RNN-Based Pitch Type Prediction: Insights into Sequential Patterns and Strategy Refinement

Song-Ze, Yu*, Yun-Zhong, Lai*, Yi-Sheng, Xiao*, Li-Tzu, Yang*, Po-Hsun, Tseng*

***Abstract*—This project explores the use of recurrent neural network (RNN) models to predict the next pitch type in baseball. By applying data slicing techniques and model ensemble methods, including a generalist-specialist voting mechanism, our approach improves predictions for less frequent pitch types. The best-performing model achieved an accuracy of 69%, significantly surpassing the Bayesian baseline at 55%. This study demonstrates the potential of RNN-based methods in uncovering hidden pitch patterns, providing practical tools for pitchers, coaches, and catchers to refine strategies and improve performance.**

***Index Terms*—RNN model, long-short-term memory, attention layer.**

I. Introduction

It is an era where baseball is heavily based on data science, but the methods used recently still relies heavily on mathematical statistics, and we think it may overlook some important abstract features like game context or pitching tendency. To address this, we used RNN models to learn the relationship between pitches, which combing data slicing techniques and model ensemble approaches. By comparing four different versions of models, the one which includes a generalist and a specialist model enhance predictions particularly for less frequent pitch types and have the best result.

II. Methods

A. Data Preprocessing

This study analyzed three pitchers. The first is Shota Imanaga, who is a starting pitcher. The second is Hunter Greene, who is also a starting pitcher. The last is Mason Miller, who is a relief pitcher. All pitch data were collected from the 2024 season. The data were obtained using the JIDBC pybaseball library, which provides functions to pull pitching data.

To train a robust deep neural network, a sufficient amount of data is usually required. However, a regular starting pitcher typically has only around 2,000 pitches per season. Since pitching is a sequential process, this study slices the pitching sequences of each plate appearance into subsequences to increase the number of training samples. Specifically, a pitch sequence with a length of 3 {1,2,3} will be sliced into the following subsequences: {1} to predict {2}, {2} to predict {3}, and {1, 2} to predict {3} .This approach preserves the relationship between pitches while increasing the dataset size, allowing for better analysis and prediction of pitch sequences. In this study, two methods were used to identify a sequence:

1) **Date-based approach**: All pitches in a single game are considered as one sequence. For this approach, the minimum subsequence length is 20, and the maximum subsequence length is the sequence length minus one.
2) **Batter-based approach**: Each plate appearance is regarded as a sequence. For this approach, the minimum subsequence length is 1, and the maximum subsequence length is the sequence length minus one.

This approach allows the study to effectively expand the dataset while maintaining the sequential relationships between pitches, thereby facilitating better predictions and insights into pitch patterns.

B. Feature Selection

The features used in this study include both numerical and categorical data, prepared through normalization, one-hot encoding, and embedding techniques. Normalized features include release_speed(0–105 mph), launch_speed(0–120 mph), launch_angle (-90 to 90 degrees), and hit_distance (0–400 feet), pitch_number(sequential number of the pitch in the current game), post_bat_score(the score of the batting team after the pitch is completed), post_fid_score(the score of the batting team after the pitch is completed), and outs_when_up(the number of outs recorded at the moment the batter steps up to the plate). Categorical features, processed through one-hot encoding, consist of pitch_type, number of strikes, number of balls, zone(14 unique zones including strike and surrounding areas), description (11 unique outcomes such as "called strike" or "foul ball"), bb_type (the type of batted ball outcome, which is categorized into four types: ground ball, fly ball, line drive, and popup), fielder_2(the defensive player positioned as the catcher), and inning(the current inning of the game). Additionally, batter is represented as an embedded feature using a unique integer ID for each player to capture latent patterns effectively. These features, carefully preprocessed, provide a robust dataset for training predictive models. The last, we also calculate pitch frequency of 2023 season and add it into features.

C. RNN Model

Recurrent Neural Networks (RNNs) are designed to process sequential data, making them particularly suited for tasks involving temporal dependencies, such as predicting the next pitch type in baseball. RNNs leverage their hidden state to retain information about previous inputs, enabling the model to learn relationships within the sequence of pitches.

For our project, we implemented a RNN model to predict pitch types. The architecture, as depicted in Fig. 1, consists of the following layers:

- **Masking Layer**: Handles variable-length input sequences by ignoring padded values, ensuring that the model focuses only on meaningful data.
- **SimpleRNN Layer**: A recurrent layer with 64 units, capturing sequential dependencies in the data.
- **Layer Normalization**: Stabilizes the training process by normalizing activations within each layer.
- **Dense Layer (ReLU)**: A fully connected layer with 32 units and ReLU activation, helping the model to extract higher-level features.
- **Dropout Layer**: Reduces overfitting by randomly setting a fraction of the layer's inputs to zero during training.
- **Batch Normalization**: Stabilizes training and improves convergence by normalizing activations after the Dropout layer.
- **Output Dense Layer (Softmax)**: Produces probability distributions over the possible pitch types.

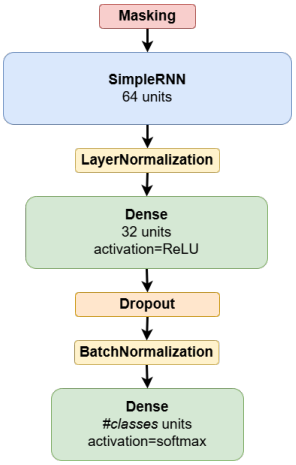

Fig. 1. Architecture of the RNN model.

The model was trained using the Adam optimizer with a learning rate of 0.001 and a sparse categorical cross-entropy loss function, suitable for multi-class classification tasks. The architecture was chosen for its simplicity and computational efficiency, serving as a baseline for more advanced models.

D. Two-Layer LSTM Model

Long Short-Term Memory (LSTM) networks are a specialized type of RNN that address the limitations of standard RNNs in learning long-term dependencies. LSTMs use memory cells and gating mechanisms to retain relevant information over longer sequences, making them particularly effective for predicting longer pitch sequences.

Our LSTM-based model, illustrated in Fig. 2, stacks two LSTM layers to increase the capability of capturing hierarchical patterns in the pitch sequence. The architecture consists of the following components (we'll omit layers previously introduced):

- **First LSTM Layer**: This layer has 64 units and outputs a sequence of hidden states.
- **Second LSTM Layer**: With 128 units, this layer captures higher-level temporal dependencies in the sequence. Unlike the first layer, it outputs only the final hidden state.

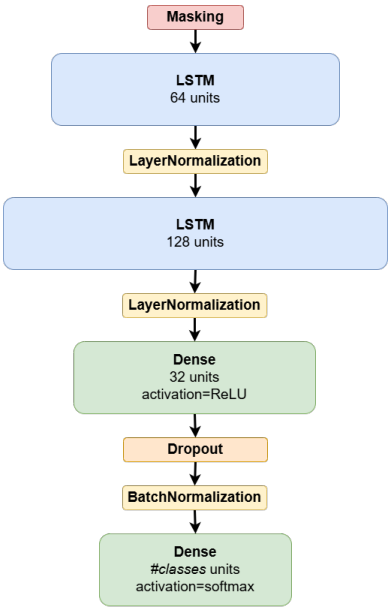

Fig. 2. Architecture of the LSTM-based model used for pitch type prediction.

The model was trained using the Adam optimizer with a learning rate of 0.001 and sparse categorical cross-entropy as the loss function. The introduction of a second LSTM layer allowed the model to capture more intricate temporal patterns, leading to potentially improved performance compared to the simpler RNN architecture.

E. Attention-Based LSTM Model

The Attention-Based LSTM model is an enhancement of the standard LSTM architecture. By integrating an attention mechanism, the model can focus on the most relevant parts of the input sequence, improving its ability to capture complex temporal dependencies in pitch sequences.

The architecture, shown in Fig. 3, consists of the following components (similarly, we'll focus on important layers only):

- **First LSTM Layer**: A recurrent layer with 64 units that processes the input sequence and outputs a sequence of hidden states.
- **Second LSTM Layer**: A recurrent layer with 128 units that outputs another sequence of hidden states.
- **Attention Layer**: Computes a context vector as a weighted sum of the LSTM outputs, where the weights are learned dynamically based on the input sequence.
- **Dense Layer (ReLU)**: A fully connected layer with 32 units and ReLU activation that extracts high-level features from the context vector.

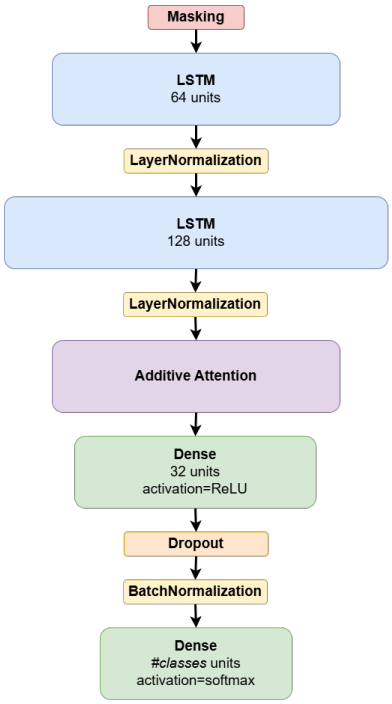

Fig. 3. Architecture of the Attention-Based LSTM model.

This model was trained using the Adam optimizer with a learning rate of 0.001 and sparse categorical cross-entropy as the loss function. By incorporating the attention mechanism, the model was able to dynamically prioritize different parts of the sequence, leading to improved performance compared to the standard LSTM architecture.

The design and implementation of this model are inspired by the approach described in the study by Yu *et al.* [1].

F. RNN-Based Specialist Model

The RNN-Based Specialist Model is designed to address the imbalance in pitch type frequencies by leveraging two sub-models: a *Normal Model* and a *Specialized Model*. Both models are based on the same RNN architecture introduced earlier, but are trained on different datasets. The final prediction is made using an ensemble system that combines the outputs of the two models. The architecture is illustrated in Fig. 4.

- **Normal Model:** This model is trained on the entire dataset and serves as the generalist, capable of predicting all pitch types, including those that occur more frequently.
- **Specialized Model:** This model is trained on a modified dataset that primarily excludes the top two most frequent pitch types, focusing on less frequent pitch types while incorporating a small mix of samples from the general dataset to avoid overly biased learning. The goal of this model is to improve the predictions for underrepresented pitch types.
- **Ensemble System:** The outputs of the Normal and Specialized Models are combined using a weighted sum mechanism, where the weights are determined based on validation performance.

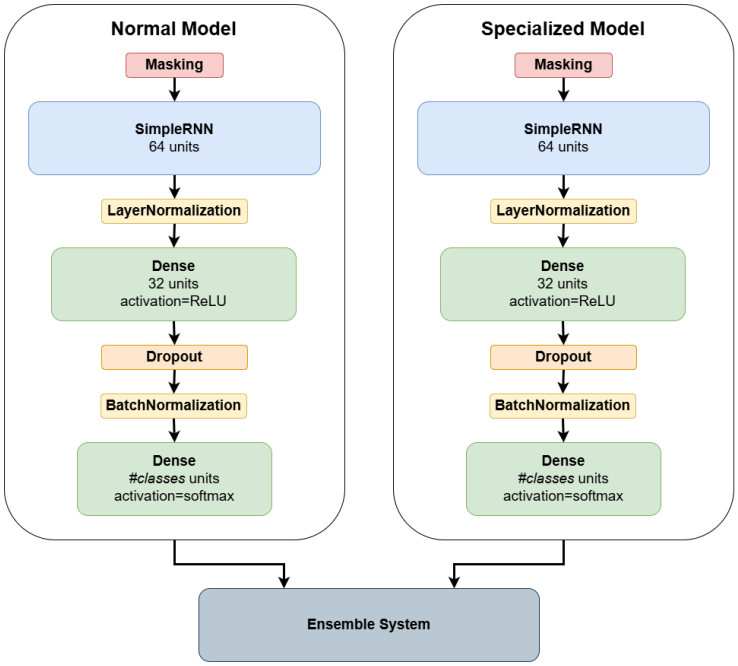

Fig. 4. Architecture of the RNN-Based Specialist Model, consisting of the Normal and Specialized Models with a weighted voting system.

Both models were trained using the Adam optimizer with a learning rate of 0.001 and sparse categorical cross-entropy as the loss function.

III. Results

Fig. 5 illustrates the performance comparison of various models (Bayes, simple RNN, LSTM, and LSTM with Attention) for Shota Imanaga. The accuracy of the models stabilizes around 0.6, with a slight improvement in the simple RNN model (0.65). The loss decreases significantly from Bayes (1.2) to simple RNN (1.0), and further stabilizes at around 1.05 for LSTM and LSTM with Attention. This trend suggests that more complex models, such as LSTM, effectively maintain accuracy while reducing loss compared to Bayes.

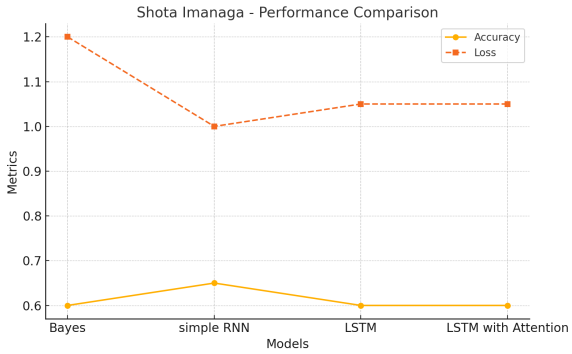

Fig. 5. The models prediction results for Shota Imanaga

Fig. 6 presents the performance of different models for Hunter Greene. The accuracy remains consistent across models, ranging from 0.6 to 0.65, with the highest accuracy

observed in the LSTM and LSTM with Attention models. Loss values show a gradual decrease, starting at 1.0 for Bayes and simple RNN, dropping to 0.95 for LSTM, and further reducing to 0.9 for LSTM with Attention. This indicates that advanced models like LSTM with Attention perform better in minimizing loss while maintaining accuracy.

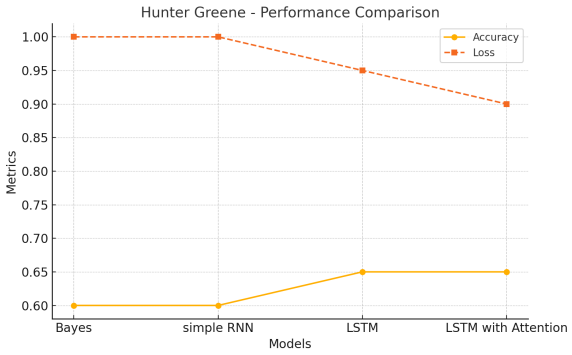

Fig. 6. The models prediction results for Hunter Greene

Fig. 7 shows the performance metrics for Mason Miller across the same set of models. Unlike the previous cases, the accuracy demonstrates a noticeable increase, from 0.55 for Bayes to 0.7 for LSTM with Attention. Meanwhile, the loss initially decreases sharply from Bayes (1.0) to simple RNN (0.8) and remains consistent for LSTM, before slightly increasing to 0.85 for LSTM with Attention. These results highlight a trade-off between achieving higher accuracy and maintaining low loss in more advanced models.

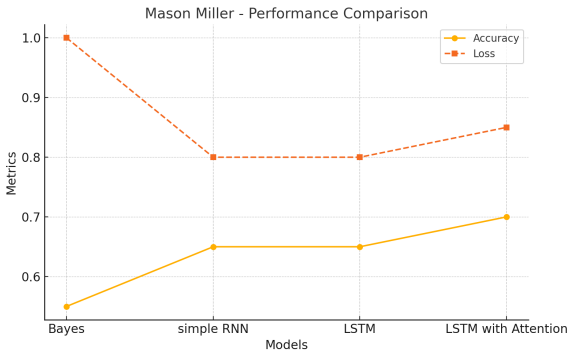

Fig. 7. The models prediction results for Mason Miller

From the analysis of the three pitchers, it can be observed that Mason Miller's model performed the best. This may be attributed to his role as a relief pitcher, where fewer pitch types result in higher prediction accuracy. Relief pitchers typically face fewer batters per game compared to starting pitchers, which might allow the model to better identify patterns and make more precise predictions. Additionally, the simplicity of the pitch repertoire for relief pitchers reduces variability, which likely contributes to improved model performance.

IV. Disscussion

This study highlights several key observations and insights into the performance and behavior of our RNN-based pitch prediction models.

1. Limitations of Accuracy and Loss Metrics

Although accuracy and loss were initially used as primary metrics for model evaluation, they proved insufficient to uncover deeper issues. By analyzing the confusion matrix, we identified a significant bias in our models: frequent over-prediction of fastballs, even when the ground truth was another pitch type. This bias probably stems from the high proportion of fastballs in the dataset, as evidenced by the pitch distribution analysis. This highlights the importance of supplementary metrics like the confusion matrix to identify and address such imbalances.

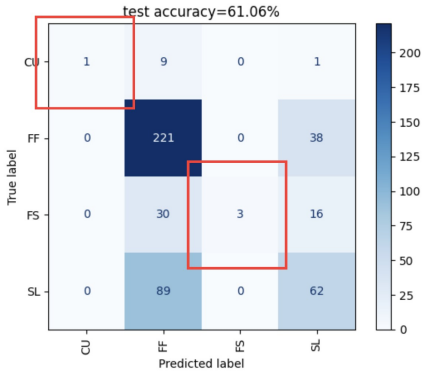

Fig. 8. Confusion Matrix for Simple RNN Predictions on Hunter Greene

2. Addressing Bias with Model Ensemble

To mitigate this bias, we implemented a generalist-specialist ensemble approach. A specialist model was trained to focus on less frequent pitch types, complementing the generalist model, which handled common cases. By introducing a voting mechanism, the ensemble approach successfully reduced the over-prediction of fastballs and improved the model's ability to predict less frequent pitch types, such as splitters and curve-balls. Although the overall accuracy only improved slightly, this method demonstrated a significant impact on the balance and robustness of predictions.

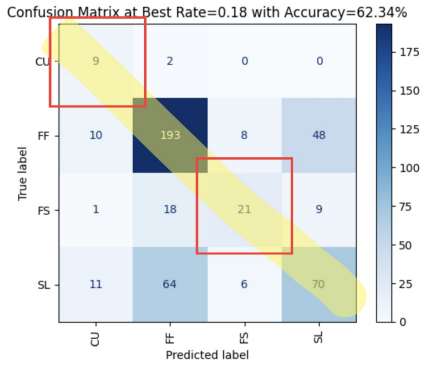

Fig. 9. Confusion Matrix for Specialist model Predictions on Hunter Greene

3. **Challenges with LSTM and Attention Mechanisms in Short Sequences**

Our subsequences ranged from 2 to 6 pitches, which limited the utility of long-term memory mechanisms like LSTM. The short length of these sequences meant that the long-term dependencies that LSTM and attention layers are designed to capture were less relevant. Moreover, these mechanisms introduced potential **bias accumulation**, where the model overemphasized high-frequency inputs, such as fastballs, while neglecting less frequent pitch types. This suggests that simpler architectures, like SimpleRNN, may be more effective for short sequences.

4. **Model Performance Across Different Pitcher Types**

Our analysis revealed that the model performed better with pitchers who have fewer pitch types, as patterns were easier to detect and learn. For pitchers with a more diverse repertoire, such as Shota Imanaga, the model's performance remained solid but less pronounced, reflecting the increased complexity in predicting such sequences. This indicates that the effectiveness of the model can vary depending on the pitcher's style and pitch variety.

V. Conclusion

This project demonstrates the effectiveness of using recurrent neural network (RNN) models for predicting the next pitch type in baseball. Compared to the Bayesian baseline, our model achieves significantly better results, with an accuracy of 69% versus 55%. This highlights the capability of RNNs to capture abstract features and sequential dependencies that traditional probabilistic approaches cannot.

Additionally, our model performs notably well for pitchers with fewer pitch types, such as Miller and Green, as patterns are easier to identify and predict. For pitchers with more diverse pitch arsenals, like Shota Imanaga, the model still remains robust, albeit with slightly less pronounced performance advantages.

The practical value of this model lies in its ability to provide actionable insights to pitchers, coaches, and catchers.

• **For Pitchers and Coaches**: The model can identify subtle patterns or habits, such as consistent release points or predictable sequences, which might give batters an advantage. Addressing these issues can help pitchers refine their strategies and improve their performance.

• **For Catchers**: The model can act as a diagnostic tool to detect habit patterns in pitch calling, allowing more effective and varied strategies during games.

This work not only advances pitch prediction accuracy, but also provides a framework for analyzing pitching tendencies and refining strategies. Future improvements could involve incorporating additional contextual features, such as batter profiles and game situations, to further enhance the performance of the model and broaden its application.

Data and Code Availability

The code for dataset downloading, preprocessing, model training, and result presentation is available in our GitHub repository: https://github.com/bretonyang/Pitch-Type-Prediction-Using-RNN.

Please refer to the README file in the repository for detailed instructions on how to run the code and reproduce the results.

Author Contribution Statements

The contributions of each team member to this project are as follows:

- **S.-Z.Y. (20%):** Data collection, data analysis, presentation, and writing.
- **P.-H.T. (20%):** Data collection, data analysis, presentation, and writing.
- **Y.-Z.L. (20%):** Model optimization, presentation, and writing.
- **Y.-S.X. (20%):** Study model design, data preprocessing, and implementation.
- **L.-T.Y. (20%):** Study model design, data preprocessing, and implementation.

All team members participated in brainstorming sessions and discussions, contributed to the project, and approved the final manuscript.

References

1 Yu, C.-C., Chang, C.-C., and Cheng, H.-Y., "Decide the next pitch: A pitch prediction model using attention-based lstm," in *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2022, pp. 1–4.

2 Pi, Y., "Predict next baseball pitch type with rnn," 2018, accessed: 2024-12-25. [Online]. Available: https://cs230.stanford.edu/projects_spring_2018/reports/8290890.pdf

3 Koseler, K. and Stephan, M., "Machine learning applications in baseball: A systematic literature review," *Applied Artificial Intelligence*, vol. 31, no. 9-10, pp. 745–763, 2017. [Online]. Available: https://doi.org/10.1080/08839514.2018.1442991