Basketball Game Predictor

Team 17: Alec Sommerhauser, Tyler Cox, Luis Servin-Perez, Ethan Rule

Project Goal

Given historical NBA play by play data:

- Predict the outcome of a game in the format of a box score. (game statistics)
- Determine which team wins.

Solution Approach

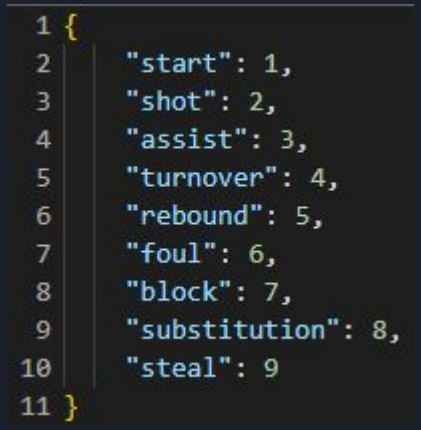
- Using NBA play-by-play data from 2000 to 2023 train an ensemble of LSTM models to predict the outcome of a game.
- Each model will predict a piece of the next play in the game.
- A list of plays represent a game. Together the models generate a game.
- Analyze the list of generated plays to decide a winner and fill out a box score.

Data Cleaning

- Convert csv data to json and extract vocabs.
- Extracting vocabs allowed us to feed data into the models in terms of integers mapped to a specific data type.

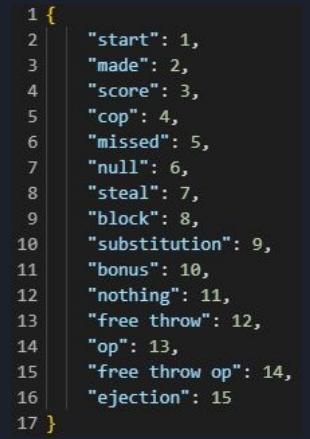
Examples:

Event Vocab



```
{
    "start": 1,
    "shot": 2,
    "assist": 3,
    "turnover": 4,
    "rebound": 5,
    "foul": 6,
    "block": 7,
    "substitution": 8,
    "steal": 9
}
```

Result Vocab



```
{
    "start": 1,
    "made": 2,
    "score": 3,
    "cop": 4,
    "missed": 5,
    "null": 6,
    "steal": 7,
    "block": 8,
    "substitution": 9,
    "bonus": 10,
    "nothing": 11,
    "free throw": 12,
    "op": 13,
    "free throw op": 14,
    "ejection": 15
}
```

What to predict at each step?

- “roster1”
- “roster2”
- “time”
- “event”
- “player”
- “type”
- “result”

The 7 Model Ensemble

- Event / Time Predictor Model
- Player Predictor Model
- Shot Type Predictor Model
- Shot Result Predictor Model
- Foul Type Predictor Model
- Turnover Type Predictor Model
- Substitution Predictor Model

Model Layers

- Embedding layers
  - For inputs with lots of noise makes it more simple for the model to understand (player, roster, type). This converts data into a more dense lower dimensional representation.

- LSTM
  - RNN that helps model remember long term historical data. Traditional RNNs struggle with with vanishing gradients while a LSTM can preserve historical context.
  - Uses the Relu activation function

- Multihead Attention Layer
  - Used for comparing the inputs impact on the result against each other (used in transformers). This layer enables the model to focus on different parts of the input at the same time.

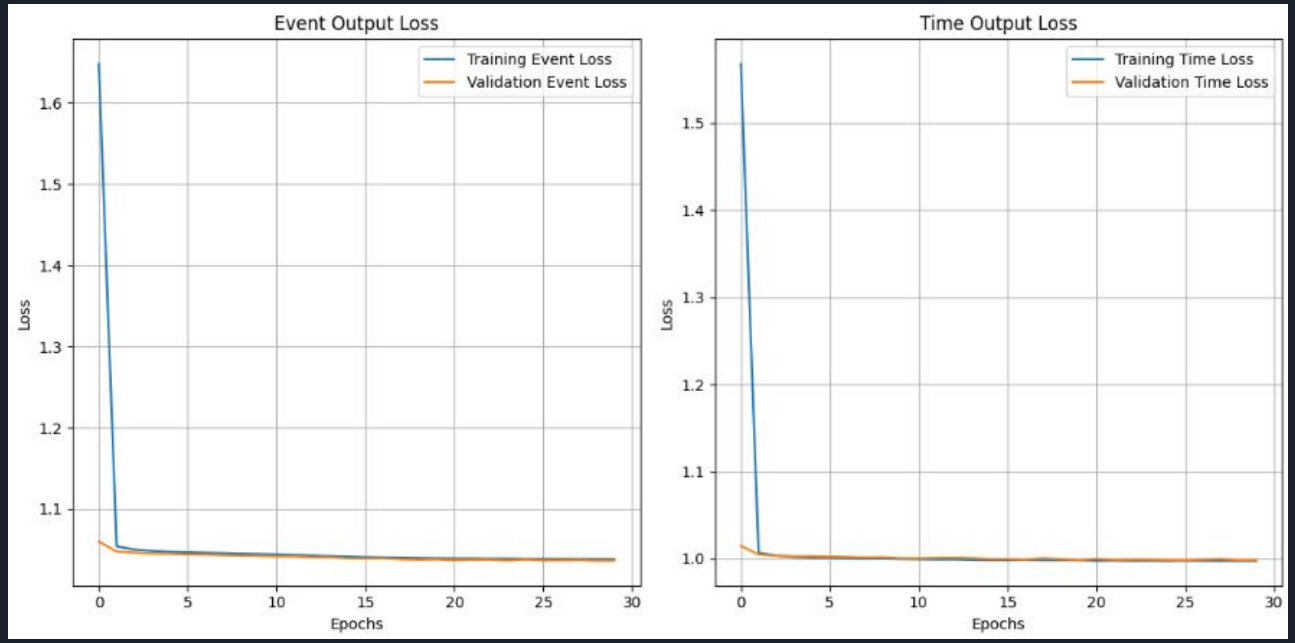
Input

- Roster1
- Roster2
- Time
- Event
- Player
- Type
- Result
- Season
- Playoff

```
# Input Layers
roster1 = tf.keras.layers.Input(shape=(800, 1), name='roster1')
roster2 = tf.keras.layers.Input(shape=(800, 1), name='roster2')
time = tf.keras.layers.Input(shape=(800, 1), name='time')
event = tf.keras.layers.Input(shape=(800, 1), name='event')
player = tf.keras.layers.Input(shape=(800, 1), name='player')
event_type = tf.keras.layers.Input(shape=(800, 1), name='type')
result = tf.keras.layers.Input(shape=(800, 1), name='result')
season = tf.keras.layers.Input(shape=(800, 1), name='season')
playoff = tf.keras.layers.Input(shape=(800, 1), name='playoff')
```

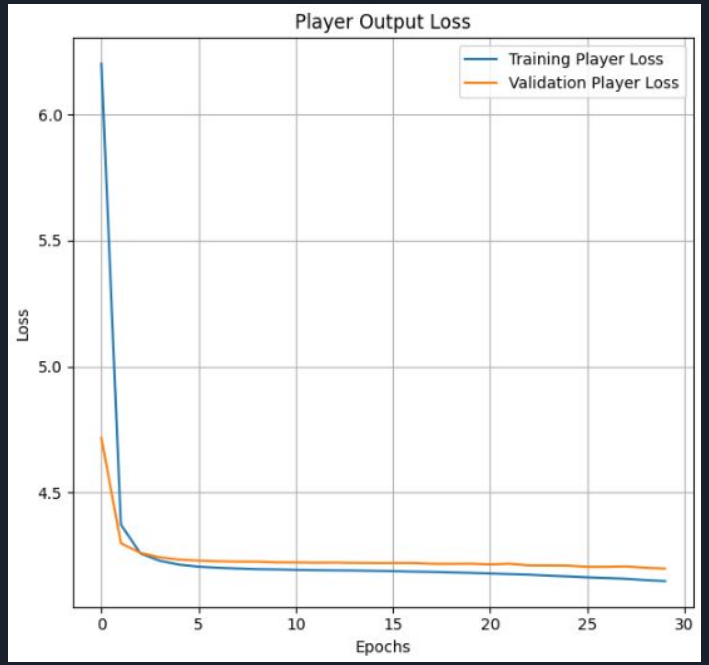
Event / Time Predictor Model

- Predicts the next event / time in a game.



Player Predictor Model

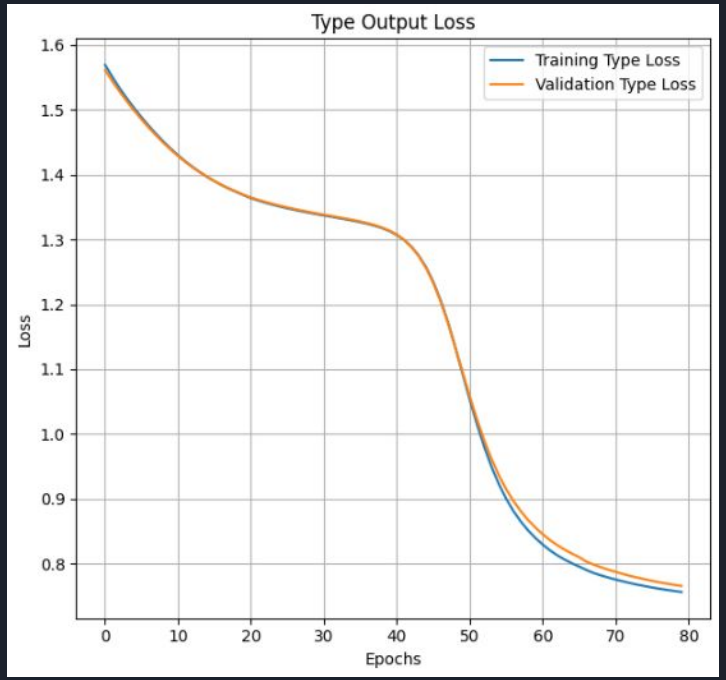
- Predicts the player in the event.



Shot Type Model

- If the next event is a shot, find the type.
  - e.g. “free throw”, “3pt”, “2pt”

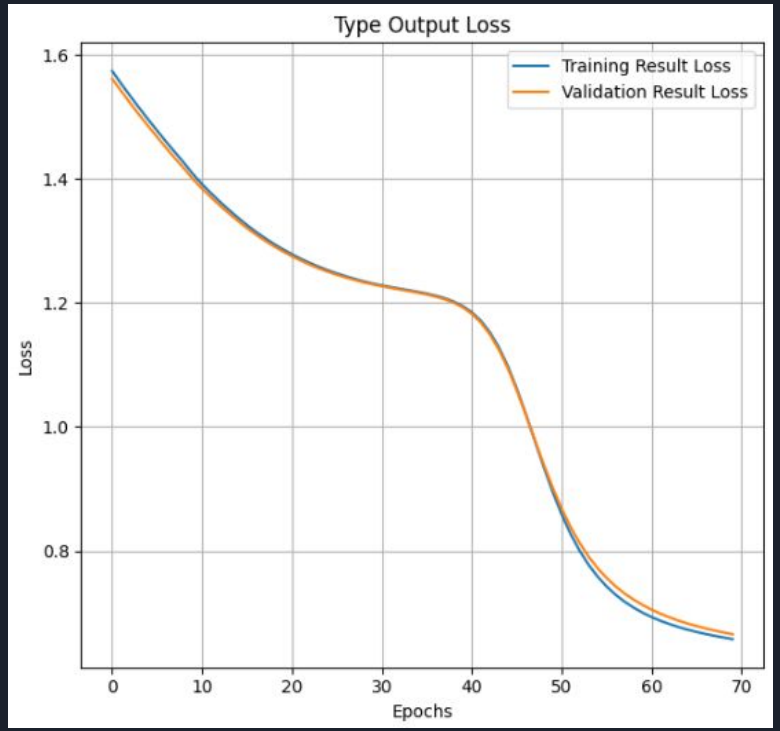
- Accuracy: 0.602



Shot Result Model

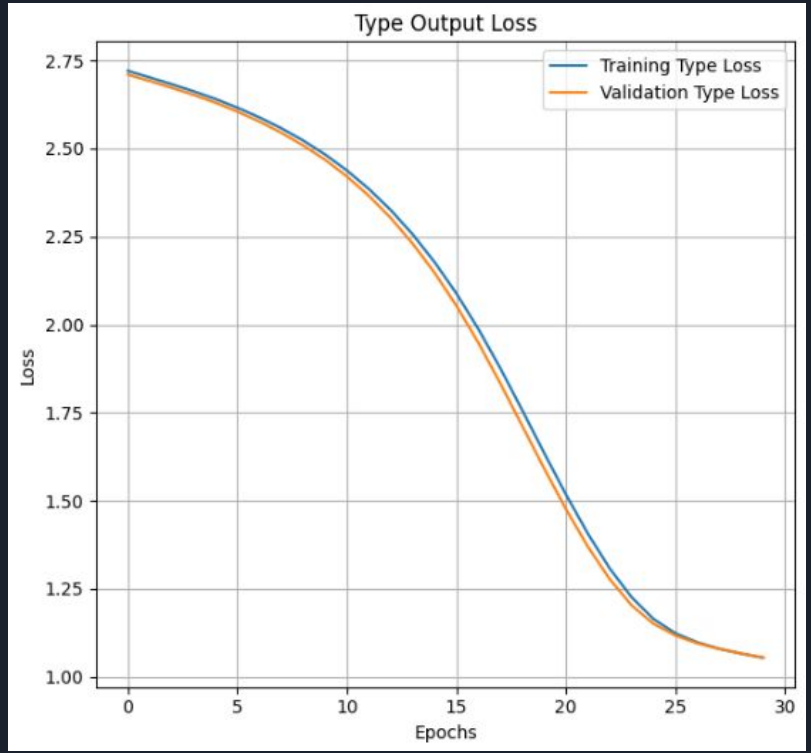
- Predict the result of a shot.
  - e.g. “made” or “missed”

- Accuracy: 0.501



Foul Type Model

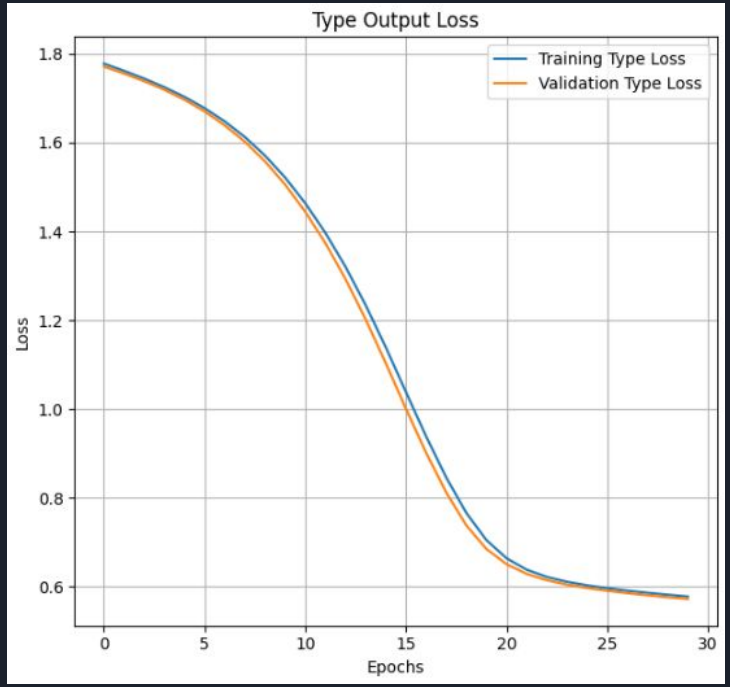
- Predict the type of a foul.
  - e.g. “personal”, “shooting”



Turnover Type Model

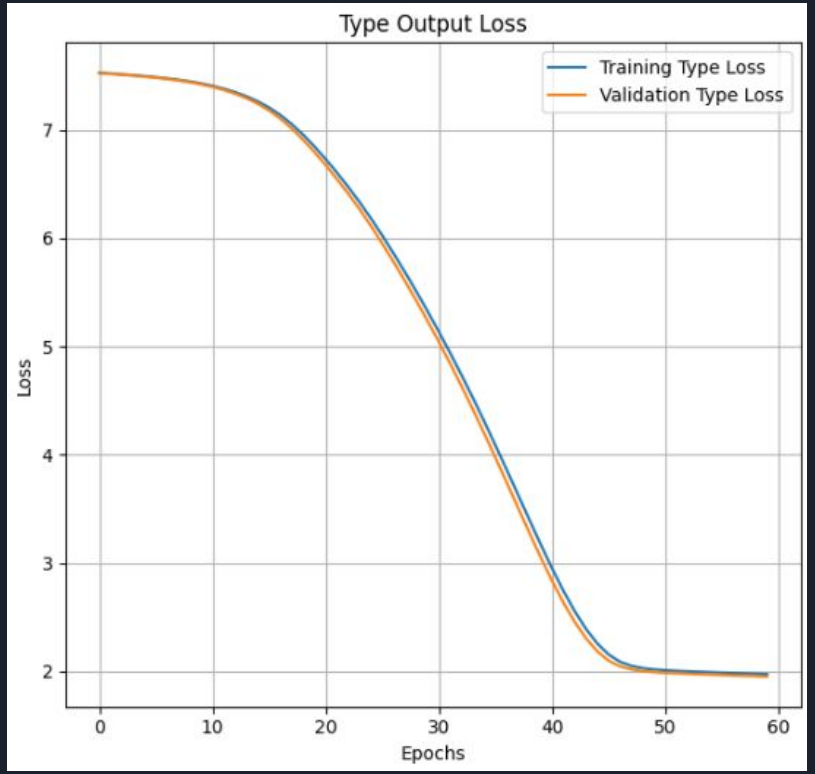
- Predict turnover type

- Accuracy
  - 0.390



Substitution Model

- Predict player substitutions



Controller

- Uses the models in accordance to NBA rules
- Generates a game using the model’s predictive tools
- WORK IN PROGRESS

What’s Next:

- Build the controller
- Predict games

Current Model Evaluations

- Models do a good job of making a probability distribution for a given task, but that distribution is too general (predicts at the rate of occurrence).
- It's possible that when all the models are put together, the accuracy could increase.
- The model predicts better than predicting at random.

Expected Results

- Between 50 - 60% success, should be slightly better than choosing at random
- The goal was 70% (rate of an NBA upset)
- We predict you could probably beat the model by choosing the team that is higher in the standings.
- With the controller, it’s possible that very small minute difference in the predicted probability distributions will lead to far better results than expected when used at a large scale.

Future Improvements

- Hyper specializing models for increased certainty.
- More models for specific tasks. (Sub models for each current model)
- Consider using transformers.
- Consider Seq2Seq encoder decoder model.