Comparative Analysis of Transformer and LSTM Architectures for Chess Move Prediction

Name: Sanjana S. Joshi
Roll No: MS2405

October 2025

1. Introduction

Deep learning models like LSTM networks [3] and Transformer architectures [5] are widely used for sequence prediction. LSTMs handle sequences using recurrence, while Transformers use self-attention to model long-range dependencies.

Chess games are sequential and strategic, making them a good testbed for these models. Each move depends on previous moves, both near and far in the game. This project compares LSTM and Transformer models for predicting the next chess move from historical game data.

2. Problem Statement

The task is to predict the next move in a chess game based on previous moves.

- **Input:** Sequence of past moves (e.g., e4, e5, Nf3, Nc6).
- **Output:** The predicted next move (e.g., Bb5).

Formally, given a sequence $X = [x_1, x_2, ..., x_T]$, the model learns $f : X \rightarrow x_{T+1}$.

3. Motivation

Chess engines like Stockfish [4] use search-based methods. Deep learning models can learn patterns from human games. Comparing LSTM and Transformer models shows:

- If attention-based models capture long-range strategies better.
- How sequence models perform on real-world structured data.
- Insights into interpretability of predictions.

4. Goals

1. Implement a Transformer from scratch for next-move prediction.
2. Use a prebuilt LSTM model as a baseline.
3. Train both models on a subset of Lichess PGN data (50k - 100k games).
4. Evaluate models using accuracy and Stockfish evaluation of moves.
5. Visualize attention weights in the Transformer to interpret predictions.

5. Methodology

5.1 Dataset

The project will use publicly available chess games in PGN format:

- Lichess Open Database [1].
- A subset of 50k–100k games will be selected for training and evaluation.
- Games can be filtered by time control (e.g., classical) or player rating (e.g., $>$ 1800 Elo) to improve data quality.

5.2 Preprocessing

- Parse PGN files using the `python-chess` library [2].
- Extract the mainline moves and convert them to Standard Algebraic Notation (SAN).
- Build a move vocabulary mapping each unique move to an integer token.
- Pad or truncate sequences to a fixed length for batching.
- Split the dataset into training (80%), validation (10%), and test (10%) sets.

5.3 Model Implementation

- **Transformer:** Implement from scratch with token embeddings, positional encodings, multi-head attention, feedforward layers, layer normalization, and residual connections. Use causal masking for autoregressive next-move prediction.
- **LSTM:** Use a standard stacked LSTM (1–2 layers) from PyTorch or Keras as a baseline, with an embedding layer and a dense softmax over the move vocabulary.

5.4 Training

- Objective: Cross-entropy loss for next-move prediction.
- Optimizer: Adam or AdamW with a learning-rate schedule.
- Batch size: 32–128 depending on GPU memory.
- Early stopping based on validation loss to avoid overfitting.
- Train for 5–12 epochs until convergence.

5.5 Evaluation

- Metrics: Top-1 accuracy.
- Chess-specific evaluation: Use Stockfish [4] to compare predicted moves with the best moves suggested by the engine.
- Compare Transformer and LSTM performance quantitatively and qualitatively.

5.6 Visualization

- Plot attention heatmaps to visualize which past moves the Transformer focuses on for prediction.
- Display example sequences of predicted moves to observe model behavior.

6. Expected Outcomes

- A functional Transformer model implemented from scratch.
- Comparison of Transformer and LSTM performance on chess move prediction.
- Visualizations showing which past moves the Transformer focuses on.
- Insights about sequence modeling in strategic games.

References

[1] Lichess open database of chess games. `https://database.lichess.org/`. Accessed: 2025-10-12.

[2] python-chess: A chess library for python. `https://python-chess.readthedocs.io/`. Accessed: 2025-10-12.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[4] Stockfish Developers. Stockfish: Open-source chess engine. `https://stockfishchess.org/`. Accessed: 2025-10-12.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.