EEC4400 - Group01 - Report

Student Details

1. Student 1: [Wang Ziqi], Student Number: [CEG24081]

2. Student 2: [Xie Zeliang], Student Number: [CEG24088]

3. Student 3: [Huang Jichen], Student Number: [CEG24028]

4. Student 4: [Zhang Junhan], Student Number: [CEG24108]

1. Data Exploration (All Students)

1.1 Cross-Correlation Heat Map

This study investigates the relationships among machine sensitivity parameters by calculating Pearson correlation coefficients and visualizing them through a covariance heatmap. The heatmap, ranging from dark to light, represents the transition from weak to strong correlations. Key findings include strong positive correlations between certain parameter pairs, such as s7 and s12, s8 and s13, and s9 and s14, which tend to increase or decrease concurrently. Conversely, strong negative correlations are observed between s7 and s15, s10 and s15, s12 and s15, s20 and s15, and s21 and s15, indicating that as one parameter increases, the other tends to decrease. Additionally, parameters setting3, s1, s5, s16, s18, and s19 are identified as constants, showing no correlation with other parameters, and are recommended to be normalized to zero in further normalization steps.

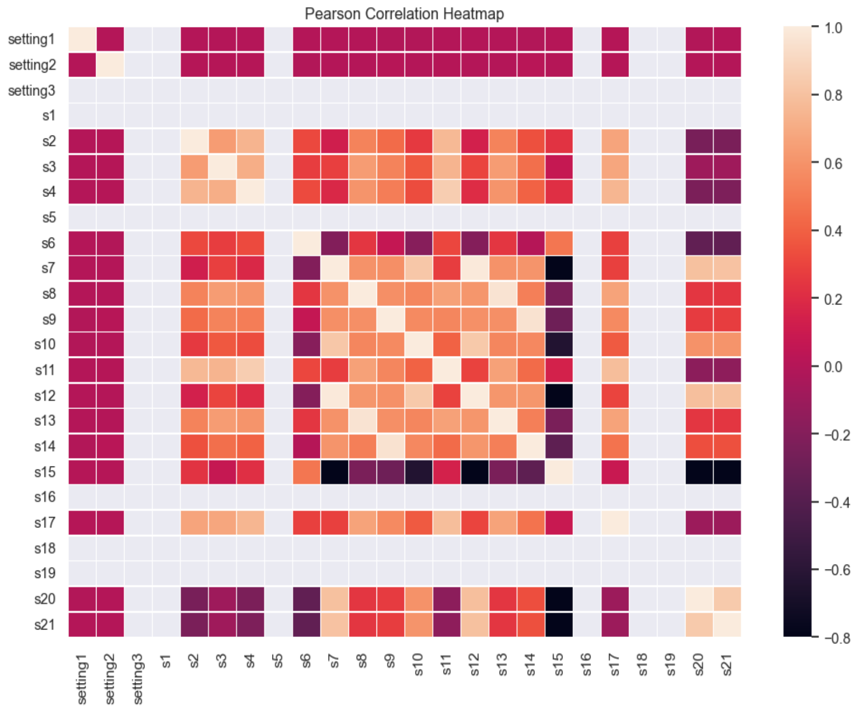


Figure1 Cross-Correlation Heat Map

1.2 Time Series and Histogram Data Plots

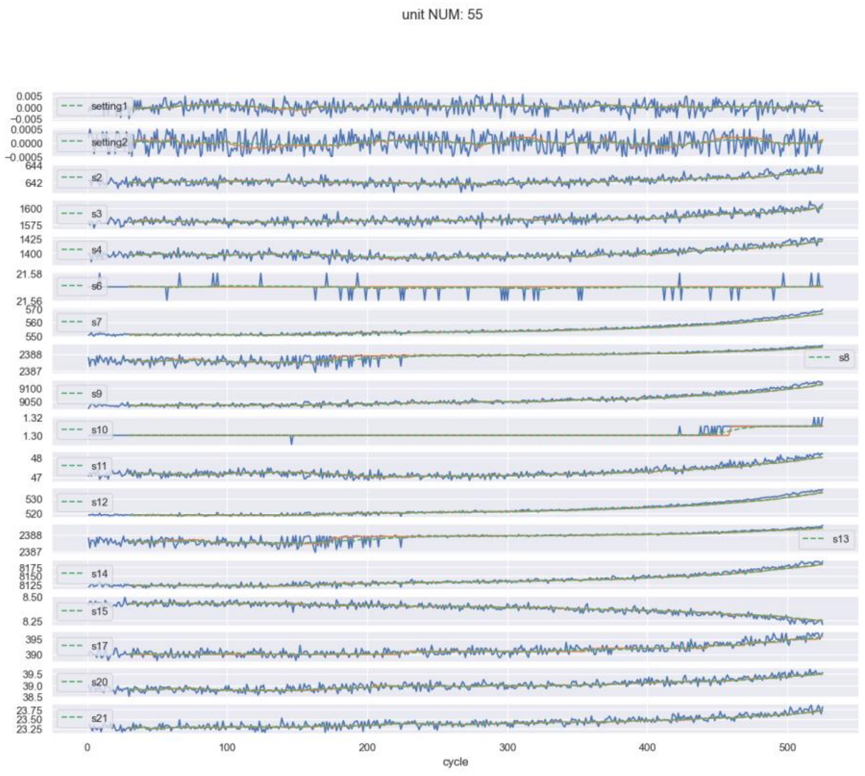


Figure2 Maximum operating time machine timing diagram

By plotting the maximum, median, and minimum values of sensor coefficients at different cycles, as well as their moving median and mean, it can be observed that in the shortest operating time: the sensor coefficients of s2, s3, s4, s8, s9, s11, s13, s15, and s17 exhibit an increasing trend with cycle progression, while s7, s12, s20, and s21 show a gradual decline. In the medium operating time: the sensor coefficients of s2, s3, s4, s8, s9, s11, s13, s15, and s17 continue to show an increasing trend, and s7, s12, s20, and s21 continue to decline. The longest operating time data exhibits more pronounced characteristics, with s2, s3, s4, s7, s8, s9, s11, s12, s13, s14, s17, s20, and s21 showing an upward trend, and s15 showing a downward trend. Additionally, the value of s6 remains relatively constant, s10 exhibits a sudden change at a certain point, the values of setting1 and setting2 fluctuate irregularly with the cycle, and the trends of s12 and s14 are very smooth.

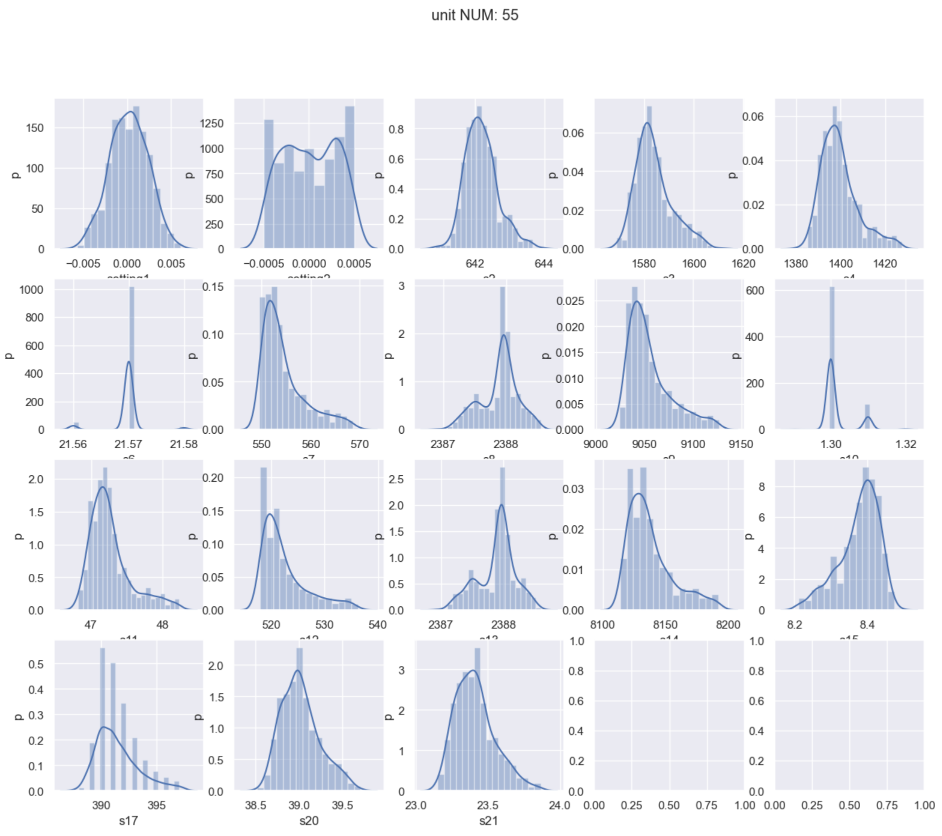


Figure3 Machine histogram of longest operating time

Upon examining the histograms of sensor readings, it is observed that the majority of sensor readings follow a normal distribution. In the longest operating time, setting1, s2, s3, s4, s7, s9, s11, s12, s14, and s15 show a positively skewed distribution, meaning that the majority (80%) of the data is concentrated within a specific range. It is also noted that setting2 is more uniformly distributed, while s6 and s10 have very concentrated data distributions, centered around one or two data points.

1.3 Correlation and Machine Lifetime

Based on the analysis, it can be inferred that as the machine's usage time increases, the values of s2, s3, s4, s7, s8, s9, s11, s12, s13, s14, s17, 20, and 21 will gradually increase, while the value of s15 will gradually decrease. Additionally, by analyzing the distribution of data points, it can be determined that supervised learning models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks can be used to make a rough prediction of the lifespan of a machine given its sensor readings. The same treatment can be applied to further refine these predictions.

2. Model Construction, Training, and Evaluation

2.1 Failure Classification – CNN (Student 1)

2.1.1 Baseline CNN Model Performance

When length=25：

Model structure: Using a three-layer convolutional structure, the first layer has 32 * 3 filters with added pooling layers, the second layer has 64 * 3 filters, and the third layer has 128 * 3 filters. After each convolutional layer, a pooling layer of size 2 is added with a dropout rate of 0.3.

Model convergence process:

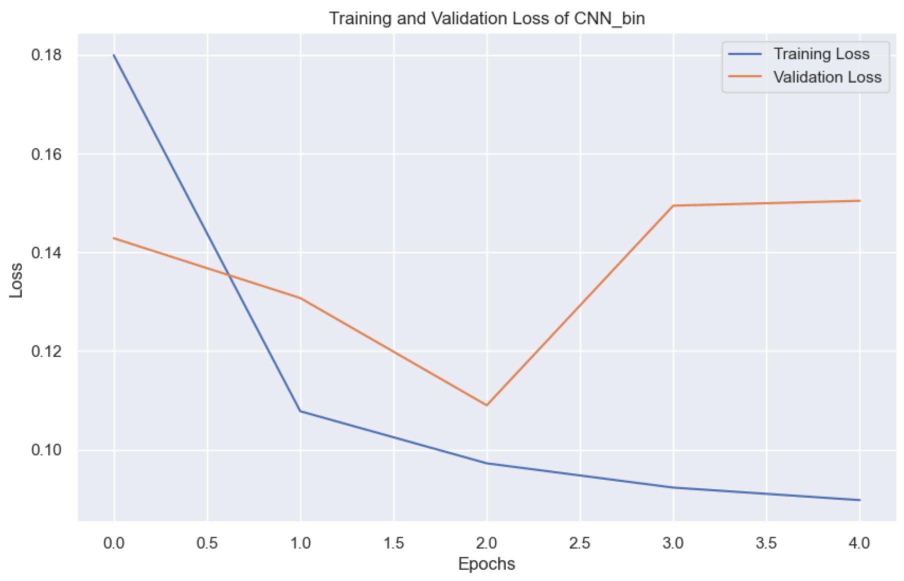


Figure4 Training and Validation Loss of CNN_bin,Seq_len = 25

Test set performance:

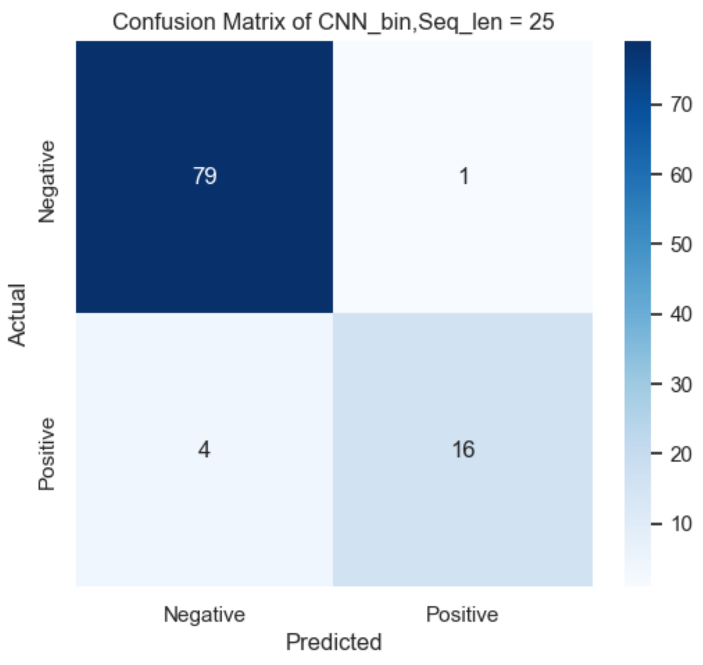


Figure5 Confusion Matrix of CNN_bin,Seq_len = 25

Accuracy on test data: 0.94

Precision: 0.85

Recall: 0.85

F1 Score: 0.85

Accuracy is the most intuitive performance metric, representing the proportion of correctly predicted samples in the total sample size. Accuracy measures the proportion of samples predicted as positive by the model that are actually positive, that is, how many of the predicted positive samples are correct. The recall rate measures the proportion of actual positive class samples that the model predicts as positive class, that is, how many actual positive class samples the model can find. The F1 score is the harmonic mean of precision and recall, attempting to find a balance between precision and recall. The higher the F1 score, the better the performance of the model. By observing the above data, it can be found that the predictive performance of the model is acceptable.

When length = 50:

Model convergence process:

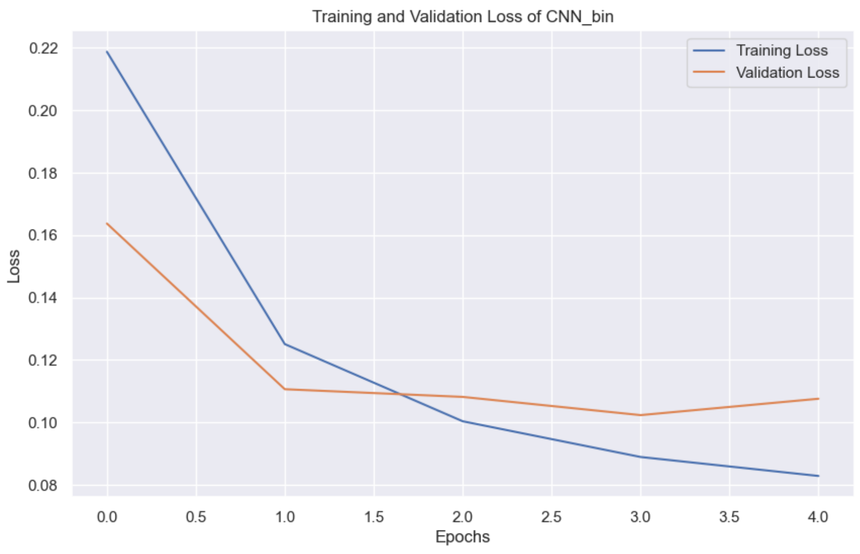


Figure6 Training and Validation Loss of CNN_bin,Seq_len = 50

Confusion Matrix:

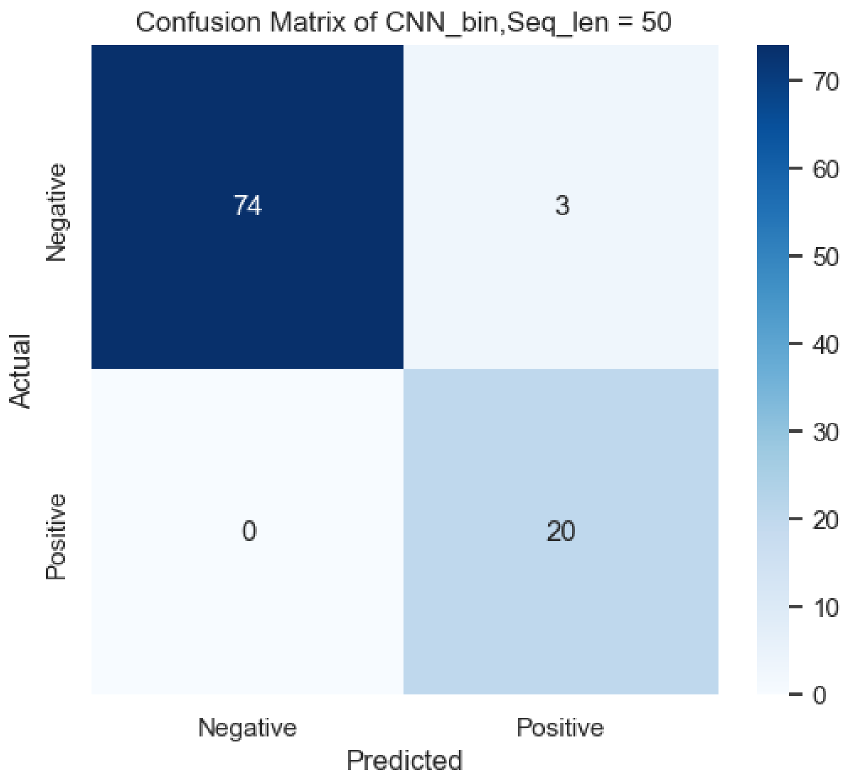


Figure7 Confusion Matrix of CNN_bin,Seq_len = 50

Accuracy on test data: 0.9690721649484536

Precision: 0.8695652173913043

Recall: 1.0

F1 Score: 0.9302325581395349

Sequence_length	Accuracy	Precision	Recall	F1 Score
25	0.94	0.85	0.85	0.85
50	0.969	0.869	1.0	0.93

2.1.2 Hyperparameter Optimisation

Model	Epoch	Drop Rate	Learning Rate	Batch Size
CNN	5	0.3	0.001	16
CNN_ALT	5	0.4	0.0005	16

Adjust the learning rate to 0.0005 and the dropout rate to 0.4

The convergence process of optimized model training:

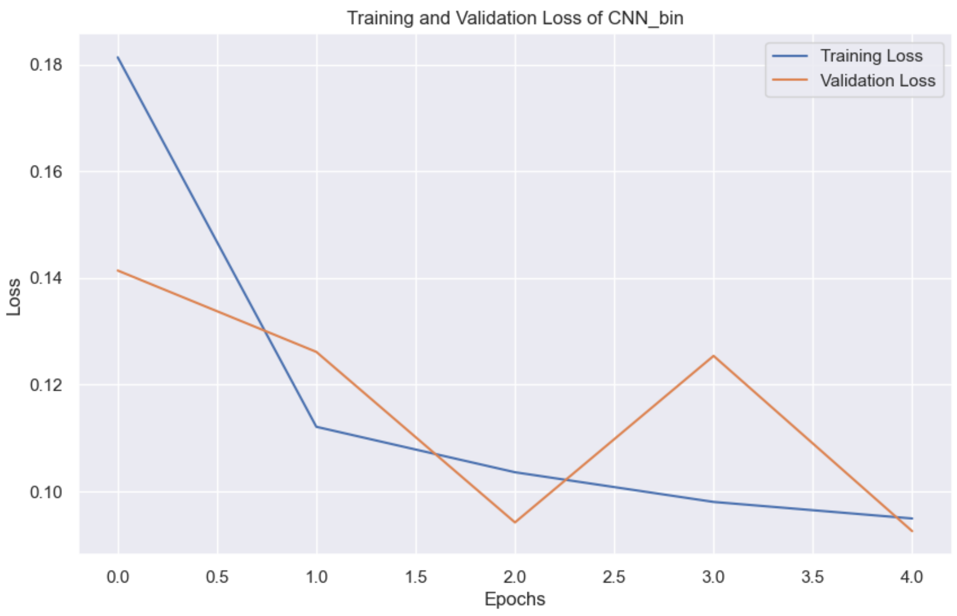


Figure8 Training and Validation Loss of CNN_bin_alt,Seq_len = 25

Test set performance:

Confusion matrix:

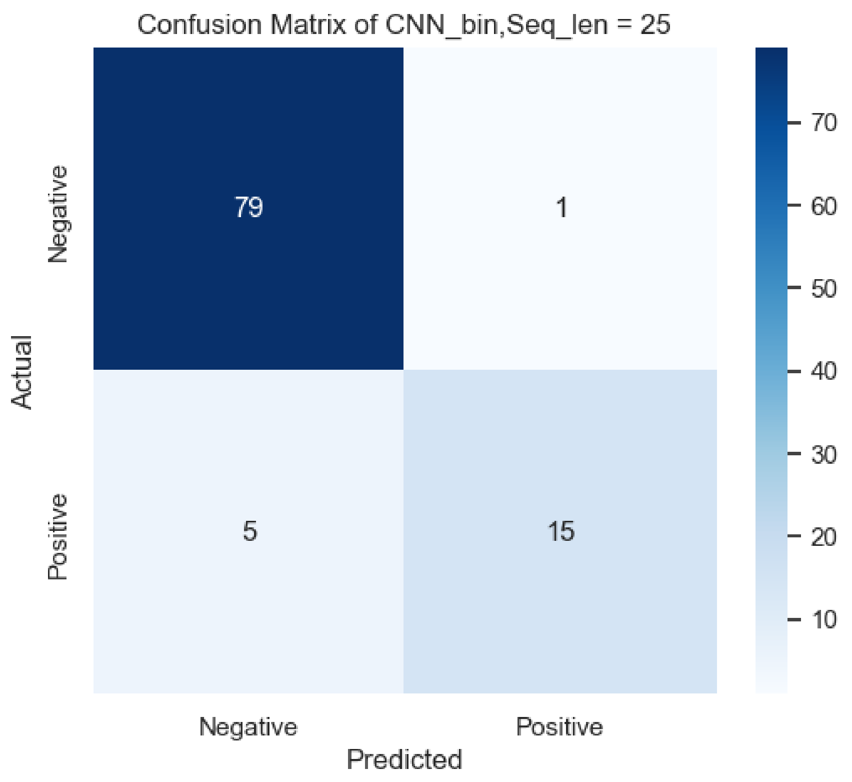


Figure9 Confusion Matrix of CNN_bin_alt,Seq_len = 25

Test Accuracy: 0.94

Test Precision: 0.9375

Test Recall: 0.75

Test F1 Score: 0.8333333333333334

When length = 50:

The convergence process of optimized model training:

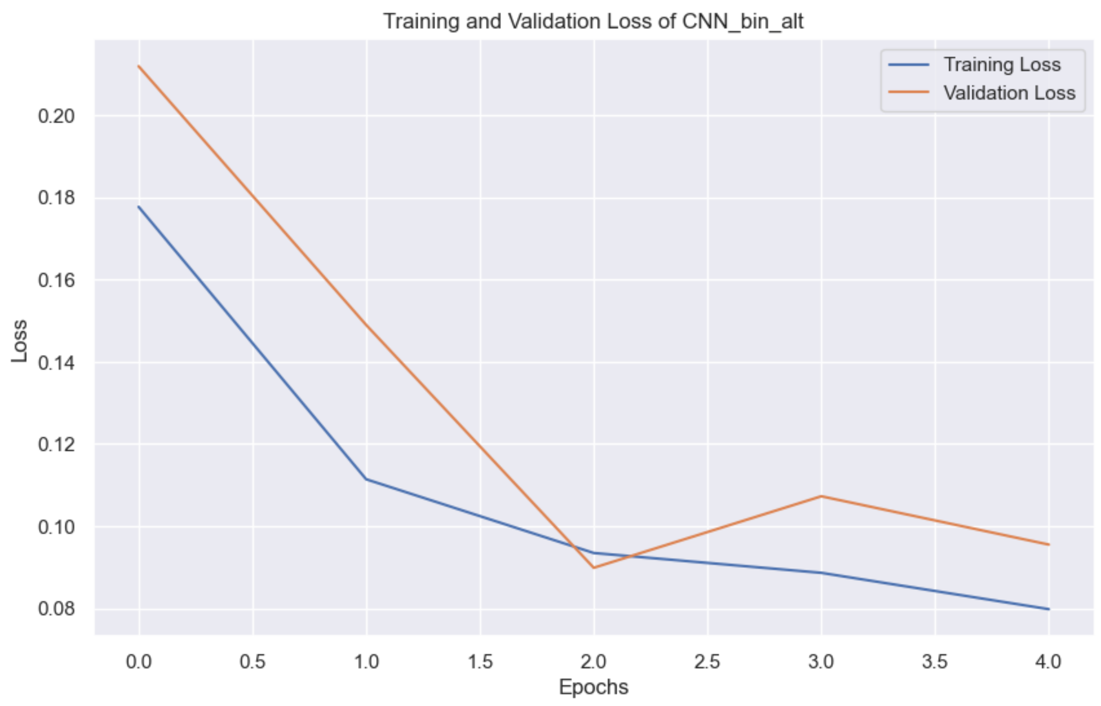


Figure10 Training and Validation Loss of CNN_bin_alt,Seq_len = 50

Test set performance:

Confusion matrix:

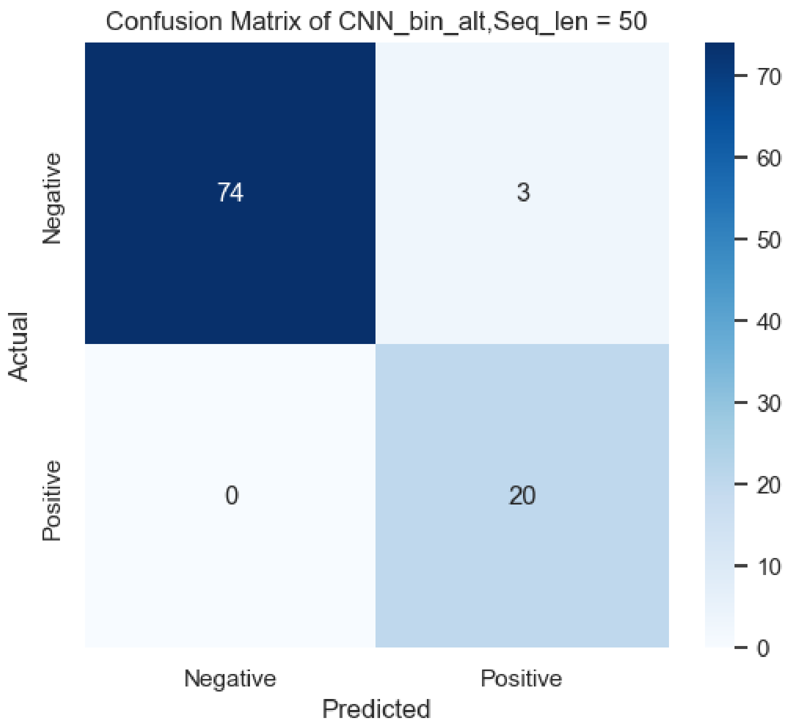


Figure11 Confusion Matrix of CNN_bin_alt,Seq_len = 25

Test Accuracy: 0.9484536082474226

Test Precision: 1.0

Test Recall: 0.75

Test F1 Score: 0.8571428571428571

Sequence_length	Accuracy	Precision	Recall	F1 Score
25	0.94	0.93	0.75	0.83
50	0.948	1.0	0.75	0.85

After optimization, it can be found that the precision has increased from 0.93 to 0.94, indicating that Dropout can reduce the model's dependence on training data and improve its generalization ability At the same time, Dropout forces different parts of the network to learn independent feature representations, which helps the model learn more robust features. A smaller Learning Rate can ensure the stability of the training process, but may result in slow convergence speed.

2.1.3 TensorBoard Analysis

length = 25

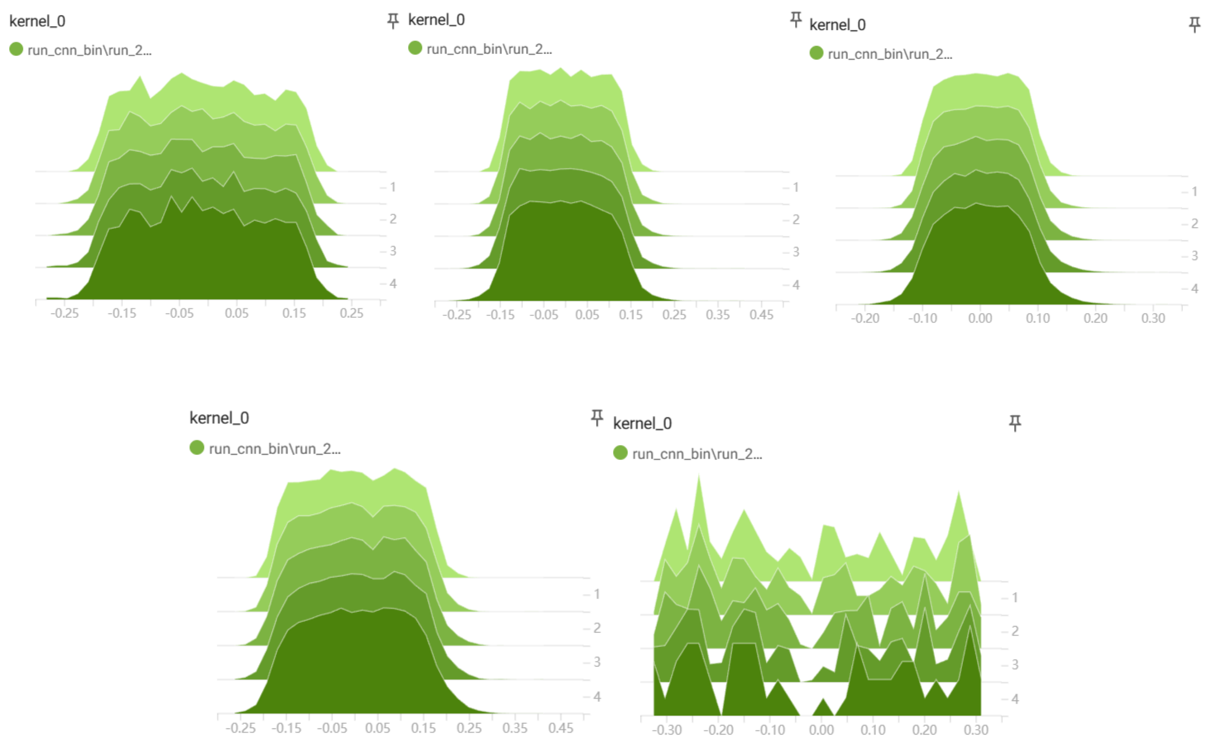


Figure12 The weight distribution of each layer in the CNN_bin,Seq_len = 25(Con1D_0,Con1D_1,Con1D_2,Dense1,Dense2)

By observing the time series diagram of weight distribution in TensorBoard, it can be found that the weight distribution is relatively scattered at the beginning. After passing through the convolutional layer, the weight distribution is slightly concentrated. After passing through the fully connected layer, the weight distribution shows regularity.

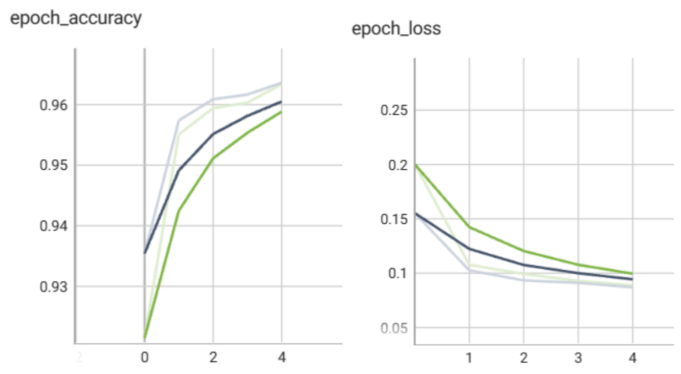
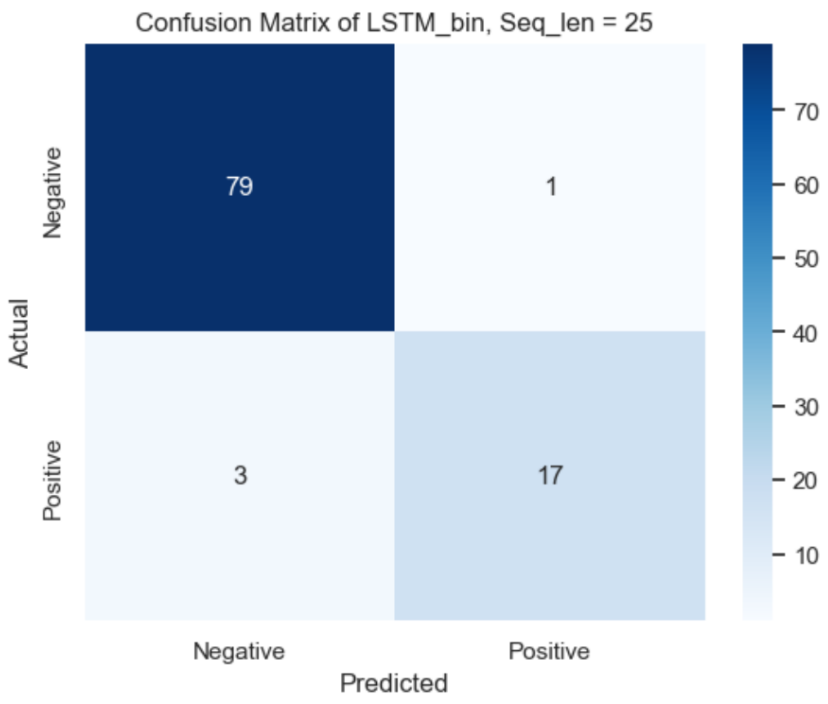


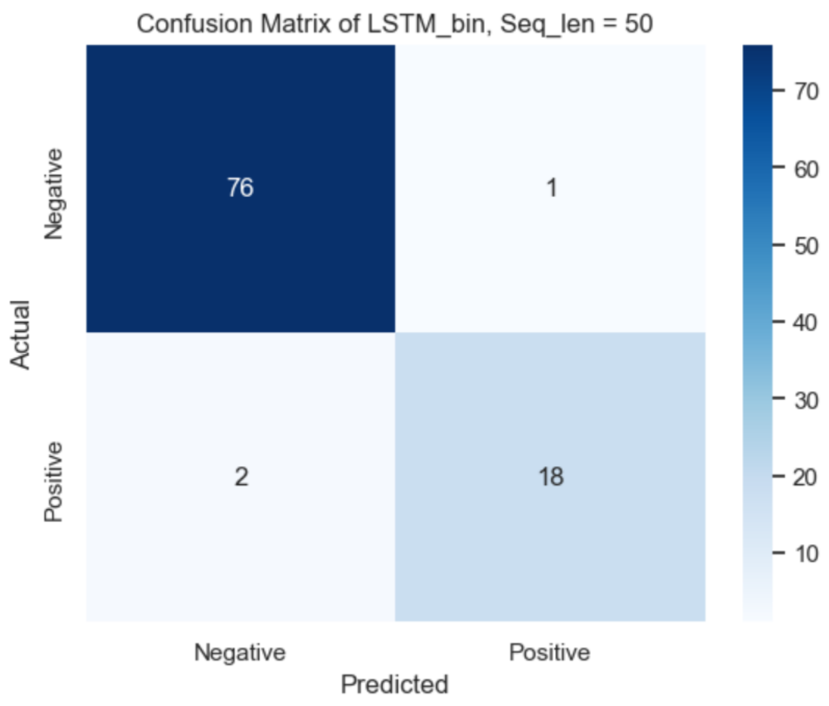
Figure13 Comparison of loss and accuracy of models before and after optimization

In the comparison of accuracy and loss function in each round (green represents before model optimization, black represents after model optimization), it can be found that after reducing the learning rate and increasing the dropout rate, the accuracy is higher, the loss is smaller, and the convergence is faster.

2.2 Failure Classification – LSTM (Student 2)

2.2.1 Baseline LSTM Model Performance





Sequence_length	**Accuracy**	**Precision**	**Recall**	**F1 Score**
25	**0.960**	**0.944**	**0.850**	**0.859**
50	**0.969**	**0.947**	**0.900**	**0.923**

The table shows the evaluation metrics when the sequence_length is 25 and 50. It can be observed that when the sequence_length doubles, all evaluation metrics show slight improvements, but the increase is not significant.

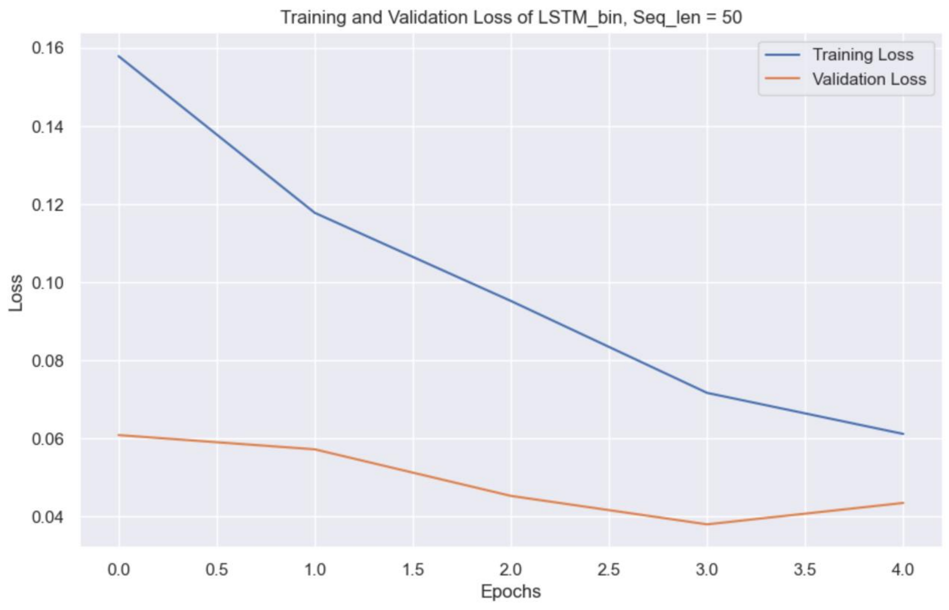
My hypothesis for this phenomenon:

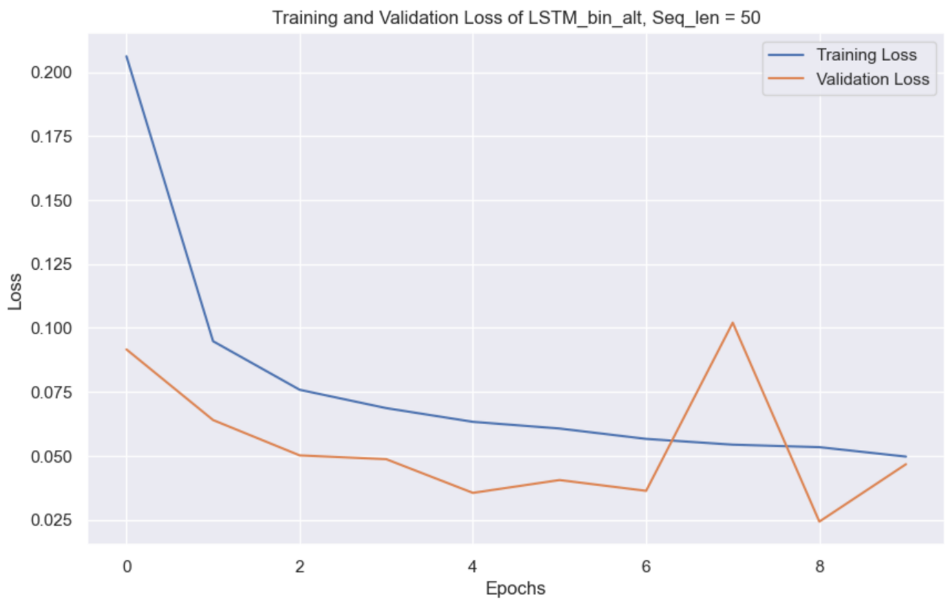
1. The model is already sufficiently complex, so it has achieved good performance with a sequence_length of 25, leaving little room for improvement.
2. The training task is simple. Since this is a binary classification problem, unlike tasks such as predicting RUL, the sequence_length might already be sufficient to capture the important features, resulting in only minor changes.

2.2.2 Hyperparameter Optimisation

The Hyperparameter of two model:

Model	Epoch	Drop Rate	Learning Rate	Batch Size
LSTM	5	0.1	0.001	16
LSTM_ALT	10	0.5	0.0005	64



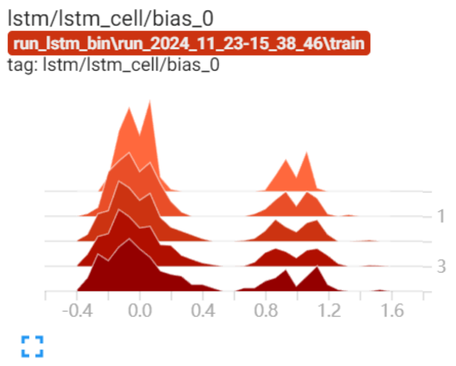


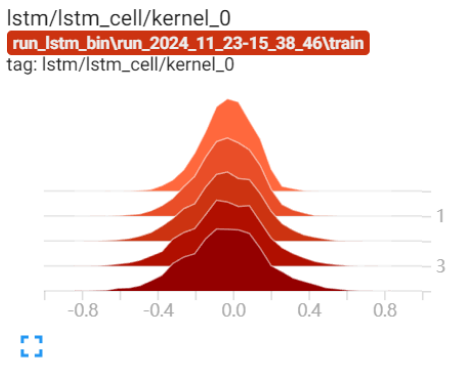
The model performance:

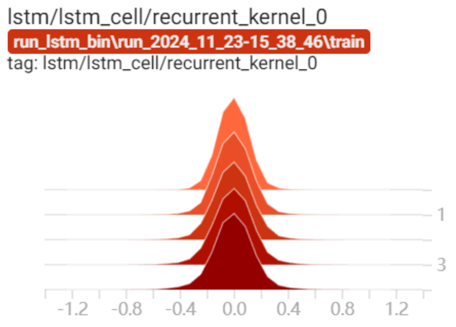
Model	Accuracy	Precision	Recall	F1 Score
LSTM	**0.969**	**0.947**	**0.900**	**0.923**
LSTM_ALT	**0.979**	**1.000**	**0.900**	**0.947**

After adjusting some hyperparameters, the evaluation metrics of the model have improved. Increasing the number of epochs allows the model to better learn patterns from the data. Raising the drop rate prevents the model from overfitting, which could otherwise affect performance. Lowering the learning rate enables the model to descend gradients more steadily, avoiding oscillations or missing the global optimum. Increasing the batch size improves the stability of gradient estimation for the model.

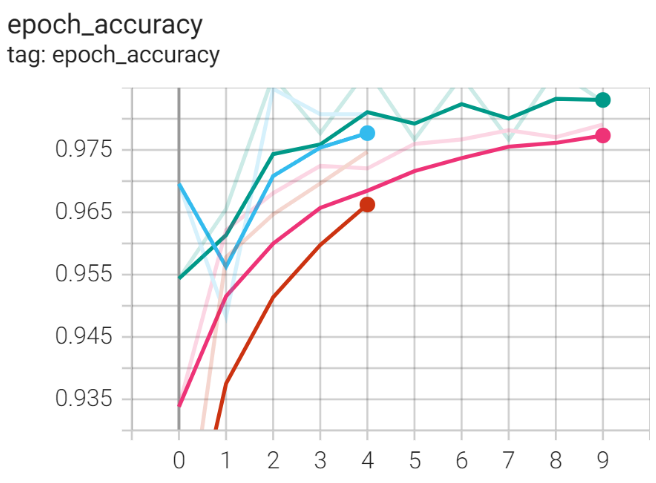
2.2.3 TensorBoard Analysis

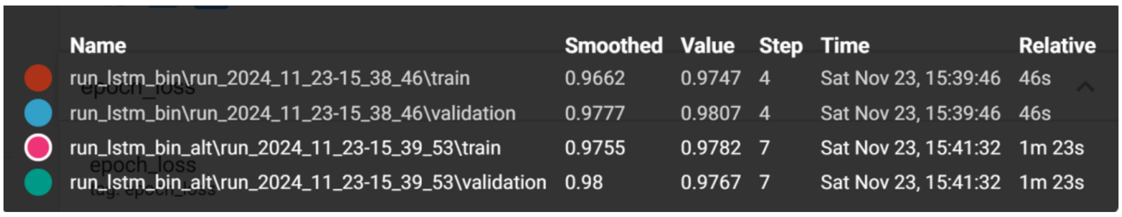






Throughout the training process, the weight distribution of kernel_0 remained stable, and the network showed normal behavior. The weights were initialized to small random values close to 0, so the distribution at the start of training may have been concentrated around 0. As training progressed, the weights gradually adjusted to fit the data, and the distribution may have slightly widened or shifted, but it still maintained a certain level of symmetry and stability.





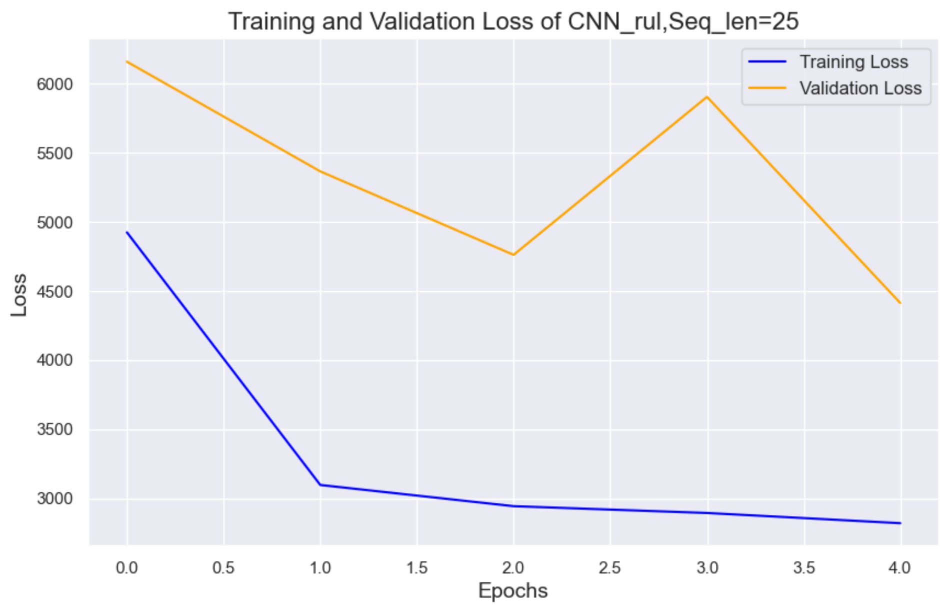
	Name	Smoothed	Value	Step	Time	Relative
●	run_lstm_bin\run_2024_11_23-15_38_46\train	0.9662	0.9747	4	Sat Nov 23, 15:39:46	46s
●	run_lstm_bin\run_2024_11_23-15_38_46\validation	0.9777	0.9807	4	Sat Nov 23, 15:39:46	46s
●	run_lstm_bin_alt\run_2024_11_23-15_39_53\train	0.9755	0.9782	7	Sat Nov 23, 15:41:32	1m 23s
●	run_lstm_bin_alt\run_2024_11_23-15_39_53\validation	0.98	0.9767	7	Sat Nov 23, 15:41:32	1m 23s

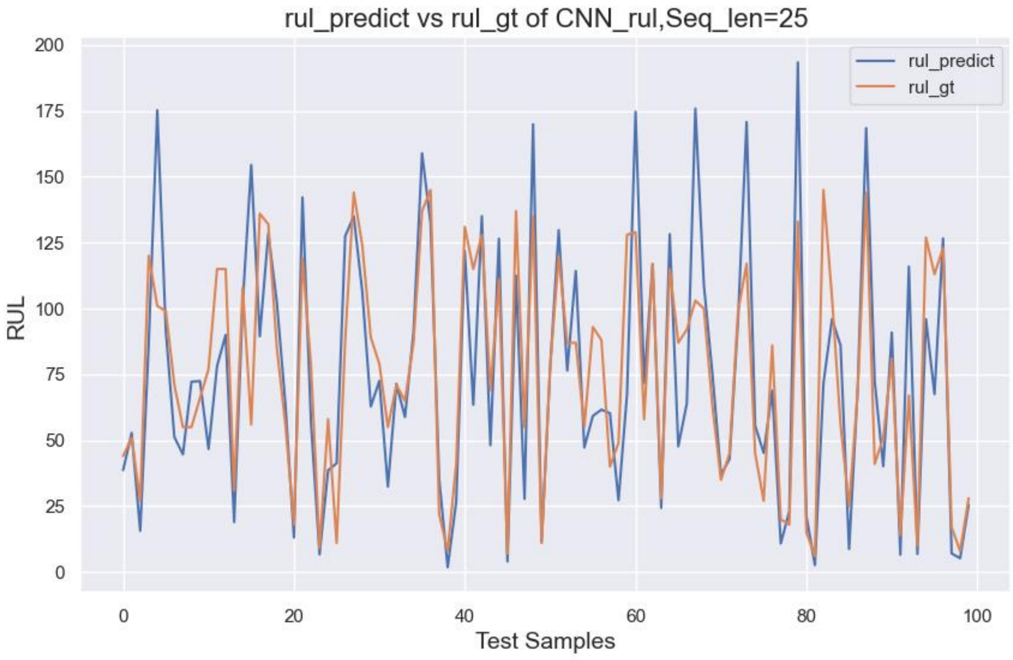
For both the training and validation sets, the accuracy of the LSTM_alt is higher than that of the original LSTM. Additionally, for the same model, the validation set have better performance than the training set. This is because dropout was applied during training but is not used during the validation phase.

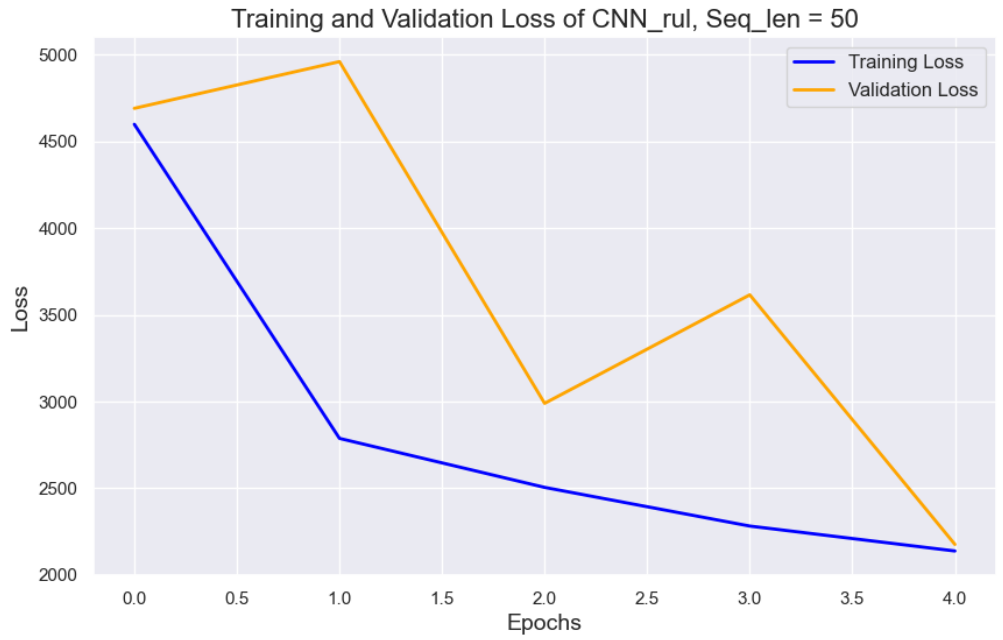
2.3 RUL Regression – CNN (Student 3)

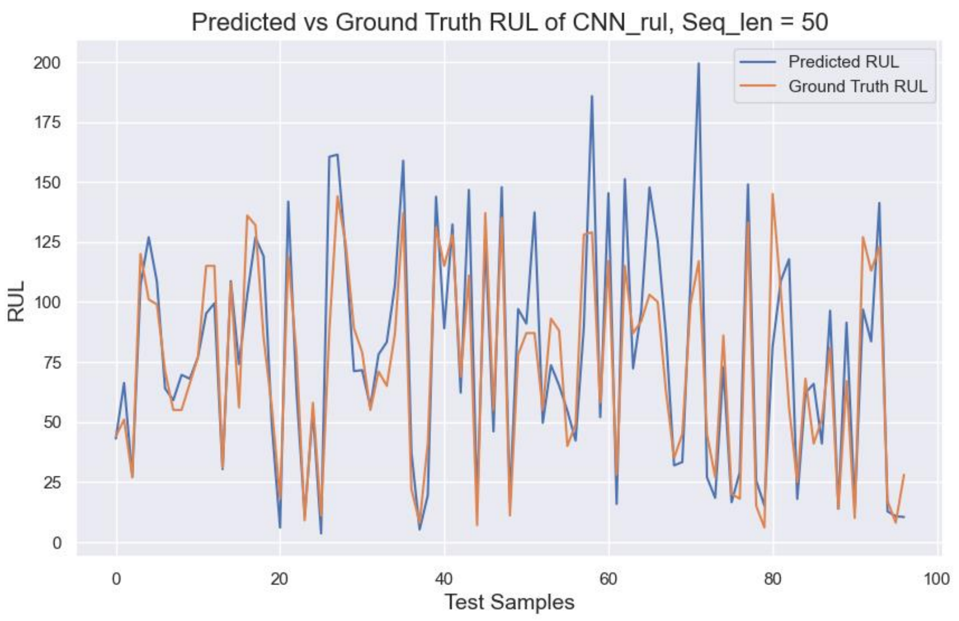
2.3.1 Baseline CNN Model Performance

Sequence_length	Training Data MAE	Test Data MAE
25	38.844	19.741
50	99.853	16.533


Training and Validation Loss of CNN_rul,Seq_len=25
Training Loss
Validation Loss
Loss
Epochs
6000
5500
5000
4500
4000
3500
3000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


rul_predict vs rul_gt of CNN_rul,Seq_len=25
rul_predict
rul_gt
RUL
Test Samples
200
175
150
125
100
75
50
25
0
0
20
40
60
80
100


Training and Validation Loss of CNN_rul, Seq_len = 50
Training Loss
Validation Loss
Loss
Epochs
5000
4500
4000
3500
3000
2500
2000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0



Training and validation loss: With Seq_len = 50, the model has lower losses on both the training and validation sets, indicating that the model may have learned the data better.

Prediction vs. actual RUL: With Seq_len = 50, the predicted values fit more closely to the actual values.

MAE: Although Seq_len = 50 has a higher MAE on the training set, it performs better on the test set, showing better generalization ability.

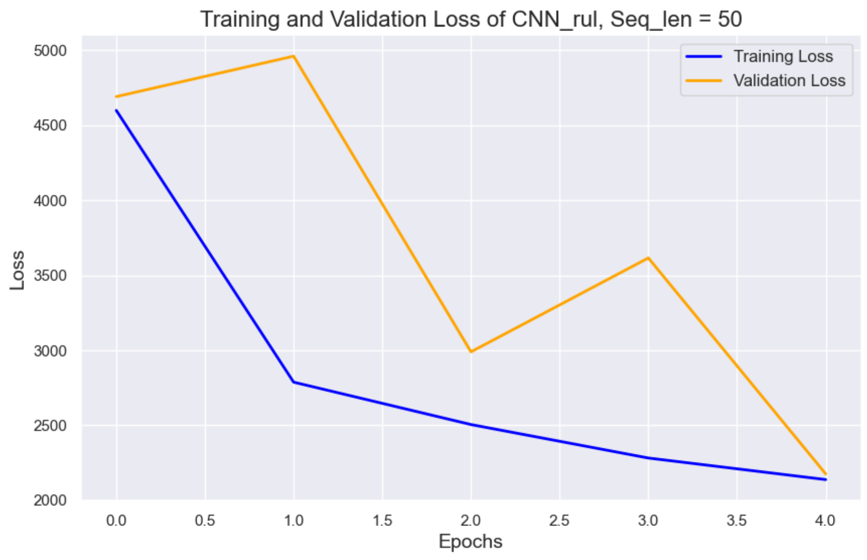
2.3.2 Hyperparameter Optimisation

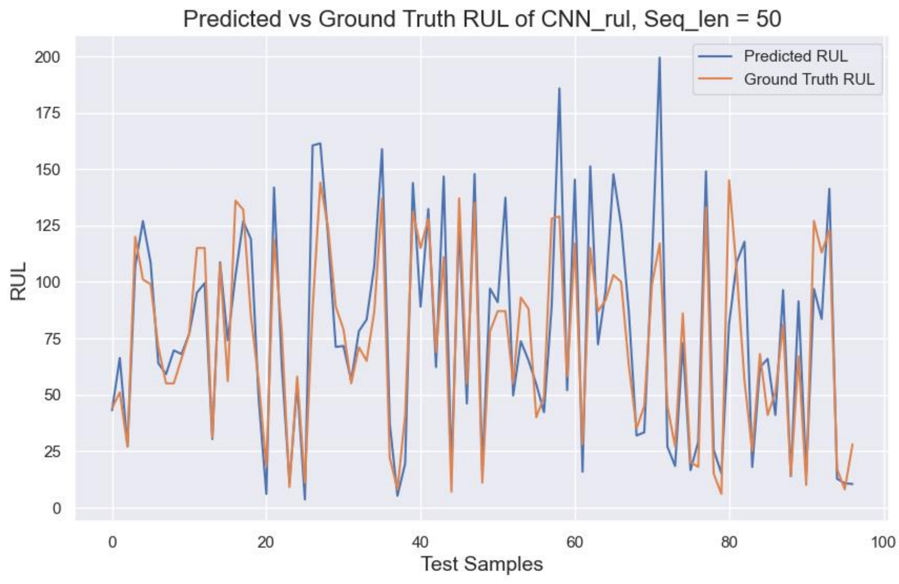
The Hyperparameter of two model:

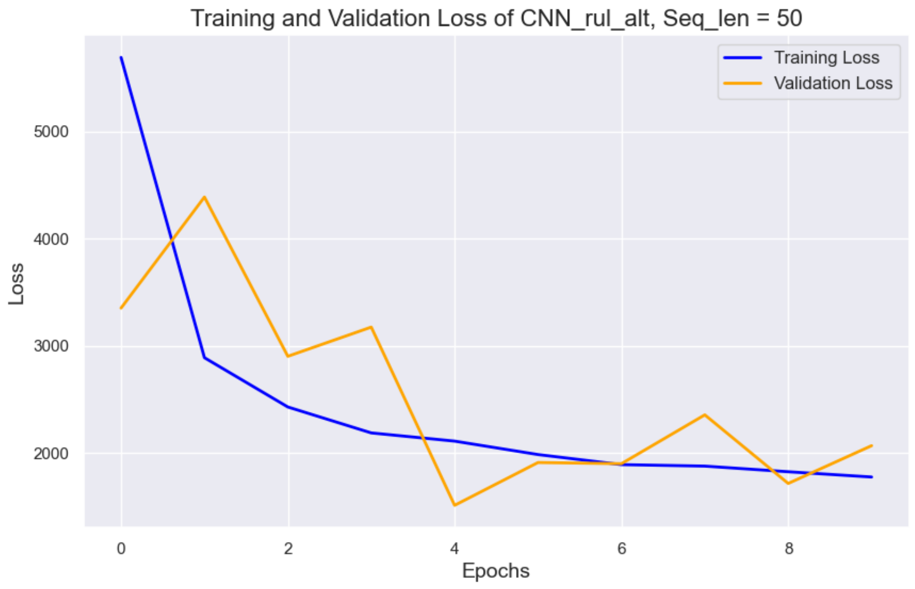
Model	Epoch	Drop Rate	Learning Rate	Batch Size
CNN	5	0.3	0.001	32
CNN_ALT	10	0.2	0.0005	64

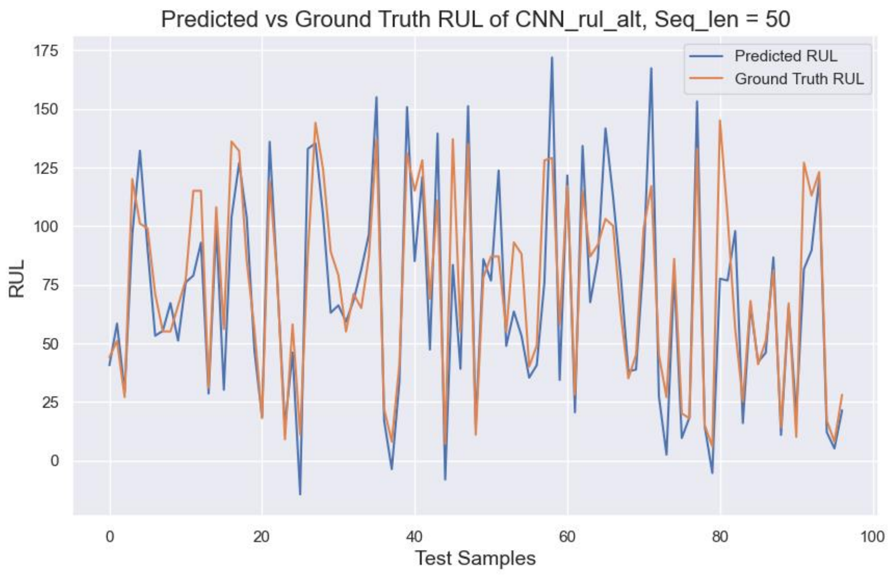
The model performance:

Model	Training Data MAE	Test Data MAE
CNN	99.853	16.533
CNN_ALT	29.668	15.943


Training and Validation Loss of CNN_rul, Seq_len = 50
Training Loss
Validation Loss
Loss
Epochs
5000
4500
4000
3500
3000
2500
2000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Predicted vs Ground Truth RUL of CNN_rul, Seq_len = 50
Predicted RUL
Ground Truth RUL
RUL
Test Samples
200
175
150
125
100
75
50
25
0
0
20
40
60
80
100

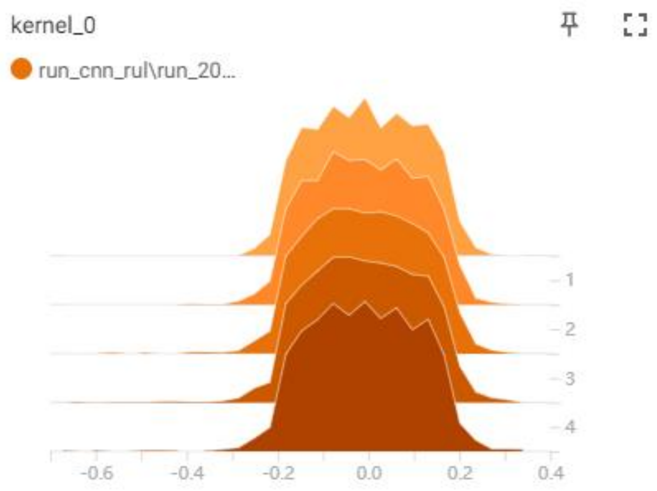

Training and Validation Loss of CNN_rul_alt, Seq_len = 50
Training Loss
Validation Loss
Loss
Epochs
5000
4000
3000
2000
0
2
4
6
8

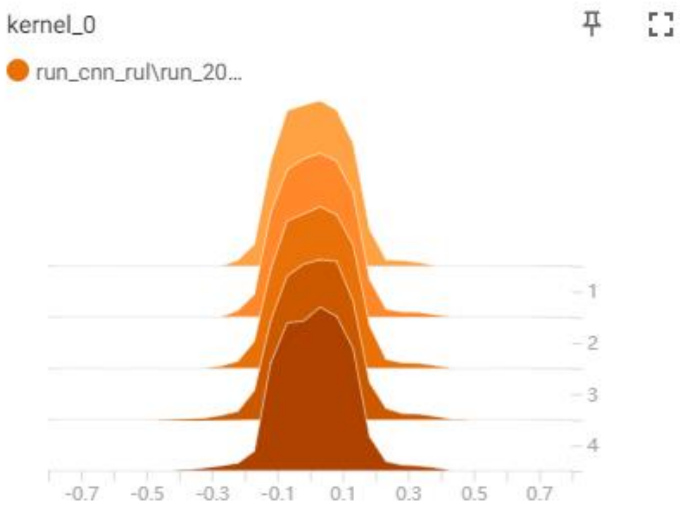


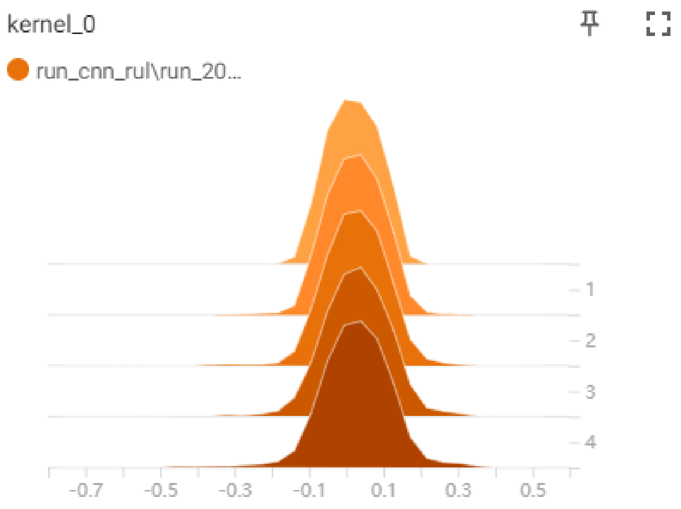
The original model was trained for 5 epochs, while the adjusted model was trained for 10 epochs. More training cycles may help the model to better learn the characteristics of the data, although this could also increase the risk of overfitting. However, judging from the validation loss, the adjusted model does not show obvious signs of overfitting. The dropout rate of the original model is 0.3, while the adjusted model has a dropout rate of 0.2. A lower dropout rate may help the model retain more information, thereby improving performance. The batch size of the original model is 32, while the adjusted model uses a batch size of 64. A larger batch size can provide a more stable gradient estimation, which contributes to the stability of the model's training.

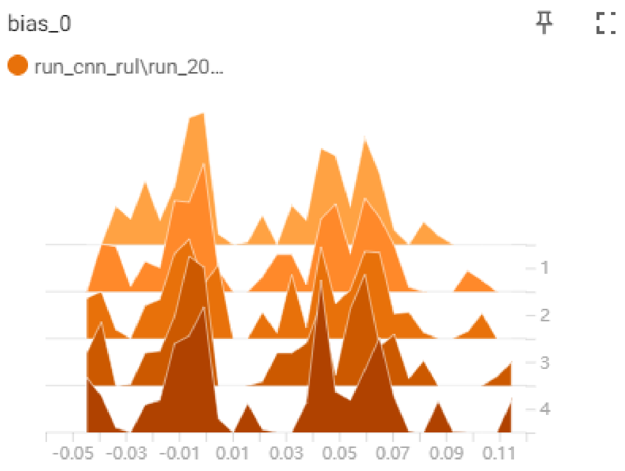
The adjusted model has a significantly lower MAE on the training data compared to the original model, indicating that the model fits the training set better. The MAE on the test data is close for both models, but the adjusted model is slightly lower, which may suggest that the adjusted model has an improved generalization capability.

2.3.3 TensorBoard Analysis

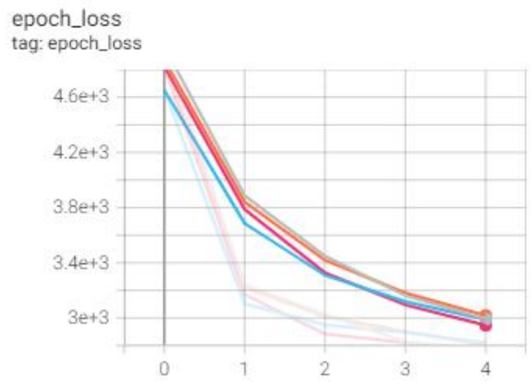


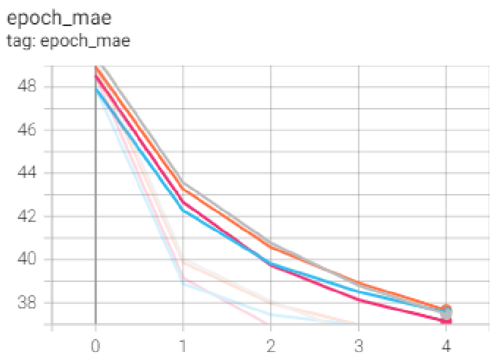






The presence of multiple peaks in the distribution of bias values may indicate that the bias parameters undergo various update cycles during the training process, or they serve different neurons that are activated at different times. The initial distribution of weights is likely concentrated near zero, which helps prevent issues with vanishing or exploding gradients. As training progresses, the distribution of weights may change, reflecting the learning process of the model. If the distribution becomes more concentrated, this could suggest that the model is converging.



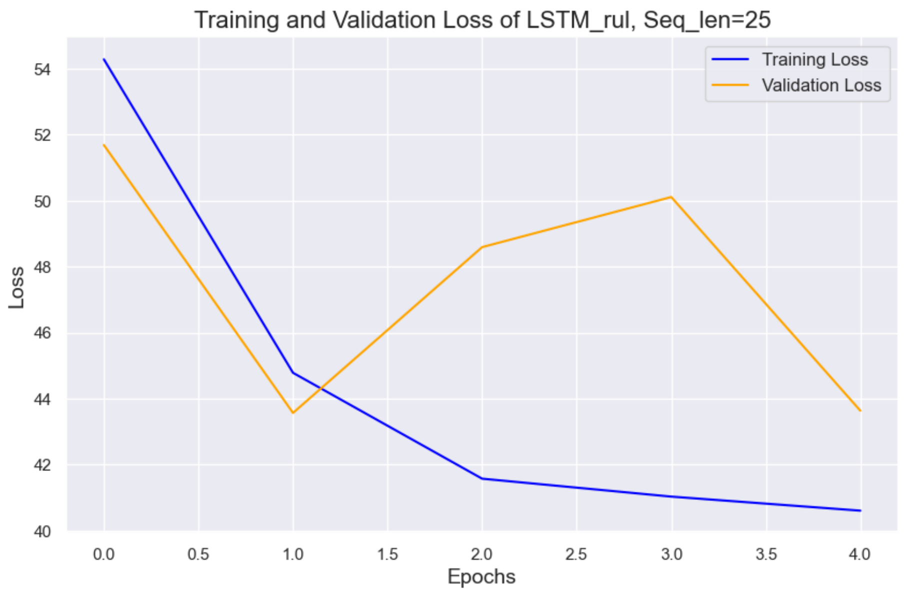


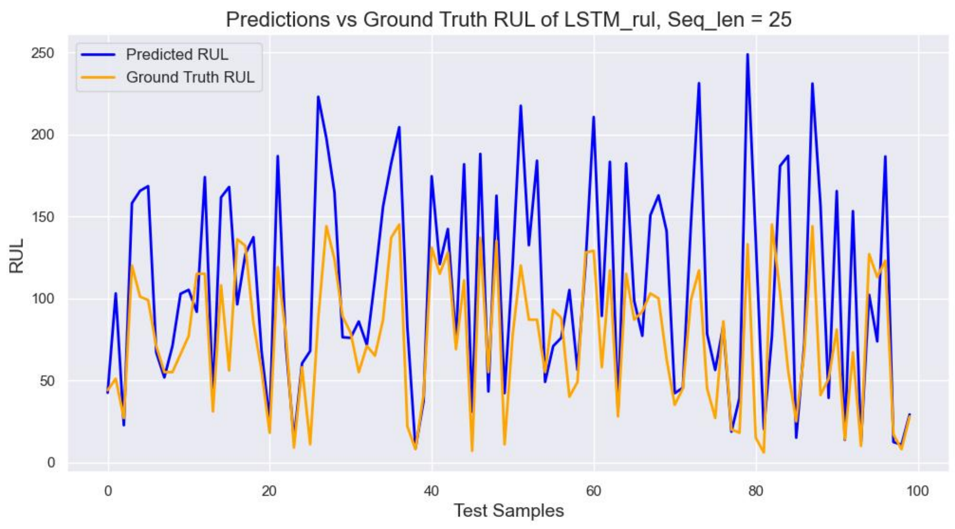
The MAE and loss graphs show a trend of decreasing loss as the number of epochs increases, indicating that the model is gradually optimizing during the learning process. The rate of loss decrease is faster in the initial epochs and then gradually slows down, suggesting that the model learns more rapidly at the beginning and then refines its parameters progressively.

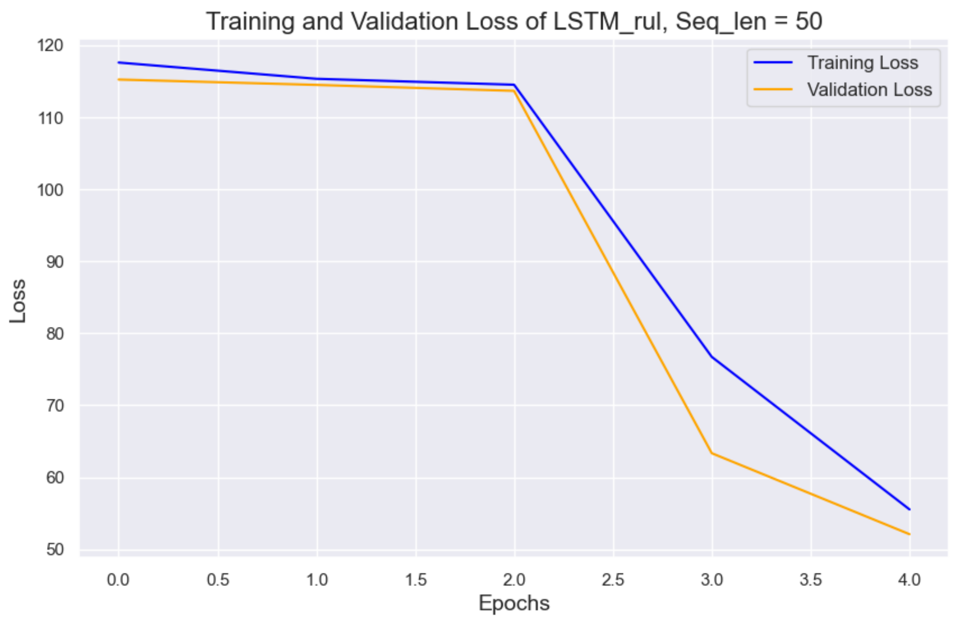
2.4 RUL Regression – LSTM (Student 4)

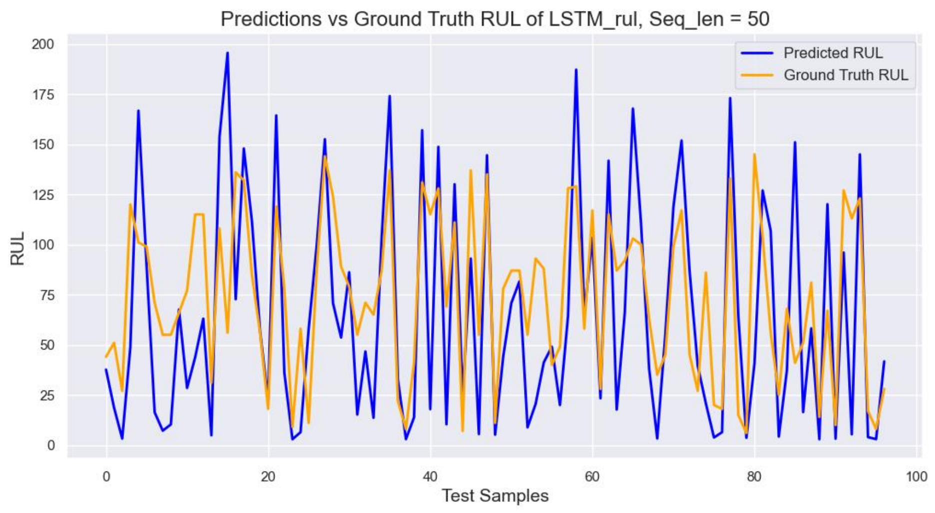
2.4.1 Baseline LSTM Model Performance

Sequence_length	Training Data MAE	Test Data MAE
25	42.935	38.619
50	50.451	34.483









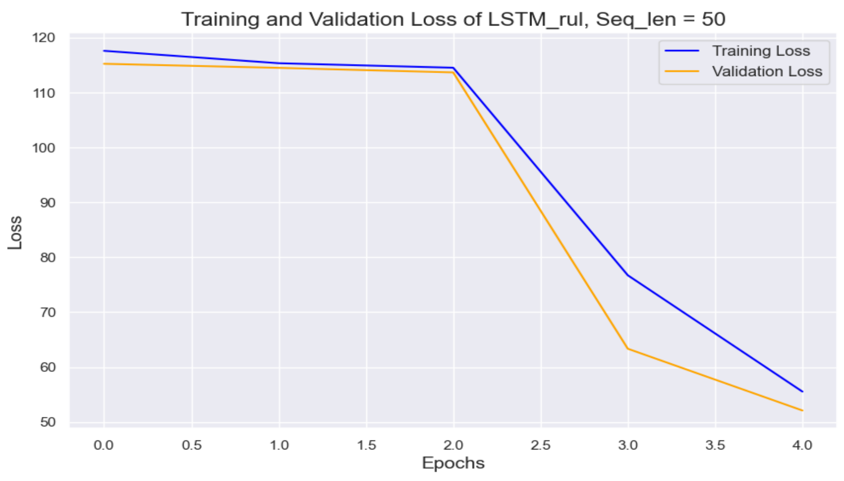
The model achieves a training MAE of 42.93 when Sequence Length = 25, which indicates the model's performance when making predictions on the training data. A relatively moderate error suggests that the model can fit the training data reasonably well. While the model has a training MAE of 50.45 for sequence length 50, higher than the MAE for sequence length 25, which suggests that increasing the sequence length may slightly makes the model more complex and harder to fit the training data as accurately as before.

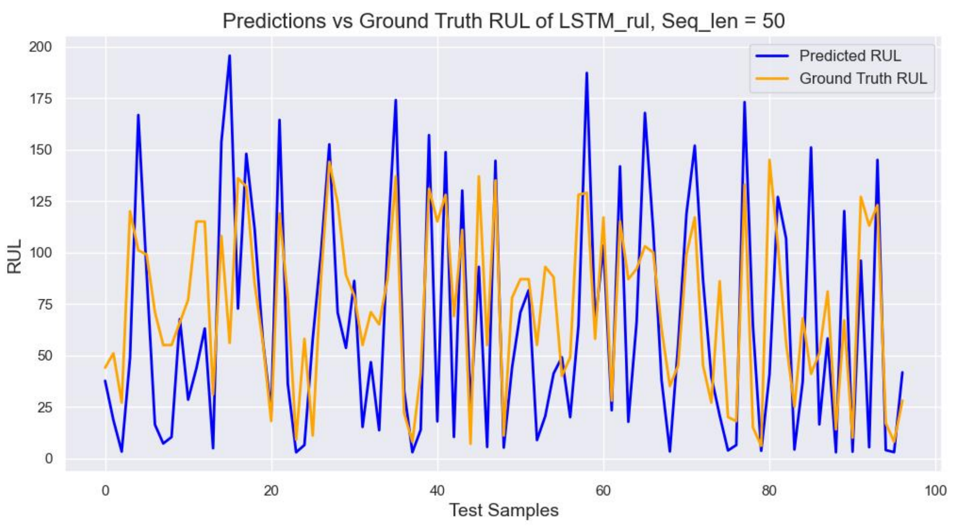
For sequence length 25, the model produces a test MAE of 38.62, which is higher than the training error, implying that the model may be overfitting the training data. However, The test MAE is 34.48 for sequence length 50, which is lower than the test MAE for sequence length 25, which indicates that increasing the sequence length may improve the model's ability to generalize, resulting in better performance on the test data.

The model performs worse on the training data with sequence length 50, likely because the model has to process more time steps, which increases the complexity and makes it harder to fit the data perfectly. While the model performs better on the test data with sequence length 50, as evidenced by the lower test MAE, which suggests that while a longer sequence may complicate training. It improves the model's ability to capture long-term dependencies, leading to better generalization and performance on unseen data.

For sequence length 50, the model exhibits slightly higher training error but better test error, suggesting an improvement in generalization despite the increased complexity. For sequence length 25, the model shows better training performance but poorer generalization to the test data. Longer sequences might help the model capture more information from the past, leading to better generalization, especially when working with time series data that requires understanding of long-term dependencies.

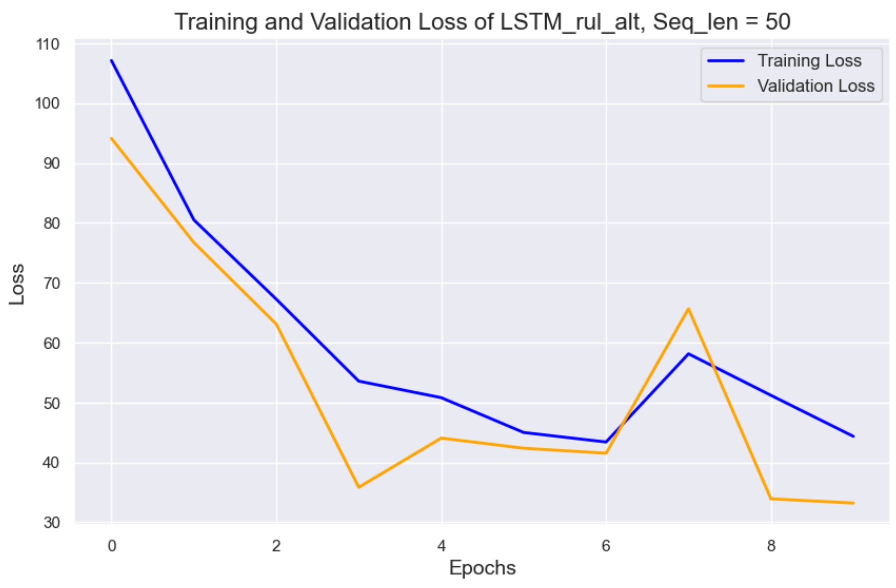
2.4.2 Hyperparameter Optimisation

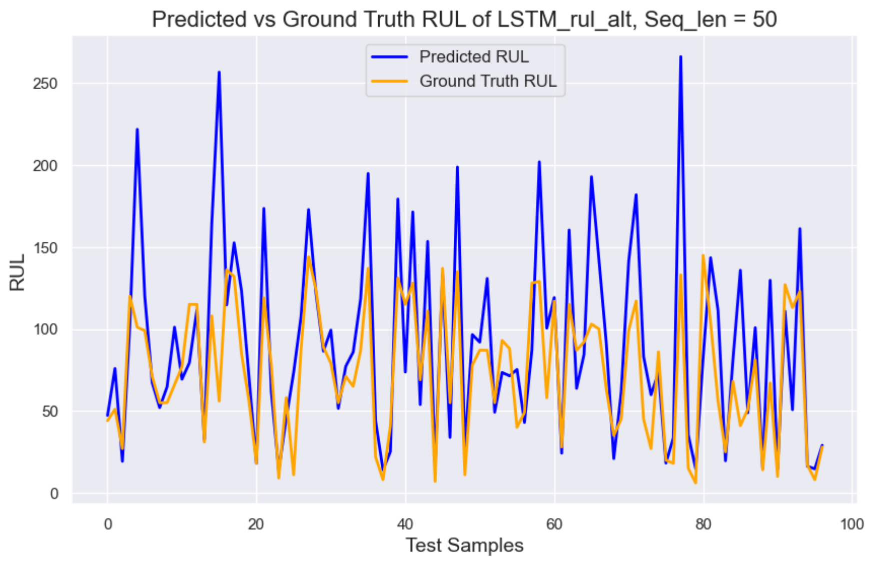




The Hyperparameter of two model:

Model	Epoch	Drop Rate	Learning Rate	Batch Size
LSTM	5	0.3	0.001	32
LSTM_ALT	10	0.3	0.001	64





Model	Training Data MAE	Test Data MAE
LSTM	**50.451**	**34.483**
LSTM_ALT	**37.052**	**28.638**

The impact of hyperparameter optimization on the LSTM models with a sequence length of 50 is evident in their training and testing performance. The higher number of epochs allowed the model to converge better, and the larger batch size stabilized the training process. These changes contributed to better generalization and lower error rates in both the training and test datasets.

Epochs:

The LSTM model was trained for 5 epochs, while the LSTM_ALT model was trained for 10 epochs. Increasing the number of epochs allows the model to learn more complex patterns, especially for longer sequences, enhancing its generalization capabilities. The LSTM_ALT model's extended training duration enabled it to minimize the loss further, resulting in lower MAE values for both the training (37.05 vs. 50.45) and testing (28.64 vs. 34.48) datasets. This demonstrates that training for more epochs can lead to better model performance, especially in more complex tasks.

Dropout Rate:

Both models used the same dropout rate of 0.3, which ensures consistent regularization. The choice of 0.3 as the dropout rate helps strike a balance between reducing overfitting and maintaining sufficient information during training. This contributes to robust model performance and is effective in both models, particularly when dealing with sequence data that requires capturing temporal dependencies.

Learning Rate:

Both models employed the same learning rate of 0.001, which is a commonly used value for LSTM models. A learning rate of 0.001 allows for stable convergence during training, preventing overshooting of the optimal solution.

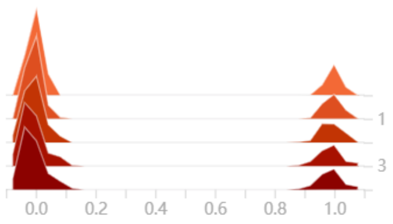
Batch Size:

The LSTM model used a batch size of 32, while the LSTM_ALT model used a batch size of 64. A larger batch size in LSTM_ALT resulted in smoother gradient updates and faster convergence during training. This improvement is evident in the reduced MAE values, which means the larger batch sizes can also help stabilize training for longer sequence lengths.

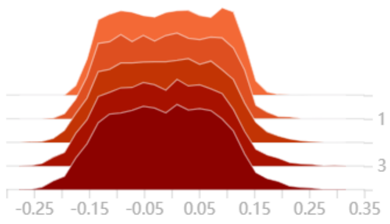
2.4.3 TensorBoard Analysis

lstm_15

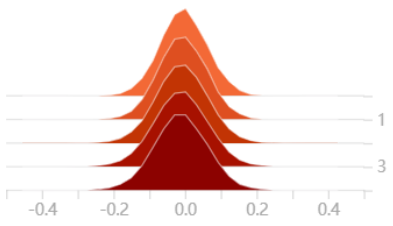
lstm_15/lstm_cell_15/bias_0
run_lstm_rul\run_2024_11_23-16_25_47\train
tag: lstm_15/lstm_cell_15/bias_0



lstm_15/lstm_cell_15/kernel_0
run_lstm_rul\run_2024_11_23-16_25_47\train
tag: lstm_15/lstm_cell_15/kernel_0

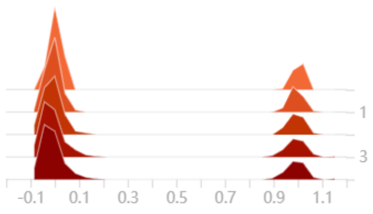


lstm_15/lstm_cell_15/recurrent_kernel_0
run_lstm_rul\run_2024_11_23-16_25_47\train
tag: lstm_15/lstm_cell_15/recurrent_kernel_0

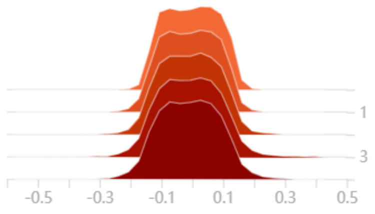


lstm_16

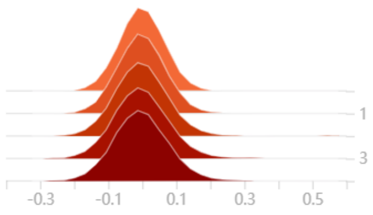
lstm_16/lstm_cell_16/bias_0
run_lstm_rul\run_2024_11_23-16_25_47\train
tag: lstm_16/lstm_cell_16/bias_0

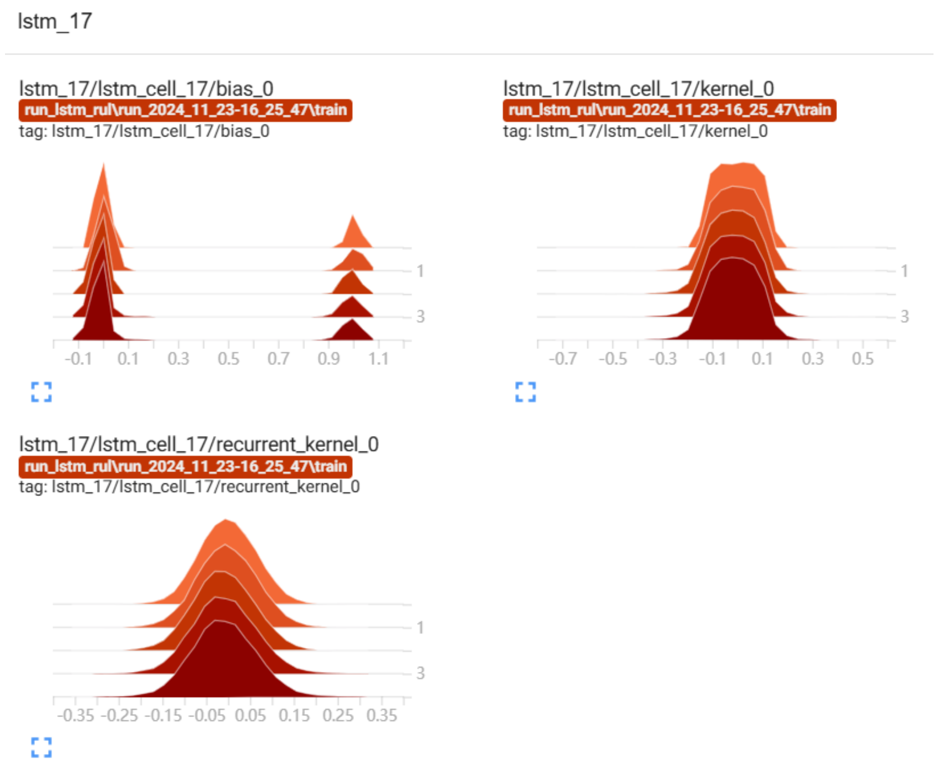


lstm_16/lstm_cell_16/kernel_0
run_lstm_rul\run_2024_11_23-16_25_47\train
tag: lstm_16/lstm_cell_16/kernel_0



lstm_16/lstm_cell_16/recurrent_kernel_0
run_lstm_rul\run_2024_11_23-16_25_47\train
tag: lstm_16/lstm_cell_16/recurrent_kernel_0

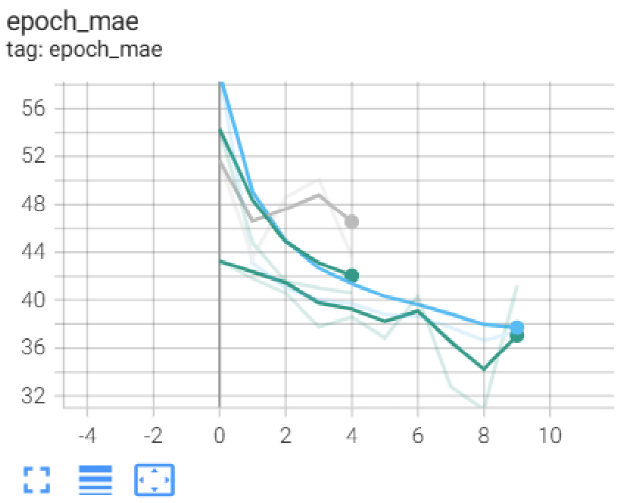


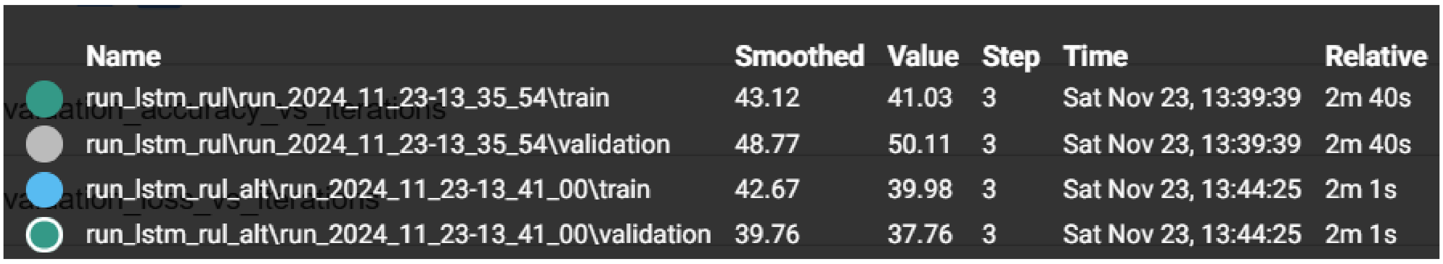


The weight distributions across the LSTM layers exhibit a pattern of initial concentration around zero, expanding in range as the network deepens, and then consolidating towards the final layer. This suggests that the model is initially stabilizing, then learning to capture more complex features, and finally refining its learning to focus on the most relevant patterns.

The magnitude of weights in the LSTM layers shows a progression from moderate initial values to larger ranges in deeper layers, particularly in the kernel weights. This indicates that the model's ability to capture complex features increases with depth, with the final layer showing a refined focus on specific features, as evidenced by the narrower range of the recurrent kernel weights.

The behavior of the LSTM layers reflects a learning process that starts with stabilization, progresses through a phase of increased complexity and feature learning, and culminates in a consolidation phase where the model fine-tunes its understanding of the most significant dependencies within the data. This sequential behavior is characteristic of deep learning models as they build upon successive layers to abstract and process information more effectively.





	Name	Smoothed	Value	Step	Time	Relative
●	run_lstm_rul\run_2024_11_23-13_35_54\train	43.12	41.03	3	Sat Nov 23, 13:39:39	2m 40s
●	run_lstm_rul\run_2024_11_23-13_35_54\validation	48.77	50.11	3	Sat Nov 23, 13:39:39	2m 40s
●	run_lstm_rul_alt\run_2024_11_23-13_41_00\train	42.67	39.98	3	Sat Nov 23, 13:44:25	2m 1s
○	run_lstm_rul_alt\run_2024_11_23-13_41_00\validation	39.76	37.76	3	Sat Nov 23, 13:44:25	2m 1s

The epoch_mae image depicting the MAE over epochs, illustrates a rapid decrease in MAE during the initial training phase, indicating swift learning and improvement in predictive accuracy. As training progresses, the rate of improvement slows, suggesting the model is approaching its performance limits. The behavior of the model, as depicted by the decreasing MAE over epochs and the comparative MAE values in the table, suggests that the model is effectively learning from the training data, with initial rapid improvements that gradually stabilize. The close MAE values between training and validation sets across different runs indicate good generalization, while the efficiency of training is underscored by the timestamps and step counts. Overall, it reflects a model that is progressively fine-tuning its parameters to enhance predictive accuracy while maintaining a balance to avoid overfitting.

---

3. Model Comparison

3.1 Joint Student 1 and Student 3

3.1.1 Trainable Parameters Verification

The number of Parameters:

Layer	Para
CNN,32	2432
Dropout	0
CNN,64	6208
Dropout	0
CNN,128	24704

Dropout	**0**
Dense	**1153**

In the given one-dimensional convolutional neural network (CNN) architecture designed for binary classification, the parameter count for each layer is calculated as follows:

1. Convolutional Layer
   a. **First Conv1D Layer:**
      i. Filters: 32
      ii. Kernel Size: 3
      iii. Input Channels: 25

      Parameters: 32×3×25+32=243232×3×25+32=2432
   b. **Second Conv1D Layer:**

      Parameters: 64×3×32+64=620864×3×32+64=6208
   c. **Third Conv1D Layer:**

      Parameters: 128×3×64+128=24704128×3×64+128=24704
2. **MaxPooling1D Layer:**
   a. No parameters.
3. **Flatten Layer:**
   a. No parameters.
4. **Dense Layer:**
   a. Output Features: 1
   b. Input Features: 1152 (output after flattening)
   c. Parameters: 1152+1=11531152+1=1153

The sum of parameters across all layers is

2432+6208+24704+0+0+1153=344972432+6208+24704+0+0+1153=34497.

3.1.2 CNN Architecture Design

The convolutional neural network (CNN) for binary classification employs a three-layer convolutional structure with max pooling and dropout for feature extraction and regularization. The first convolutional layer utilizes 32 filters of size 3x1, followed by a max pooling layer with a pool size of 2 and a dropout layer with a 0.3 dropout rate to prevent overfitting. The second layer consists of 64 filters of size 3x1, again followed by a max pooling layer and a dropout layer with the same configuration. The third layer features 128 filters of size 3x1, succeeded by another max pooling and dropout layer. Post-convolutional layers, the feature maps are flattened, and the model includes a dense layer with a single sigmoid-activated neuron for binary output, indicating the probability of the input belonging to one class. The model is optimized using the Adam optimizer with a learning rate of 0.001 and is trained with binary cross-entropy loss, focusing on accuracy as the performance metric.

3.2 Joint Student 2 and Student 4

3.2.1 Trainable Parameters Verification

The number of Parameters:

Layer	**Para**
LSTM,64	**23040**
Dropout	**0**
LSTM,64	**33024**
Dropout	**0**
LSTM,64	**33024**
Dropout	**0**
Dense	**65**

1. **LSTM:**

   Para = 4 × ((input + units) × units) + 4 × units (bias)

   so Para (Layer 1) = 4 × ((25 + 64) × 64) + 4 × 64 = 23040

   Para (Layer 2&3) = 4 × ((64 + 64) × 64) + 4 × 64 = 33024

2. **Dropout**

   The Dropout layer has no trainable parameters because it simply randomly drops a certain proportion of neurons, so the number of parameters in these layers is 0.

3. **Dense**

   Para = input × output + output

   so Para = 64 × 1 + 1 = 65

3.2.2 LSTM Architecture Design

The baseline LSTM network architecture is designed with a standard sequential structure, using multiple LSTM layers to capture temporal dependencies in the data. It includes dropout layers for regularization and a final dense layer for output.

The key parameter choices in the baseline model are:

LSTM: A fixed number of units (64) in each LSTM layer to balance model complexity and computational efficiency.

Dropout: A dropout rate (0.2) is applied to prevent overfitting.

Learning rate: A moderate learning rate (0.01) for stable convergence during training.

Batch size: A standard batch size (16) to ensure stable gradient updates.

Epochs: A number of epochs (5) to allow the model to learn effectively without overfitting.

The key parameter choices in the baseline model include:

Learning rate: Reducing the learning rate (0.0005) for more precise weight updates.

Batch size: Increasing the batch size (64) for more stable gradient estimation.

Dropout rate: Increasing dropout (0.5) to mitigate overfitting.

3.3 CNN vs LSTM for failure classification (Joint Student 1 and Student 2)

3.3.1 Performance Comparison

Model	Accuracy	Precision	Recall	F1 Score
CNN	**0.940**	**0.850**	**0.85**	**0.850**
LSTM	**0.940**	**0.889**	**0.80**	**0.842**

3.3.2 Strengths and Weaknesses

From the table, it can be seen that the accuracy and F1-score of LSTM and CNN are almost identical.

However, CNN has a slightly higher recall, while LSTM achieves a higher precision. We speculate that LSTM, being adept at handling sequential data and capturing long-term dependencies, tends to be more cautious when making predictions. It only classifies instances as positive when it is highly confident, thereby reducing false positives. So, LSTM is better suited for tasks with low tolerance for false positives, such as cancer screening.

On the other hand, CNN focuses more on extracting local features and has limited ability to capture global or long-term dependencies. This characteristic makes CNN more inclined to classify instances as positive, even in uncertain situations, which decreases precision but increases recall. In that way, CNN is more suitable for tasks with low tolerance for false negatives, such as disaster monitoring.

In addition, during the actual training process, CNN trains much faster than LSTM, which indicates that CNN has higher computational efficiency and is more suitable for real-time tasks.

3.4 CNN vs LSTM for RUL Regression (Joint Student 3 and Student 4)

3.4.1 Performance Comparison

Model	Training Data MAE	Test Data MAE
CNN	99.853	16.533
CNN_ALT	29.668	15.943
LSTM	50.451	34.483
LSTM_ALT	37.052	28.638

CNN demonstrates superior generalization ability with significantly lower test MAE compared to LSTM. The improved CNN_ALT achieves the lowest test MAE (15.94), indicating its robustness and accuracy in predicting unseen data. While CNN struggles with training data (high training MAE), it performs exceptionally well on the test set.LSTM Performance: LSTM has better training performance (lower training MAE, e.g., 50.45 compared to CNN's 99.85) due to its ability to model long-term temporal dependencies. However, its test MAE (e.g., 34.48) is much higher, suggesting overfitting and weaker generalization.

3.4.2 Strengths and Weaknesses

CNN Strengths:

CNN excels at extracting localized patterns and features from raw sensor data, such as sudden changes or anomalies, which are critical for RUL prediction. Despite the poor performance during training, CNNs show good generalization on the test data, indicating that they effectively identify localized patterns in the sensor data and can make reliable predictions for unseen data. It also trains faster due to its parallel computing capabilities.

CNN Weaknesses:

The standard CNN model struggles to capture the overall patterns in the training data, which is reflected in its high training MAE. CNNs may not effectively capture long-term dependencies or patterns across time, which are crucial for RUL predictions.

LSTM Strengths:

LSTM is designed to capture long-term dependencies in sequential data, making it ideal for time-series modeling, which is beneficial for time-series tasks like RUL prediction. However, in RUL tasks, where localized patterns and short-term trends often dominate, LSTM's sequential modeling advantage may not fully align with the problem requirements. Despite their training success (moderate MAE), they fail to generalize well, showing that temporal dependencies alone may not be sufficient for accurate predictions.

LSTM Weaknesses:

LSTM models are computationally expensive, prone to overfitting, and often require careful tuning to generalize well. With fixed-length sequences, their performance can lag when the task relies more on local feature extraction. The LSTM shows a significant drop in performance on the test data, which suggests overfitting. LSTMs can memorize training sequences well, but they struggle with generalization to unseen test data, which is a crucial issue for RUL regression where historical data might not fully represent future conditions. Despite performing better than CNN in terms of learning from the data, the LSTM still exhibits higher test error, showing that capturing temporal patterns alone is not enough for robust RUL predictions. What's more, during the actual training process, CNN trains much faster than LSTM, which indicates that CNN has higher computational efficiency and is more suitable for real-time tasks.

---

4. Execution Time and No. of Parameters (All Students)

4.1 Comparison for sequence_length = 50

Sequence_length = 50

Epoch = 5

Batch size = 100

Model	Number of trainable parameters	execution times(seconds)	Performance
CNN_bin	66241	5.93	0.940(Accurcay)
LSTM_bin	89153	66.77	0.940(Accurcay)
CNN_RUL	66241	6.85	16.533(MAE)
LSTM_RUL	89153	54.73	34.483(MAE)

4.2 Justification

1. **CNN_bin vs LSTM_bin**

   Both models have an accuracy of 0.940, performing equally well.

   The number of trainable parameters for LSTM_bin (89,153) is approximately 34.6% higher than that of CNN_bin (66,241), but the execution time of LSTM_bin (66.77 seconds) is over 10 times longer than that of CNN_bin (5.93 seconds).

   For binary classification tasks, the advantage of LSTM typically is handling sequences and capturing long-term dependencies. However, in this case, the input sequence length is only 25 or 50, which does not fully leverage the strengths of LSTM. Compared to CNN, it offers no performance improvement while requiring more computational resources. Therefore, under these circumstances, CNN performs better efficiently.

2. **CNN_RUL vs LSTM_RUL**

   The MAE of CNN_RUL (16.533) is significantly better than that of LSTM_RUL (34.483), indicating superior performance.

   Although LSTM_RUL has a higher number of trainable parameters and execution time compared to CNN_RUL (34.6% more parameters and approximately 8 times longer execution time), its performance is noticeably inferior to CNN_RUL.

   For tasks such as predicting RUL, CNN may perform better by extracting local features. The computational resources in LSTM did not translate into performance advantages in this case,making the performance and cost of LSTM_RUL less justified.

5. Conclusion

5.1 Summary of Findings

From this report, we have the following findings:

1. In binary classification task, the training results of CNN and LSTM models are similar, but the training time of LSTM is significantly higher than that of CNN.
2. In RUL regression task, the training effect of CNN is slightly better than that of LSTM.Moreover, CNN significantly outperforms LSTM in terms of training time, number of parameters, and other metrics.
3. The training effect of a sequence length of 50 is significantly better than that of a sequence length of 25.