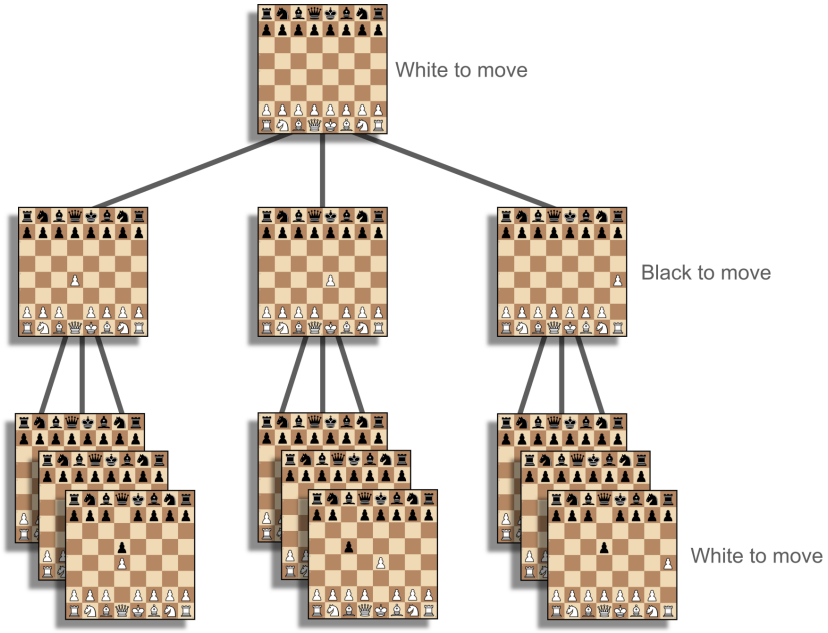
Designing Chess Playing Programs using Deep Learning Algorithms



Banuijan Yognanathan, 4a

KS Im Lee, Supervising Teacher : Cyril Wendl

Prepared for 21.10.24

Contents

List of Figures

0 Idea and Motivation

Not long ago, the invention of ChatGPT became an interesting topic in the mass media. Along with it, other AI technologies have gained more relevance. Nowadays, AI is prevalent in our world, and it seems to solve problems of any difficulty much faster than a human being. As an aspiring programmer, this topic piqued my interest, especially when I thought about my Matura project. I had the idea of combining these two topics and creating a chess-playing system using deep learning techniques. More precisely, I wanted to explore how well such a chess-playing program can adapt to a specific skill level.

Brief Introduction into Deep Learning

Deep Learning makes up the very first building block of an artificial intelligence. A so-called "Neural Network" is the fundamental algorithm in order to exercise deep learning. What differentiates deep learning of machine learning is the fact that deep neural networks use more than 3 layers as "normal" neural nets use exactly 3 (1 input layer, 1 hidden layer, 1 output layer). Machine Learning is commonly used in occasions where human experts need to determine the characteristics of the data (labelled data), where in deep Learning Algorithms, one the neural network will find the characteristics of the (unlabeled) data itself (at least you hope so). When using "labelled data" one

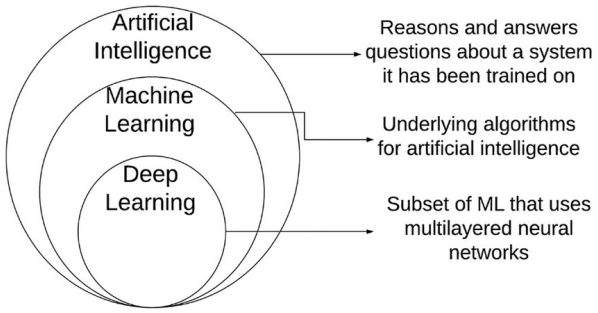


Figure 1: Venn-Diagram

refers to "supervised learning". Deep Learning do not necessarily require labelled data. If one uses "unlabeled data", one refers to "unsupervised learning".

Choosing the right Network

As I began browsing the internet for examples of chess engines, I found multiple options. However, since I wanted to use neural networks, two types of procedures caught my attention: reinforcement learning and the use of a convolutional neural network. Reinforcement learning (RL) is an interdisciplinary area of machine learning and optimal control concerned with how an intelligent agent should take actions in a dynamic environment in order to maximize a reward signal [1]. In respect to my matura project, this means that for each move the enemy makes, the neural network will counter with the best move based on the position of the board.

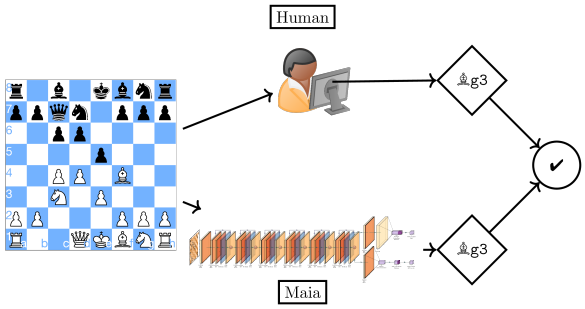


Figure 2: CNN

The type of neural network I chose is the convolutional neural network. A convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns features by itself via filter (or kernel) optimization [2]. This type of deep learning network has been applied to process and make predictions from many different types of data including text, images and audio. The question which might arises is why do I choose convolutional neural networks for something which has nothing to do with images and audio. As you will later see, convolutional neural networks are known for capturing

positional information, and since chess is a game which deals a lot with positional play, I found this approach very intuitive.

Idea

As you can see in the illustration on the left, I created so-called "feature maps" for every single chess piece, using the python chess module. I chose to make this project in python due to this module and a module called pygame, which we will discuss later when we discuss the visualisation part. With such matrices, captured in an array the neural network is able to capture positional information, and learn using convolutional layers. The feature maps shown on the image are feature maps for pawns, rooks, bishops and knight, where "-1" represents the black and "1" the white pieces. Each input feature map will encode the location of a different type of game piece. This way, the neural network can learn the rules of the game and make strategic moves based on the state of the board [3].

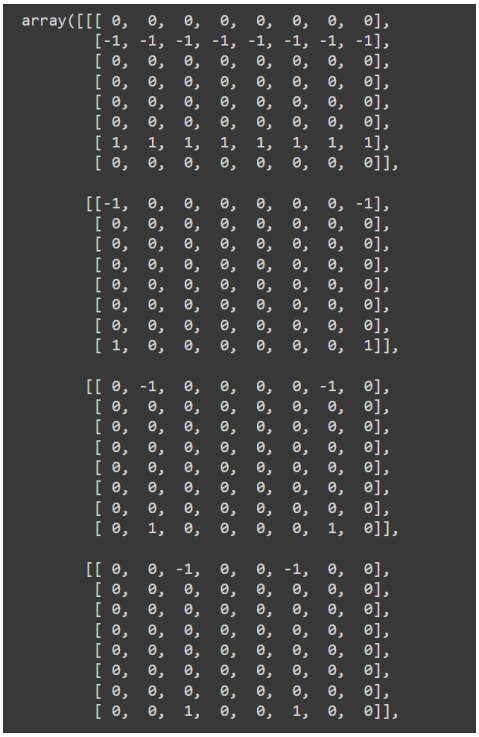


Figure 3: Feature Maps

I oriented myself on a YouTube tutorial from "Moran Reznik" which guided me through the different aspects one has to think about when building such a neural network.

```python
class ChessDataset(Dataset):

  def __init__(self,games):
    super(ChessDataset, self).__init__()                    #call the constructor of its parent class
    self.games = games


  def __len__(self):
      return 40_000                                         #use 40k games, because moves are randomly selected anyway (save data?)

  def __getitem__(self, index):
    game_i = np.random.randint(self.games.shape[0])         #random integer from "Dataset"
    random_game = chess_data['AN'].values[game_i]           #this integer = random game
    moves = create_move_list(random_game)                   #clean up notation by calling above created function
    game_state_i = np.random.randint(len(moves)-1)          #pick random move, and -1
    next_move = moves[game_state_i]                         #next move is game_state_i + 1
    moves = moves[:game_state_i]                            #all moves until current move
    board = chess.Board()
    for move in moves :
      board.push_san(move)                                  #loop through all moves

    x = board_2_rep(board)                                  #above function for feature map of every position
    y = move_2_rep(next_move, board)                        #feature map for played move
    if game_state_i % 2 == 1:                               #always play positive values
      x*= -1
    return x, y
```

Figure 4: PyTorch Classes

But the Idea is very simple. After creating a feature map for every piece, we can create a move function with the aid of those feature maps. Before we can create a way how to make the neural network learn, we edit the dataset in a convenient way, such that the input is easy to understand. The neural network picks a random move from a game, loops through all the moves until the selected moves and creates the respective feature maps. Using multiple functions and a convolutional neural network, it captures positional information. This way, the neural network is able to learn.

1 Main Part

2 Closing Words

3 More

4 Appendix (Sources)

[1] https://en.wikipedia.org/wiki/Reinforcement_learning, 20.10.24

[2] https://en.wikipedia.org/wiki/Convolutional_neural_network, 20.10.24

[3] https://www.youtube.com/watch?v=aOwvRvTPQrs&t=800s, 20.10.24

Image sources

Figure 1: https://www.researchgate.net/figure/Venn-Diagram-relationship-between-artificial-intelligence-machine-learning-and-deep_fig2_363930739

Figure 2: https://www.maiachess.com/, 20.10.24

Figure 3: own code

Figure 4: own code