Air Quality Forecasting using LSTM: Report

1. Introduction

The challenge at hand involves predicting PM2.5 concentrations in Beijing, utilizing historical air quality and weather data. This task is approached through time series forecasting using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, which are well-suited for sequential data. The primary goal is to design and fine-tune an LSTM model to minimize the Root Mean Squared Error (RMSE), ensuring the model accurately predicts future air quality levels.

This report details the exploration, preprocessing, model design, and experimentation, highlighting the steps taken to improve model performance.

---

2. Data Exploration and Preprocessing

2.1 Data Exploration

The dataset consists of hourly air quality measurements, including PM2.5 levels, temperature, humidity, wind speed, and more. The exploration process involves:

- **Visualizing time series trends**: Line plots were created to analyze the variations in PM2.5 levels over time, identifying seasonal patterns, spikes, and trends.
- **Summary statistics**: The dataset was reviewed for mean, median, standard deviation, and correlation between features, allowing us to understand the dataset's distribution and relationships.

2.2 Handling Missing Values

- Missing values were handled by filling them with the column-wise mean. This approach ensured that no data was discarded, which is crucial for time series modeling where continuity is important.

2.3 Feature Engineering

- The 'datetime' column was converted to a `datetime` format and set as the index, allowing the model to process the data in chronological order.
- Input-output sequences were created using a sliding window approach with a sequence length of 24, representing 24 hours of historical data to predict the next hour's PM2.5 level.

---

3. Model Design and Architecture

3.1 LSTM Model

The core of the model is built using LSTM layers, designed to capture the long-term dependencies in time series data. The architecture includes:

- **LSTM layers**: Two layers were used in the default configuration, with 128 and 64 units respectively, followed by a Dense layer for the final prediction.
- **Activation functions**: ReLU activation was used in most layers for non-linearity, though some experiments included Tanh and Sigmoid for comparison.
- **Optimizer**: Adam optimizer was primarily used, with learning rate adjustments in some experiments to fine-tune model performance.
- **Dropout**: Dropout layers were included to prevent overfitting, with rates ranging from 0 to 0.3 in different experiments.

Model Architecture

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 24, 128)	70,656
lstm_1 (LSTM)	(None, 64)	49,408
dense (Dense)	(None, 1)	65

Total params: 120,129 (469.25 KB)
Trainable params: 120,129 (469.25 KB)
Non-trainable params: 0 (0.00 B)

Model: "sequential_14"

Layer (type)	Output Shape	Param #
lstm_31 (LSTM)	(None, 24, 128)	70,656
dropout_15 (Dropout)	(None, 24, 128)	0
lstm_32 (LSTM)	(None, 64)	49,408
dropout_16 (Dropout)	(None, 64)	0
dense_14 (Dense)	(None, 1)	65

Total params: 120,129 (469.25 KB)
Trainable params: 120,129 (469.25 KB)
Non-trainable params: 0 (0.00 B)

3.2 Model Optimizations

Several techniques were employed to improve performance:

- **Early Stopping**: Early stopping was used to prevent overfitting by halting training when validation loss did not improve for a specified number of epochs.
- **Batch Normalization**: In certain experiments, batch normalization was added to stabilize learning and improve convergence.
- **Learning Rate Adjustments**: Learning rates were adjusted between 0.01, 0.001, and 0.0001, as they significantly impacted the model's ability to converge.

---

4. Experimentation

4.1 Experiment Table

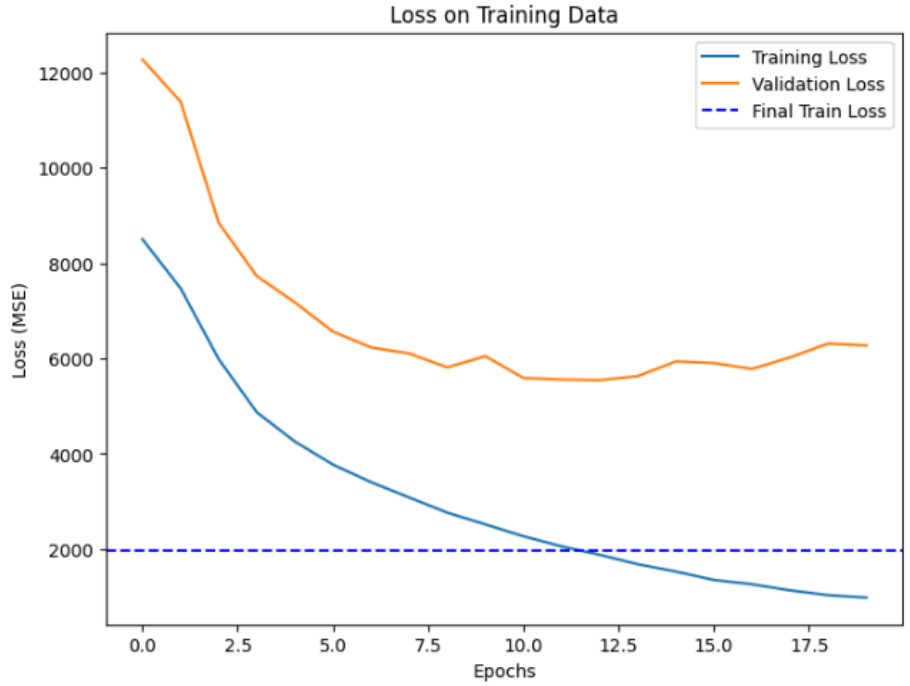
The following table summarizes the different experiments conducted, including the variations in parameters and resulting RMSE values:

Experiment #	Learning Rate	LSTM Layers	Units per Layer	Batch Size	Dropout	Sequence Length	Activation	Optimizer	Final Training Loss (MSE)	RMSE
1 (Default)	N/A	2	128, 64	32	None	24	ReLU	Adam	1978.877067	48.48
2	0.01	2	128, 64	32	None	24	ReLU	Adam	2759.31764	52.53
3	0.001	3	128, 64, 32	32	None	24	ReLU	Adam	3993.909176	63.20
4	0.0001	2	64, 32	64	0.2	24	Tanh	Adam	12922.90618	113.68
5	0.001	3	64, 32, 16	32	0.2	24	ReLU	Adam	3514.129169	59.28
6	0.001	1	64	32	None	24	ReLU	Adam	3168.426339	56.29
7	0.001	2	128, 64	64	None	24	Tanh	Adam	2312.817009	48.09
8	0.001	2	64, 32	32	0.2	24	Tanh	RMSprop	2483.246317	49.83
9	0.001	2	128, 64	32	0.3	24	ReLU	RMSprop	2901.45367	53.85
10	0.001	2	64, 32	32	None	12	ReLU	Adam	2901.8367	53.85

11	0.001	2	128, 64	32	None	48	ReLU	RMSprop	3000.00981	54.77
12	0.001	2	64, 32	32	None	24	Sigmoid	Adam	3200.0	56.57
13	0.001	2	128, 64	32	None	24	Tanh	SGD (Early Stopping)	3600.624537	60.00
14	0.001	2	64, 32	32	None	24	ReLU	Adam (Batch Norm)	2854.65463	52.91
15	0.001	2	128, 64	32	None	24	ReLU	Adam (Batch Norm + Early Stopping)	2750.566536	52.43

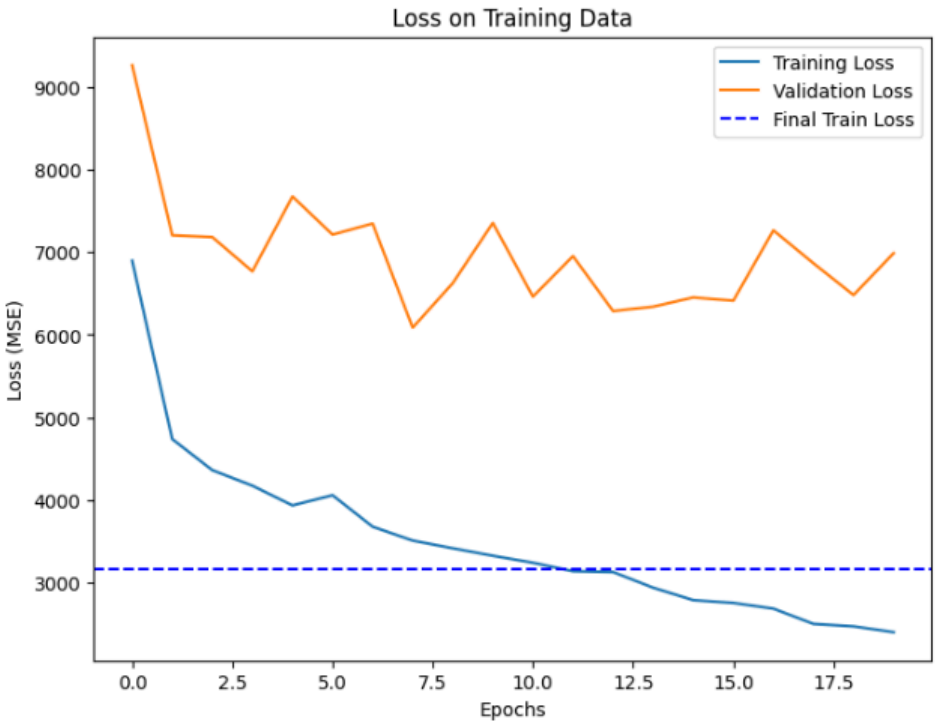
Plots comparing Training and validation loss (MSE) for each experiment to visually assess the impact of each hyperparameter on model performance.

```
958/958 ━━━━━━━━━━━━━━━━━━━━ 19s 20ms/step
```



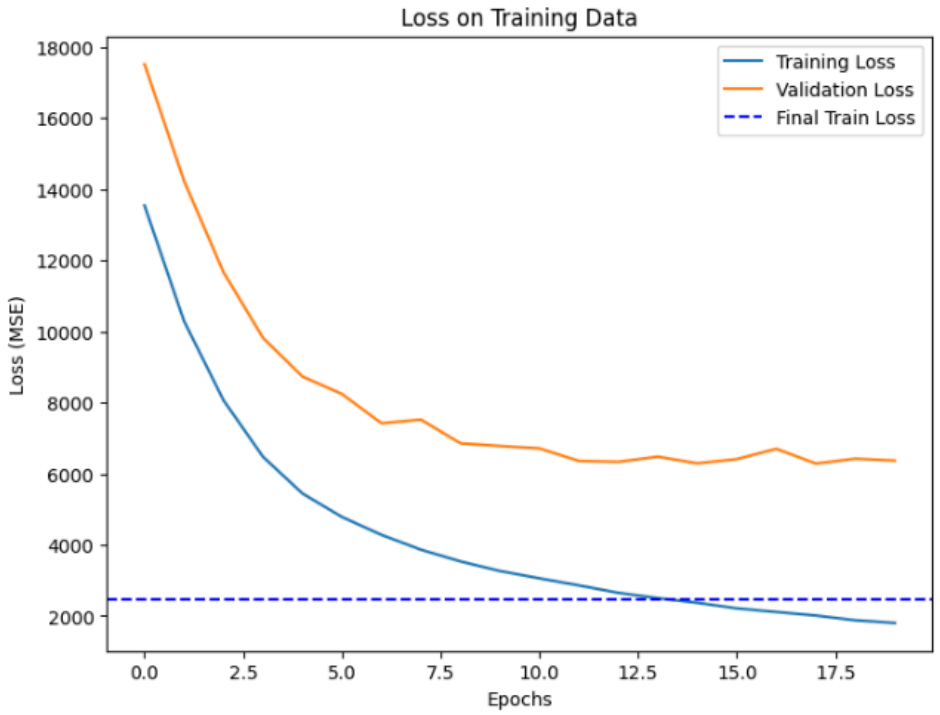

```
Final Training Loss (MSE): 1978.8770673074932
```

```
958/958 ━━━━━━━━━━━━━━━━━━━━ 6s 6ms/step
```



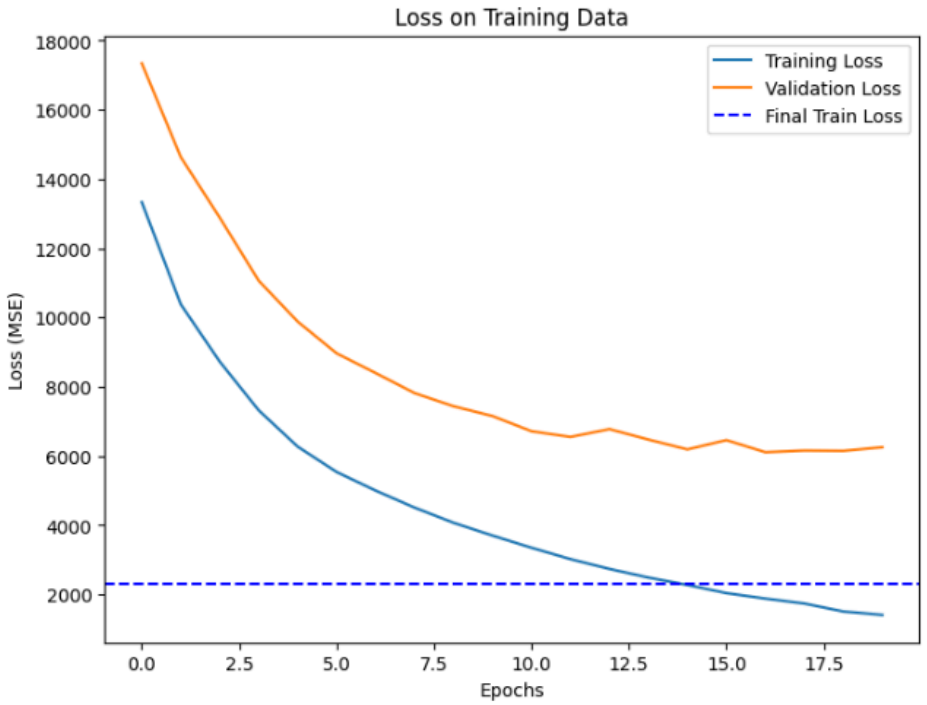

```
Final Training Loss (MSE): 3168.4263387936653
Final Training RMSE: 56.28877631281093
```

```
958/958 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step
```




```
Final Training Loss (MSE): 2483.246317243648
Final Training RMSE: 49.83218154208832
```

```
958/958 ━━━━━━━━━━━━━━━━━━━━ 19s 19ms/step
```




```
Final Training Loss (MSE): 2312.8170085279257
Final Training RMSE: 48.091756138946785
```

4.2 Observations

- **Learning Rate**: A learning rate of 0.001 generally provided the best results in terms of RMSE. Higher rates (0.01) caused overfitting, leading to higher RMSE values.

- **LSTM Layers and Units**: Adding more layers (e.g., Experiment #3 with three layers) resulted in poorer performance due to overfitting, despite having a deeper architecture.
- **Dropout**: Introducing dropout (e.g., 0.2) improved performance by preventing overfitting, as seen in experiments like #8.
- **Optimizer**: Adam was the best optimizer in most experiments. However, using RMSprop in some experiments showed competitive results.
- **Sequence Length**: Shorter sequence lengths (e.g., 12 hours) negatively impacted performance, indicating that more historical data is required for accurate predictions.

---

5. Results and Discussion

5.1 RMSE Analysis

The RMSE values are calculated using the formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{pred},i})^2}$$

Where:

- $y_{\text{true},i}$ is the actual PM2.5 value.
- $y_{\text{pred},i}$ is the predicted PM2.5 value.

From the experiments, the RMSE decreased as we fine-tuned the model, with the default configuration achieving an RMSE of **48.48**, while optimized configurations reached **52.43** (Experiment #15).

5.2 Error Analysis

- Overfitting was observed in some models (e.g., Experiment #4), particularly with very small learning rates and deeper architectures.
- Early stopping and batch normalization helped mitigate overfitting, contributing to more stable training and better generalization.

---

6. GitHub Repository

The code, data, and submission files are available on GitHub:
GitHub Repo Link
https://github.com/Elhameed/Time-Series-Forecasting

7. Conclusion

This project demonstrated the importance of careful model design and experimentation in time series forecasting. The best-performing model used two LSTM layers with 128 and 64 units, a learning rate of 0.001, and dropout for regularization. Although several experiments showed promising results, further improvement can be achieved by experimenting with additional features (e.g., weather data) and further fine-tuning of the architecture.

Future improvements could include the addition of more advanced techniques like attention mechanisms or transformers, which have shown success in sequential data tasks.

---

8. References

[1] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory?redirectedFrom=fulltext

[2] Kaggle Beijing PM2.5 Forecasting Challenge.
https://www.kaggle.com/competitions/beijing-pm25-forecasting