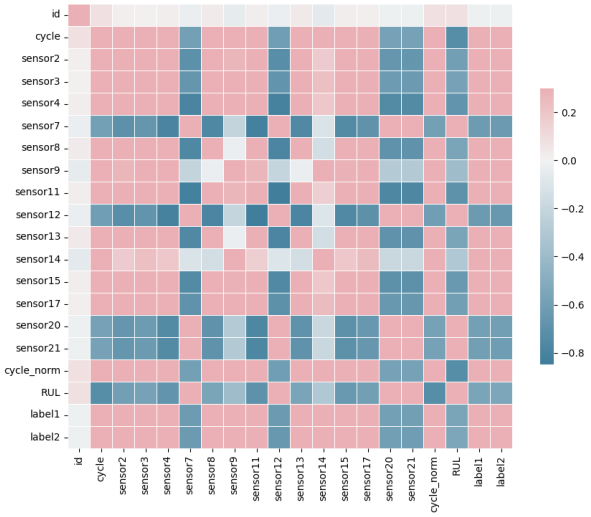
95820 Assignment 1 Report – George Pan

A. Feature Selection

By printing out the dataframe, I first noticed that columns 26, 27 are all NaN values. Therefore, I first drop these two columns. In addtion, settings 1, settings2, settings 3, sensor 1, sensor 10, sensor 18 and sensor 19 have almost zero standard deviation, which indicates that they carry low information for model learning. I will drop these features as well. Finally, I will drop sensor 5, sensor 6, sensor 16 which have identical mean, minimum, and maximum values, indicating no variability.

By following the starter-notebook, we can then construct our final training and testing dataframe. Below is the correlation matrix of the training dataframe.



As we can see, after dropping rebundant columns, most remaining features have fairly good correlation with each other.

B. Training

For fair comparison, I used the same training hyper-parameters for both LSTM and GRU. The details are listed below.

i. Epochs = 100
ii. Batch Size = 64
iii. Early Stopping
    a. Patience = 10
    b. Min Delta = 0.00001

C. Precision, Recall, and F1- Score

Before diving into the performances of LSTM and GRU, I would like to introduce the metrics used to measure the model performances. In addition to accuracy, precision and recall scores are also used:

a. $Precision\ Score = \frac{True\ Positive}{True\ Positive + False\ Positive}$
b. $Recall\ Score = \frac{True\ Positive}{True\ Positive + False\ Negative}$
c. $F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

As we can see above, precision score accounts for the fail cases where the model incorrectly classify instances as positive. On the other hand, recall score accounts for the fail cases where the model cannot identify positive cases. Finally, F1 score is simply the harmonic mean of precision and recall.

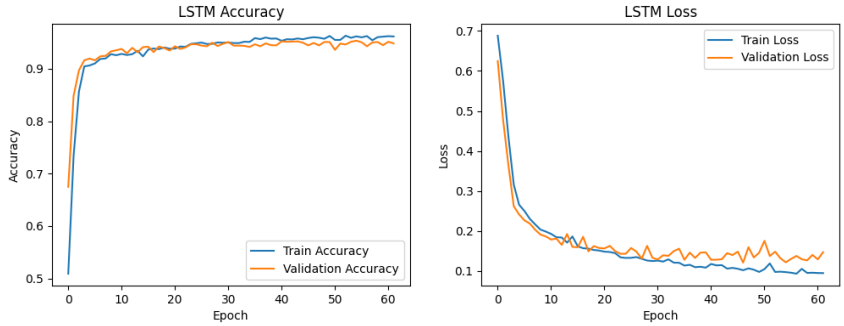
D. LSTM

After finishing data pre-processing, we can start training both the LSTM and GRU. In this section, I will show the results of LSTM. To limit the model parameters under 10,000, I use only a single layer LSTM with sequence length equals to 128. The total parameters count is 745 which is significantly smaller than the limit, as shown below:

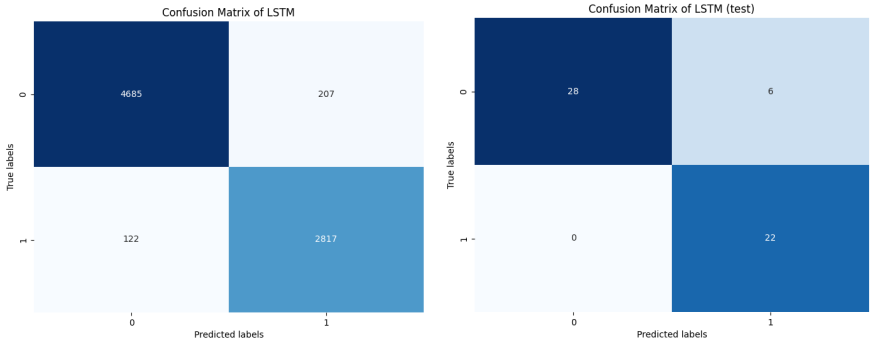
Layer (type)	Output Shape	Param #
lstm_16 (LSTM)	(None, 8)	736
dense_10 (Dense)	(None, 1)	9

Total params: 745 (2.91 KB)
Trainable params: 745 (2.91 KB)
Non-trainable params: 0 (0.00 B)

Here is the training and validation loss/accuracy of LSTM.

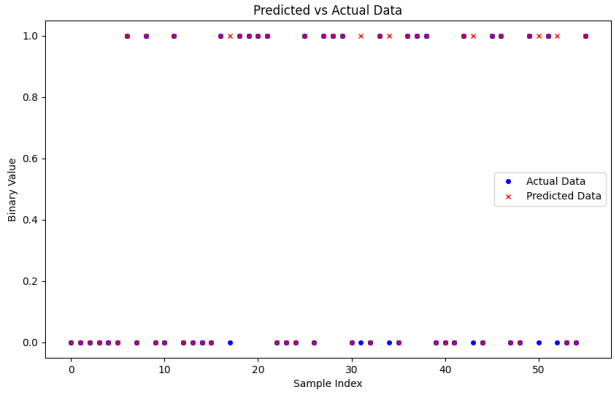


From the loss curves, we can see that LSTM slightly overfit the training dataset with the validation loss exceeding the training loss. In addition, we can also plot the confusion matrix of LSTM on training set and testing set .



As we can see, LSTM achieved a fairly good performance on predicting the Remaining Useful Life of each engine.

Finally, we can visualize the data points of predicted sample and true labels.



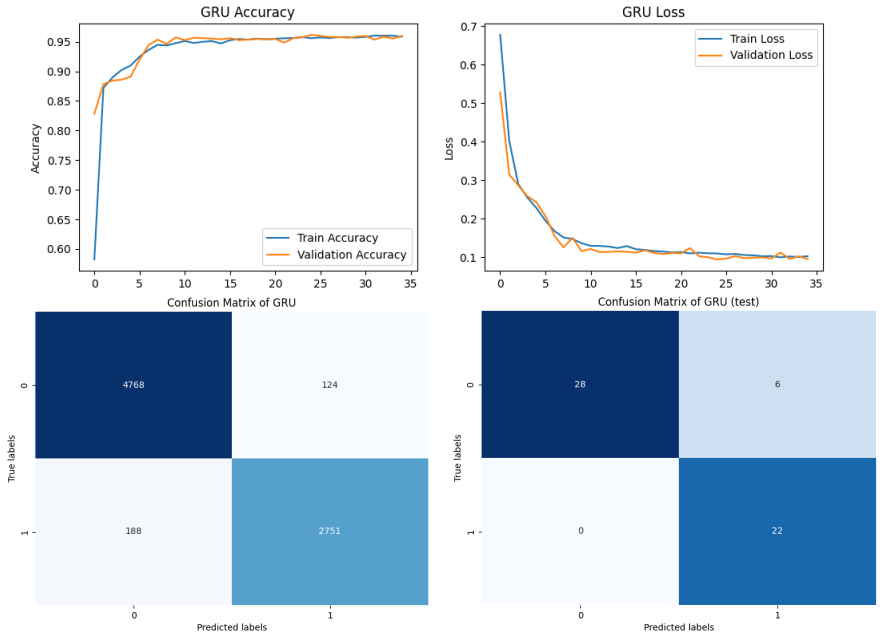
E. GRU

Similarly, we can do the same analysis on GRU. The GRU model consists of two layers with Dropout layers as shown below:

Layer (type)	Output Shape	Param #
gru_4 (GRU)	(None, 8)	576
dense_11 (Dense)	(None, 1)	9

Total params: 585 (2.29 KB)
Trainable params: 585 (2.29 KB)
Non-trainable params: 0 (0.00 B)

The training and validation loss/accuracy curves, confusion matrices are shown in figures below:



As we can see, GRU does not show the same over-fitting phenomenon as LSTM. This is probably due to the less parameters of GRU compared to LSTM. In addition, GRU has less false positive but more false negative cases compared to LSTM on the training set. However, both of them have similar performances on the testing set.

F. LSTM V.S. GRU

In this section, we will discuss more details about the comparison between LSTM and GRU.

Metrics\Models	LSTM	GRU
Training Accuracy	0.958	**0.960**
Training Precision	0.932	**0.957**
Training Recall	**0.958**	0.936
Testing Accuracy	0.893	0.893
Testing Precision	0.786	0.786
Testing Recall	1.0	1.0
Testing F1 Score	0.88	0.88

Although LSTM has a slightly higher recall on the training set, GRU shows marginally better performance in accuracy and precision with fewer trainable parameters. This suggests that GRU might be a more efficient model for learning certain patterns in the training data.

However, when evaluated on the testing set, both models demonstrate the exact same performance across all metrics, including accuracy, precision, recall, and F1 score. This identical performance suggests that both LSTM and GRU generalize equally well to unseen data for this particular task, despite the differences in their training results.

Interestingly, both models achieved a perfect recall score on the testing set. This indicates that neither model produced any false negatives, meaning they successfully identified all positive instances in the testing set. However, this might also suggest potential overfitting since perfect recall might not always align with optimal generalization. Given that precision is relatively lower, the models might be over-predicting the positive class, which could lead to a trade-off between precision and recall.

In conclusion, while GRU appears to be more efficient during training with slightly better precision and accuracy, both models perform identically in testing, implying that the choice between LSTM and GRU might depend on computational efficiency and the specific dataset in use. Further investigation into

precision-recall trade-offs and the evaluation of additional metrics like the confusion matrix may help in making a final decision on model selection.

G. Model Architecture

In addition to the models discussed above, I have also tried more layers of LSTM and GRU with Drop-Out layers to test the performances. Since both LSTM and GRU have the same trends of performances when increasing the number of parameters, I will show the details of larger LSTM only for more concise discussion. If interested, one can also check out the figures for larger GRU in the *visualizations* folder.

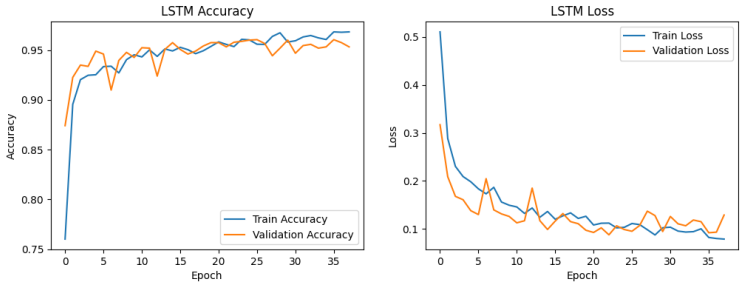
First of all, the following picture shows the model summary of larger LSTM. Note that the number of parameters is still below the constraint (10,000):

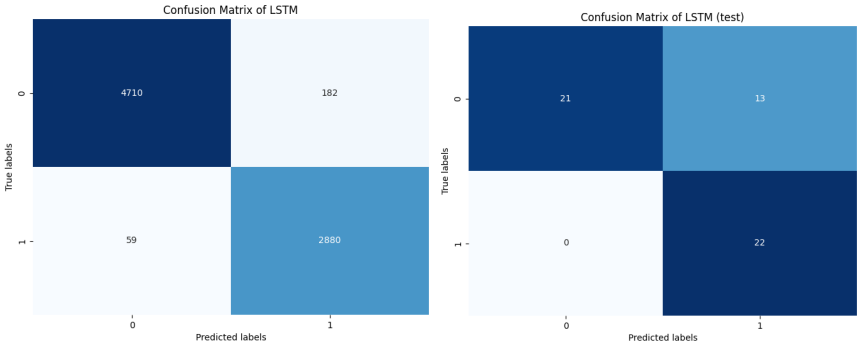
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 128, 32)	6,016
dropout (Dropout)	(None, 128, 32)	0
lstm_1 (LSTM)	(None, 16)	3,136
dropout_1 (Dropout)	(None, 16)	0
dense (Dense)	(None, 1)	17

Total params: 9,169 (35.82 KB)
Trainable params: 9,169 (35.82 KB)
Non-trainable params: 0 (0.00 B)

As we can see, the number of parameters for this large LSTM is about 11 times more than the smaller one. Ideally, we will think that this large model should perform better than the smaller one.

To investigate the training process and performance, we can check out the following loss/accuracy curves and the confusion matrices:





From the loss/accuracy curves, we can see that the larger LSTM is subjected to even worse overfitting phenomenon. In addition, the instability might be the result of Drop-Out layers due to the randomness of them. In general, the larger LSTM still perform well on the training set and validation set.

Now let's take a look at the performance on testing set. Below is the confusion matrix for testing set and the table to compare the performance between small and large LSTM:

Metrics\Models	Small LSTM	Large LSTM
Training Accuracy	0.958	**0.969**
Training Precision	0.932	**0.941**
Training Recall	0.958	**0.980**
Testing Accuracy	**0.893**	0.768
Testing Precision	**0.786**	0.629
Testing Recall	**1.0**	1.0
Testing F1 Score	**0.88**	0.772

As we can see, the larger LSTM has much worse performance on testing set compared to small LSTM. This is exactly the result of overfitting where the model fit too much toward the training set and fail to generalize to the unseen data. Therefore, I use the small version of both LSTM and GRU as the best models for this assignment.

H. Future Work

While LSTM and GRU already achieved decent performances, there are still possible improvements that can be made. As taught in classes, **Bi-directional LSTM (GRU)** might be one of these approaches. By seeing the future data of an engine, the models could possibly learn more patterns of the RUL of the engine, resulting in better predictions.

Another approach is the utilization of **ensemble method**. By combining LSTM and GRU, the performances might generalize better to testing data. By averaging predictions or using voting mechanisms, the strengths of both models can be combined, potentially reducing errors. Regularizations such as **L2-Regularization** could also help mitigate the overfitting problem.

Finally, attention mechanism could potentially improve the performances as well. Attention layers allow the model to focus on the most relevant parts of the sequence, which can improve the interpretability of the model and its performance by helping the model prioritize important temporal features.