ATM Post Season

# 1. Feature Engineering:

## Incorporate Domain Knowledge**:**

Create new features that capture more context about the games. For example:

* + **Team Form:** Add features that track the recent performance of each team (e.g., last 5 games' results).
  + **Head-to-Head Records:** Incorporate historical performance data between the two teams.
  + **Venue Impact:** Include features that capture how well teams perform at specific venues.
  + **Injury Reports/Player Availability:** If available, this could significantly influence the outcome of games.
  + **Weather Conditions:** Incorporate weather data, as it can influence game dynamics.

# 2. Model Complexity and Regularization:

## Simplify the Neural Network Architecture:

* + The current network is quite deep and may be overfitting, especially if the dataset isn't large. Try reducing the number of layers or neurons per layer.
  + Alternatively, experiment with fewer layers or smaller layer sizes to see if a simpler model generalizes better.

## Adjust Dropout Rates:

* + Dropout is currently set at 50% for each layer. This might be too aggressive, leading to underfitting. Consider lowering the dropout rate to 20-30% or using it selectively on only some layers.

# 3. Optimizer Tuning:

## Switch to a Different Optimizer:

* + Replace SGD with a more adaptive optimizer like **Adam** or **RMSprop**. These optimizers tend to converge faster and might find a better minimum in the loss landscape.

# 4. Hyperparameter Tuning:

## Systematic Hyperparameter Tuning:

* + Use GridSearchCV or RandomizedSearchCV to systematically tune key hyperparameters like learning rate, dropout rate, number of layers, and batch size. This will help you find the best model configuration.

# 5. Address Class Imbalance:

## Handle Class Imbalance:

* + If the target classes (BW, LW, D, LL, BL) are imbalanced, the model might be biased towards predicting the majority class. Consider:
    - Using class weights in the loss function to give more importance to minority classes.
    - Oversampling the minority classes or undersampling the majority class in the training set.

# 6. Evaluation Metrics:

## Use Multiple Metrics:

While accuracy is a useful metric, also consider using precision, recall, and F1-score, especially if class imbalance is present. These metrics can provide more insight into the model's performance across different classes.

# 7. Ensemble Learning:

## Blend Models:

Consider combining the neural network with other machine learning models (e.g., decision trees, gradient boosting machines). Ensemble methods often perform better than single models because they can capture different aspects of the data.

# 8. Cross-Validation:

## Use Cross-Validation:

Instead of a simple train-test split, use k-fold cross-validation to get a more reliable estimate of the model's performance. This helps in understanding how well the model generalizes to unseen data.

# 9. Model Interpretability:

## Implement Model Explainability Techniques:

* + Use tools like **SHAP** or **LIME** to understand which features are contributing most to the model’s predictions. This can provide valuable insights into model behavior and guide further feature engineering.

# Implementation Plan:

1. **Start with Feature Engineering:** Develop new features based on domain knowledge. This will likely have the most immediate impact.
2. **Simplify the Model Architecture:** Begin with a simpler network and gradually add complexity if needed.
3. **Switch to Adam or RMSprop:** Replace SGD with one of these optimizers to improve convergence.
4. **Tune Hyperparameters:** Use a grid or random search to find the best configuration.
5. **Handle Class Imbalance:** Adjust the dataset or model training process to better handle imbalanced classes.
6. **Evaluate with Cross-Validation:** Implement k-fold cross-validation to better assess model performance.

# Additional

Adding in next weeks fixture to be able to click a button and have all 9 games predictions output at once