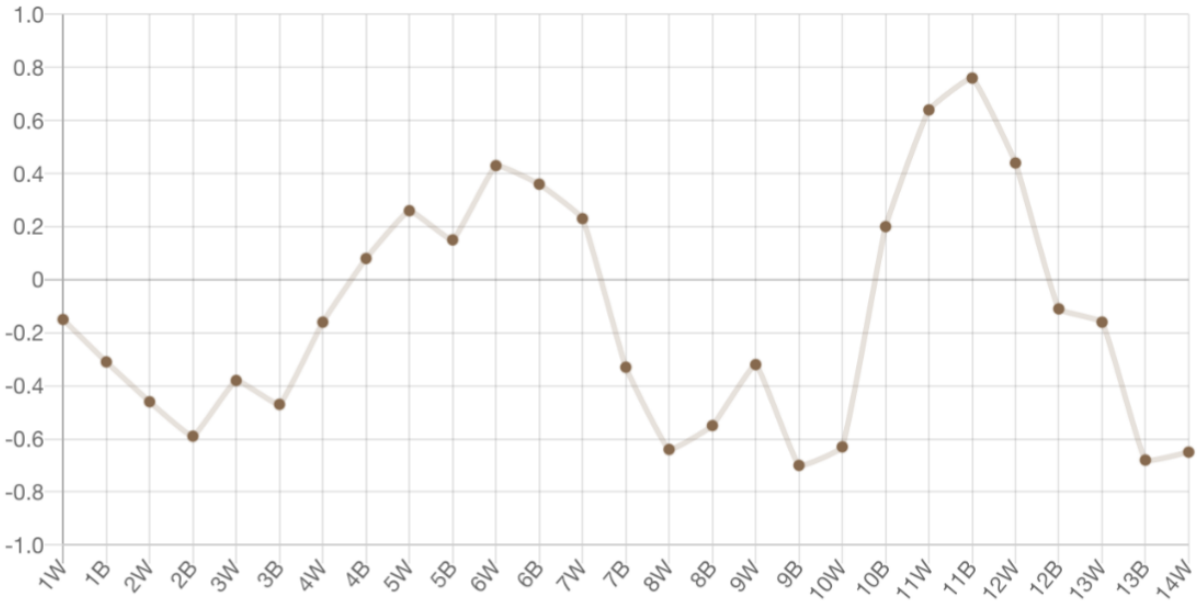
Chess Winner Prediction

A data science project to predict the winner of a chess game based on board moves, made publicly available at www.kohlenz.com/chess



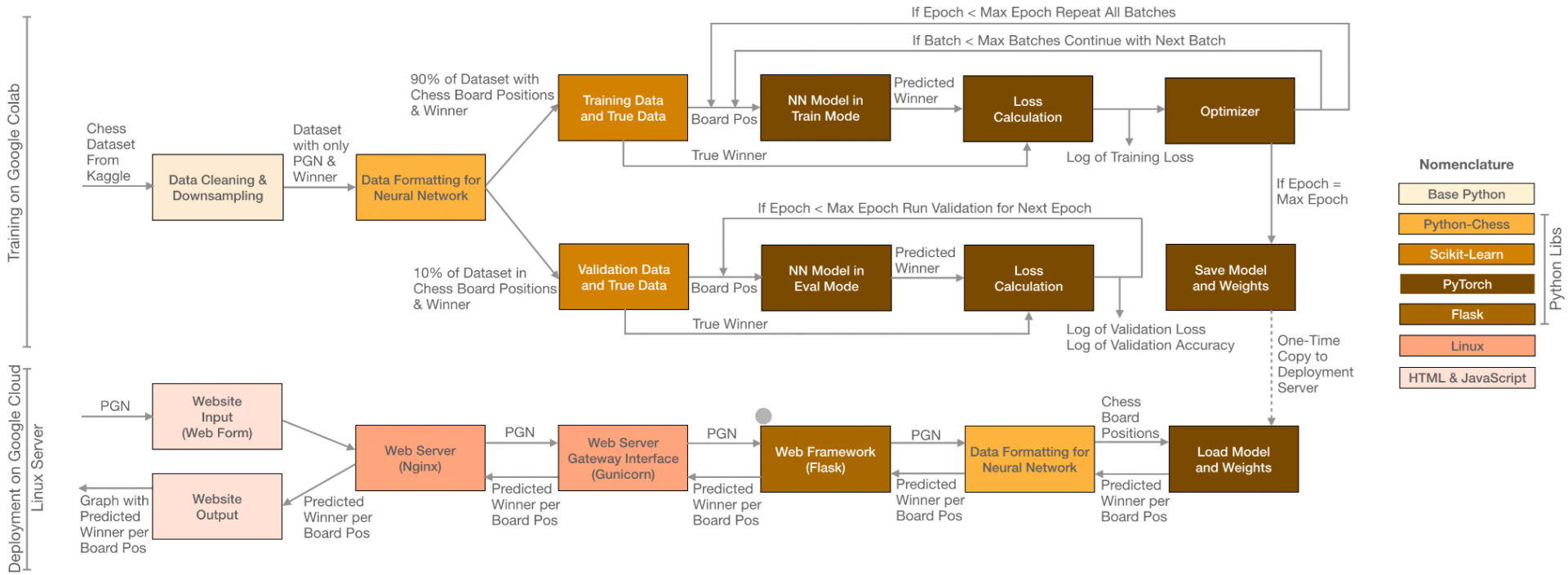
Moritz Kohlenz

10/07/2023

TABLE OF CONTENTS

METHODOLOGY

The project is split into two major parts, Training and Deployment, which are both running independently on different compute platforms. Both parts are connected through a trained model, which forms the output of Training and backbone for all computations in Deployment.



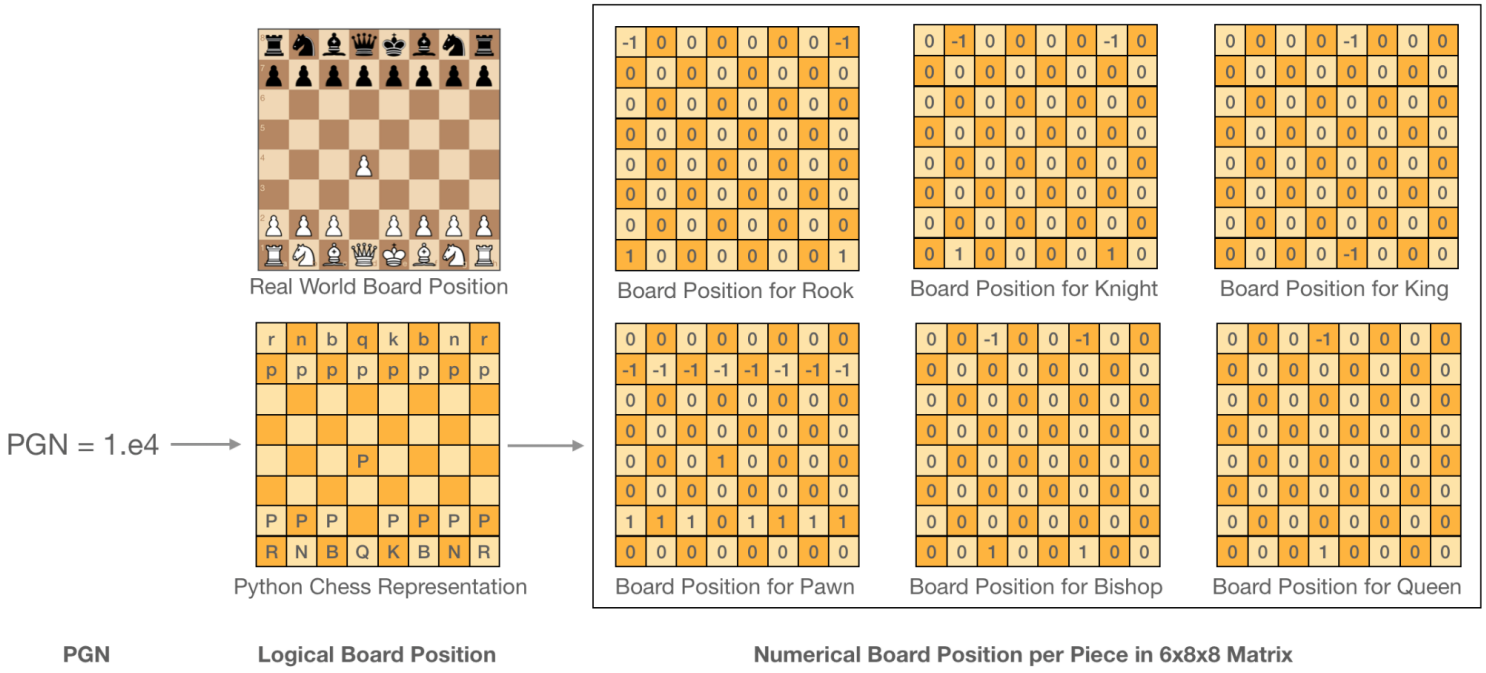
The document is structured into Training and Deployment sections explaining each of the steps in the above diagram. This is followed by a section on Model Tuning, which explains how varying parameters affect model performance thereby prediction quality.

TRAINING

The input to training is a dataset and the output is a trained model. During training, the dataset is reduced to the parameters relevant data cleaning and for training. For this project, the training data is a PGN (Portable Game Notation). The true data that the model aims to predict is the winner of the game, either black, white or a tie.

DATA CLEANING & DOWNSAMPLING

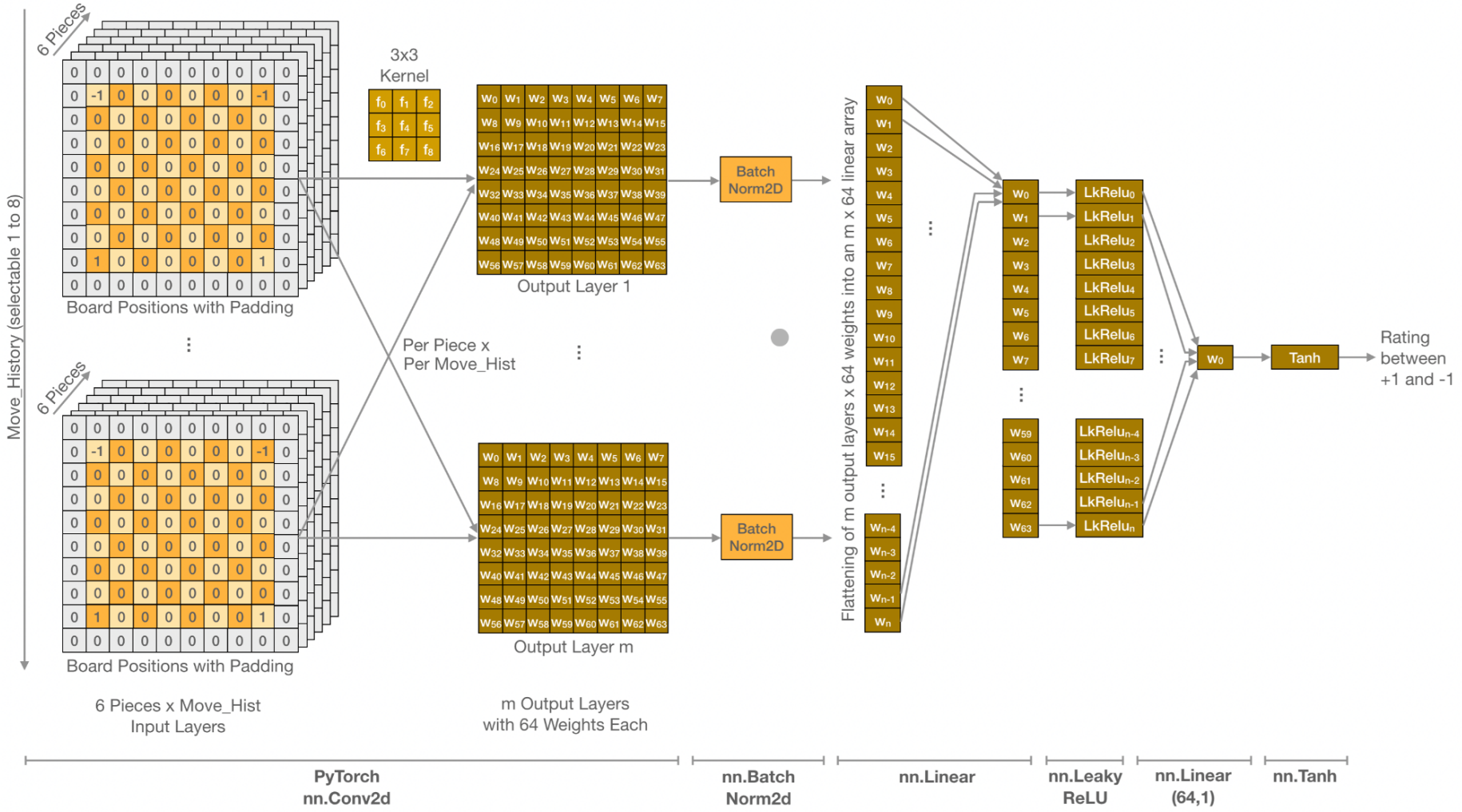
DATA FORMATTING FOR NEURAL NETWORK (NN)



TRAINING AND VALIDATION DATA SPLIT

NEURAL NETWORK (NN) MODEL

Rename Output Layer to Output Channel and Input Layer to Input Channel



To keep the diagram simple, biases are not shown in the diagram.

Even though the number of parameters of Conv2d are less than in linear, in terms of computations, there are 64 Weights in each nn.Conv2d Output, therefore the number of

Model	Complexity In General Formulas	Complexity In Model Parameters
nn.Conv2d	(Number_of_Input_Channels x Number_of_Output_Channels x Kernel_Size + Number_of_Output_Bias) x Number_of_Outputs per Output_Channel	(6 Pieces x 8 Move_History x 6 Output_Channel x 3x3 Kernel + 6 Bias per Output_Channel) x 8x8 Outputs per Output_Channel = **2598 x 64 Parameters**
nn.BatchNorm2d	Number_of_Outputs x BatchNormDimension	6 Output_Layer x 2 Dimensions = **12 Parameters**
nn.Linear(384,64)	Number_of_Inputs x Number_of_Outputs + Number_of_Output_Bias	(6 Output_Layer x 8x8 Parameters per Output_Channel) x 64 Outputs + 64 Bias per Output = **24640 Parameters**
nn.Linear(64,1)	Number_of_Inputs x Number_of_Outputs + Number_of_Output_Bias	64 Inputs x 1 Output + 1 Bias per Output =

		65 Parameters

LOSS CALCULATION

OPTIMIZER

SAVE MODEL AND WEIGHTS

DEPLOYMENT

WEBSITE INPUT

WEB SERVER

WEB SERVER GATEWAY INTERFACE (WSGI)

WEB FRAMEWORK

DATA FORMATTING FOR NEURAL NETWORK (NN)

LOAD MODEL AND WEIGHTS

WEBSITE OUTPUT

MODEL TUNING

HYPER PARAMETERS

EVALUATION PARAMETERS

Main parameter is validation accuracy. Model has never seen this data.

Validation loss is the output of the loss function based. Prediction is run validation data that the model has not used for training.

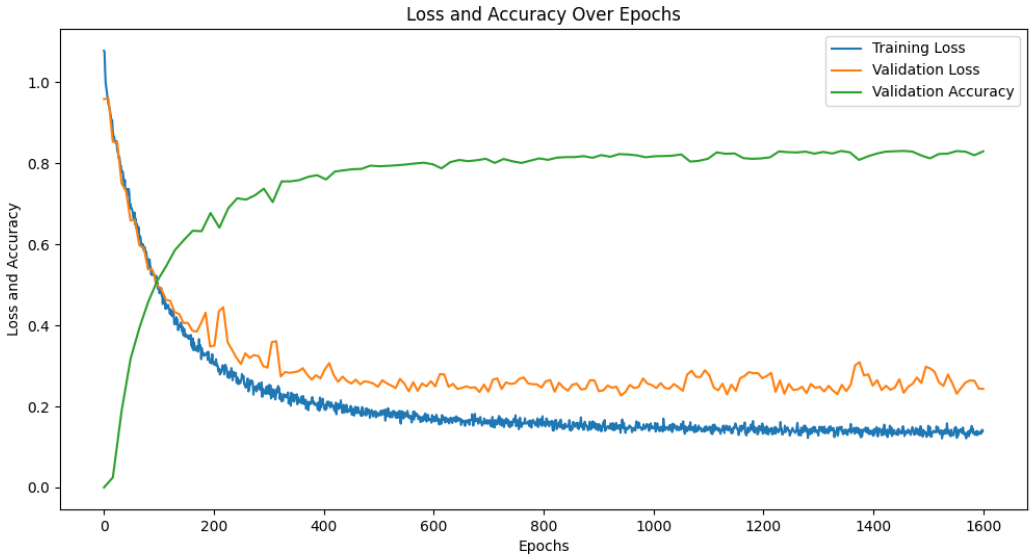
Training loss is the output of the loss function. Prediction is run on the data that it is trained on. Counterintuitive why this parameter is used. Used for spotting overfitting

DATA PREPARATION PARAMETER EVAL

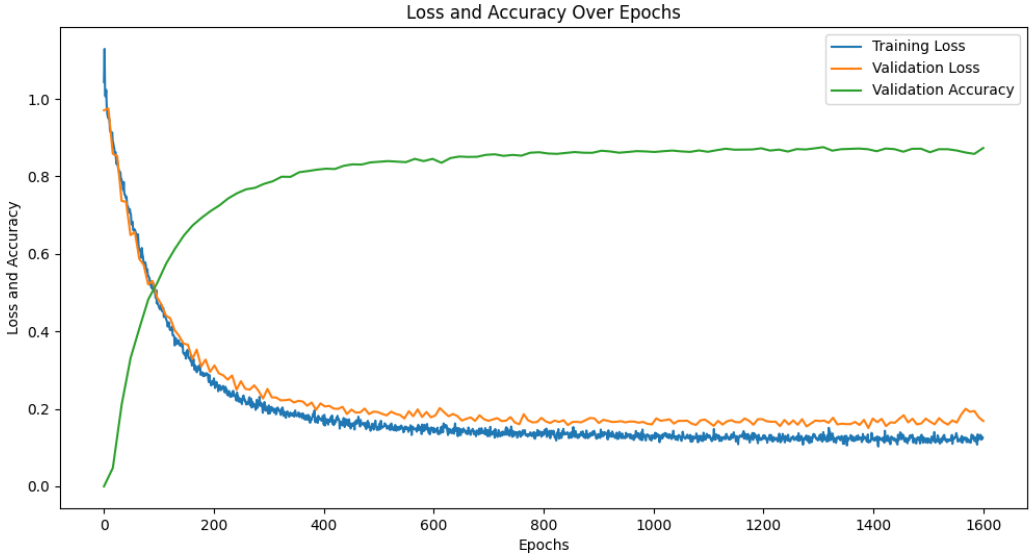
Move_History

Model	Number of Games	Move History	Remove Ties	Max Epochs	Validation Accuracy
Baseline	500	1	Yes	100	**82.9%**
Baseline	500	2	Yes	100	**84.5%**
Baseline	500	4	Yes	100	**87.3%**
Baseline	500	6	Yes	100	**87.5%**
Baseline	500	8	Yes	100	**87.7%**

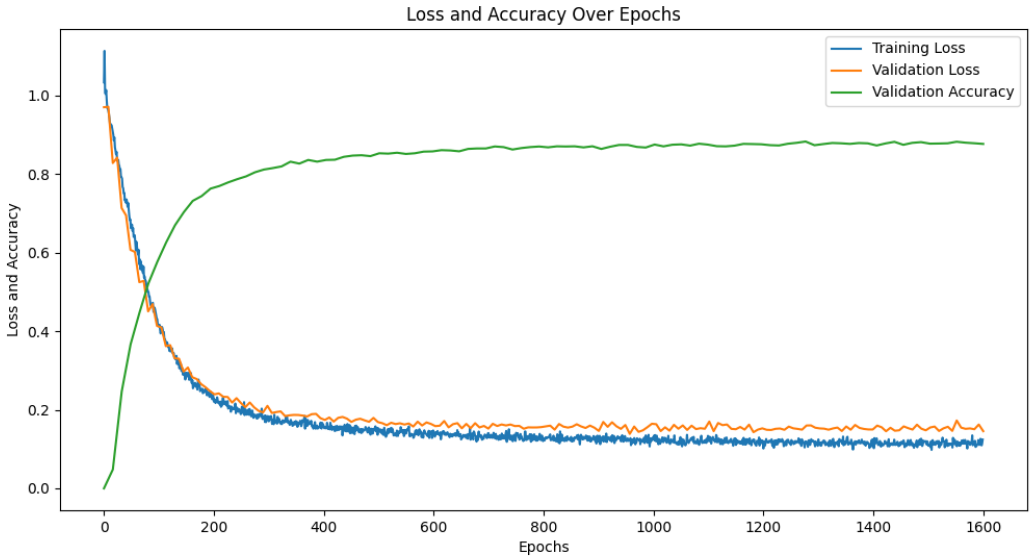
Move_History = 1, Training_Loss = 13.5%, Validation_Loss = 24.3%



Move_History = 8, Training_Loss = 11.5%, Validation_Loss = 15.4%



Move_History = 8, Training_Loss = 12.3%, Validation_Loss = 17.3%



DATA SELECTION PARAMETER EVAL

Remove Ties

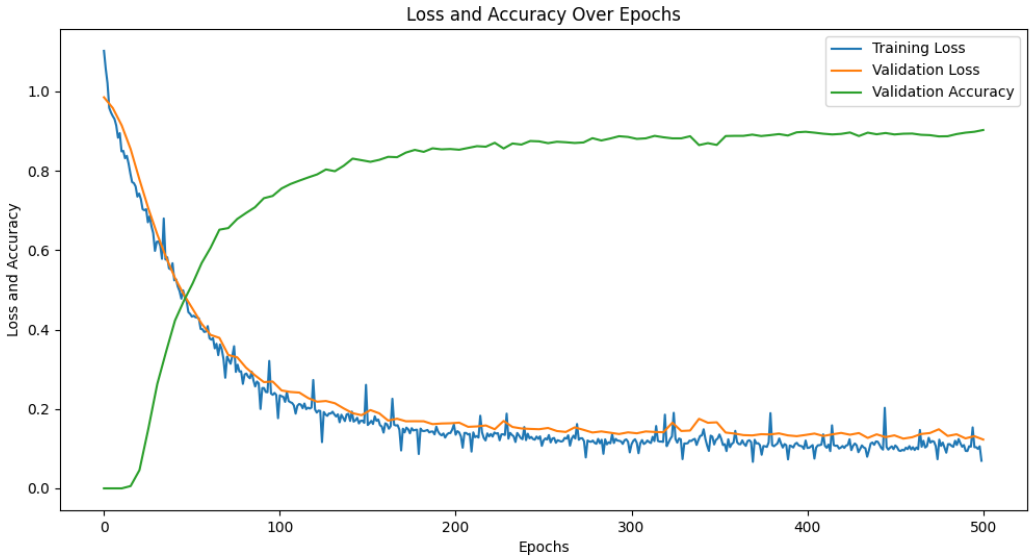
Model	Number of Games	Move History	Remove Ties	Max Epochs	Validation Accuracy
Baseline	500	4	No	50	**81.3%**
Baseline	500	4	Yes	50	**85.7%**

Number of Games

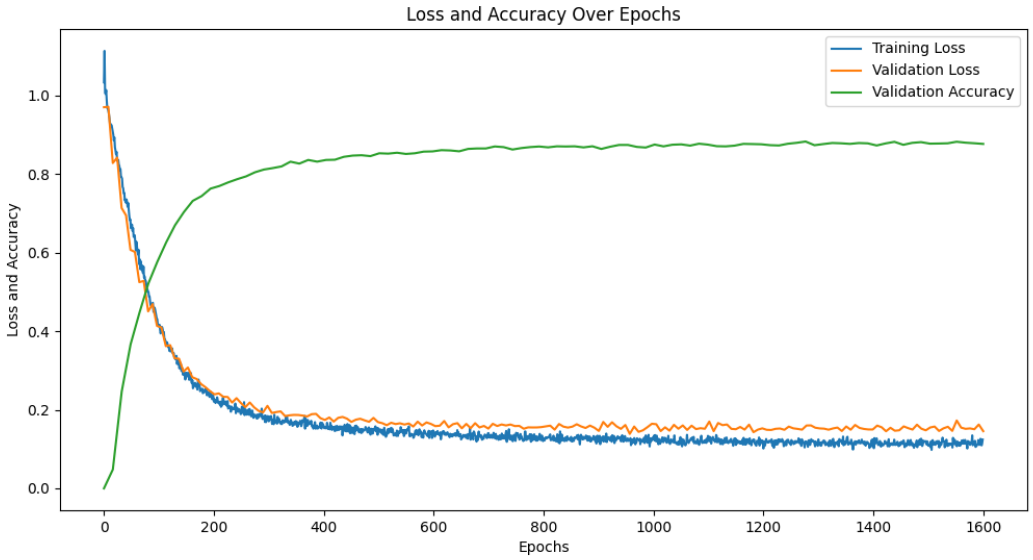
Model	Number of Games	Move History	Remove Ties	Max Epochs	Validation Accuracy
Baseline	125	8	Yes	100	**90.2%**

Baseline	250	8	Yes	100	**89.0%**
Baseline	500	8	Yes	100	**87.7%**
Baseline	1000	8	Yes	100	**85.1%**
Baseline	2000	8	Yes	100	**80.5%**

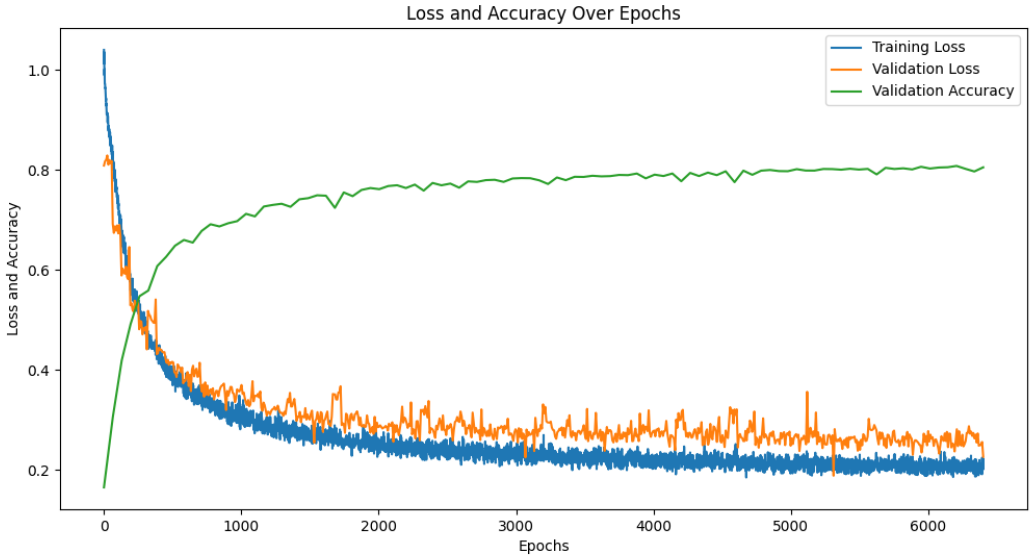
Number_of_Games = 125, Training_Loss = 9.6%, Validation_Loss = 12.3%



Number_of_Games = 500, Training_Loss = 11.5%, Validation_Loss = 15.4%



Number_of_Games = 2000, Training_Loss = 20.4%, Validation_Loss = 25.4%



MODEL EVAL

NEURAL NETWORK (NN) MODEL

Group by Piece collapses 6 pieces into one input for a move history of 8

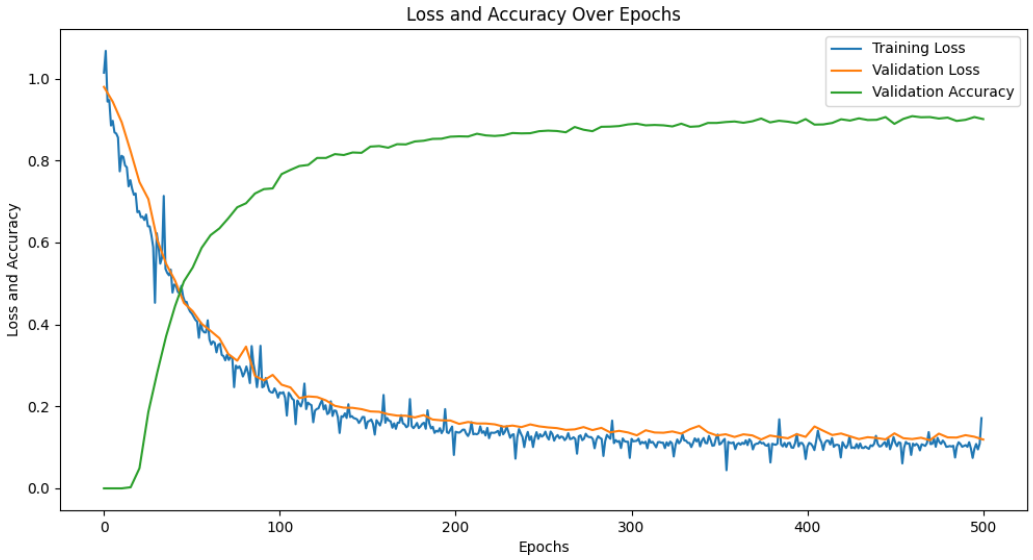
Group by Move collapses 8

Input_Channels need to be divisible by Output_Channels, hence adjustment in Output_Channels

Model	Complexity In Model Parameters
Baseline	2598 x 64 for Conv2D + 12 for BatchNorm2D + 20,640 for Linear(384,64) + 65 for Linear(64,1) = **186,989 Parameters**
Additional Conv2D Layer	(6 Pieces x 8 Move_History x 6x8 Output_Layer x 3x3 Kernel + 6x8 Bias per Output_Channel) x 8x8 Outputs per Output_Channel for Conv2D + 96 for BatchNorm2D + 27,315 Parameters for Baseline = **1,357,587 Parameters**
Group by Piece	(8 Move_History x 6 Output_Channel x 3x3 Kernel + 6 Bias per Output_Channel) x 8x8 Outputs per Output_Channel for Conv2D + 12 for BatchNorm2D + 20,640 for Linear(384,64) + 65 for Linear(64,1) = **48,749 Parameters**
Group by Move_History	(6 Pieces x 8 Output_Channel x 3x3 Kernel + 8 Bias per Output_Channel) + 8x8 Outputs per Output_Channel for Conv2D + 16 for BatchNorm2D + 32,832 for Linear(512,64) + 65 for Linear(64,1) = **61,073 Parameters**

Model	Number of Games	Move History	Remove Ties	Max Epochs	Validation Accuracy
Baseline	125	8	Yes	100	**90.1%**

Additional Conv2D Layer	125	8	Yes	100	**88.9%**
Group by Piece	125	8	Yes	100	**89.2%**
Group by Move_History	125	8	Yes	100	**88.3%**



An interesting result is that the number of model parameters is no indicator for validation accuracy

CONCLUSION

Solid prediction quality

Sample set small

Not good that validation accuracy decreased with increasing the number of games. This points to similar move sequences resulting in different outcomes. Further study could be done by selecting shorter games with dominant player based on ELO. Another way would be to train or even more games until a clearer picture emerges.

REFERENCES

References

Kaggle. 2019. “3.5 Million Chess Games.”

https://www.kaggle.com/datasets/milesh1/35-million-chess-games.