Carnegie Mellon University – Heinz College
95820-A1 Fall 2024

Predictive Maintenance Using LSTM and GRU
NASA Turbofan Engine Dataset

Ahmad Furqan – afurqan

A. Executive Summary

This project explores the application of Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) models for predicting the Remaining Useful Life (RUL) of aircraft engines, a key task in predictive maintenance (Kaplan, 2024). Accurate RUL prediction optimizes maintenance schedules, reduces unplanned downtime, and improves safety in industries like aerospace.

LSTM and GRU observed two datasets from NASA's CMAPSS database, FD001 and FD002 (Nasa, 2024). FD001 represents a simpler scenario with one operating condition, while FD002 introduces more complexity with six operating conditions. Both models were evaluated on accuracy, precision, recall, F1-score, and computational efficiency. GRU demonstrated faster training times, making it more computationally efficient for simpler datasets (FD001). LSTM, however, showed better performance on the more complex FD002 dataset, handling long-term dependencies with higher validation accuracy. Our results highlight the trade-offs between LSTM and GRU: LSTM excels in complex scenarios due to its ability to capture long-term dependencies, while GRU is more efficient for less complex tasks. Future work could involve bi-directional LSTM, ensemble models, and exploring other datasets for broader generalization.

B. Introduction

Predictive maintenance is critical in sectors where equipment failure can lead to operational losses and safety hazards, such as aerospace. It focuses on anticipating equipment failures before they occur, enabling proactive interventions that minimize downtime. One crucial aspect of predictive maintenance is predicting the Remaining Useful Life (RUL) of machinery like aircraft engines.

This project focuses on using deep learning techniques, specifically LSTM and GRU models, to predict the RUL of aircraft engines. These models are well-suited for time-series forecasting because they capture temporal dependencies within data. LSTM manages long-term dependencies better but is computationally heavier, while GRU offers faster training with slightly simpler architecture. I used two datasets from NASA's CMAPSS database: FD001 (one operating condition) and FD002 (six operating conditions). These datasets provided a controlled scenario (FD001) and a more complex environment (FD002) to compare the models' ability to generalize across different levels of complexity

C. Methodology

This project utilized a modular Python program for training and comparing LSTM and GRU models on NASA's CMAPSS FD001 and FD002 datasets. The modularity ensures flexibility, maintainability, and scalability, with each phase handled separately in dedicated scripts: data preprocessing, model building, hyperparameter tuning, and evaluation.

1. General Approach

The project is structured around a command-line interface (CUI) that allows users to switch between FD001 and FD002 datasets and choose whether to train models with static or tuned hyperparameters. This modularity enhances flexibility, enabling easy model comparisons and reducing unnecessary retraining by offering pre-trained model evaluation. My approach in bringing The CUI to main_rul.py enables users to switch between datasets and modes (static vs. tuned hyperparameters) easily, making the project more flexible. Given computational resource constraints (laptop environment), I provided options for training models with static parameters and evaluating pre-trained models directly, reducing unnecessary retraining if we have done before.

2. Data Preprocessing

Data preprocessing includes cleaning, feature engineering, normalization, and sequence generation:

- Data Cleaning: Unnecessary columns were dropped, and the data was cleaned for missing values.
- Feature Engineering: RUL was calculated as the difference between the maximum cycle and the current cycle, and binary labels were generated based on RUL thresholds.
- Normalization: A MinMaxScaler was applied to scale sensor data and operational settings.
- Sequence Generation: A sliding window approach (window size: 50 cycles) was used to generate sequences for input to the LSTM and GRU models.

3. Model Building

Both LSTM and GRU models were built with two recurrent layers and dropout layers to prevent overfitting. A binary classification task was performed using a sigmoid activation function. Models were trained with early stopping to prevent overfitting and model checkpointing to save the best-performing models.

4. **Hyperparameter Tuning**
   Hyperparameter tuning was conducted using grid search, optimizing for:
   - Units: 32 or 50 units.
   - Dropout Rate: 0.1 or 0.2.
   - Batch Size: 16 or 32.
   - Optimizer: Adam and RMSprop were compared. The best-performing models were selected based on validation accuracy and generalization (accuracy gap), and training time was tracked to compare computational efficiency.

5. **Model Evaluation**
   Evaluation metrics included accuracy, precision, recall, and F1-score. Visualization tools such as accuracy/loss plots, actual vs. predicted RUL plots, and confidence intervals were generated to compare the models' performance on both FD001 and FD002.

D. Results and Insights

1. FD001 Dataset (see **Appendix A)**

a. Hyperparameter Tuning Process

Model	Avg Train Acc	Avg Validation Acc	Total Time
LSTM	98.22%	97.53%	84.15 minutes
GRU	98.37%	97.51%	109.85 minutes

LSTM models had a slightly shorter training time compared to GRU models, which makes them computationally more efficient in this case. However, the GRU models consistently showed higher training accuracy, potentially due to their simpler architecture. (See **Appendix B**)

Hyperparameter Effects on Performance Metrics (see **Appendix C**):

- Accuracy Gap: The GRU models maintained a smaller accuracy gap (train vs. validation accuracy) across different hyperparameter settings, indicating better generalization. LSTM models, while sometimes achieving higher validation accuracy, had a larger accuracy gap, suggesting a greater tendency to overfit.
- Training Time: LSTM models had shorter training times compared to GRU, making them more computationally efficient. However, GRU models showed higher training accuracy, possibly due to their simpler architecture and better capacity for convergence.
- Best Models: The best LSTM model had a validation accuracy of 98.46% with an accuracy gap of 0.01%, and the best GRU model had a validation accuracy of 98.21% with an accuracy gap of 0.09%. These models represent the best trade-off between validation accuracy and generalization.

Validation Accuracy vs Hyperparameters (see **Appendix E)**:

- Units: Both LSTM and GRU models showed the highest validation accuracy with 32 units. Increasing the number of units to 50 generally resulted in a slight performance decrease, especially for LSTM models. GRU models were more stable across unit variations.
- Dropout Rate: A lower dropout rate of 0.1 produced better validation accuracy for both models, while a dropout rate of 0.2 caused minor drops in accuracy. LSTM models were more sensitive to dropout, showing greater performance variance.
- Batch Size: Both models performed better with larger batch sizes (32). LSTM models showed more fluctuation in validation accuracy across different batch sizes compared to GRU, which remained more consistent.
- Optimizer: The Adam optimizer consistently outperformed RMSprop for both models in terms of validation accuracy.

Insights from Accuracy vs Accuracy Gap: The chart shows that models with high validation accuracy and low accuracy gaps are clustered in the lower right corner, which is ideal for both high performance and generalization. While LSTM models can achieve slightly higher validation accuracy, GRU models consistently exhibit smaller accuracy gaps, making them more robust in avoiding overfitting. (See **Appendix G**)

The hyperparameter tuning process revealed that GRU models offer more stable performance across various hyperparameter configurations, particularly with smaller accuracy gaps, indicating better generalization. LSTM models, while sometimes reaching slightly higher validation accuracy, are more prone to overfitting, as shown by the larger accuracy gaps in some configurations. These findings highlight the trade-offs between achieving high accuracy and ensuring robust generalization, with GRU models being the more reliable choice in terms of overall stability and performance.

b. Model Accuracy and Loss Performance Evaluation

Model Accuracy (see **Appendix E)**:

- The GRU model achieved a peak validation accuracy of 98.21%, with a smooth convergence between training and validation accuracy. The model reached high accuracy early, stabilizing with fewer epochs, showing strong generalization.
- The LSTM model performed slightly better with a validation accuracy of 98.46%, but required more epochs to stabilize. The validation accuracy fluctuated more than GRU, indicating a more complex learning process.

Model Loss (see **Appendix F):**

- The GRU model quickly reduced its loss, with both training and validation loss curves converging after a few epochs, showing efficient and stable learning.
- The LSTM model showed more volatility in validation loss, suggesting occasional challenges in generalization, though it eventually achieved a lower final loss than GRU.

General Insights: Both models showed minimal overfitting, with small accuracy gaps. However, GRU's consistent performance and shorter training time make it a strong candidate for time-sensitive or resource-limited applications. Meanwhile, LSTM's higher accuracy may be favored when computational resources allow for longer training periods. Eventually, GRU is preferable for quicker, resource-efficient tasks, while LSTM offers marginally better accuracy at the cost of longer training times and potential overfitting risks.

c. Actual vs Predicted RUL Performance (see **Appendix J)**

The comparison between actual and predicted Remaining Useful Life (RUL) for both LSTM and GRU models highlights key insights into their performance for FD001. From the charts, we can observe the following:

- **LSTM Model**: The LSTM demonstrates a relatively close alignment with the actual RUL values. However, the predictions have some fluctuations in tracking the real RUL, showing a delay or over-prediction in some samples. Despite this, the model's overall prediction pattern follows the trend of actual RUL reasonably well.
- **GRU Model**: Similarly, the GRU model also exhibits prediction oscillations but maintains an acceptable level of accuracy in predicting the RUL. However, like the LSTM model, it shows instances where predictions deviate from actual values, particularly over-predicting in certain cases.
- Meanwhile, the visual difference between the two models is minimal in terms of prediction behavior, but LSTM shows marginally better generalization to unseen samples based on its higher validation accuracy, as discussed earlier. Nonetheless, both models suffer from some degree of over-prediction in individual cycles, which can be addressed with more complex architectures or additional tuning.

Furthermore, the key findings from the Actual vs. Predicted RUL with Prediction Intervals plots for both LSTM and GRU models highlight a significant trend. Both models produced reasonably close predictions to the actual Remaining Useful Life (RUL), as seen from the overlapping lines in several segments of the graphs. The predicted mean tracks the actual RUL well, although there are instances where the predicted values deviate from the true values.

In terms of confidence intervals (represented as the 95% prediction interval), both models offer narrow bands in several regions, indicating that the predictions are stable with low variance. The intervals, however, show that there are cases with more considerable uncertainty (wider intervals), which suggests potential difficulty in handling certain time steps. Despite this, the models generally provide tight intervals, especially when the predicted RUL closely follows the actual RUL, indicating a strong confidence level.

The GRU model's prediction intervals appear slightly tighter in several areas compared to LSTM, indicating that GRU might be less sensitive to fluctuations and sensor noise. However, the overall performance of both models remains consistent in most cases, with relatively small deviations. The combination of high accuracy and low variance in these intervals suggests that both models can be effectively used for predictive maintenance tasks, although specific improvements in certain conditions could further enhance performance.

d. Model Comparison Metrics (see **Appendix L)**
In evaluating the models' performance on the FD001 dataset, I examined four key metrics: accuracy, precision, recall, and F1-score. The bar chart comparing LSTM and GRU models reveals the following insights:

- *Accuracy*: Both LSTM and GRU models achieve high accuracy levels, with the GRU slightly outperforming LSTM. This indicates that GRU models are marginally better at correctly predicting the remaining useful life (RUL) overall.
- *Precision*: Precision scores for both models are very close, reflecting their similar ability to avoid false positives. GRU shows a slight edge in precision, which could suggest its slightly better ability to avoid false alarms in predictive maintenance.
- *Recall*: The recall values for both LSTM and GRU models are also comparable, but GRU again performs marginally better. Higher recall in the GRU model indicates it is more effective at identifying when the RUL prediction is correct, reducing missed predictions.
- *F1-score*: As a balance between precision and recall, the F1-score shows GRU leading slightly. This suggests GRU models maintain a better balance between catching faults (recall) and avoiding false alarms (precision).

2. Comparison of LSTM and GRU Models on FD001 and FD002 Datasets
Handling Complexity: FD001 vs. FD002
FD001 represents a simpler scenario with only one operating condition and one fault mode, making it easier for models like LSTM and GRU to learn and generalize. On the other hand, FD002 introduces six operating conditions with a single fault mode, significantly increasing the complexity and variability the models must account for during prediction.

FD001 Insights: LSTM models performed slightly better in terms of validation accuracy due to their ability to capture long-term dependencies. This was evident in the steady convergence and higher accuracy when handling the simpler dataset. GRU, being computationally more efficient, converged faster while maintaining similar performance to LSTM, though its simpler architecture led to a slight dip in accuracy when compared to LSTM on FD001.

FD002 Insights: With the introduction of multiple operating conditions in FD002, LSTM models showed better performance in capturing the more complex patterns, but this came at the cost of increased training time and higher risk of overfitting, as seen in the higher volatility in validation accuracy and larger accuracy gaps in some hyperparameter configurations. GRU struggled more with FD002, showing larger fluctuations in validation accuracy. However, GRU models exhibited more consistent results across different hyperparameters, with a generally smaller accuracy gap between training and validation.

Performance Boost Through Hyperparameter Tuning on FD002
Our hyperparameter tuning efforts were critical in optimizing the models' performance, especially in FD002 where the increased complexity required more careful tuning to avoid overfitting and improve generalization. The table of results for FD002 provides insight into the effect of tuning hyperparameters like units, dropout rate, batch size, and optimizer choice.

- LSTM models reached the highest validation accuracy of 94.95% with a relatively small accuracy gap of 0.0157 using 32 units, 0.2 dropout, batch size of 32, and the RMSprop optimizer. This highlights how proper tuning can enhance the model's ability to generalize despite the complexity of FD002.
- GRU models performed best with 32 units, a 0.1 dropout rate, batch size of 32, and the RMSprop optimizer, achieving a validation accuracy of 94.22%. Although GRU models required less training time compared to LSTM, they showed a slightly larger accuracy gap in most cases, reflecting their limited ability to handle the increased complexity of FD002.

Trade-offs Between Training Time and Performance
One of the main advantages of GRU over LSTM is its faster training time, which was evident throughout both FD001 and FD002. While LSTM outperformed GRU in terms of validation accuracy on both datasets, particularly on FD002, the time it took to train LSTM models was significantly higher. For instance: Best LSTM configuration (32 units, 0.2 dropout, RMSprop) on FD002 took 16.02 minutes to train. In comparison, the best GRU configuration (32 units, 0.2 dropout, RMSprop) took only 11.05 minutes, while achieving a slightly lower validation accuracy of 94.22%.

The accuracy gap between training and validation is a crucial indicator of overfitting. In many cases, LSTM models had smaller gaps, indicating better generalization, especially when tuned with appropriate hyperparameters. However, GRU, despite its simpler architecture, showed competitive results, making it a viable option when computational efficiency is prioritized.

Challenges and Improvements from Hyperparameter Tuning

- Higher dropout rates generally improved the model's generalization ability by preventing overfitting, particularly in the complex FD002. However, this also led to longer training times.
- Adam consistently outperformed RMSprop in most cases in terms of validation accuracy, though at the cost of increased training time.
- Larger batch sizes (32 vs. 16) improved validation accuracy and reduced training time, particularly for GRU models.

These tuning efforts resulted in significant performance improvements, especially in FD002, but they also came at the cost of increased training times. For example, some of the best-performing configurations for GRU and LSTM took over 15-18 minutes per run, compared to other less optimal configurations that completed in under 10 minutes.

LSTM shows its superior performance on FD002. However, it comes with the cost of higher training times and a greater tendency to overfit. GRU offers a balance between speed and performance, making it a strong candidate for less complex scenarios like FD001 or time-constrained environments. While it struggled slightly with the increased complexity in FD002, hyperparameter tuning allowed GRU to achieve competitive results with smaller accuracy gaps and faster training times.

E. Discussion

Advantages of LSTM: In our project, LSTM performed well on both the FD001 and FD002 datasets, but it showed a more noticeable advantage in FD002, which involves multiple operating conditions. The increased complexity of FD002 required the model to handle more varied sequences and dependencies, and LSTM's architecture allowed it to capture this complexity better than GRU. The model achieved higher validation accuracy on FD002, indicating its robustness in handling more challenging datasets with long-term temporal patterns. LSTM's capacity to store information over long sequences gave it an edge in predicting more complex operational states and degradation trends, which is essential in predictive maintenance.

Advantages of GRU: In both FD001 and FD002, GRU consistently showed faster training times compared to LSTM while maintaining competitive accuracy. On FD001, which is a simpler dataset with a single operating condition, GRU performed similarly to LSTM in terms of accuracy, but with significantly shorter training times. This makes GRU an attractive choice for real-time predictive maintenance applications where computational resources may be limited, or faster model iteration is required. Moreover, GRU models demonstrated more stable performance in terms of generalization, with smaller accuracy gaps between training and validation accuracy, especially on FD001.

Disadvantages of LSTM: LSTM tends to overfit on complex datasets if not properly regularized. In our training experiments, LSTM models showed larger accuracy gaps between training and validation accuracy, especially on FD002, suggesting a higher likelihood of overfitting. This tendency can be managed with techniques like dropout, early stopping, and regularization, but LSTM's complexity means that it requires more careful tuning than GRU. Additionally, LSTM models are computationally more expensive than GRU models, especially on larger or more complex datasets like FD002. This higher computational cost comes from the multiple gates in LSTM, which increase the number of parameters that need to be trained. As a result, LSTM models took longer to train and were more resource-intensive, making them less ideal for real-time applications or scenarios with limited computational resources.

Disadvantages of GRU: In our experiments, GRU performed well on FD001 but slightly underperformed compared to LSTM on FD002, where long-term dependencies and complex temporal patterns are more prominent. The simpler gating mechanism in GRU can result in the model discarding important information prematurely, which may lead to lower accuracy in scenarios requiring deeper sequence memory. Additionally, while GRU models demonstrated smaller accuracy gaps (indicating better generalization), they may miss subtle patterns that LSTM captures due to their simpler architecture. This could lead to lower performance in highly variable operating conditions, as seen in FD002.

Future Work and Potential Improvements

- Bi-directional LSTM could help further improve RUL prediction by providing the model with more contextual information. In complex dataset, such as FD002, bi-directional models could capture more intricate temporal dependencies, potentially leading to better performance.
- Ensemble Models: Another promising avenue for improvement is ensemble learning. Combining LSTM and GRU models, or integrating other architectures like Convolutional Neural Networks (CNN) or Transformer models, could improve performance by leveraging the strengths of each model (Cao et al., 2024). Ensembles are known to improve robustness and generalization by reducing model variance and bias. By combining the strengths of LSTM's memory capacity and GRU's computational efficiency, an ensemble approach could offer better overall performance.
- Other RNN Variants: Exploring attention-based models or Transformers may provide better scalability and performance in handling complex sequences, especially in datasets like FD002.

F. References

Cao, K., Zhang, T., & Huang, J. (2024). Advanced hybrid LSTM-transformer architecture for real-time multi-task prediction in engineering systems. *Scientific Reports*, *14*(1), 4890. https://doi.org/10.1038/s41598-024-55483-x

Kaplan, S. (2024, January 22). *Smooth Sailing in the Sky: The Role of Predictive Maintenance in Aviation*. https://praxie.com/predictive-maintenance-in-aviation/

Nasa. (2024, May 15). *CMAPSS Jet Engine Simulated Data | NASA Open Data Portal*. NASA's Open Data Portal. https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6/about_data

G. Appendix

Appendix A – FD001 Dataset Snapshot

```
1  print(train_df.info())
✓ 0.0s
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20631 entries, 0 to 20630
Data columns (total 26 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        20631 non-null  int64
 1   cycle     20631 non-null  int64
 2   setting1  20631 non-null  float64
 3   setting2  20631 non-null  float64
 4   setting3  20631 non-null  float64
 5   sensor1   20631 non-null  float64
 6   sensor2   20631 non-null  float64
 7   sensor3   20631 non-null  float64
 8   sensor4   20631 non-null  float64
 9   sensor5   20631 non-null  float64
 10  sensor6   20631 non-null  float64
 11  sensor7   20631 non-null  float64
 12  sensor8   20631 non-null  float64
 13  sensor9   20631 non-null  float64
 14  sensor10  20631 non-null  float64
 15  sensor11  20631 non-null  float64
 16  sensor12  20631 non-null  float64
 17  sensor13  20631 non-null  float64
 18  sensor14  20631 non-null  float64
 19  sensor15  20631 non-null  float64
...
 25  sensor21  20631 non-null  float64
dtypes: float64(22), int64(4)
memory usage: 4.1 MB
None
```

```
1  print(test_df.info())
✓ 0.0s
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13096 entries, 0 to 13095
Data columns (total 26 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        13096 non-null  int64
 1   cycle     13096 non-null  int64
 2   setting1  13096 non-null  float64
 3   setting2  13096 non-null  float64
 4   setting3  13096 non-null  float64
 5   sensor1   13096 non-null  float64
 6   sensor2   13096 non-null  float64
 7   sensor3   13096 non-null  float64
 8   sensor4   13096 non-null  float64
 9   sensor5   13096 non-null  float64
 10  sensor6   13096 non-null  float64
 11  sensor7   13096 non-null  float64
 12  sensor8   13096 non-null  float64
 13  sensor9   13096 non-null  float64
 14  sensor10  13096 non-null  float64
 15  sensor11  13096 non-null  float64
 16  sensor12  13096 non-null  float64
 17  sensor13  13096 non-null  float64
 18  sensor14  13096 non-null  float64
 19  sensor15  13096 non-null  float64
...
 25  sensor21  13096 non-null  float64
dtypes: float64(22), int64(4)
memory usage: 2.6 MB
None
```

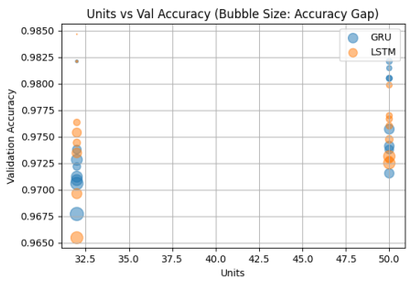
```
1  print(truth_df.info())
✓ 0.0s
```

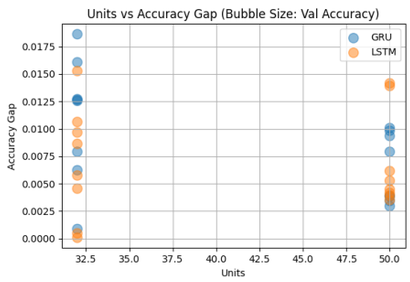
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Truth   100 non-null    int64
dtypes: int64(1)
memory usage: 932.0 bytes
None
```

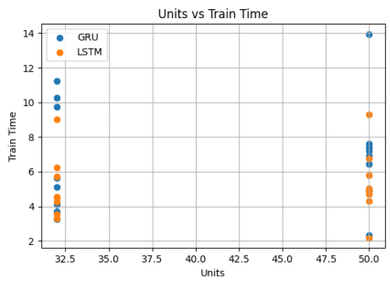
Appendix B – Hyperparameter Tuning History on FD001

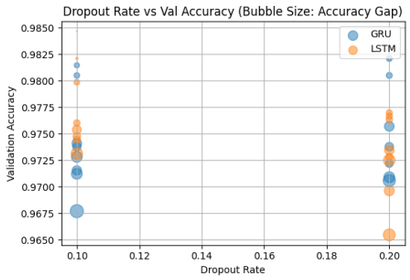
Model_Type	Units	Dropout_Rate	Batch_Size	Optimizer	Train_Accuracy	Val_Accuracy	Accuracy_Gap	Train_Time
GRU	32	0.2	32	adam	0.982965	0.982091	0.000874	3.264460
GRU	50	0.2	16	rmsprop	0.985924	0.982091	0.003833	13.937182
GRU	50	0.1	32	adam	0.984405	0.981452	0.002953	7.201793
GRU	50	0.1	16	rmsprop	0.983925	0.980492	0.003433	6.921180
GRU	50	0.2	32	rmsprop	0.984405	0.980492	0.003913	7.390501
GRU	50	0.2	16	adam	0.985765	0.975696	0.010069	6.452336
GRU	50	0.1	16	adam	0.983925	0.974097	0.009829	7.597557
GRU	32	0.1	16	adam	0.981686	0.973777	0.007909	5.656541
GRU	50	0.2	32	adam	0.981686	0.973777	0.007909	2.320309
GRU	32	0.1	32	adam	0.985445	0.972817	0.012627	5.093081
GRU	32	0.2	32	rmsprop	0.978407	0.972178	0.006229	3.719772
GRU	50	0.1	32	rmsprop	0.980886	0.971538	0.009348	4.929930
GRU	32	0.1	32	rmsprop	0.983765	0.971218	0.012547	4.104933
GRU	32	0.2	16	rmsprop	0.983605	0.970899	0.012707	9.743698
GRU	32	0.2	16	adam	0.986644	0.970579	0.016065	11.250013
GRU	32	0.1	16	rmsprop	0.986324	0.967701	0.018624	10.269759
LSTM	32	0.1	32	rmsprop	0.984725	0.984650	0.000075	6.252695
LSTM	32	0.1	16	rmsprop	0.982566	0.982091	0.000474	4.560625
LSTM	50	0.1	32	rmsprop	0.983365	0.979853	0.003512	6.756887
LSTM	50	0.2	16	adam	0.980886	0.976975	0.003911	5.045361
LSTM	50	0.2	32	rmsprop	0.980806	0.976655	0.004151	4.895299
LSTM	32	0.2	32	adam	0.980886	0.976335	0.004551	4.278146
LSTM	50	0.1	32	adam	0.980486	0.976015	0.004471	2.187423
LSTM	32	0.1	16	adam	0.984005	0.975376	0.008629	4.511128
LSTM	50	0.1	16	rmsprop	0.980886	0.974736	0.006150	4.697140
LSTM	32	0.1	32	adam	0.980166	0.974416	0.005750	3.284815
LSTM	32	0.2	16	rmsprop	0.983125	0.973457	0.009668	9.029596
LSTM	50	0.1	16	adam	0.987044	0.973137	0.013907	5.800133
LSTM	50	0.2	16	rmsprop	0.978087	0.972817	0.005270	4.294108
LSTM	50	0.2	32	adam	0.986644	0.972498	0.014147	9.286542
LSTM	32	0.2	32	rmsprop	0.980246	0.969619	0.010627	5.729603
LSTM	32	0.2	16	adam	0.980726	0.965462	0.015264	3.542714

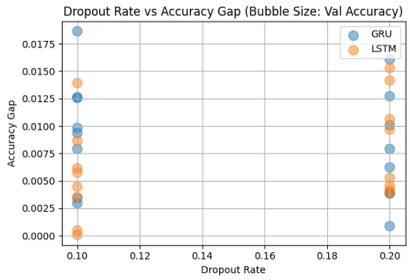
Appendix C - Hyperparameter Effects on Performance Metrics on FD001

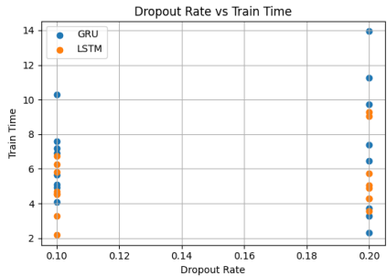


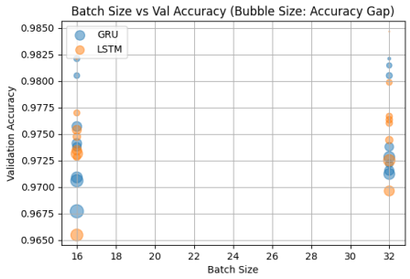


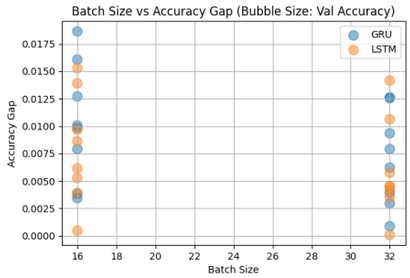


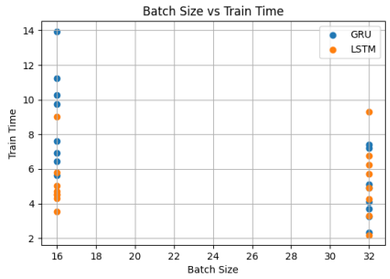


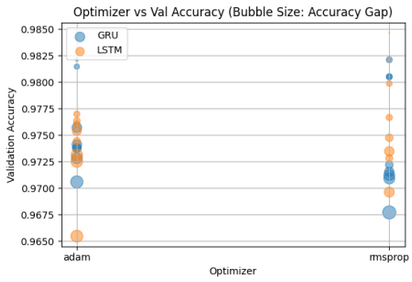


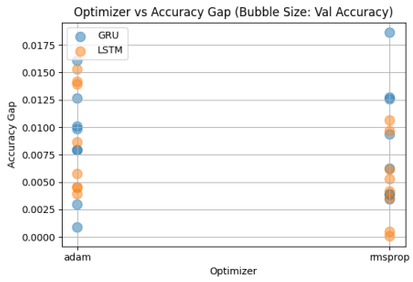


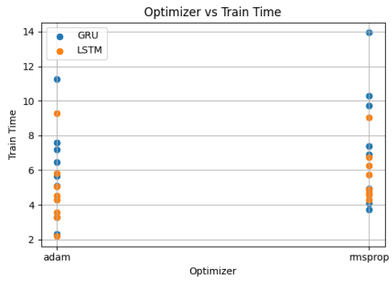




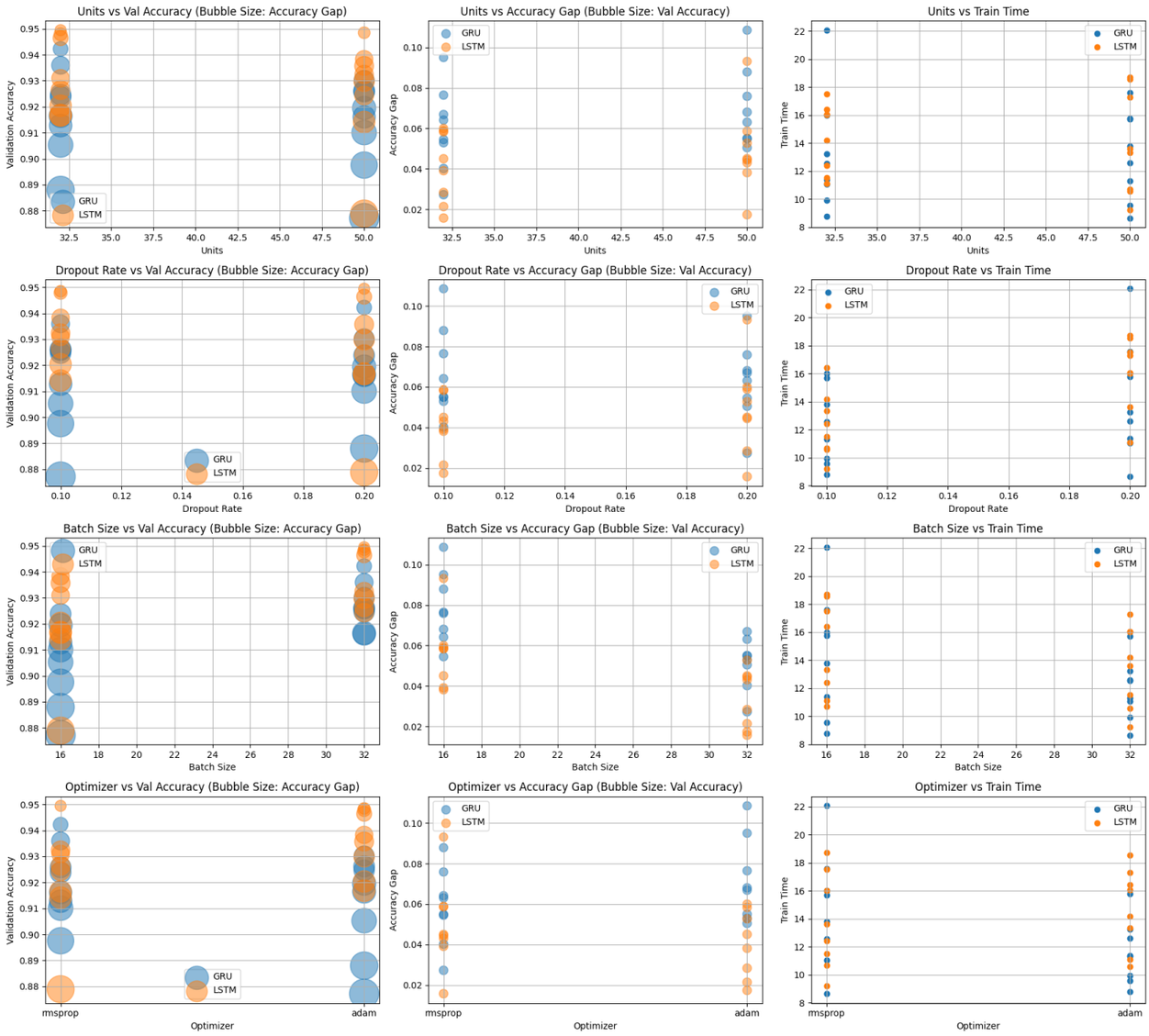




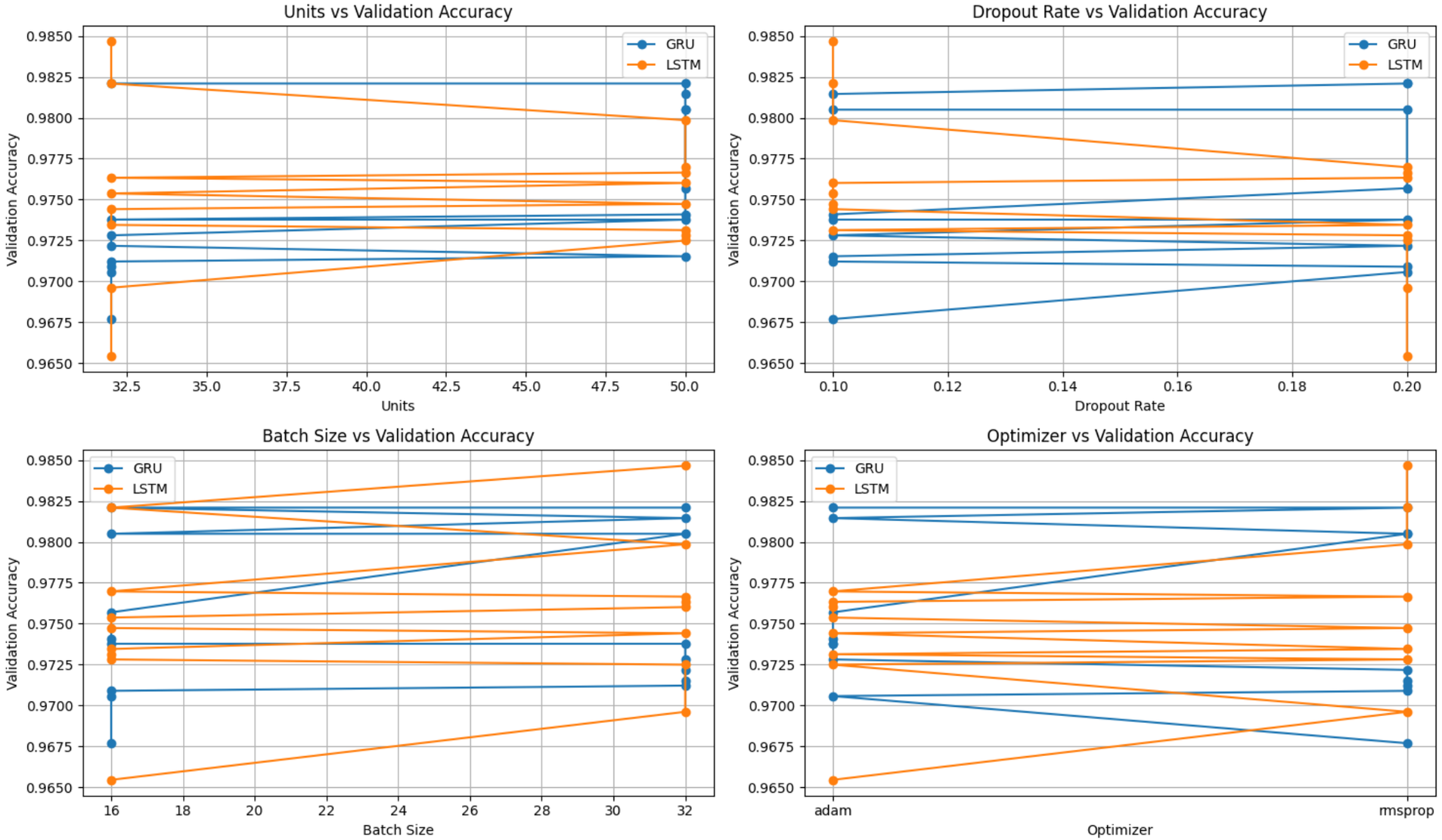




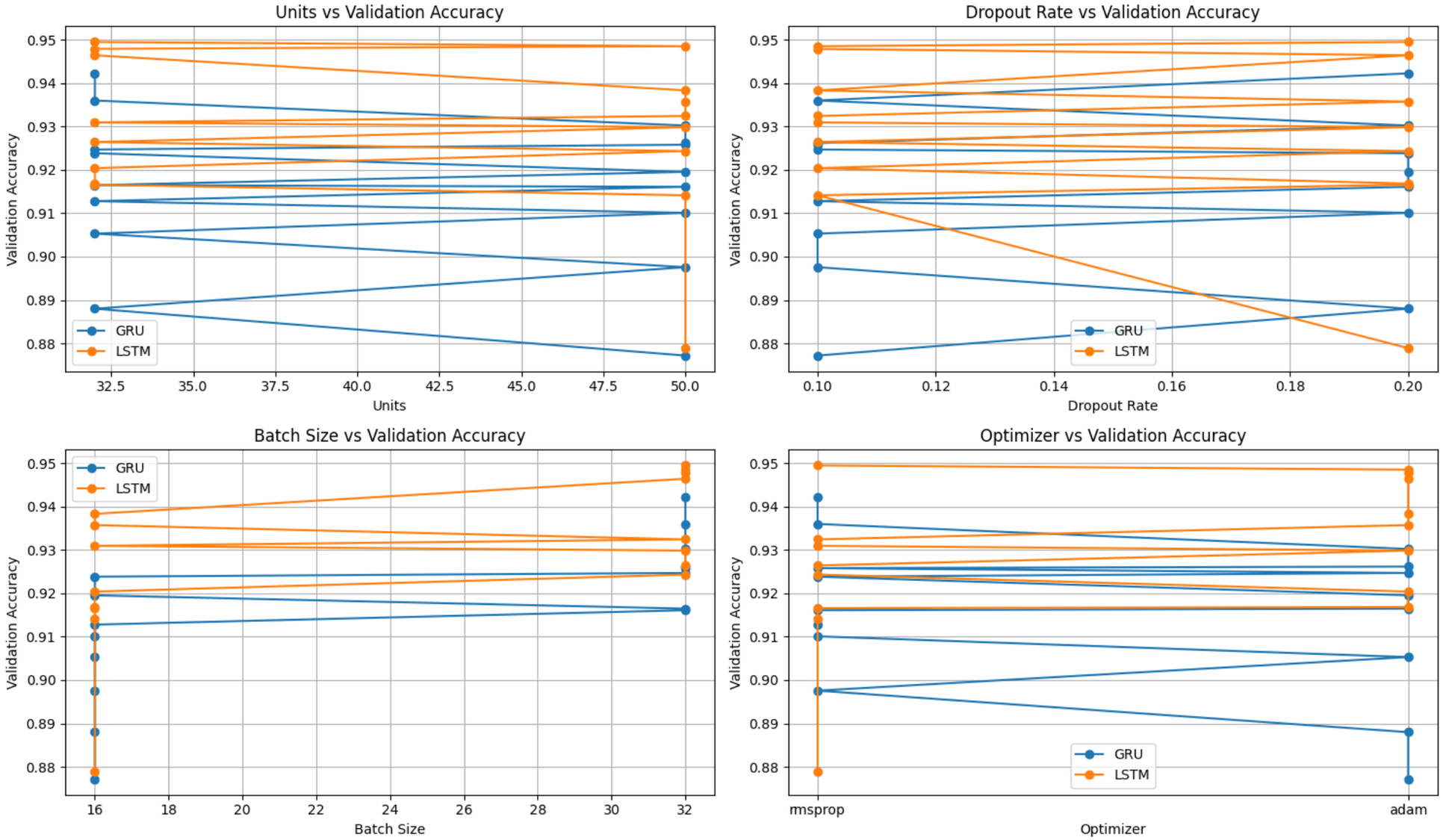
Appendix D - Hyperparameter Effects on Performance Metrics on FD002



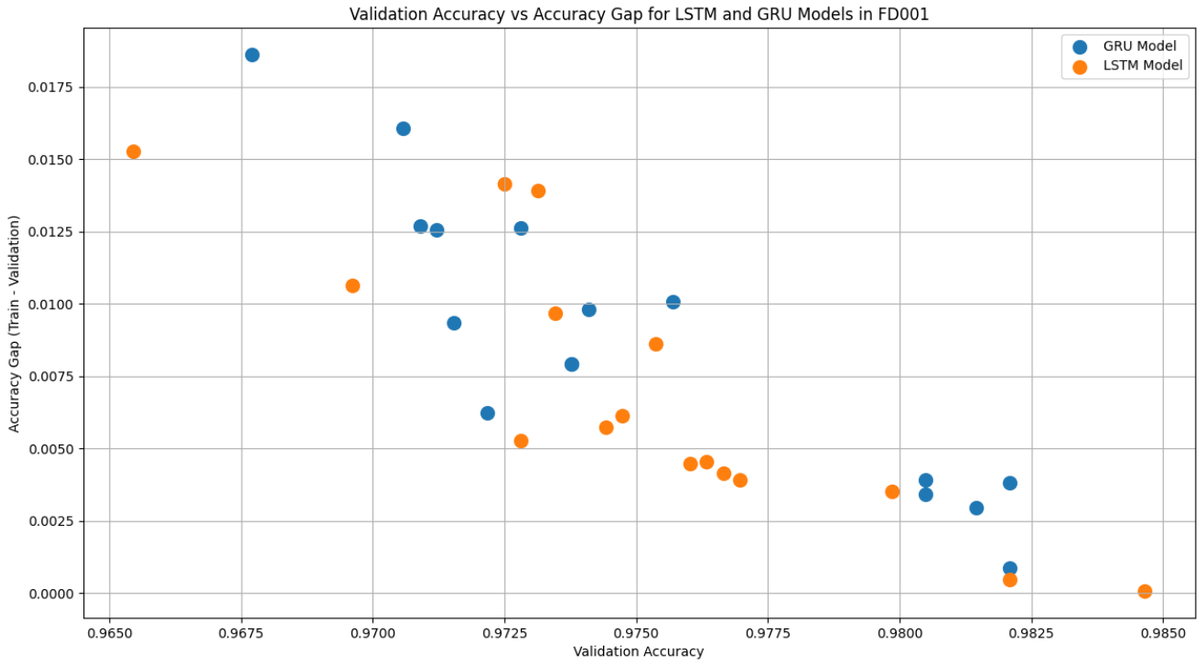
Appendix E - Validation Accuracy vs Hyperparameters for LSTM and GRU on FD001

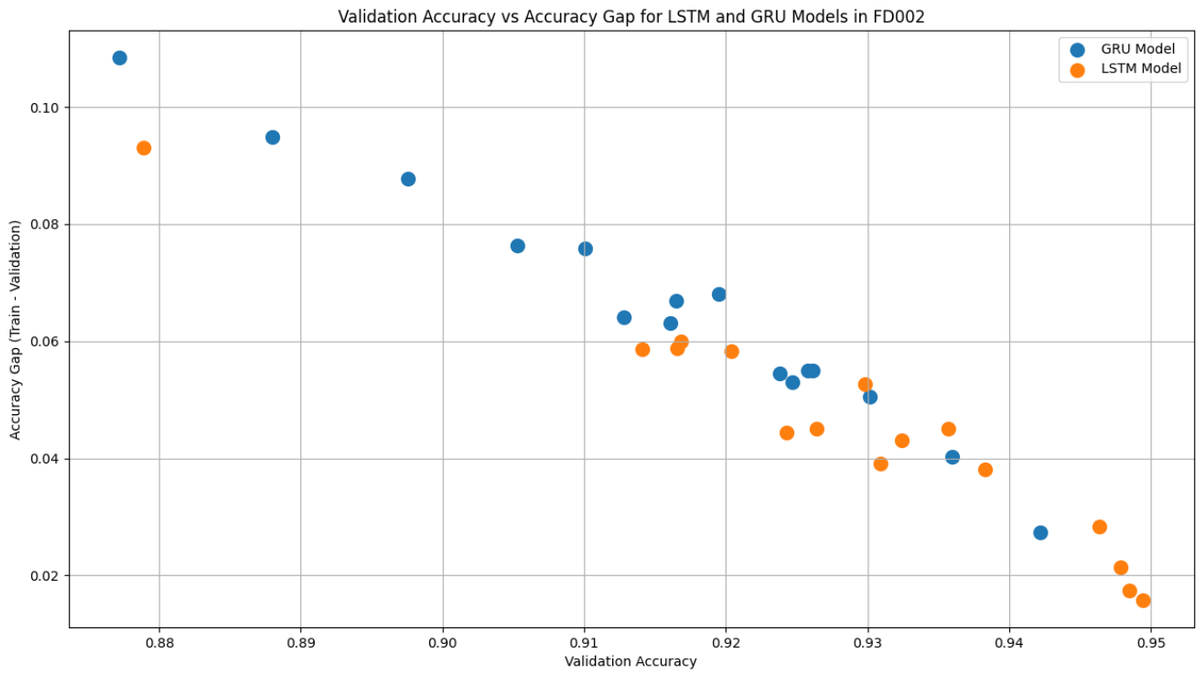


Appendix F - Hyperparameters vs Validation Accuracy for LSTM and GRU on FD002

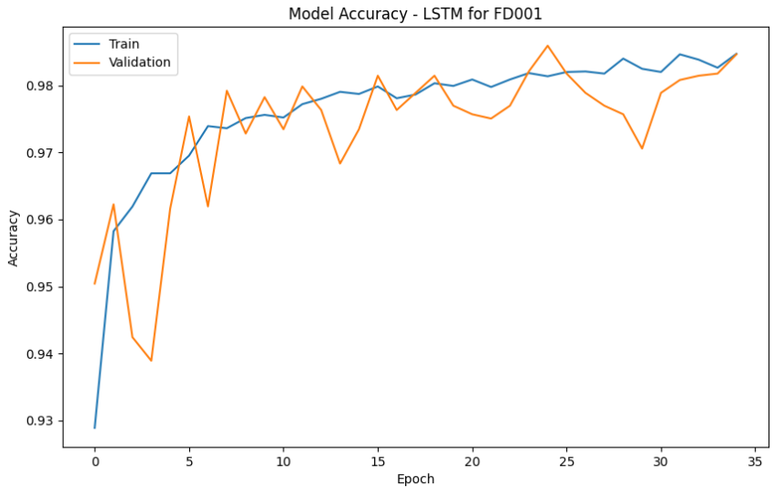


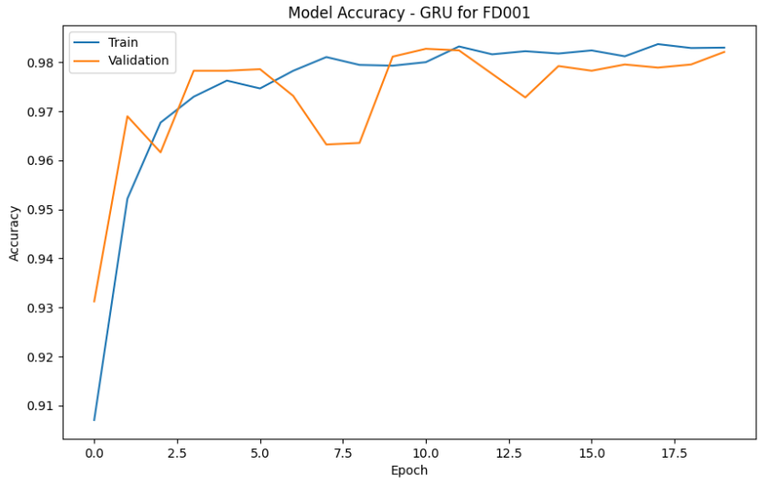
Appendix G - Validation Accuracy vs Accuracy Gap on FD001 vs FD002

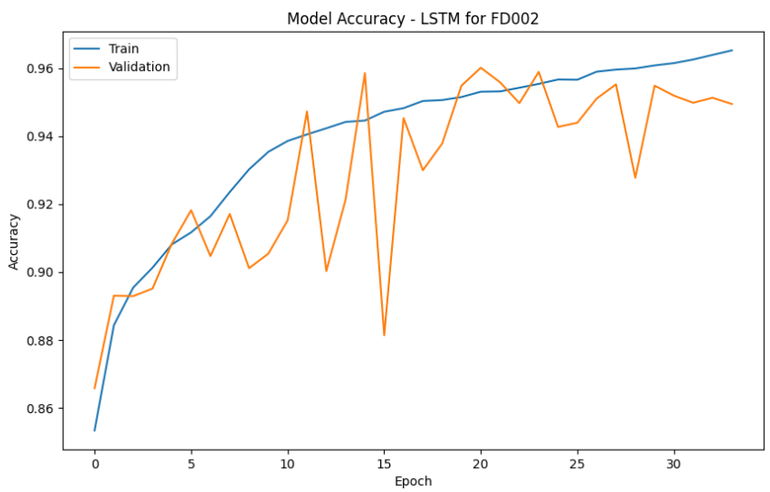


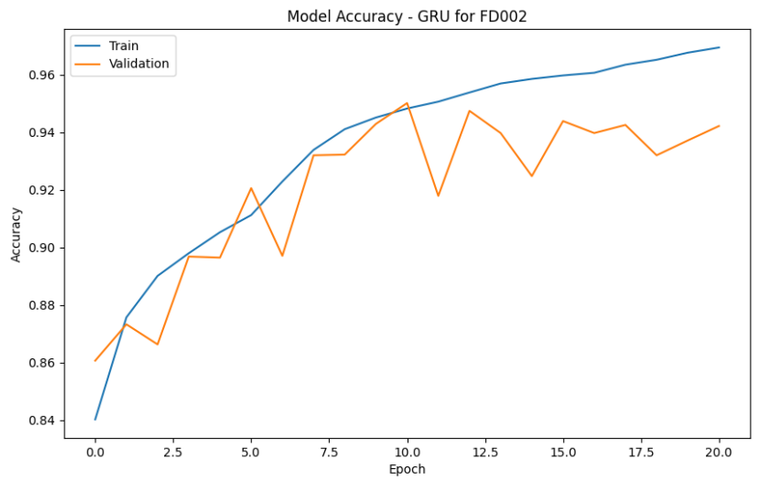


Appendix H – Model Accuracy on FD001 vs FD002

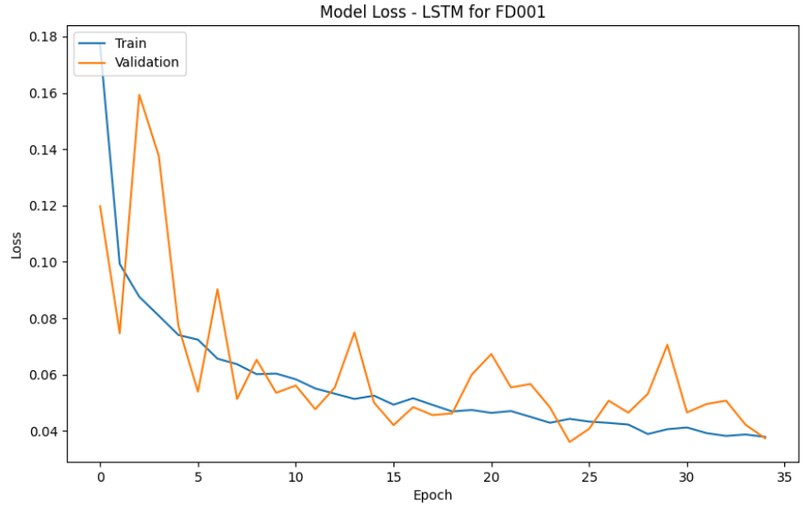


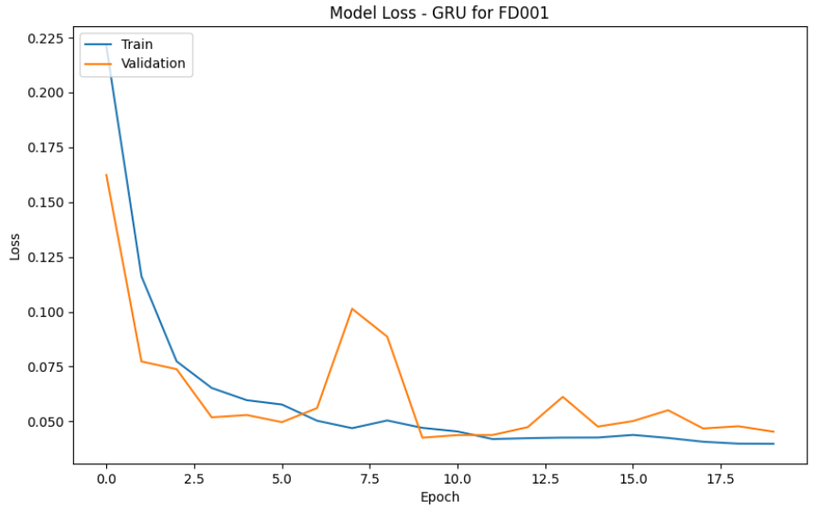


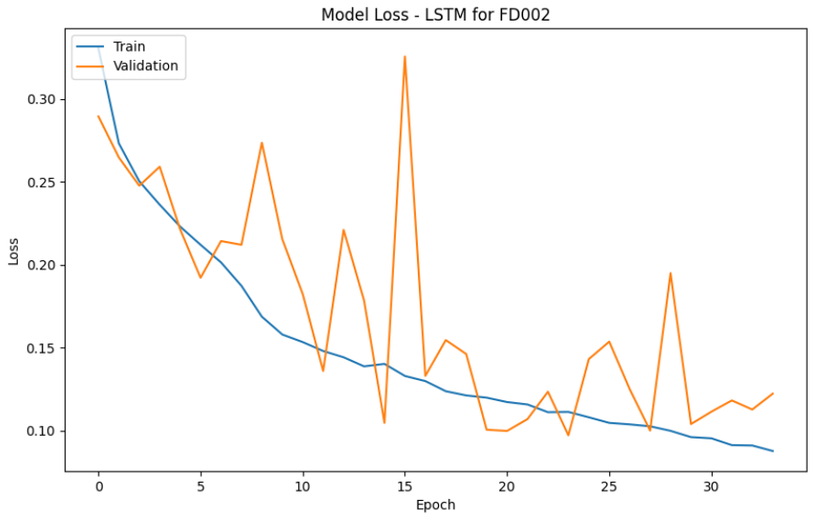


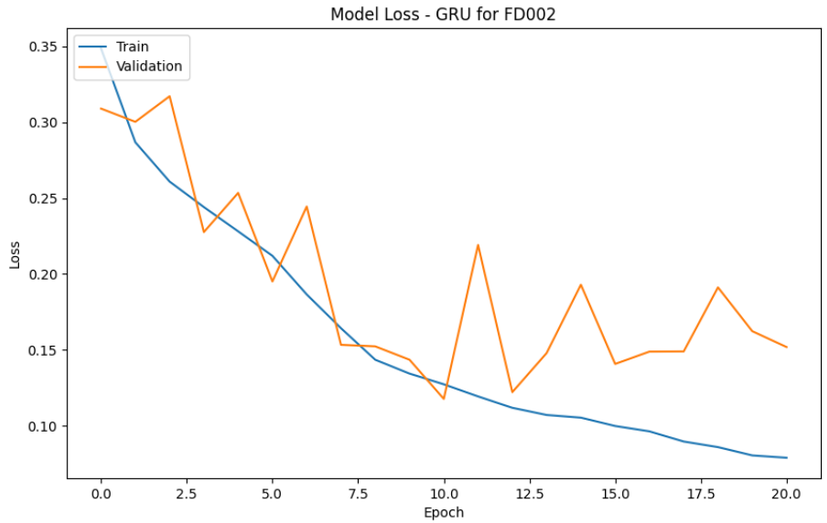


Appendix I – Model Loss on FD001 vs FD002

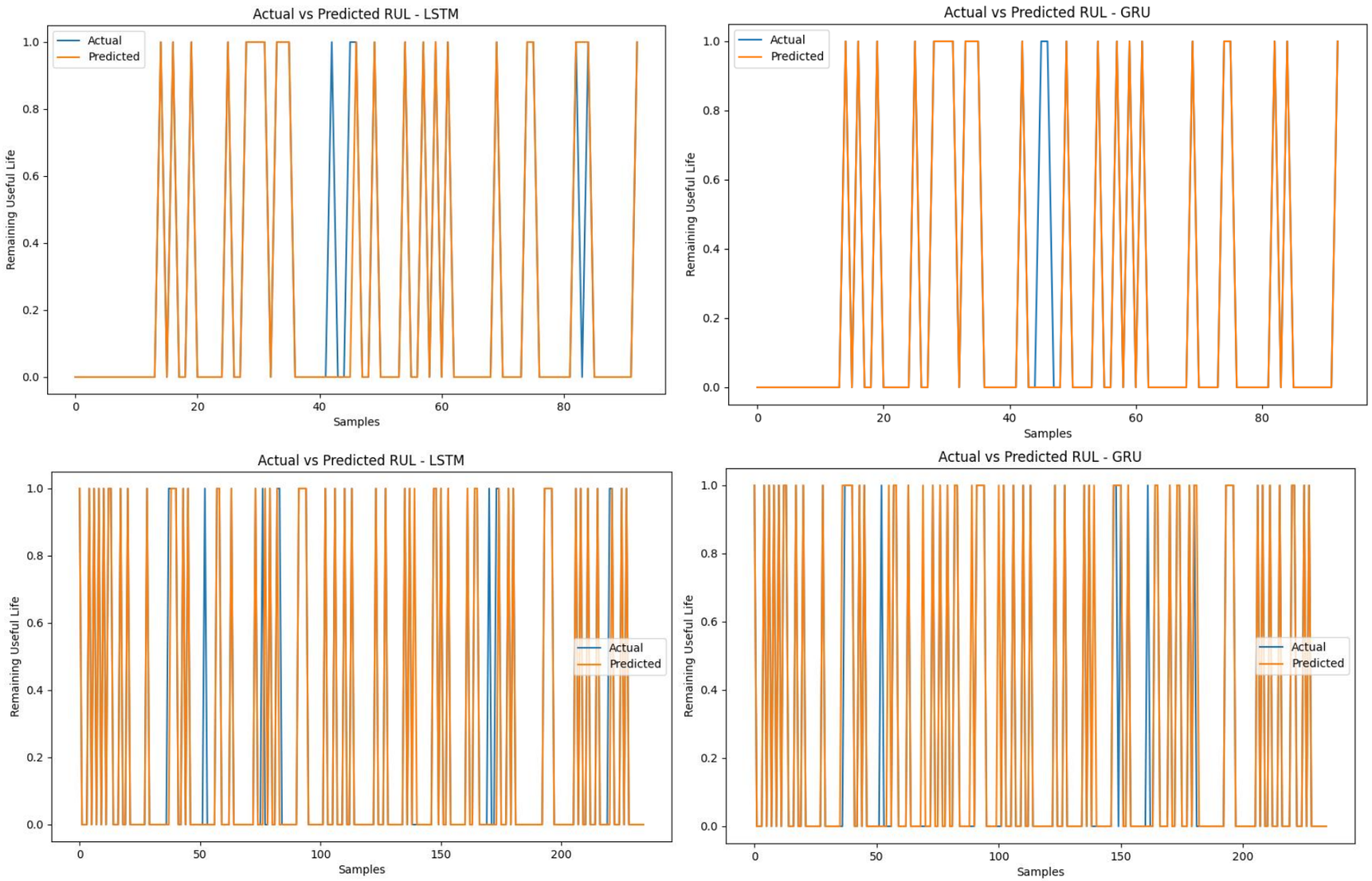




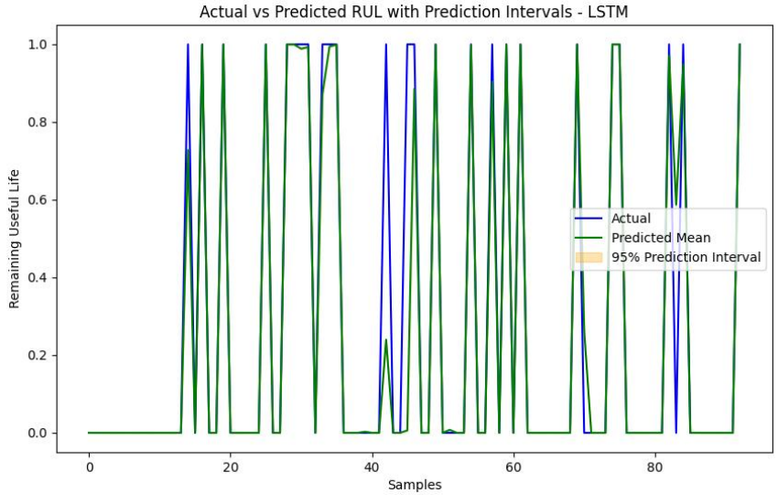


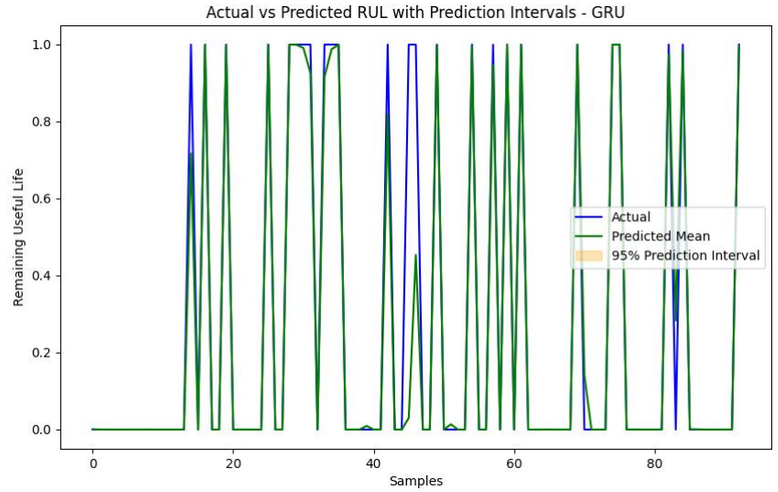


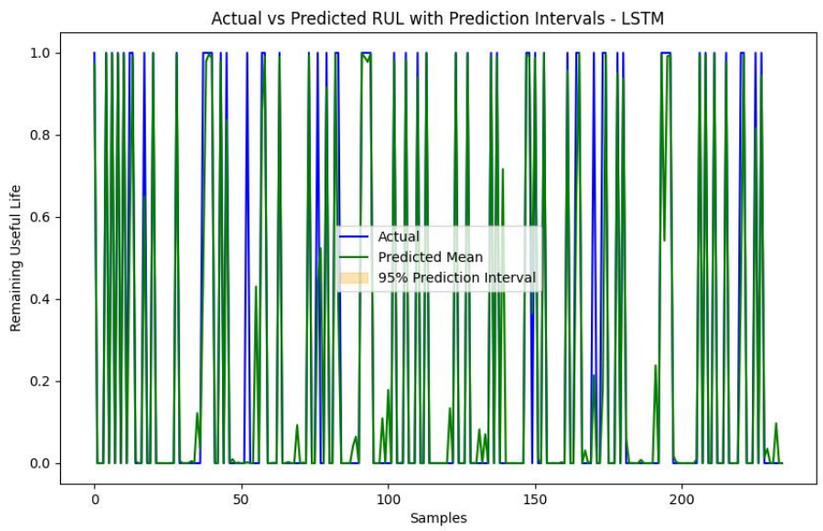
Appendix J – Actual vs Predicted RUL on FD001 vs FD002

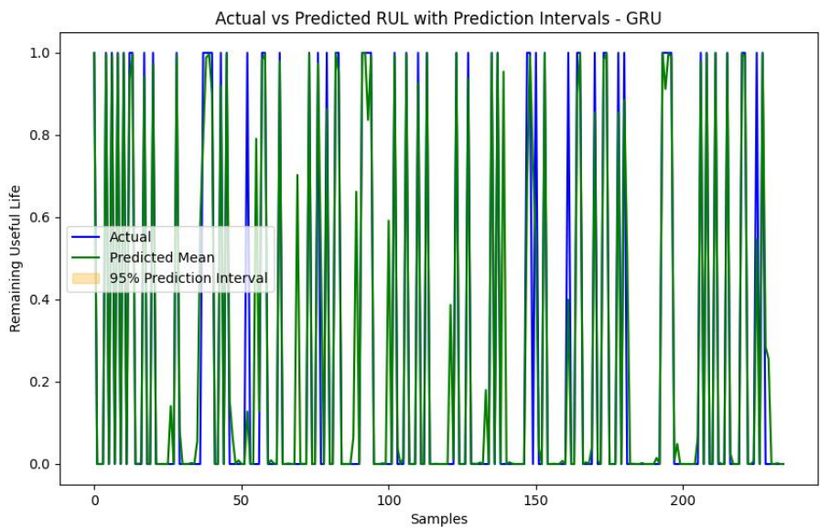


Appendix K – Actual vs Predicted RUL with Confidential Interval on FD001 vs FD002

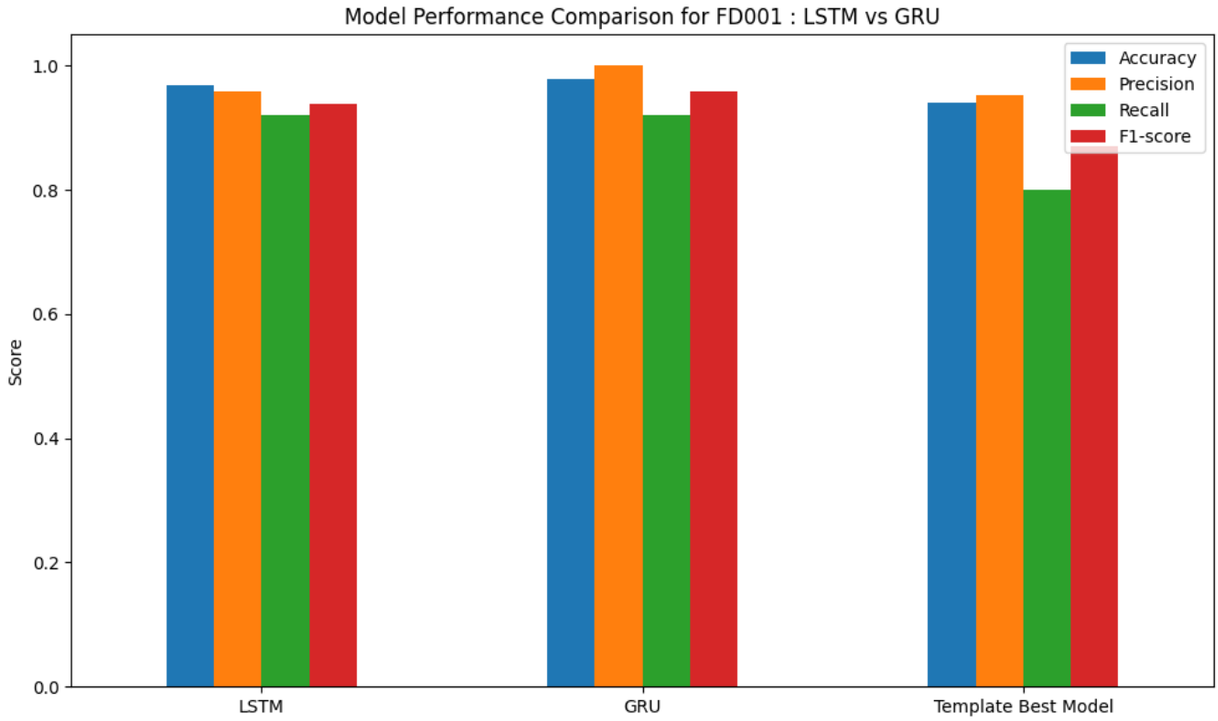


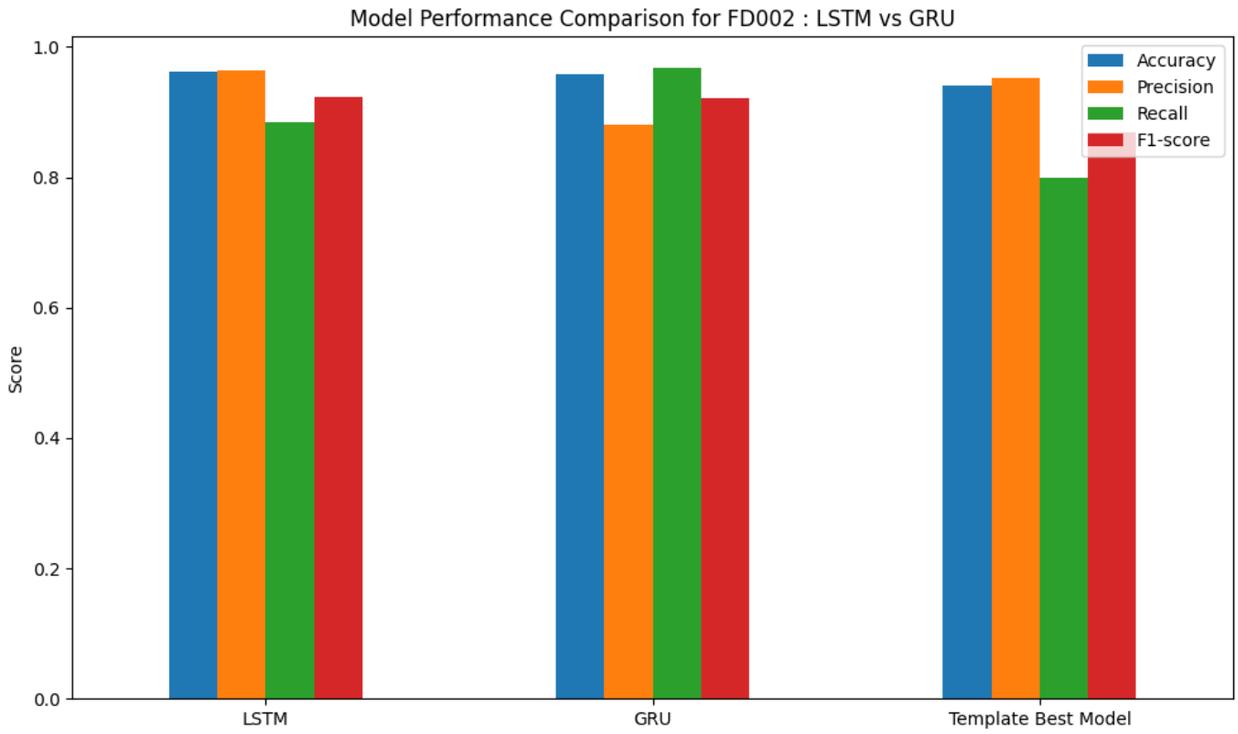






Appendix L – Comparison Metrics on FD001 vs FD002





Appendix M – ChatGPT Interaction