NASA RUL Report

Assignment

This assignment focuses on predictive maintenance by predicting the Remaining Useful Life (RUL) of aircraft engines using both Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU). NASA CMAPSS Turbofan Engine dataset was used to develop and compare the performance of these models.

Data Collection

The FD001 dataset from the NASA CMAPSS Turbofan Engine dataset was utilized for this assignment. I used the train_FD001 for the training data, test_FD001 for the testing data, and the RUL_FD001 for the truth data. The data was ingested into data frames and then various tests were run to check for high correlations, low variances, and missing data.

Data Preprocessing

This phase was divided into two parts beginning with processing the training the data and the second part was processing the testing data. Both parts included calculating the maximum cycle value for each engine, calculating the RUL, generating binary labels, and normalizing the data. Within the testing data, the truth_df was used to generate the maximum cycle value for test data to calculate the RUL for the dataset. Weights were then calculated for both labels to help balance the data but were not used in the model. The binary labels are as follows: Label1's threshold was 1 if x<=30 and 0 if else. Label2's thresholds were 1 if x<=30, 2 if x<=15, and 0 if else.

Model Development

LSTM

I chose a sequence length of 50 and used 19 feature columns for the model. An LSTM layer with 32 units processes these sequences, which have a length of 50 and 19 features. The LSTM returns the output only from the last time step, which is common in many sequence models. To help prevent overfitting, a dropout layer with a 20% dropout rate was added.

The final layer is a dense layer with 1 unit, using a sigmoid activation function, which is appropriate for binary classification tasks. The binary cross-entropy loss function was selected since it is standard for binary classification problems.

The Adam optimizer was used to adjust the model weights. Adam is one of the most popular optimization algorithms in deep learning because it combines the benefits of momentum and adaptive learning rates, making it effective and efficient for training.

During training, early stopping and model checkpointing were employed. Early stopping halts the training process if the validation loss does not improve for 10 consecutive epochs, which helps prevent overfitting. Model checkpointing saves the best model, based on validation loss, to a file named best_model.keras, ensuring that the model with the lowest validation loss is retained.

Key hyperparameters include 50 training epochs, a batch size of 32, a validation split of 20%, and the two callbacks (early stopping and model checkpointing).

GRU

For the gated recurrent unit (GRU) model, I used a sequence length of 50 and 19 feature columns, which are the same features that were used in the LSTM model. The first GRU layer contains 128 units, with L2 regularization applied to help prevent overfitting. This layer outputs the full sequence for further processing by the next GRU layer. After this, a dropout layer with a rate of 30% is added to reduce overfitting.

A second GRU layer with 64 units is included, which only returns the final output (i.e., return_sequences=False). Another dropout layer with a rate of 30% follows to continue addressing overfitting risks. The final layer is a dense layer with 1 unit, utilizing a sigmoid activation function for binary classification.

Similar to the LSTM model, binary cross-entropy is used as the loss function, while the Adam optimizer is employed with a learning rate of 0.001. The model was trained for 50 epochs with a batch size of 32. Early stopping was applied to halt training if validation loss stopped improving, and model checkpointing was used to save the best-performing model based on validation loss.

Model Evaluation and Comparison

LSTM

Test Set

Accuracy: 97.84%

Loss: .0426

The model performs exceptionally well, with high accuracy, precision, and recall. The confusion matrix shows a small number of misclassifications, but the overall performance metrics indicate the model is reliable.

Precision(95%) and recall(98%) shows a balanced ability to correctly identify both positive instances and avoid false positives. The high accuracy, combined with these metrics, suggests that the model generalizes well and is not overfitting.

Validation Set

Accuracy: 95.70%

Loss: .0686

The model has performed quite well on the validation set with a 95.70% accuracy, strong precision, recall, and an F1-score of 0.95.

The confusion matrix shows that while the model makes a few false positive predictions, it makes no false negative predictions, meaning it is very effective at identifying actual positives.

Overall, this means this model is well trained and generalizes well to unseen data.

GRU

Test Set

Accuracy: 1.00

Loss: .0547

At first seeing 100% accuracy looks like a good thing, but there is most likely overfitting occurring in this model. Precision, recall, and F-1 score are all 100% also which also indicate overfitting. The loss is low, which would indicate that the model is generalizing well, but since the other scores are so high this can be misleading. I would not recommend using this model.

Validation Set

Accuracy: 88.24%

Loss: .9002

The accuracy is acceptable, but not as good as the test set accuracy. This gives more indication that overfitting is taking place. The loss is relatively high and shows that the model is not performing well in making accurate predictions. Precision, recall, and F-1 are low further proving that the model is not accurate or reliable. These scores show that there is overfitting happening with the training data and class imbalance. Further tuning would have to take place for this model to be acceptable to use.

LSTM vs. GRU

Overall, the LSTM model outperforms the GRU model. The LSTM has higher accuracy, precision, recall, and F-1 score. This model is also not subject to overfitting and generalizes well to unseen data. The GRU model is struggling with overfitting and class imbalances and has very poor performance in comparison to the LSTM model. I found it very difficult trying to tune the GRU to get better results. When I would introduce class weights it often made it worse. I would like to see the utilization of cross-validation to ensure the model is tested on multiple subsets of the data. This could help with the overfitting issue that is occurring within the current GRU model.

The LSTM has the advantage over the GRU regardless due to its more complex structure. LSTMs generally perform better because they can learn and retain longer sequences due to their ability to manage and store long-term dependencies. GRUs are faster and it would be more beneficial to use one if there are fewer data points. LSTMs are recommended to use for time series forecasting and sequence to sequence tasks.

With the current models I would recommend using the LSTM because the metrics are all around better than the GRU model. I will be introducing k-fold cross validation to the GRU model in hopes of eliminating the overfitting issues that I'm running into.

GRU with K-Fold Cross Validation

Utilizing f-fold cross validation does not improve the performance of the model. Further in-depth tuning will have to occur to improve the performance of this model to make it competitive to the LSTM model. Right now the LSTM model is the better performing and will be preferred over the GRU.