lai, M. (2015). Giraffe: Using deep reinforcement learning to play chess. arXiv preprint arXiv:1509.01549.
- [5] : Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815.
- [6] : Arenz, O. (2022). Monte carlo chess (Bachelor's thesis, Technische Universität Darmstadt).
- [7] : Rosemarin, H., & Rosenfeld, A. (2019, September). Playing chess at a human desired level and style. In Proceedings of the 7th International Conference on Human-Agent Interaction (pp. 76-80).
- [8] : فائق دحدوح. (1984). الكامل في الشطرنج. دار النشر طلاس دمشق.
- [9] : Haque, R., Wei, T. H., & Müller, M. (2021). On the road to perfection? Evaluating Leela chess zero against endgame tablebases. In Advances in Computer Games (pp. 142-152). Cham: Springer International Publishing.
- [10]: Parashar, A., Jha, A. K., & Kumar, M. (2022). Analyzing a Chess Engine Based on Alpha–Beta Pruning, Enhanced with Iterative Deepening. In Expert Clouds and Applications: Proceedings of ICOECA 2022 (pp. 691-700). Singapore: Springer Nature Singapore.
- [11] : Tomašev, N., Paquet, U., Hassabis, D., & Kramnik, V. (2022). Reimagining chess with AlphaZero. Communications of the ACM, 65(2), 60-66.

- [12] : McGrath, T., Kapishnikov, A., Tomašev, N., Pearce, A., Wattenberg, M., Hassabis, D., ... & Kramnik, V. (2022). Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47), e2206625119.
- [13] : Stöckl, A. (2021, September). Watching a language model learning chess. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)* (pp. 1369-1379).
- [14] : Dreżewski, R., & Wątor, G. (2021). Chess as sequential data in a chess match outcome prediction using deep learning with various chessboard representations. *Procedia Computer Science*, 192, 1760-1769.
- [15]: chess puzzles: [lichess.org open database https://database.lichess.org/#puzzles](https://database.lichess.org/#puzzles)
- [16]: chess games: [FICS Games Database - Download https://www.ficsgames.org/download.html](https://www.ficsgames.org/download.html)
- [17]: Sabatelli, M., Bidoia, F., Codreanu, V., & Wiering, M. A. (2018, January). Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead. In *ICPRAM* (pp. 276-283).
- [18] : Hesham, N., Abu-Elnasr, O., & Elmougy, S. (2021). A New Action-Based Reasoning Approach for Playing Chess. *Computers, Materials & Continua*, 69(1).
- [19] : Lemley, J., Andonie, R., Odaht, A., Heer, P., Widger, J., Erkul, B., ... & Littlefield, K. (2018, August). CWU-Chess: An Adaptive Chess Program that Improves After Each Game. In *2018 IEEE Games, Entertainment, Media Conference (GEM)* (pp. 1-9). IEEE.
- [20] : Chole, V., & Gadicha, V. (2023). Hybrid fly optimization tuned artificial neural network for AI-based chess playing system. *Multimedia Tools and Applications*, 82(13), 20453-20475.
- [21] : Dreżewski, R., & Wątor, G. (2021). Chess as sequential data in a chess match outcome prediction using deep learning with various chessboard representations. *Procedia Computer Science*, 192, 1760-1769.
- [22] : Gomboc, D., & Shelton, C. R. (2021). Chess endgame compression via logic minimization. In *Advances in Computer Games* (pp. 153-162). Cham: Springer International Publishing.

- [23] : Rosemarin, H., & Rosenfeld, A. (2019, September). Playing chess at a human desired level and style. In Proceedings of the 7th International Conference on Human-Agent Interaction (pp. 76-80).
- [24] : Huang, C. (2018). Research and Analysis on the Search Algorithm Based on Artificial Intelligence About Chess Game. In International Conference on Applications and Techniques in Cyber Security and Intelligence: Applications and Techniques in Cyber Security and Intelligence (pp. 509-517). Springer International Publishing.
- [25] : Patel, M., Pandey, H., Wagh, T., Hujare, A. D., & Dangi, R. (2022, December). Vecma: An advance chess engine. In 2022 IEEE Pune Section International Conference (PuneCon) (pp. 1-6). IEEE.
- [26] : Upasani, N., Gaikwad, A., Patel, A., Modani, N., Bijamwar, P., & Patil, S. (2021, June). Dev-Zero: A Chess Engine. In 2021 International Conference on Communication information and Computing Technology (ICCICT) (pp. 1-6). IEEE.

الخلاصة

الشطرنج هو لعبة استراتيجية للاعبين تُلعب على لوح شطرنج، وهو لوح للعبة يحتوي على 64 مربعاً مرتبة في شبكة 8×8 . لقد قامت التطورات التكنولوجية الحالية بتغيير طريقة التي نلعب بها الشطرنج، حيث أصبحنا نلعب الشطرنج وندرس تقنياتها ونقوم بتحليل اللعبة. التنبؤ بالخطوة التالية في لعبة الشطرنج ليس مهمة سهلة، حيث أن معدل الفروع في الشطرنج وسطياً هو 35. وبالتالي، يجب البحث عن خوارزميات جديدة لإنشاء محرك لعبة مثالي. في هذا البحث، تمت محاولة دمج الشبكات العصبية التلافيفية (CNN) وخوارزمية بحث شجرة Alpha-Beta لإنشاء ذكاء اصطناعي هجين ماهر في لعب الشطرنج. يوفر جوهر الشطرنج، المليء بالتفكير الاستراتيجي واتخاذ القرارات المعقدة، اختباراً مثالياً لدمج التقنيات المتطورة كالشبكات العصبونية مع مبادئ نظرية اللعبة الكلاسيكية كشجرة البحث. يتضمن هذا النهج مجموعات بيانات منسقة ومفلترة بدقة تخضع لعملية تحويل لإنشاء مصفوفات لتكون دخلاً مناسباً للشبكات العصبونية. تم إنشاء سبع شبكات CNN يمكن تقسيمهم إلى مجموعتين، الأولى تهدف إلى تحديد القطعة المراد تحريكها وتحتوي على شبكة وحيدة، والمجموعة الثانية تهدف لتحديد الخلية المراد تحريك القطعة إليها وتضم 6 شبكات كل منها مسؤول عن نوع محدد من القطع. بعد التدريب، يتم استخدام دمج مخرجات الشبكة لتحقيق أكثر الحركات الواعدة. يؤكد هذا البحث على القوة من دمج كل من التعلم العميق والتفكير الاستراتيجي في صياغة الذكاء الاصطناعي المتفوق في الشطرنج، تم التدريب على مجموعتين من البيانات، الأولى ذات الـ 20 ألف لعبة والثانية تحتوي على 140 ألف لغز شطرنجي، وعند تجريب النظام ضد Stockfish 1000-1500 Elo نجد أنه فاز بـ 7 ألعاب وخسر 3 وتعادل 2 من أصل 12 لعبة ونقدر الـ Elo للنموذج الهجين بـ 1700.

Abstract

Chess is a strategy game played by two players on a chessboard, which is a game board with 64 squares arranged in an 8x8 grid. Recent technological developments have changed how we play and study chess. Predicting the next move in chess is not an easy task, as, on average, there are 35 possible moves to consider. This makes it important to find new methods to create a great computer chess player. In this research, we tried combining Convolutional Neural Networks (CNNs) with the Alpha-Beta tree search algorithm to make a clever hybrid artificial intelligence for playing chess. Chess is a perfect game to use these two things together because it needs both smart thinking and understanding patterns. We do this by picking and changing groups of data that the computer can learn from. We make seven groups of computer programs to help us choose the best moves. Some of these groups help us choose which piece to move, and others help us decide where to move that piece. After training these programs, we put together what they say to find the best moves. This research shows that combining these two methods makes a better chess computer. We trained our programs using two sets of examples: one with 20,000 chess games and another with 140,000 chess puzzles. When we tested Our chess program against Stockfish, a strong chess engine, it won 7 games, lost 3, and tied 2 out of 12 games. We estimate that our hybrid model has a skill level (Elo) of around 1700. In conclusion, putting together Convolutional Neural Networks and the Alpha-Beta tree search makes a strong chess computer. This research doesn't just help chess; it also shows how different ways of thinking can work together to make computers smarter in many areas.