Air Quality Forecasting Report

Air pollution is a global issue worldwide according to  World Health Organization (WHO) “Air Pollution Kills An Estimated Seven Million People Worldwide Every Year” according to this report Air pollution by WHO air pollution is also seen in other sectors apart from the general health such as urban planning

After learning about RNN’s and LSTM’s i took on the challenge to help contribute by predicting the air pollutions levels and reinforce my knowledge in machine learning since am an SE student

By contribution i was tasked to predict the pm25 levels or in other words the the air quality of Beijing from july 2013 to december 2014,

My approach to solve this problem was to first classify the data i was given to work with obviously it was time series data as you can tell

(1) My first step in data exploration more specifically was to find missing values.
For the first rows I removed them because methods like Backward Fill (bfill) fills with the missing values previous known value. Over things like median because it calculates on all values but destroy patters
(2) for middle missing values i used Linear Interpolation This uses the timestamps on your index to compute missing values.
(3) i removed unnecessary column which not do contribute in this dataset it “No”
(4) I converted the datetime column to datetime and set it as an index because time series operations use it to understand the temporal order.
(5) i then slit the train set and validation set from the original dataset,using the last 6100 rows as validation roughly 20% and scaled them between 0-1 using MinMaxScaler why ? because neural nets converge(training or learning process) faster when inputs are in a small

These were the steps I used in the data pipeline or in other words preparing the initial dataset for training. The next step was to create a model

(1) I choose to create an LSTM model over traditional RNN because first LSTM’s remembered
(2) I went with two LSTM layers (128 and 64 units) that capture temporal dependencies (in other words previous behavior on current behavior)
(3) return_sequences=true in the first layer allows passing full sequences to the next
(4) dropout (0.2) in between layers to reduce overfitting
(5) dense(64, relu + l2 regularization) learns complex nonlinear patterns while controlling overfitting.and do regularization
(6) final dense(1, linear) outputs the pm2.5 prediction.

The next to to find optimal values for model so below is a table with the 4 model values i experimented with

| batch size | Optimizer | Layers | Activation & regularizer | Dropout | Observations | |
|---|---|---|---|---|---|---|
| 15 | adam | 128 | tanh, tanh + l1_l2 | 0.3 | model captures long sequences but slight overfitting. | |
| 25 | adam | 128, 128 | tanh, tanh + l2 | 0.3 | Deeper network improves learning, slower convergence | |
| 50 | adam | 128, 64, 32 | tanh, tanh, relu + l2 | 0.3 | Added depth helps short + long dependencies, stable loss. | |
| 32 | adam | 128, 64, 64 | Relu relu, relu + l2 | 0.2 | ReLU throughout speeds training but less stable on time series | |
| 64 | adam | 128, 64, 32 | Relu relu, relu + l2 | 0.3 | Good trade-off between complexity and stability. | |
| 25 | rmsprop | 128, 64 | tanh, tanh, relu + l2 | 0.2 | Slight underfitting with larger batch, RMSE stable. | |
| 50 | rmsprop | 128, 64, 32 | tanh, tanh, relu + l2 | 0.3 | RMSProp handled learning rate well, smooth convergence. | |
| 16 | adam | 128, 64, 32 | tanh, tanh, relu + l2 | 0.2 | Stronger generalization, slightly better MAE. | |
| 32 | adam | 64, 64, 32 | tanh, tanh, relu + l2 | 0.1 | Smaller batch improved generalization, but noisier training. | |
| 25 | adam | 64, 64, 64 | Relu relu, relu + l2 | 0.1 | Smaller first layer reduced capacity, slightly worse MAE. | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 50 | adam | 128, 128, 64 | Relu relu, relu + l2 | 0.1 | Balanced architecture, steady improvements in RMSE. | |
| 32 | adam | 256, 128 | Relu relu, relu + l2 | 0.2 | Deeper model captured complex trends, slower training. | |
| 64 | adam | 128, 64, 32 | Relu relu, relu + l2 | 0.2 | Larger LSTM units captured long dependencies, but risk of overfitting. | |
| 25 | adam | 128, 64, 64 | Relu relu, relu + l2 | 0.3 | Stable performance, but higher dropout improved generalization slightly. | |

Key findings and conclusion

The first key finding In the beginning I was confused if should include the the target values inside the features so called lagged target values as some online materials suggested.but i found out that it causes data leakage

The next small finding was not fit transform both train and validations sets, rather fitting and transform only on train set and transform on the test and validation

These where the small challenges i faced when i started doing the challenge

Now on metrics i used the root mean squared error (RMSE) which measures prediction accuracy by penalizing larger errors more strongly. Formular : rmse = $\sqrt{(\sigma(y_i - \hat{y}_i)^2 / n)}$ and Visualizations are shown in the notebook

The final findings for my final model was that
(1) Increasing the first LSTM layer size to 128 improved the model's ability to capture long-term dependencies.
(2) The second LSTM layer (64 units) helped refine short-term dynamics. But also decreased it from which was an over kill and taking too long to train
(3) Regularization was key in stabilizing performance. It make my prediction better not by much but better

On improvements last semester i came across a technique of using pretrained neurons with more capacity than the one we can build our selves's and i added my own for my custom problem so i will try doing the same with RNN/LSTM the pre trained layers was a CNN i was taking about