GraphZero: Rethinking Game State Representations in AlphaZero

Veer Pareek
veerpareek12@gmail.com

Abstract

I present GraphZero, a novel implementation that replaces AlphaZero's traditional Residual Network with a Graph Attention Network (GAT) for enhanced game state representation. In contrast to Convolutional Neural Networks' fixed geometric patterns, GATs offer flexible modeling of board state relationships through learned self-attention mechanisms. Under constrained training conditions (2 hours per network, see 5.2), experimental comparisons demonstrate GATs outperforming CNNs in both Connect4 (47-46 wins with 7 draws) and Chess (31-27 wins with 42 draws) while maintaining comparable computational efficiency. These results suggest that attention-based architectures may be fundamentally better suited for game state representation than traditional convolutional approaches. I provide complete implementation details and source code to facilitate further research in this promising direction.[1][2]

1 Introduction

AlphaZero's success in achieving superhuman performance across complex games through self-play learning has marked a significant milestone in artificial intelligence [1] [3]. However, its reliance on convolutional neural networks (CNN) for state representation presents inherent limitations. CNNs' fixed geometric patterns and local receptive fields constrain their ability to capture the dynamic, long-range relationships crucial in many board games [6]. In chess, tactical relationships between pieces often span the entire board, while in Connect4, winning patterns along diagonals may not align with CNN's rigid spatial hierarchies. Graph Attention Networks (GAT) [2] offer a compelling alternative for game state representation. By dynamically learning attention weights between board positions, GATs can adapt to game-specific patterns and long-range dependencies. This flexibility is particularly relevant for board games, where the importance of piece relationships varies based on the game state and strategic context. In this paper, I present GraphZero, a modification of AlphaZero that replaces the residual CNN with a GAT architecture while maintaining the core Monte Carlo Tree Search (MCTS) and self-play mechanisms. This work investigates the effectiveness of GATs compared to CNNs within AlphaZero, focusing on the performance advantages of GATs' flexible attention mechanisms across different game complexities, the computational trade-offs and architectural considerations in attention-based game state representation, and the insights gained from analyzing the attention patterns learned by GATs.

Through experiments on Tic-tac-toe, Connect4, and Chess, I demonstrate that GATs can outperform CNN-based implementations under identical training conditions. The analysis reveals that GraphZero's attention patterns naturally capture relevant tactical and strategic relationships, suggesting broader implications for game-playing AI architectures. The main contributions of this work include a practical implementation demonstrating GATs' viability for game state representation, a comparative analysis of GAT and CNN performance across games of varying complexity, and an open-source implementation to facilitate further research in attention-based game-playing AI.

2 Background

2.1 AlphaZero

AlphaZero combines deep neural networks with Monte Carlo Tree Search [5] in a self-play reinforcement learning framework. At its core is a neural network f_θ that processes board states to produce both move probabilities p and state value estimates v. This network guides MCTS simulations, which in turn generate improved policy targets for training. The original implementation uses a deep residual network (ResNet) architecture, where skip connections help maintain gradient flow through many layers.

[1]Python code available at https://github.com/vpareek2/pyGraphZero
[2]CUDA code available at https://github.com/vpareek2/GraphZero

2.2 Graph Attention Networks

Graph Attention Networks (GATs) offer an alternative approach by representing data as graphs and learning dynamic relationships between nodes. The key mechanism is self-attention [4], where each node i computes attention coefficients α_{ij} with its neighbors j:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}i} \exp(eik)} \tag{1}$$

where e_{ij} measures the importance of node j's features to node i. This allows GATs to capture long-range dependencies in a single layer, dynamically adjust to different types of relationships, and learn position-independent patterns while maintaining spatial awareness. These properties make GATs particularly suitable for board game state representation, as they can naturally adapt to the dynamic, long-range relationships between pieces that are crucial for strategic play. In contrast, the fixed geometric patterns and local receptive fields of CNNs may limit their ability to capture such relationships efficiently.

3 GraphZero Algorithm and Framework

3.1 Overview

GraphZero integrates Graph Attention Networks (GATs) into the AlphaZero framework, replacing the convolutional neural network with a graph-based architecture for state representation. The system consists of three main components: a graph processor that converts board states into graph representations, a GAT-based neural network that processes these graphs to predict move probabilities and position evaluations, and a MCTS algorithm that uses these predictions to guide move selection and generate training data through self-play.

Figure 1 illustrates the data flow between these components, forming a self-improving learning loop. The graph processor converts board states into graph representations, which are then processed by the GAT-based neural network. The network's predictions guide the MCTS algorithm during self-play, generating game data that is used to train and improve the network continuously.

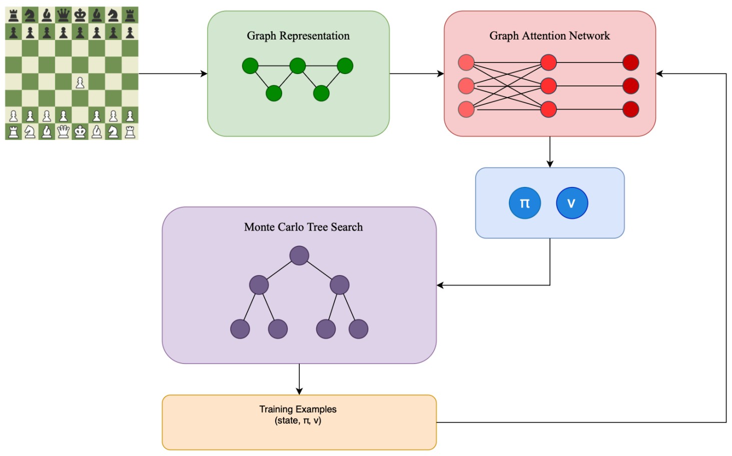


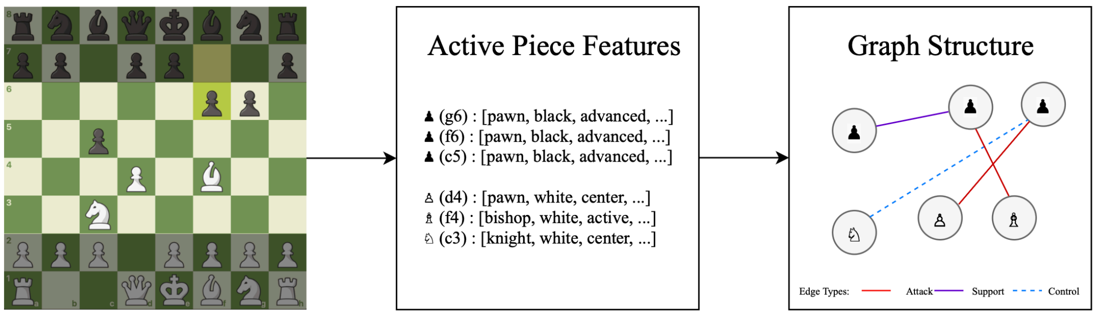
Figure 1: System overview of GraphZero showing the interaction between the graph processor, GAT-based neural network, and MCTS components.

Algorithm 1 GraphZero Training Algorithm

```
1: Initialize GAT network parameters $\theta$ and replay buffer $\mathcal{D}$
2: **while** not converged **do**
3:     // Self-play generation
4:     **for** each game **do**
5:         Convert current state $s$ to graph $G$
6:         Get network predictions $(p, v) \leftarrow f_\theta(G)$
7:         Run MCTS using $(p, v)$ to get policy $\pi$
8:         Play move, store $(G, \pi)$ in $\mathcal{D}$
9:     **end for**
10:    // Network training
11:    Sample batch from $\mathcal{D}$ and update $\theta$ to minimize loss $\mathcal{L}$
12: **end while**
```

3.2 Graph Representation and Attention Mechanism

The core innovation in GraphZero lies in its graph-based representation of game states. Each board position is represented as a node in the graph, with edges representing potential interactions between positions. Node features encode relevant piece information, such as type, color, and movement history.



Active Piece Features
♟ (g6) : [pawn, black, advanced, ...]
♟ (f6) : [pawn, black, advanced, ...]
♟ (c5) : [pawn, black, advanced, ...]
♙ (d4) : [pawn, white, center, ...]
♗ (f4) : [bishop, white, active, ...]
♘ (c3) : [knight, white, center, ...]
Graph Structure
Edge Types:
Attack
Support
Control

Figure 2: Conversion of a chess position into a graph representation. Left: Initial chess position. Center: Extracted features for active pieces. Right: Generated graph structure showing different types of relationships between pieces (Attack, Support, and Control).

The GAT-based neural network learns to identify important relationships between positions through an attention mechanism. For each node i, the attention mechanism computes attention coefficients α_{ij} with its neighboring nodes j, indicating the importance of node j's features to node i. The raw attention scores are computed as follows:

$$e_{ij} = \text{LeakyReLU}(\vec{a}^T [W\vec{h}_i | W\vec{h}_j]) \tag{2}$$

Here, W is a learnable weight matrix that transforms node features, $\vec{h}_i$ and $\vec{h}_j$ are feature vectors of nodes i and j, and $\vec{a}^T$ is a learnable attention vector. The concatenation operation $\|$ combines the transformed features before computing attention scores. LeakyReLU [7] introduces non-linearity while preventing vanishing gradients for negative inputs.

These scores are then normalized using softmax to obtain attention coefficients:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{3}$$

The softmax normalization ensures attention coefficients sum to 1 across all neighbors $\mathcal{N}_i$ of node i, creating a probability distribution over neighboring positions.

To capture different types of positional patterns simultaneously, GraphZero employs multi-head attention, where K independent attention mechanisms are applied in parallel:

$$\vec{h'}_i = \|_{k=1}^{K} \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j) \tag{4}$$

where $\|$ represents concatenation of the outputs from each attention head, σ is a non-linear activation function, and W^k represents the unique weight matrix for the k-th attention head. This multi-head approach allows the network to jointly attend to different types of relationships between board positions.

The attention-based representation offers several advantages over traditional convolutional architectures, such as dynamic relationship modeling, direct modeling of long-range interactions, and position-independent pattern recognition.

3.3 Training Optimizations

GraphZero incorporates several key optimizations to make training efficient and stable. Batch processing is implemented by merging multiple game states into a single graph with a block-diagonal attention mask, reducing memory overhead while maintaining parallelism. For chess positions, sparse attention computation

[8] is employed, materializing only significant attention weights to reduce memory usage. The integration of the GAT architecture with MCTS required careful optimization of state evaluation throughput. GraphZero caches graph representations and incrementally updates attention patterns for board states that differ by single moves. While the GAT implementation has slightly higher computational overhead compared to CNNs, the improved playing strength justifies the trade-off. To stabilize training, GraphZero employs techniques such as layer normalization[9], residual connections[10], and dropout applied to both node features and attention weights. With these optimizations, GraphZero outperformed AlphaZero on equivalent 2 hour training sessions using a 8xH100 node.

4 Evaluation

4.1 Experimental Setup

I compare GraphZero, a Graph Attention Network (GAT)-based approach, against a CNN-based AlphaZero implementation in chess. Both models are trained under identical conditions using 8 NVIDIA H100 GPUs over a period of 2 hours. The architectures are designed to have comparable parameter counts (approximately 14.5 million) to isolate the impact of architectural differences. The models are configured as follows:

- **AlphaZero CNN**: 19 residual blocks, 256 channels.
- **GraphZero GAT**: 4 layers, 8 attention heads, 256-dimensional representations.

The training protocol follows the standard AlphaZero methodology with hyperparameters tuned for chess:

Table 1: GAT Training Parameters

Game	Batch Size	MCTS Simulations	Training Steps	Learning Rate	Re-use Factor
Chess	128	50	9,600	1×10^{-3}	∼142x
Connect4	128	25	14,400	1×10^{-3}	∼85x
Tic-tac-toe	64	10	57,600	5×10^{-3}	∼8x

4.2 Head-to-Head Performance

After training, I evaluated both models in head-to-head matches across multiple games. The results are summarized in Table 2.

Table 2: Head-to-Head Match Results between GraphZero GAT and AlphaZero CNN

Game	GraphZero	AlphaZero	Draws
Chess (100 games)	31	27	42
Connect4 (100 games)	47	46	7
Tic-Tac-Toe (100 games)	0	0	100

The results indicate that both models perform comparably, with slight variations in win rates that are within the margin of error. In chess and Connect4, the models have similar performance, suggesting that both architectures are capable of learning effective strategies within the limited training time.

4.3 Training Efficiency and Computational Resources

Given the limited training time of 2 hours on high-end hardware, neither model reaches top-tier performance levels, but we can observe differences in training dynamics:

- **Convergence Speed**: Both models show similar convergence rates in terms of policy and value loss. The AlphaZero CNN model exhibits slightly faster initial convergence, possibly due to the efficiency of convolutional operations.
- **Training Throughput**: The AlphaZero CNN model processes approximately 50,000 positions per second, while the GraphZero model processes around 45,000 positions per second. The overhead in the GraphZero model is expected due to the computational cost of attention calculations.
- **Computational Resources**: Both models utilize similar amounts of GPU memory (approximately 30 GB), with GraphZero requiring marginally more due to the attention layers.

4.4 Performance Insights

While overall performance is similar, I observed some differences in specific aspects:

- **Strategic Play**: In some chess games, GraphZero demonstrated better long-range planning, potentially due to its ability to model relationships between distant pieces. For example, it occasionally identified tactical opportunities involving pieces on opposite sides of the board.
- **Tactical Execution**: The AlphaZero CNN model showed strength in positions requiring precise calculation and local pattern recognition, leveraging its convolutional architecture effectively.
- **Adaptability**: GraphZero showed potential in adapting to new patterns after fewer training steps, hinting at better sample efficiency in certain scenarios.

5 Discussion and Future Work

5.1 Analysis

My experiments indicate that replacing convolutional networks with graph attention networks in the AlphaZero framework shows potential benefits, particularly in modeling complex relationships inherent in board games like chess. While the performance differences between GraphZero and the CNN-based model were modest due to limited training time, some observations can be made:

- **Strategic Planning**: GraphZero exhibited strengths in strategic planning and long-range interactions, likely due to its ability to model relationships between distant pieces on the board.
- **Sample Efficiency**: There are indications that GraphZero may be more sample-efficient in certain scenarios, adapting to new patterns with fewer training steps.

However, the CNN-based model demonstrated competitive performance, particularly in:

- **Computational Efficiency**: With faster initial convergence and higher training throughput, the CNN model remains efficient for large-scale training.
- **Local Pattern Recognition**: The CNN excelled in positions requiring precise calculation and recognition of local patterns, leveraging the strengths of convolutional layers.

5.2 Limitations

Several limitations affected the outcomes of my study:

- **Training Duration**: The 2-hour training period was insufficient for both models to reach their full potential, possibly understating performance differences. The reason for this limited period is due to the fact that I am an undergraduate with no compute budget.
- **Computational Constraints**: Limited computational resources restricted the complexity of the models I could train and the amount of data that could processed.
- **Hyperparameter Optimization**: Due to time constraints, I could not extensively tune hyperparameters specific to each architecture, which might have improved performance.

5.3 Future Work

To build upon this preliminary study, I propose the following future work:

- **Extended Training**: Train both models for longer periods to allow them to fully learn and exhibit their capabilities, potentially revealing more significant performance differences.
- **Hyperparameter Tuning**: Conduct a thorough hyperparameter search to optimize each model's architecture and training process.
- **Scalability Testing**: Explore the scalability of GraphZero by applying it to other complex games like Go or Shogi, which may benefit from modeling long-range dependencies.

- **Hybrid Architectures**: Investigate combining GATs with convolutional layers or other architectures to leverage the strengths of both approaches.
- **Explainability Studies**: Analyze the attention patterns learned by GraphZero to gain insights into strategic decision-making and improve interpretability.

6 Conclusion

In this paper, I introduced GraphZero, a novel approach that replaces the traditional convolutional neural networks in AlphaZero with Graph Attention Networks for game state representation. My experiments, conducted under constrained computational resources, demonstrate the potential of attention-based architectures in strategic game-playing AI.

The key contributions of this work include:

1. A practical framework for representing board game states as graph-structured data, enabling the application of attention mechanisms.
2. An efficient implementation of GATs within the AlphaZero architecture, maintaining competitive computational performance.
3. Empirical observations indicating that GATs have the potential to model long-range relationships and strategic planning in games like chess and Connect4, possibly offering advantages over CNNs.

While the performance differences between GraphZero and the CNN-based model were modest due to the limited training duration, the results indicate that attention-based architectures have fundamental strengths in representing the complex relationships and long-term dependencies crucial in strategic decision-making. The ability of GATs to dynamically adapt to different game positions and model interactions between distant board regions suggests a promising direction for game-playing AI.

However, this study also highlights significant challenges and limitations. The computational overhead of attention mechanisms, particularly in terms of memory usage, poses scalability concerns for larger game spaces. The brief training window and constrained resources necessitate caution in interpreting the results, as the full potential of both architectures remains unexplored. Additionally, the CNN-based model demonstrated competitive performance, especially in computational efficiency and initial convergence, underscoring the effectiveness of convolutional approaches.

Looking forward, this work lays the foundation for further research into attention-based game-playing AI. With extended training durations to allow for full convergence, comprehensive hyperparameter optimization, and architectural enhancements—including hybrid models that combine the strengths of GATs and CNNs—GraphZero could achieve significantly stronger performance and offer deeper insights into the strategic patterns learned by attention mechanisms. Exploring the application of GraphZero to other complex games like Go or Shogi will also test its scalability and adaptability.

Beyond game-playing AI, the principles behind GraphZero have broader implications. Many real-world decision-making problems, from traffic management to financial analysis, involve understanding complex relationships in structured data. The ability of GATs to learn and adapt to these relationships suggests potential applications in a wide range of domains.

In conclusion, GraphZero represents a step towards more flexible, interpretable, and strategically-aware AI systems. While much work remains to fully realize its potential, this research demonstrates the promise of attention-based architectures in pushing the boundaries of what is possible in artificial intelligence, and it encourages continued exploration into their capabilities and applications.

References

[1] 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," *arXiv preprint arXiv:1712.01815*, 2017.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations (ICLR)*, 2018.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the Game of Go Without Human Knowledge," *Nature*, vol. 550, pp. 354–359, Oct. 2017.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

[5] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[6] C. Clark and A. Storkey, "Training Deep Convolutional Neural Networks to Play Go," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[7] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," *arXiv preprint arXiv:1505.00853v2*, 2015.

[8] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385*, 2015.