Assessing Deep Reinforcement Learning Architectures for Shogi Game Engines

1. Introduction

Deep Reinforcement Learning (DRL) has demonstrated remarkable success in mastering complex strategic board games, with Shogi (Japanese chess) presenting unique challenges due to its large board, piece drops, and promotions.[1] This report assesses three distinct DRL architectural proposals for their suitability in a Shogi DRL experiment. The evaluation considers critical factors such as state and action representation, learning efficiency, generalization, and architectural appropriateness for Shogi's intricate dynamics. The objective is to identify the most promising approach by analyzing how each proposal addresses these factors, the implications of their differences, and their overall suitability based on a devised assessment schema.

2. Critical Factors for Shogi DRL Experiments

Several critical factors must be considered when designing a DRL agent for Shogi. These factors, derived from general DRL challenges and Shogi's specific characteristics, will form the basis for evaluating the proposed architectures.

- **A. State Representation Efficacy:**
  - The DRL agent requires a comprehensive and efficient representation of the Shogi board state. This includes the positions of all pieces for both players, their promotion status, the number and type of pieces in hand for each player, relevant game history (e.g., for repetition detection), and other game-specific states like whose turn it is and move counts.[1] Shogi's complexity, particularly the "drop" rule where captured pieces can be reintroduced, significantly expands the state space compared to chess.[2] The representation must be suitable for the chosen neural network paradigm (e.g., grid-like for CNNs, sequential for Transformers).
- **B. Action Space & Policy Output Effectiveness:**
  - Shogi has a vast action space. Actions include moving pieces on the board (with optional promotion in the promotion zone) and dropping captured pieces onto almost any empty square.[4] The policy output of the DRL agent must be able to represent this diverse set of actions. For instance, AlphaZero's Shogi policy head uses a 9×9×139 tensor to represent possible moves.[1] The learning algorithm must plausibly map states to this complex policy output, and illegal moves must be handled, typically via action masking guided by game rules during Monte Carlo Tree Search (MCTS) or directly in the policy network.[3]
- **C. Learning Efficiency and Scalability:**
  - DRL algorithms can be sample inefficient, requiring millions or even billions of

interactions or gradient steps to train effective policies.[7] For complex games like Shogi, this translates to a need for efficient learning from self-play games. Scalability in terms of computational resources (CPU, GPU, TPU) for both training and inference is crucial.[8] The architecture should support efficient training and ideally converge to strong performance relatively quickly. AlphaZero, for example, achieved superhuman Shogi play in under 2 hours of training on specialized hardware.[1]

- **D. Generalization and Robustness:**
  - The DRL agent should generalize well, meaning it performs effectively against a wide variety of opponent strategies and in positions not explicitly seen during training.[7] Robustness implies that the agent's performance does not degrade significantly due to minor changes in the game state or opponent tactics. Learning *tabula rasa* through self-play, as AlphaZero does, is a powerful mechanism for fostering generalization.[1]
- **E. Architectural Suitability for Shogi's Strategic Depth:**
  - The neural network architecture must be capable of recognizing complex patterns and understanding the deep strategic nuances of Shogi. This includes local tactical motifs (forks, pins, sacrifices [11]) as well as long-range strategic considerations, king safety, and the profound impact of piece drops which can alter the game's complexion instantly.[2] The architecture's ability to capture both local features and global board context is paramount.
- **F. Implementation Complexity & Maturity:**
  - The ease of implementing the architecture for Shogi and the maturity of the approach in board game AI are practical considerations. Well-established architectures might offer more community support and pre-existing knowledge, reducing development risk and time.

These factors are interconnected. For instance, a highly detailed state representation might improve strategic understanding but increase computational load and reduce learning efficiency if not managed well. The subsequent proposals will be evaluated against these criteria.

3. Proposal 1: CNN-ResNet Architecture (AlphaZero-like)

This proposal is based on the successful AlphaZero architecture, which employs a deep Convolutional Neural Network (CNN) with residual blocks (ResNet) combined with Monte Carlo Tree Search (MCTS) and self-play reinforcement learning.[1]

- **3.1. Detailed Architectural Design:**
  - **Overall Framework:** The system learns *tabula rasa*, with the neural network $f_\theta(s)=(p,v)$ taking the board state s as input and outputting a policy vector p (move probabilities) and a scalar value v (expected outcome).[3] MCTS uses these outputs to guide its search and generate improved policy targets for training the

network.

- **Neural Network Structure:**
  - **Input:** The Shogi board state is represented as a stack of N×N planes (for Shogi, N=9).
  - **Body:** A "body" consisting of an initial convolutional layer followed by a stack of residual blocks. AlphaZero used 19 or 39 residual blocks.[13] Each residual block typically contains two convolutional layers with batch normalization and ReLU activations.[13]
  - **Heads:** Two separate "heads" follow the body:
    - **Policy Head:** Outputs a probability distribution over all possible actions. For Shogi, this is a 9×9×139 set of planes.[1]
    - **Value Head:** Outputs a single scalar value estimating the expected game outcome from the current player's perspective (e.g., win/loss/draw mapped to +1/-1/0).[3]
- **Training:** The network parameters θ are updated to minimize the error between the predicted value v and the actual game outcome z, and to maximize the similarity between the policy vector p and the MCTS search probabilities π, often with an L2 regularization term.[3] l=(z−v)2−πTlogp+c||θ||2.

- **3.2. Analysis of State and Action Representation for Shogi:**
  - **State Input (AlphaZero for Shogi based on [1]):**
    - The input is an N×N×(MT+L) image stack. For Shogi (N=9), T is the number of past board positions included (history). AlphaZero for chess used T=8. If T=1 for Shogi:
      - **Piece Presence (Current Player):** 14 planes (7 unpromoted piece types: P, L, N, S, G, B, R + 7 promoted piece types: +P, +L, +N, +S, +R, +B; King and Gold do not promote but are included in the base 7). Each plane is 9×9, binary indicating presence.
      - **Piece Presence (Opponent):** 14 similar planes for the opponent.
      - **Pieces in Hand (Current Player):** 7 planes, one for each droppable piece type (P, L, N, S, G, B, R). The encoding can vary, e.g., a plane filled with 1 if ≥1 piece of that type is in hand, or planes indicating counts (e.g., 1 if count=1, 1 if count=2, etc.). [1] suggests "planes indicating the number of captured prisoners of each type". A common approach is to use 2 planes per piece type in hand: one if count = 1, one if count ≥2. This would be 7×2=14 planes.
      - **Pieces in Hand (Opponent):** 14 similar planes for the opponent.
      - **Game State Planes (L planes):**
        - Player's color (1 plane).
        - Total move count (1 plane, possibly normalized).
        - Repetition count for the current position (e.g., 2 planes: 1 if repeated once, 1 if repeated twice for draw by Sennichite considerations).[4] AlphaZero used 2 planes for chess for 3-fold

repetition.[3]
        - No-progress move count (1 plane, for impasse rules, though less common in pro games).
    - A simplified interpretation focusing on [1]'s "prisoners" and common AlphaZero-like Shogi implementations (e.g., dlshogi [17]) might use:
        - Own pieces on board (14 types, including promoted): 14 planes.
        - Opponent pieces on board (14 types): 14 planes.
        - Own pieces in hand (7 types, count ≥1): 7 planes.
        - Own pieces in hand (7 types, count ≥2): 7 planes. (Some implementations use more planes for higher counts, or planes filled with the count).
        - Opponent pieces in hand (7 types, count ≥1): 7 planes.
        - Opponent pieces in hand (7 types, count ≥2): 7 planes.
        - Constant plane indicating player to move: 1 plane.
        - Repetition count planes (e.g., previous state was same, 2-states ago was same): 2 planes.
        - This totals 14+14+7+7+7+7+1+2=59 planes for T=1. If history T=2 is used for piece positions and hand counts, this would be (14+14+7+7+7+7)×2+1+2=112+3=115 planes. The original AlphaZero paper mentions N×N×(MT+L) inputs; for Shogi, this was 9×9×(planes for T steps)+L constant planes.[1] The exact number can vary, but a typical AlphaZero-like Shogi input often uses around 100-150 planes depending on history length and hand piece encoding. For example, one detailed list for chess (adaptable to Shogi) includes 34 base features, some of which are repeated for history.[15]
- **Action Output (Policy Head for Shogi [1]):**
    - A 9×9×139 stack of planes, representing a probability distribution over 9×9×139=11,259 possible moves.
    - **Move Categories in the 139 planes [1]:**
        - **Planes 1-64: Non-promoting "Queen-like" moves.** These cover sliding piece moves (Rook, Bishop, Lance) and single-step moves (King, Gold, Silver, Pawn) in 8 directions, up to 8 squares away. Each of the 64 planes corresponds to a specific direction and distance (e.g., plane for "North, 3 squares").
        - **Planes 65-66: Non-promoting Knight moves.** Two planes representing the two possible Knight moves from any square.
        - **Planes 67-130: Promoting "Queen-like" moves.** An additional 64 planes, identical in structure to planes 1-64, but representing moves that also result in promotion. This applies to pieces like Pawns, Lances, Silvers, Rooks, Bishops when they enter, move within, or leave the promotion zone and choose to promote.
        - **Planes 131-132: Promoting Knight moves.** An additional 2 planes for

        Knight moves that result in promotion.
        - **Planes 133-139: Drops.** Seven planes, one for each of the 7 piece types that can be dropped (Pawn, Lance, Knight, Silver, Gold, Bishop, Rook). For a given drop plane (e.g., "drop Pawn"), a probability at (x,y) on this 9×9 plane indicates the likelihood of dropping a Pawn at square (x,y).
    - Illegal moves (e.g., dropping a pawn on a file already containing an unpromoted friendly pawn, or moves that leave the king in check) are assigned zero probability by the MCTS using perfect knowledge of the game rules.[3] The network learns to assign low probabilities to obviously bad or illegal moves over time.
- **3.3. Addressing Critical Factors:**
  - **State Representation:** The multi-plane input is comprehensive, capturing piece locations, types, promotions, pieces in hand, and limited history.[1] CNNs are inherently well-suited for processing grid-like board data, capturing local patterns of pieces.[2] Residual connections are crucial for training very deep networks, which allows for the learning of hierarchical features, progressively combining local information into more global understanding.[13]
  - **Action Representation:** The 9×9×139 policy output explicitly covers Shogi's diverse move types including different piece movements, drops, and promotions.[1] The MCTS component effectively handles the legality of moves, ensuring the agent only considers valid actions.[3]
  - **Learning Efficiency:** AlphaZero demonstrated remarkable sample efficiency and rapid learning in Shogi, surpassing the world-champion program Elmo in under two hours of training.[1] This efficiency stems from the powerful combination of deep ResNets learning from self-play data generated and refined by MCTS.[10]
  - **Generalization:** The AlphaZero algorithm demonstrated strong performance across Chess, Shogi, and Go, indicating that the general-purpose reinforcement learning algorithm can achieve superhuman performance across multiple challenging domains by learning from self-play without game-specific human data.[1]
  - **Architectural Suitability:** Deep ResNets can learn highly complex, non-linear functions. The convolutional nature captures local spatial configurations of pieces, and the depth allows for an increasingly larger receptive field, enabling the network to eventually consider more global aspects of the board state.[13] The success of AlphaZero implies this architecture is sufficiently powerful for Shogi's strategic depth.
- **3.4. Implications:**
  - **Advantages:**
    - **Proven Superhuman Performance:** This architecture has a track record of achieving superhuman performance in Shogi, defeating existing champion programs.[1]

        - **Tabula Rasa Learning:** The ability to learn from self-play without relying on human game databases or handcrafted heuristics (beyond the game rules themselves) is a significant advantage, allowing the discovery of novel strategies.[1]
        - **Strong Baseline:** It is a well-understood and widely replicated (in principle) architecture, providing a solid and reliable foundation for DRL experiments in Shogi.
    - **Disadvantages:**
        - **Computational Cost:** Training AlphaZero-like models requires substantial computational resources, both for generating self-play games via MCTS and for training the deep neural networks.[2] AlphaZero, for instance, utilized 5,000 first-generation TPUs for game generation and 64 second-generation TPUs for network training.[9]
        - **CNN Receptive Field Limitations:** While deep CNNs can achieve a global receptive field eventually, the explicit modeling of very long-range dependencies (e.g., the influence of a piece drop on a distant part of the board, or long-range tactical sequences) might be less direct or efficient compared to architectures specifically designed for global context, such as Transformers.[19] The hierarchical nature of feature learning in CNNs means global understanding emerges in deeper layers.
        - **MCTS Bottleneck:** The MCTS search, while guided by the neural network, still requires significant computation per move during inference. Although AlphaZero searched far fewer positions per second than traditional engines like Stockfish or Elmo, it compensated by focusing more effectively on promising variations.[9] However, for applications requiring very fast decision-making, this could be a constraint.

The success of AlphaZero's relatively generic CNN-ResNet architecture across diverse board games like Chess, Shogi, and Go [3], despite their significant rule differences and the varying importance of local versus global patterns, is quite revealing. Shogi, with its drop rule, presents a unique challenge where the global board state can be rapidly altered. The fact that the same fundamental network structure (CNN body, ResNet blocks, policy/value heads, adapted for game-specific inputs/outputs) performs at a superhuman level in all three suggests that these networks are highly effective at learning relevant spatial hierarchies and features from the board state. The MCTS component plays a crucial role in complementing the neural network, performing deep lookahead and compensating for any imperfections in the network's learned policy or value estimations. This implies the network doesn't need to achieve perfect, all-encompassing knowledge of every game nuance solely through its architecture; it needs to be sufficiently proficient to guide the MCTS effectively towards strong lines of play.However, a notable characteristic of deep CNNs, particularly in complex DRL agents like AlphaZero, is their "black box" nature.[20] While immensely powerful, the features learned automatically by the network are not immediately interpretable in terms of human Shogi concepts (e.g., specific castle formations, piece development principles, tactical motifs). This lack of transparency can make it challenging to diagnose weaknesses if the agent exhibits unexpected or suboptimal play, or to understand precisely which strategic elements it has mastered. While performance is often the primary metric in game-playing AI, this opacity is a general challenge for complex DRL systems, potentially hindering targeted improvements or building deeper trust in scenarios where understanding the decision-making process is important. Emerging fields like Explainable AI (XAI) aim to address these issues but often introduce additional complexity.[20]

4. Proposal 2: Enhanced CNN-ResNet with Squeeze-and-Excite (SE) Blocks

This proposal aims to enhance the CNN-ResNet architecture of Proposal 1 by incorporating Squeeze-and-Excitation (SE) blocks. SE blocks are designed to improve the representational power of a network by adaptively recalibrating channel-wise feature responses, effectively allowing the network to pay more attention to informative feature channels.[22]

- **4.1. Detailed Architectural Design:**
    - **Core Idea:** Augment a strong CNN-ResNet backbone, similar to that in Proposal 1, with SE blocks integrated into its residual blocks.
    - **SE Block Mechanism [22]:**
        - An SE block operates on the feature maps output by a standard convolutional block.
        - **Squeeze Operation:** Global Average Pooling is applied across the spatial dimensions (H×W) of each feature map. If the input to the SE block has C channels, this operation produces a channel descriptor vector of size 1×1×C. This vector embeds global spatial information for each channel.[22]
        - **Excitation Operation:** This channel descriptor is then passed through a small neural network, typically consisting of two fully connected (FC) layers, to learn channel-wise dependencies:
            1. A first FC layer reduces the number of channels from C to C/r, where r is a hyperparameter known as the reduction ratio (e.g., r=16 is a common default [22]). This is followed by a ReLU activation function.
            2. A second FC layer then projects the dimensionality back up from C/r to C channels. This is followed by a sigmoid activation function, which outputs a set of channel-wise attention weights (values between 0 and 1).
        - **Scale (Recalibration):** The original input feature maps (with C channels) are then multiplied channel-wise by these learned attention weights. This scaling operation selectively emphasizes important feature channels and diminishes the influence of less relevant ones.[22]
    - **Integration:** SE blocks are designed as "drop-in" modules. They can be inserted into existing CNN architectures, often after the main convolutional operations within a residual block and before the final element-wise addition of the shortcut connection.[22]
    - **Rest of Architecture:** The overall DRL framework, including MCTS integration, self-play for data generation, loss functions for policy and value heads, and the

    specific input/output representations for Shogi, would remain largely the same as described in Proposal 1.

- **4.2. Analysis of State and Action Representation:**
    - **State Input:** The state input representation for Shogi would be identical to that of Proposal 1 (e.g., the AlphaZero-style multi-plane representation [1]). SE blocks do not alter the input to the network; rather, they operate on the intermediate feature maps generated by the convolutional layers within the network's body.
    - **Action Output:** The policy output structure (e.g., the 9×9×139 tensor for Shogi [1]) would also remain the same. The SE blocks aim to improve the quality of the learned features that are ultimately fed into the policy and value heads, potentially leading to more accurate predictions.
    - **Impact of SE Blocks on Representation:** By adaptively weighting feature channels, SE blocks can enable the network to learn more discriminative and context-aware features from the Shogi input planes. For Shogi, this could translate to a better ability to identify critical piece configurations, subtle threats, king safety vulnerabilities, or the strategic importance of certain pieces in hand, based on which input-derived feature channels are deemed most relevant in a given board state.
- **4.3. Addressing Critical Factors:**
    - **State Representation:** SE blocks enhance the network's capacity to model interdependencies between the channels of its convolutional features.[23] This allows the network to learn which features are more informative in the current context. In the early layers, SE blocks can bolster the quality of shared, lower-level representations. In deeper layers, they can become increasingly specialized, responding to different inputs in a highly class-specific (or in this case, game-state-specific) manner.[24] This could lead to a richer and more nuanced internal representation of the Shogi state.
    - **Action Representation:** With improved quality of features feeding into the policy and value heads, the network might produce more accurate policy probabilities and more reliable value estimates, indirectly enhancing the effectiveness of the action representation.
    - **Learning Efficiency:** SE blocks have demonstrated the ability to provide significant performance improvements in various computer vision tasks with only a marginal increase in computational cost and parameters.[22] For instance, adding SE blocks to ResNet-50 resulted in a ~1% improvement in top-1 error on ImageNet with less than a 1% increase in MFLOPs for a reduction ratio of 16.[22] This suggests that an SE-enhanced Shogi agent might achieve faster convergence to a target ELO or reach a higher peak ELO for a similar amount of training.
    - **Generalization:** By learning to focus on more relevant features and suppress noise or less useful information, networks incorporating SE blocks may exhibit improved generalization capabilities across different datasets and tasks.[23]
    - **Architectural Suitability:** The channel-wise attention mechanism introduced by

SE blocks allows the network to dynamically adjust the importance of different features. This is particularly relevant for Shogi, where the significance of certain pieces (e.g., a rook) or board regions can change dramatically depending on the game phase (opening, middlegame, endgame) or specific tactical situations (e.g., an impending attack).

- **4.4. Implications:**
    - **Advantages:**
        - **Performance Boost with Low Overhead:** SE blocks are proven to improve the accuracy of CNNs on a variety of tasks with minimal additional computational complexity and parameters, making them an efficient upgrade.[22]
        - **Enhanced Feature Learning:** They explicitly model interdependencies between channels, leading to the learning of more discriminative and robust feature representations.[24]
        - **Ease of Integration:** SE blocks can be readily incorporated into existing state-of-the-art CNN architectures like ResNets without requiring major structural modifications.[22]
    - **Disadvantages:**
        - **Still Fundamentally CNN-Based:** While enhancing channel features, this proposal inherits the fundamental characteristics and potential limitations of the underlying CNN-ResNet architecture, such as its inductive biases towards local patterns and the way receptive fields grow with depth.
        - **Additional Hyperparameter:** The reduction ratio 'r' in the excitation stage is an additional hyperparameter that needs to be tuned.[22] Common values for 'r' include 2, 4, 8, 16, and 32, with r=16 often used as a default and showing good performance in image classification tasks.[26] The optimal value for Shogi might require empirical investigation.
        - **Limited Scope of Attention:** SE blocks provide channel-wise attention. They do not fundamentally alter how spatial information is processed across different locations in the same way that spatial attention mechanisms or Transformer self-attention layers do. Their global understanding is mediated through the global average pooling per channel.

The "squeeze" operation, typically global average pooling, within an SE block provides a condensed summary of the information contained in each feature channel across the entire spatial extent of the feature map.[22] This global descriptor for each channel is then used by the "excitation" stage to compute channel-specific weights. This mechanism allows the network to dynamically recalibrate the importance of each channel based on a global understanding of its content. For a game like Shogi, this implies that the network could learn to, for instance, increase the weight of channels associated with "opponent's rook activity" if, globally, these rooks appear to be coordinating for a decisive attack, even if local features around individual rooks might be ambiguous in isolation. This ability to modulate local feature responses based on a global, per-channel context is a key strength.The reduction ratio 'r' in the SE block's excitation module (the two FC layers) creates an information bottleneck.[22] This bottleneck is designed to balance model capacity—the ability to learn complex interdependencies between channels—against computational cost and the risk of overfitting. The first FC layer compresses C channels to C/r, and the second expands it back to C. A well-chosen 'r' is critical. If 'r' is too large (leading to excessive compression), the network might lose subtle but important information about channel relationships, particularly if the input channels themselves are sparse or carry nuanced signals. This could be relevant in Shogi where input planes representing pieces in hand are often sparse, or where the presence of a single, specific piece in a certain configuration is critical. Conversely, if 'r' is too small (minimal compression), the SE block will have more parameters, increasing both computational load and the potential for overfitting to the specific channel statistics of the training data. Some research has even explored variants like "Effective Squeeze-and-Excitation" (eSE) that use only one FC layer in the excitation stage to avoid this dimensionality reduction, though this typically increases the parameter count compared to standard SE blocks with r>1.[27] The choice of 'r' for Shogi would thus require careful consideration and possibly empirical tuning to ensure that vital information, such as the presence of a critical piece in hand or a subtle positional advantage indicated by a few active features, is not lost during this compression phase.

5. Proposal 3: Transformer-Based Architecture

This proposal explores the use of Transformer networks, which have revolutionized Natural Language Processing (NLP) and are increasingly applied to computer vision and other domains, for playing Shogi. Transformers leverage self-attention mechanisms to capture global dependencies within sequential data.[28]

- **5.1. Detailed Architectural Design:**
  - **Core Idea:** Employ a Transformer architecture to process the Shogi board state, represented as a sequence of tokens, allowing the model to weigh the importance of different board elements (squares, pieces) in relation to each other.
  - **Input Representation (Tokenization):**
    - The 9×9 Shogi board must be transformed into a sequence of tokens. A common approach for board games is to treat each square as a token, resulting in a sequence of 81 tokens for Shogi.[30]
    - Each token would be represented by an embedding vector. This embedding could encode:
      - The piece on the square (type, color, promotion status).
      - Positional information (e.g., via learned positional embeddings or more sophisticated 2D relative position encodings as used in Chessformer [30]).
      - Historical information (e.g., piece on this square in previous T timesteps, as Chessformer does for T=8 [30]).
    - **Pieces in Hand:** These are crucial in Shogi. They could be handled by:
      - Appending special "global tokens" to the sequence for each piece type in hand for each player (e.g., ,). All square tokens could then attend to these hand-piece tokens.[32]

        - Incorporating hand piece counts into the embedding of each square token or a dedicated global `` token.
        - Chessformer's input encoding primarily focuses on on-board pieces and game metadata per square token.[30] A Shogi adaptation would need a robust way to integrate hand piece information, likely through dedicated tokens or by enriching the square token embeddings with global game state features that include hand counts.
    - **Game Metadata:** Information like player to move, repetition counts, total moves could be embedded into a special `` token, prepended to the sequence, or incorporated into all token embeddings.
- **Network Structure (Typically Encoder-Only):**
    - An encoder-only Transformer architecture (similar to BERT or the encoder part of the original Transformer) is often suitable for tasks that require understanding an input sequence to make a prediction, like policy/value estimation from a board state.[30]
    - This consists of a stack of identical Transformer blocks. Each block typically contains:
        - **Multi-Head Self-Attention (MHSA):** This mechanism allows each token in the input sequence to interact with and weigh the importance of all other tokens in the sequence. It computes query, key, and value vectors for each token and then calculates attention scores to produce a contextually enriched representation for each token. Multiple "heads" allow the model to focus on different aspects of the relationships simultaneously.[28]
        - **Feed-Forward Network (FFN):** A position-wise fully connected feed-forward network applied independently to each token's representation after the self-attention step.[28]
        - Residual connections and layer normalization are used around each sub-layer to stabilize training and enable deeper networks.
- **Policy and Value Heads:**
    - The sequence of output token representations from the final Transformer encoder layer is then used to predict the policy and value.
    - **Value Head:** Often, the representation of a special `` token (if used) or an aggregation (e.g., mean pooling) of all output token representations is passed through an MLP to predict the scalar state value.[30]
    - **Policy Head:** This is more complex for Shogi's action space. Options include:
        - A simple MLP over the `` token's representation to output probabilities for a flattened, very large action space (similar to AlphaZero's output, but potentially less structured if not carefully designed).
        - A more structured approach, as seen in Chessformer [30], which uses

an attention-based policy head. For chess, this involves predicting a start square and an end square. For Shogi, this would need to be extended to handle drops (e.g., predict "drop action", then "piece type to drop", then "target square") and promotions. This might involve predicting a sequence of action components or using multiple output heads.

- **Training:** Transformers can be trained using self-play with MCTS guidance (similar to AlphaZero, as done by Chessformer [30]), or with policy gradient methods like PPO directly [36], or even through supervised learning on existing game data.

- **5.2. Analysis of State and Action Representation:**
  - **State Input:** The token-based sequence representation is a departure from the grid-based planes of CNNs.
    - **Board Squares:** Representing each of the 81 squares as a token, with embeddings encoding piece type, color, promotion status, and potentially historical data for that square, is a viable approach.[30] The effectiveness heavily relies on the richness of these embeddings and the positional encoding scheme.
    - **Pieces in Hand:** As discussed, these could be special global tokens (e.g., 7 piece types × 2 players = 14 hand-piece tokens, perhaps with counts embedded or multiple tokens for multiple pieces) or their counts embedded into other tokens. This allows the self-attention mechanism to directly model the influence of available drops on any board square.
    - **History & Game Metadata:** Can be incorporated into the token embeddings themselves (as in Chessformer, which encodes 8 steps of history per square token [30]) or via dedicated global state tokens.
  - **Action Output:** Mapping the Transformer's sequence output to Shogi's complex action space (9×9×139 possibilities or a structured equivalent) is a key challenge.
    - The Chessformer approach of an attention-based policy head, predicting start and end squares, is promising.[30] For Shogi, this would need to be augmented for drops (e.g., a "drop" action type, then selecting a piece from hand tokens, then selecting a target square token) and promotions (e.g., an additional output indicating promotion choice).
    - Action masking based on game rules remains essential to filter out illegal moves from the policy distribution.
- **5.3. Addressing Critical Factors:**
  - **State Representation:** The self-attention mechanism allows the model to, in principle, capture complex, long-range dependencies between any two pieces (or squares/hand pieces) on the board, regardless of their distance.[30] This is a significant theoretical advantage for games like Shogi with prevalent global interactions (e.g., drops, long-range attacks). The quality of this representation is highly dependent on the chosen tokenization strategy and, crucially, the

positional encoding method.[30]

  - **Action Representation:** Designing a policy head that effectively and learnably maps the Transformer's output embeddings to Shogi's structured action space is a non-trivial engineering and research problem. The Chessformer's attention-based policy head offers a strong precedent.[30]
  - **Learning Efficiency:** Transformers are generally known to be data-hungry and computationally intensive to train compared to CNNs, especially from scratch.[36] Sample efficiency might be a concern, particularly if not augmented by strong search like MCTS. However, recent work like Chessformer has shown that Transformers can match AlphaZero-level performance with significantly less computation (8x less for Chessformer vs. AlphaZero in chess) when trained effectively on large self-play datasets.[30]
  - **Generalization:** Their capacity to model global context and complex relationships might lead to better generalization, provided they are trained on sufficiently diverse and extensive data.
  - **Architectural Suitability:** Transformers are exceptionally well-suited for tasks requiring understanding of long-range dependencies and contextual relationships. This aligns well with Shogi's strategic depth, where the value of a square or piece is highly context-dependent, and threats or opportunities can arise from anywhere on the board, including from pieces in hand.[30]
- **5.4. Implications:**
  - **Advantages:**
    - **Superior Global Context Modeling:** The self-attention mechanism is inherently designed to model relationships between all elements in a sequence, providing a powerful tool for capturing global board context from the earliest layers.[30] This is highly relevant for Shogi drops and long-range piece play.
    - **Flexibility in Representation:** Transformers are less constrained by the local grid-based inductive biases of CNNs, potentially allowing them to learn more diverse types of patterns and relationships.
    - **State-of-the-Art Potential:** As Transformers are at the forefront of AI research, leveraging them could lead to new breakthroughs in game AI, as demonstrated by Chessformer's strong results.[30]
  - **Disadvantages:**
    - **Computational Cost and Data Requirements:** Training large Transformer models typically requires significant computational resources (GPUs/TPUs) and vast amounts of training data.[36] Chessformer's largest model (CF-240M) was trained on 500 million games using 8 A100 GPUs.[30]
    - **Complexity of Design for Board Games:** Adapting Transformers effectively for board games like Shogi involves non-trivial design choices for tokenization, positional encoding, and the policy head structure. These are active areas of research.[30]

- **Training Stability:** Transformers can sometimes be more challenging to train stably than well-established ResNet architectures, often requiring careful hyperparameter tuning, large batch sizes, and specific initialization or normalization schemes.[38]
- **Loss of CNN Inductive Bias:** CNNs possess useful inductive biases for image-like data, such as locality (pixels nearby are related) and some degree of translation invariance, which helps them learn efficiently from visual patterns. Standard Transformers lack these biases, meaning they might require more data to learn these fundamental spatial patterns, or these biases need to be explicitly reintroduced (e.g., through convolutional stems at the input, or by designing attention mechanisms that incorporate locality).

A critical aspect of applying Transformers to grid-based games like Shogi is overcoming the permutation-invariance of the self-attention mechanism. Self-attention, by default, treats its input as an unordered set of tokens; it does not inherently understand that token i is "to the left of" token j, or that token k is "two squares diagonally away" from token l. This spatial understanding is fundamental to Shogi. Therefore, the choice and design of the positional encoding scheme are paramount.[30] Simple 1D positional encodings borrowed from NLP are unlikely to be optimal for a 2D board. Research in applying Transformers to chess, such as the Chessformer project, has underscored the necessity of specialized 2D relative positional encodings that can explicitly model relationships based on horizontal and vertical displacements between squares.[30] For Shogi, an effective positional encoding would need to capture not just adjacency but also the specific movement patterns of Shogi pieces (e.g., the Knight's jump, the Lance's forward-only movement, the Bishop's diagonal paths). The success of a Shogi Transformer would be heavily contingent on developing or adapting such an expressive positional encoding.Furthermore, one of the most compelling potential advantages of a Transformer architecture for Shogi lies in its ability to directly model the global influence of pieces in hand. If pieces in hand are represented as distinct global tokens within the input sequence (e.g., ``), the self-attention mechanism allows every board square token to directly attend to these hand-piece tokens.[35] This means a token representing a specific empty square on the board can, in a single attention step, assess the relevance of, for example, a Bishop in the opponent's hand to that square's strategic value or tactical vulnerability. The attention weights could learn to quantify how threatening or opportune a particular piece in hand is with respect to any given square. This provides a more direct and potentially more powerful mechanism for understanding the far-reaching impact of drop tactics compared to CNNs, where such global influence is learned more implicitly through the hierarchical stacking of local receptive fields over many layers. This direct modeling of the interplay between on-board positions and available drops could lead to a more profound strategic understanding.

6. Devised Assessment Schema for Shogi DRL Proposals

To facilitate a structured and objective comparison of the three DRL proposals for Shogi, the following assessment schema is devised. It evaluates each proposal against the critical factors identified previously, using qualitative ratings supported by evidence from the provided research.

- 6.1. Introduction to the Schema:
  This schema aims to provide a rigorous framework for assessing the suitability of each DRL architectural proposal for a Shogi experiment. It breaks down the critical factors into measurable or assessable metrics, allowing for a comparative analysis.
- **6.2. Assessment Criteria and Metrics:**
  - **A. State Representation Efficacy:**
    - A1. Completeness for Shogi Rules (board, pieces, promotions, hand, history, repetitions): (Rating: Fully, Mostly, Partially, Lacking)
    - A2. Efficiency of Encoding (dimensionality, potential for sparsity, computational overhead of processing): (Rating: High Efficiency, Medium Efficiency, Low Efficiency)
    - A3. Suitability for Architectural Paradigm (e.g., grid for CNNs, sequence for Transformers): (Rating: Excellent Fit, Good Fit, Fair Fit, Poor Fit)
  - **B. Action Space & Policy Output Effectiveness:**
    - B1. Coverage of Shogi Moves (board moves, promotions, drops): (Rating: Full, Near Full, Partial)
    - B2. Plausibility of Learning (complexity of mapping internal representation to policy output): (Rating: High Plausibility, Medium Plausibility, Low Plausibility)
    - B3. Integration with Action Masking (for illegal moves): (Rating: Seamless, Workable, Difficult)
  - **C. Learning Efficiency and Scalability:**
    - C1. Anticipated Sample Efficiency (compared to a baseline like AlphaZero): (Rating: Higher, Similar, Lower)
    - C2. Anticipated Training Convergence Speed: (Rating: Faster, Similar, Slower)
    - C3. Computational Resource Demand (Training & Inference): (Rating: Very High, High, Medium, Low)
    - C4. Scalability (improvement with more model parameters/data): (Rating: Good, Moderate, Limited)
  - **D. Generalization and Robustness:**
    - D1. Potential for Generalization (to unseen states/strategies): (Rating: High, Medium, Low)
    - D2. Robustness to Diverse Strategies/Noisy Inputs (theoretical): (Rating: High, Medium, Low)
  - **E. Architectural Suitability for Shogi's Strategic Depth:**
    - E1. Capacity for Complex Pattern Recognition (local tactics & global strategy): (Rating: High, Medium, Low)
    - E2. Handling of Long-Range Dependencies (especially impact of drops): (Rating: Excellent, Good, Fair, Poor)
    - E3. Potential for Deep Strategic Understanding (beyond pattern matching):

(Rating: High, Medium, Low)

- **F. Implementation Complexity & Maturity:**
  - F1. Ease of Implementation for Shogi (adapting existing frameworks): (Rating: Relatively Easier, Moderate, Harder)
  - F2. Maturity of Approach in Board Game AI: (Rating: Mature, Developing, Experimental)

It is important to recognize that these criteria are not always independent. For example, a highly comprehensive state representation (criterion A1) might lead to increased input dimensionality, which could negatively impact computational efficiency (criterion A2) and place greater demands on learning efficiency (criterion C1). A very expressive policy output covering all Shogi actions (criterion B1) might be more challenging for the network to learn effectively (criterion B2). The assessment must therefore consider these trade-offs rather than simply summing scores. The subsequent comparative analysis will delve into these interdependencies.Furthermore, the "Maturity of Approach" (criterion F2) is a significant pragmatic consideration that can heavily influence the risk profile and development timeline of a research and development project. An architecture that is cutting-edge but less mature in the context of board games might offer a higher potential for breakthrough performance but also carries a greater risk of encountering unforeseen engineering challenges or requiring extensive bespoke development. Conversely, a more mature architecture provides a more predictable path. AlphaZero-style CNN-ResNets are relatively mature and well-understood for board games [1], with many principles replicated in open-source projects. Squeeze-and-Excite networks are also standard components in deep learning, and their integration is generally straightforward.[22] Transformers, while dominant in other AI fields, are still more experimental for board games, although their application is rapidly advancing with projects like Chessformer demonstrating significant promise.[30] Key aspects like tokenization and policy head design for the unique characteristics of Shogi would require careful adaptation and possibly novel research if a Transformer approach is chosen. This balance between risk and potential reward is a critical element in selecting a suitable architecture.

7. Comparative Analysis and Suitability Assessment

This section provides a side-by-side comparison of the three proposals using the assessment schema, discusses trade-offs, highlights Shogi-specific challenges, and offers an overall suitability assessment.

- **7.1. Comparative Table of Proposals**

Criterion	**Metric**	**Proposal 1: CNN-ResNet (AlphaZero-like)**	**Proposal 2: SE-CNN-ResNet**	**Proposal 3: Transformer-Based**
A. State Representation Efficacy	A1. Completeness for Shogi	Fully (with appropriate plane design [1])	Fully (same input as P1)	Fully (with comprehensive tokenization for board & hand [30])
	A2. Encoding Efficiency	Medium (many planes, but CNNs	Medium (slight overhead from SE	Potentially Lower (token

		process grids efficiently)	ops [22])	embeddings can be large; sequence processing)
	A3. Architectural Fit	Excellent (CNNs for grid data [18])	Excellent (SE enhances CNNs [24])	Good (if tokenization & positional encoding are well-designed for 2D grid + global elements [30])
B. Action Space & Policy Output	B1. Shogi Move Coverage	Full (e.g., 9x9x139 output [1])	Full (same output as P1)	Full (requires careful policy head design, e.g., adapting Chessformer [30])
	B2. Learning Plausibility	High (proven by AlphaZero [1])	High (builds on P1)	Medium to High (complex mapping, but Chessformer shows promise [30])
	B3. Action Masking	Seamless (standard in MCTS [3])	Seamless (same as P1)	Workable (masking logits or during structured prediction [6])
C. Learning Efficiency & Scalability	C1. Sample Efficiency	Medium (MCTS helps, but deep nets need data [7])	Medium to Potentially Higher (SE may improve feature learning [22])	Potentially Lower initially (Transformers are data-hungry [36]), but Chessformer achieved good results with large datasets [30]
	C2. Convergence Speed	Medium (AlphaZero Shogi <2hrs on TPUs [1])	Potentially Faster (SE can accelerate learning [22])	Medium to Slower (depends on scale and data [38])
	C3. Resource Demand	Very High (MCTS + deep ResNet [9])	Very High (similar to P1, minor SE overhead)	Very High to Extremely High (large Transformers are demanding [30])

	C4. Scalability	Good (deeper/wider nets, more MCTS)	Good (similar to P1)	Excellent (Transformers scale very well with data/compute [30])
D. Generalization & Robustness	D1. Generalization Potential	High (proven by AlphaZero across games [3])	High to Potentially Higher (better features from SE [24])	High (global context modeling [30])
	D2. Robustness	Medium (CNNs can be susceptible to adversarial inputs, though MCTS adds robustness)	Medium (similar to P1)	Medium to High (attention may offer some robustness)
E. Architectural Suitability for Shogi	E1. Complex Pattern Recognition	High (deep CNNs learn hierarchies [13])	High to Very High (SE enhances feature discrimination [24])	Very High (attention for complex relationships [30])
	E2. Long-Range Dependencies (drops)	Fair to Good (learned implicitly via depth)	Good (SE global squeeze helps channel context)	Excellent (direct modeling via self-attention [34])
	E3. Deep Strategic Understanding	Medium to High (emergent from MCTS + NN)	High (potentially better value/policy functions)	Potentially Very High (if effectively trained)
F. Implementation Complexity & Maturity	F1. Ease of Implementation	Moderate (many existing principles, e.g., dlshogi [17])	Moderate (SE blocks are standard additions)	Harder (Shogi tokenization, positional encoding, policy head are research areas [30])
	F2. Maturity in Board Games	Mature (AlphaZero paradigm [10])	Mature (SE is well-established in vision)	Developing (Transformers for board games are newer, e.g., Chessformer [31])

- **7.2. Discussion of Trade-offs and Synergies:**
  - **CNN-ResNet (P1) vs. SE-CNN-ResNet (P2):** Proposal 2 offers an incremental improvement over Proposal 1. The introduction of SE blocks aims to enhance the feature learning capabilities of the CNN by allowing adaptive recalibration of channel-wise feature responses.[22] This typically yields performance gains with only a slight increase in computational parameters and complexity. The trade-off

is minimal, making SE blocks an attractive addition if the base CNN-ResNet is already being implemented. The core architecture remains the same, leveraging the proven strengths of AlphaZero's design while potentially extracting more nuanced information from the input planes.

  - **CNN-Based (P1 & P2) vs. Transformer-Based (P3):** This represents a more fundamental architectural divergence.
    - **Local vs. Global Context:** CNNs inherently possess an inductive bias towards local patterns due to their convolutional filters. Global context is built up hierarchically through successive layers increasing the receptive field.[19] Transformers, via their self-attention mechanism, can model relationships between any two tokens in the input sequence from the very first layer, providing an immediate global receptive field.[30] For Shogi, which features both local tactical engagements and global strategic considerations (especially the pervasive threat of drops from hand), the Transformer's ability to directly model long-range dependencies is theoretically advantageous.
    - **Data and Computational Needs:** Historically, Transformers have been more data-hungry and computationally expensive to train than CNNs.[36] While this remains a consideration, recent work like Chessformer has demonstrated that Transformer-based models can achieve or surpass AlphaZero-level performance with comparable or even reduced (though still substantial) computational budgets when scaled appropriately and trained on large self-play datasets.[30]
    - **Maturity and Implementation Risk:** CNN-based approaches for board games, particularly following the AlphaZero paradigm, are more mature and well-documented. Adapting them for Shogi has precedents (e.g., dlshogi [17]). Transformers for board games are a more recent development. While results are promising (e.g., Chessformer), designing effective Shogi-specific tokenization, positional encodings, and policy heads requires more research and carries higher implementation risk.
  - **Potential Synergies (Beyond Scope but Relevant):** Hybrid architectures that combine convolutional layers (for efficient local feature extraction) with Transformer layers (for global context modeling) are an active area of research in computer vision and could offer a compelling future direction for board game AI.[39] For example, a CNN "stem" could process the input planes to generate patch embeddings, which are then fed into a Transformer body.
- **7.3. Specific Challenges for Each Proposal in Shogi:**
  - **Proposal 1 (CNN-ResNet):** The primary challenge is ensuring that the deep convolutional network can effectively learn the global implications of Shogi's drop rule and coordinate long-range piece interactions purely through its hierarchical feature extraction. While AlphaZero's success indicates this is possible, it might not be the most direct or efficient way to capture these global aspects.

  Additionally, the efficiency of the MCTS component is critical, as it performs the heavy lifting of lookahead search.
  - **Proposal 2 (SE-CNN-ResNet):** Shares the challenges of Proposal 1, as SE blocks are an enhancement rather than a fundamental architectural shift. A specific challenge is the optimal tuning of the SE block's reduction ratio 'r' for Shogi. An inappropriate 'r' could either fail to capture important channel interdependencies or add unnecessary parameters.[22]
  - **Proposal 3 (Transformer):**
    - **Tokenization:** Devising an optimal tokenization scheme for Shogi is crucial. This includes how to represent pieces on the 81 squares, their promotion status, and critically, the 7 types of pieces in hand for both players (and their counts). The representation must be amenable to the Transformer's sequence processing.
    - **Positional Encoding:** Standard 1D positional encodings from NLP are insufficient for a 2D board. Shogi requires a positional encoding that captures its unique geometry and piece movements (e.g., Knight's jump, Lance's forward-only restriction, diagonal movements). Adapting or developing 2D relative positional encodings like those used in Chessformer is essential.[30]
    - **Policy Head Design:** Mapping the Transformer's output token embeddings to Shogi's complex action space (potentially 9×9×139 distinct actions, including moves, drops, and promotions) in a learnable and efficient manner is a significant hurdle. The Chessformer policy head (predicting start and end squares via attention) provides a template but needs substantial adaptation for Shogi drops and promotions.[30]
    - **Computational Resources and Training Stability:** Managing the high computational demands and ensuring stable training of large Transformer models for Shogi would require careful engineering and hyperparameter optimization.[38]
- 7.4. Overall Suitability Assessment:
  Each proposal offers a viable path for a Shogi DRL experiment, but with different risk-reward profiles and suitability depending on the experiment's primary objectives.
  - **Proposal 1 (CNN-ResNet)** is the most established and lowest-risk approach for achieving strong performance, leveraging the proven AlphaZero framework. Its main drawback is the high computational cost and the indirect way CNNs handle global context.
  - **Proposal 2 (SE-CNN-ResNet)** offers a compelling low-risk, moderate-reward enhancement to Proposal 1. It promises improved feature learning and potentially better performance or efficiency with minimal architectural changes, making it an attractive option for refining a strong baseline.
  - **Proposal 3 (Transformer)** is the highest-risk, highest-potential-reward option. It offers a theoretically superior way to model Shogi's global dynamics, especially

drops. However, it requires significant research and engineering effort to adapt for Shogi's specific state and action representations and to manage its computational demands.

The "best" proposal is contingent on the specific goals of the DRL experiment. If the aim is to replicate or build upon known SOTA performance with high confidence, Proposal 1 or 2 are strong candidates. If the objective is to explore novel architectures with the potential to fundamentally improve how global strategic elements like drops are modeled, Proposal 3 is more aligned, despite its challenges. The iterative nature of research suggests that starting with a robust and well-understood baseline (like Proposal 1 or 2) can provide a solid platform for future explorations into more advanced architectures like Transformers. Success with an SE-enhanced ResNet, for instance, could provide valuable insights and a stronger foundation before tackling the complexities of a full Transformer model tailored for Shogi.Furthermore, the choice of the underlying DRL training algorithm (e.g., AlphaZero's MCTS-guided policy iteration versus a direct actor-critic method like PPO) interacts significantly with the neural architecture. While all three architectures could theoretically be trained with different algorithms, their strengths may align better with specific paradigms. AlphaZero's ResNet is tightly coupled with the policy targets derived from MCTS.[3] A Transformer might also benefit from MCTS (as in Chessformer [30]) or could potentially learn complex policies directly with an algorithm like PPO if the state and action representations are sufficiently rich and the training process is well-tuned. This report primarily assumes an AlphaZero-like framework due to its proven success in board games, but the architectural choice has implications for the entire DRL system.

8. Conclusion and Recommendation

This report has assessed three distinct Deep Reinforcement Learning architectural proposals for their suitability in a Shogi experiment, evaluating them against critical factors such as state/action representation, learning efficiency, generalization, and architectural fit for Shogi's complexities.

- **8.1. Summary of Findings:**
  - **Proposal 1 (CNN-ResNet - AlphaZero-like):** Offers a proven, mature path to superhuman performance in Shogi, learning *tabula rasa*. Its strengths lie in its established framework and the power of deep residual networks combined with MCTS. However, it is computationally very expensive and relies on CNNs' hierarchical, localized feature learning to capture global game dynamics.
  - **Proposal 2 (SE-CNN-ResNet):** Provides an incremental but potentially significant enhancement to Proposal 1 by incorporating Squeeze-and-Excitation blocks. This allows for adaptive channel-wise feature recalibration, potentially improving feature quality and learning efficiency with minimal additional computational overhead or implementation complexity. It retains the strengths and most weaknesses of the base CNN-ResNet.
  - **Proposal 3 (Transformer-Based):** Represents a more cutting-edge approach with the theoretical advantage of directly modeling global dependencies via self-attention, which is highly relevant for Shogi's drop rule and long-range strategy. However, it faces challenges in Shogi-specific tokenization, positional

encoding, policy head design, and typically has higher data and computational requirements, along with greater implementation complexity and less maturity in the board game domain compared to CNNs.

- 8.2. Recommended Proposal:
  For a Shogi DRL experiment aiming for a balance of strong performance, manageable research risk, and the potential for tangible improvement over established baselines, Proposal 2: Enhanced CNN-ResNet with Squeeze-and-Excite (SE) Blocks is recommended as the most suitable.
- 8.3. In-Depth Justification for Recommendation:
  Proposal 2 is recommended because it builds upon the robust and empirically validated foundation of the AlphaZero architecture (Proposal 1) while introducing a well-understood and computationally inexpensive mechanism (SE blocks) to potentially enhance performance.
  - **Leveraging a Proven Baseline:** The AlphaZero framework has demonstrated its capability to master Shogi at a superhuman level.[1] Proposal 2 inherits this strong starting point, reducing the risk associated with entirely novel architectural designs.
  - **Targeted Improvement:** SE blocks specifically address the quality of feature representation by enabling the network to dynamically emphasize more informative feature channels.[22] In Shogi, where the relevance of different pieces or board configurations can shift dramatically, this adaptive recalibration could lead to more nuanced state evaluations and policy decisions. The global information captured by the "squeeze" operation can inform channel importance, potentially improving the network's understanding of how local features contribute to the global board assessment.
  - **Manageable Complexity and Cost:** Integrating SE blocks into a ResNet architecture is a relatively standard procedure in deep learning, with minimal increase in parameters and computational load compared to the base ResNet.[22] This makes it a practical enhancement.
  - **Balanced Risk-Reward:** Compared to Proposal 3 (Transformer), Proposal 2 offers a more predictable development path. While Transformers hold great promise for modeling global interactions in Shogi, their application to this specific game, particularly regarding optimal tokenization for pieces in hand, appropriate 2D positional encodings, and a policy head for drops and promotions, still involves significant research questions and engineering challenges.[30] Proposal 2 allows for focused experimentation on feature enhancement within a largely known framework.
  - **Addressing Shogi's Nuances:** The ability of SE blocks to help the network "focus" on relevant features could be particularly beneficial for Shogi's complex state space, where distinguishing critical signals (e.g., a key piece in hand, a subtle king attack) from less relevant information is vital.

  Proposal 1, while a strong contender, represents the existing state-of-the-art baseline. An experiment would likely aim to improve upon or gain new insights from this baseline.

Proposal 3, the Transformer-based approach, while theoretically powerful for Shogi's global aspects, introduces a higher degree of implementation complexity and research risk, especially in designing Shogi-specific components for tokenization and policy output. The Chessformer results are for chess and its direct adaptation to Shogi's unique drop rule and piece set is non-trivial.[30] Therefore, Proposal 2 strikes an effective balance, offering a pathway for innovation and performance improvement with a manageable increase in complexity over a proven system.

- **8.4. Considerations for Implementation (Proposal 2):**
    - **Base Architecture:** A deep ResNet similar to AlphaZero (e.g., 20 to 40 residual blocks) should be used as the backbone.[13]
    - **Input Representation:** A comprehensive set of input planes for Shogi, similar to AlphaZero's design, is crucial. This should include planes for piece locations (own and opponent, promoted and unpromoted), pieces in hand (for own and opponent, potentially encoding counts like ≥1, ≥2), and game state information (player to move, repetition history).[1] Total planes likely in the range of 100-150 if history is included.
    - **Policy Output:** The $9 \times 9 \times 139$ policy head structure detailed by AlphaZero for Shogi should be adopted, covering normal moves, promotions, and drops.[1]
    - **SE Block Integration:** SE blocks should be integrated into each residual block, typically after the second convolutional layer and before the addition of the shortcut connection.
    - **SE Reduction Ratio (r):** This is a key hyperparameter. While $r=16$ is a common default in vision tasks [26], experimentation with values like $r \in \{8,16,32\}$ would be advisable for Shogi to balance feature compression and model capacity. Given Shogi's potentially sparse input features (e.g., pieces in hand), a smaller r (less compression) might be beneficial.
    - **Training Framework:** An AlphaZero-like self-play reinforcement learning loop with MCTS is recommended. The MCTS will generate policy targets (π) and game outcomes (z) for training the SE-CNN-ResNet.
    - **Action Masking:** Essential for handling illegal moves. MCTS inherently uses game rules to only explore legal actions. The policy network's output will be a distribution over all $9 \times 9 \times 139$ actions, and illegal ones will have their probabilities effectively masked (or set to zero by the search) before action selection.[3]
    - **Hyperparameter Tuning:** Standard DRL hyperparameters (learning rate, batch size, MCTS simulation count, exploration parameters like cpuct) will require careful tuning.[37] For PPO, if used as an alternative or supplementary training algorithm, typical ranges for batch size (discrete: 32-512), learning rate (10−5 to 10−3), and epsilon (0.1-0.3) are starting points.[45]
- **8.5. Potential Future Research Directions:**
    - If Proposal 2 yields significant improvements, further investigation into more sophisticated attention mechanisms could be warranted, potentially leading towards hybrid CNN-Transformer architectures that leverage the strengths of both paradigms for Shogi.

- Exploring different SE block variants (e.g., eSE [27]) or spatial attention mechanisms in conjunction with channel attention.
- Developing XAI techniques to understand what specific Shogi concepts the SE-enhanced network learns more effectively.
- Investigating the optimal history length (T) for Shogi input representations in conjunction with SE blocks.

By systematically building upon a proven architecture with a well-understood enhancement, Proposal 2 offers the most promising and pragmatic path for advancing DRL capabilities in the complex domain of Shogi.

Works cited

1. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm arXiv:1712.01815v1 [cs.AI] 5 Dec 2017, accessed May 27, 2025, https://arxiv.org/pdf/1712.01815
2. Online Match Prediction in Shogi Using Deep Convolutional Neural Networks - SciTePress, accessed May 27, 2025, https://www.scitepress.org/Papers/2024/130181/130181.pdf
3. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, accessed May 27, 2025, https://ar5iv.labs.arxiv.org/html/1712.01815
4. SHOGI Rule, accessed May 27, 2025, http://www.shogi.ricoh/rules/erules.html
5. Shogi - Wikipedia, accessed May 27, 2025, https://en.wikipedia.org/wiki/Shogi
6. arXiv:2504.02662v1 [cs.LG] 3 Apr 2025, accessed May 27, 2025, https://arxiv.org/pdf/2504.02662?
7. Deep reinforcement learning for real-world quadrupedal locomotion: a comprehensive review - OAE Publishing Inc., accessed May 27, 2025, https://www.oaepublish.com/articles/ir.2022.20
8. Reinforcement Learning in Strategy-Based and Atari Games: A Review of Google DeepMind's Innovations - arXiv, accessed May 27, 2025, https://arxiv.org/html/2502.10303
9. AlphaZero - Wikipedia, accessed May 27, 2025, https://en.wikipedia.org/wiki/AlphaZero
10. How did AlphaZero achieve superhuman performance in games like chess and Shōgi within hours, and what does this indicate about the efficiency of its learning process? - EITCA Academy, accessed May 27, 2025, https://eitca.org/artificial-intelligence/eitc-ai-arl-advanced-reinforcement-learning/case-studies/alphazero-mastering-chess-shogi-and-go/examination-review-alphazero-mastering-chess-shogi-and-go/how-did-alphazero-achieve-superhuman-performance-in-games-like-chess-and-shogi-within-hours-and-what-does-this-indicate-about-the-efficiency-of-its-learning-process/
11. Shogi tactics - Wikipedia, accessed May 27, 2025, https://en.wikipedia.org/wiki/Shogi_tactics
12. AlphaZero - Notes on AI, accessed May 27, 2025, https://notesonai.com/alphazero
13. AlphaZero - Chessprogramming wiki, accessed May 27, 2025, https://www.chessprogramming.org/AlphaZero

14. Simple Alpha Zero - Surag Nair, accessed May 27, 2025, https://suragnair.github.io/posts/alphazero.html
15. Learning to Play the Chess Variant Crazyhouse Above World Champion Level With Deep Neural Networks and Human Data, accessed May 27, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC7861260/
16. How to play Shogi(将棋) -Lesson#15- Repetition("Sen-nichi-te") - YouTube, accessed May 27, 2025, https://www.youtube.com/watch?v=7SZpl_a4aC0
17. python-dlshogi/pydlshogi/features.py at master · TadaoYamaoka ..., accessed May 27, 2025, https://github.com/TadaoYamaoka/python-dlshogi/blob/master/pydlshogi/features.py
18. A Convolutional Neural Network Approach to General Game Playing - IOS Press Ebooks, accessed May 27, 2025, https://ebooks.iospress.nl/pdf/doi/10.3233/FAIA230563
19. Limitations of Shallow Networks, accessed May 27, 2025, https://www.cs.cas.cz/~vera/publications/books/C3.pdf
20. A Survey on Explainable Deep Reinforcement Learning - arXiv, accessed May 27, 2025, https://arxiv.org/html/2502.06869v1
21. StateMask: Explaining Deep Reinforcement Learning through State Mask, accessed May 27, 2025, https://proceedings.neurips.cc/paper_files/paper/2023/file/c4bf73386022473a652a18941e9ea6f8-Paper-Conference.pdf
22. Squeeze and Excitation Networks: A Performance Upgrade - viso.ai, accessed May 27, 2025, https://viso.ai/deep-learning/squeeze-and-excite-networks/
23. [1709.01507] Squeeze-and-Excitation Networks - arXiv, accessed May 27, 2025, https://arxiv.org/abs/1709.01507
24. arXiv:1709.01507v2 [cs.CV] 5 Apr 2018, accessed May 27, 2025, https://arxiv.org/pdf/1709.01507v2/1000
25. Squeeze-and-Excitation Block Explained - Papers With Code, accessed May 27, 2025, https://paperswithcode.com/method/squeeze-and-excitation-block
26. Channel Attention and Squeeze-and-Excitation Networks (SENet) | DigitalOcean, accessed May 27, 2025, https://www.digitalocean.com/community/tutorials/channel-attention-squeeze-and-excitation-networks
27. Effective Squeeze-and-Excitation Block Explained | Papers With Code, accessed May 27, 2025, https://paperswithcode.com/method/effective-squeeze-and-excitation-block
28. LLM Basics: Embedding Spaces - Transformer Token Vectors Are Not Points in Space, accessed May 27, 2025, https://www.alignmentforum.org/posts/pHPmMGEMYefk9jLeh/llm-basics-embedding-spaces-transformer-token-vectors-are
29. 9 Transformers – 6.390 - Intro to Machine Learning, accessed May 27, 2025, https://introml.mit.edu/notes/transformers.html
30. (PDF) Mastering Chess with a Transformer Model - ResearchGate, accessed May 27, 2025,

https://www.researchgate.net/publication/384155624_Mastering_Chess_with_a_Transformer_Model
31. Mastering Chess with a Transformer Model - arXiv, accessed May 27, 2025, https://arxiv.org/html/2409.12272v2
32. Explainability of Transformer-Based Reinforcement Learning - DiVA portal, accessed May 27, 2025, https://www.diva-portal.org/smash/get/diva2:1940023/FULLTEXT01.pdf
33. ReaCritic: Large Reasoning Transformer-based DRL Critic-model Scaling For Heterogeneous Networks - ResearchGate, accessed May 27, 2025, https://www.researchgate.net/publication/391856644_ReaCritic_Large_Reasoning_Transformer-based_DRL_Critic-model_Scaling_For_Heterogeneous_Networks
34. INVESTIGATING SELF-ATTENTION: ITS IMPACT ON SAMPLE EFFICIENCY IN DEEP REINFORCEMENT LEARNING - OpenReview, accessed May 27, 2025, https://openreview.net/pdf?id=J5s6EG6ual
35. Understanding and Coding the Self-Attention Mechanism of Large Language Models From Scratch - Sebastian Raschka, accessed May 27, 2025, https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html
36. Transformers as Policies for Variable Action Environments - ResearchGate, accessed May 27, 2025, https://www.researchgate.net/publication/367020022_Transformers_as_Policies_for_Variable_Action_Environments
37. SHA-ZA: Advanced Reinforcement Learning for Othello Mastery Using Proximal Policy Optimization, accessed May 27, 2025, https://www.ijml.org/vol15/IJML-V15N1-1173.pdf
38. TF-DDRL: A Transformer-Enhanced Distributed DRL Technique for Scheduling IoT Applications in Edge and Cloud Computing Environments, accessed May 27, 2025, https://www.computer.org/csdl/journal/sc/2025/02/10836729/23oDvmlYGGY
39. Representation Matters for Mastering Chess: Improved Feature Representation in AlphaZero Outperforms Switching to Transformers - IOS Press Ebooks, accessed May 27, 2025, https://ebooks.iospress.nl/pdf/doi/10.3233/FAIA240763
40. Enhancing Chess Reinforcement Learning with Graph Representation - arXiv, accessed May 27, 2025, https://arxiv.org/html/2410.23753v1
41. CNN-Transformer for Active Cell Balancing with Proximal Policy Optimization and Hyperparameter Tuning | Code Ocean, accessed May 27, 2025, https://codeocean.com/explore/3e5f39f0-88cc-4797-a7fa-920a30c1e9ef
42. A Reinforcement Learning Project using PPO + Transformer - GitHub, accessed May 27, 2025, https://github.com/datvodinh/ppo-transformer
43. AobaZero, accessed May 27, 2025, http://www.yss-aya.com/aobazero/index_e.html
44. Scaling Multi-Frame Transformers for End-to-End Driving - SciTePress, accessed May 27, 2025, https://www.scitepress.org/Papers/2025/131567/131567.pdf
45. PPO/best-practices-ppo.md at master · EmbersArc/PPO · GitHub, accessed May 27, 2025, https://github.com/EmbersArc/PPO/blob/master/best-practices-ppo.md
46. Deep Reinforcement Learning: A Chronological Overview and Methods - MDPI,

accessed May 27, 2025, https://www.mdpi.com/2673-2688/6/3/46