Engine Failure Prediction Challenge - Documentation

Model Selection Rationale

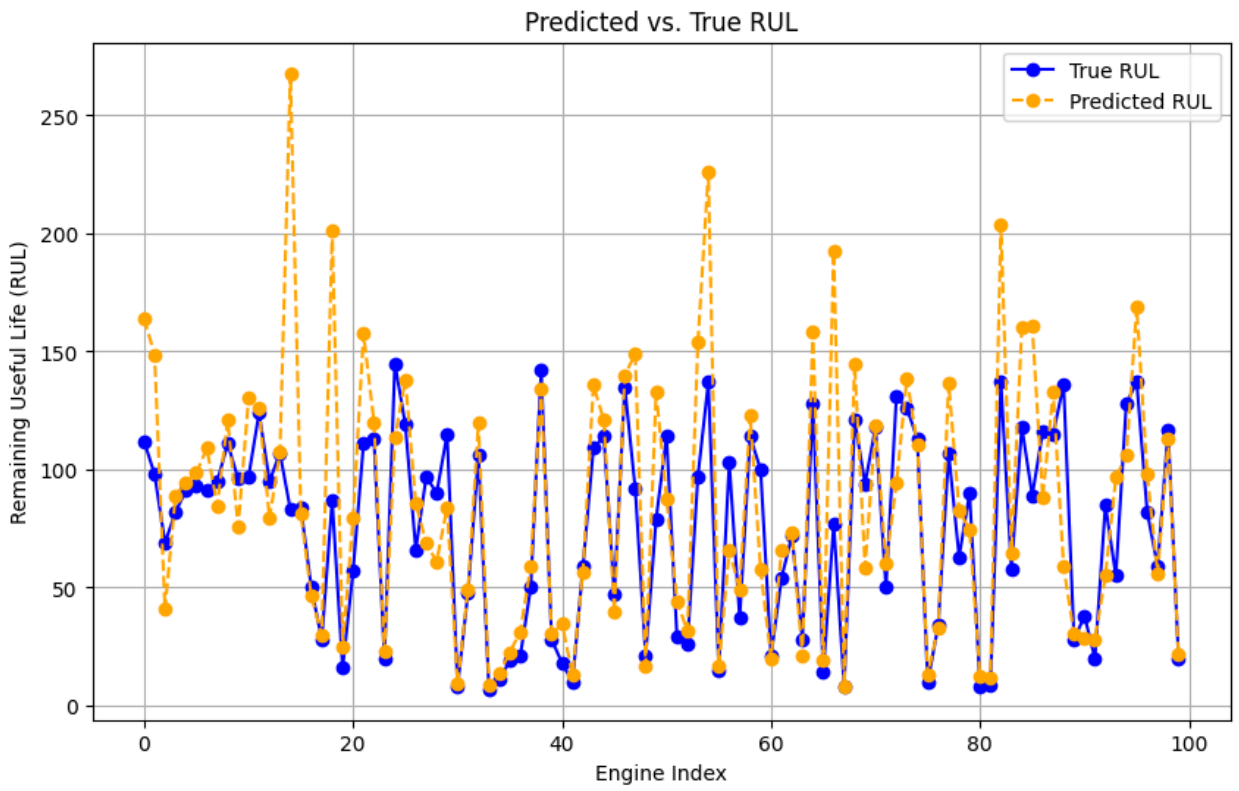
For this project, I ended up selecting a Long Short-Term Memory (LSTM) for this prediction task. The temporal nature of the engine data, where the timeline of events matters for predicting engine failure, made LSTMs an appropriate choice since LSTMs are particularly suited for sequential data due to their ability to retain memory through their cell states. My network architecture consists of three LSTM layers, each with 128 hidden units, followed by three fully connected layers. The first two fully connected layers use ReLU activation functions, while the final layer outputs the predicted Remaining Useful Life (RUL) without activation. A dropout rate of 0.5 was applied to mitigate overfitting, which was observed in earlier implementations. The mean squared error (MSE) loss function was chosen to penalize large deviations between predictions and ground truth. Additionally, a learning rate scheduler was incorporated to fine-tune the learning process, especially as the model began to overfit during initial training.

Feature Engineering Decisions

As part of the data pre-processing, I initially dropped the first two columns from the data set which were the unit identifiers and the current cycle number since they were mostly there for labeling purposes rather than being predictive features. I also standardized all features to have a zero mean using standard scaling to ensure consistent feature contribution and facilitate convergence during training. Since I was dealing with temporal data, I used the sliding window technique with window size 30 to maintain the temporal structure of the data since feeding the model a cycle of data does not help its prediction.For the training data, the final remaining useful life(RUL) value of the final sequence per window was calculated and added at the end. For testing data, I used only the last window per engine to predict RUL since I was operating under the assumption earlier windows contributed less to the current state of the engine.

Performance Metrics and Key Findings

Root Mean Squared Error (RMSE) was the primary evaluation metric while I also made comparisons of the predicted RUL values against ground truth values from the RUL.txt file after training for 500 epochs. For my key findings, based on the graph of the predicted RUL values vs the ground truth, I found that my model was rather decent at capturing degradation trends, but for predictions with high RUL values, there was some overfitting as the predicted value deviated significantly from the ground truth value. With the model trained at 500 epochs, my average RMSE was 33.37 while my last RMSE was around 32.2, which while not perfect is still not far off. This is consistent with the previous iterations of the model I had where I ran into significant overfitting issues with my model, and there is definitely room for improvement if I trained longer or experimented more with the hyperparameters.



Limitations and Assumptions

There definitely could have been more feature engineering I could have done. I utilized all 24 sensor data for the prediction and only dropped the labels of unit id and cycle number, but the dataset could have been simplified and the process made less computationally expensive if I eliminated other features with low variance. This implementation assumes all 24 feature readings are important for engine failure prediction but some key readings might have really low variance and not really play a large influence in final prediction. The model can struggle with really high RUL values and having the sliding windows is crucial for prediction. Removal of sliding windows and simply feeding in the training data would drastically lower its performance(from prior implementation experience). Some assumptions made from this particular implementation is that the sensor data is completely accurate as I did not eliminate any outliers from the dataset, and also since I only had to train it on the first dataset in the entire kaggle dataset, I did not need to worry about external factors like operational conditions such as weather.

Implementations Recommendations

This system could be used for proactive maintenance planning where given the current state of an engine, the model could be used to predict whether or not the respective engine is close to engine failure or not. I think most importantly, this model could be integrated into a broader maintenance framework where it focused on scoring engines by urgency where it gave predictions given the current state of the engine and the broader framework could score engines

based on the predicted RUL value as part of a priority queue. Ideally this model or the broader framework would be implemented for real-time prediction for testing purposes.