Deep Learning Project : Neural Network Models as Othello Player

1. Introduction to Othello:

Othello is the trading name of a much older board game, Reversi. Often referred to as a game of abstract strategy, Othello can only be played as a 2 player game. Made up of 8 rows and 8 columns, the board is populated with pieces from both players in turn. Each player's pieces will be of one color, with the most common occurrence being black and white.

1.1. How to Play Othello

Players battle to finish the game with more of their own pieces on the board than their opponent. The game is classed as finished when there are no spaces left on the board or there are no more possible legal moves for either opponent.

1.2. The Start

Both players begin the game with two pieces on the board in the four center squares. No two matching colors are connected vertically or horizontally. In the typical set-ups where it is black versus white, the person using black chips must make the first move.

1.3. The Game

Both players take it in turns to make their move, which consists of placing one piece down in a legally acceptable position and then turning any of the opposing player's pieces where applicable. A legal move is one that consists of, for example, a black piece being placed on the board that creates a straight line (vertical, horizontal or diagonal) made up of a black piece at either end and only white pieces in between. When a player achieves this, they must complete the move by turning any white pieces in between the two black so that the line becomes entirely black. Players will then continue to move alternately until they get to the end of the game and a winner is decided. This decision is reached by identifying which of the two opponents has the most pieces on the board.
Players take alternate turns. If one player cannot make a valid move, play passes back to the other player. When neither player can move, the game ends.
Link of online game : https://www.eothello.com/

2. Othello Game dataset:

The logs of more than 25k Othello games, played by humans, are available in Kaggle database.
After a preprocessing, this data has been pruned and transformed to new format.

Deep Learning Project : Neural Network Models as Othello Player

2.1. Pruning:

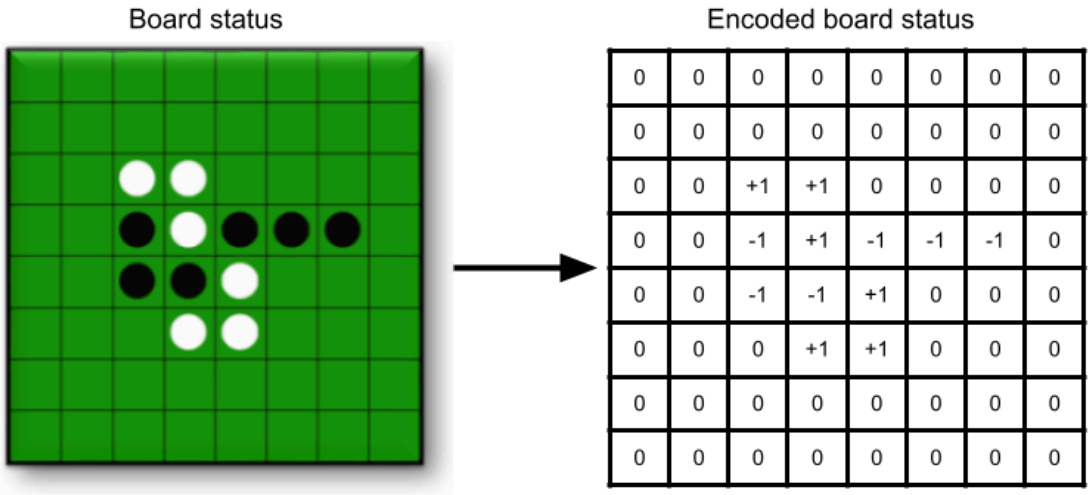
First, all games, that contain at least one pass, have been filtered out. All games, that does not have complete 60 moves, have been filtered out. It remains, 8399 complete games.

2.2. Transforming format:

A game has been coded as two series of array.
One series represent the status of board (8×8 array) which contain 3 values, 0 as an empty cell in board, +1 as a cell of board which is occupied by player 1, and -1 as a cell of board which is occupied by player 2.
Another series represent all moves of players. For each move, a spars array (8×8) is defined. A move array is a zero array with a single cell contains +1 which represents the player's move.
The order of these arrays in the series is corresponded to the steps of game.
Since we have only complete games, The shape of game array is (2,60,8,8).

- **8,8 :** 8×8 array represent a board status or a move
- **60 :** represent series of board status or moves. The first move is done by black and next by white, and it turns alternate one by one. Since the game board contain 8×8 or 64 cells and the game starts with 4 occupied cells, it requires 60 moves (30 moves by each player).
- **2 :** represent one series of array for board status and one for moves.

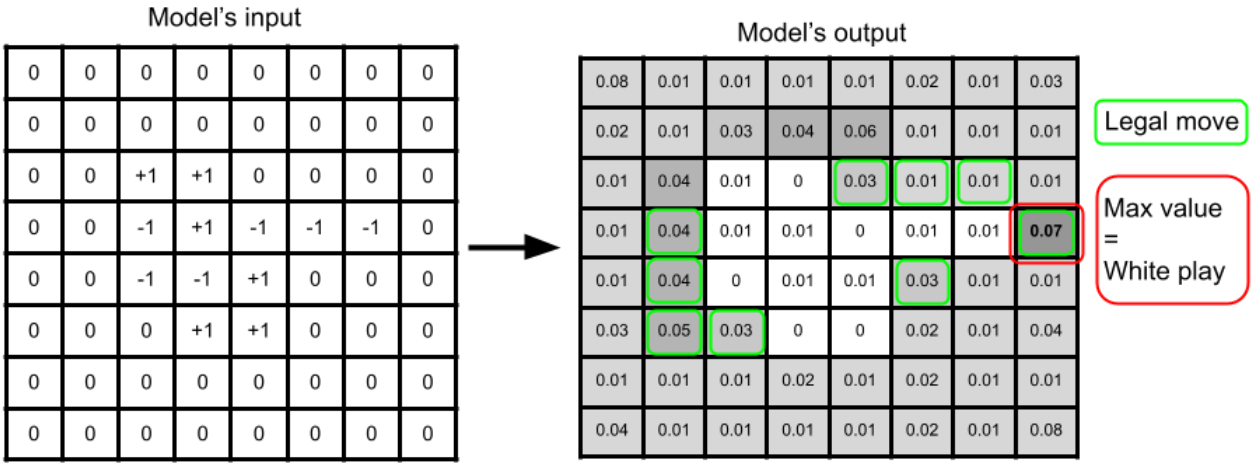
In the below figure, you can see an encoded format of board status to an input array.



Deep Learning Project : Neural Network Models as Othello Player

3. Training a neural network model

The objective is to predict a move array based on a board status (or several consecutive board statuses as time series input).
You need to train a network as a player. The output of the network is an 8×8 array which provide a probability for each cell of the board as the player move. The output array would be filtered by all legal moves by the corresponding player, and then the cell with maximum value would be selected as the player's move. In the below figure, the next move of the white player has been calculated.



Model's input							
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	-1	-1	0
0	0	-1	-1	+1	0	0	0
0	0	0	+1	+1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Model's output							
0.08	0.01	0.01	0.01	0.01	0.02	0.01	0.03
0.02	0.01	0.03	0.04	0.06	0.01	0.01	0.01
0.01	0.04	0.01	0	0.03	0.01	0.01	0.01
0.01	0.04	0.01	0.01	0	0.01	0.01	**0.07**
0.01	0.04	0	0.01	0.01	0.03	0.01	0.01
0.03	0.05	0.03	0	0	0.02	0.01	0.04
0.01	0.01	0.01	0.02	0.01	0.02	0.01	0.01
0.04	0.01	0.01	0.01	0.01	0.02	0.01	0.08

In order to have one model (player) who is capable of playing white and black, we transform all board status to a white player turn. It means when the model has to predict a move for black, the board status would be multiplied by -1 (black cell change to white and white cell change to black). Indeed, there would not be any change for white player.

3.1. Python package:

A package of code has been provided as baseline to facilitate the game and the training. This package contains files as follows:
game.py : This code is used for playing Othello game between two models. The game would be run two times with different colors for each player (different starting player) and it generates a GIF file as the log of each game.

- training_[x].py : this code is provided as an example of training a model.
- networks_[your_student_number].py : This file contains your networks. You can define and compare several models with different class name in this file. **You should change the name of this files to your student number, which is unique, as the first step before any training.**

Deep Learning Project : Neural Network Models as Othello Player

- utils.py : this file contains functions that will be used as the rule and different steps of games.
- data.py: This file is responsible for handling data loading and preprocessing tasks. It includes functions to prepare datasets for training and evaluation, ensuring compatibility with the models defined in the networks file.

You can download the package from GitLab by following address:

https://git-lium.univ-lemans.fr/mshamsi/deeplearning2playothello

Use below command line in terminal to download the package:

```
git clone https://git-lium.univ-lemans.fr/mshamsi/deeplearning2playothello
```

To prepare all dependencies, it is needed to install all packages in requirement by:

```
pip install scikit-learn pandas tabulate matplotlib h5py scipy tqdm torch torchvision
```

or:

```
pip install -r requirements.txt
```

3.2. Evaluation metric for training:

The main objective of a trained model is to win the game. The models are trained to mimic the human players. The game.py provide the chance of competing two players and create a log of game. It means not only you can compare two models that you prepared, but you can also compete with others' trained model.

You can also prepare two models and based on the log of their game create a new training sample.

In an ideal situation, we will have a league of Othello that all models can play with each other to find the champion of ENSIM AI Othello. Don't forget, "*if there is no rule, then there is no limitation*".

4. Report of TP:

This TP is 70% of your final note. It requires a report of your work that explain all experimental design and the result of your work that you have done in order to find your best model.

The report should cover:

Your data partitioning or your approach for collecting more data.

Your proposed and tested architectures and hyperparameters

The result of your model's performance based on accuracy for all tested architectures and hyperparameters **(tables or figures with number)**.

At the end of TP course, you should present your work in 5 minutes as well

Deep Learning Project : Neural Network Models as Othello Player

5. Evaluation chart

Here, you can find a suggested TODO list that you can follow. The importance and credits of each task are indicated in the second column. This is only a proposal, and you can certainly do more than this and go beyond this list.

Calculate and Compare the results of provided MLP and LSTM baseline	5
Optimizing the architecture of MLP (reporting the number of trainable weights is necessary)	5
Optimizing the architecture of LSTM (reporting the number of trainable weights is necessary)	5
Testing different optimizer (at least two optimizers)	5
Optimizing learning rate (testing 0.0001, 0.001, 0.01, 0.1)	5
Checking the impact of different epochs and batch size	5
Using all data and not only winner samples	5
Calculating the learning curve (training/dev performance on epochs)	5
Analysing learning curves	10
Not choosing a valid evaluation metric (the reported accuracy, loss should be on dev/test)	-10
Experiment design with logical conclusion (how they find the best model?)	5
Analysing results to diagnose the problem or improve results	10
Analysing different evaluation metrics such as “game win ratio” or “legal move ratio”	5
Using more data in training after finding the best model (adding test or dev in final training)	5
Implementation of a CNN network and optimizing its architecture	10
Implementation of a new architecture like CNN-LSTM or attention based (Transformer)	10
Implementation of pretrained model/architecture, for example from huggingface	10
Generate new data and adding to training set (for example, using the log of two AI players)	20
Presentation (respecting time, choosing the important parts to present, reporting as table or figure)	10