Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead

The 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM)

M. Sabatelli[1,2], F. Bidoia[1], V. Codreanu[3], **M.A. Wiering**[1]

[1]Institute of Artificial Intelligence and Cognitive Engineering, University of Groningen, The Netherlands

[2]Montefiore Institute, Department of Electrical Engineering and Computer Science, Université de Liège, Belgium

[3]Surfsara BV, Amsterdam, The Netherlands

Overview

Introduction

Introduction

- Human chess Grandmasters are able to evaluate chess positions precisely without having to rely on very deep calculations
- They manage to do so thanks to their pattern recognition abilities that allow them to understand a large variety of positions very quickly
- We show how to train different Artificial Neural Networks (ANNs) to replicate this skill by proposing a **novel supervised learning** training procedure that allows the ANNs to play high level chess without having to rely on any lookahead algorithms.
- We both investigate Multilayer Perceptrons (MLPs) as Convolutional Neural Networks (CNNs)

Main Contributions

- First attempt to learn to play chess by discarding lookahead algorithms
- One of the first papers exploring the use of CNNs in chess
- Extension of the Kaufman test (Δcp) for assessing the strength of chess programs that do not look ahead more than one move
- Creation of 4 open-source different Datasets that can be used for the pre-training stage in Reinforcement Learning approaches [1]

[1] https://github.com/paintception/DeepChess

State of the Art

State of the Art

Despite not being the first ones approaching the game of chess with ANNs this is the first attempt of training a system that aims to play chess without using lookahead

- KnightCap (Baxter, 1999): Alpha-Beta Pruning and TD-Learning
- Giraffe (Lai, 2015): Probability Limited MinMax and TD-Learning
- AlphaZero (Silver, 2017): MCTS in combination with Reinforcement Learning

Methods

Methods

- We have downloaded $\approx 3{,}000{,}000$ different chess positions from games played by highly skilled chess players
- We assign an evaluation to each position according to the evaluation function of *Stockfish*: one of the strongest chess engines
- The evaluation is expressed with the *cp* measurement, a numerical value corresponding to $1/100th$ of a pawn: the higher this value is, the higher the winning chances for *White* are (and vice-versa)
- According to the value of this *cp* measurement we have created 4 different *Datasets*

Methods

We assign a new label φ when the following conditions are satisfied

- **Dataset 1**
  - L_{φ}: cp < -1.5
  - D_{φ}: $-1.5 \leq$ cp ≤ 1.5
  - W_{φ}: cp > 1.5

In total we have **3** different labels that correspond to the potential outcomes of the game

Methods

We assign a new label to the **Winning** and **Losing** classes every time the cp evaluation increases by **1** until the following conditions are satisfied

- **Dataset 2**
  - $L_{\varphi+1}$:$-8.5 \leq$ cp < -1.5
  - Dφ: $-1.5 \leq$ cp ≤ 1.5
  - $W_{\varphi+1}$: $1.5 <$ cp ≤ 8.5

In total we obtain **15** different labels that should maximize/minimize the chances of Winning/Losing

Methods

We assign a different **Draw** label each time the cp evaluation increases by **0.5**

- **Dataset 3**
  - $L_{\varphi+1}$: $-8.5 \leq \text{cp} < -1.5$
  - $D_{\varphi+0.5}$: $-1.5 \leq \text{cp} \leq 1.5$
  - $W_{\varphi+1}$: $1.5 < \text{cp} \leq 8.5$

In total we obtain **20** different labels that should maximize the chances of Winning even in apparently *Draw* positions.

Methods

- **Dataset 4** We do not make use of any **categorical labels** but try to approximate *Stockfish's* evaluation function as close as possible by treating this as a **regression problem**.

Loss functions for the Experiments

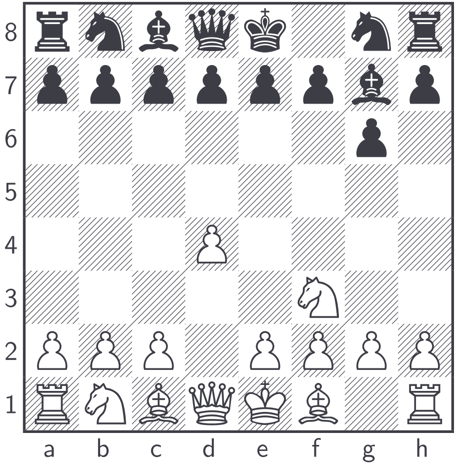
- Datasets 1, 2 and 3:

$$L(X, Y) = -\frac{1}{n} \sum_{t=1}^{n} \sum_{i=1}^{C} y_t^i \log f_\theta^i(x_t)$$

- Dataset 4:

$$MSE = \frac{1}{n} \sum_{t=1}^{n} (y_t - f_\theta(x_t))^2$$

Chessboard Representations



- **Bitmap Input**:
  - $[0, 0, 0, 0, 0, 0, 0, 0]$
  - $[1, 1, 1, 0, 1, 1, 1, 1]$
  - $\ldots$
  - $[0, 0, 0, 0, 0, 1, 0, 0,]$
  - $\ldots$
- **Algebraic Input**:
  - $[0, 0, 0, 0, 0, 0, 0, 0]$
  - $\ldots$
  - $\ldots$
  - $[0, 0, 0, 0, 0, 0, -3, 0]$
- 768 Inputs for the MLP
- $8 \times 8 \times 12$ tensor for the CNN

Artificial Neural Network Architectures

Multi-layer Perceptron:

- For dataset 1, we used a three hidden layer deep MLP with 1048, 500 and 50 hidden units
- For datasets 2 and 3, the MLP consists of 2048, 2048, 1050 hidden units
- For dataset 4, the MLP uses three layers with 2048 hidden units

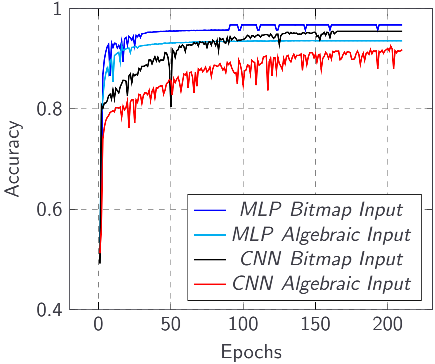
Convolutional Neural Network:

- The CNN consists of two *2D* convolution layers followed by a final fully connected layer of 500 units.
- The first convolution layer uses 20 5×5 filters. The second convolution layer uses 50 3×3 filters

Results

Results

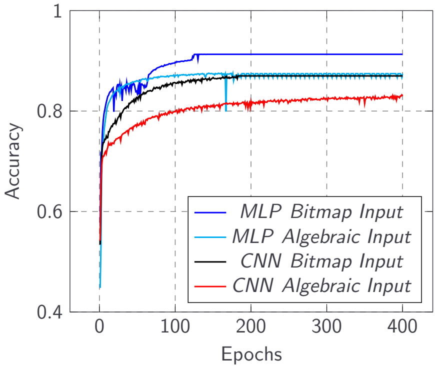
Dataset 1



ANN	ValSet	TestSet
MLP Bit	98.67%	**96.07%**
MLP Alg	96.95%	93.58%
CNN Bit	95.40%	95.15%
CNN Alg	91.70%	90.33%

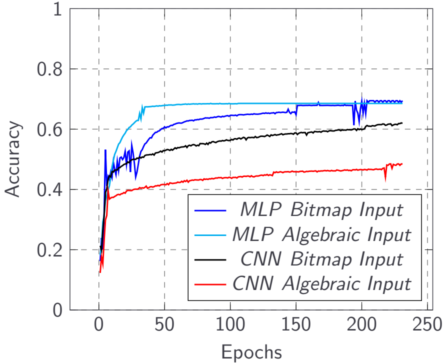
Results

Dataset 2



ANN	ValSet	TestSet
MLP Bit	93.73%	**93.41%**
MLP Alg	87.45%	87.28%
CNN Bit	87.24%	87.10%
CNN Alg	83.88%	83.72%

Dataset 3



ANN	ValSet	TestSet
MLP Bit	69.44%	**69.33%**
MLP Alg	69.88%	66.21%
CNN Bit	62.06%	61.97%
CNN Alg	48.48%	46.86%

Dataset 4

ANN	Bitmap Input		Algebraic Input	
	ValSet	*TestSet*	*ValSet*	*TestSet*
MLP	0.0011	**0.0016**	0.0019	0.0021
CNN	0.0020	0.0022	0.0021	0.0022

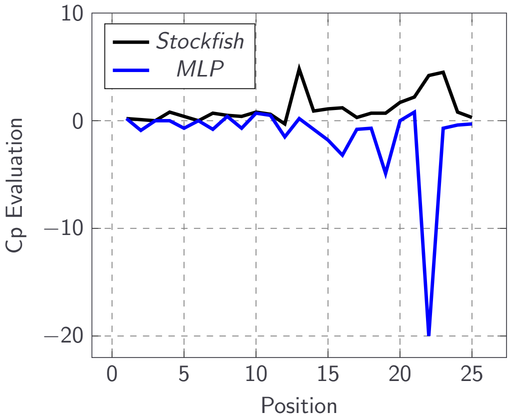
Table: The MSE of the ANNs on the regression experiment.

- Both MLPs and CNNs are powerful function approximators
- Best ANN evaluation is only ≈ 0.035 cp different from *Stockfish*

Final Game Performances: Kaufman Test

- Testbed of 25 very complex chess positions
- Find the best move given a board state S_t
- We only evaluate S_{t+1}
- Check if the move played by the ANN is the one of the test
- $\Delta cp = \delta K - \delta NN$
- δ : deepest evaluation given by *Stockfish*

Final Game Performances: Kaufman Test



- 2 times the MLP plays *Stockfish's* move
- 3 times the MLP makes a move leading to a losing position
- $\mu(\Delta \text{ cp}) = \mathbf{1.56}$

Final Game Performances

- ANN played 30 games on a reputable chess server [2]
- Opponents with an ELO rating between 1741 and 2140
- All games against opponents with a rating < 2000 ($\approx 70\%$ of the games) were easily won
- However against Master titled players (Elo > 2000) only 2 draws were obtained
- ANN without lookahead makes mistakes in heavy tactical positions

[2] https://chess24.com/en

Discussion and Conclusion

Discussion and Future Work

- MLPs provide a better neural architecture than CNNs
- Algebraic Input performs worse than the Bitmap Input
- The results obtained on the Kaufman Test (with the Δcp) and on the chess server, make it possible to state that playing high level chess without relying on expensive lookahead algorithms is possible, but has limitations

- Combination between current ANN and **quiescence search algorithms**
- Improve the performance of CNNs by adding extra feature layers to the input

Thank you

Thank you for your attention.

Questions?