Othello CNN

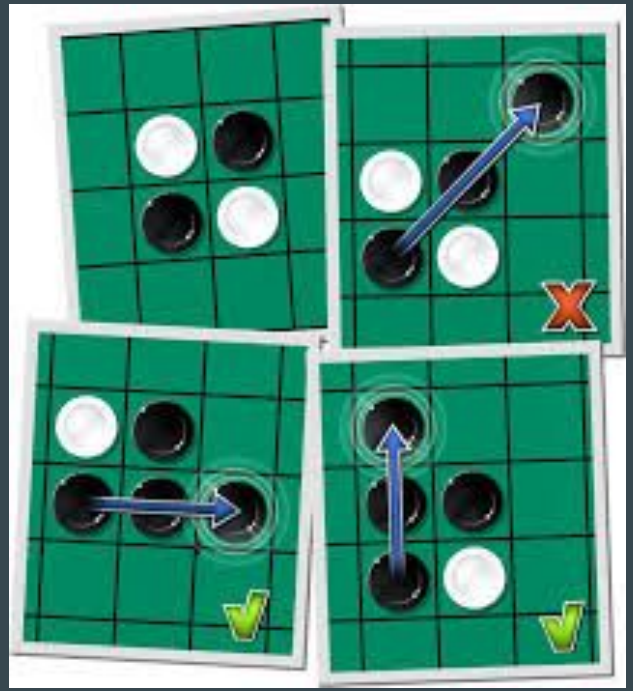
Anthony Hordesky, Luka Starcevic

Brief Background

The goal of the project was to develop a Convolutional Neural Network for the game Othello. The planned approaches included:

- Supervised Learning
- Reinforcement Learning
- Performance Benchmarking/Testing

Only performance benchmarking/testing was completed against a minimax algorithm bot. Supervised learning and reinforcement learning were not fully explored.

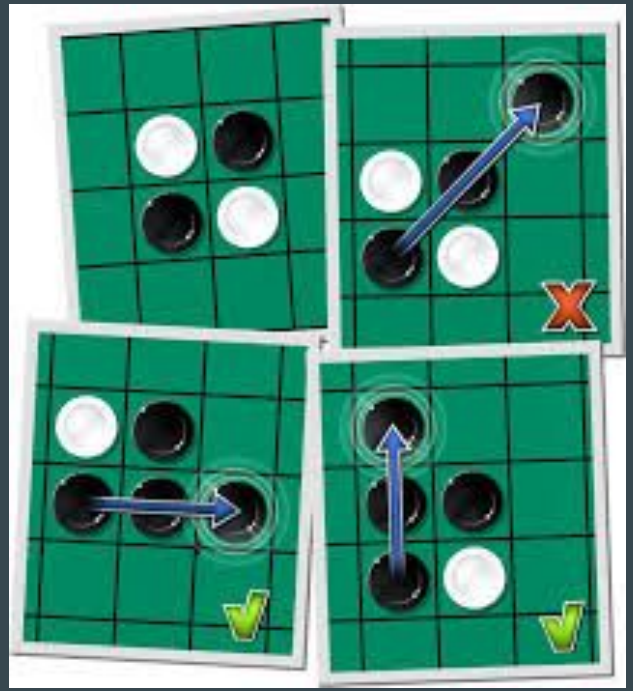


Brief Background

The project was done using three main steps:

- Data Cleaning/Preprocessing
- Training and Tuning
- Performance Benchmarking/Testing

The dataset we used was from the eOthello database, a csv file containing >25,000 games from the 100 best Othello players on the website.



Data Cleaning/Preprocessing: Data Cleaning

In the CSV file, specific rows were removed if they were missing a winner or a game string.

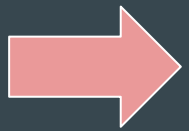
354387	-1	d3c3c4c5f6e3c6f4b5a6f2f3b4a5a3b3a2b6d6f1c2a4a7d2f5e2d1e6g5e1c1b1g3g6e7f7f8h5h4h3h6g4h2c7g2d7e8d8b2b7...
353677		c4e3f5e6d3c5d6c6f3c3b4b3d2f4f6c2b5a5a4a3e2d1e7g5g4h4d7c7b6g3f2g6h3h2d8f7h5h6g2h1g1f8e8c8b7f1g7h8e1g8...
359712	1	f5f6e6d6e7f4g5d7c6e8c5c4c8d8f8g4d3f7c3c7e3g3g8e2h3h5h4h2g6f3d2b4b3b6f2c2d1f1e1c1b8a3b5g7a4g1a5a6b1a1...

354387	-1	d3c3c4c5f6e3c6f4b5a6f2f3b4a5a3b3a2b6d6f1c2a4a7d2f5e2d1e6g5e1c1b1g3g6e7f7f8h5h4h3h6g4h2c7g2d7e8d8b2b7...
359712	1	f5f6e6d6e7f4g5d7c6e8c5c4c8d8f8g4d3f7c3c7e3g3g8e2h3h5h4h2g6f3d2b4b3b6f2c2d1f1e1c1b8a3b5g7a4g1a5a6b1a1...

Data Cleaning/Preprocessing: Game Objects

CNNs interpret visual data like patterns in images, so the game strings were converted to board instances. These instances included 8x8 matrices containing 0s, 1s, and -1s, referring to empty spaces, black pieces, and white pieces in that order.

359712	1	f5f6e6d6e7f4g5d7c6e8c5c4c8d8f8g4d3f7c3c7e3g3g8e2h3h5h4h2g6f3d2b4b3b6f2c2d1f1e1c1b8a3b5g7a4g1a5a6b1a1...



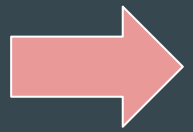
```
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, 1, 0, 0, 0],
[0, 0, 0, 1, 1, 1, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0]
```

Data Cleaning/Preprocessing: Pruning

To better optimize the training data, the original 25000 games were reduced to 10500. Of this, 5000 were white wins, 5000 are were wins, and 500 were ties. The board instances of these games were then converted into a PyTorch Tensor.

25000 Games

483716	1	f5f6e6f4g6f7g5e7d6c7d7h4h5c4h3c8c5g4f8e8g3g8f3c6c3c2d8h7h6h2e3g7h8d2h1e2g2f2d3g1f1e1d1c1b6b5a5a6a7b7...
482856	1	f5f4e3d6f3g5f6g3g4e7h6g6h4d3f2f1e6h5h3f7d7d8f8e8c8c7g8g2b7g7e2h2h1g1h8d1h7b8a8d2c1b1b6c2c6c5c4b4a7a6...
482854	1	f5f4e3f6d3c5d6c7d7c4e6f2f3c6c3d8f1e2d2c2g5g6e7d1g4e1c1h3h4f8f7e8h6g3b1h5h2g7b3b5a6a4h7b2b4a3a5a7a1b6...



5000 white wins - 5000 black wins - 500 ties -

1	f5f6e6f4g6f7g5e7d6c7d7h4h5c4h3c8c5g4f8e8g3g8f3c6c3c2d8h7h6h2e3g7h8d2h1e2g2f2d3g1f1e1d1c1b6b5a5a6a7b7...
1	f5f4e3d6f3g5f6g3g4e7h6g6h4d3f2f1e6h5h3f7d7d8f8e8c8c7g8g2b7g7e2h2h1g1h8d1h7b8a8d2c1b1b6c2c6c5c4b4a7a6...

Data Cleaning/Preprocessing: Final Training Dataset

Finally, the data was ready to be used. To train the CNN directly, the data had to be processed into a PyTorch Dataset. To do so, the subsequent steps were followed:

1. OthelloDataset was created to iterate through each game and its sequence of board instances
2. A second channel was added to the PyTorch Dataset to keep track of player moves associated with each board state
3. A move index was calculated for each individual tensor; this data was appended to the OthelloDataset
4. A PyTorch dataloader was created using the dataset with a batch size of 32 randomized games

Training and Tuning: Model Setup

The Othello model was initialized and works by taking in a board instance, extracting spatial patterns via four specified convolutional layers, and by using these four connected layers to output 64 move probabilities (per board move). We initially used 3 layers, but added a 4th one due to high loss.

Fully connected layers (FC1 & FC2) process flattened convolutional features using PyTorch's nn.Linear. The optimizer (PyTorch Adam) updates weights with a learning rate of 0.001 to balance precision and stability.These layers perform matrix operations to transform the input board instances.

Training and Tuning: Training Loops

The training loop runs for 10 epochs on our game dataset. Train 1 took 30 minutes (8.6MB), while Train 2 with the extra layer took 2 hours (10MB). The loss function, found in model.py, applies weighted cross-entropy based on player color and game outcome. Loss is back-propagated, and weights are updated after each batch.

The loss and training statistics are printed after each epoch to help track model performance.

Training and Tuning: Train 1 Output

The results were saved as a PyTorch state dictionary to store all the data and parameters. Below are the results of the second main train.

Train 1 Output (3 conv layers):
Data set up
Model set up
Epoch [1/10], Loss: 5.8806
Epoch [2/10], Loss: 5.3936
Epoch [3/10], Loss: 5.2227
Epoch [4/10], Loss: 5.0452
Epoch [5/10], Loss: 4.8826
Epoch [6/10], Loss: 4.7033
Epoch [7/10], Loss: 4.5016
Epoch [8/10], Loss: 4.2197
Epoch [9/10], Loss: 3.9136
Epoch [10/10], Loss: 3.5624

Training and Tuning: Train 2 Output

The results were saved as a PyTorch state dictionary to store all the data and parameters. Below are the results of the second main train.

Train 2 Output (4 conv layers):
Data set up
Model set up
Epoch [1/10], Loss: 3.4500
Epoch [2/10], Loss: 2.8779
Epoch [3/10], Loss: 2.7225
Epoch [4/10], Loss: 2.6138
Epoch [5/10], Loss: 2.5312
Epoch [6/10], Loss: 2.4689
Epoch [7/10], Loss: 2.4217
Epoch [8/10], Loss: 2.3854
Epoch [9/10], Loss: 2.3561
Epoch [10/10], Loss: 2.3320

Performance Benchmarking/Training

For the performance testing of our CNN, we let it play against a minimax algorithm for Othello since it is a commonly used algorithm for Othello due to its efficiency.

We set the depth of the minimax algorithm to 8, to balance between the playing strength and computation time. In the 20 games played, our CNN bot tied every game as white, and lost every game as black.

Game	Starting Color (CNN)	Winner	Score
1	Black	MM	30-34
2	Black	MM	30-34
3	Black	MM	30-34
4	Black	MM	30-34
5	Black	MM	30-34
6	Black	MM	30-34
7	Black	MM	30-34
8	Black	MM	30-34
9	Black	MM	30-34
10	Black	MM	30-34
11	White	Tie	32-32
12	White	Tie	32-32
13	White	Tie	32-32
14	White	Tie	32-32
15	White	Tie	32-32
16	White	Tie	32-32
17	White	Tie	32-32
18	White	Tie	32-32
19	White	Tie	32-32
20	White	Tie	32-32

Conclusion

We developed an Othello bot CNN based on a set of expert games and achieved a significant loss reduction through two trains, with our addition of another convolutional layer. While the model demonstrated efficient move prediction, its performance against a minimax Othello bot at ply 8 revealed limitations that we hope to address in the future.

We'd like to improve our bot further by implementing reinforcement learning on our current CNN. This training would play itself, using a Q-score to measure performance. We would also like to train our Othello bot against other online bots, since they likely use advanced heuristic models for their AI.