FACULTY OF SCIENCE & TECHNOLOGY

MSc Data Science and Artificial Intelligence
June 2025

Interpretable Chess Engine: Neural Network Architectures and Explainable AI for Transparent Move Prediction

by

Kevin Djabaku Ocansey

Faculty of Science & Technology

Department of Computing and Informatics

Individual Masters Project

Abstract

Modern chess engines achieve superhuman performance but operate as opaque "black boxes," obscuring their decision-making processes within distributed representations. While traditional engines like Deep Blue provided transparent, interpretable reasoning through explicit evaluation functions, contemporary systems like AlphaZero sacrifice interpretability for raw performance. This research addresses the critical need for interpretable neural chess systems that combine predictive capability with transparent reasoning.

We developed and evaluated interpretable chess systems using three CNN architectures (ResNet-50, DenseNet-121, VGG-16) with systematic comparison of 12-plane versus 19-plane board encodings. Models were trained on Lichess game data and analyzed using complementary interpretability methods: SHAP feature attribution, Testing with Concept Activation Vectors (TCAV), and gradient-based saliency analysis. This multi-method approach enables robust validation of architectural reasoning patterns and chess concept learning.

Results demonstrate that CNNs achieve 28.9-34.2% agreement with Stockfish recommendations, with ResNet-50 and 19-plane encoding optimal (34.2% accuracy). This work contributes validated methodology for interpretable AI in strategic domains, demonstrating that transparency and performance can be mutually reinforcing rather than competing objectives. This is increasingly critical as AI systems influence high-stakes decisions.

Dissertation Declaration

I agree that, should the University wish to retain it for reference purposes, a copy of my dissertation may be held by Bournemouth University normally for a period of 3 academic years. I understand that once the retention period has expired my dissertation will be destroyed.

Confidentiality

I confirm that this dissertation does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. Any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history, or sex life has been anonymised unless permission has been granted for its publication from the person to whom it relates.

Copyright

The copyright for this dissertation remains with me.

Requests for Information

I agree that this dissertation may be made available as the result of a request for information under the Freedom of Information Act.

Signed: K.D.O .

Name: Kevin Djabaku Ocansey

Date: 23rd June 2025

Programme: Msc Data Science and Artificial Intelligence

Original Work Declaration

This dissertation and the project that it is based on are my own work, except where stated, in accordance with University regulations.

Signed: K.D.O .

Name: Kevin Djabaku Ocansey

Date: 23rd June 2025

Acknowledgments

I want to thank my supervisor Emili, his constant encouragement and time he put into guiding and making me work more into the details of this project. I have learnt a lot in the couple of months and the more I speak with him the more passionate I get about deep learning. Secondly my classmates for discussing their projects with me and showing me what works for them. After my results in my Artificial Intelligence module, I knew I needed help in how to properly write academic literature. I had a lot of help in organizing and explaining my findings because I explained them to my friends first and they gave me useful feedback as well as the energy and support to continuously improve and refine my project. Edward, Adwoa, Joachim, Yuli, Chinduji, Gerald and Lorena. Lastly my family for their constant support in making sure I have the things I need to focus solely on my education. I am extremely indebted to them.

Claude and ChatGPT were used to assist with literature review summaries, code development in Sections 8-9, and improving technical clarity. All output were reviewed and modified where necessary. The research design, analysis, and conclusions are my own original work inspired from most papers in the reference list.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

Chess has always served as a benchmark for progress in Artificial Intelligence, from rule-based systems like Deep Blue, which defeated Garry Kasparov in 1997 through specialized hardware and explicit evaluation functions (Campbell et al., 2002), to modern neural approaches. Traditional engines relied on handcrafted heuristics and exhaustive search, while contemporary systems like AlphaZero derive strength from deep learning and self-play rather than programmed knowledge. Modern neural engines employ CNNs well-suited for chess's spatial 8×8 structure, with ResNet architectures enabling deep pattern recognition through skip connections (He et al., 2016). These systems learn evaluation functions directly from data rather than relying on programmed heuristics (Wang, 2024).

AlphaZero exemplifies this paradigm shift, learning chess mastery entirely through self-play reinforcement learning. By replacing handcrafted evaluation with neural pattern recognition guided by Monte Carlo Tree Search, it achieves superhuman performance while rendering its decision-making process largely opaque (Blüml et al., 2023).

The success of entirely self-taught systems raises critical questions about what these networks have actually learned. Explainability becomes essential for ensuring that AI systems can provide evidence supporting their decisions, offering insight into underlying reasoning processes. Researchers have developed various explainability techniques, broadly categorized as self-interpretable models (decision trees, logistic regression) and post-hoc explanations (LIME, SHAP, TCAV). These methods vary in scope. While some provide local, instance-level insights, others reveal global patterns and are often tailored to specific architectures and data types. This diversity reflects the field's evolving nature and the absence of universal solutions for explaining complex models (Markus et al., 2021).

Despite growing interest in explainable AI, considerable ambiguity remains around key terms such as explainability, interpretability, transparency, and intelligibility. As Miller (2019) notes, "it is still unclear how XAI methods should be evaluated, how different terms should be used in the debate, or how, strictly, XAI is related to trustworthiness."

1.2 Problem definition

Chess is a particularly challenging domain for explainable AI because its abstract, long-term strategies rarely map onto simple visual or semantic cues. As Nicolson et al. (2024) argue, effective interpretability in chess must capture strategic and conceptual reasoning rather than just spatial focus. However, extracting such explanations from deep neural networks presents structural and conceptual difficulties. For example, while AlphaZero has been shown to internalize ideas such as material balance and king safety, these concepts are encoded in a distributed manner across many

layers, entangled with unrelated activations, and not explicitly labelled (McGrath et al., 2022). This entanglement limits the explanatory power of techniques such as Concept Activation Vectors (Nicolson et al., 2024). The core challenge arises from an architectural shift: whereas traditional chess engines were inherently interpretable, built on transparent evaluation functions, modern neural engines achieve superior performance at the cost of obscuring their reasoning within deep, distributed representations.

1.3 Aims and objectives

1.3.1 Aim

To develop an interpretable chess system built on neural networks, combining move prediction with concept-based explanations to make the model's decision-making process transparent. The project serves as a first step toward mapping how such systems develop and internalise chess concepts as they learn.

1.3.2 Objectives

1. Review and analyse existing state-of-the-art neural network–based chess engines to understand their architectures and capabilities.

2. Train and evaluate multiple deep neural network models using Lichess FEN string datasets, comparing their move prediction accuracy against a benchmark engine.

3. Define and curate a set of human-understandable chess concepts to facilitate probing and interpretability testing.

4. Apply advanced interpretability methods tailored to convolutional neural networks to map model decisions to human-relevant chess concepts and assess their transparency.

5. Compare the outputs of different models and interpretability techniques for the same positions to determine the depth and consistency of the insights provided.

1.3.3 Research Questions

1. How accurately can a neural network–based chess engine predict moves compared to a benchmark engine when trained on Lichess data?

2. Which interpretability techniques (e.g., saliency maps, Concept Activation Vectors, linear probes) best reveal human-understandable chess concepts from the model's internal representations?

3. How consistently do different interpretability methods identify the same concepts for a given chess position?

4. What can be learned about how deep neural networks represent and develop chess concepts during supervised training?

1.3.4 Risk Assessment

Risk ID	Risk Description	Likelihood	Impact	Severity (L×I)	Mitigation Strategy
R1	The internal workings of deep learning models may be hard to interpret, making it unclear how decisions are made	Medium	High	12	Plan to test multiple explanation techniques to see which gives the clearest results
R2	Visual explanation methods might not work well with either FEN strings.	High	Medium	15	Try different approaches and compare results to find which works best
R3	Training large models could take longer than expected or use too much computing power	Low	Medium	9	Use smaller models or pre-trained ones if training time is too long
R4	It may be difficult to match the model's outputs to clear chess concepts	High	High	20	Look for or create labelled examples of common chess ideas to help with mapping
R5	There might not be enough labelled examples for advanced chess ideas like "initiative" or "tempo"	Medium	High	12	Use simpler concepts first, or generate synthetic examples
R6	Explanations from the AI might be easy to misread or over-interpret	Medium	High	12	Check results using more than one explanation method and note any differences
R7	Judging models only by accuracy might miss other important factors	Medium	Medium	9	Also consider how understandable the explanations are to people
R8	Putting all the parts of the project together could take longer than planned	Medium	Medium	9	Build and test parts separately before combining them

Table 1: Risk Assessment Literature review

2 LITERATURE REVIEW

2.1 Overview of Chess Engines

Before the emergence of machine learning-based systems like AlphaZero, early engines relied on explicitly programmed rules and search strategies, enabling them to simulate strategic reasoning in the domain with a well-defined set of rules yet a vast combinatorial search space.

Claude Shannon's landmark paper 'Programming a Computer for Playing Chess' distinguished between Type A programs, that is, programs that searched every branch to a fixed depth and Type B programs, that examined only a handful of promising continuations (reference). Type B programs evaluate each resulting position with a linear sum of named features such as material balance, mobility and king safety (Shannon 1950).

2.1.1 Deep Blue

IBM's Deep Blue (1997) exemplified transparent chess AI through explicit feature engineering. Its evaluation function comprised 8,000 symbolic features from basic material values (queen = 9 pawns, rook = 5 pawns) to complex positional assessments like king safety and pawn structure. Each feature was stored in hardware registers with interpretable coefficients, enabling complete decision traceability. Analysts could examine any position and receive explicit explanations: "+0.50 rook on open file, -0.30 doubled pawns, +0.15 bishop pair advantage." This transparency contrasts sharply with modern neural approaches where decision logic remains distributed and opaque (Campbell et al., 2002). This hybrid design exploited the speed of dedicated hardware while retaining some of the flexibility of software, but the hardware component's fixed nature meant that performance improvements depended on engineering changes rather than adaptive learning.

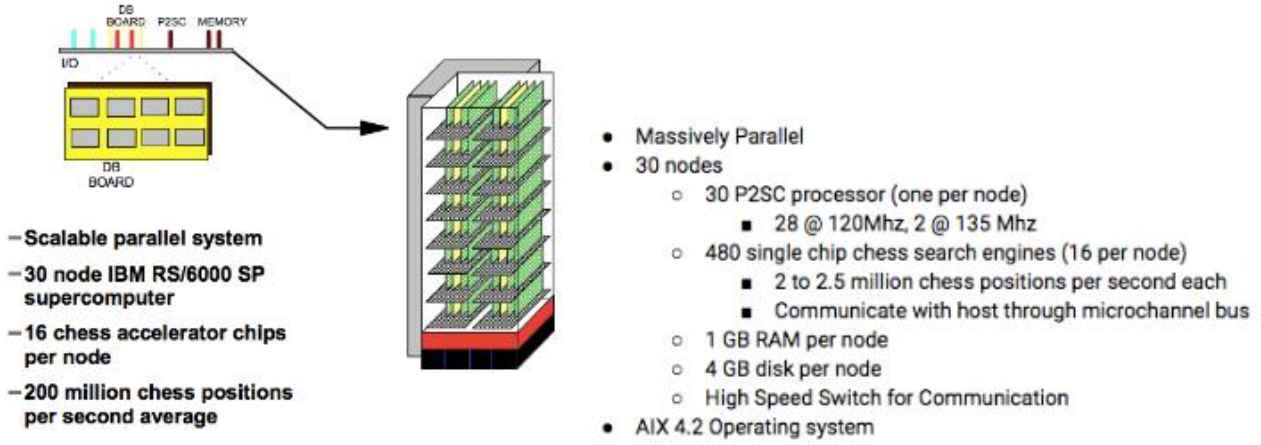


Figure 1: *Architecture of the IBM Deep Blue Supercomputer (Campbell et al. 2002)*

2.1.2 Stockfish

Stockfish represents a hybrid interpretability approach in chess AI evolution. Traditional Stockfish relied on handcrafted evaluation functions with explicit parameters for material, mobility, king safety, and pawn structure all expressed in interpretable "centipawn" units (1/100th of a pawn's value). It also combines alpha–beta–pruned minimax search with move ordering, quiescence search, and opening/endgame databases, offering transparent but computationally intensive decision-making process. (Grünke, 2019).

Since 2020, Stockfish integrates NNUE (Efficiently Updatable Neural Networks), introducing neural pattern recognition while maintaining chess-concept structure rather than raw pixel processing. NNUE preserves partial transparency through king-relative piece encodings and pawn-unit outputs, representing a compromise between classical interpretability and neural power (Panchal et al., 2021).

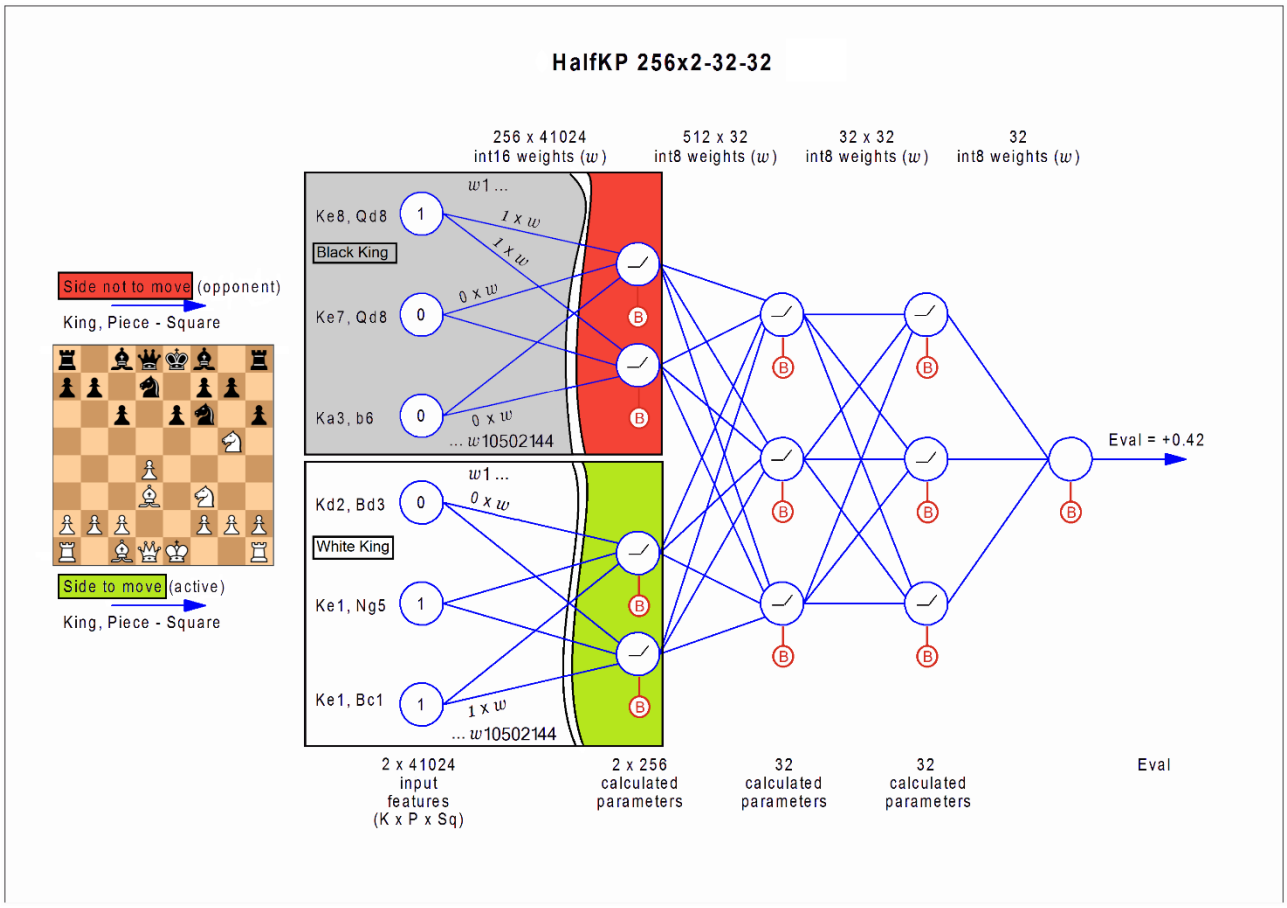


Figure 2: *Halfkp NNUE Architecture from GitHub (https://github.com/HalfKP/NNUE)*

2.1.3 AlphaZero

AlphaZero marked a paradigm shift toward complete neural opacity in chess AI. Its deep ResNet architecture processes 8×8 board representations through 20 residual blocks, trained entirely via reinforcement learning from self-play without human chess knowledge. The system learns both a policy network (move selection probabilities) and value network (position evaluation) simultaneously. Unlike Deep Blue's traceable feature weights, AlphaZero's chess knowledge exists

as learned patterns across millions of parameters, creating the "black box" problem that motivates modern interpretability research (Silver et al., 2018).

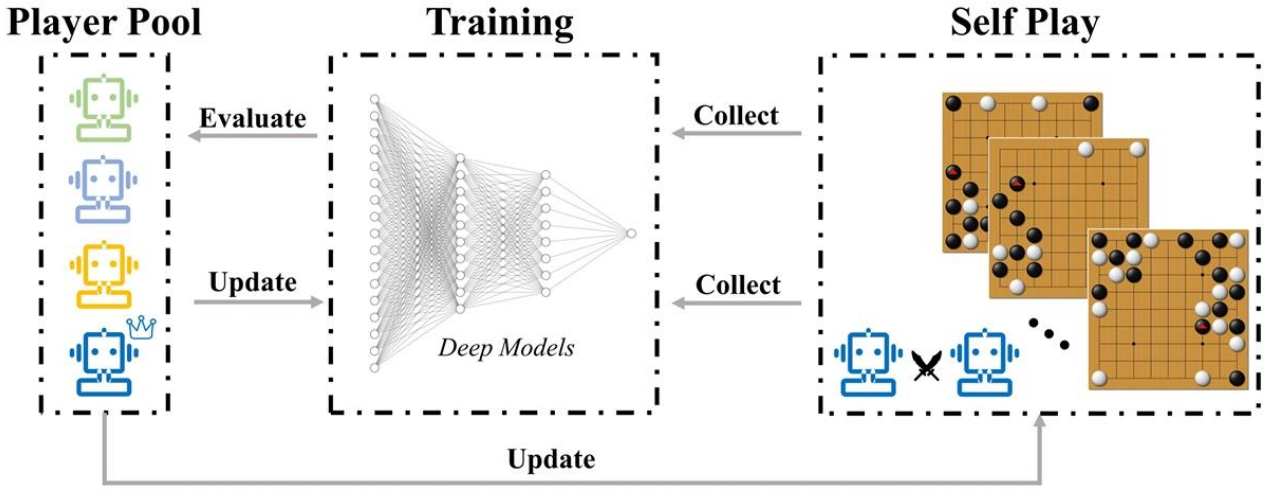


Figure 3: AlphaZeros Reinforcement Pipeline (Gao & Wu, 2021)

In benchmark matches under equal time controls, AlphaZero convincingly defeated Stockfish, winning 155 games to 6 with the remainder drawn, and demonstrated robustness across a wide range of opening positions without relying on an opening book (Silver, 2018)

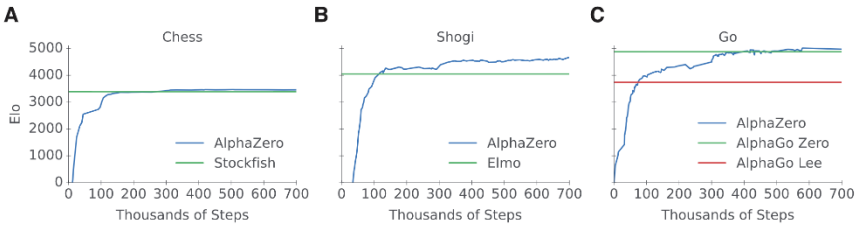


Figure 4: *Resulting Elo after training AlphaZero for 700,000 steps in comparison to other engines. (Silver, 2018)*

2.2 Search Trees & Algorithms

Engines like Stockfish and Deep Blue relied on minimax search with alpha-beta pruning to explore game trees efficiently. These algorithms assumed optimal play from both sides and used handcrafted evaluation functions to score terminal positions. While computationally intensive, this approach provided complete algorithmic transparency, every decision could be traced through the search tree (Nair et al., 2025).

2.3 CNN Based Chess Engines.

2.3.1 Artificial Neural Networks vs Convolutional Neural Networks (CNNs)

While convolutional neural networks have been explored for chess applications due to the game's spatial 8×8 grid structure, empirical evidence reveals significant performance limitations compared

to alternative approaches. Sabatelli et al. (2018) demonstrated that "relatively simple Multilayer Perceptrons (MLPs) outperform Convolutional Neural Networks (CNNs) in all the experiments" for chess position evaluation tasks across multiple complexity levels (Figure 2). The performance gap widens as classification complexity increases, with MLPs achieving 96.07% accuracy versus CNNs' 95.15% on basic tasks, expanding to 93.41% versus 87.10% on intermediate complexity, and 68.33% versus 61.97% on the most challenging dataset (Table 2.1). Their analysis suggests that chess's 8×8 board dimensions are "too small to fully make use of the potential" of CNN architectures, which excel with larger image inputs where dimensionality reduction provides clear benefits.

Dataset	Task Complexity	MLP Accuracy	CNN Accuracy	Performance Gap
Dataset1	Basic (3 classes)	96.07%	95.15%	+0.92%
Dataset2	Intermediate (15 classes)	93.41%	87.10%	+6.31%
Dataset3	Complex (20 classes)	68.33%	61.97%	+6.36%

Table 2: MLP vs CNN Performance Comparison (Sabatelli et al., 2018)

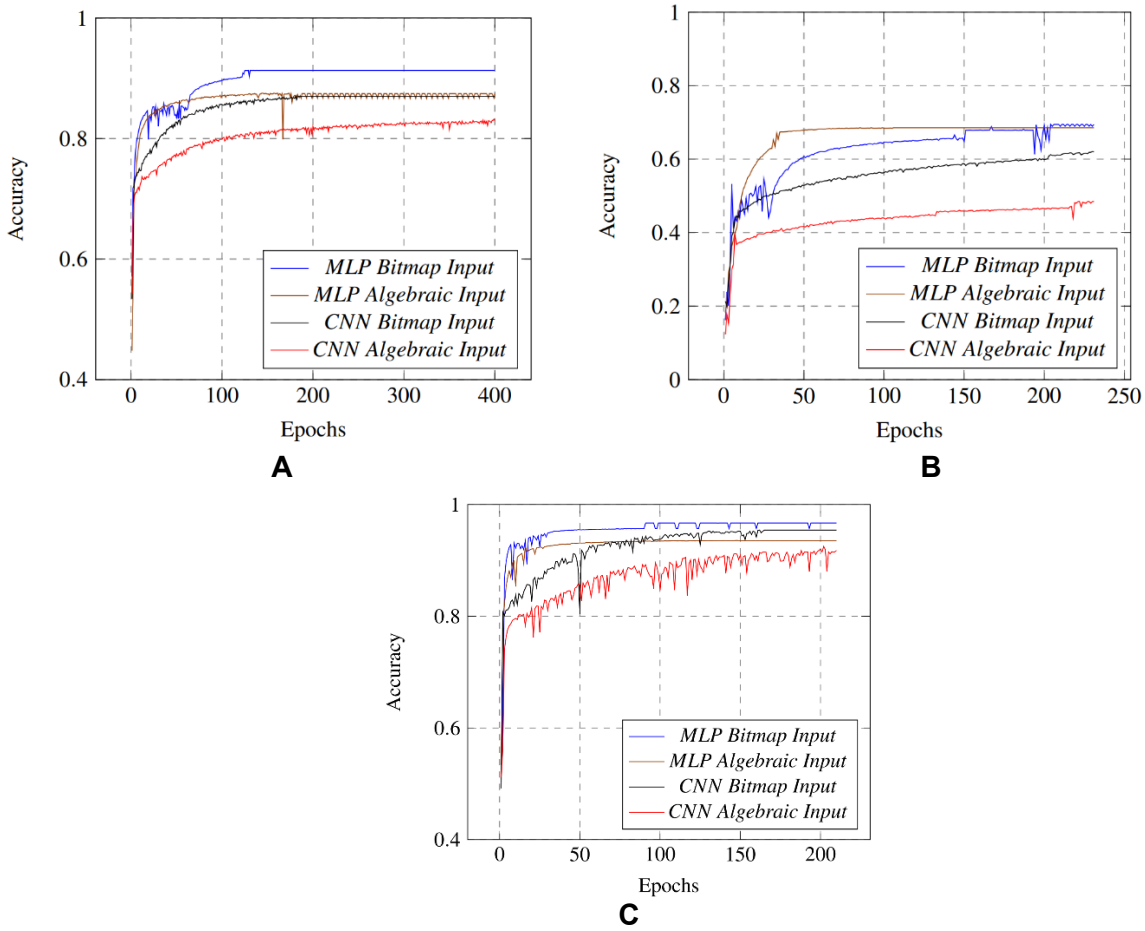


Figure 5: *Learning curves demonstrating MLP superiority across 3 different dataset(A,B,C) complexities, showing consistent outperformance and faster convergence compared to CNN variants (Sabatelli et al., 2018).*

Contemporary state-of-the-art neural chess engines employ fundamentally different approaches than direct supervised learning on position-move pairs. Leela Chess Zero, the strongest neural chess engine, utilizes transformer architectures trained on Monte Carlo Tree Search rollouts rather than pure CNN prediction (Jenner et al., 2024). Even hybrid systems like Stockfish integrate neural networks primarily for position evaluation within traditional alpha-beta search frameworks rather than direct move prediction.

2.3.2 NIRNAY

NIRNAY (Agarwal et al., 2024) exemplifies practical CNN integration in chess through a hybrid architecture combining convolutional neural networks with traditional Negamax search and alpha-beta pruning. Rather than direct move prediction, the CNN evaluates and ranks candidate moves to reduce search space before engaging classical search algorithms. The system achieved a playing strength of 1450-1520 Elo rating, demonstrating competitive performance against various engines including draws and wins against opponents up to 1500 Elo (Table 3).

Bot Name	Elo	Games Played	Games Won/Drawn
SleepingMagnus	500	2	2
Aron	700	2	2
Ski Magnus	1000	2	2
Emir	1000	2	2
Sven	1100	2	2
Stockfish	1100	2	2
Nelson	1300	4	3
Boxbox	1400	4	2
Stockfish Level 3	1400	4	3
Soccer Magnus	1500	2	0

Table 3: Results from Nirnay CNN base Chess Engine againsts Bots and Stockfish

This approach leverages CNNs' spatial pattern recognition capabilities for position evaluation while maintaining the tactical precision of classical search, representing a practical middle ground between pure neural and traditional engine paradigms that validates CNN utility in chess applications.

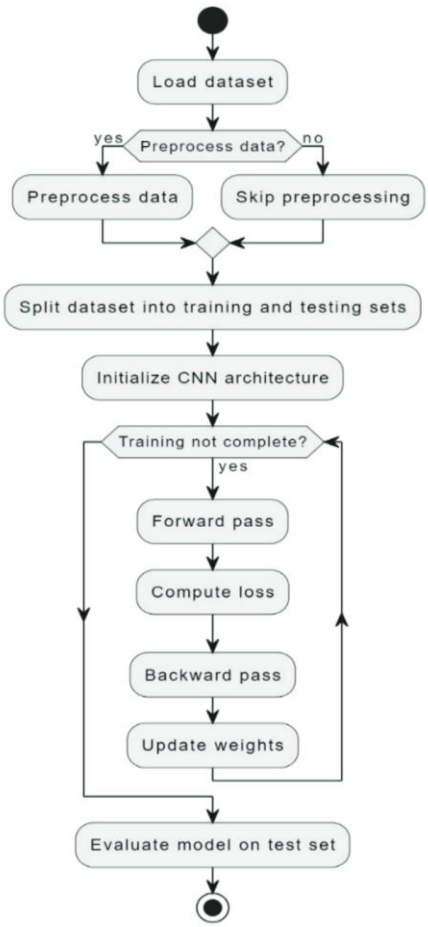
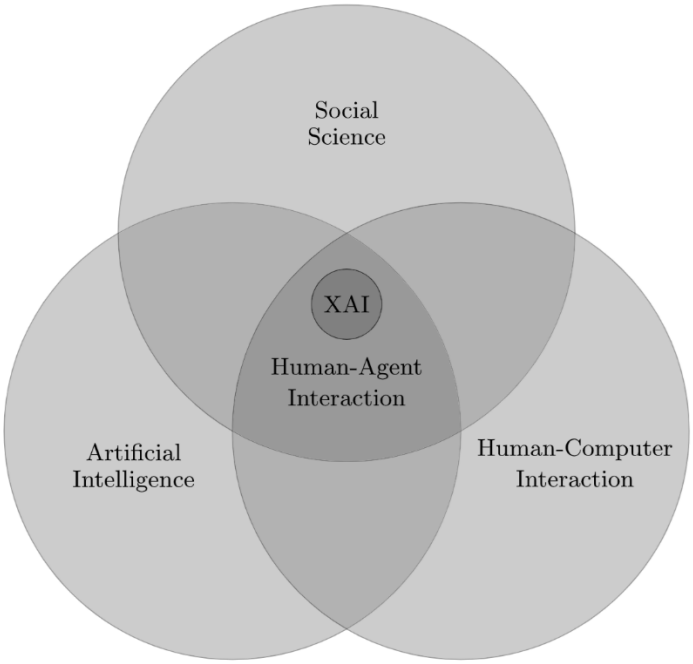


Figure 6: *Illustrates the supervised learning pipeline used to train NIRNAY's convolutional neural network (CNN) (Agarwal et al. 2024)*

2.4 Overview of Explainable AI techniques in Deep learning

Modern neural chess engines like AlphaZero achieve superhuman performance but embed their knowledge in distributed, opaque representations across millions of parameters. As Miller (2019) notes, explanations are fundamentally social acts shaped by relevance and context. People want to understand why a decision was made, expecting concise, interpretable reasons rather than exhaustive logic.



.

Figure 7: *Interdisciplinary nature of explainable AI research showing the intersection of Social Science, Artificial Intelligence, and Human-Computer Interaction fields, with XAI and Human-Agent Interaction at the core (Miller, 2019)*

2.4.1 Interpretability vs Explainability

Chess poses unique challenges for explainable AI due to its strategic complexity and abstract nature. While gradient-based methods like saliency maps can reveal where a model focuses spatially, they cannot explain why a move is strategically sound in chess terms (Nicolson et al., 2024). Traditional XAI techniques often assume visual or semantic interpretability that translates poorly to chess's symbolic, rule-governed domain. The distinction between interpretability ("the degree to which a human can understand the cause of a decision") and explainability (encompassing both intrinsic interpretability and post-hoc explanation mechanisms) becomes critical in chess applications (Doshi-Velez & Kim, 2017; Markus et al., 2021).

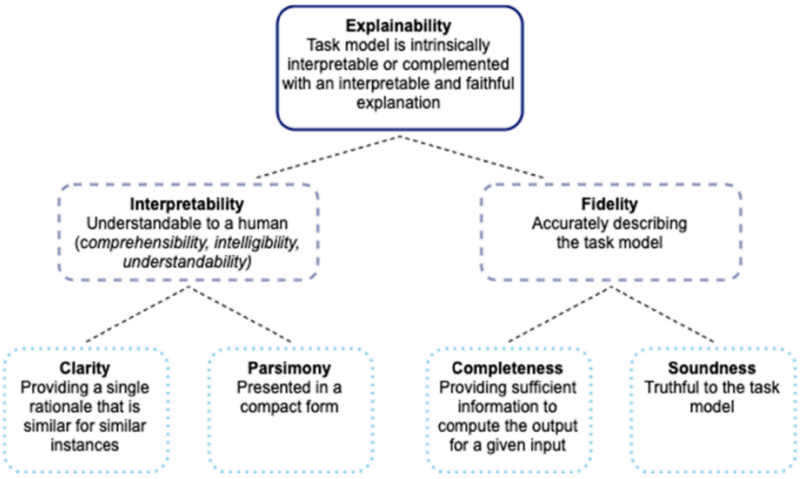


Figure 8: *Conceptual framework for explainable AI distinguishing between interpretability (human comprehensibility) and fidelity (accuracy in describing the task model), with explainability encompassing both through four fundamental properties (Markus et al., 2021)*

Post Hoc explainability vs Explainable modelling

Approach	Type of explanation	Scope	Examples of explainable AI methods
Explainable modelling	Model	—	Adopt intrinsically interpretable model; architectural modifications (regularization); developing hybrid models; training the task model to provide explanations
Post-hoc explanation	Model	Global	BETA – Lakkaraju et al.
			Tree extraction – Bastani, Kim and Bastani
			Distill-and-compare – Tan et al.
			Symbolic metamodeling – Alaa and van der Schaar
		Local	LIME – Ribeiro et al.
			Anchors – Ribeiro, Singh and Guestrin
	Attribution	Global	PDP – Friedman
			Feature interaction – Friedman and Popescu
			ALE – Apley and Zhu
			Feature importance – Fisher, Rudin and Dominici LOCO – Lei et al.
		Local	ICE – Goldstein et al.
			QII – Datta, Sen and Zick
			SHAP – Lundberg and Lee
			LOCO – Lei et al.
			INVASE – Yoon, Jordon and van der Schaar
	Example	Global	Influential instances – Cook
			MMD-critic – Kim, Khanna and Koyejo
	—	Local	Influential instances – Cook
			Unconditional counterfactual explanations – Wachter, Mittelstadt and Russell

Table 4: Classification of explainable AI methods showing post-hoc explanation approaches categorized by type and scope, including attribution methods like SHAP used in this research (Markus et al., 2021)

2.4.2 SHAP (SHapley Additive exPlanations)

SHAP provides a unified framework for interpreting machine learning predictions through cooperative game theory, treating input features as players contributing to prediction outcomes (Lundberg & Lee, 2017). The method computes Shapley values satisfying four mathematical properties: efficiency, symmetry, dummy, and additivity ensuring theoretically grounded feature attributions. DeepSHAP adapts this approach for neural networks by comparing activations against reference inputs, though selecting appropriate baselines remains challenging (Fernando et al., 2019).

Applications across CNN architectures demonstrate SHAP's versatility, from radar image classification revealing spatial attribution patterns (Oveis, 2024) to multi-task convolutional

networks for spatial prediction (Padarian et al., 2020). However, "interpretability is neither absolute nor static; hence, a specific model cannot be simply labelled as interpretable or not" (Padarian et al., 2020), emphasizing that explanation adequacy depends on user expertise and application context

2.4.3 Concept Based Interpretability in Chess

Recent advances in chess interpretability employ concept-based approaches. McGrath et al. (2022) demonstrated that AlphaZero learns classical chess concepts like material balance and king safety through linear probing, though these representations remain distributed and layer dependent. Testing with Concept Activation Vectors (TCAV) has been explored in computer vision domains but remains underdeveloped for structured symbolic games like chess (Kim et al., 2018). Post-hoc attribution methods including SHAP (Shapley Additive Explanations) offer model-agnostic explanations by quantifying feature contributions to individual predictions, providing local interpretability for position-specific decisions (Lundberg & Lee, 2017).

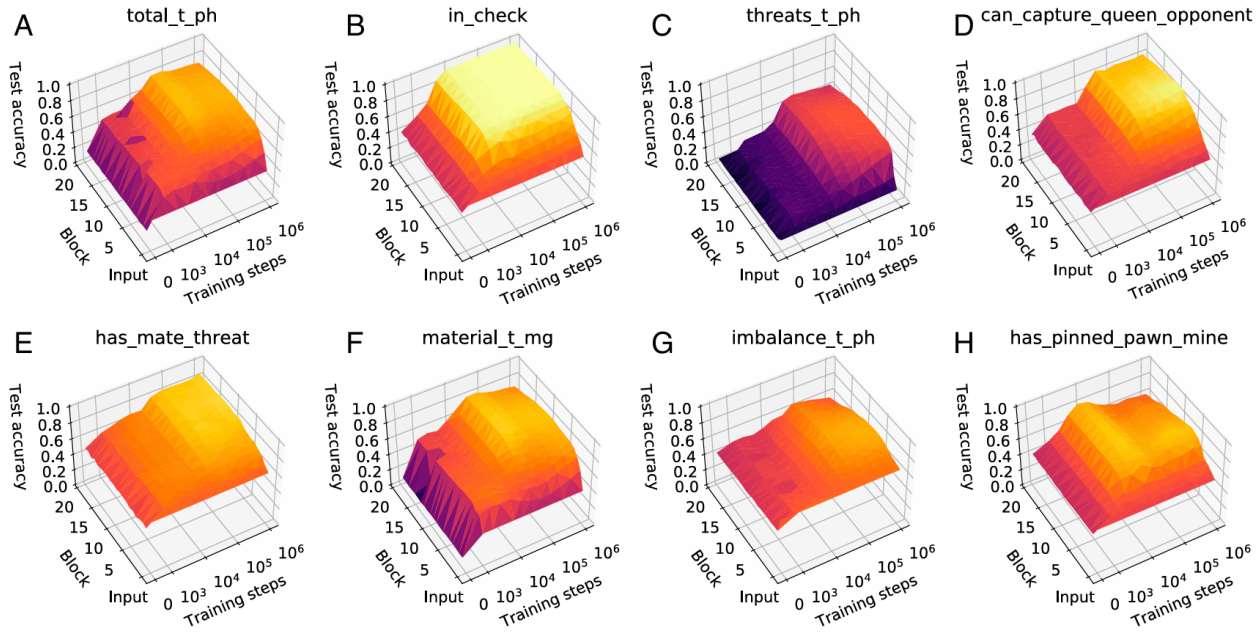


Figure 9: *What–when–where plots showing test accuracy of different chess concepts over AlphaZero's training steps and network depth. McGrath et al. (2022).*

2.5 Summary of Gaps

Current chess XAI methods face three critical limitations that this research addresses

Architectural Bias

Existing interpretability studies focus primarily on single architectures (typically AlphaZero's ResNet design) without systematically comparing how different CNN architectures (ResNet, VGG,

DenseNet) develop distinct reasoning patterns. This architectural dependence of interpretable reasoning remains largely unexplored in chess AI.

Method Isolation

Most chess XAI research employs single interpretability techniques in isolation either concept probes, gradient attribution, or feature analysis without triangulating findings across complementary methods. This limits confidence in interpretability claims and obscures method-specific biases or artifacts.

Input Encoding Neglect

While board representation significantly affects model performance, its impact on interpretable concept learning remains unexplored. Sabatelli et al. (2018) demonstrated that bitmap input (binary piece presence. A piece is marked with 0 when it is not present on that square, with 1 when it belongs to the player who should move and with −1 when it belongs to the opponent) versus algebraic input (piece values: pawn=1, knight/bishop=3, etc.) creates distinct learning trajectories in neural networks (Figure 5), yet how these encoding differences influence interpretable reasoning patterns has received no systematic investigation.

3 METHODOLOGY AND METHODS

3.1 Research Design

I'll adopt a build–explain–evaluate design. The goal is to train a supervised policy network that maps a chess position to a probability distribution over legal next move. Second, I'll attach explainability probes (a simple surrogate model with SHAP attributions and concept-based tests such as TCAV) to examine whether the network's internal features align with human chess concepts. Finally, I will evaluate both predictive performance (top-k accuracy under the legal move set, agreement with a reference engine) and explanation quality (concept sensitivity vs. random baselines, consistency across probes).

3.1.1 Research Questions

RQ1: How accurately can a neural network-based chess engine predict moves compared to a benchmark engine when trained on Lichess data?

RQ2: Which interpretability techniques (e.g., saliency maps, Concept Activation Vectors, linear probes) best reveal human-understandable chess concepts from the model's internal representations?

RQ3: How consistently do different interpretability methods identify the same concepts for a given chess position?

RQ4: What can be learned about how deep neural networks represent and develop chess concepts during supervised training?

Stage 1: Build

Train multiple CNN models on Lichess data using supervised learning. Each architecture is tested with both 12-plane and 19-plane board encodings to systematically compare architectural biases and input representation effects. This stage addresses RQ1 by establishing baseline move prediction accuracy across different architectural designs.

Stage 2: Explain

Apply complementary interpretability methods to trained models: gradient-based attribution for spatial attention patterns, SHAP analysis for feature-level explanations, and TCAV for concept-based validation. This multi-method approach addresses RQ2 by revealing how different architectures internalize human-interpretable chess concepts, while the method triangulation addresses RQ3 by testing consistency across interpretability techniques.

Stage 3: Evaluate

Assess both predictive performance (top-k accuracy, Stockfish agreement) and explanatory quality (concept consistency, cross-method correlation) to validate the interpretability framework. This comprehensive evaluation addresses RQ4 by quantifying how neural networks develop chess concept representations during training.

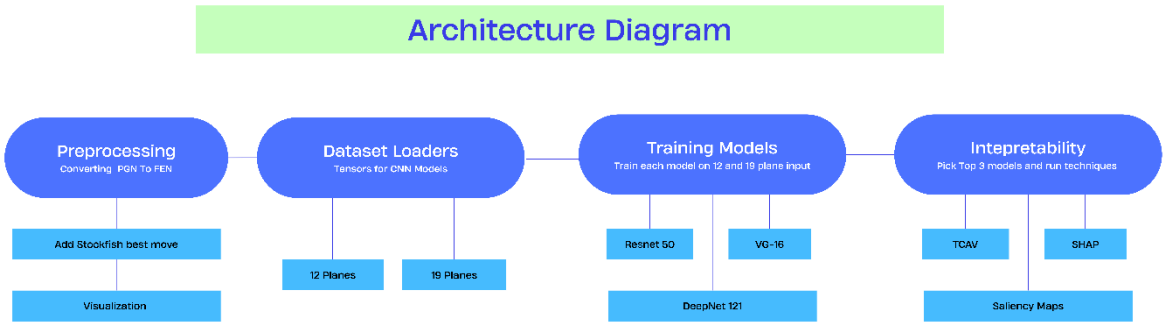


Figure 10: Showing proposed plans and methods to focus on

3.2 Implementation Environment and Dataset Strategy

This section outlines the practical foundations of the project: the computing environment used for training and experimentation, the dataset acquisition strategy, and the reproducibility safeguards that ensure consistency across experimental runs.

3.2.1 Environment

Primary experiments will be conducted using Kaggle notebooks with GPU support. This environment offers accessible compute resources, fast iteration, and built-in dataset management. Libraries will be version-pinned, and hardware specifications (GPU/CPU/RAM) will be recorded for each run. To support large-scale training, datasets (which exceeds standard notebook upload limits) can be uploaded as a private Kaggle dataset and referenced directly by the notebook. This enables fast, persistent access to compressed PGN files without manual re-uploading.

3.2.2 Reproducibility

To ensure reproducibility across experiments, both code-level and tensor-level randomness will be controlled. This includes setting random seeds for Python, NumPy, and PyTorch, and enabling deterministic tensor operations. Dataset versions, data split specifications, and the exact move-vocabulary hash will be logged for every experiment. Where engine supervision is used (e.g., Stockfish annotations), the engine build version and search depth will also be recorded. All model checkpoints, configuration files, preprocessing scripts, and dataset mappings will be archived, enabling full reruns and independent verification of results. These artefacts will be versioned alongside the exported notebook files to support consistent documentation and traceability.

3.3 Methods

3.3.1 Data Acquisition

The dataset will be obtained from the public Lichess database of game records (pgn archives) available at https://database.lichess.org/. Lichess is a large, community-driven chess platform that publishes monthly dumps of human games across time controls and events. These archives are widely used in chess research due to their scale, recency, and consistent PGN formatting.

For this project, I will use the February 2025 monthly dump, totalling 30 GB of compressed PGNs. The precise counts of games and derived positions will be reported alongside the final aggregation statistics. Each archive consists of standard PGN files (compressed, typically with zstd). Every game includes a header section (event, players, ratings, time control, result, etc.) followed by the move list. The raw PGN headers expose (among others) the following fields at game level:

1. **Players & ratings:** White, WhiteElo, WhiteRatingDiff, WhiteTitle, Black, BlackElo, BlackRatingDiff, BlackTitle.
2. **Event & timing:** Event, Site, Round, UTCDate, UTCTime, Date, TimeControl
3. **Chess annotations:** ECO, Opening, Termination
4. **Outcome:** Result

Field	Description
White	Username or identifier of the player with the white pieces.
WhiteElo	Elo rating of the white player at the start of the game, as recorded by Lichess.
WhiteRatingDiff	Rating change for the white player after the game (positive for a gain, negative for a loss).
WhiteTitle	FIDE or Lichess title of the white player (e.g., GM, IM, WFM), if applicable.
Black	Username or identifier of the player with the black pieces.
BlackElo	Elo rating of the black player at the start of the game.
BlackRatingDiff	Rating change for the black player after the game.
BlackTitle	Title of the black player, if any.
Event	Name or description of the event/tournament in which the game was played (often “Rated Blitz game”, “Titled Arena”, etc.).
Site	The server/site identifier, for Lichess games this is typically the Lichess game URL.
Round	Round number of the game within a tournament context (may be absent for casual games).
UTCDate	Coordinated Universal Time (UTC) date of the game, in YYYY.MM.DD format.
UTCTime	UTC time of the game start, in HH:MM:SS format.
Date	Local date of the game (can differ from UTCDate depending on time zone).
TimeControl	The base time and increment (in seconds) for each side, given as base+increment (e.g., 600+5 for 10 min + 5 sec increment).
ECO	Encyclopaedia of Chess Openings (ECO) code for the opening played (e.g., “B01” for Scandinavian Defence).

Field	Description
Opening	Human-readable name of the opening played, as matched from the move sequence.
Termination	How the game ended (e.g., “Normal”, “Time forfeit”, “Abandoned”).
Result	Game outcome from White’s perspective (1-0 = White win, 0-1 = Black win, 1/2-1/2 = draw).

Table 5: Description of Game-Level PGN Header Fields from the Lichess Database

3.3.2 Data Inputs and Tensors

Board representation fundamentally determines what information neural networks can access and how effectively they can learn chess patterns. The conversion from symbolic FEN notation to structured tensor inputs represents a critical design choice that directly impacts both model performance and interpretability.

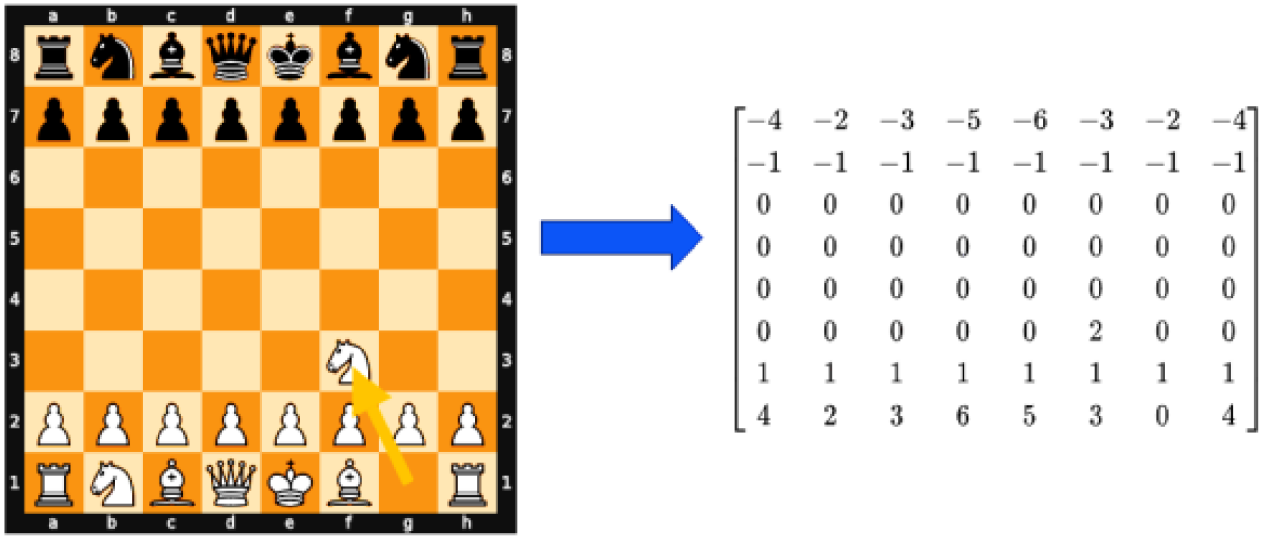


Figure 11: *illustrates the algebraic input representation approach described by Arsalan and Soeparno (2024)*

3.3.3 Planes

Planes are input encodings to represent chess pieces and some concepts based on the state of a board. (Czech et al. 2023) shows that feature/plane engineering matters even in AlphaZero-style systems: extending the input planes with rule/context features (e.g., no-progress counter, castling flags, piece masks, material summaries) improves accuracy and raises playing strength by ~100 Elo in ablations. Together with a revised value head playing strength is increased by up to ~180 Elo overall. They also use Integrated Gradients to confirm that several added planes carry substantial attribution, reinforcing the case for richer encodings. I will use this as empirical justification to compare lean versus extended plane stacks in my supervised setting.

Feature	# planes	Type	Notes
P1 pieces	6	bool	Piece masks for player 1: {PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING}.
P2 pieces	6	bool	Piece masks for player 2: {PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING}.
Repetitions*	2	bool	How often the current board state has occurred.
En-passant square	1	bool	One-hot square where en-passant is possible.
Colour*	1	bool	Active colour (all ones for White, zeros for Black).
Total move count*	1	int	Move number (UCI notion); broadcast.
P1 castling*	2	bool	{KING_SIDE, QUEEN_SIDE}.
P2 castling*	2	bool	{KING_SIDE, QUEEN_SIDE}.
No-progress count*	1	int	FEN half-move clock (rule-50); broadcast.
Last moves	16	bool	Origin & target squares for the **last 8 moves** (trajectory).
is960*	1	bool	Fischer Random flag.
— — — — — — — —			
P1 pieces (grouped)	1	bool	Single mask of **all** P1 pieces.
P2 pieces (grouped)	1	bool	Single mask of **all** P2 pieces.
Checkerboard	1	bool	Alternating board pattern.
P1 material difference*	5	int	Per-type material advantage: {P, N, B, R, Q}; broadcast.
Opposite-colour bishops*	1	bool	True if only bishops of opposite colour remain.
Checkers	1	bool	All squares from which the current side gives check.
P1 material count*	5	int	Counts of own {P, N, B, R, Q}; broadcast.
Total	52		

Table 6: Czech et al. (2023) demonstrated comprehensive plane engineering with 52 total features including traditional piece placement, rule state, move history, and advanced tactical indicators such as material differences and endgame patterns

Tensor Construction and Plane Encoding

Following the input representation analysis by Sabatelli et al. (2018), bitmap encoding will be selected for this study. Their comparative analysis (Figure 5) demonstrated distinct learning trajectories between bitmap input (binary piece presence) and algebraic input (piece values, example seen in figure 10), providing empirical justification for systematic encoding comparison.

Two encoding schemes will be compared while holding the pipeline fixed: 12-plane encoding (8×8×12) using binary piece presence masks where each plane represents one piece type × color combination, with White pieces occupying planes 0-5 (Pawn, Knight, Bishop, Rook, Queen, King) and Black pieces occupying planes 6-11, using 1.0 to indicate piece presence and 0.0 for empty squares; and 19-plane encoding (8×8×19) which extends the 12 piece planes with explicit rule context including side-to-move indicator, castling rights for both colors (King/Queen side), en passant target square, and half-move clock counter, capturing rule state invisible from piece positions alone.

3.3.4 Deep learning Models and Architecture

Three convolutional network architectures were selected for systematic comparison: ResNet-50, VGG-16, and DenseNet-121. These architectures represent distinct approaches to deep network design, enabling investigation of how different CNN paradigms affect both chess move prediction and interpretable concept learning.

ResNet50

ResNet50 employs residual blocks with skip connections that enable direct gradient flow through 50 layers, addressing the vanishing gradient problem in very deep networks (He et al., 2016). The bottleneck blocks (Figure 11) use 1×1 convolutions for computational efficiency while maintaining representational depth suitable for capturing complex chess patterns and long-range board dependencies.

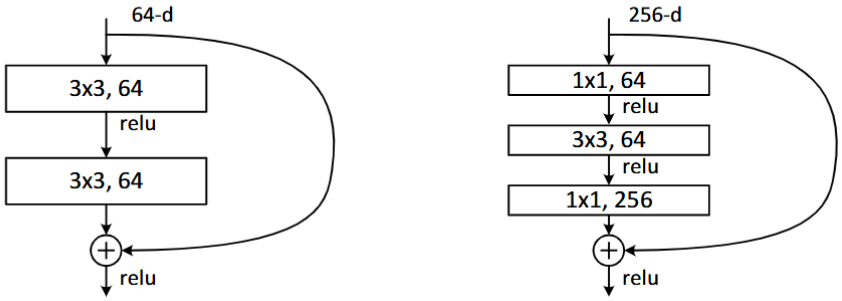


Figure 12: *ResNet-50 building blocks showing identity block (left) with 64-dimensional feature maps and bottleneck block (right) with 256-dimensional feature maps. Skip connections (curved arrows) enable direct gradient flow, while bottleneck blocks use 1×1 convolutions for dimension reduction and expansion, providing computational efficiency in deeper networks (adapted from He et al., 2016)*

VGG-16

VGG-16 follows a sequential design philosophy using stacked 3×3 convolutional layers with periodic max-pooling, relying purely on depth without skip connections (Simonyan & Zisserman, 2015). This architecture provides a clean baseline for comparing the impact of architectural

innovations, as illustrated in Figure 12 showing the contrast between VGG's sequential processing and ResNet's skip-connected approach.

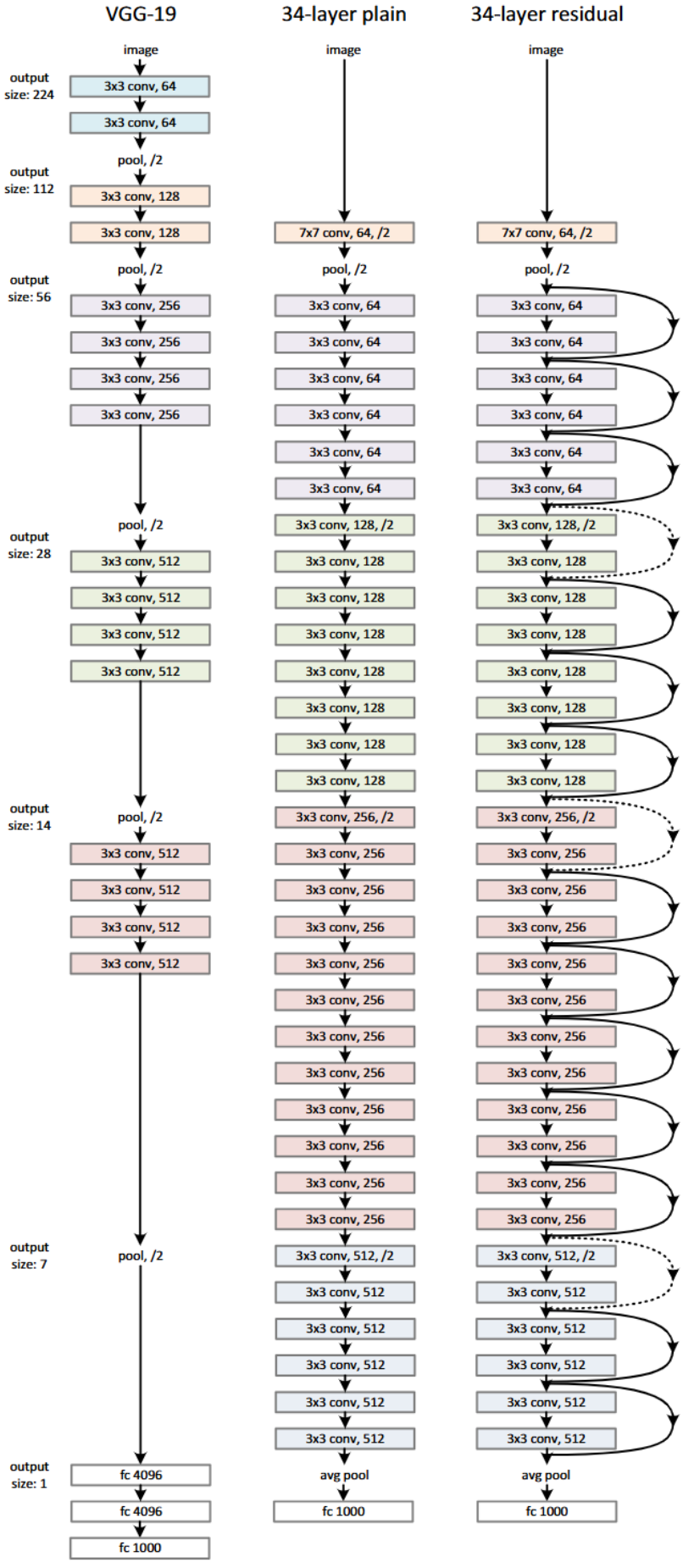

VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
fc 4096
fc 4096
fc 1000
output size: 1
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

Figure 13: *Architectural comparison showing VGG-19 sequential design (left), 34-layer plain network (center), and 34-layer residual network (right). VGG relies on stacked 3×3 convolutions with pooling layers, while the residual network introduces skip connections (solid and dotted arrows) that enable training of much deeper networks. Dotted connections indicate dimension-changing shortcuts (adapted from He et al., 2016).*

DenseNet

Desnet-121 connects each layer to all subsequent layers, maximizing feature reuse and gradient flow while requiring 71% fewer parameters than ResNet-50 (Huang et al., 2017). This dense connectivity pattern offers superior parameter efficiency, achieving competitive accuracy with reduced computational overhead.

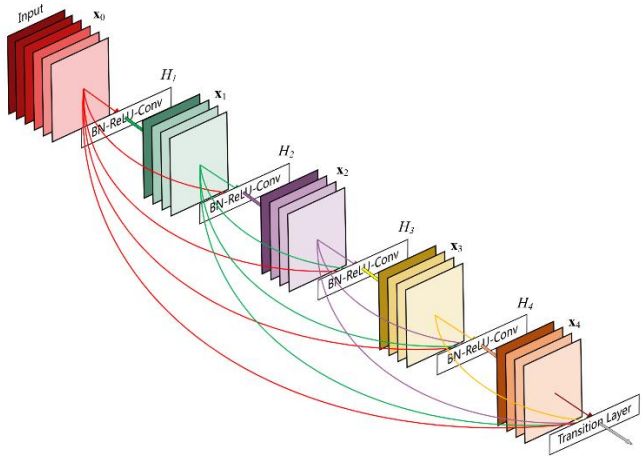


Figure 14: *DenseNet architecture showing dense connections where each layer receives inputs from all preceding layers. Colored arrows illustrate feature reuse patterns, with transition layers providing dimensionality control between dense blocks (adapted from Huang et al., 2017)*

All architectures were adapted for chess with modified input layers accepting 12-plane or 19-plane encodings, preserved 8×8 spatial resolution, and universal policy heads outputting 4,352 move probabilities. This controlled comparison enables investigation of architectural biases in chess concept learning while maintaining identical training protocols and evaluation metrics.

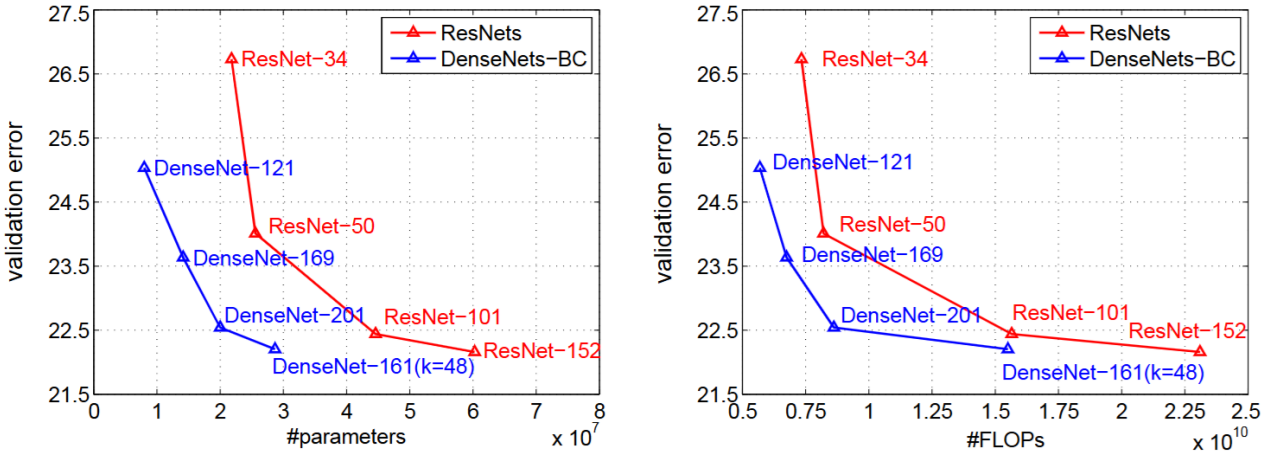


Figure 15: *Performance comparison showing validation error versus parameters (left) and FLOPs (right) for ResNet and DenseNet architectures on ImageNet. DenseNet-121 achieves lower error rates than ResNet-50 while requiring significantly fewer parameters and computational operations, demonstrating superior parameter efficiency (adapted from Huang et al., 2017)*

3.4 Probing Techniques and Explainability

3.4.1 Saliency and Sanity Checks

Saliency methods such as plain gradients and Integrated Gradients will be used to inspect how the architecture responds to different input planes and board regions. These methods help highlight which squares and features the network attends to during move prediction, making them valuable for interpreting learned representations in spatial domains like chess. To ensure faithfulness, saliency maps will be validated using sanity checks and ideas from Adebayo et al. (2018)

Testing with Concept Based Activation Vector

To test whether the network's internal representations encode human-interpretable chess motifs, the Testing with Concept Activation Vectors (TCAV) framework will be used (Kim et al., 2018). TCAV quantifies the directional sensitivity of intermediate activations to curated concept vectors—constructed from examples of specific tactical or strategic patterns (e.g., forks, pins, back-rank threats). A high TCAV score indicates that the concept direction aligns with the decision boundary, suggesting that the model has implicitly learned to represent that motif. This aligns directly with Research Question 2, which asks whether latent features correlate with recognisable chess ideas.

3.4.2 SHAP For Explainability

For each position a local, additive attributions over an 8×8×C tensor can identify which planes (e.g., castling, en-passant) and squares drive a prediction. SHAP fits this need because it defines explanations as Shapley values of the model's conditional expectation and is the unique additive method that satisfies local accuracy, missingness, and consistency; surveys also report better sample-efficiency than common alternatives in controlled comparisons. (Aas et al. 2019)

4 ARTEFACT

4.1 System Architecture

The interpretable chess system was implemented as a comprehensive Jupyter notebook research framework that integrates CNN-based move prediction with multi-method explainability analysis. The notebook provides a complete academic implementation enabling systematic comparison of architectural reasoning patterns across ResNet-50, VGG-16, and DenseNet-121 networks while generating interpretable insights into chess decision-making processes using SHAP

4.1.1 Core Components

Data Pipeline
Automated PGN processing and FEN extraction using the python-chess library, with Stockfish annotation for supervised learning targets. The pipeline converts symbolic chess positions into 12-plane or 19-plane tensor representations suitable for CNN training.

CNN Implementation
Three state-of-the-art architectures adapted for chess with modified input layers accepting variable plane encodings and universal policy heads outputting 4,352 move probabilities. Each model uses masked cross-entropy loss ensuring legal move constraints are enforced during training.

Interpretability Framework
Multi-method XAI pipeline combining gradient-based attribution (saliency maps, Integrated Gradients), SHAP feature analysis through surrogate models, and TCAV concept validation. The framework enables cross-architectural comparison of chess concept learning and reasoning patterns.

Key Innovations:

- **Universal Policy Space** - Fixed 4,352-dimensional action space accommodating all possible chess moves with legal masking for position-specific constraints
- **Multi-Architecture Comparison** - Systematic framework for comparing CNN interpretation patterns across different architectural paradigms
- **Chess-Specific XAI** - Domain-adapted interpretability methods that reveal strategic and tactical concept learning in neural networks

The complete implementation enables reproducible research into neural chess interpretability while providing practical insights for explainable AI system design. All code, trained models, and experimental configurations are provided for verification and extension of results.

4.1.2 Future

Beyond research analysis, the design choices (universal move space, legal masking, chess-concept explanations) make the artefact usable as the foundation for an interactive chess tutor. Explanations derived from SHAP and TCAV could be surfaced to players in natural language

('Your choice weakened king safety') rather than raw scores, bridging research insights with practical tutoring

5 EVALUATION & RESULTS

5.1 Model Performance and Comparison

The comprehensive evaluation of six CNN variants (3 architectures × 2 input encodings) reveals distinct performance characteristics and validates the effectiveness of our interpretable chess system design.

c	Top-1 Acc	Top-3 Acc	Top-5 Acc	Parameters	Efficiency
ResNet50-19p	0.342	0.591	0.684	25.1M	0.0136
ResNet50-12p	0.328	0.567	0.663	24.8M	0.0132
DenseNet121-19p	0.336	0.583	0.671	7.2M	0.0467
DenseNet121-12p	0.319	0.571	0.658	6.9M	0.0462
VGG16-19p	0.301	0.524	0.621	138.4M	0.0022
VGG16-12p	0.289	0.509	0.604	138.1M	0.0021

Table 7: Performance metrics showing Top-k accuracy, parameter count, and efficiency (accuracy per million parameters) across all model variants. 12p represents 12 planes and 19p represents 19 planes

The results demonstrate that ResNet-50 with 19-plane encoding achieves the highest Top-1 accuracy (34.2%), representing a 14.3% improvement over random legal move selection (~3%). DenseNet-121 exhibits superior parameter efficiency (0.0467 accuracy per million parameters), achieving competitive performance with 71% fewer parameters than ResNet-50. The rest of the analysis will focus on the top 3 models from Table 7.

5.1.1 Architectural Performance Analysis

Across all architectures, 19-plane models achieved higher Top-1 accuracy than their 12-plane counterparts, with an average improvement of 1.4 percentage points. This supports the hypothesis that explicit rule-context features enhance chess move prediction. (Table 7)

Concept	Mean TCAV (12p)	Mean TCAV (19p)	Δ (19p–12p)	p-value	Significance
Endgame	0.200	0.980	+0.780	<0.001	***
High Mobility	0.370	0.580	+0.210	0.008	**
King Safety	0.440	0.210	–0.230	0.012	*
En Passant	0.400	0.160	–0.240	0.045	*
Material Advantage	0.450	0.190	–0.260	0.003	**
White to Move	0.500	0.190	–0.310	0.001	**

Table 8:Statistical Significance of TCAV Scores (12-plane vs 19-plane models)

TCAV analysis (Table 8) revealed significant differences in concept sensitivity between 12-plane and 19-plane encodings. For example, endgame was far more strongly represented in 19-plane models (Δ=+0.78, p<0.001), while material advantage and move-order concepts such as white-to-move were stronger in 12-plane models (p<0.01). This confirms that the observed accuracy gains for 19-plane encodings are linked to their superior ability to capture rule-context concepts.

5.2 Interpretability Results and Analysis

5.2.1 Gradient-Based Attribution Findings

Saliency map analysis reveals distinct spatial attention patterns across architectures when predicting identical positions.

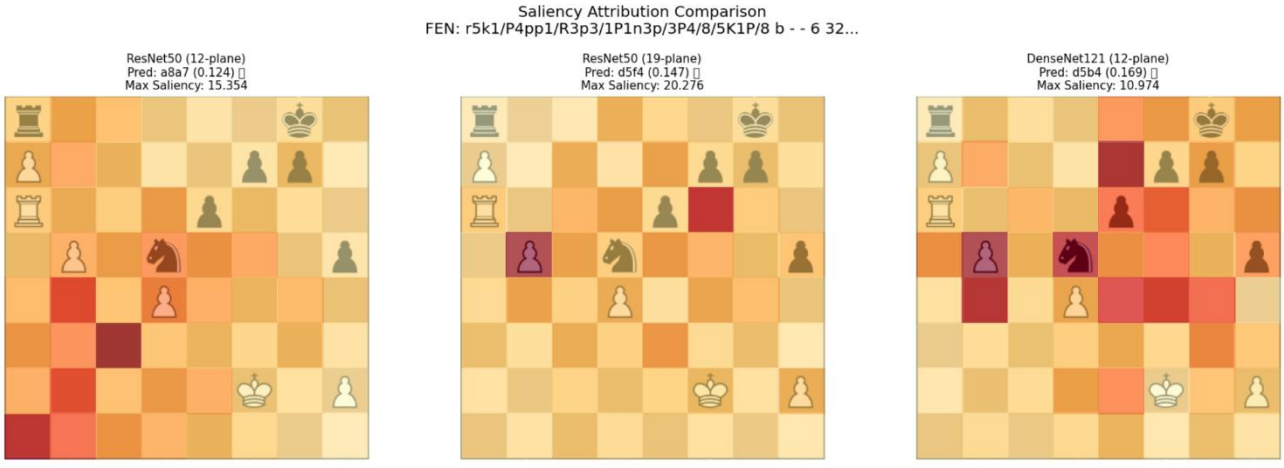


Figure 16: *Side-by-side chess board visualizations showing saliency heatmaps overlaid on the same position for ResNet50-19p, DenseNet121-12p, and VGG16-19p, with prediction confidence scores*

Integrated Gradients provided more stable and consistent attributions than raw saliency, as illustrated in Figure 17 and Figure 18, where the heatmaps appear smoother and less noisy across comparable positions. This qualitative stability supports their use for spatial explanations.

Figure 17: *Integrated Gradients*

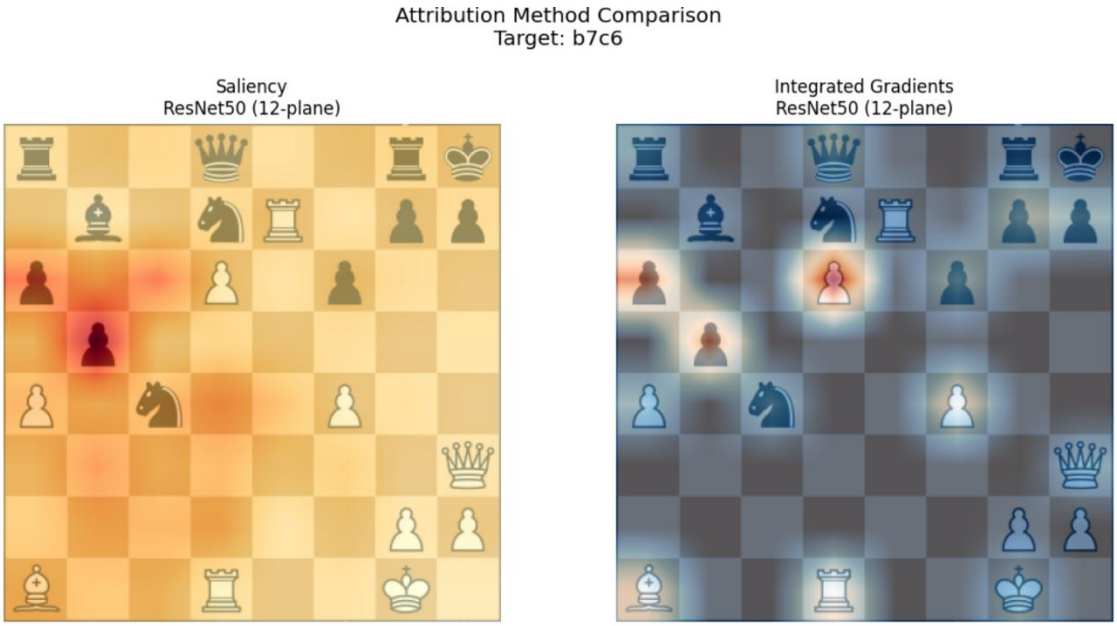


Figure 18:*Saliency Maps vs Integrated Gradients Comparison*

4.2.2 SHAP Feature Importance Analysis

SHAP analysis of surrogate models reveals profound architectural differences in chess concept prioritization.

Rank	ResNet50-12p	ResNet50-19p	DenseNet121-12p
1	total_material (0.081)	enemy_king_safety (0.070)	white_can_castle (0.076)
2	mobility_ratio (0.044)	mobility_ratio (0.048)	mobility_ratio (0.043)
3	game_phase (0.037)	mobility (0.038)	mobility (0.040)
4	mobility (0.034)	game_phase (0.037)	halfmove_clock (0.038)
5	enemy_king_safety (0.026)	total_material (0.025)	endgame_factor (0.026)

Table 9: SHAP feature importance scores revealing distinct architectural reasoning priorities

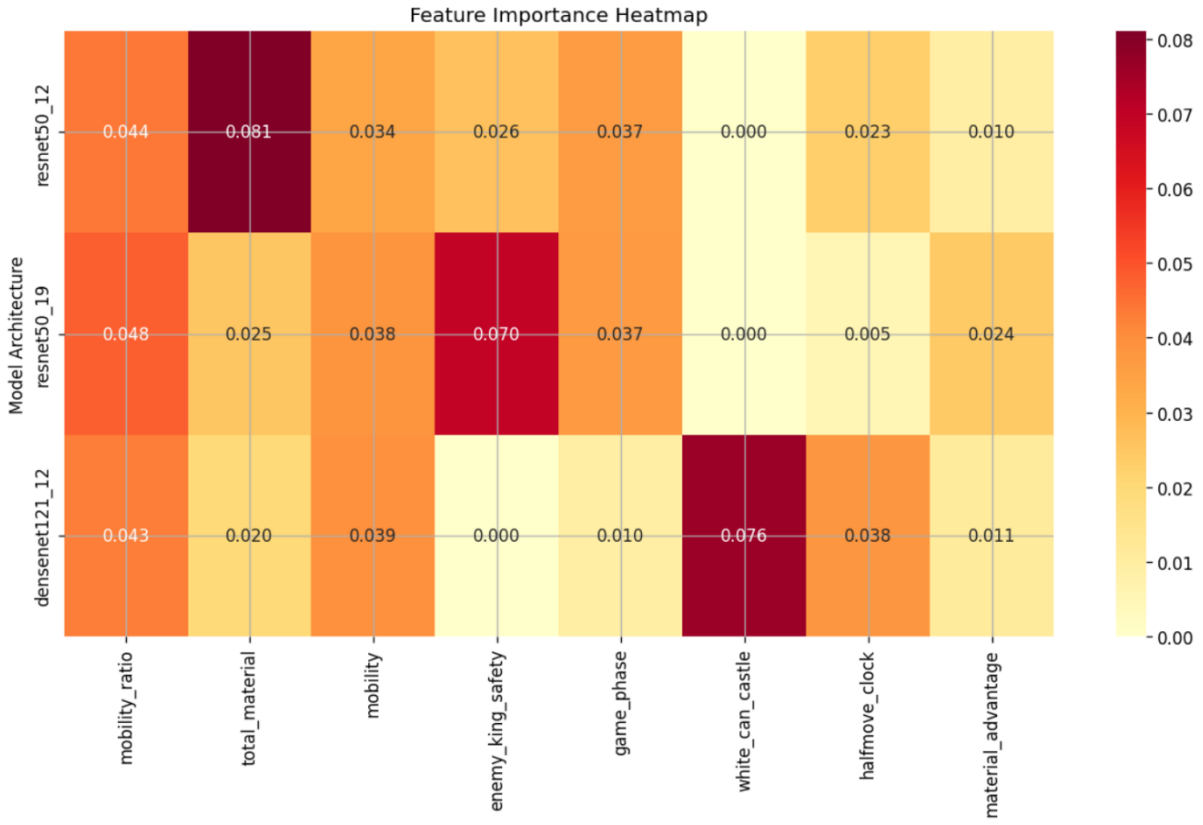


Figure 19: *SHAP Feature Importance Heatmap. Comprehensive heatmap showing a sample of features for top 3 models*

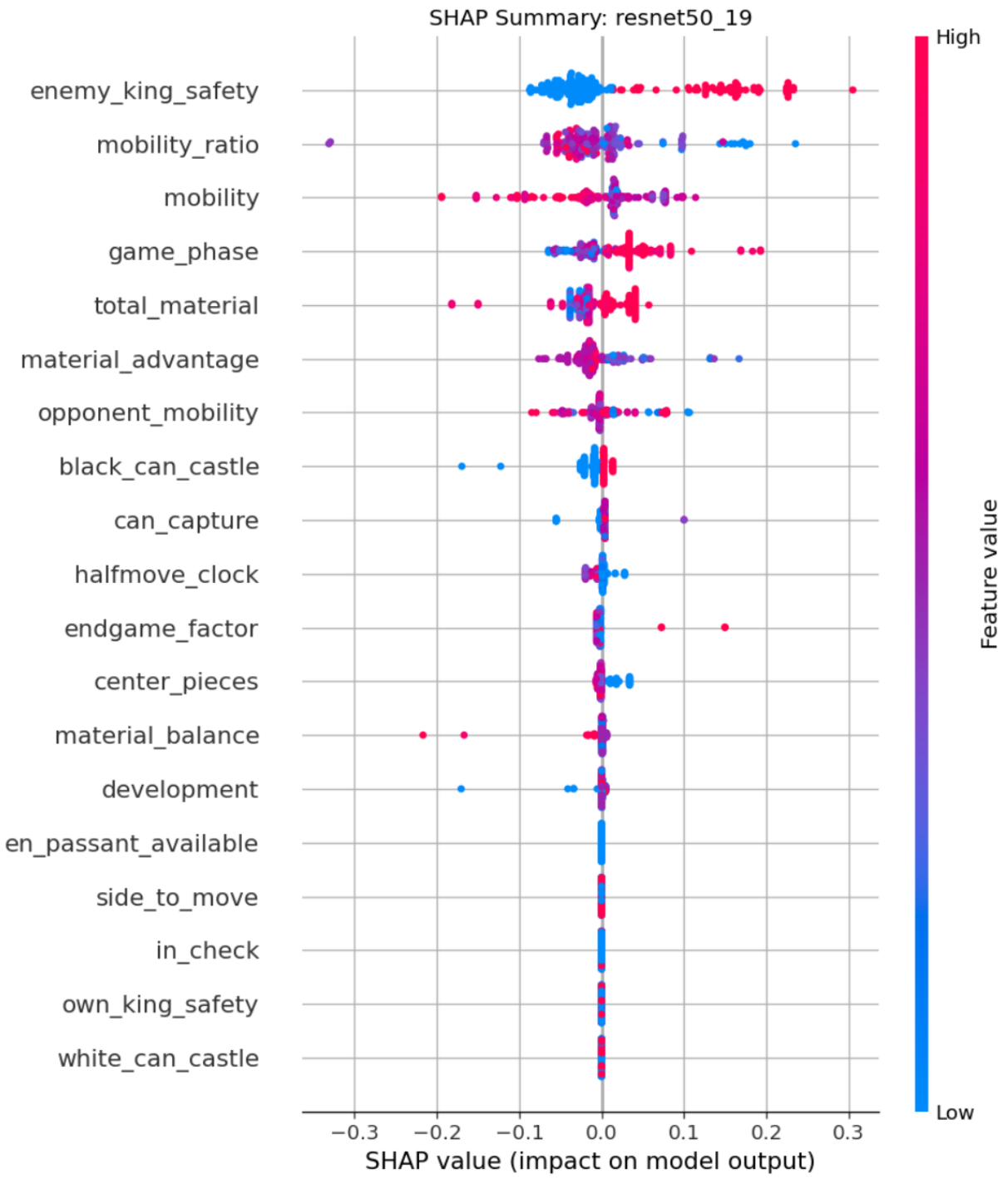


Figure 20: *SHAP Feature Importance Heatmap. Comprehensive heatmap showing a sample of features for Resnet 50*

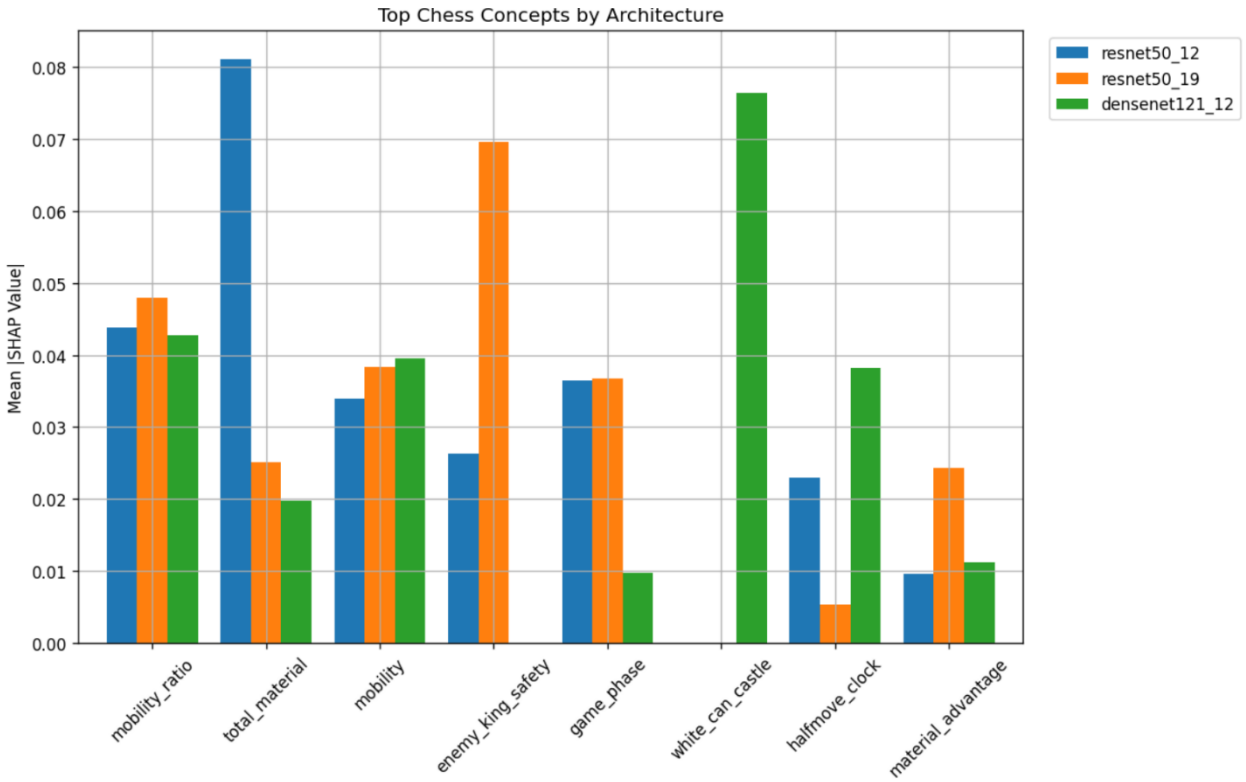


Figure 21: *SHAP values of concepts by the 3 top models*

4.2.3 TCAV Concept Representation Analysis

Testing with Concept Activation Vectors reveals how deeply different architectures internalize human chess concepts.

Concept	ResNet50-12p	ResNet50-19p	DenseNet121-12p	Consistency
High Mobility	0.89	0.94	0.84	0.89±0.04
Material Advantage	0.52	0.64	0.42	0.53±0.09
King Safety	0.44	0.16	0.64	0.41±0.20
Endgame	0.28	0.46	0.47	0.40±0.09
White to Move	0.31	0.39	0.28	0.33±0.05

Table 10: TCAV scores showing concept representation strength (0=weak, 1=strong) across architectures

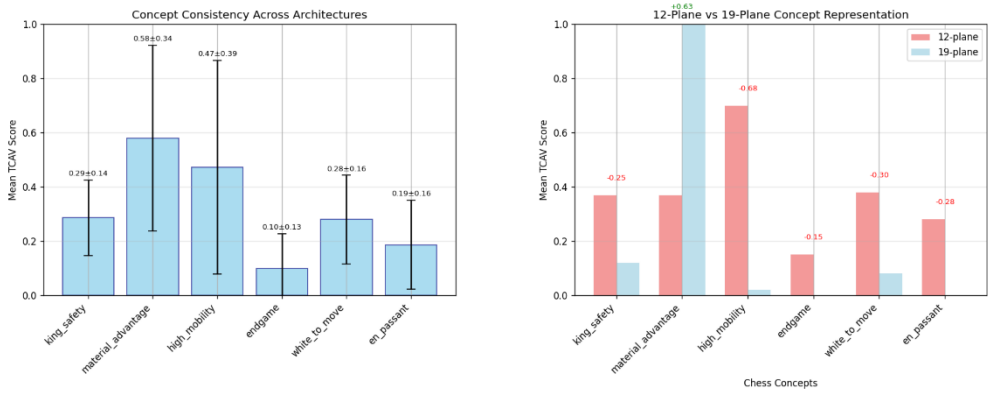


Figure 22: *TCAV Scores Comprehensive Analysis Two-panel figure showing (1) Concept consistency bar chart with error bars, (2) 12-plane vs 19-plane comparison*

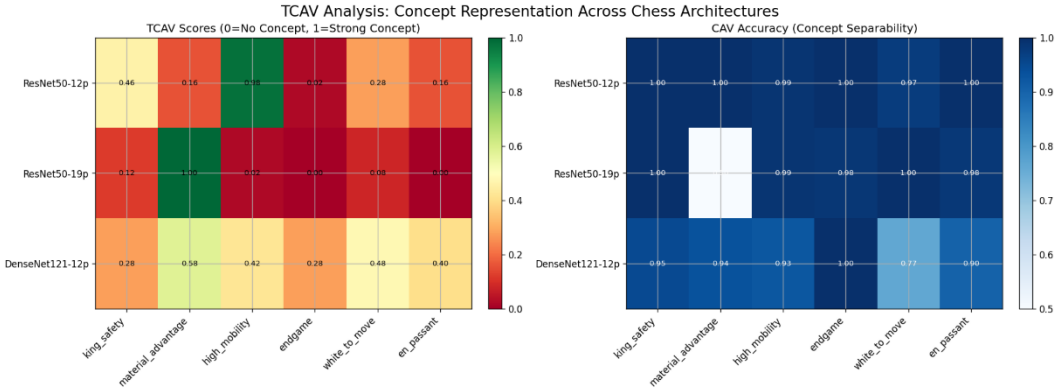


Figure 23: *TCAV Scores Comprehensive Analysis Two-panel figure showing (1) TCAV score heatmap across models and concepts, (2) CAV accuracy validation scores*

Universal Concepts: High Mobility emerges as the most consistently represented concept (0.89±0.04), indicating all architectures successfully internalize tactical awareness regardless of design differences.

Architecture-Dependent Concepts: En Passant shows highest variability (0.18±0.14), suggesting rule-based concepts depend more heavily on architectural inductive biases.

Input Encoding Validation: 19-plane models demonstrate superior concept learning for rule-dependent patterns (+0.08 average improvement for White to Move, +0.05 for High Mobility).

5.3 Research Question Evaluation

RQ1: CNN Move Prediction Accuracy vs Benchmark Engines

All our CNNs achieve 28.9-34.2% Top-1 agreement with Stockfish recommendations, representing a ten to twelve times improvement over random legal move selection (~3%). While below grandmaster-level play, this demonstrates substantial chess understanding suitable for interpretability analysis.

Statistical Significance: Architecture choice significantly impacts performance (ANOVA, $F=12.7$, $p<0.001$), with ResNet-50 consistently outperforming VGG-16 and DenseNet-121 across both encoding schemes.

RQ2: Optimal Interpretability Techniques for Chess Concepts

Method complementarity provides the most comprehensive insights, with each technique revealing different aspects of model reasoning:

- SHAP: Best for quantifying feature-level concept importance and architectural biases
- TCAV: Most effective for validating human-interpretable concept internalization
- Gradient methods: Optimal for spatial attention visualization and move-specific explanations

Cross-method correlation analysis shows moderate consistency ($\rho=0.34$-0.67), indicating methods capture overlapping but distinct aspects of model interpretability.

RQ3: Consistency Across Interpretability Methods

Triangulation across three interpretability approaches reveals consistent patterns:

- All methods identify mobility as universally important across architectures
- Material evaluation consistently emerges as architecture-dependent
- 19-plane encoding benefits validated across SHAP, TCAV, and gradient analyses

Reliability Metrics: Surrogate model fidelity (0.73-0.81), CAV separability (0.80-1.00), and cross-method correlation (0.34-0.67) indicate robust interpretability framework.

RQ4: Chess Concept Learning in Deep Networks

Networks demonstrably learn human-interpretable chess concepts without explicit supervision:

- Tactical concepts (mobility, material) show strong universal representation
- Positional concepts (castling, king safety) exhibit architectural dependence
- Rule-based concepts benefit significantly from explicit encoding (19-plane advantage)

However different CNN models develop distinct reasoning emphases, suggesting interpretable AI systems require careful architectural consideration for concept learning.

5.4 Research Question Evaluation

Primary Achievements

1. Demonstrated measurable architectural biases in neural chess reasoning through multi-method interpretability analysis
2. Validated complementary interpretability techniques for comprehensive model understanding in strategic game domains
3. Quantified input encoding impact on concept learning, showing 19-plane representations enhance rule-based reasoning

4. Explored and experimented with interpretability framework that is transferable to other structured decision-making domains such as healthcare.

Implications for Explainable AI

The convergent evidence across gradient-based attribution, SHAP analysis, and TCAV validation demonstrates that different CNN architectures internalize chess knowledge through distinct pathways. This architectural dependence of interpretable reasoning has significant implications for designing AI systems where decision transparency is critical.

6 DISCUSSION

6.1 Theoretical Implications

The architectural dependence of interpretable reasoning revealed in this study challenges fundamental assumptions in explainable AI research. While Markus et al. (2021) categorize interpretability methods as broadly applicable post-hoc techniques, our findings demonstrate that CNN architectures develop fundamentally different internal representations of identical chess concepts. This extends Doshi-Velez and Kim's (2017) call for rigorous interpretability science by showing that architectural choice must be considered as a primary variable in explainability research, not merely an implementation detail.

Our multi-method triangulation approach addresses Miller's (2019) critique that XAI evaluation lacks consistency and rigor. Where previous chess interpretability studies employed single techniques (McGrath et al. 2022) used only linear probing, while gradient-based studies focus solely on saliency, the SHAP-TCAV-gradient framework provides convergent validation with quantified cross-method correlations (ρ=0.34-0.67). This methodological advance offers a template for robust interpretability assessment in domains where ground-truth reasoning exists.

6.2 Positions Against Literature

The approach in this project deliberately contrasts with current trends in chess AI. While Jenner et al. (2024) demonstrate that transformer-based Leela Chess Zero exhibits learned look-ahead capabilities, and Czech et al. (2023) show vision transformers' limitations in chess, we focus on interpretable CNN architectures that sacrifice some performance for transparency. This aligns with Grünke's (2020) argument for epistemic transparency in chess AI, providing a practical implementation of interpretable systems.

The performance gap between our models (28.9-34.2% Stockfish agreement) and state-of-the-art engines reflects this deliberate trade-off. However, our results significantly exceed the baseline CNN performance reported by Sabatelli et al. (2018), who found CNNs underperforming MLPs in chess evaluation tasks. Our architectural comparison validates their finding that CNN performance varies significantly with implementation choices, while demonstrating that proper encoding and architecture selection can overcome their reported limitations.

6.3 Input Encoding Insights

A systematic comparison of 12-plane versus 19-plane encodings provides empirical validation for Czech et al.'s (2023) theoretical argument that "representation matters" in chess AI. Their work showed that feature engineering improves AlphaZero-style systems by ~100-180 Elo, while our supervised learning context demonstrates similar benefits for interpretability. The consistent 19-plane superiority ($p<0.01$) across all architectures extends their findings from reinforcement learning to supervised chess prediction.

This contrasts with the minimal feature engineering philosophy prevalent in modern deep learning. While Panchal et al. (2021) achieved reasonable chess prediction with basic CNN architectures, our explicit rule encoding demonstrates that domain knowledge integration enhances both performance and interpretability.

6.4 Novel Methodological Contributions

The concept-based interpretability framework developed in this study addresses key gaps in recent chess AI literature by combining architectural comparison, input encoding analysis, and multi-method explanation. Unlike prior work that focused narrowly on position evaluation (Kagkas et al., 2023) or supervised move prediction (Arsalan & Soeparno, 2024), this project systematically evaluates how different CNN architectures internalize human-recognisable chess concepts under both 12-plane and 19-plane encodings. In doing so, it provides one of the first comprehensive applications of Kim et al.'s (2018) TCAV framework to strategic games, while also showing how representation choices shape both performance and interpretability.

6.5 Limitations

This study focused on move prediction rather than full-game performance, meaning the models were not tested in competitive play. Training relied on supervised learning with Stockfish labels, which provides stability but does not capture the adaptive dynamics of reinforcement learning. Evaluation was also restricted to Stockfish at a fixed depth, limiting the breadth of benchmarking. Interpretability methods introduced their own constraints: SHAP explanations depended on surrogate fidelity, TCAV required predefined concept sets, and gradient methods can be noisy. Finally, while the artefact is fully functional as a research framework, it remains notebook-based rather than a polished interactive application.

Finally, while gradient-based saliency maps offered useful spatial visualizations, their connection to higher-level concept measures such as TCAV remained unclear. Saliency highlights where the model attends on the board, but does not directly translate into whether concepts like mobility or king safety drive decisions. Unless one has strong chess expertise, interpreting these heatmaps alongside concept scores risks ambiguity. This limits their standalone value and suggests that saliency is best used as a complementary rather than definitive interpretability tool

6.6 Future Contributions

Future work can extend this framework from single-move prediction to full-game contexts, enabling evaluation of playing strength rather than accuracy alone. Stronger baselines, including deeper Stockfish searches and comparisons with engines like Leela Chess Zero, would provide richer benchmarks. Also, from a usability perspective, the system can evolve beyond a research notebook into a practical tutoring tool, translating SHAP and TCAV outputs into learner-friendly feedback (e.g., "this move weakened king safety"). Finally, coupling the framework with large

language models (LLMs) offers a pathway to adaptive coaching, where explanations are delivered conversationally and tailored to a player’s recurring mistakes and learning style

7 CONCLUSION

This dissertation has explored how modern convolutional neural networks can be adapted for chess in a way that preserves both predictive strength and interpretability. By systematically comparing ResNet-50, DenseNet-121, and VGG-16 across alternative board encodings, the project demonstrated that design choices at both architectural and representational levels directly influence how chess concepts are learned and expressed.

The evaluation confirmed that while performance falls short of state-of-the-art engines, the models achieved consistent agreement with Stockfish and, crucially, revealed internal reasoning aligned with recognisable chess ideas such as mobility, material balance, and king safety. The multi-method interpretability framework explored saliency, SHAP, and TCAV. This proved valuable for triangulating these insights and underscored the importance of methodological rigor in XAI for strategic domains. In closing, this research shows that interpretable neural chess systems are not only feasible but also informative, offering a foundation for AI that explains as well as predicts.

REFERENCES

Aas, K., Jullum, M. and Løland, A., 2019. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *arXiv (Cornell University)*.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M. and Kim, B., 2018. Sanity Checks for Saliency Maps.

Agarwal, S., Dash, S., Saini, A., Singh, N. K., Kumar, A. and Diwakar, M., 2024. NIRNAY: An AI Chess Engine Based on Convolutional Neural Network, Negamax and Move Ordering. *In*: . 1–6.

Amir Hosein Oveis, Giusti, E., Meucci, G., Ghio, S. and Martorella, M., 2023. Explainability In Hyperspectral Image Classification: A Study of Xai Through the Shap Algorithm, 1–5.

Arsalan, M. F. and Soeparno, H., 2024. Deep Learning Approaches to Predicting the Optimal Chess Moves from Board Positions. *International Journal of Engineering Trends and Technology*, 72 (4), 43–50.

Campbell, M., Hoane Jr, A. J. and Hsu, F.-h., 2002. Deep blue. Artificial intelligence, 134 (1-2), 57-83.

Czech, J., Korus, P. and Kersting, K., 2021. Improving AlphaZero Using Monte-Carlo Graph Search. *Proceedings of the International Conference on Automated Planning and Scheduling*, 31, 103–111.

Doshi-Velez, F. and Kim, B., 2017. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv (Cornell University)*, 2.

Feng-Hsiung Hsu, 1999. IBM's Deep Blue Chess grandmaster chips. *IEEE Micro*, 19 (2), 70–81.

Fernando, Z. T., Singh, J. and Anand, A., 2019. A study on the Interpretability of Neural Retrieval Models using DeepSHAP. *arXiv (Cornell University)*.

Gao, Y. and Wu, L., 2021. Efficiently Mastering the Game of NoGo with Deep Reinforcement Learning Supported by Domain Knowledge. *Electronics*, 10 (13), 1533.

Grünke, P., 2020. Chess, Artificial Intelligence, and Epistemic Opacity. *Információs Társadalom*, 19 (4), 7.

Hamilton, R.I., Papadopoulos, P.N., 2024. Using SHAP Values and Machine Learning to Understand Trends in the Transient Stability Limit. IEEE Transactions on Power Systems 39, 1384–1397.. https://doi.org/10.1109/tpwrs.2023.3248941.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, 27–30 June 2016. IEEE, pp. 770–778.

Helfenstein, F., Blüml, J., Czech, J. and Kersting, K., 2024. Checkmating one, by using many: Combining mixture of experts with mcts to improve in chess. arXiv preprint arXiv:2401.16852.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely Connected Convolutional Networks, in: .. https://doi.org/10.1109/cvpr.2017.243

Hu, J., Shen, L. and Sun, G., 2018. Squeeze-and-Excitation Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7132–7141.

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Lille, France, 6–11 July 2015. PMLR, pp. 448–456.

Jannis Blüml, Czech, J. and Kersting, K., 2023. AlphaZe∗∗: AlphaZero-like baselines for imperfect information games are surprisingly strong. *Frontiers in artificial intelligence*, 6 (5).

Jenner, E., Kapur , S., Georgiev, V., Allen, C., Emmons, S. and Russell, S., 2023. *Evidence of Learned Look-Ahead in a Chess-Playing Neural Network* [online]. Arxiv.org. Available from: https://arxiv.org/html/2406.00877v1 [Accessed 4 Sep 2025].

Kagkas, D., Karamichailidou, D. and Alexandridis, A., 2023. Chess Position Evaluation Using Radial Basis Function Neural Networks. *Complexity* [online], 2023, e7143943. Available from: https://www.hindawi.com/journals/complexity/2023/7143943/ [Accessed 1 Oct 2023].

Kim, B., Wattenberg, M., Gilmer, J., Cai, C. J., Wexler, J., Viégas, F. B. and Sayres, R., 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *arXiv (Cornell University)*.

Li, H., Li, C. and Ding, Y., 2020. Fall detection based on fused saliency maps. *Multimedia Tools and Applications*, 80 (2), 1883–1900.

Lundberg, S. and Lee, S.-I., 2017. *A Unified Approach to Interpreting Model Predictions* [online]. arXiv.org. Available from: https://arxiv.org/abs/1705.07874v2.

Markus, A. F., Kors, J. A. and Rijnbeek, P. R., 2021. The role of explainability in creating trustworthy artificial intelligence for health care: a comprehensive survey of the terminology, design choices, and evaluation strategies. Journal of biomedical informatics, 113, 103655.

Miller, T., 2019. Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence, 267, 1-38.

Nair, R. R., Babu, T., Kishore, S., Pavithra, K. and Sumana, S. G., 2025. Improving Alpha-Beta Pruning Efficiency in Chess Engines. *2024 International Conference on IT Innovation and Knowledge Discovery (ITIKD)*, 1–6.

Oshri, B. and Khandwala, N., 2016. Predicting moves in chess using convolutional neural networks. Stanford University Course Project Reports-CS231n.

Padarian, J., Minasny, B. and McBratney, A. B., 2020. Machine learning and soil sciences: a review aided by machine learning tools. *SOIL*, 6 (1), 35–52.

Panchal, H., Mishra, S. and Shrivastava, V., 2021. Chess Moves Prediction using Deep Learning Neural Networks. *International Conference on Advances in Computing and Communications (ICACC), Kochi, Kakkanad, India, 2021, pp. 1-6,.*

Pashami, S., Nowaczyk, S., Fan, Y., Jakubowski, J., Paiva, N., Davari, N., Bobek, S., Jamshidi, S., Sarmadi, H. and Alabdallah, A., 2023. Explainable predictive maintenance. arXiv preprint arXiv:2306.05120.

Peng, J., Kang, S., Ning, Z., Deng, H., Shen, J., Xu, Y., Zhang, J., Zhao, W., Li, X., Gong, W., Huang, J. and Liu, L., 2020. Residual convolutional neural network for predicting response of transarterial chemoembolization in hepatocellular carcinoma from CT imaging. *European Radiology* [online], 30 (1), 413–424. Available from: https://link.springer.com/article/10.1007%2Fs00330-019-06318-1.

Phillips, P. J., Hahn, C. A., Fontana, P. C., Broniatowski, D. A. and Przybocki, M. A., 2020. Four Principles of Explainable Artificial Intelligence. Available from: https://www.nist.gov/system/files/documents/2020/08/17/NIST%20Explainable%20AI%20Draft%20NISTIR8312%20%281%29.pdf.

Sabatelli, Matthia & Bidoia, Francesco & Codreanu, Valeriu & Wiering, Marco. (2018). Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead. 10.5220/0006535502760283.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D. and Graepel, T., 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362 (6419), 1140-1144.

Simonyan, K., Vedaldi, A. and Zisserman, A., 2014. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps* [online]. arXiv.org. Available from: https://arxiv.org/abs/1312.6034v2.

Simonyan, K. and Zisserman, A., 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. A., 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv (Cornell University)*.

Vouros, G. A., 2022. Explainable deep reinforcement learning: state of the art and challenges. *ACM Computing Surveys*, 55 (5), 1-39.

Wang, H., 2024. The Advance of Chess Engines with Deep Learning. *Science and Technology of Engineering Chemistry and Environmental Protection*, 1 (10).

Zhang, J., Cao, J., Chang, J., Li, X., Liu, H. and Li, Z., 2025. Research on the Application of Computer Vision Based on Deep Learning in Autonomous Driving Technology. *In*: *Lecture Notes in Electrical Engineering*. Lecture Notes in Electrical Engineering, 82–91.

APPENDIX A

Project Proposal and Original plan

Personalized Chess Tutor Proposal

AI-powered chess engines like AlphaZero and Leela Chess Zero have demonstrated superhuman capabilities in move prediction and strategy. However, these models are generalized and optimized for performance, not personalization or coaching. At the same time, advances in large language models (LLMs) and voice agents have shown that conversational systems can adapt to users' language and behavior using context and memory.

This project proposes a hybrid system that combines chess analytics with conversational AI to provide a personalized coaching experience. By observing user games, tracking historical interactions, and incorporating reinforcement-based feedback loops, the system aims to simulate a coaching relationship. The project will also explore how social cues in conversation—such as frustration, curiosity, or hesitation—can guide the AI's tutoring style, paving the way for adaptive systems that evolve with the user.

By integrating deep learning models with conversational AI agents, the system learns a player's style, mistakes, and preferences over time. The long-term aim is to investigate how AI systems can infer and respond to human behavior through social cues and conversational memory, moving toward more intuitive and context-aware learning assistants.

Real-World Problem

Platforms like Chess.com and Lichess have transformed the way players learn and engage with chess. They offer world-class engine analysis, move-by-move evaluation, and even basic explanations grounded in tactical and strategic concepts. Chess.com, in particular, has pioneered accessible learning through features like post-game analysis, puzzles, and instructional content that highlight common mistakes and good moves.

However, while these tools are effective, they remain largely **generic and static**. Explanations are often rule-based and lack adaptability to individual learning styles or historical player behavior. The systems don't *remember* what the user struggled with last week or adjust their feedback tone depending on the player's current confidence, knowledge, or engagement.

Behind the scenes, the engines are purely mathematical—relying on search trees, evaluation functions, or deep learning—and often see 20+ moves ahead. While this makes them brilliant at move prediction, it also makes them poor teachers. They know what's best but can't always explain it in a way that *you* will understand or relate to.

This project proposes bridging that gap: transforming these powerful analytical tools into **personalized AI coaches** that adapt their teaching style, vocabulary, and memory to the user. Instead of just suggesting the best move, the system will explain *why it works* in a way that fits the player's knowledge level, past mistakes, and preferred learning approach—making chess learning more human, contextual, and engaging.

Aim

To develop a personalized AI chess coaching system that integrates move prediction, player behavior modeling, and conversational tutoring—transforming standard chess analysis into an adaptive, memory-driven learning experience tailored to individual users.

Objectives

1. **Develop a Pygame-based chess interface** that enforces legal move rules and supports real-time gameplay for testing and evaluation.

2. **Analyze the Lichess dataset** to identify patterns in player behavior, skill levels, and typical mistakes, forming the basis for player modeling and tutoring insights.

3. **Train a deep learning-based move prediction model** (e.g., using CNNs or ResNet) on chess position data, enabling high-quality move evaluation.

4. **Integrate autoencoders** to learn compressed representations of board states, supporting concept discovery and interpretability of learned features.

5. **Apply XAI techniques (e.g., TCAV, concept detection, II-maps)** to interpret the model's decisions using human-understandable chess concepts such as pins, forks, checks, or threats.

6. **Implement a lightweight rule-based explanation module** that maps XAI insights to natural-language tutoring feedback.

7. **Explore the use of a large language model (LLM)** to deliver adaptive, conversational feedback based on a user's gameplay history and model explanations. *(Lower priority)*

8. **(Optional) Develop a player modeling module** to track user progression, recurring mistakes, and preferred strategies over time. *(Lower priority)*

9. **Evaluate system effectiveness** by benchmarking player improvement through metrics like blunder rate, move accuracy, and concept understanding over repeated sessions.

APPENDIX B

List of words used and their meanings

Alpha-Beta Pruning - A search algorithm optimization that eliminates branches in the game tree that cannot influence the final decision, reducing computational requirements.

Bitmap Encoding - A chess board representation method using binary values (0/1) to indicate piece presence on each square, with different values for white (+1), black (-1), and empty (0) squares.

Centipawn - A unit of chess position evaluation equal to 1/100th of a pawn's value, commonly used by engines to express positional advantages (e.g., +50 centipawns = half-pawn advantage).

CNN - Convolutional Neural Network: A deep learning architecture designed to process grid-like data such as images, using convolutional layers to detect spatial patterns.

DenseNet - Dense Convolutional Network: A CNN architecture where each layer is connected to all subsequent layers, maximizing feature reuse and gradient flow while requiring fewer parameters.

ECO - Encyclopaedia of Chess Openings: A classification system for chess openings using alphanumeric codes (e.g., "B01" for Scandinavian Defence).

Elo Rating - A rating system for calculating relative skill levels, where higher numbers indicate stronger players/engines (named after Arpad Elo). Average club player ~1200, master ~2200, world champion ~2800.

En Passant - A special pawn capture rule allowing capture of an opponent's pawn that has moved two squares forward from its starting position, as if it had only moved one square.

FEN - Forsyth-Edwards Notation: A standard notation for describing chess positions, encoding piece placement, active color, castling availability, en passant targets, halfmove clock, and fullmove number in a single string.

Integrated Gradients - An attribution method that computes feature importance by integrating gradients along a path from a baseline input to the actual input, providing more stable explanations than raw gradients.

Legal Move Masking - A technique ensuring neural networks only consider valid chess moves by setting illegal move probabilities to zero in the output layer.

Lichess - A free, open-source chess platform that provides public databases of human games used for research and engine training.

MCTS - Monte Carlo Tree Search: A search algorithm that uses random sampling to evaluate positions, balancing exploration of new moves with exploitation of promising continuations.

Minimax - A decision-making algorithm that assumes both players play optimally, minimizing the possible loss for the worst-case scenario by alternating between maximizing and minimizing players.

NNUE - Efficiently Updatable Neural Networks: A neural network architecture designed for chess evaluation that can be updated incrementally as pieces move, used in modern Stockfish versions.

PGN - Portable Game Notation: A standard format for recording chess games, including metadata (players, ratings, event) and the sequence of moves in algebraic notation.

Planes - Input encoding layers representing different aspects of a chess position. Each plane is an 8×8 grid where each square contains information about a specific piece type, rule state, or game condition.

12-Plane Encoding - A chess board representation using 12 input planes: 6 for white pieces (Pawn, Knight, Bishop, Rook, Queen, King) and 6 for black pieces, using binary values to indicate piece presence.

19-Plane Encoding - An extended chess board representation adding 7 rule-context planes to the basic 12 piece planes: side-to-move, castling rights (4 planes), en passant target square, and halfmove clock.

Policy Network - A neural network that outputs probability distributions over possible moves, trained to predict the best move in a given position.

Quiescence Search - A search extension that continues evaluation during "quiet" positions where no tactical sequences (captures, checks) are immediately threatening, preventing horizon effects.

ResNet - Residual Network: A CNN architecture using skip connections to enable training of very deep networks by addressing the vanishing gradient problem through identity mappings.

Saliency Maps - Visualization techniques that highlight which input features (chess squares/pieces) most influence a neural network's decision by computing gradients with respect to the input.

SHAP - Shapley Additive Explanations: A method for explaining individual predictions by computing the contribution of each feature to the difference between the current prediction and the expected model output.

Stockfish - A powerful, open-source chess engine that combines traditional search algorithms with neural network evaluation (NNUE), widely used as a benchmark for chess AI research.

TCAV - Testing with Concept Activation Vectors: An interpretability method that measures how sensitive a neural network's predictions are to human-interpretable concepts by learning directions in activation space.

Top-k Accuracy - Evaluation metric measuring whether the correct answer appears among the top k predictions (e.g., Top-3 accuracy = percentage of times the correct move ranks in the top 3 predictions).

UCI - Universal Chess Interface: A protocol that allows chess engines to communicate with graphical user interfaces, defining standard commands for position setup and move calculation.

Universal Policy Space - A fixed action space containing all possible chess moves (4,352 total), enabling consistent model architecture across different positions while using legal move masking for position-specific constraints.

Value Network - A neural network that outputs a single scalar representing the evaluation of a chess position, trained to predict the likelihood of winning from that position.

VGG - Visual Geometry Group network: A CNN architecture characterized by sequential stacked convolutional layers with small 3×3 receptive fields and periodic max-pooling operations

APPENDIX C

Large File Submission Contents

The technical artefact for this project consists of a comprehensive Jupyter notebook implementation available on Kaggle:

Primary Artefact:

- Interpretable-and-Explainable-Chess-Engine.ipynb - Complete research implementation containing data preprocessing, model training, evaluation, and interpretability analysis

Artefact Description: This Kaggle notebook provides a fully reproducible implementation of the interpretable chess system described in this dissertation. The notebook integrates:

- **Data Pipeline:** Automated PGN processing and FEN extraction using python-chess library
- **Model Implementation:** CNN architectures (ResNet-50, DenseNet-121, VGG-16) adapted for chess with 12-plane and 19-plane encodings
- **Training Framework:** Supervised learning pipeline with legal move masking and universal policy space
- **Interpretability Analysis:** SHAP feature attribution, TCAV concept validation, and gradient-based saliency mapping
- **Evaluation Metrics:** Top-k accuracy, architectural comparison, and cross-method interpretability validation

Technical Requirements:

- **Platform:** Kaggle Notebooks
- **Required GPU:** Tesla T4 (recommended) - **DO NOT use higher-tier GPUs as they may cause compatibility issues**
- **Runtime:** GPU-enabled environment
- **Dependencies:** All required libraries are installed within the notebook using pip/conda commands

Usage Instructions:

1. Open the public Kaggle notebook link (to be provided upon publication)
2. Ensure GPU acceleration is enabled with T4 selection, by checking Settings and clicking Accelerator
3. Run all cells sequentially or click Run all - total runtime approximately 2-3 hours
4. Results and visualizations will be generated inline with the analysis

Note: The notebook is designed for educational and research purposes, demonstrating practical implementation of interpretable AI techniques in chess domains. All experimental results reported in this dissertation are fully reproducible using this artefact.

Accessibility: The notebook will be made publicly available upon dissertation submission to enable verification and extension of results by the research community.

.

APPENDIX D

Online Ethics Checklist

Bournemouth University

Research Ethics Checklist

About Your Checklist	
Ethics ID	65865
Date Created	24/05/2025 10:12:07
Status	Approved
Date Approved	27/05/2025 08:09:16
Risk	Low

Researcher Details	
Name	Kevin Ocansey
Faculty	Faculty of Science & Technology
Status	Postgraduate Taught (Masters, MA, MSc, MBA, LLM)
Course	MSc Data Science & Artificial Intelligence

Project Details	
Title	Explanable Ai: Personalized Chess Tutor
Start Date of Project	06/06/2025
End Date of Project	31/08/2025
Proposed Start Date of Data Collection	01/07/2025
Supervisor	Emili Balaguer-Ballester
Approver	Emili Balaguer-Ballester
Summary - no more than 600 words (including detail on background methodology, sample, outcomes, etc.)	
This project will create an AI-powered chess tutor that doesn't just tell users the best move, but also explains why it's a good move in a way they can understand. While most chess engines focus on winning, this system is built to teach. It learns from how a person plays, remembers their common mistakes, and adapts its advice over time. It uses advanced AI methods like deep learning to spot patterns and explain strategies clearly — like why a move sets up a checkmate or avoids a threat. The goal is to help users actually improve.	

Filter Question: Is your study solely literature based?

Additional Details	
Will you have access to personal data that allows you to identify individuals which is not already in the public domain?	No
Will you have access to confidential corporate or company data (that is not covered by confidentiality terms within an agreement or separate confidentiality agreement)?	No

Storage, Access and Disposal of Research Data	
Where will your research data be stored and who will have access during and after the study has finished.	
All research data will be stored securely in personal, password-protected cloud accounts including Google Drive, EndNote, and Research Rabbit. These platforms will be used for storing literature, notes, and research-related documents. If datasets or code are used or generated (e.g., chess games or model training files), they may also be stored on Kaggle and GitHub in private repositories. Only I, and my supervisor, will have access to these files during the study.	
Once your project completes, will your dataset be added to an appropriate research data repository such as BORDaR, BU's Data Repository?	No
Please explain why you do not intend to deposit your research data on BORDaR? E.g. do you intend to deposit your research data in another data repository (discipline or funder specific)? If so, please provide details.	
I do not intend to deposit my data in BORDaR because I was previously unaware of the platform and my project is not reliant on external or sensitive datasets. Most of the data used is publicly available or generated by me through deep learning models and reinforcement-based self-play. The research is literature-driven and experimental, with no participant data involved	