Enhancing Chess Reinforcement Learning with Graph Representation

Tomas Rigaux[*]
Kyoto University
Kyoto, Japan
tomas@rigaux.com

Hisashi Kashima
Kyoto University
Kyoto, Japan
kashima@i.kyoto-u.ac.jp

Abstract

Mastering games is a hard task, as games can be extremely complex, and still fundamentally different in structure from one another. While the AlphaZero algorithm has demonstrated an impressive ability to learn the rules and strategy of a large variety of games, ranging from Go and Chess, to Atari games, its reliance on extensive computational resources and rigid Convolutional Neural Network (CNN) architecture limits its adaptability and scalability. A model trained to play on a 19×19 Go board cannot be used to play on a smaller 13×13 board, despite the similarity between the two Go variants. In this paper, we focus on Chess, and explore using a more generic Graph-based Representation of a game state, rather than a grid-based one, to introduce a more general architecture based on Graph Neural Networks (GNN). We also expand the classical Graph Attention Network (GAT) layer to incorporate edge-features, to naturally provide a generic policy output format. Our experiments, performed on smaller networks than the initial AlphaZero paper, show that this new architecture outperforms previous architectures with a similar number of parameters, being able to increase playing strength an order of magnitude faster. We also show that the model, when trained on a smaller 5×5 variant of chess, is able to be quickly fine-tuned to play on regular 8×8 chess, suggesting that this approach yields promising generalization abilities. Our code is available at `https://github.com/akulen/AlphaGateau`.

1 Introduction

In the past decade, combining Reinforcement Learning (RL) with Deep Neural Networks (DNNs) has proven to be a powerful way to design game agents for a wide range of games. Notable achievements include AlphaGo's dominance in Go [17], AlphaZero's human-like style of play in Chess and Shogi [18], and MuZero's proficiency across various Atari games [15]. They use self-play and Monte Carlo Tree Search (MCTS) [6] to iteratively improve their performance, mirroring the way humans learn through experience, or intuition, and game-tree exploration.

Previous attempts to make RL-based chess engines were unsuccessful as the MCTS exploration requires a precise position heuristic to guide its exploration. Handcrafted heuristics such as the ones used in traditional minimax exhaustive tree searches were too simplistic, and lacked the degree of sophistication that the random tree explorations of MCTS expects to be able to more accurately evaluate a complex chess position. By combining the advances in computing powers with the progress of the field of Deep Learning, AlphaZero was able to provide an adequate heuristic in the form of a Deep Neural Network that was able to learn in symbiosis with the MCTS algorithm to iteratively improve itself.

[*]`tomas.rigaux.com`

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

增强棋类强化学习的图表示方法

Tomas Rigaux[*]
Kyoto University
日本京都 tomas@rigaux.com

Hisashi Kashima
Kyoto University 京都, 日本
kashima@i.kyoto-u.ac.jp

摘要

掌握游戏是一项艰巨的任务，因为游戏可以极其复杂，并且在结构上彼此之间仍然存在根本性的不同。虽然AlphaZero算法展示了其学习各种游戏规则和策略的强大能力，从围棋和国际象棋到Atari游戏，但其对大量计算资源和刚性的卷积神经网络（CNN）架构的依赖限制了其适应性和可扩展性。一个训练用于在19×19围棋棋盘上玩游戏的模型不能用于在较小的13×13棋盘上玩游戏，尽管这两种围棋变体之间存在相似性。在本文中，我们专注于国际象棋，并探索使用基于图的游戏状态表示，而不是基于网格的表示，以引入基于图神经网络（GNN）的更通用架构。我们还将经典的图注意网络（GAT）层扩展为包含边特征，以自然地提供通用的策略输出格式。我们在比初始AlphaZero论文中更小的网络上进行的实验表明，这种新架构在参数数量相似的情况下表现出色，能够比以前的架构快一个数量级地提高游戏实力。我们还展示了当模型在较小的5×5国际象棋变体上进行训练时，能够快速微调以在标准8×8国际象棋上玩游戏，这表明这种方法具有令人鼓舞的泛化能力。我们的代码可在https://github.com/akulen/AlphaGateau获取。

1 介绍

在过去的十年里，将强化学习（RL）与深度神经网络（DNNs）相结合，已被证明是设计广泛游戏的游戏代理的一种强大方式。值得注意的成就包括AlphaGo在围棋中的主导地位[17]，AlphaZero在国际象棋和将棋中的人类风格的游戏方式[18]，以及MuZero在各种Atari游戏中的熟练程度[15]。它们使用自我对弈和蒙特卡洛树搜索（MCTS）[6]来逐步提高其性能，这种方式类似于人类通过经验、直觉和游戏树探索来学习的方式。

Previous attempts to制作基于RL的国际象棋引擎是不成功的，因为MCTS探索需要一个精确的位置启发式来引导其探索。传统的手工启发式，如在彻底搜索树搜索中使用的启发式过于简单，缺乏MCTS随机树探索所期望的复杂程度，以更准确地评估复杂的国际象棋位置。通过结合计算能力的进步与深度学习领域的进展，AlphaZero能够提供一种适当的形式，即深度神经网络启发式，在这种启发式与MCTS算法共生学习的过程中，逐步提高自身。

[*]tomas.rigaux.com

However, these approaches rely on rigid, game-specific neural network architectures, often representing games states using grid-based data structures, and process them with Convolutional Neural Networks (CNNs), which limits their flexibility and generalization capabilities. For example, a model trained on a standard 19×19 Go board cannot easily adapt to play on a smaller 13×13 board without significant changes to its internal structure, manual parameter transfer, and retraining, despite the underlying similarity of the game dynamics. This inflexibility is further compounded by the extensive computational resources required for training these large-scale models from scratch for each specific game or board configuration. If it was possible to make a single model train of various variants of a game, and on various games at the same time, it would be possible to speed up the training by starting to learn the fundamental rules on a simplified and smaller variant of a game, before presenting the model with the more complex version. Similarly, if a model learned all the rules of chess, it could serve as a strong starting point to learn the rules of Shogi, for example.

It could be possible to design a more general architecture for games such as Go, where moves can be mapped one-to-one with the board grid, so that a model could still use CNN layers and handle differently sized boards simultaneously, but this solution is no longer feasible when the moves become more complex, including having to move pieces between squares, or even dropping captured pieces back onto the board in Shogi.

Those moves evoke a graph-like structure, where pieces, initially positioned on squares, are moved to different new squares, following an edge between those two squares, or nodes. As such, it is natural to consider basing an improved model on a graph representation, instead of a grid representation. We explore replacing CNN layers with GNN layers to implement that idea, and more specifically consider in this paper attention-based GNN layers, reflecting how chess players usually remember the structures that the pieces form, and how they interact with each other, instead of remembering where each individual piece is placed, when thinking about a position.

Representing moves as edges in a graph also introduces the possibility to link the output policy size with the number of edges, to make the model able to handle different game variants with different move structures simultaneously. To do so, it becomes important to have edge features as well as node features, as they will be used to compute for each edge the equivalent move logit. As the classical attention GNN layer, the Graph Attention Network [19] (GAT) only defines and updates node-features, we propose a novel modification of the GAT layer, that we call Graph Attention neTwork with Edge features from Attention weight Updates (GATEAU), to introduce edge-features. We also describe the full model architecture integrating the GATEAU layer that can handle differently sized input graphs as AlphaGateau.

Our experimental results demonstrate that this new architecture, when implemented with smaller networks compared to the original AlphaZero, outperforms previous architectures with a similar number of parameters. AlphaGateau exhibits significantly faster learning, achieving a substantial increase in playing strength in a fraction of the training time. Additionally, our approach shows promising generalization capabilities: a model trained on a smaller 5×5 variant of chess can be quickly fine-tuned to play on the standard 8×8 chessboard, achieving competitive performance with much less computational effort.

2 Related Work

Reinforcement Learning. AlphaGo [17], AlphaZero [18], MuZero [15], and others have introduced a powerful framework to exploit Reinforcement Learning techniques in order to generate self-play data used to train from scratch a neural network to play a game.

However, those frameworks use rigid neural networks, that have to be specialized for one specific game. As such, the training process requires a lot of computation resources. It is also not possible to reuse the training on one type of game to train for another one, or to start the training on a smaller and simpler variant of the game, before introducing more complexity.

Scalable AlphaZero. In the research of Ben-Assayag and El-Yaniv [2], using Graph Neural Networks has been investigated as a way to solve those issues. Using a GNN-based model, it becomes possible to feed as input differently-sized samples, such as Othello boards of size between 5 and 16, enabling the model to learn how to play in a simpler version of the game.

然而，这些方法依赖于刚性的、针对特定游戏的神经网络架构，通常使用基于网格的数据结构表示游戏状态，并使用卷积神经网络（CNNs）处理这些数据，这限制了它们的灵活性和泛化能力。例如，一个在标准19×19 围棋棋盘上训练的模型，如果不进行其内部结构的重大修改、手动参数转移和重新训练，很难适应在较小的13×13 棋盘上进行游戏，尽管这些游戏的基本游戏动态相似。这种不灵活性进一步受到从头开始为每个特定游戏或棋盘配置训练这些大规模模型所需大量计算资源的影响。如果有可能训练一个单一模型来学习各种游戏的不同变体，甚至同时学习多种游戏，那么通过先在简化且较小的变体游戏中学习基本规则，然后再向模型呈现更复杂的版本，可以加快训练速度。同样，如果一个模型学会了国际象棋的所有规则，它就可以作为一个强大的起点来学习将棋的规则，例如。

它可以设计一种更通用的游戏架构，例如围棋，其中移动可以一对一地映射到棋盘网格，这样模型仍然可以使用CNN层并同时处理不同大小的棋盘，但在移动变得更加复杂时，包括在不同方格之间移动棋子，甚至在将捕获的棋子重新放回棋盘上时，这种解决方案就不再可行了。

那些动作唤起了一种图状结构，其中棋子最初位于方格上，然后按照连接这两个方格的边或节点移动到不同的新方格。因此，自然地考虑基于图表示而不是网格表示建立改进的模型。我们探索用GNN层替换CNN层来实现这一想法，并且在本文中特别考虑基于注意力的GNN层，反映棋手通常如何记住棋子形成的结构以及它们之间的相互作用，而不是记住每个单独棋子的位置。

表示移动作为图中的边也可以引入将输出策略大小与边的数量联系起来的可能性，从而使模型能够同时处理具有不同移动结构的不同游戏变体。为此，除了节点特征外，还需要边特征，因为这些特征将用于计算每条边的等效移动概率。与经典的注意力GNN层——图注意力网络（GAT）[19]仅定义和更新节点特征不同，我们提出了一种新的GAT层的修改版本，我们称之为图注意力网络从注意力权重更新中获取边特征（GATEAU），以引入边特征。我们还描述了包含GATEAU层的完整模型架构，该架构可以处理不同大小的输入图，类似于AlphaGateau。

我们的实验结果表明，与原始AlphaZero相比，使用较小网络实现的新架构在具有相似数量参数的情况下表现出更优的性能。AlphaGateau展示了显著更快的学习速度，在极短的训练时间内实现了显著的棋力提升。此外，我们的方法展示了良好的泛化能力：在较小的5×5变体国际象棋上训练的模型可以快速微调以在标准8×8国际象棋棋盘上进行游戏，并且在较少的计算资源消耗下达到竞争力的性能。

2 相关工作

强化学习。AlphaGo [17]、AlphaZero [18]、MuZero [15] 以及其他方法引入了一种强大的框架，利用强化学习技术生成自对弈数据，用于从头训练一个神经网络来玩棋类游戏。

然而，这些框架使用的是刚性的神经网络，这些网络需要针对一个特定的游戏进行专门化。因此，训练过程需要大量的计算资源。此外，一种类型的游戏的训练无法重用于另一种类型的游戏，也无法先在游戏的较小和较简单的变体上开始训练，然后再引入更多复杂性。

可扩展的AlphaZero。在Ben-Assayag和El-Yaniv [2]的研究中，使用图神经网络已被探索作为一种解决这些问题的方法。基于GNN的模型使得能够输入不同大小的样本，例如大小在5到16之间的Othello棋盘，从而使模型能够学习如何在游戏的简化版本中进行游戏。

Algorithm 1: Self-Play Training

```
Parameters: $N_{iter} = 100, N_{games} = 256, N_{sim} = 128, ws = 10^6, N_{train} = 1, bs = 2048$
$\theta \leftarrow$ model.init();
for $i \leftarrow 1$ to $N_{iter}$ do
    data ← selfplay($\theta, N_{games}, N_{sim}$) ;            /* We generate self-play data */
    frame_window ← (data||frame_window)[$1 : ws$] ;  /* The new frame window consists of
     the newly generated data and an uniform sample of the previous window */
    for $j \leftarrow 1$ to $N_{train}$ do
        frame_window ← frame_window.shuffle();
        for batch in frame_window.batches($bs$) do
            $\theta \leftarrow$ apply($\theta$, gradient($\theta$, batch));
        end
    end
end
```

This approach had promising results, with 10 times faster training time than the AlphaZero baseline. It was however limited to Othello and Gomoku, and using the GNN layers (GIN layers [20]) only as a scalable variant of CNN layers, keeping a rigid grid structure.

Edge-featured GNNs. There exists a large variety of GNN variants, specialized for different use case s and data properties. For this work, a simple layer was enough to experiment with the merits of the proposed approach, except it was critical that the chosen layer handled both node-features and edge-features. We chose to use an attention-based layer.

Gong and Cheng [10] introduce the EGNN(a) layer, where each dimension of an edge-feature is used for a different attention head. We wanted edge features to be treated as a closer equivalent to node-features, so we did not use this layer.

The EGAT layer introduced by Chen and Chen [4] is better for our case, as they construct a dual graph where edges and nodes have reversed roles, so the node features in the dual graph are edge features for the initial graph. However this method requires building the dual graph, and is quadratic in the maximal node degree. As this was quite complex, we decided to introduce GATEAU, which solves the problem in a simpler and more natural way.

3 Setting

Our architecture is based on the AlphaZero framework, which employs a neural network f_θ with parameters θ that is used as an oracle for a Monte-Carlo Tree Search (MCTS) algorithm to generate self-play games. When given a board state s, the neural network predicts a (value, policy) pair $(v(s), \pi(s, \cdot)) = f_\theta(s)$, where the value $v \in [-1, 1]$ is the expected outcome of the game, and the policy π is a probability distribution over the moves.

We utilize Algorithm 1 to train the models in this paper, with modifications to incorporate Gumbel MuZero [7] with a gumbel scale of 1.0 as our MCTS variant.

4 Proposed Models

4.1 Motivation: Representing the Game State as a Graph

Many games, including chess, are not best represented as a grid. For example, chess moves are analogous to edges in a grid graph, and games like Risk naturally form planar graphs based on the map. As such, it makes natural sense to encode more information through graphs in the neural network layers that are part of the model.

This research focuses on implementing this idea in the context of chess. This requires to answer two questions: how to represent a chess position as a graph, and how to output a policy vector that is edge-based, and not node-based.

The architecture presented in this paper is based on GNNs, but using node features to evaluate the value head, and edge features to evaluate the policy head. As such, a GNN layer that is able to handle

Algorithm 1: Self-Play Training

```
Parameters: $N_{iter} = 100, N_{games} = 256, N_{sim} = 128, ws = 10^6, N_{train} = 1, bs = 2048$
$\theta \leftarrow$ model.init();
for $i \leftarrow 1$ to $N_{iter}$ do
    data $\leftarrow$ selfplay($\theta, N_{games}, N_{sim}$) ;                /* We generate self-play data */
    frame_window $\leftarrow$ (data||frame_window)$[1 : ws]$ ;  /* The new frame window consists of
     the newly generated data and an uniform sample of the previous window */
    for $j \leftarrow 1$ to $N_{train}$ do
        frame_window $\leftarrow$ frame_window.shuffle();
        for batch in frame_window.batches($bs$) do
            $\theta \leftarrow$ apply($\theta$, gradient($\theta$, batch));
        end
    end
end
```

这种方法取得了有希望的结果，比AlphaZero基线快10倍的训练时间。然而，这种方法仅限于将围棋和五子棋，且仅使用GNN层（GIN层[20]）作为CNN层的可扩展变体，保持了刚性的网格结构。

Edge-特征 GNNs。存在大量不同用途和数据属性专门化的 GNN 变体。对于这项工作，一个简单的层已经足够用来探索所提出方法的优点，除非选择的层能够处理节点-特征和边-特征。我们选择使用基于注意力的层。

Gong和Cheng [10] 引入了EGNN(a)层，其中每条边特征的每个维度用于不同的注意力头。我们希望边特征能更接近节点特征的等价物，因此没有使用这一层。

EGAT 层由 Chen 和 Chen [4] 引入，更适合我们的应用场景，因为他们构建了一个双图，在这个双图中，边和节点的角色被反转，因此双图中的节点特征对应于初始图中的边特征。然而，这种方法需要构建双图，并且其复杂度与最大节点度成二次关系。由于这相当复杂，我们决定引入 GATEAU，它以一种更简单且更自然的方式解决了这个问题。

3 设置

我们的架构基于AlphaZero框架，该框架采用了一个具有参数θ的神经网络f_θ，作为蒙特卡洛树搜索（MCTS）算法的先验知识，用于生成自对弈游戏。当给定一个棋盘状态s时，神经网络预测一个（价值，策略）对$(v(s), \pi(s, \cdot)) = f_\theta(s)$，其中价值$v \in [-1, 1]$是游戏的预期结果，而策略$\pi$是动作的概率分布。

我们利用算法 1 训练本文中的模型，并对其实行修改，以结合具有 gumbel 规模 1.0 的 Gumbel MuZero [7] 作为我们的 MCTS 变体。

4 提出的模型

4.1 动机：将游戏状态表示为图

许多游戏，包括国际象棋，都不适合用网格来最佳表示。例如，国际象棋的走法类似于网格图中的边，而像风险这样的游戏则自然地基于地图形成了平面图。因此，在模型中作为神经网络层的一部分，通过图来编码更多信息是合乎逻辑的选择。

This research 着重于将这一理念应用于国际象棋的上下文。这需要回答两个问题：如何将国际象棋的局面表示为一个图，以及如何输出一个基于边的策略向量，而不是基于节点的。

本文介绍的架构基于GNNs，但使用节点特征来评估价值头，使用边特征来评估策略头。因此，这是一种能够处理

Table 1: Node features

Dimensions	Description
12	The piece on the square, as a 12-dimensional soft one-hot
2	Whether the position was repeated before
98	For each of the previous 7 moves, we repeat the last 14 dimensions to describe the corresponding positions
1	The current player
1	The total move count
4	Castling rights for each player
1	The number of moves with no progress

Table 2: Edge features

Dimensions	Description
1	Is this move legal in the current position?
2	How many squares {to the left/up} does this edge move?
4	Would a pawn promote to a {knight/bishop/rook/queen} if it did this move?
2	Could a {white/black} pawn do this move?
4	Could a {knight/bishop/rook/queen} do this move?
2	Could a {white/black} king do this move?

both node and edge features is required. This paper will introduce the GATEAU layer, that is a natural extension of the GAT layer [19] to edge features.

4.2 Graph Design

In AlphaZero, a chess position is encoded as a $n \times n \times 119$ matrix, where each square on the $n \times n$ chess board is associated to a feature vector of size 119, containing information about the corresponding square for the current position, as well as the last 7 positions, as described in Table 1.

We will instead represent the board state as a graph $G(V, E)$, with the $n \times n$ squares being nodes V, and the edges E being moves, based on the action design of AlphaZero. Each AlphaZero action is a pair (source square, move), with $n \times n$ possible source squares. In 8×8 chess, the 73 moves (resp. 49 in 5×5 chess) are divided into 56 queen moves (resp. 32), 8 knight moves (resp. 8), and 9 underpromotions (resp. 9) for a total of 4672 actions (resp. 1225). The edge associated with an action is oriented from the node corresponding to the source square, to the destination square of the associated move. In 8×8 chess, castling is represented with the action going from the king's starting square going laterally 2 squares. As this action encoding is a little too large, containing moves ending outside of the board that do not correspond to real edges, the constructed graph only contains 1858 edges (resp. 455), corresponding only to valid moves.

The node and edge features, of initial size 119 and 15, are detailed in Tables 1 and 2, respectively. Node features are based on AlphaZero's features, including piece type, game state information, and historical move data. Edge features encode move legality, direction, potential promotions, and piece-specific move capabilities. In the case of 5×5 chess, we include the unused castling information, in order to have the same vector size of the 8×8 models. It would be possible to preprocess the node and edge features differently for different games or variants, but for simplicity we didn't do it.

The starting positions for all games played in our experiments were either the classical board setup in 8×8 chess, or the Gardner setup for 5×5 chess, illustrated in Figure 1.

4.3 GATEAU: A New GNN Layer, with Edge Features

The Graph Attention Network layer introduced by Veličković et al. [19] updates the node features by averaging the features of the neighboring nodes, weighted by some attention coefficient. To be more precise, given the node features $h \in \mathbb{R}^{N \times K}$, attention coefficients are defined as

$$e_{ij} = W_u h_i + W_v h_j \tag{1}$$

with parameters $W_u, W_v \in \mathbb{R}^{K \times K'}$. In the original paper, $W_u = W_v$, but as we are working with a directed graph, we differentiate them to treat the source and destination node asymmetrically. Then

表 1: 节点特征

Dimensions	Description
12	The piece on the square, as a 12-dimensional soft one-hot
2	Whether the position was repeated before
98	For each of the previous 7 moves, we repeat the last 14 dimensions to describe the corresponding positions
1	The current player
1	The total move count
4	Castling rights for each player
1	The number of moves with no progress

Table 2: Edge features

Dimensions	Description
1	Is this move legal in the current position?
2	How many squares {to the left/up} does this edge move?
4	Would a pawn promote to a {knight/bishop/rook/queen} if it did this move?
2	Could a {white/black} pawn do this move?
4	Could a {knight/bishop/rook/queen} do this move?
2	Could a {white/black} king do this move?

都需要节点和边特征。本文将介绍 GATEAU 层，这是 GAT 层 [19] 对边特征的自然扩展。

4.2 图设计

在AlphaZero中，一个国际象棋位置被编码为一个$n\times n\times 119$矩阵，其中每个棋盘上的$n\times n$方格关联到一个大小为119的特征向量，包含当前位置以及过去7个位置的对应方格信息，如表1所述。

我们将棋盘状态表示为一个图$G(V,E)$，其中$n\times n$个方格是节点V，边E表示移动，基于AlphaZero的动作设计。每个AlphaZero动作是一对（起始方格，移动），有$n\times n$个可能的起始方格。在8×8国际象棋中，73个移动（分别在5×5国际象棋中有49个）被分为56个皇后移动（分别有32个），8个骑士移动（分别有8个），以及9个升变（分别有9个），总共4672个动作（分别有1225个）。与一个动作相关的边是从对应起始方格的节点指向关联移动的目的方格。在8×8国际象棋中，王车易位用一个动作从国王的起始方格横向移动2格来表示。由于这种动作编码稍微太大，包含了一些结束于棋盘外且不对应真实边的实际移动，因此构建的图只包含1858条边（分别有455条），仅对应于有效的移动。

节点和边特征，初始大小分别为119和15，分别详细列于表1和表2。节点特征基于AlphaZero的特征，包括棋子类型、游戏状态信息和历史走法数据。边特征编码了走法合法性、方向、潜在的升变以及棋子特定的走法能力。在5×5国际象棋的情况下，我们包括未使用的易位信息，以使向量大小与8×8模型相同。对于不同的游戏或变体，可以预先处理节点和边特征，但为了简化，我们没有这样做。

在我们实验中所有游戏的起始位置要么是标准的8×8国际象棋布局，要么是5×5加德纳棋的布局，如图1所示。

4.3 GATEAU:一种新的GNN层，带有边特征

The Graph Attention Network 层由 Veličković 等人 [19] 引入，通过加权平均邻节点特征来更新节点特征。更准确地说，给定节点特征 $h\in\mathbb{R}^{N\times K}$，加权系数定义为

$$e_{ij} = W_u h_i + W_v h_j \tag{1}$$

带有参数 $W_u, W_v \in \mathbb{R}^{K\times K'}$。在原始论文中，$W_u = W_v$，但由于我们处理的是有向图，我们对它们进行了区分以不对称地处理源节点和目标节点。然后

we can compute attention weights α, and use them to update the node features:

$$\alpha^0_{ij} = \mathrm{softmax}_j\left(\mathrm{LeakyReLU}\left(a^T e_{i\cdot}\right)\right) = \frac{\exp^{\mathrm{LeakyReLU}\left(a^T e_{ij}\right)}}{\sum_k \exp^{\mathrm{LeakyReLU}\left(a^T e_{ik}\right)}}, \tag{2}$$

$$h'_i = \sum_{j \in \mathcal{N}_i} \alpha^0_{ij} W h_j \tag{3}$$

with parameters $W \in \mathbb{R}^{K \times K''}$ and $a \in \mathbb{R}^{K'}$.

The main observation motivating GATEAU is that in this process, the attention coefficients e_{ij} serve a role similar to node features, being a vector encoding some information between nodes i and j. As such, we propose to introduce edge features in place of those attention coefficients.

Our proposed layer, called Graph Attention neTwork with Edge features from Attention weight Updates (GATEAU) takes the node features $h \in \mathbb{R}^{N \times K}$ and edge features $g_{i,j} \in \mathbb{R}^{N \times N \times K'}$ as inputs. We start by simply updating the edge features similarly to Eq. 1:

$$g'_{ij} = W_u h_i + \underline{W_e g_{ij}} + W_v h_j \tag{4}$$

with parameters $W_u, W_v \in \mathbb{R}^{K \times K'}$ and $W_e \in \mathbb{R}^{K' \times K'}$. Then the attention weights are obtained as in Eq. 2, by substituting the attention coefficients with our new edge features:

$$\alpha_{ij} = \mathrm{softmax}_j\left(\mathrm{LeakyReLU}\left(a^T g'_{i\cdot}\right)\right) \tag{5}$$

with parameter $a \in \mathbb{R}^{K'}$. Finally, we update the node features as in Eq. 3:

$$h'_i = W_0 h_i + \sum_{j \in \mathcal{N}_i} \alpha_{ij}\left(W_h h_j + W_g g_{ij}\right) \tag{6}$$

with parameters $W_0, W_h \in \mathbb{R}^{K \times K''}$ and $W_g \in \mathbb{R}^{K' \times K''}$. We add the self-edges manually as it is inconvenient for the policy head if they are included in the graph, and we mix back the values of the edge features back in the node features.

4.4 AlphaGateau: A Full Model Architecture Based on AlphaZero and GATEAU

Following the structure of the AlphaZero neural network, we introduce AlphaGateau, combining AlphaZero with GATEAU instead of CNN layers, and redesign the value and policy head to be able to exploit node and edge features respectively to handle arbitrarily sized inputs with the same number of parameters.

We define the following layers, which are used to describe AlphaGateau in Figure 2.

Attention Pooling. In order to compute a value for a given graph, we need to pool the features together. Node features seem to be the more closely related to positional information, so we pool them instead of edge features. For this, we use an attention-based pooling layer, similar to the one described in Eq. 7 by Li et al. [14], which, for node features $h \in \mathbb{R}^{N \times K}$ and a parameter vector $a \in \mathbb{R}^K$, outputs

$$\begin{aligned} \alpha^p_i &= \mathrm{softmax}_i\left(\mathrm{LeakyReLU}\left(a^T h_\cdot\right)\right), \\ H &= \sum_i \alpha^p_i h_i, \end{aligned} \tag{7}$$

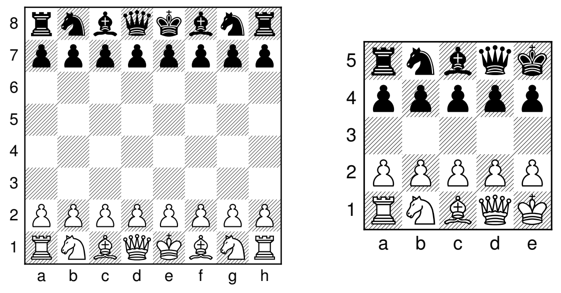

Figure 1: The starting positions of 8×8 and 5×5 chess games

我们可以计算注意力权重 α，并使用它们来更新节点特征：

$$\alpha_{ij}^0 = \mathrm{softmax}_j\left(\mathrm{LeakyReLU}\left(a^T e_{i\cdot}\right)\right) = \frac{\exp^{\mathrm{LeakyReLU}\left(a^T e_{ij}\right)}}{\sum_k \exp^{\mathrm{LeakyReLU}\left(a^T e_{ik}\right)}}, \tag{2}$$

$$h_i' = \sum_{j\in\mathcal{N}_i} \alpha_{ij}^0 W h_j \tag{3}$$

带有参数 $W \in \mathbb{R}^{K\times K''}$ 和 $a \in \mathbb{R}^{K'}$。

The main observation motivating GATEAU is that in this process, the attention coefficients e_{ij} ser ve a role similar to node features, being a vector encoding some information between nodes i and j . As such, we propose to introduce edge features in place of those attention coefficients.

我们提出的层称为Graph Attention neTwork with Edge features from Attention weight Updates (GATEAU)，它以节点特征$h \in \mathbb{R}^{N\times K}$和边特征$g_{i,j} \in \mathbb{R}^{N\times N\times K'}$作为输入。我们首先像等式1那样简单地更新边特征：

$$g_{ij}' = W_u h_i + \underline{W_e g_{ij}} + W_v h_j \tag{4}$$

带有参数 $W_u, W_v \in \mathbb{R}^{K\times K'}$ 和 $W_e \in \mathbb{R}^{K'\times K'}$。然后通过用我们的新边特征替换注意力系数来获得注意力权重，如 Eq. 2 所示：

$$\alpha_{ij} = \mathrm{softmax}_j\left(\mathrm{LeakyReLU}\left(a^T g_{i\cdot}'\right)\right) \tag{5}$$

带有参数 $a \in \mathbb{R}^{K'}$。最后，我们更新节点特征如公式 3 所示：

$$h_i' = W_0 h_i + \sum_{j\in\mathcal{N}_i} \alpha_{ij}\left(W_h h_j + W_g g_{ij}\right) \tag{6}$$

带有参数 $W_0, W_h \in \mathbb{R}^{K\times K''}$ 和 $W_g \in \mathbb{R}^{K'\times K''}$。我们手动添加自边，因为在图中包含自边会给策略头带来不便，我们会将边特征的值重新混合到节点特征中。

4.4 AlphaGateau: 一个基于AlphaZero和GATEAU的完整模型架构

遵循AlphaZero神经网络的结构，我们引入AlphaGateau，将AlphaZero与GATEAU结合替代C NN层，并重新设计价值头和策略头，分别利用节点和边特征以处理任意大小的输入同时保持相同数量的参数。

我们定义了以下层，用于描述图2中的AlphaGateau。

Attention 汇聚。为了计算给定图的价值，我们需要将特征聚合起来。节点特征似乎与位置信息更相关，所以我们使用节点特征 $h \in \mathbb{R}^{N\times K}$ 而不是边特征 $a \in \mathbb{R}^{K}$ 进行汇聚。为此，我们使用一种基于注意力的汇聚层，类似于 Li 等人 [14] 在 Eq. 7 中描述的，该层对于节点特征 $h \in \mathbb{R}^{N\times K}$ 和一个参数向量 $a \in \mathbb{R}^{K}$，输出

$$\begin{aligned} \alpha_i^p &= \mathrm{softmax}_i\left(\mathrm{LeakyReLU}\left(a^T h_\cdot\right)\right), \\ H &= \sum_i \alpha_i^p h_i, \end{aligned} \tag{7}$$

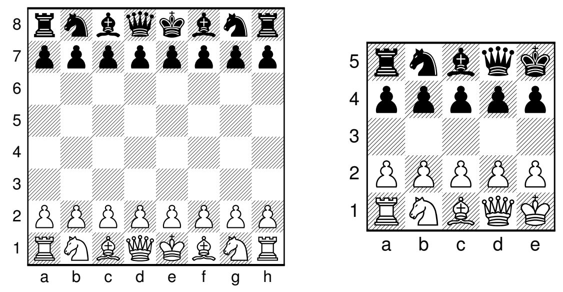

图 1: 8 × 8 和 5 × 5 棋局的起始位置

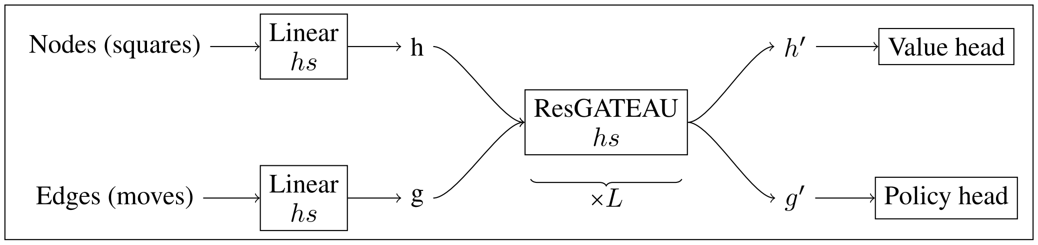

Figure 2: The AlphaGateau network, hs is the inner size of the feature vectors, and L is the number of residual blocks.

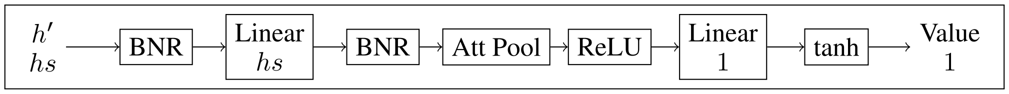

Figure 3: Value head

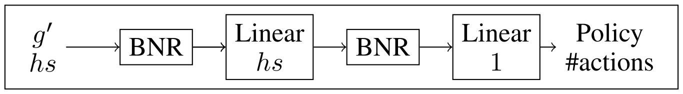

Figure 4: Policy head

where $H \in \mathbb{R}^K$ is a global feature vector.

Batch Normalization and Non-linearity (BNR). As they are a pair of operations that often occur, we group Batch Normalization and a ReLU layer together under the notation BNR:

$$\mathrm{BNR}(x) = \mathrm{ReLU}(\mathrm{BatchNorm}(x)). \tag{8}$$

Residual GATEAU (ResGATEAU). Mirroring the AlphaZero residual block architecture, we introduce ResGATEAU, which similarly sums a normalized output from two stacked GATEAU layers to the input:

$$\mathrm{ResGATEAU}(h, g) = (h, g) + \mathrm{GATEAU}(\mathrm{BNR}(\mathrm{GATEAU}(\mathrm{BNR}(h, g)))). \tag{9}$$

5 Experiments

We evaluate AlphaGateau's performance in learning regular 8×8 chess from scratch and generalizing from a 5×5 variant to the standard 8×8 chessboard. The metric used to evaluate the models is the Elo rating, calculated through games played against other models (or players) with similar ratings. Due to computational constraints, we couldn't replicate the full 40 residual layers used in the initial AlphaZero paper, and experimented with 5 and 6 layer models. We also started exploring 8 layers, but these models required to generate a lot more data, which would make the experiment run an order of magnitude longer. Our results indicate that AlphaGateau learns significantly faster than a traditional CNN-based model with similar structure and depth, and can be efficiently fine-tuned from 5×5 to 8×8 chess, achieving competitive performance with fewer training iterations. [2] All models used in these experiments are trained with the Adam optimizer [12] with a learning rate of 0.001. All feature vectors have an embedding dimension of 128. The loss function is the same as for the original AlphaZero, which is, for $f_\theta(s) = \tilde{\pi}, \tilde{v}$,

$$L(\pi, v, \tilde{\pi}, \tilde{v}) = -\pi^T \log(\tilde{\pi}) + (v - \tilde{v})^2. \tag{10}$$

[2]In Silver et al.[18], the training of AlphaZero is described in terms of steps, which consists of one mini-batch of size 4096, while the generation of games through self-play is done in parallel by other TPUs. In this experiment, an iteration consists of generating 256 games through self-play, then doing one epoch of training, split into 3904 mini-batches of size 256, after 7 iterations once the frame window is full. In terms of positions seen, one iteration is equivalent to 244 steps.

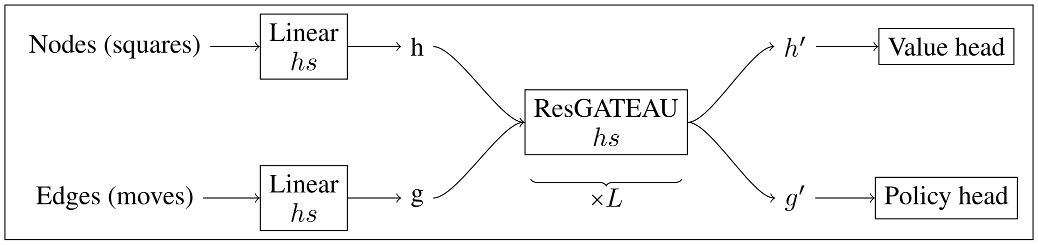

图 2：AlphaGateau 网络，hs 是特征向量的内尺寸，L 是残差块的数量。

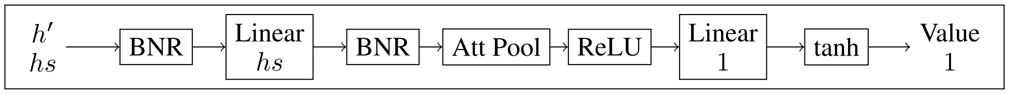

Figure 3: Value head

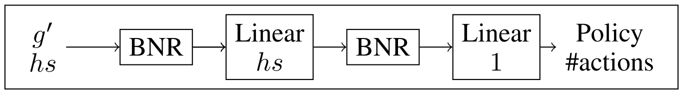

图4: 策略头

where $H \in \mathbb{R}^K$是全局特征向量。

批标准化和非线性（BNR）。由于它们是一对经常同时出现的操作，我们将批标准化和ReLU层组合在一起，用BNR表示：

$$\mathrm{BNR}(x) = \mathrm{ReLU}(\mathrm{BatchNorm}(x)). \tag{8}$$

Residual GATEAU (ResGATEAU). 类似于AlphaZero的残差块架构，我们引入了ResGATEAU，它同样将两个堆叠的GATEAU层的归一化输出与输入相加：

$$\mathrm{ResGATEAU}(h, g) = (h, g) + \mathrm{GATEAU}(\mathrm{BNR}(\mathrm{GATEAU}(\mathrm{BNR}(h, g)))). \tag{9}$$

5 实验

我们评估了AlphaGateau从头学习标准8×8国际象棋以及从5×5变体泛化到标准8×8棋盘的性能。用于评估模型的指标是Elo等级分，通过与其他具有相似等级分的模型（或玩家）进行对局来计算。由于计算资源的限制，我们无法复制初始AlphaZero论文中使用的全部40个残差层，并尝试了5层和6层模型。我们还开始探索8层模型，但这些模型需要生成更多的数据，这会使实验运行时间增加一个数量级。我们的结果显示，AlphaGateau比具有相似结构和深度的传统CNN模型学习速度快得多，并且可以从5×5棋盘高效地微调到8×8棋盘，通过较少的训练迭代就能达到竞争力的性能。[2]这些实验中使用的所有模型都是使用Adam优化器[12]（学习率为0.001）训练的。所有特征向量的嵌入维度为128。损失函数与原始AlphaZero相同，即对于$f_\theta(s) = \tilde{\pi}, \tilde{v}$，

$$L(\pi, v, \tilde{\pi}, \tilde{v}) = -\pi^T \log(\tilde{\pi}) + (v - \tilde{v})^2. \tag{10}$$

[2]In Silver et al.[18], the training of AlphaZero is described in terms of steps, which consists of one mini-batch of size 4096, while the generation of games through self-play is done in parallel by other TPUs. In this experiment, an iteration consists of generating 256 games through self-play, then doing one epoch of training, split into 3904 mini-batches of size 256, after 7 iterations once the frame window is full. In terms of positions seen, one iteration is equivalent to 244 steps.

5.1 Implementation

Jax and PGX. As the MCTS algorithm requires a lot of model calls weaved throughout the tree exploration, it is essential to have optimized GPU code running both the model calls, and the MCTS search. In order to leverage the MCTX [8] implementations of Gumbel MuZero, all our models and experiments were implemented in Jax [3] and its ecosystem [11] [9]. PGX [13] was used for the chess implementation, and we based our AlphZero implementation on the PGX implementation of AZNet. We used Aim [1] to log all our experiments.

To estimate the Elo ratings, we use the statsmodels package [16] implementation of Weighted Least Squares (WLS).

Hardware. All our models were trained using multiple Nvidia RTX A5000 GPUs (*Learning speed* used 8 and *Fine-tuning* used 6), and their Elo ratings were estimated using 6 of those GPUs.

5.2 Evaluation

As each training and evaluation lasted a little under a week, we were not able to train each model configuration several times so far. As such, each model presented in the results was trained only once, and the confidence intervals that we include are on the Elo rating that we estimated for each of them, as described in the following.

During training, at regular intervals (each 2, 5, or 10 iterations), the model parameters were saved, and we used this dataset of parameters to evaluate Elo ratings. In this section, we will call a pair (model, parameters) a player, and compute a rating for every player.

We initially chose 10 players, and simulated 60 games between each pair of players, to get initial match data M. For each pair of players that played a match, we store the number of wins w_{ij}, draws d_{ij}, and losses l_{ij}: $M_{ij} = (w_{ij}, d_{ij}, l_{ij})$. Using this data, we can roughly estimate the ratings $r \in \mathbb{R}^{N_{players}}$ of the players present in M using a linear regression on the following set of equations:

$$\begin{cases} r_j - r_i & = \dfrac{400}{\log(10)} \log\left(\dfrac{w_{ij} + d_{ij} + l_{ij} + 1}{w_{ij} + \frac{d_{ij}+1}{2}} - 1 \right) \quad \text{for } i \in M, j \in M_i, \\ \sum_{i \in M} r_i & = |M| \times 1000. \end{cases} \tag{11}$$

We artificially add one draw to avoid extreme cases where there are only wins for one player and no losses, in which case the rating difference would theoretically be infinite. This is equivalent to a Jeffreys prior. The last equation fixes the average rating to 1000, as the Elo ratings are collectively invariant by translation.

We then ran Algorithm 2 to generate a densely connected match data graph, where each player played against at least 5 other players of similar playing strength. Finally, we used this dataset to fit a linear regression model (Weighted Least Squares) to get Elo ratings that we used in the results figures for the experiments. The confidence intervals were estimated by assuming that the normalized match outcomes followed a Gaussian distribution. If $\hat{p}_{ij} = \frac{w_{ij} + \frac{d_{ij}+1}{2}}{w_{ij}+d_{ij}+l_{ij}+1}$ is the estimated probability that player i beats player j, we approximate the distribution that p_{ij} follows as a Gaussian, and using the delta method, we derive that $r_j - r_j$ asymptotically follows a Gaussian distribution of mean $\frac{400}{\log(10)} \log\left(\frac{1}{p_{ij}} - 1\right)$ and variance $\left(\frac{400}{\log(10)}\right)^2 \frac{1}{(w_{ij}+d_{ij}+l_{ij})p_{ij}(1-p_{ij})}$. The proof is detailed in Appendix A.2. Using the WLS linear model of statsmodels [16], we get Elo ratings for every player, as well as their standard deviations, which we use in the following to derive 2-sigma confidence intervals.

5.3 Experiments

Learning Speed. Our first experiment compares the baseline ability of AlphaGateau to learn how to play 8×8 chess from scratch, and compares it with a scaled down AlphaZero model. The AlphaZero model has 5 residual layers (containing 10 CNN layers) and a total of 2.2M parameters, and the AlphaGateau model also has 5 ResGATEAU layers (containing 10 GATEAU layers) and a total of 1.0M parameters, as it doesn't need a $N_{nodes} \times hs \times N_{actions}$ fully connected layer in the policy head, and the GATEAU layers use 2/3 of the parameters a 3×3 CNN does.

5.1 实施

Jax和PGX。由于MCTS算法在树探索过程中需要大量模型调用，因此必须拥有优化的GPU代码，同时运行模型调用和MCTS搜索。为了利用MCTX [8]中Gumbel MuZero的实现，我们所有的模型和实验都在Jax [3]及其生态系统 [11] [9]中实现。PGX [13]用于象棋实现，我们的AlphaZero实现基于PGX的AZNet实现。我们使用Aim [1]记录所有实验。

为了估计Elo评分，我们使用了statsmodels包[16]中实现的加权最小二乘法（WLS）。

硬件。所有模型都是使用多个Nvidia RTX A5000 GPU训练的（*Learning speed*使用了8个，*Fine-tuning*使用了6个），他们的 Elo 等级是使用其中6个GPU估算的。

5.2 评估

由于每次训练和评估持续时间约为一周，我们目前还没有机会对每种模型配置进行多次训练。因此，我们在结果中呈现的每种模型只训练了一次，我们包括的置信区间是基于我们为它们估计的Elo评分，如以下所述。

在训练过程中，每隔一定次数（每2次、5次或10次迭代），模型参数会被保存，并使用这一组参数来评估Elo等级。在本节中，我们将一对（模型，参数）称为一个玩家，并为每个玩家计算一个等级。

我们最初选择了10名选手，并模拟了每对选手之间进行60场比赛，以获取初始比赛数据M。对于每一对进行了比赛的选手，我们存储了胜场数w_{ij}、平局数d_{ij}和负场数l_{ij}: $M_{ij} = (w_{ij}, d_{ij}, l_{ij})$。利用这些数据，我们可以使用以下方程组进行线性回归，粗略估计M中在场选手的等级$r \in \mathbb{R}^{N_{players}}$:

$$\begin{cases} r_j - r_i &= \dfrac{400}{\log(10)} \log\left(\dfrac{w_{ij} + d_{ij} + l_{ij} + 1}{w_{ij} + \frac{d_{ij}+1}{2}} - 1\right) \quad \text{for } i \in M, j \in M_i, \\ \sum_{i \in M} r_i &= |M| \times 1000. \end{cases} \tag{11}$$

我们人为地添加一局平局，以避免只有一方只有胜利而没有失败的极端情况，在这种情况下，评级差异理论上会无限大。这相当于一个Jeffreys先验。最后一个方程将平均评级固定为1000，因为Elo评级整体上通过平移是不变的。

我们随后运行了算法 2 生成一个密集连接的比赛数据图，其中每个玩家至少与 5 名实力相近的其他玩家进行了比赛。最后，我们使用该数据集拟合了一个加权最小二乘回归模型来获得我们在实验结果图中使用的 Elo 排名。置信区间通过假设归一化后的比赛结果遵循正态分布来估计。如果 $\hat{p}_{ij} = \frac{w_{ij} + \frac{d_{ij}+1}{2}}{w_{ij}+d_{ij}+l_{ij}+1}$ 是玩家 i 击败玩家 j 的估计概率，我们近似认为 p_{ij} 遵循的分布为正态分布，并使用 delta 方法推导出 $r_j - r_j$ 渐近地遵循均值为$\frac{400}{\log(10)} \log\left(\frac{1}{p_{ij}} - 1\right)$、方差为 $\left(\frac{400}{\log(10)}\right)^2 \frac{1}{(w_{ij}+d_{ij}+l_{ij})p_{ij}(1-p_{ij})}$ 的正态分布。证明详见附录 A.2。使用 statsmodels [16] 的加权最小二乘线性模型，我们为每个玩家获得了 Elo 排名及其标准差，我们使用这些标准差在后续中推导出 2 倍标准差的置信区间。

5.3 实验

学习速度。我们的第一个实验比较了AlphaGateau基线能力从零开始学习如何玩8×8国际象棋，并将其与一个缩小版的AlphaZero模型进行了比较。AlphaZero模型包含5个残差层（包含10个CNN层）和总共220万参数，而AlphaGateau模型也有5个ResGATEAU层（包含10个GATEAU层）和总共100万参数，因为它在策略头中不需要一个完全连接层，而GATEAU层使用的是3×3 CNN参数的2/3。

Algorithm 2: Matching Players

```
Parameters: $N_{games} = 60, N_{sim} = 128$
Input: $M$
for player in unmatched players do
    for $i \leftarrow 1$ to $5$ do
        $r \leftarrow$ Elo_LR($M$) ;                    /* The Linear Regression is run on Eq. 11 */
        opponent $\leftarrow \operatorname{argmin}_{j \in M} |r_j - r_{\text{player}}|$ ;      /* If player $\notin M$, we set $r_{\text{player}} = 1000$ */
        $(w, d, l) \leftarrow$ play(player, opponent, $N_{games}, N_{sim}$);
        $M_{\text{player,opponent}} \leftarrow M_{\text{player,opponent}} + (w, d, l)$;
        $M_{\text{opponent,player}} \leftarrow M_{\text{opponent,player}} + (l, d, w)$;
    end
end
```

For this experiment, we generated 256 games of length 512 at each iteration, totalling 131 072 frames, and kept a frame window of 1M frames (all of the newly generated frames, and uniform sampling over the frame window of the previous iteration), over 500 iterations. During the neural network training, we used a batch size of 256. The training for AlphaGateau lasted 13 days and 16 hours, while AlphaZero took 10 days and 3 hours.

We report the estimated Elo ratings with a 2-sigma confidence interval in Figure 5. AlphaZero was only able to reach an Elo of 667 ± 38 in 500 iterations, and would likely continue to improve with more time, while AlphaGateau reached an Elo of 2105 ± 42, with an explosive first 50 iterations, and achieving results comparable to the final Elo of AlphaZero after only 10 iterations.

Although those results are promising, it is important to note that we only compared to a simplified version of AlphaZero, using only 5 layers instead of the original 40, and without spending large efforts to optimize the hyper-parameters of the model. As such, it is possible that the performance of AlphaZero could be greatly improved in this context with more parameter engineering. Both AlphaGateau and AlphaZero have not reached a performance plateau after 500 iterations, showing slow but consistent growth.

Fine-tuning. In our second experiment, we trained a AlphaGateau model with 10 residual layers on 5×5 chess for 100 iterations, then fine-tuned this model on 8×8 chess for 100 iterations. This model has a total of 1.7M parameters.

For this experiment, we generated 1024 games of length 256 at each iteration on 5×5 chess, and 256 games of length 512 while fine-tuning on 8×8 chess, and kept a frame window of 1M frames. The initial training lasted 2 days and 7 hours, and the fine-tuning 5 days and 15 hours.

We report the estimated Elo ratings with a 2-sigma confidence interval in Figure 6. The initial training ended with an Elo rating of 807 ± 46 when evaluated on 8×8 chess games, which suggests that it was able to learn general rules of chess on 5×5, and apply them with some success on 8×8 chess without having seen any 8×8 chess position during its training. Once the fine-tuning starts, the model jumps to an Elo of 1181 ± 50 after a couple iterations, suggesting the baseline learned on 5×5 was of high quality. After fine-tuning, the model had an Elo of 1876 ± 47, reaching comparable performances to the smaller model using roughly the same amount of iterations and GPU-time, despite being twice as big. Preliminary testing suggests that this model would become stronger if more data was generated at each iteration, but that would linearly increase the training time, as generating self-play games took half of the 5 days of training.

5.4 Impact of the Frame Window and the Number of Self-play Games

In order to train deeper networks, we experimented with the number of self-play games generated at each generation, and with the size of the frame window. if seems from our results in Figure 7 that having more newly generated data helps the model learn faster. However, the time taken for each iteration scales linearly with the number of generated games, and the model is still able to improve using older data. As such, keeping a portion of the frame window from previous iterations makes for a good compromise. There are however options to improve our frame window selection:

Algorithm 2: Matching Players

```
Parameters: $N_{games} = 60, N_{sim} = 128$
Input: $M$
for player in unmatched players do
    for $i \leftarrow 1$ to $5$ do
        $r \leftarrow$ Elo_LR($M$) ;                           /* The Linear Regression is run on Eq. 11 */
        opponent $\leftarrow \mathrm{argmin}_{j \in M} |r_j - r_{\text{player}}|$ ;        /* If player $\notin M$, we set $r_{\text{player}} = 1000$ */
        $(w, d, l) \leftarrow$ play(player, opponent, $N_{games}, N_{sim}$);
        $M_{\text{player,opponent}} \leftarrow M_{\text{player,opponent}} + (w, d, l)$;
        $M_{\text{opponent,player}} \leftarrow M_{\text{opponent,player}} + (l, d, w)$;
    end
end
```

对于这个实验，我们在每次迭代中生成了256个长度为512的游戏，总共生成了131 072帧，并保持了一个1M帧的帧窗口（所有新生成的帧以及对前一次迭代的帧窗口进行均匀采样），共进行了500次迭代。在神经网络训练过程中，我们使用了批量大小为256。AlphaGateau的训练持续了13天16小时，而AlphaZero则用了10天3小时。

我们在图5中报告了估算的Elo评分及其2sigma置信区间。AlphaZero在500次迭代中仅能达到667 ±38的Elo评分，并且在获得更多时间的情况下很可能会继续改进，而AlphaGateau达到了2105 ±42的Elo评分，在最初的50次迭代中表现出爆炸性的增长，并且在仅10次迭代后就达到了与AlphaZero最终Elo评分相媲美的结果。

尽管这些结果很有前景，但值得注意的是，我们仅将结果与简化版的AlphaZero进行了比较，后者只使用了5层结构，而不是原始的40层，并且没有花费大量努力来优化模型的超参数。因此，在这种情况下，通过更多的参数工程，AlphaZero的性能可能有很大的提升空间。AlphaGateau和AlphaZero在500次迭代后都没有达到性能 plateau，显示出缓慢但一致的增长。

Fine-tuning. 在我们的第二次实验中，我们训练了一个带有10个残差层的AlphaGateau模型，在5×5国际象棋上训练了100个迭代，然后在这个模型上使用8×8国际象棋进行了100个迭代的微调。该模型总共有1.7M个参数。

对于这个实验，在每次迭代中，我们在5×5国际象棋上生成了1024个长度为256的游戏，并在微调8×8国际象棋时生成了256个长度为512的游戏，保持了100万帧的窗口。初始训练持续了2天7小时，微调持续了5天15小时。

我们在图6中报告了带有2sigma置信区间的估计Elo评分。初始训练在评估了8×8国际象棋游戏后结束， Elo评分为807 ± 46，这表明它能够在5×5国际象棋中学习到一般的棋规，并在没有见过任何8×8国际象棋局面的情况下，成功地应用到8×8国际象棋中。一旦开始微调，模型在几次迭代后跳到了1181 ± 50的Elo评分，这表明在5×5国际象棋上学习到的基线质量很高。经过微调后，模型的Elo评分为1876 ± 47，尽管其大小是较小模型的两倍，但在大致相同的迭代次数和GPU时间下达到了相当的性能。初步测试表明，如果每次迭代生成更多的数据，该模型会变得更强大，但这也意味着训练时间将线性增加，因为生成自我对弈游戏占用了五天训练时间的一半。

5.4 框架窗口和自我对弈游戏数量的影响

为了训练更深的网络，我们实验了每一代生成的自我对弈游戏的数量，以及帧窗口的大小。从图7的结果来看，有更多的新生成数据似乎有助于模型更快地学习。然而，每个迭代所需的时间与生成的游戏数量成线性关系，模型仍然能够使用较旧的数据进行改进。因此，保留一部分之前迭代的帧窗口是一个不错的折中方案。然而，我们还有改进帧窗口选择的方法:

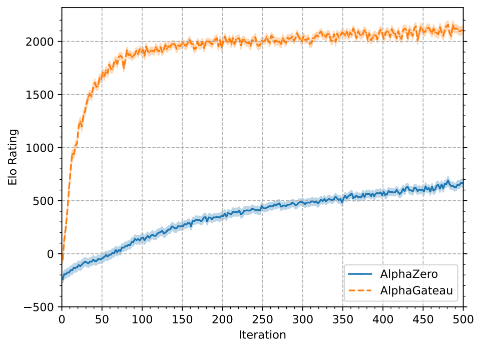

Figure 5: The Elo ratings of AlphaZero and AlphaGateau with 5 residual layers trained over 500 iterations. The AlphaGateau model initially learns ~10 times faster than the AlphaZero model, and settles after 100 iterations to a comparable speed of growth to that of AlphaZero.

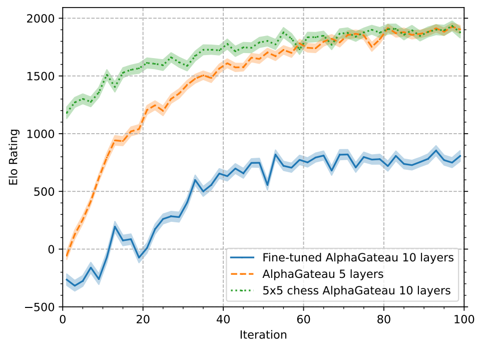

Figure 6: The Elo ratings of the first 100 iterations of the AlphaGateau model from Figure 5 was included for comparison. The initial training on 5×5 chess is able to increase its rating while evaluated on 8×8 chess during training, even without seeing any 8×8 chess position. The fine-tuned model starts with a good baseline, and reaches comparable performances to the 5-layer model despite being undertrained for its size.

- Which previous samples should be selected? We selected uniformly at random from the previous frame window to complement the newly generated samples, but it might be preferable to select fewer samples, but chosen as to represent a wide range of different positions.
- Past samples are by design of dubious quality. As they come from self-play games with previous model parameters, they correspond to games played with a lower playing strength, and the policy output by the MCTS is also worse. Keeping a sample that is too old might cause a drop of performance rather than help the model learn. We experimented a little with keeping the 1M most recent samples, but with little success.

We also initially tried to increase the number of epochs in one iteration, but only saw marginal gains, suggesting that mixing new data among the previous frame window helps the model extract more training information from previous samples.

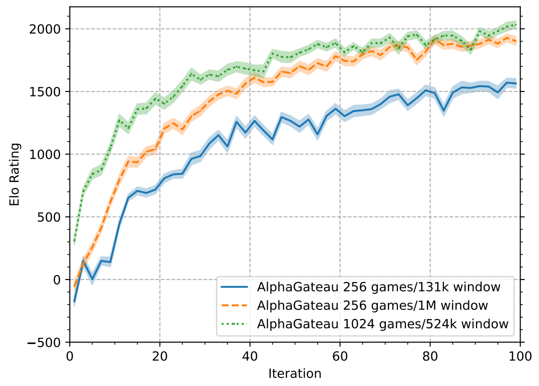

Figure 7: The two models with a frame window of size 131 072 only kept the latest generated games in the frame window. The model keeping no frame window and generating 256 games was trained in only 39 hours, but had the worst performance. Adding a 1M frame window improved the performance a little and lasted 60 hours, while increasing the number of self-play games to 1024 performed the best, but took 198 hours.

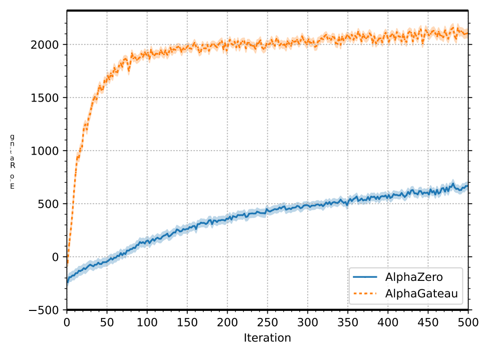

图 5：AlphaZero 和 Alpha-Gateau（带有 5 个残差层）在 500 次迭代中训练的 Elo 评级。AlphaGateau 模型最初比 AlphaZero 模型快 10 倍的学习，经过 100 次迭代后，其增长速度与 AlphaZero 相当。

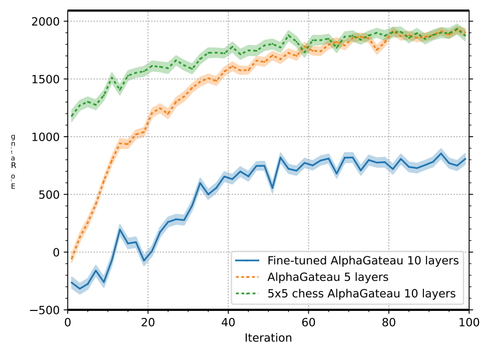

图 6：将 AlphaGateau 模型（如图 5 所示）前 100 次迭代的 Elo 评级包括在内，用于比较。初始在 5×5 围棋上的训练能够提高其评级，即使在训练过程中评估的是 8×8 围棋，也没有见过任何 8×8 围棋的局面。微调后的模型从一个良好的基线开始，并达到了与 5 层模型相当的性能。

尽管规模较大，但训练不足。

- 哪些先前的样本应该被选择？我们从之前的帧窗口中随机选择样本来补充新生成的样本，但选择更少的样本可能更好，这些样本能够代表不同的位置范围。

- 过去的数据设计上质量可疑。因为它们来自使用之前模型参数的自我对弈游戏，所以这些数据对应的是使用较低棋力水平进行的游戏，而MCTS输出的策略也较差。保留太旧的数据样本可能会导致性能下降而不是帮助模型学习。我们尝试过保留最近的100万样本，但效果不佳。

我们最初也尝试在一次迭代中增加迭代次数，但只看到了边际收益，这表明在先前帧窗口中混合新数据有助于模型从先前样本中提取更多的训练信息。

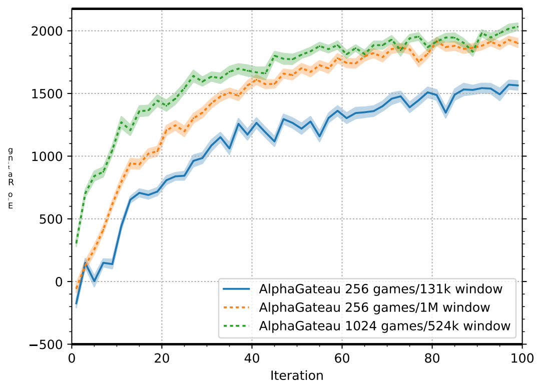

Figure 7: 两个模型使用大小为131 072的帧窗口，只保留了帧窗口中最新生成的游戏。不使用帧窗口并生成256个游戏的模型仅用了39小时进行训练，但性能最差。添加1M大小的帧窗口稍微改善了性能，耗时60小时，而将自对弈游戏的数量增加到1024个性能最佳，但需要198小时。

6 Conclusion

In this paper, we introduce AlphaGateau, a variant on the AlphaZero model design that represents a chess game state as a graph, in order to improve performance and enable it to better generalize to variants of a game. This change from grid-based CNN to graph-based GNN yields impressive increase in performance, and seems promising to enable more research on reinforcement-learning based game-playing agent research, as it reduces the resources required to train one.

We also introduce a variant of GAT, GATEAU, that we designed in order to handle edge features in a simple manner performed well, and efficiently.

Future Work. As our models were relatively shallow when compared to the initial AlphaZero, it would be important to confirm that AlphaGateau still outperforms AlphaZero when both are trained with a full 40-deep architecture. This will require a lot more computing time and resources.

As discussed in Section 5.4, our design of the frame window is a little unsatisfactory, and a future improvement would be to define an efficiently computable similarity metric between chess positions, that helps the neural network generalize.

We focused on chess for this paper, but there are other games that could benefit from this new approach. The first one would be shogi, as it has similar rules to chess, and the promising generalization results from AlphaGateau could be used to either train a model on one game, and fine-tune it on the other, or to jointly train it on both games, to have a more generalized game-playing agent. As alluded to in the Graph Design 4.2, more features engineering would be required to have node and edge features compatibility between chess graphs, and shogi graphs. It could also be possible to change the model architecture to handle games with more challenging properties, such as the game Risk, which has more than 2 players, randomness, hidden information, and varying maps, but is even more suited to being represented as a graph.

Acknowledgments and Disclosure of Funding

This work was supported by JST BOOST, Grant Number JPMJBS2407 and JST CREST, Grant Number JPMJCR21D1. We thank Armand Barbot and Jill-Jênn Vie for their helpful comments and insightful discussions.

References

[1] G. Arakelyan, G. Soghomonyan, and The Aim team. Aim, June 2020. URL `https://github.com/aimhubio/aim`.

[2] S. Ben-Assayag and R. El-Yaniv. Train on Small, Play the Large: Scaling Up Board Games with AlphaZero and GNN, July 2021. URL `http://arxiv.org/abs/2107.08387`. arXiv:2107.08387 [cs].

[3] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[4] J. Chen and H. Chen. Edge-Featured Graph Attention Network, Jan. 2021. URL `http://arxiv.org/abs/2101.07671`. arXiv:2101.07671 [cs].

[5] R. Coulom. Whole-History Rating: A Bayesian Rating System for Players of Time-Varying Strength. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, H. J. Van Den Herik, X. Xu, Z. Ma, and M. H. M. Winands, editors, *Computers and Games*, volume 5131, pages 113–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-87607-6 978-3-540-87608-3. doi: 10.1007/978-3-540-87608-3_11. URL `https://link.springer.com/10.1007/978-3-540-87608-3_11`. Series Title: Lecture Notes in Computer Science.

[6] J. Czech, P. Korus, and K. Kersting. Monte-Carlo Graph Search for AlphaZero, Dec. 2020. URL `https://arxiv.org/abs/2012.11045v1`.

6 结论

在本文中，我们介绍了AlphaGateau，这是一种AlphaZero模型设计的变体，将国际象棋游戏状态表示为图，以提高性能并使其能够更好地泛化到游戏的变体中。从基于网格的CNN到基于图的GNN的这一变化带来了性能的显著提升，并似乎有望促进基于强化学习的游戏玩牌代理研究，因为它减少了训练所需资源。

我们还介绍了一种GAT的变体GATEAU，我们设计它以简单的方式处理边特征，并且表现良好且效率高。

Future Work. 由于我们的模型在与初始AlphaZero相比时相对较浅，因此需要确认当两者都使用完整的40层架构进行训练时，AlphaGateau仍然优于AlphaZero。这将需要更多的计算时间和资源。

如第5.4节所述，我们设计的帧窗口略显不足，未来的一个改进是定义一个高效可计算的棋局相似度度量，这有助于神经网络泛化。

我们在这篇论文中专注于国际象棋，但其他游戏也可以从这种新方法中受益。第一个就是将棋，因为它有类似国际象棋的规则，AlphaGateau 的有希望的一般化结果可以用于在一种游戏中训练模型，然后在另一种游戏中微调它，或者同时在这两种游戏中训练它，以拥有一个更通用的游戏代理。如图设计 4.2 所暗示的，为了使国际象棋图和将棋图的节点和边特征兼容，需要更多的特征工程。也有可能改变模型架构以处理具有更具挑战性特性的游戏，例如有超过两名玩家、随机性、隐藏信息和变化地图的Risk游戏，但将棋更适合作为图来表示。

Acknowledgments 和 资金披露

这项工作得到了JST BOOST的支持，资助编号JPMJBS2407，以及JST CREST的支持，资助编号JPMJCR21D1。我们感谢Armand Barbot和Jill-Jênn Vie提供的宝贵意见和深入讨论。

参考文献

[1] G. Arakelyan, G. Soghomonyan, 和 The Aim 团队. Aim, 2020年6月. URL <https://github.com/aimhubio/aim>.

[2] S. Ben-Assayag 和 R. El-Yaniv. 小规模训练，大规模游戏：使用 AlphaZero 和 GNN 扩大规模棋盘游戏，2021年7月。URL http://arxiv.org/abs/2107.08387。arXiv:2107.08387 [cs]。

[3] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, 和 Q. Zhang. JAX: 2018 年可组合变换的 Python+NumPy 程序, URL http://github.com/google/jax.

[4] 陈健和陈华. 边特征图注意网络，2021年1月. URL http://arxiv.org/abs/2101.07671. arXiv:2101.07671 [cs].

[5] R. Coulom. 整个历史评级：时间变化强度玩家的贝叶斯评级系统. 在 D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, H. J. Van Den Herik, X. Xu, Z. Ma, 和 M. H. M. Winands 编辑，*Comput- ers and Games*，第 5131 卷，第 113–124 页。Springer Berlin Heidelberg，柏林，德国，2008 年。ISBN 978-3-540-87607-6 978-3-540-87608-3。doi: 10.1007/978-3-540-87608-3_11。URL https://link.springer.com/10.1007/978-3-540-87608-3_11。系列标题：计算机科学讲义。

[6] J. 捷克, P. 科鲁斯, 和 K. 基尔斯坦. AlphaZero 的 Monte-Carlo 图搜索, 2020年12月. URL https://arxiv.org/abs/2012.11045v1.

[7] I. Danihelka, A. Guez, J. Schrittwieser, and D. Silver. Policy improvement by planning with Gumbel. Oct. 2021. URL `https://openreview.net/forum?id=bERaNdoegnO`.

[8] DeepMind, I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, L. Sartran, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, M. Stanojević, W. Stokowiec, L. Wang, G. Zhou, and F. Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/deepmind`.

[9] J. Godwin*, T. Keck*, P. Battaglia, V. Bapst, T. Kipf, Y. Li, K. Stachenfeld, P. Veličković, and A. Sanchez-Gonzalez. Jraph: A library for graph neural networks in jax., 2020. URL `http://github.com/deepmind/jraph`.

[10] L. Gong and Q. Cheng. Exploiting Edge Features in Graph Neural Networks, Jan. 2019. URL `http://arxiv.org/abs/1809.02709`. arXiv:1809.02709 [cs, stat].

[11] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. v. Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL `http://github.com/google/flax`.

[12] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, Jan. 2017. URL `http://arxiv.org/abs/1412.6980`. arXiv:1412.6980 [cs].

[13] S. Koyamada, S. Okano, S. Nishimori, Y. Murata, K. Habara, H. Kita, and S. Ishii. Pgx: Hardware-Accelerated Parallel Game Simulators for Reinforcement Learning, Jan. 2024. URL `http://arxiv.org/abs/2303.17503`. arXiv:2303.17503 [cs].

[14] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated Graph Sequence Neural Networks, Sept. 2017. URL `http://arxiv.org/abs/1511.05493`. arXiv:1511.05493 [cs, stat].

[15] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 588(7839):604–609, Dec. 2020. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-03051-4. URL `http://arxiv.org/abs/1911.08265`. arXiv:1911.08265 [cs, stat].

[16] S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[17] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, Jan. 2016. doi: 10.1038/nature16961.

[18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, Dec. 2017. URL `https://arxiv.org/abs/1712.01815v1`.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks, Oct. 2017. URL `https://arxiv.org/abs/1710.10903v3`.

[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How Powerful are Graph Neural Networks?, Oct. 2018. URL `https://arxiv.org/abs/1810.00826v3`.

[7] I. Danihelka, A. Guez, J. Schrittwieser, 和 D. Silver. 使用Gumbel进行规划以改进策略。2021年10月. URL https://openreview.net/forum?id=bERaNdoegnO.

[8] DeepMind, I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Bud-den, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, L. Sartran, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, M. Stanojević, W. Stokowiec, L. Wang, G. Zhou, 和 F. Viola. DeepMind JAX 生态系统, 2020. URL http://github.com/deepmind.

[9] J. Godwin*， T. Keck*， P. Battaglia， V. Bapst， T. Kipf， Y. Li， K. Stachenfeld， P. Veličković， 和 A. Sanchez-Gonzalez. Jraph：一个在 Jax 中的图神经网络库， 2020. URL http://github.com/deepmind/jraph。

[10] Gong L. 和 Cheng Q. 利用图神经网络中的边特征, 2019年1月. URL http://arxiv.org/abs/1809.02709. arXiv:1809.02709 [cs, stat].

[11] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, 和 M. v. Zee. Flax: 一个针对 JAX 的神经网络库及其生态系统， 2023. URL http://github.com/google/flax.

[12] D. P. Kingma 和 J. Ba. Adam：一种随机优化方法， 2017年1月. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

[13] S. 科山达, S. 岡ano, S. 西森ori, Y. 森田, K. 北原, H. 北田, 和 S. 是枝. Pgx: 为强化学习加速的并行游戏模拟器硬件, 2024年1月. URL http://arxiv.org/abs/2303.17503. arXiv:2303.17503 [cs].

[14] 李洋, 大卫·塔罗, 马库斯·布罗克肖姆, 和 里克·泽梅尔. 门控图序列神经网络, 2017年9月. URL http://arxiv.org/abs/1511.05493. arXiv:1511.05493 [cs, stat].

[15] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. 通过规划使用学习模型掌握雅达利、围棋、国际象棋和将棋。*Nature*, 588(7839):604–609, 2020年12月。ISSN 0028-0836, 1476-4687。doi: 10.1038/s41586-020-03051-4。URL http://arxiv.org/abs/1911.08265。arXiv:1911.08265 [cs, stat]。

[16] S. Seabold 和 J. Perktold. statsmodels: 使用 Python 进行计量经济和统计建模. 在 *9th Python in Science Conference*, 2010.

[17] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. 使用深度神经网络和树搜索掌握围棋游戏。*Nature*, 529:484–489, 2016年1月。doi: 10.1038/nature16961。

[18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, 和 D. Hassabis. 通过自我对弈掌握国际象棋和将棋：一种通用强化学习算法， 2017年12月。URL https://arxiv.org/abs/1712.01815v1。

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, 和 Y. Bengio. 图注意力网络, 2017年10月. URL https://arxiv.org/abs/1710.10903v3.

[20] K. Xu, W. Hu, J. Leskovec, 和 S. Jegelka. 图神经网络有多强大？, 2018年10月. URL https://arxiv.org/abs/1810.00826v3.

A Appendix / supplemental material

A.1 Elo

The Elo rating system was initially introduced as a way to assign a value to the playing strength of chess players. The rating of each player is supposed to be dynamic and be adjusted after each game they play to follow their evolution.

The Elo ratings are defined to respect the following property: If two players with Elo rating R_A and R_B played a game, the probability that player A wins is

$$E_A = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}}. \tag{12}$$

A.2 Variance of estimated Elo rating difference

We assume that the outcome of a game between two players of Elo R_A and R_B is a bernouilli trial, with a probability that player A win being given by the central Elo equation 12. If we want to estimate the Elo of both players, we need to estimate that probability p. To do so, we can make the two players play n games, and sum the wins of player A, as well as half his draws, to get $x = w + \frac{d}{2}$. From this, we can estimate the value of p using the estimator $\hat{p} = \frac{x}{n}$. In the case that one of the two players is significantly stronger than the other, $\hat{p}$ could be close to 0 or 1, in which case this estimator is wildly inacurrate. To remedy this, we will instead rely on a Jeffreys prior, to get the estimator $\hat{p}_{Jeffreys} = \frac{x+1/2}{n+1}$. We will note this estimator $\hat{p}$ in the following.

From our assumptions, we have that x follows a binomial distribution $B(n, p)$, and we will approximate the distribution that $\hat{p}$ follows by a normal distribution $\hat{p} \sim \mathcal{N}(p, \frac{p(1-p)}{n})$.

By inverting the Elo equation 12, we can get the rating difference from the probability that A wins as

$$R_B - R_A = \frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right), . \tag{13}$$

Therefore, posing $g(y) = \frac{400}{\log(10)} \log\left(\frac{1}{y} - 1\right)$, which is differentiable, we can use the delta method to get that $g(\hat{p})$ is asymptotically Gaussian. The derivative of g is

$$\begin{aligned} g'(y) &= \frac{400}{\log(10)} \left(-\frac{1}{y^2}\right) \left(\frac{1}{\frac{1}{y} - 1}\right) \\ &= -\frac{400}{\log(10)} \frac{1}{y^2} \frac{y}{y-1} \\ &= -\frac{400}{\log(10)} \frac{1}{y(y-1)}, \end{aligned} \tag{14}$$

which gives

$$\begin{aligned} \hat{R}_B - \hat{R}_A = g(\hat{p}) &\sim \mathcal{N}\left(g(p), \frac{p(1-p)}{n} (g'(p))^2\right) \\ &\sim \mathcal{N}\left(\frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right), \frac{p(1-p)}{n} \left(-\frac{400}{\log(10)} \frac{1}{p(p-1)}\right)^2\right) \\ &\sim \mathcal{N}\left(\frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right), \frac{400^2}{\log(10)^2} \frac{p(1-p)}{n} \frac{1}{(p(p-1))^2}\right) \\ &\sim \mathcal{N}\left(\frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right), \frac{400^2}{\log(10)^2} \frac{1}{np(p-1)}\right). \end{aligned} \tag{15}$$

A.3 Comparison with BayesElo

We also used BayesElo[5] to evaluate the Elo ratings of our models using all the PGN files that were generated recording the games played between all our models. In Figure 8, we plotted a point

附录 / 补充材料

A.1 棋手 Elo

Elo排名系统最初被引入作为一种评估国际象棋玩家棋力的方法。每位玩家的排名应该是动态的，并在他们每场比赛后进行调整以反映他们的进步。

The Elo排名定义为满足以下性质：如果两名玩家的Elo排名分别为R_A和R_B进行了比赛，那么玩家A获胜的概率为

$$E_A = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}}. \tag{12}$$

A.2 估计Elo等级差的方差

我们假设两名Elo等级为R_A和R_B的玩家之间的比赛结果是一个伯努利试验，玩家A获胜的概率由中心Elo方程12给出。如果我们想估计两名玩家的Elo等级，我们需要估计这个概率p。为此，我们可以让两名玩家进行n场比赛，并将玩家A的胜利次数与其一半的平局次数相加，得到$x = w + \frac{d}{2}$。从这个结果，我们可以使用估计器$\hat{p} = \frac{x}{n}$来估计p的值。如果其中一名玩家明显比另一名玩家强得多，$\hat{p}$可能接近0或1，在这种情况下，这个估计器会非常不准确。为了解决这个问题，我们将依赖Jeffreys先验来得到估计器$\hat{p}_{Jeffreys} = \frac{x+1/2}{n+1}$。我们将在以下部分将这个估计器记作$\hat{p}$。

从我们的假设出发，我们有 x 遵循二项分布 $B(n,p)$，并且我们将用正态分布 $\hat{p} \sim \mathcal{N}(p, \frac{p(1-p)}{n})$ 近似 $\hat{p}$ 的分布。

通过反转Elo方程12，我们可以从A获胜的概率得到评级差为

$$R_B - R_A = \frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right),. \tag{13}$$

因此，提出 $g(y) = \frac{400}{\log(10)} \log\left(\frac{1}{y} - 1\right)$，这是一个可微函数，我们可以使用delta方法得到 $g(\hat{p})$ 是渐近正态的。g 的导数为

$$\begin{aligned} g'(y) &= \frac{400}{\log(10)} \left(-\frac{1}{y^2}\right)\left(\frac{1}{\frac{1}{y} - 1}\right) \\ &= -\frac{400}{\log(10)} \frac{1}{y^2} \frac{y}{y-1} \\ &= -\frac{400}{\log(10)} \frac{1}{y(y-1)}, \end{aligned} \tag{14}$$

which gives

$$\begin{aligned} \hat{R}_B - \hat{R}_A = g(\hat{p}) &\sim \mathcal{N}\left(g(p), \frac{p(1-p)}{n} \left(g'(p)\right)^2\right) \\ &\sim \mathcal{N}\left(\frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right), \frac{p(1-p)}{n}\left(-\frac{400}{\log(10)} \frac{1}{p(p-1)}\right)^2\right) \\ &\sim \mathcal{N}\left(\frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right), \frac{400^2}{\log(10)^2} \frac{p(1-p)}{n} \frac{1}{\left(p(p-1)\right)^2}\right) \\ &\sim \mathcal{N}\left(\frac{400}{\log(10)} \log\left(\frac{1}{p} - 1\right), \frac{400^2}{\log(10)^2} \frac{1}{np(p-1)}\right). \end{aligned} \tag{15}$$

A.3 与 BayesElo 的比较

我们还使用了 BayesElo[5] 来使用记录的所有模型之间对弈生成的 PGN 文件评估我们模型的 Elo 评分。在图 8 中，我们绘制了一个点

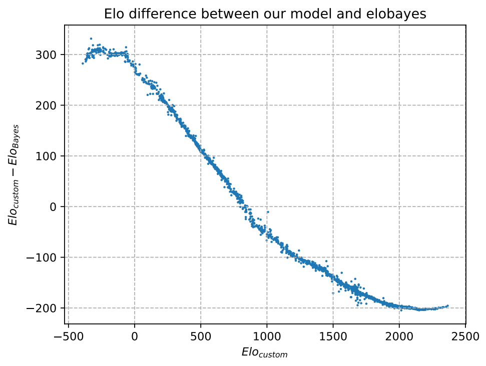

Figure 8: The difference between BayesElo ratings and the Elo ratings according to our method. We removed to each Elo the average Elo of all players in its respective method, such that the average effective Elo for both BayesElo and our method is 0

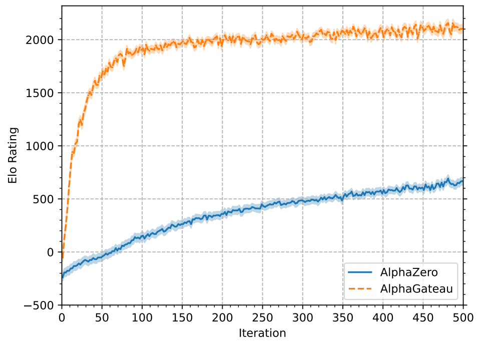

Figure 9: A copy of Figure 5 for comparison.

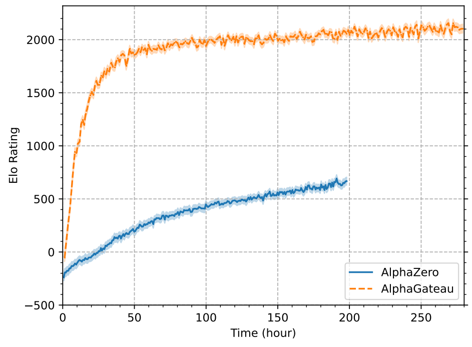

Figure 10: Using running time instead of iteration for Figure 5 doesn't change much, as AlphaZero is only a little bit faster than AlphaGateau.

on (x, y) for each player where x is its Elo rating following our method and y is the difference in predicted Elo between our method and BayesElo.

In practice, weaker models tend to be overrated by our method compared with BayesElo, while stronger models are underrated. As this relation is smooth and monotone, this suggest that both methods order the relatives strength of all players similarly, with small differences only being due to noise. The main difference being that our range of Elo ratings is compacted towards the extremes, which we assume is due to our practical Jeffreys prior, that implies that strong models drew at least one game against all their opponents, handicapping their Elo, and similarly weak models always manage to not lose at least one game.

A.4 FLOPs comparison

We didn't record the FLOPs of our models, however, we did record the time taken for each iteration. In Figure 10, 12, and 14, we plotted the three main plots of the main paper using that data on the X-axis. As some experiments were run using 6 GPUs instead of *, we multiplied their recorded time by a factor $\frac{6}{8}$ for better comparison.

A.5 PGNs

We include with Figure 15, 16, and 17 a few PGNs showing games played by our models.

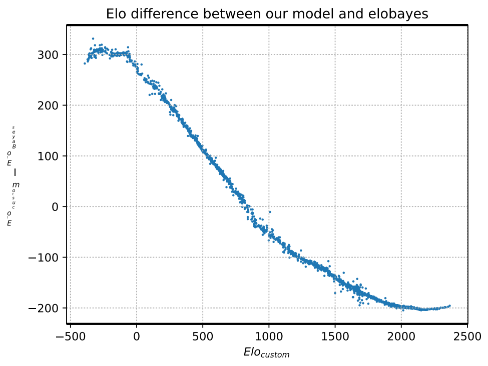

图 8：根据我们方法计算的 BayesElo 评级与 Elo 评级之间的差异。我们从每个 Elo 中减去了其相应方法中所有玩家的平均 Elo，使得 BayesElo 和我们方法的平均有效 Elo 值均为 0

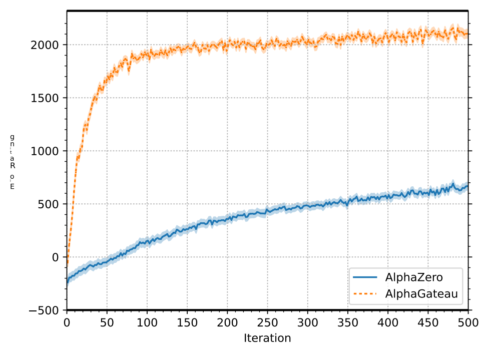

Figure 9: 与图5比较的副本。

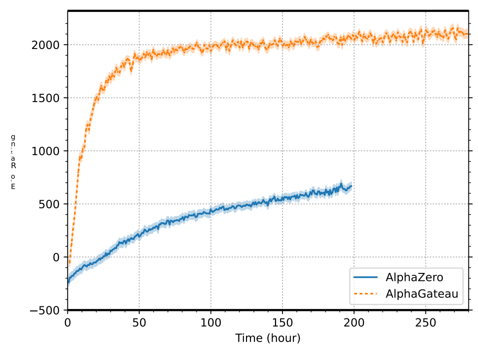

图 10: 将图 5 中的迭代次数改为使用运行时间几乎没有变化，因为 AlphaZero 只比 AlphaGateau 稍微快一点。

在每个玩家的 (x, y) 上，其中 x 是其按照我们方法计算的Elo评分，而 y 是我们方法与Bayes Elo预测Elo差异。

在实践中，我们的方法倾向于将较弱的模型高估，相对于BayesElo而言，而较强的模型则被低估。由于这种关系是平滑且单调的，这表明两种方法以相似的方式对所有玩家的相对实力进行排序，只有细微的差异可能是由于噪声造成的。主要的区别在于我们的Elo评分范围向极端压缩，我们假设这是由于我们实用的Jeffreys先验导致的，这种先验表明较强的模型至少与所有对手进行过一场比赛，从而限制了它们的Elo评分；类似地，较弱的模型总是能够至少避免输给某个对手。

A.4 FLOPs 比较

我们没有记录模型的FLOPs，但是我们记录了每个迭代所需的时间。在图10、12和14中，我们使用这些数据在X轴上绘制了主要论文的三个主要图表。由于有些实验使用了6块GPU而不是*，我们将它们记录的时间乘以一个因子$\frac{6}{8}$以进行更好的比较。

A.5 PGNs

我们随附了图15、16和17中显示的由我们的模型对弈的部分PGN棋局。

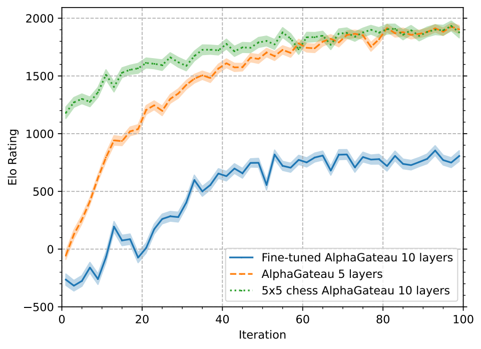

Figure 11: A copy of Figure 6 for comparison.

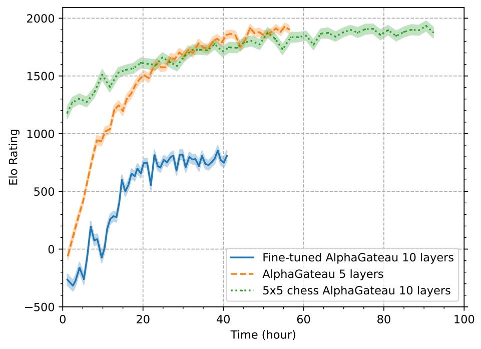

Figure 12: Using running time instead of iteration for Figure 5. Training the deeped model takes roughly 40 hours longer, for a similar amount of generated games and training steps.

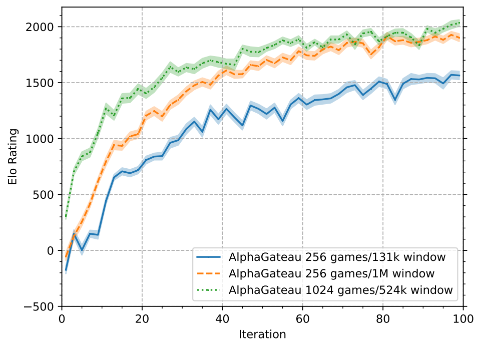

Figure 13: A copy of Figure 7 for comparison.

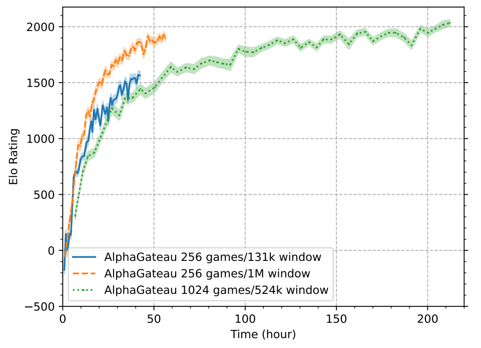

Figure 14: Using running time instead of iteration for Figure 5 shows that although using more newly generated games instead of relying on a frame window of previous data makes the model improve more per iteration, it doe result in a slower training in practice.

Figure 15 contains the PGN of a game played on 8×8 chess by the last iteration of the 5×5 AlphaGateau model of our fine-tuning experiment, to showcase its playing style while having never seen an 8×8 board during its training.

Figure 16 and 17 contains the PGNs of games played by the last iteration (iteration 499) of the full AlphaGateau model of our first experiment as white and black respectively, selected among the games played against other models to evaluate all the Elo rankings, selected using the script book.py in the GitHub repo, following the most played move each ply. The game played as white in Figure 16 is played against the iteration 61 of a fine-tuned AlphaGateau model using 8 layers, 1024 generated games per iteration, but only 32 MCTS simulations, with an estimated Elo rating of 2124. The game played as black in Figure 17 is played against the iteration 439 of the same model.

A.6 Lichess Evaluation

We ran the final iteration of the 5-layer AlphaGateau model on Lichess, as the bot AlphaGateau (`https://lichess.org/@/AlphaGateau`). We let it play against other bots and some human players in bullet, blitz, and rapid time formats by varying the number of MCTS simulations to adjust the time required to play each move. This was implemented using the default Lichess bot bridge (`https://github.com/lichess-bot-devs/lichess-bot`), and the relevant code is in the lichess folder of our GitHub repo.

At the end of October 2024, after between 100 and 200 games per time format, AlphaGateau was able to reach a bullet Elo of 1991, a blitz Elo of 1829, and a rapid Elo of 1884.

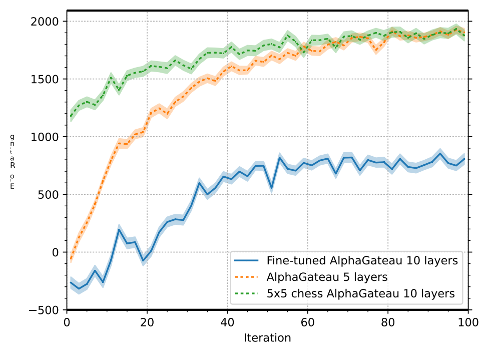

Figure 11: 与图6比较的副本。

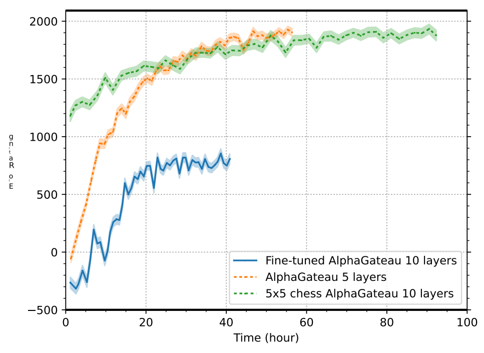

图 12: 使用运行时间而不是迭代次数来表示图 5。训练深度模型大约需要多出 40 小时，生成的游戏数量和训练步骤相似。

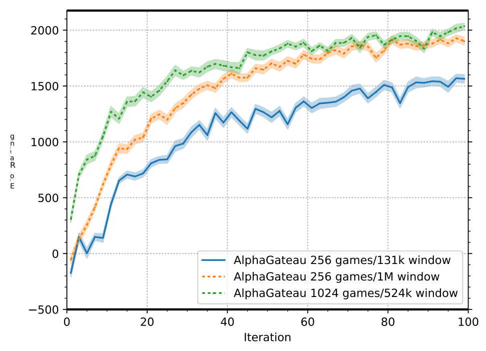

Figure 13: 与图7比较的副本。

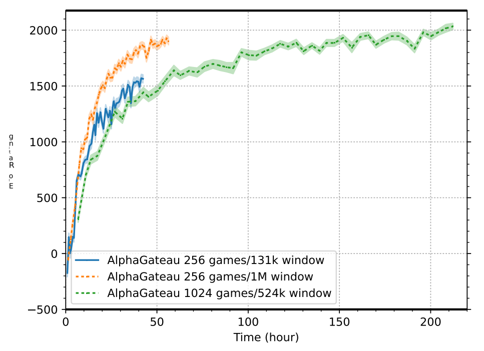

图 14: 将运行时间而不是迭代用于图 5 显示，尽管使用更多新生成的游戏而不是依赖于之前数据的帧窗口可以使模型每迭代改进更多，但实际上这会导致训练速度变慢。

图 15 包含了在 8 × 8 棋盘上由我们精细调优实验的最后一轮迭代的 5 × 5 AlphaGateau 模型下的一局游戏的 PGN，以展示其游戏风格，同时在其训练过程中从未见过 8 × 8 的棋盘。

图16和图17包含我们在第一次实验中完整AlphaGateau模型的最后一轮（第499轮）作为白方和黑方所下棋局的PGN，这些棋局是从与其他模型对弈的棋局中挑选出来的，用于评估所有Elo排名，使用GitHub仓库中的book.py脚本选择，并按照每步棋最常走的棋路。图16中作为白方的棋局是对抗一个使用8层、每轮生成1024场游戏但仅进行32次MCTS模拟的细调AlphaGateau模型的第61轮。图17中作为黑方的棋局是对抗同一模型的第439轮。

A.6 谁仕士评价

我们对5层AlphaGateau模型进行了最终迭代的运行，在Lichess上使用了名为AlphaGateau的机器人（https://lichess.org/@/AlphaGateau）。我们通过改变MCTS模拟次数来调整每步棋所需的时间，让机器人与其他机器人和一些人类玩家在标准、 blitz和快速对局格式中进行对弈。这通过使用默认的Lichess机器人桥接程序（https://github.com/lichess-bot-devs/lichess-bot）实现，相关的代码位于我们GitHub仓库的lichess文件夹中。

在2024年10月底，经过每次进行100到200场比赛后，AlphaGateau 达到了超快棋 Elo 1991、快棋 Elo 1829 和 慢棋 Elo 1884。

1. e4 c6 2. h3 h6 3. ♗e2 ♘a6 4. g4 g6 5. a4 ♖h7 6. f4 f6 7. ♘f3 ♗g7 8. h4 d5 9. e5 ♘b4 10. c3 ♗xg4 11. cxb4 ♗h5 12. ♘g1 ♗h8 13. ♘c3 d4 14. ♖h3 a5 15. exf6 ♘xf6 16. bxa5 e6 17. ♗xh5 gxh5 18. b4 ♘g8 19. ♖d3 ♘f6 20. ♘ge2 ♘g4 21. ♕c2 ♕xh4+ 22. ♘g3 ♕h2 23. ♘ge4 ♕h1+ 24. ♔e2 ♖g7 25. ♖xd4 ♕h2+ 26. ♔f3 ♕h1+ 27. ♔e2 ♖f7 28. ♖d3 ♕g2+ 29. ♔e1 ♖g7 30. ♖a2 ♕h1+ 31. ♔e2 ♔e7 32. ♖g3 ♖e8 33. ♕d3 ♖d8 34. ♕c4 ♕h4 35. ♖a1 ♖f7 36. ♖a2 ♖xf4 37. d4 ♖f7 38. ♗e3 ♖g7 39. ♖c2 ♔f7 40. ♕c5 ♔e8 41. ♖f3 ♕h2+ 42. ♔d3 ♕h1 43. ♖f4 ♘xe3 44. ♔xe3 ♖g1 45. ♖e2 ♖xd4 46. ♘f2 ♖xf4 47. ♘xh1 ♖h4 48. ♘f2 ♔d7 49. b5 ♖g3+ 50. ♔d2 ♗xc3+ 51. ♔c2 ♗h8 52. ♕b6 ♔d6 53. a6 bxa6 54. ♕d8+ ♔c5 55. ♕xh4 ♖g8 56. bxc6 ♔xc6 57. ♕f4 ♗g7 58. ♖xe6+ ♔d5 59. ♕f5+ ♔c4 60. ♕f3 ♔b4 61. ♕xh5 ♔xa4 62. ♖xa6+ ♔b4 63. ♕f5 ♖h8 64. ♖a7 ♗d4 65. ♖b7+ ♗b6 66. ♘d3+ ♔a4 67. ♖e7 h5 68. ♕f6 ♗d8 69. ♕f4+ ♔b5 70. ♖e5+ ♔b6 71. ♘f2 ♔c7 72. ♘e4 ♔c6 73. ♔c3 ♔b6 74. ♔d4 ♔c6 75. ♘g3 ♔b6 76. ♔c4 ♔b7 77. ♕f3+ ♔c7 78. ♔d5 h4 79. ♘e2 h3 80. ♕f4 ♔d7 81. ♘g3 ♔c7 82. ♖h5+ ♔d7 83. ♖xh8 h2 84. ♖xh2 ♗c7 85. ♕f2 ♔c8 86. ♕f5+ ♔b7 87. ♕d7 ♔a8 88. ♔e4 ♗b8 89. ♖h6 ♗a7 90. ♖f6 ♗g1 91. ♖f5 ♔b8 92. ♖f8#

Figure 15: Game played by a model fully trained only on 5 × 5 chess as white. White is able to use their white bishop to eliminate black's white bishop, but seems to undervalue their knight on move 14, probably because it is a worse piece in 5 × 5 chess due to being more constrained and harder to effectively employ

1. e4 c5 2. ♘b1c3 e6 3. ♘g1f3 ♘b8c6 4. d4 d4 5. ♘f3d4 ♘g8f6 6. ♘d4c6 c6 7. e5 ♘f6d5 8. ♘c3e4 ♕d8c7 9. f4 ♕c7b6 10. ♗f1e2 ♗c8a6 11. ♗e2a6 ♕b6a6 12. a3 h5 13. ♕d1e2 ♕a6e2 14. ♔e1e2 f5 15. ♘e4c3 a5 16. ♘c3d5 d5 17. ♗c1e3 ♗f8e7 18. c3 ♖h8g8 19. ♔e2f3 g5 20. g3 g4 21. ♔f3e2 h4 22. b4 ♔e8f7 23. ♖a1c1 h3 24. ♗e3b6 b4 25. b4 ♖a8a3 26. ♖c1c7 ♖g8b8 27. ♗b6c5 ♗e7c5 28. ♖c7c5 ♖b8b4 29. ♖c5c2 d4 30. ♖h1d1 ♔f7e7 31. ♔e2f2 ♖a3f3 32. ♔f2g1 d3 33. ♖c2c3 ♖b4b2 34. ♖c3d3 ♖f3d3 35. ♖d1d3 ♖b2c2 36. ♖d3d1 ♖c2g2 37. ♔g1h1 ♖g2a2 38. ♔h1g1 ♔e7e8 39. ♖d1b1 ♖a2g2 40. ♔g1h1 ♖g2c2 41. ♔h1g1 ♖c2c3 42. ♖b1b8 ♔e8e7 43. ♖b8b1 ♖c3c2 44. ♖b1b7 ♖c2c1 45. ♔g1f2 ♖c1h1 46. ♔f2e2 ♖h1h2 47. ♔e2f1 ♖h2g2 48. ♖b7b3 ♖g2d2 49. ♔f1g1 ♔e7f7 50. ♖b3b7 ♔f7f8 51. ♖b7b8 ♔f8f7 52. ♔g1h1 ♖d2g2 53. ♖b8b7 ♔f7e7 54. ♖b7d7 ♔e7e8 55. ♖d7d3 ♖g2f2 56. ♖d3d6 ♔e8e7 57. ♖d6c6 ♖f2f3 58. ♖c6c7 ♔e7d8 59. ♖c7c6 ♔d8d7 60. ♖c6d6 ♔d7e7 61. ♔h1h2 ♖f3f2 62. ♔h2h1 ♖f2f3 63. ♔h1h2 ♖f3f2 64. ♔h2h1 ♖f2g2 65. ♖d6d3 ♖g2c2 66. ♖d3d4 ♖c2c3 67. ♔h1h2 ♖c3a3 68. ♖d4d2 ♖a3a6 69. ♖d2e2 ♖a6a5 70. ♖e2d2 ♖a5a1 71. ♖d2b2 ♖a1a8 72. ♔h2h1 ♖a8c8 73. ♔h1g1 ♖c8a8 74. ♖b2b6 ♔e7d7 75. ♖b6d6 ♔d7e7 76. ♔g1h1 ♖a8a3 77. ♔h1h2 ♖a3a2 78. ♔h2h1 ♖a2a7 79. ♔h1g1 ♖a7a1 80. ♔g1h2 ♖a1a2 81. ♔h2h1 ♖a2e2 82. ♖d6d4 ♖e2g2 83. ♖d4d3 ♔e7e8 84. ♖d3a3 ♖g2e2 85. ♖a3d3 ♖e2e1 86. ♔h1h2 ♖e1e2 87. ♔h2h1 ♖e2b2 88. ♖d3c3 ♖b2a2 89. ♖c3c6 ♔e8d7 90. ♖c6d6 ♔d7e7 91. ♖d6c6 ♖a2e2 92. ♖c6c3 ♖e2g2 93. ♖c3d3

Figure 16: AlphaGateau starts with a closed Sicilian, transposing into the Four Knights Sicilian, following a popular line until move 10, when white moves its white bishop to e2. The pawn structure locks the situation by move 35. Nothing much happens before the game ends in a draw, besides an interesting stalemate trick on move 54

1. e4 e5 2. ♘f3 ♘c6 3. ♗b5 ♘f6 4. d3 ♗c5 5. ♗xc6 dxc6 6. O-O ♕e7 7. ♗g5 O-O 8. ♗h4 h6 9. ♘bd2 b5 10. ♕e1 a5 11. h3 ♗b6 12. ♗g3 ♖e8 13. ♗xe5 a4 14. ♗c3 ♘h5 15. a3 ♘f4 16. ♔h2 f5 17. ♕d1 ♖f8 18. exf5 ♖xf5 19. ♕e1 ♕f7 20. g4 ♖c5 21. ♘e4 ♖d5 22. ♖g1 ♘e6 23. ♘h4 ♘g5 24. ♘f6+ gxf6 25. f4 ♘e6 26. ♖g3 ♘d4 27. ♕e4 ♗d7 28. ♖e1 ♖e8 29. ♕g2 ♖xe1 30. ♗xe1 ♕e6 31. ♗f2 ♘e2 32. ♗xb6 cxb6 33. f5 ♕e5 34. ♘f3 ♕xg3+ 35. ♕xg3 ♘xg3 36. ♔xg3 ♗c8 37. ♔f4 ♖d7 38. ♘d2 ♔f7 39. ♘e4 ♗a6 40. h4 c5 41. g5 fxg5+ 42. hxg5 hxg5+ 43. ♘xg5+ ♔g7 44. ♔e5 ♖e7+ 45. ♘e6+ ♔f7 46. ♔d6 ♗c8 47. ♘g5+ ♔f6 48. ♘e4+ ♔f7 49. ♘g5+ ♔f8 50. f6 ♖d7+ 51. ♔c6 c4 52. ♘e6+ ♔f7 53. ♘f4 ♔xf6 54. ♘d5+ ♔e6 55. ♘xb6 cxd3 56. cxd3 ♗b7+ 57. ♔xb5 ♖xd3 58. ♘xa4 ♔d6 59. ♔c4 ♖h3 60. ♘c3 ♔c6 61. b4 ♗c8 62. a4 ♗e6+ 63. ♔d4 ♗b3 64. a5 ♖h4+ 65. ♔e5 ♖xb4 66. a6 ♔b6 67. ♘b1 ♗c2 68. ♘c3 ♗b3 69. ♘e2 ♔c5 70. ♘f4 ♖a4 71. ♘d3+ ♔c6 72. ♘f4 ♗c4 73. ♘g6 ♗xa6 74. ♘f4 ♗c4 75. ♘g2 ♗b3 76. ♘f4 ♗c2 77. ♘e6 ♖e4+ 78. ♔f5 ♖e1+ 79. ♔f6 ♔d6 80. ♘f4 ♖g1 81. ♘e2 ♖f1+ 82. ♔g5 ♔e5 83. ♘g3 ♖f7 84. ♘h5 ♗d3 85. ♘g3 ♖g7+ 86. ♔h4 ♖g8 87. ♔h3 ♔f4 88. ♘h5+ ♔g5 89. ♘g3 ♖h8+ 90. ♔g2 ♔f4 91. ♘f1 ♖b8 92. ♘g3 ♖b2+ 93. ♔h3 ♗g6 94. ♘h5+ ♗xh5 95. ♔h4 ♖h2#

Figure 17: AlphaGateau starts playing a Berlin defense, without any book, and diverts by move 4 into a popular line, with a rare bishop move on move 7. The midgame revolves around black strong knight, until white is forced to give up its remaining rook to stop the attack, leaving black up a rook for a pawn.

1. e4 c6 2. h3 h6 3. Be2 Na6 4. g4 g6 5. a4 Rh7 6. f4 f6 7. Nf3 Bg7 8. h4 d5 9. e5 Nb4 10. c3 Bxg4 11. cxb4 Bh5 12. Ng1 Bh8 13. Nc3 d4 14. Rh3 a5 15. exf6 Nxf6 16. bxa5 e6 17. Bxh5 gxh5 18. b4 Ng8 19. Rd3 Nf6 20. Nge2 Ng4 21. Qc2 Qxh4+ 22. Ng3 Qh2 23. Nge4 Qh1+ 24. Ke2 Rg7 25. Rx d4 Qh2+ 26. Kf3 Qh1+ 27. Ke2 Rf7 28. Rd3 Qg2+ 29. Ke1 Rg7 30. Ra2 Qh1+ 31. Ke2 Ke7 32. Rg3 Re8 33. Qd3 Rd8 34. Qc4 Qh4 35. Ra1 Rf7 36. Ra2 Rxf4 37. d4 Rf7 38. Be3 Rg7 39. Rc2 Kf7 40. Qc5 Ke8 41. Rf3 Qh2+ 42. Kd3 Qh1 43. Rf4 Nxe3 44. Kxe3 Rg1 45. Re2 Rxd4 46. Nf2 Rxf4 47. Nxh1 Rh4 48. Nf2 Kd7 49. b5 Rg3+ 50. Kd2 Bxc3+ 51. Kc2 Bh8 52. Qb6 Kd6 53. a6 bxa6 54 . Qd8+ Kc5 55. Qxh4 Rg8 56. bxc6 Kxc6 57. Qf4 Bg7 58. Rxe6+ Kd5 59. Qf5+ Kc4 60. Qf3 Kb4 61. Qxh5 Kxa4 62. Rxa6+ Kb4 63. Qf5 Rh8 64. Ra7 Bd4 65. Rb7+ Bb6 66. Nd3+ Ka4 67. Re7 h5 68. Qf6 Bd8 69. Qf4+ Kb5 70. Re5+ Kb6 71. Nf2 Kc7 72. Ne4 Kc6 73. Kc3 Kb6 74. Kd4 Kc6 75. Ng3 Kb6 76. Kc4 Kb7 77. Qf3+ Kc7 78. Kd5 h4 79. Ne2 h3 80. Qf4 Kd7 81. Ng3 Kc7 82. Rh5+ Kd7 83. Rxh8 h2 84. Rxh2 Bc7 85. Qf2 Kc8 86. Qf5+ Kb7 87. Qd7 Ka8 88. Ke4 Bb8 89. Rh6 Ba7 90. Rf6 Bg1 91. Rf5 Kb8 92. Rf8#

图15：仅使用5×5国际象棋（作为白方）完全训练的模型下的一局棋。白方能够使用他们的白象来消除黑方的白象，但在第14步似乎低估了他们的马，可能是因为在5×5国际象棋中马的性能较差，由于活动范围受限且更难有效运用

1. e4 c5 2. Nb1c3 e6 3. Ng1f3 Nb8c6 4. d4 d4 5. Nf3d4 Ng8f6 6. Nd4c6 c6 7. e5 Nf6d5 8. Nc3e4 Qd8c7 9. f4 Qc7b6 10. Bf1e2 Bc8a6 11. Be2a6 Qb6a6 12. a3 h5 13. Qd1e2 Qa6e2 14. Ke1e2 f5 15 . Ne4c3 a5 16. Nc3d5 d5 17. Bc1e3 Bf8e7 18. c3 Rh8g8 19. Ke2f3 g5 20. g3 g4 21. Kf3e2 h4 22. b 4 Ke8f7 23. Ra1c1 h3 24. Be3b6 b4 25. b4 Ra8a3 26. Rc1c7 Rg8b8 27. Bb6c5 Be7c5 28. Rc7c5 R b8b4 29. Rc5c2 d4 30. Rh1d1 Kf7e7 31. Ke2f2 Ra3f3 32. Kf2g1 d3 33. Rc2c3 Rb4b2 34. Rc3d3 R f3d3 35. Rd1d3 Rb2c2 36. Rd3d1 Rc2g2 37. Kg1h1 Rg2a2 38. Kh1g1 Ke7e8 39. Rd1b1 Ra2g2 40. Kg1h1 Rg2c2 41. Kh1g1 Rc2c3 42. Rb1b8 Ke8e7 43. Rb8b1 Rc3c2 44. Rb1b7 Rc2c1 45. Kg1f2 R c1h1 46. Kf2e2 Rh1h2 47. Ke2f1 Rh2g2 48. Rb7b3 Rg2d2 49. Kf1g1 Ke7f7 50. Rb3b7 Kf7f8 51. Rb7b8 Kf8f7 52. Kg1h1 Rd2g2 53. Rb8b7 Kf7e7 54. Rb7d7 Ke7e8 55. Rd7d3 Rg2f2 56. Rd3d6 K e8e7 57. Rd6c6 Rf2f3 58. Rc6c7 Ke7d8 59. Rc7c6 Kd8d7 60. Rc6d6 Kd7e7 61. Kh1h2 Rf3f2 62. Kh2h1 Rf2f3 63. Kh1h2 Rf3f2 64. Kh2h1 Rf2g2 65. Rd6d3 Rg2c2 66. Rd3d4 Rc2c3 67. Kh1h2 Rc 3a3 68. Rd4d2 Ra3a6 69. Rd2e2 Ra6a5 70. Re2d2 Ra5a1 71. Rd2b2 Ra1a8 72. Kh2h1 Ra8c8 73. K h1g1 Rc8a8 74. Rb2b6 Ke7d7 75. Rb6d6 Kd7e7 76. Kg1h1 Ra8a3 77. Kh1h2 Ra3a2 78. Kh2h1 Ra 2a7 79. Kh1g1 Ra7a1 80. Kg1h2 Ra1a2 81. Kh2h1 Ra2e2 82. Rd6d4 Re2g2 83. Rd4d3 Ke7e8 84. Rd3a3 Rg2e2 85. Ra3d3 Re2e1 86. Kh1h2 Re1e2 87. Kh2h1 Re2b2 88. Rd3c3 Rb2a2 89. Rc3c6 K e8d7 90. Rc6d6 Kd7e7 91. Rd6c6 Ra2e2 92. Rc6c3 Re2g2 93. Rc3d3

图16：AlphaGateau 从一个封闭的西西里开局，转为四骑士西西里，遵循一条流行路线直到第10回合，此时白方将其白象移至e2。在第35回合时，棋盘的兵形结构锁定了局势。在比赛结束成和棋之前，除了第54回合的一个有趣的僵局技巧外，没有什么事情发生。

1. e4 e5 2. Nf3 Nc6 3. Bb5 Nf6 4. d3 Bc5 5. Bxc6 dxc6 6. O-O Qe7 7. Bg5 O-O 8. Bh4 h6 9. Nbd2 b5 10. Qe1 a5 11. h3 Bb6 12. Bg3 Re8 13. Bxe5 a4 14. Bc3 Nh5 15. a3 Nf4 16. Kh2 f5 17. Qd1 Rf 8 18. exf5 Rxf5 19. Qe1 Qf7 20. g4 Rc5 21. Ne4 Rd5 22. Rg1 Ne6 23. Nh4 Ng5 24. Nf6+ gxf6 25. f4 Ne6 26. Rg3 Nd4 27. Qe4 Bd7 28. Re1 Re8 29. Qg2 Rxe1 30. Bxe1 Qe6 31. Bf2 Ne2 32. Bxb6 cxb6 33. f5 Qe5 34. Nf3 Qxg3+ 35. Qxg3 Nxg3 36. Kxg3 Bc8 37. Kf4 Rd7 38. Nd2 Kf7 39. Ne4 Ba6 40. h4 c5 41. g5 fxg5+ 42. hxg5 hxg5+ 43. Nxg5+ Kg7 44. Ke5 Re7+ 45. Ne6+ Kf7 46. Kd6 Bc8 47. Ng5+ Kf6 48. Ne4+ Kf7 49. Ng5+ Kf8 50. f6 Rd7+ 51. Kc6 c4 52. Ne6+ Kf7 53. Nf4 Kx f6 54. Nd5+ Ke6 55. Nxb6 cxd3 56. cxd3 Bb7+ 57. Kxb5 Rxd3 58. Nc3 Kd6 59. Kc4 Rh3 60. Nc3 Kc6 61. b4 Bc8 62. a4 Be6+ 63. Kd4 Bb3 64. a5 Rh4+ 65. Ke5 Rxb4 66. a6 Kb6 67. Nb1 Bc2 68. Nc3 Bb3 69. Ne2 Kc5 70. Nf4 Ra4 71. Nd3+ Kc6 72. Nf4 Bc4 73. Ng6 Bxa6 74. Nf4 Bc4 75. Ng2 Bb3 76. Nf4 Bc2 77. Ne6 Re4+ 78. Kf5 Re1+ 79. Kf6 Kd6 80. Nf4 Rg1 81. Ne2 Rf1+ 82. Kg5 Ke 5 83. Ng3 Rf7 84. Nh5 Bd3 85. Ng3 Rg7+ 86. Kh4 Rg8 87. Kh3 Kf4 88. Nh5+ Kg5 89. Ng3 Rh8 + 90. Kg2 Kf4 91. Nf1 Rb8 92. Ng3 Rb2+ 93. Kh3 Bg6 94. Nh5+ Bxh5 95. Kh4 Rh2#

图17：AlphaGateau开始走柏林防御，没有参考任何棋谱，并在第4步偏离常规路线进入一个流行开局，第7步的罕见bishop走法。中局围绕着黑方强大的骑士展开，直到白方被迫放弃剩余的车以阻止攻击，从而黑方多一车一兵的优势。

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The results discussed in the experiments and illustrated in Figures 5, 6, and 7 support the claims in the abstract and the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Our main limitations are that we had limited computing resources and had models of depth 5 or 10 when 40 would be better, and we only focused on chess and not other games. We could also only run the experiments once in the time available to us, so the confidence intervals only apply to the rating estimations, and were not extimated over several training runs.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

NeurIPS论文检查清单

1. 声称

   问题：摘要和引言中提出的主旨是否准确反映了论文的贡献和范围?

   Answer: [是的]

   论证：实验结果和图5、6和7中所示的支持了摘要和引言中的论点。

   指南：

   - The answer NA意味着摘要和引言不包括论文中提出的主张。
   - 摘要和/或引言应清楚地陈述所提出的主张，包括论文中的贡献、重要假设和限制。对这个问题的回答如果是“No”或“NA”，不会被评审人很好地接受。
   - 声明应与理论和实验结果相符，并反映这些结果预期能在其他设置中泛化的能力。
   - 它可以包含进取性的目标以作为激励，只要这些目标清楚地不是论文所达到的。

2. 限制

   问题: 论文是否讨论了作者所完成工作的局限性?

   Answer: [是的]

   正当性说明：我们的主要限制是计算资源有限，模型的深度为5或10，而40会更好，并且我们仅专注于国际象棋而没有涉及其他游戏。此外，由于可用时间限制，我们只能运行一次实验，因此置信区间仅适用于评级估计，而不是基于多次训练运行的估计。

   指南：

   - The answer NA意味着论文没有限制，而回答No意味着论文有未讨论的限制。
   - 作者被鼓励在他们的论文中创建一个单独的“局限性”部分。
   - 论文应指出任何强有力的假设，并说明这些假设被违反时结果的稳健性（例如，独立性假设、无噪声设置、模型良好指定、渐近近似仅局部成立）。作者应反思这些假设在实践中可能如何被违反及其影响。
   - 作者应该反思所提出的主张的范围，例如，如果该方法仅在几个数据集或几次运行中进行了测试。一般来说，实证结果往往依赖于一些隐含的假设，这些假设应该被阐明。
   - 作者应该反思影响该方法性能的因素。例如，当图像分辨率低或图像在低光环境下拍摄时，面部识别算法可能会表现不佳。或者，由于无法处理技术术语，语音转文本系统可能无法可靠地为在线讲座提供字幕。
   - 作者应该讨论所提出的算法的计算效率及其随数据集大小的扩展情况。
   - 如果适用，作者应讨论其方法的可能局限性，以解决隐私和公平性问题。
   - 虽然作者可能会担心完全诚实地承认局限性可能会被审稿人用作拒绝论文的理由，但更糟糕的结果可能是审稿人发现论文中没有承认的局限性。作者应该运用他们的最佳判断，并认识到个体制裁有利于透明度的行为在培养维护社区 integrity 的规范方面发挥着重要作用。审稿人将被明确指示不要因承认局限性的诚实而对其进行惩罚。

3. 理论假设与证明

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The only theoretical result is the closed form derivation of the confidence interval for the estimated Elo ratings, included in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We included all the experimental hyperparameters and methodology in the paper, and include all the code that was used in the GitHub repository mentioned in the abstract. However, as the parallelized GPU code is not fully deterministic, it is not possible to replicate the exact training results we present, so we will also publish in the git repository some important model parameter checkpoints that we saved.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

问题: 对于每个理论结果，论文是否提供了完整的假设集和完整的（正确）证明?

Answer: [是的]

证明：唯一的理论结果是估计的Elo评分的置信区间闭式推导，详见附录。

指南:

- The answer NA意味着论文中不包括理论结果。
- 论文中的所有定理、公式和证明都应该编号并交叉引用。
- 所有假设应在任何定理的陈述中明确指出或引用。
- 证明可以出现在主要论文或补充材料中，但如果证明出现在补充材料中，作者被鼓励提供一个简短的证明草图以提供直观理解。
- 相反，论文核心部分提供的任何非正式证明应由附录或补充材料中提供的正式证明加以补充。
- 证明依赖的定理和引理应适当引用。

4. 实验结果可重复性

问题: 论文是否充分披露了所有用于重现论文主要实验结果的信息，以至影响了论文的主要主张和/或结论（无论代码和数据是否提供）？

Answer: [是的]

Justification: 我们在论文中包含了所有实验超参数和方法论，并在摘要中提到的GitHub仓库中发布了所有使用的代码。然而，由于并行化的GPU代码不是完全确定性的，因此无法复制我们呈现的精确训练结果，因此我们还将发布一些我们在git仓库中保存的重要模型参数检查点。

指南:

- The answer NA意味着论文中没有包含实验。
- 如果论文包含实验，对这个问题给出“没有答案”将不会被评审人很好地接受：无论是否提供代码和数据，使论文可重复是非常重要的。
- 如果贡献是一个数据集和/或模型，作者应该描述他们采取的步骤以使其结果可重现或可验证。
- 根据贡献的不同，可重现实验可以采用多种方式实现。例如，如果贡献是一种新颖的架构，那么完整描述该架构可能就足够了；如果贡献是一种特定的模型和经验性评估，那么可能需要使其他人能够使用相同的数据集复现该模型，或者提供模型的访问途径。一般来说，发布代码和数据通常是实现这一目标的一个好方法，但可重现实验也可以通过提供详细的复现结果的步骤说明、访问托管模型（例如，在大型语言模型的情况下）、发布模型检查点或其他适用于所进行研究的方法来实现。
- 虽然 NeurIPS 不要求发布代码，但该会议要求所有提交内容提供某种合理的可再现途径，这可能取决于贡献的性质。例如：(a) 如果贡献主要是一种新算法，论文应清楚说明如何再现该算法。(b) 如果贡献主要是一种新的模型架构，论文应清楚完整地描述该架构。(c) 如果贡献是一种新模型（例如，一个大型语言模型），则应有一种方法可以访问该模型以再现结果，或者有一种方法可以再现该模型（例如，使用开源数据集或如何构建数据集的说明）。

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The full code is published in the GitHub repository Akulen/AlphaGateau.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The full training procedure is described in the paper, and the provided code contains the implementation details, such as the gpu data split procedure.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide confidence intervals on our estimated Elo ratings, however, as each model was only trained once, they are incomplete confidence intervals.

   Guidelines:

(d) 我们认识到在某些情况下可重复性可能会有些棘手，在这种情况下，作者可以描述他们提供可重复性的具体方式。对于封闭源代码的模型，可能访问模型的方式会受到某种限制（例如，仅限注册用户），但其他研究人员应该能够找到某种途径来重现或验证结果。

5. 数据和代码的开放访问

   问题: 论文是否提供了开放访问的数据和代码，并提供了足够的说明以忠实重现主要实验结果（如补充材料中所述）？

   Answer: [是的]

   Justification: 完整代码发布在 GitHub 仓库 Akulen/AlphaGateau。

   指南:

   - The answer NA意味着论文不包括需要代码的实验。
   - 请参阅 NeurIPS 代码和数据提交指南 (https://nips.cc/public/guides/CodeSubmissionPolicy) 以获取更多详细信息。
   - 虽然我们鼓励发布代码和数据，但我们理解这可能并不总是可行的，因此“不行”是一个可接受的答案。论文不能仅仅因为没有包含代码而被拒绝，除非这对于贡献至关重要（例如，对于一个新的开源基准）。
   - 指令应包含用于重现结果所需的精确命令和环境。更多细节请参见 NeurIPS 代码和数据提交指南 (https://nips.cc/public/guides/CodeSubmissionPolicy)。
   - 作者应提供数据访问和准备的说明，包括如何访问原始数据、预处理数据、中间数据和生成数据等。
   - 作者应该提供脚本来重现新提出的方法和基线的所有实验结果。如果只有部分实验可以重现，他们应该说明哪些实验被脚本省略以及原因。
   - 提交时，为了保护匿名性，作者应发布匿名版本（如果适用）。
   - 提供尽可能多的补充材料（附在论文后面）中的信息是推荐的，但包含数据和代码的URL也是允许的。

6. 实验设置/细节

   问题: 研究论文是否详细指明了所有用于理解结果的训练和测试细节（例如，数据分割、超参数、选择方式、优化器类型等）？

   Answer: [是的]

   正当性说明：完整的训练流程已在论文中描述，提供的代码中包含了实施细节，例如gpu数据分割过程。

   指南:

   - The answer NA意味着论文中没有包含实验。
   - 实验设置应在论文的核心部分以足够的细节呈现，以便读者能够理解并解读结果。
   - 完整的细节可以与代码一起提供，在附录中，或作为补充材料。

7. 实验统计显著性

   问题: 文章是否适当地并正确地报告了误差棒或其他关于实验统计显著性的适当信息?

   Answer: [是的]

   证明：我们提供了我们估计的Elo等级的置信区间，然而，由于每个模型只训练了一次，这些是不完整的置信区间。

   指南:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include the number and type of GPU, as well as the experiment run time.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

- The answer NA意味着论文中没有包含实验。
- 作者应该在结果伴有误差棒、置信区间或统计显著性检验（至少是支持论文主要论点的实验）时回答“是”。
- 误差棒捕获的可变性因素应明确说明（例如，训练/测试分割、初始化、某些参数的随机抽取，或给定实验条件下的整体运行）。
- 计算误差棒的方法应该解释清楚（闭形式公式、调用库函数、-bootstrap等）
- 所做的假设应该被给出（例如，正态分布的误差）。
- 应该清楚误差条是标准偏差还是标准误。
- 可以报告1-σ误差棒，但应加以说明。如果误差的正态性假设未被验证，作者最好报告2-σ误差棒，而不是声明其置信区间为96%。
- 对于非对称分布，作者在表格或图表中不应显示对称误差棒，以免得出超出范围的结果（例如负误差率）。
- 如果在表格或图表中报告了误差条，则作者应在文本中解释这些误差条是如何计算的，并引用相应的图表或表格。

8. 实验计算资源

   问题: 对于每项实验，论文是否提供了足够的信息（包括计算工作者的类型、内存、执行时间）以重现这些实验?

   Answer: [是的]

   Justification: 我们包括GPU的数量和类型，以及实验运行时间。

   指南:

   - The answer NA意味着论文中没有包含实验。
   - 论文应表明计算工作者的类型，是CPU或GPU，内部集群，还是云提供商，并包括相关的内存和存储信息。
   - 论文应该提供每项单独实验所需的计算量，并估计总计算量。
   - 论文应披露整个研究项目所需的计算资源是否超过了论文中报告的实验（例如，未纳入论文的初步或失败的实验）所需的计算资源。

9. 廉洁守则

   问题: 研究论文中的研究在每一个方面都符合 NeurIPS 伦理守则 https://neurips.cc/public/EthicsGuidelines?

   Answer: [是的]

   证明：Justification:证明确实是“证明”的意思，但源文本中“Justification”一词在不同上下文中可能有不同

   指南:

   - The answer NA意味着作者尚未审核NeurIPS伦理代码。
   - 如果作者回答否，他们应该解释需要偏离伦理守则的特殊情况。
   - 作者应该确保保护匿名性（例如，如果由于其管辖区域的法律或规定有特殊考虑）。

10. 更广泛的影响力

    问题: 文章是否讨论了该工作可能产生的正面社会影响和负面社会影响?

    Answer: [NA]

Justification: We introduce an improvement to an existing DRL architecture to train a game agent. There should be no direct societal impact from this, as the results are currently limited to making research on this topic more accessible, due to the training efficiency gains.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The released models are way less powerful than current existing chess engines, as they are limited in size and depth for compute reasons, so there is no risk in publishing them.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite every non-standard python library used to develop and run the experiments presented in this paper. We do not use any previous dataset or asset besides those libraries.

正当性：我们对现有的DRL架构进行改进，以训练游戏代理。目前这一改进没有直接的社会影响，因为其结果仅限于使该主题的研究更加 accessible，得益于训练效率的提升。

指南：

- The answer NA意味着该项工作没有社会影响。
- 如果作者回答NA或No，他们应该解释为什么他们的工作没有社会影响，或者为什么论文没有涉及社会影响。
- 负面社会影响的例子包括潜在的恶意或意外使用（例如，虚假信息、生成虚假资料、监控），公平性考虑（例如，部署可能不公平地影响特定群体的决策的技术），隐私考虑，以及安全考虑。
- 会议预计许多论文将是基础研究，并且不会特别针对特定的应用，更不用说部署了。然而，如果有直接途径导致任何负面应用，作者应该指出这一点。例如，指出生成模型的质量改进可能被用于生成用于虚假信息的深fake是合理的。另一方面，并不需要指出一个用于优化神经网络的通用算法可以使人们更快地训练生成Deepfake的模型。
- 作者应该考虑该技术按预期使用且功能正常时可能产生的潜在危害，技术按预期使用但给出错误结果时可能产生的潜在危害，以及技术被故意或无意误用时可能产生的危害。
- 如果存在负面影响，作者也可以讨论可能的缓解策略（例如，模型的受控发布、提供防御而非仅仅攻击、监测滥用的机制、监测系统如何随时间反馈学习的机制、提高机器学习的效率和可访问性）。

11. 保障措施

    问题: 文章是否描述了为负责任地发布高风险易误用的数据或模型（例如，预训练语言模型、图像生成器或抓取的数据集）所采取的保护措施?

    Answer: [NA]

    证明：释放的模型远不如当前现有的国际象棋引擎强大，因为它们在计算原因上受到大小和深度的限制，因此发布它们是没有风险的。

    指南：

    - The answer NA意味着该论文不存在此类风险。
    - 释放具有高风险误用或双重用途的模型时，应采取必要措施以控制模型的使用，例如要求用户遵守使用指南或限制访问模型，或实施安全过滤。
    - 从互联网抓取的数据集可能会带来安全风险。作者应该描述他们是如何避免发布不安全图像的。
    - 我们认识到提供有效的保护措施具有挑战性，而且许多论文不需要这样做，但我们鼓励作者将这一点考虑在内，并尽最大努力。

12. 现有资产的许可

    问题: 研究中使用的资产（例如，代码、数据、模型）的创建者或原始所有者是否得到了适当的认可，并且许可证和使用条款是否明确提及并得到了适当尊重?

    Answer: [是的]

    正当性说明：我们引用了开发和运行本文中呈现的实验所使用的每一个非标准Python库。除了这些库之外，我们没有使用任何先前的数据集或资产。

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The Git repo that will be published along the paper will have well structured code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

指南:

- The answer NA意味着论文未使用现有资产。
- 作者应该引用原始论文，该论文产生了代码包或数据集。
- 作者应声明使用的是资产的哪个版本，并在可能的情况下提供一个URL。
- 每个资产都应该包含许可名称（例如，CC-BY 4.0）。
- 对于从特定来源（例如网站）抓取的数据，应该提供该来源的版权和服务条款。
- 如果释放了资产，包中应提供许可证、版权信息和使用条款。对于流行的数据集，paperswithcode.com/datasets 已为一些数据集整理了许可证。他们的许可指南可以帮助确定数据集的许可证。
- 对于重新打包的现有数据集，应提供原始许可和衍生资产的许可（如果已更改），{v*}
- 如果这些信息无法在线获得，作者们被鼓励联系资产的创作者。

13. 新资产

问题: 文中引入的新资产是否得到了充分记录，并且这些记录是否与资产一同提供？

Answer: [是的]

正当性说明：随论文发布的 Git 仓库将包含结构良好的代码。

指南:

- The answer NA意味着论文不发布新资产。
- 研究人员应通过结构化的模板在其提交中传达数据集/代码/模型的详细信息。这包括训练细节、许可、限制等。
- 论文应该讨论是否以及是如何获得使用他人资产时的同意。
- 在提交时，请记住对您的资产进行匿名化（如果适用）。您可以创建一个匿名化链接或包含一个匿名化的压缩文件。

14. 众包与人类受试者研究

问题: 对于包含人类受试者的 crowdsourcing 实验和研究，论文是否包含了提供给参与者的所有说明文本（如果适用），以及屏幕截图，还有关于补偿的详细信息（如果有）？

Answer: [NA]

证明：Justification:证明确实是“证明”的意思，但源文本中“Justification”一词在不同上下文中可能有不同

指南:

- The answer NA意味着论文既不涉及众包也不涉及涉及人类受试者的研究。
- 包括这些信息在补充材料中是可以的，但如果论文的主要贡献涉及人类受试者，那么尽可能多的细节应该包含在主要论文中。
- 根据 NeurIPS 的伦理规范，参与数据收集、整理或其他劳动的工人应至少获得数据收集国的最低工资。

15. 机构审查委员会（IRB）批准或等效批准用于人类受试者的研究

问题: 论文是否描述了研究参与者可能面临的潜在风险，这些风险是否已向受试者披露，以及是否获得了机构审查委员会（IRB）的批准（或等效的批准/审查，基于您所在国家或机构的要求）？

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

Answer: [NA]

证明：Justification:证明确实是“证明”的意思，但源文本中“Justification”一词在不同上下文中可能有不同

指南：

- The answer NA意味着论文既不涉及众包也不涉及涉及人类受试者的研究。
- 根据进行研究的国家不同，任何涉及人类受试者的研究可能都需要IRB批准（或同等批准）。如果您获得了IRB批准，您应在论文中明确说明这一点。
- 我们认识到这些程序在不同机构和地点之间可能会有很大差异，并期望作者遵守NeurIPS伦理守则及其所在机构的指导原则。
- 对于初始提交，如果不适用，请勿包含任何可能破坏匿名性的信息，例如进行审查的机构。