Queens Gambit Accepted:

Universal Chess Piece Recognition for Enhanced AI Move Prediction

Zachary Belles [1], Joshua Czajka [2], Mohammad Hafezi-Mashhadi [3], Tony Le [4]
[1] Department of Computer Science, North Central College, Naperville, Illinois, United States
[2] Department of Computer Science, North Central College, Naperville, Illinois, United States
[3] Department of Computer Science, North Central College, Naperville, Illinois, United States
[4] Department of Computer Science, North Central College, Naperville, Illinois, United States

Abstract— **The advancement of artificial intelligence (AI) in the realm of chess has brought forth highly skilled AI players, yet the application of these algorithms on various board states, whether physical or digital, has not been explored to its full potential. In this paper, we introduce a novel machine learning model designed to accurately identify and distinguish chess pieces on any given board state physical or digital, which subsequently enables chess move prediction models to adapt to any possible board state seamlessly. Our approach focuses on training a robust and versatile model capable of extracting meaningful features from chess pieces with a 90% accuracy, allowing newer move prediction models to be imported into any game.**

Keywords - chess, machine learning, Convolutional Neural Network (CNN), python, pandas, pytorch, google colab, chess.com, image classification, Kaggle, Automated Intelligence, AI, chessvision.ai, COVID-19, backpropagation, tensors, image classification, chess pieces

I. Introduction

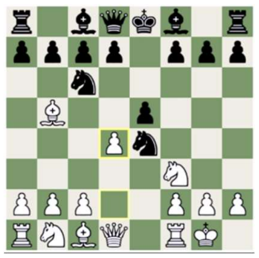


Figure 1.1

While many machine learning models related to chess focus on move prediction or board state awareness, not a lot of research has been done into the classification of different artistic representations of chess pieces in both the physical and digital world. While all bishops have the same basic shape, any artists take different creative liberties when crafting or designing chess pieces for their respective chess sets. This is shocking to us as a group since worldwide interest in chess is at an all-time high[1]. While services such as chessvision.ai exist allowing users to analyze board states from eBooks, this only works for a single type of board and piece representation, such as the one you can find at chess.com and in figure 1.1. We believe that now that chess best move prediction algorithms have gotten so advanced that more focus should now be put into allowing these models to work on any type of piece *and* any possible board state. Given the growth of chess, especially in teens and young adults[2], since the beginning of the COVID-19 pandemic, along with the accessibility of technology and chess in the United States, anyone from any age should be able to put in the board state of whatever game they're playing, online or in person, and be able to see not only what the best fit move is for that orientation, but also *why* it is the best move for that occasion.

II. Methods

(i) Experiment

a. Our primary objective was to optimize the accuracy of our models for identifying chess pieces, as we strive to enhance the capabilities of existing chess AI to enable compatibility with diverse artistic representations of chess pieces. In addition, we sought to test the practical impact of adding additional convolution layers on classic machine learning complications such as vanishing gradient, overfitting, oversaturation, and degradation. By conducting these tests, we aimed to achieve a more comprehensive understanding of the performance characteristics of our models, and to inform future improvements in our neural network architecture to be integrated into chess move prediction models and software's.

(ii) Model overview

a. Our research explores the efficacy of utilizing a Convolutional Neural Network (CNN) for identifying chess pieces in images. The CNN architecture was chosen due to its suitability for processing image datasets, with the ability to extract meaningful features that differentiate various chess pieces from one another. This obviates the need for explicit feature engineering as the network can learn these features automatically via the backpropagation process during training. By leveraging the power of CNNs, our study demonstrates promising results for accurate and efficient chess piece recognition. A summary of our models can be found in figures 2.1, 2.2, and 2.3.

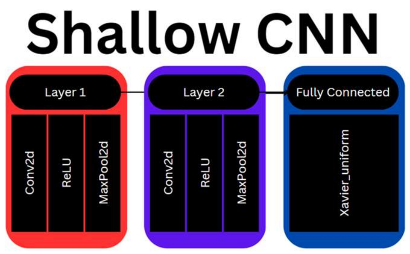


Figure 2.1

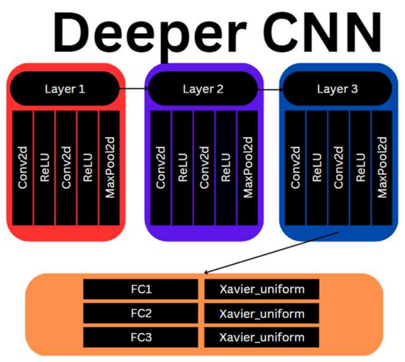


Figure 2.2

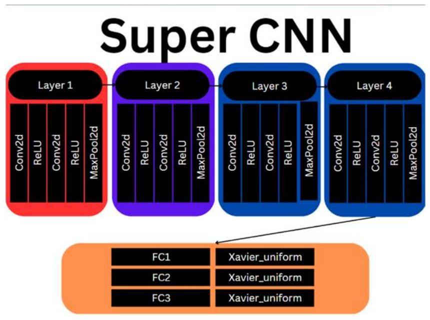


Figure 2.3

(iii) Shallow CNN

a. The shallow CNN model makes use of two convolutional layers each with their own ReLU activation function. The first convolutional layer takes a 3-channel input (corresponding to RGB color space) and applies 32 filters of size 3x3. The second layer takes 32 feature maps from the previous layer and applies 64 filters of the same size. The output of the second convolutional layer is flattened and fed into a fully connected layer with 6 output neurons, which correspond to the 6 classes of chess pieces in the dataset. The model's weights are initialized using the Xavier initialization method to which the forward function takes the input tensor and applies the layers sequentially, returning the output tensor.

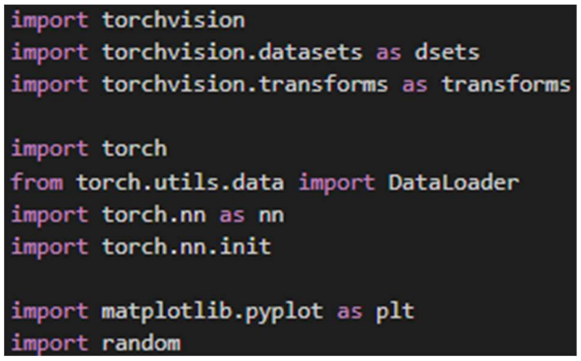
(iv) Deeper CNN

a. The Deeper CNN features an additional convolutional layer, resulting in a deeper architecture. The first two convolutional layers are identical to the Shallow CNN however the third layer applies 128 filters, followed by ReLU activation and another 128 filters. Along with this each convolutional layer is followed by max-pooling operations and the output of the third convolutional layer is flattened and fed into three fully connected layers, each with their own ReLU activation function. The first fully connected layer has 256 output neurons, the second has 128, and the final layer has 6 output neurons corresponding to the 6 chess piece classes. The weights of the fully connected layers are initialized using the Xavier initialization method. The forward function applies the layers sequentially, returning the output tensor.

(v) Super CNN

a. The Super CNN model is another implementation of a convolutional neural network in PyTorch. This model is also functionally equivalent to the Deeper CNN except it has a fourth convolutional layer and each convolutional layer is followed by max-pooling operations. The output of the fourth convolutional layer is flattened and fed into three fully connected layers, each with their own ReLU activation function. The first fully connected layer has 512 output neurons, the second has 256, and the final layer has 6 output neurons, corresponding to the 6 classes of chess pieces in the dataset. The weights of the fully connected layers are initialized using the Xavier initialization method. The forward function applies the layers sequentially, returning the output tensor. Compared to the previous models, the Super CNN model is more complex, featuring more layers and more filters, with the ability to process higher resolution images. At the same time, due to its complexity, we hypothesized it will befall the typical shortcomings of too complex of a neural network such as degradation and overfitting. This hypothesis was subsequently confirmed during testing and evaluation.

(vi) Libraries used



```
import torchvision
import torchvision.datasets as dsets
import torchvision.transforms as transforms

import torch
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.nn.init

import matplotlib.pyplot as plt
import random
```

(vii) Hyper-parameters

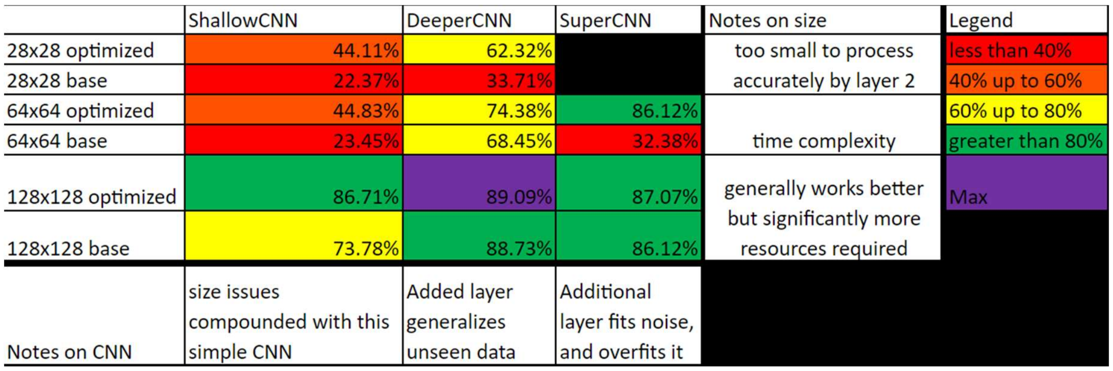
	Shallow CNN	**Deeper CNN**	**Super CNN**
Learning Rate	$5.0 \cdot 10^{-5}$	$5.0 \cdot 10^{-4}$	$5.0 \cdot 10^{-5}$
Batch Size	100	80	50
Epochs	50	50	50

(viii) Sample image statistics

Piece Type	**# in training set**	**# in testing set**
Bishop	138	28
King	141	13
Knight	74	34
Pawn	173	16
Queen	114	23
Rook	138	27
Total	**778**	**141**

III. Results

As shown and described earlier in the methodologies section, the dataset was resized and split to a train and test dataset and analyzed using a deep Convolution Neural Network model labeled "Deeper CNN". Though, for the entire experiment, three CNN models, with different sets of optimized hyperparameters, were used and compared with each other, and out of the three models, the Deeper CNN produced the best results during testing. Another core component in addition to the models was the inclusion of a loss function and an optimizer to ensure that the model produced the best results for training and testing and minimized the cost. In order to maximize the training for the Deeper CNN model, the hyperparameters set was 50 epochs, batch size set to 80, and learning rate set to 0.0005. The result of the testing for all the models are depicted in Figure 3.1 below.



	ShallowCNN	DeeperCNN	SuperCNN	Notes on size	Legend
28x28 optimized	44.11%	62.32%		too small to process accurately by layer 2	less than 40%
28x28 base	22.37%	33.71%			40% up to 60%
64x64 optimized	44.83%	74.38%	86.12%	time complexity	60% up to 80%
64x64 base	23.45%	68.45%	32.38%		greater than 80%
128x128 optimized	86.71%	89.09%	87.07%	generally works better but significantly more resources required	Max
128x128 base	73.78%	88.73%	86.12%		
Notes on CNN	size issues compounded with this simple CNN	Added layer generalizes unseen data	Additional layer fits noise, and overfits it		

Figure 3.1. 28x28 not run on Super CNN as we were aware of the issues with the size at that complexity.

As illustrated in Figure 3.1, our most effective model is the Deeper CNN with a resolution of 128x128 with an accuracy of 89.09%. This, coupled with our cost and accuracy data from running the testing and training, confirms our hypothesis that in a real-world testing environment there is a threshold beyond which the addition of extra layers allows the model to start fitting noise to the images, leading to poor generalization performance. Due to this, in addition to issues that arise from model degredation[3], the accuracy of the model saturates and then starts to degrade as the network depth increases.

IV. Conclusion

Numerous machine learning models pertaining to chess have been developed, such as systems which read physical board states with arduino[4], or using cameras to read multiple chess images at once imported from an older camera phone[5]; however, the research on the classification of various artistic representations of chess pieces remains limited.

Existing services, such as chessvision.ai, can analyze board states but are restricted to a singular type of board and piece representation. We posit that, in parallel with the advancements in move prediction algorithms, it is crucial to focus on enabling these models to function with diverse piece types and potential board states. The implementation of image classification models, utilizing Convolutional Neural Networks (CNN) in Python and incorporating tools such as Pandas, Torch, and Google Colab, can contribute to bridging this gap. Furthermore, data repositories and platforms such as Kaggle and Chess.com can serve as valuable sources for data acquisition and model testing.

REFERENCES

[1] Keener, G. (2022, June 17). Chess is Booming. The New York Times. https://www.nytimes.com/2022/06/17/crosswords/chess/chess-is-booming.html

[2] Carpenter, Nicole. “Why Teens Are Suddenly Obsessed with Chess.” Polygon, Vox Media, 12 Apr. 2023, www.polygon.com/tabletop-games/23679440/teens-love-chess-memes-boom-2023. Accessed 26 Apr. 2023.

[3] Vela, D., Sharp, A., Zhang, R. et al. Temporal quality degradation in AI models. Sci Rep 12, 11654 (2022). https://doi.org/10.1038/s41598-022-15245-z

[4] “Piece Recognition - Chessprogramming Wiki.” Www.chessprogramming.org, www.chessprogramming.org/Piece_Recognition#:~:text=the%20ability%20of%20dedicated%20chess%20computers%20or%20chess. Accessed 26 Apr. 2023.

[5] Czyzewski, Maciej & Laskowski, Artur & Wasik, Szymon. (2018). Chessboard and chess piece recognition with the support of neural networks