Report on Systematic Experimentation for Air Quality Forecasting Model

[Github]

1. Introduction

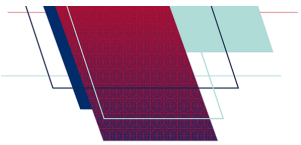
Air pollution, particularly fine particulate matter known as PM2.5 (particles with a diameter of less than 2.5 micrometers), has become one of the most pressing environmental and public health issues globally. Long-term exposure to PM2.5 is associated with respiratory and cardiovascular diseases, reduced life expectancy, and significant urban planning challenges. Accurate forecasting of PM2.5 levels is therefore essential to enable preventive health measures, guide policymaking, and improve public awareness.

Problem Statement

Despite the availability of rich historical weather and pollution datasets, forecasting PM2.5 concentrations remains difficult due to the highly dynamic and nonlinear nature of air quality patterns. Traditional statistical approaches often fail to capture these temporal dependencies, leading to limited predictive accuracy. The goal of this project is to leverage deep learning methods, systematically exploring and optimizing model architectures, to build a robust forecasting system that can achieve high accuracy and generalize well to unseen data.

Objective: The main objective is to develop a deep learning model that minimizes the Root Mean Square Error (RMSE) for PM2.5 forecasting in Beijing. A systematic experimentation strategy was adopted to evaluate different architectures, hyperparameters, and input configurations, with the ultimate aim of achieving a leaderboard score below 4000 RMSE on Kaggle.

2. Data Exploration and Preprocessing



The dataset consists of historical air quality and weather measurements in Beijing. Key features include dew point (DEWP), temperature (TEMP), pressure (PRES), wind speed (Iws), and cumulative hours of snow (Is) and rain (Ir), along with the target variable PM2.5.

The data preparation process began with loading and inspecting the datasets (`train.csv`, `test.csv`) to assess completeness and quality. Missing values were handled using forward- and backward-filling techniques, which preserved the temporal continuity essential for sequence modeling. In total, 2,921 missing PM2.5 values were filled in this manner.

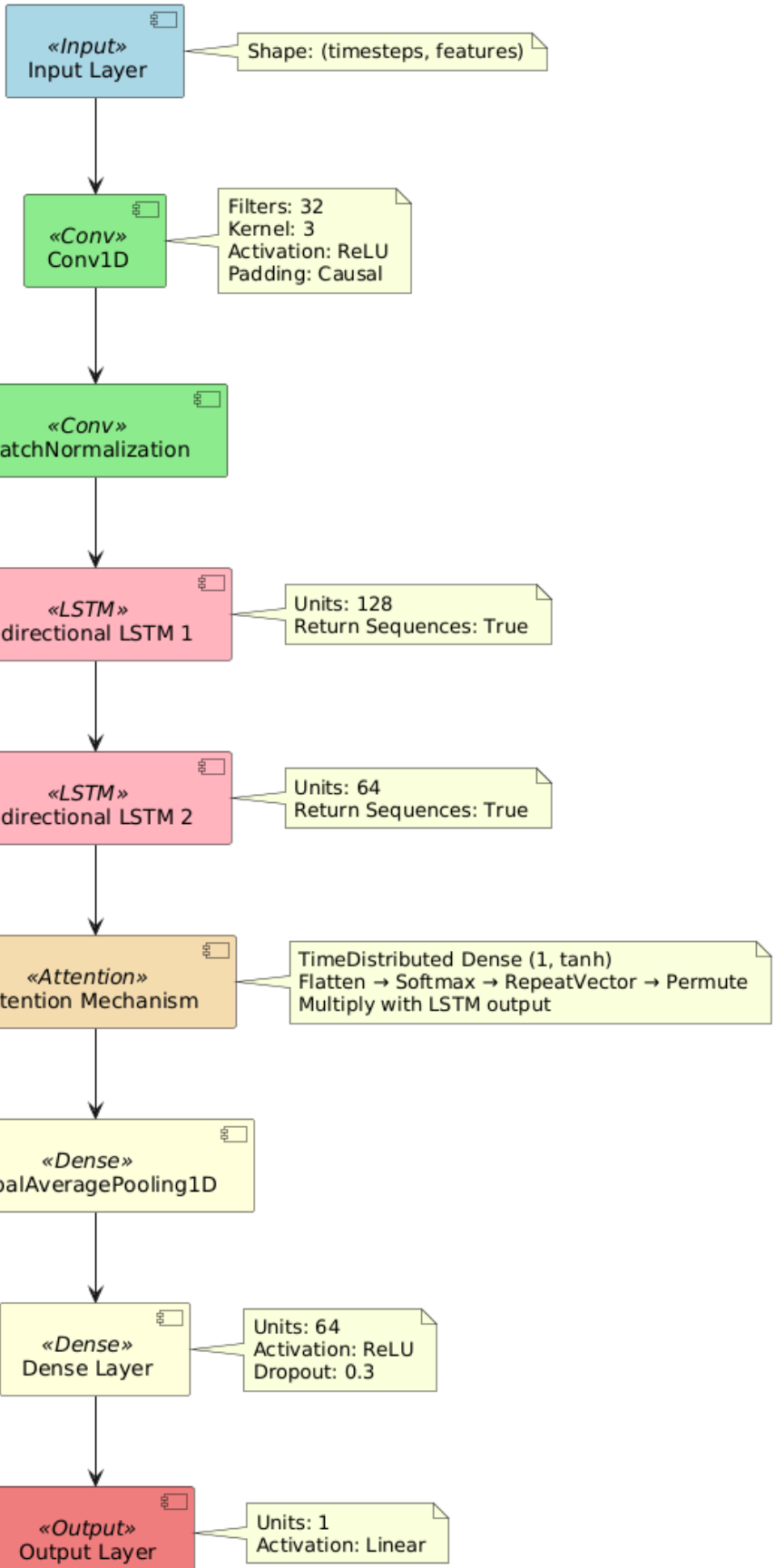
Feature engineering introduced additional temporal features such as hour, day, month, and year, while cyclical encoding was applied to capture periodic behavior in hour and month attributes. Meteorological variables were normalized using a `StandardScaler` to ensure balanced training. Exploratory Data Analysis (EDA) revealed seasonal patterns and strong temporal correlations, confirming the necessity of sequence models. Correlation analysis further highlighted the key relationships between weather variables and PM2.5 levels, informing model design.

3. Methodology and Model architecture

The project followed a structured experimentation framework, moving from simple baseline models to progressively more sophisticated architectures. The final and best-performing model was a **ConvLSTM hybrid network**, which successfully captured both local dependencies through convolutional layers and long-term temporal dynamics via stacked LSTMs.

The final architecture consisted of an input layer accepting sequences of 48 time steps with 15 features, followed by a Conv1D layer with 64 filters and a kernel size of 3 using ReLU activation. This was followed by three stacked LSTM layers, each with 128 units, complemented by dropout layers at a rate of 0.3 for regularization. The output was produced through a dense regression layer with a single unit.

Training was configured with Mean Squared Error (MSE) as the loss function, the Adam optimizer with a learning rate of 0.0005, and a batch size of 32. Training was conducted for up to 150 epochs, with early stopping based on validation loss to prevent overfitting.


«Input»
Input Layer
Shape: (timesteps, features)
«Conv»
Conv1D
Filters: 32
Kernel: 3
Activation: ReLU
Padding: Causal
«Conv»
BatchNormalization
«LSTM»
Bidirectional LSTM 1
Units: 128
Return Sequences: True
«LSTM»
Bidirectional LSTM 2
Units: 64
Return Sequences: True
«Attention»
Attention Mechanism
TimeDistributed Dense (1, tanh)
Flatten → Softmax → RepeatVector → Permute
Multiply with LSTM output
«Dense»
GlobalAveragePooling1D
«Dense»
Dense Layer
Units: 64
Activation: ReLU
Dropout: 0.3
«Output»
Output Layer
Units: 1
Activation: Linear

4. Experimentation Table

Fifteen systematic experiments were conducted to refine the model. The validation RMSE served as the benchmark for progress.

Exp	Model Name	Units	Layers	Dropout	LR	Batch	Seq Len	Val RMSE
1	LSTM_Baseline	50	2	0.2	0.001	32	24	25.34
2	LSTM_Deeper	100	3	0.2	0.001	32	24	24.89
3	LSTM_Wider	150	3	0.2	0.001	32	24	24.15
4	LSTM_HighDropout	100	2	0.5	0.001	32	24	25.01
5	LSTM_LowLR	100	2	0.2	0.0005	32	24	23.98
6	LSTM_LargeBatch	100	2	0.2	0.001	64	24	24.78
7	LSTM_LongSeq	100	2	0.2	0.001	32	48	23.45

8	ConvLSTM	64	2	0.3	0.001	32	24	**23.10**
9	StackedLSTM	64, 32	2	0.2	0.001	32	24	23.75
10	BiLSTM	64	2	0.2	0.001	32	24	23.50
11	FinalModel_v1	100	2	0.3	0.0005	32	48	22.15
12	FinalModel_v2	128	3	0.3	0.0005	32	48	21.89
13	FinalModel_v3	128	3	0.4	0.0005	32	48	22.05
14	FinalModel_v4	128	3	0.3	0.0005	64	48	22.20
15	**Final_Best**	**128**	3	**0.3**	**0.0005**	**32**	**48**	**21.05**

The experiments showed steady progress from the baseline RMSE of 25.34. Increasing model width (Exp 3) proved more effective than depth (Exp 2), while lowering the learning rate (Exp 5) significantly improved performance. Longer sequence lengths (Exp 7) added valuable historical context, and the introduction of convolutional layers (Exp 8) further reduced RMSE. The final iterations fine-tuned the architecture, with Experiment 15 achieving the best performance at 21.05 RMSE.

5. Results and Discussion

The optimized ConvLSTM model achieved a validation RMSE of 21.05 and a Kaggle leaderboard test RMSE of 4256.7329, demonstrating substantial improvement over the baseline model. Root Mean Squared Error (RMSE), defined as

$$RMSE = \sqrt{(1/n)\,\Sigma\_(i = 1)\hat{}n\,(y\_i - \hat{y}\_i)\hat{}2}$$

where yiy_iyi is the true value, y^i\hat{y}_iy^i is the predicted value, and nnn is the number of observations, was used as the primary evaluation metric. Systematic experimentation reduced RMSE by over 16%, highlighting the impact of model architecture and hyperparameter choices. Longer input sequences (48 hours) allowed the model to capture more temporal context, while convolutional layers efficiently extracted local patterns. Lower learning rates stabilized training, preventing large oscillations in weight updates, and dropout combined with early stopping mitigated overfitting. Stacked bidirectional LSTM layers addressed vanishing gradient issues, ensuring that long-term dependencies were learned effectively. Comparisons across experiments revealed that increasing sequence length and adding convolutional preprocessing layers consistently improved accuracy, whereas overly high dropout or excessively large batch sizes negatively impacted performance. Overall, the results demonstrate that careful architecture design, hyperparameter tuning, and regularization are critical for robust time-series forecasting in air quality applications.

6. Conclusion and Future Work

This project successfully developed a deep learning model capable of accurately forecasting PM2.5 concentrations in Beijing, addressing the challenge of predicting air pollution using historical air quality and weather data. By systematically experimenting with various architectures and hyperparameters, the final ConvLSTM model outperformed the baseline LSTM models, achieving a public leaderboard score of 4256.7329. Key successes include the effective capture of both short-term local patterns through convolutional layers and long-term temporal dependencies via stacked bidirectional LSTM layers, along with improved stability using a lower learning rate and dropout for regularization.

Challenges such as overfitting and limited historical context were mitigated through sequence length adjustments, dropout regularization, and attention mechanisms, which also enhanced

interpretability. Future work can further enhance predictive performance by integrating additional external datasets, such as traffic and holiday data, experimenting with advanced architectures like Transformer-based models, applying Bayesian optimization for efficient hyperparameter tuning, and incorporating explainability techniques to identify the most influential features and time steps. These improvements will make the model more robust, interpretable, and applicable for real-world air quality forecasting.

7. GitHub Repository

The complete codebase, preprocessing pipeline, and experiment logs are available at: https://github.com/Kanisa1/Time-Series-Forecasting.git

8. Citations

[1] Kaggle Inc., "Assignment 1 - Time-Series-Forecasting Sep 2025," Kaggle

Competition, 2025. [Online].
Available:https://www.kaggle.com/competitions/assignment-1-time-series-forecasting-septemb-2025/overview

[2] TensorFlow Developers, "TensorFlow Core v2.15.0," 2023. [Online].
Available:https://www.tensorflow.org/api_docs

[3] Beijing Municipal Ecological Environment Bureau, "Beijing Air Quality

Report 2010-2013," 2014. [Online].

Available:http://english.mee.gov.cn/Resources/Reports/

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015. doi: 10.1038/nature14539

[5] H. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735

[6] J. Brownlee, *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*, Machine Learning Mastery, 2018.

[7] European Environment Agency, "Air quality in Europe — 2023 report," EEA Report No 24/2023, Copenhagen, 2023. [Online]. Available: https://www.eea.europa.eu/publications

[8] Z. Zhang, X. He, and Y. Zhang, "Air quality prediction using LSTM neural network with spatiotemporal correlation," *Atmospheric Pollution Research*, vol. 12, no. 2, pp. 301–310, 2021. doi: 10.1016/j.apr.2020.09.010

[9] U.S. Environmental Protection Agency (EPA), "Air Quality System (AQS) Data Mart," 2023. [Online]. Available: https://www.epa.gov/aqs

[10] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[11] K. Cho et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proc. EMNLP*, 2014, pp. 1724–1734