DEEPCHECKERS

Muhammet Şeramet
mseramet02@gmail.com

Cem Güleç
cem.ggulecc@gmail.com

Büşra Gökmen
busragokmen67@gmail.com

Asst. Prof. Mehmet Kadir Baran

Department of Computer Engineering, Marmara University, Turkey

INTRODUCTION

DeepCheckers is an approach to master the game of Turkish Checkers (Dama) starting from tabula rasa -empty state- and without any guidance from human knowledge.

Board games: Good test environments for scientists due to complexity, popularity and simplicity. (Chess, Go, etc..)

DeepBlue: Consists of single-chip chess search engine and game database consists of Grandmaster game-play information. Lost against Kasparov in 1996 (4-2) and won in 1997 (3.5-2.5).

AlphaGo(AlphaZero): Game search tree and two Deep Neural Networks. Won against Lee Sedol in 2016 (4-1).

We use a lightweight re-implementation of AlphaZero: AlphaZeroGeneral created by Suragnair[1], which combines conceptual structures we use in our implementation.

Problems

- Turkish checkers was a dead game, it has almost zero literature surveys.
- Turkish checkers has repetition moves, this were causing stack overflow. Founding this problem was very difficult.

Solution

- For the purpose of creating a spark in a field that is forgotten, we offer most up-to-date technological concepts.
- To solve repetition, use another structure to detect and remove repetitions (Zobrist Hashing).

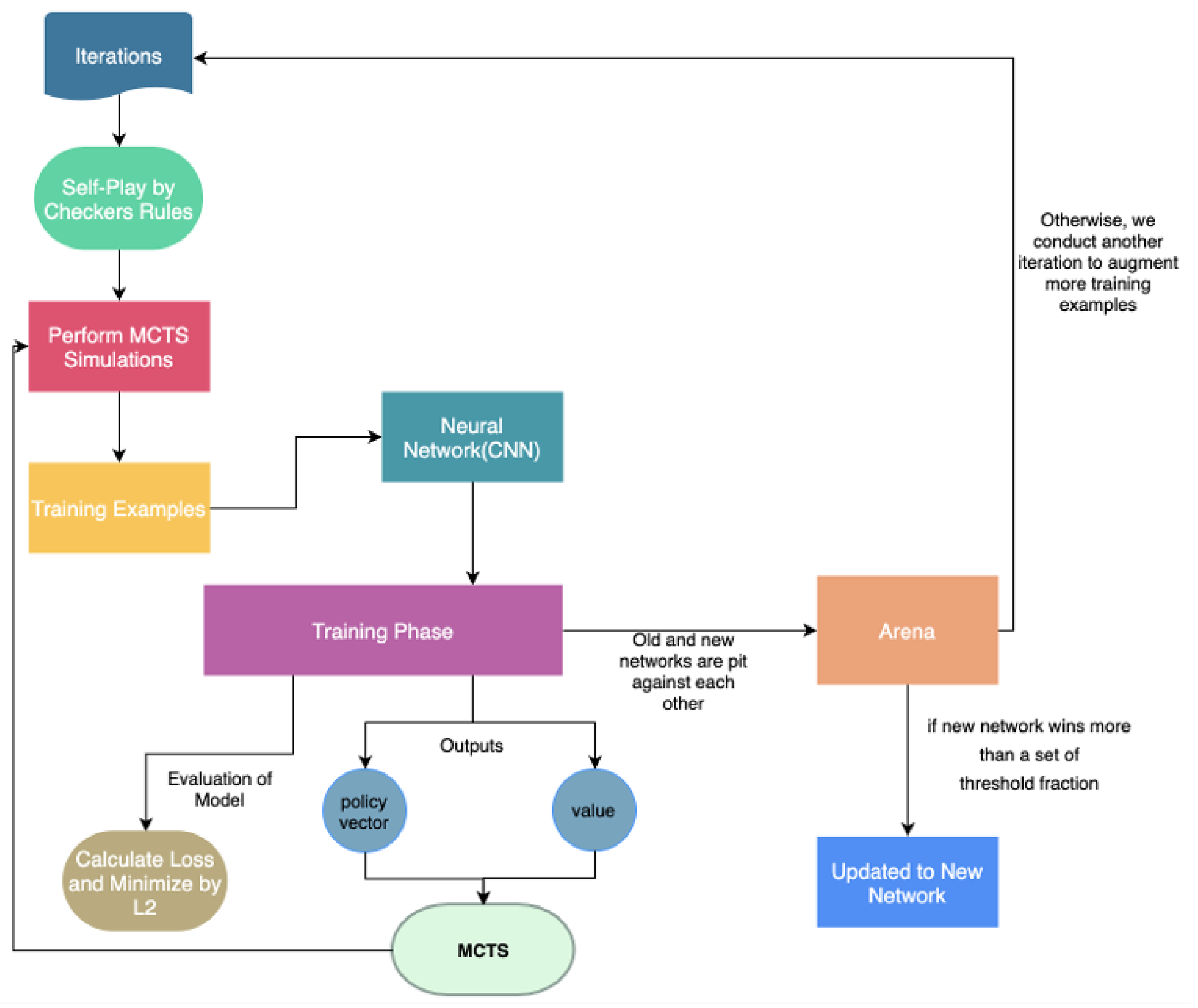
Iterations
Self-Play by Checkers Rules
Perform MCTS Simulations
Training Examples
Neural Network(CNN)
Training Phase
Old and new networks are pit against each other
Arena
Otherwise, we conduct another iteration to augment more training examples
if new network wins more than a set of threshold fraction
Updated to New Network
Outputs
policy vector
value
Evaluation of Model
Calculate Loss and Minimize by L2
MCTS

Figure 1: Flowchart of the system

METHODOLOGY

Convolutional Neural Network Approach

We used two different convolution neural network based model approaches in order to learn the checkers game better.

- *board*: current board in its canonical form
- *pi*: a policy vector for the current board- a numpy array of length
- *v*: a float in [-1,1] that gives the value of the current board

Architecture 1

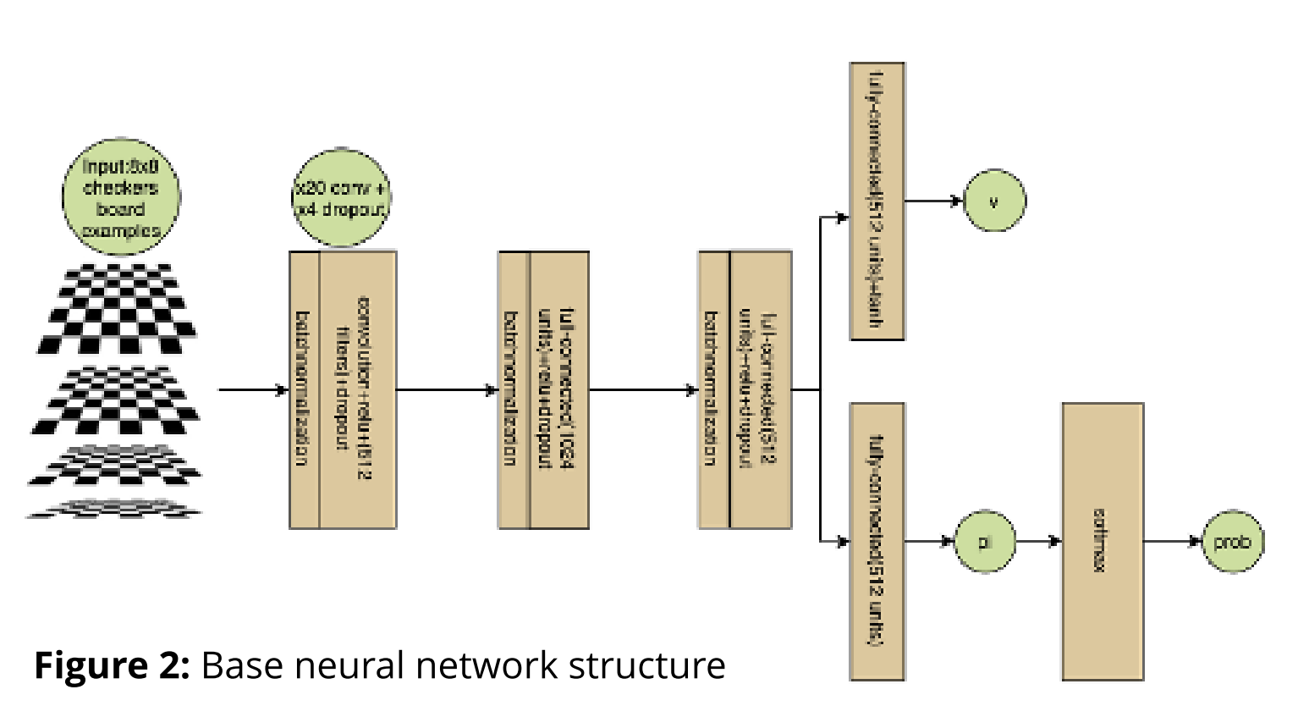
input 8x8 checkers board examples
batchnormalization
convolution+relu+(512 filters)+dropout
fully-connected(1024 units)+relu+dropout
fully-connected(512 units)+relu+dropout
tanh
v
fully-connected(512 units)
pi
softmax
prob

Figure 2: Base neural network structure

Architecture 2

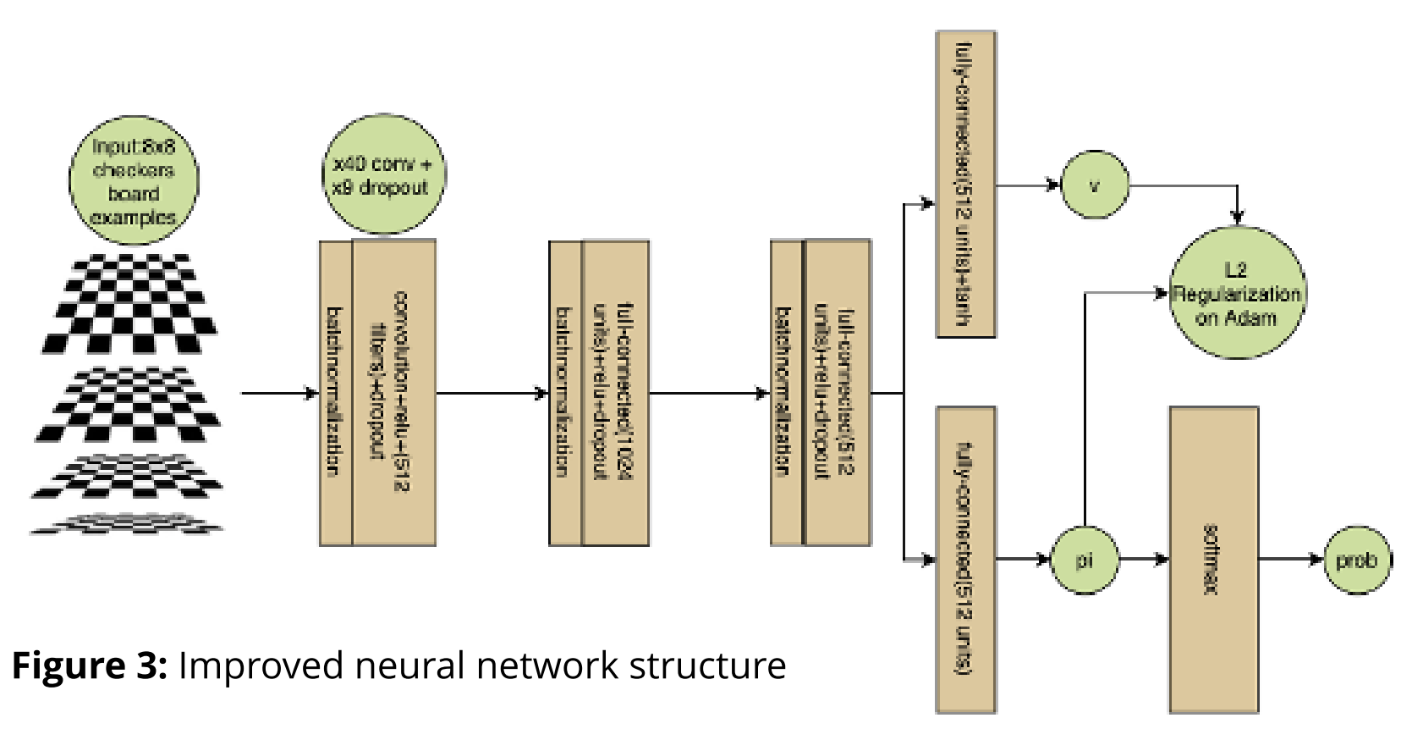

Figure 3: Improved neural network structure

Loss function (excluding regularization terms):

$$l = \sum_t (v_\theta(s_t) - z_t)^2 - \vec{\pi}_t \cdot \log(\vec{p}_\theta(s_t))$$

Formula 1: Loss function in neural network

Optimizing loss function by L2 Regularization:

$$\text{Cost} = \overbrace{\underbrace{\sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij} W_j\right)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}}^{\text{L2 Regularization}}$$

Formula 2: Loss function with regularization

Monte Carlo Tree Search

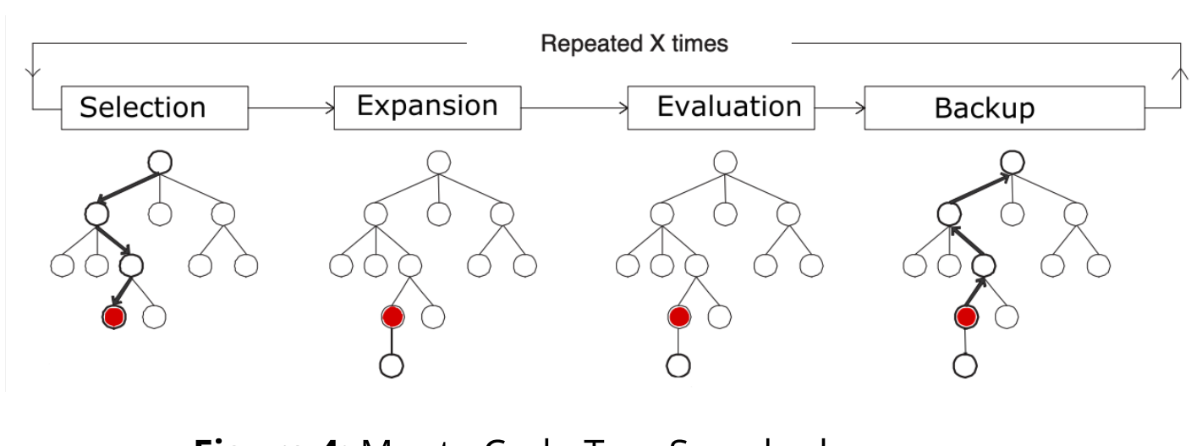

Figure 4: Monte Carlo Tree Search phases

Select: The selection function is applied recursively until a leaf node is reached. Selecting criteria is decided by UCT formula:

$$UCT(v_i, v) = \frac{Q(v_i)}{N(v_i)} + c\sqrt{\frac{log(N(v))}{N(v_i)}}$$

Formula 3: Upper confidence bound for tree

Expand: Child nodes for each action are added. In this case, only one possible action is

Simulation: Evaluation of the node from neural network.

Backpropagation: Updating the tree by backpropagating the value estimated from the leaf node.

Zobrist Hashing

Zobrist hashing used in order to find out repetition moves occurred during any play.

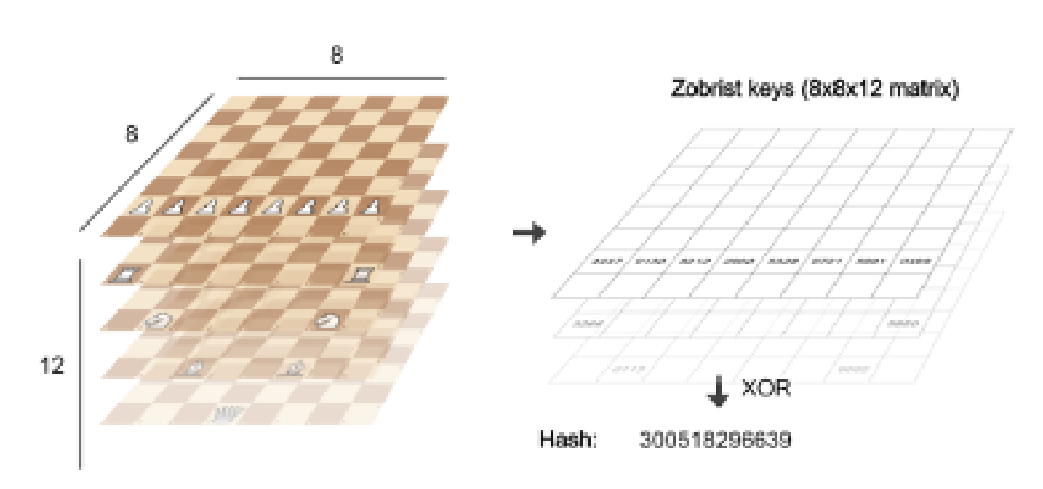

Figure 5: Zobrist hashing in a game of chess

For each piece-type corresponding to the location of that type, an integer numbers are generated, then all generated numbers put into XOR gate.

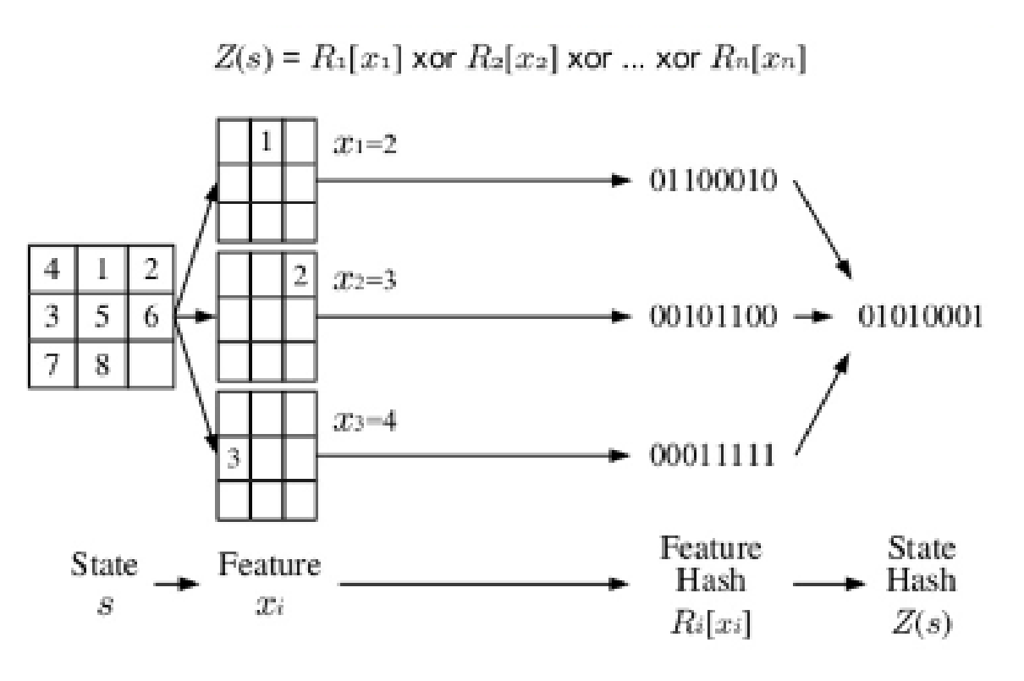

Figure 6: Zobrist hashing in application

EXPERIMENT RESULTS

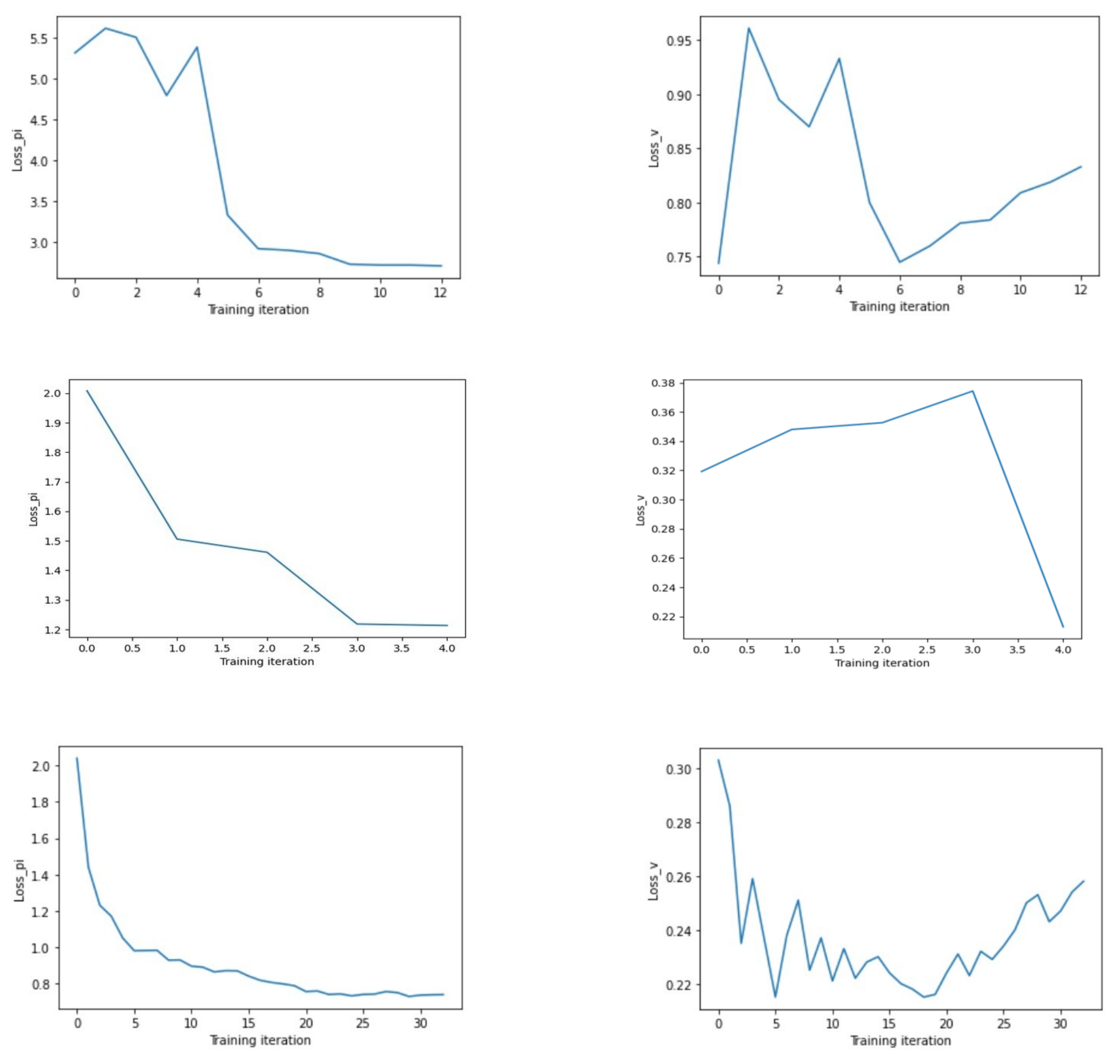

Figure 7: Different neural network architecture experiment results

Iteration Number	*Average # of nodes created*	*Average # of repetition*	*Percentage saved*
#1	16806.72	75.90	0.45
#2	49859.86	1859.90	3.73
#3	16915.34	80.57	0.47
#4	17326.12	107.97	0.62
#5	42960.84	793.85	1.84

Table 1: Different node counts over iterations.

Parameter Name	*20 Layer NNet Structure*	*40 Layer NNet Structure*
Cpuct	1.0	2.0
Number of Mcts Sims	75	100
Threshold	-	10
Number of Epochs	10	15
Batch Size	64	32
Learning Rate	0.0001	0.001
Dropout	0.3	0.4

Table 2: Parameters used with different values [2].

Conclusion

In our project, we successfully integrated Turkish Checkers into AlphaZeroGeneral which consists of several different games. While integrating the game, we encountered several different cases which are special to Turkish Checkers such as tendency to repetition in moves. In order to solve this problem, we proposed an algorithm consisting of Zobrist Hashing method, which enables us to detect these repetitions. Then, prevented those repetitions from happening.

After integration phase is completed, we embarked into optimizing the model to have better results. As we can see from the experimental results, optimization of hyper-parameters have a huge effect on the model.

For the future surveys that focuses on Turkish Checkers, we have created a good foundation.

Future Work

- Residual Network usage in modeling part with Tensorflow & Pytorch
- Hyper-parameter and general parameters tuning
- Working on finding out most suitable neural network structure for the game of Turkish Checkers.
- Since it gets better with every iteration completed, more training time could our model to be in better position
- Compete with professional players
- User friendly game user interface (UI) design

References

[1] Nair, S. Alpha Zero General. 2017, https://github.com/suragnair/alpha-zero-general

[2] Wang, H., Emmerich, M., Preuss, M., & Plaat, A. (2019). Hyper-parameter sweep on alphazero general.*arXiv preprint arXiv:1903.08129*.

[3] Silver, D., Hubert, T., Schrittwieser, J., & Hassabis, D. (2018). AlphaZero: Shedding new light on the grand games of chess, shogi and Go. *DeepMind blog*.

Used Technologies