Air Quality Forecasting Using Convolutional LSTM

Shuo Sun
Department of Computer Science
Stanford University
sunshuo@stanford.edu

Gaoxiang Liu
Department of Computer Science
Stanford University
gaoxiang@stanford.edu

Abstract

Air quality is now one of the biggest concerns around the world, especially in developing countries. Different air pollutants can harm people's health in a variety of ways. Being able to forecast air quality allows people to prepare beforehand and reduces the impact of hazardous weather. In this paper, we leverage the power of Convolutional LSTM architecture to tackle air quality forecasting problem. Convolutional LSTM, by combining convolution operation with LSTM architecture, has shown to be powerful with spatiotemporal sequences. By applying Convolutional LSTM, we achieve significant improvement over traditional fully connected LSTM and get top-tier performance compared to other teams in the KDD competition.

1 Introduction

Over the past decades, as industrialization progressed around the world, air pollution becomes an increasingly pervasive issue. PM2.5, one of the common yet very harmful air pollutants, is shown to be able to penetrate lungs and cause vascular inflammation and hardening of the arteries [1]. Providing air quality forecast plays a crucial role in reducing the impact of hazardous weather on people's health. The goal of air quality forecasting is predicting future air quality metrics, given historical weather and air pollutant concentration data of a region.

Since the objective is to predict sequential data, RNN is a natural choice of model. LSTM has been proven to be a powerful model for sequential data prediction [2]. However, the traditional fully connected LSTM (FC-LSTM) cannot efficiently catch the spatial correlation within the data, while the weather and air quality data are naturally correlated spatially, i.e. at one location, its future values are mostly related to its nearby locations. Meanwhile, convolutional neural network has shown its effectiveness of capturing the spatial correlation with sparsity of connection and parameter sharing.

Therefore, by combining LSTM and CNN, the model can be good at both predicting sequential data and capturing spatial relation. Shi et al. proposed the Convolutional LSTM algorithm applying to precipitation nowcasting [3]. The algorithm demonstrates its advantage and significantly outperforms the traditional FC-LSTM. In this paper, we use Convolutional LSTM to build air quality forecasting model, which takes a series of hourly weather and air quality metrics of Beijing of the past 24 hours, to predict the air pollutant concentrations measured by 35 air quality stations for the future 48 hours.

2 Related Work

Weather and air quality forecasting used to be a domain of statistical and mathematical models. In the past decades, researchers and mathematicians have built a variety of mathematical models dedicated to a range of tasks. For weather forecasting, a model based Ensemble Model Output Statistics (EMOS) is very effective [4]. For air pollution, a model based on Auto-Regressive Moving Average (ARMA) has shown to be powerful in SO_2 forecasting [5]. These methods, although can achieve state-of-the-art performance, require researchers' extensive effort to develop, and the results may highly depend on the climate of the target city or area.

CS230: Deep Learning, Spring 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

In addition, with the increasing availability of data and big data processing systems such as Hadoop, scientists proposed air quality forecasting methods based on massive weather and air quality data collection and processing system. In [6], the author designed a system based on GIS, sensors across the city, and complex computer data processing systems to predict the concentrations of PM2.5, PM10, and NO_2. This approach can potentially achieve very good performance with fast updates, but also requires immense computing power and high-coverage accurate sensor system.

In addition, with the significant progress of machine learning in the past decade, researchers have proposed many machine learning approaches for weather and air quality forecasting as well. In [7], the author uses neural network to forecast the concentration of SO_2, PM10 and CO. Zheng et al. propose an ensemble model consisting of linear regression, neural networks, dynamic aggregator, and inflection predictor to do fine-grained air quality forecasting [8]. Other models such as LSTM are also applied to hourly pollutant forecasting by Li et al. [9]. The machine learning approach is gaining dynamics in the recent years, but the performance is currently not as good as the traditional models.

3 Dataset and Features

3.1 Data Format

This project is based on the KDD competition of 2018. Therefore, KDD provides both weather and air quality data [10]. Each record of the weather contains station name, longitude, latitude, time, temperature, pressure, humidity, wind direction, and wind speed. Each record of air quality contains station name, time, PM2.5, PM10, NO_2, CO, O_3, and SO_2. The weather data of each timestamp is a $21 \times 31 \times 5$ tensor, with cells 0.1 degrees away in longitude and latitude from each other. The air quality stations are not uniformly distributed within the area. Both data are hourly records.

KDD competition provides full-year data of 2017, April 2018 data for pre-competition testing, and May 2018 data for final judgment. In order to align with the competition setup, we use the data of 2017 for training, that of April 2018 for validation, and that of May 2018 for final testing. During training, we split the data into 72-hour sequences, as the final goal is to use 24 hours of data to predict 48 hours of metrics. In test time, we split the one-month data into 30 24-hour sequences, and let the model predict for each day the next 48-hour metrics to calculate the quality metrics.

3.2 Mapping Air Quality Data to Weather Data Grid

While the weather data are tensors, the air quality data are not. Therefore, we need to map the air quality metrics to the weather grid. For this project, we do the mapping using the weighted average of the metrics of all 35 stations, where the weight is the inverse squared distance from the grid cell to each station, which is shown in (1). Here, $d_{i,cell}$ is the distance between the grid cell and station i, and ϵ is a small number to prevent divide-by-zero error:

$$AQI_{cell} = \frac{1}{\sum_i \frac{1}{d_{i,cell}^2+\epsilon}} \sum_i \frac{1}{d_{i,cell}^2 + \epsilon} AQI_i \tag{1}$$

3.3 Data Normalization

Weather and air quality data are of a wide range of scales. As shown in Figure 1, pressure data usually are of range $(800, 1100)$. Input and target of such large values make the optimization problem harder and can cause extremely large gradients. Therefore, in our project, we apply normalization for each sequence. During training time, we compute the mean and variance of the whole 72-hour sequence and apply normalization on both input and target. During test time, we compute the mean and variance based on the input 24-hour sequence and use the same values to re-scale the model output back to the unnormalized range.

4 Methods

To take advantage of the spatiotemporal relation of the weather and air quality forecasting problem, we use Convolutional LSTM (ConvLSTM). In traditional FC-LSTM, each cell state transition takes all the input values and previous hidden states as input, which introduces a very large number of parameters. In contrast, ConvLSTM uses convolution operation instead, which significantly reduces the number of parameters and improves the performance.

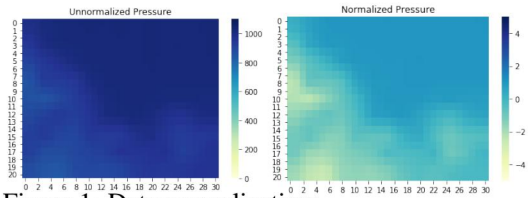

Figure 1: Data normalization: unnormalized vs normalized

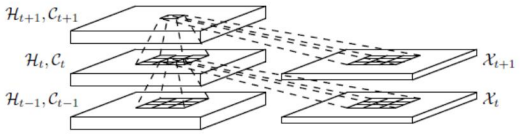

Figure 2: Inner structure of ConvLSTM [3]

4.1 Convolutional LSTM

Convolutional LSTM, proposed by Shi et al., is a powerful model when the sequential data show correlations in space. It uses convolution operation to computer the state transition gates, leveraging parameter sharing and sparsity of connection of data. As shown in Figure 2 [3], each state transition of a ConvLSTM cell only uses a very small subset of the input and the previous hidden states, making both training and computing much more efficient. The formulas of ConvLSTM are shown in (2). Here, we denote X_t as the input tensor, H_t as the hidden state tensor, and C_t as the cell state tensor.

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_t + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_t + W_{cf} \circ C_{t-1} + b_f) \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_t + W_{co} \circ C_{t-1} + b_o) \\
C_t &= f_t \circ C_{t-1} + i_t \circ tanh(W_{xc} * X_t + W_{hc} * H_t + b_g) \\
H_t &= o_t \circ tanh(C_t)
\end{aligned}
\tag{2}
$$

4.2 Air Quality Forecasting Model

The overall model is shown in Figure 3. The model takes the normalized input tensor and feeds it to a multi-layer ConvLSTM network. The output of the ConvLSTM is then used to generate two predictions. The first one is the air quality measure of each air quality station. To compute that, the model takes the closest 3×3 tensor surrounding each air quality station and applies another 3×3 convolution to make it into a vector. Then the model uses a fully connected linear layer to generate the concentration predictions of the 6 pollutants.

In addition, the model also uses the output of the ConvLSTM to predict the weather and air quality grid for the next time stamp. We choose to let the model uses and predicts weather data as well because we believe that weather and air quality predictions are closely related tasks. Therefore, by doing multitask learning, the model can achieve better performance. To predicts the grid, the model applies a 1 convolution with linear activation to the output of the ConvLSTM.

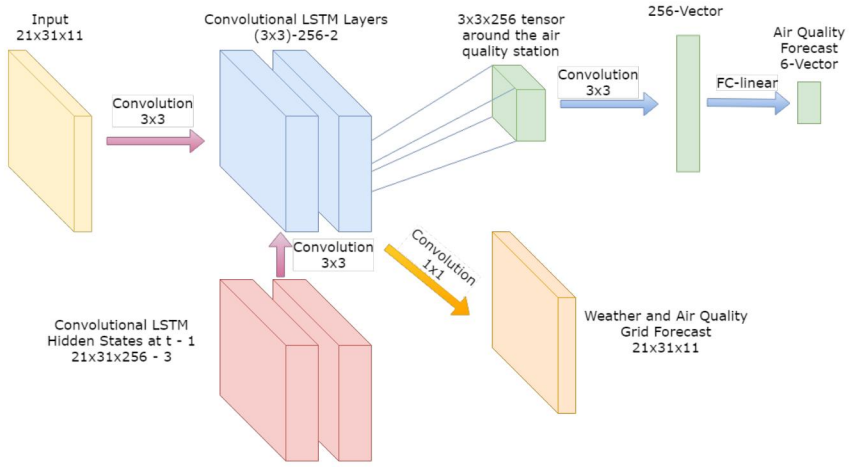

Figure 3: Air quality forecasting model

5 Experiments and Discussions

5.1 Experiment Setup and Metrics

In this paper, we compare the performances of FC-LSTM and ConvLSTM with different filter sizes, channels sizes, and numbers of layers. The model is trained using Adam optimizer with mean squared error (MSE) loss because all the prediction values are real numbers. To evaluate the model performance, we adopt the same metric used in the KDD competition, which is symmetric mean absolute percentage error (SMAPE), define as (3). Here, F_t and A_t are the forecasted and actual value respectively. The SMAPE score for one day is the average of SMAPE scores of the three grading metrics (PM2.5, PM10, and O_3) of all 35 air quality stations for the next 48 hours. The SMAPE score of the model is the average of smallest 25 daily scores of the 30-day testing data.

$$\text{SMAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2} \tag{3}$$

For this project, we use PyTorch [11] and NumPy [12] to implement our model, and use Matplotlib [13] and seaborn [14] to do visualization. We search online for PyTorch ConvLSTM implementation, but because PyTorch is relatively new and being updated frequently, we do not find a well-structured and up-to-date implementation of the algorithm. We end up writing our own version of it.

5.2 Hyperparameters and Training Process

In our experiment, we choose different combinations of filter size, hidden state channel size, and the number of layers. We experiment on filter size of 1×1, 3×3, and 5×5, in order to explore how important spatial correlation is to the model performance. We also train models with hidden states of 128 channels and 256 channels. This is to show how performance is related to hidden state size. In addition, we set up models with 2 and 3 layers of ConvLSTM networks. This is to show if a higher level of abstraction can help improve the model performance for this task.

To train the model, we apply gradient clipping by norm. Because our model is predicting real number values, having gradient clipping is important to prevent exploding gradient in case of encountering outliers in the input or target values. We first train our models without gradient clipping and find that after just one epoch, the $tanh$ function gets saturated and all the cell states of ConvLSTM become -1 or 1. By trying our different values, we find that norm clipping threshold of 10^{-5} and learning rate of 0.01 are a good combination to train the model fast enough and prevent exploding gradient.

We also apply learning rate decay and early stopping. With experimentation, we decide to drop learning rate to 0.001 after 5 epoch. We also keep track of dev set errors and stop training when dev set error reaches the minimum to prevent overfitting.

5.3 Baseline Model

We use FC-LSTM to build our baseline model. In the baseline model, we first use one 3×3 convolution with stride 1 and one 5×5 convolution with stride 2 to reduce the input size. Then we flatten the result into a vector and feed it to a 2-layer FC-LSTM with 2048-dimensional hidden states. Finally, the model applies FC linear layers to the LSTM output to predict the weather grid and air quality measures of the next timestamp.

5.4 Results

The final results are shown in Table 1, and the training and dev set loss curves are shown in Figure 4. For the table, ConvLSTM 3x3-256-2 means using ConvLSTM with 3×3 filter, hidden state of 256 channels, and 2 layers. We can see that all ConvLSTM models significantly outperform FC-LSTM model. In addition, ConvLSTM model with 1×1 filter performs worse than other ConvLSTM models. This shows that being able to capture the spatial correlation within the data is important to achieve good performance. But comparing to FC-LSTM, just by having much fewer parameters allows it to be trained much faster and achieve significantly better performance. However, having larger than 3×3 filter and deeper than 2 layer network, in this case, do not improve the model performance.

We think this is because, for weather and air quality data, each cell in the tensor correlates mostly just with cells next to it. Cells farther away do not provide much useful information. Therefore, looking at a larger area or trying to abstract higher-level information cannot help the model achieve better performance. Furthermore, having a larger filter and deeper network significantly increases the number of parameters, which makes the model harder to train. Therefore, in our experiment, the ConvLSTM 3x3-256-2 model performs the best in both test loss and the final SMAPE score, because it is easier to train compared with larger models, but also more expressive than the model with 128 channels. Comparing our result to the KDD competition participants, although we do not have the exact evaluation data used by the competition (the downloaded data used for forecasting has many missing values and time stamps), SMAPE score around $3.7 \sim 3.9$ range is still top tier performance.

Model	Dev Loss	Test Loss	Test SMAPE	Num of Params
FC-LSTM-2048-2	0.71110	0.77009	0.45841	126315819
ConvLSTM 1x1-256-2	0.50785	0.60865	0.40018	1462033
ConvLSTM 3x3-256-2	**0.46798**	**0.56391**	**0.36646**	**7843601**
ConvLSTM 5x5-256-2	0.46131	0.56712	0.37865	20606737
ConvLSTM 3x3-256-3	0.47290	0.56988	0.38748	12564241
ConvLSTM 5x5-256-3	0.48175	0.57809	0.38135	33715985
ConvLSTM 3x3-128-2	0.47511	0.57423	0.37960	1988497

Table 1: Results and metrics for different models

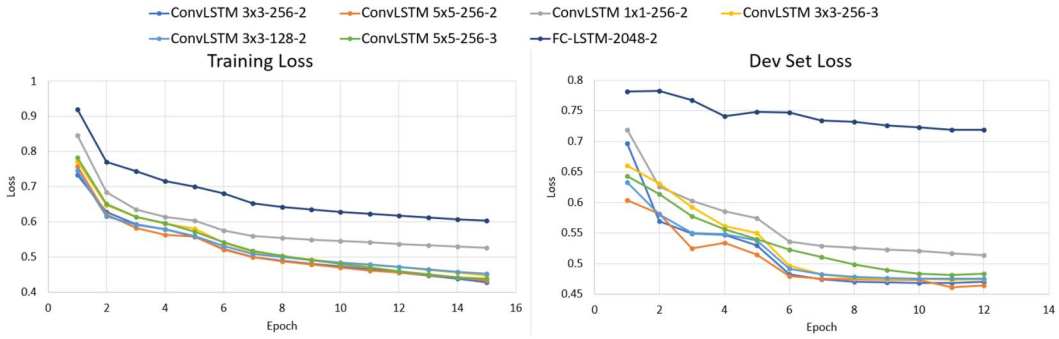

Figure 4: Training and dev losses of different models

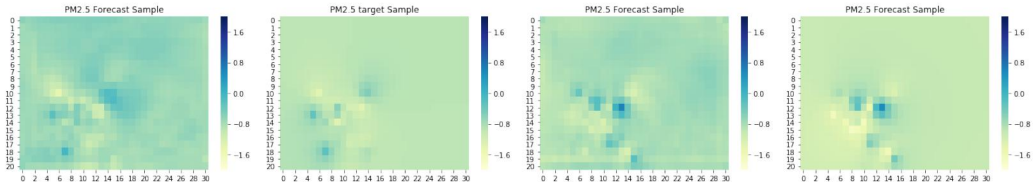

Figure 5: Prediction and Target Samples, for PM2.5 and O_3 respectively

6 Conclusion and Future Work

In this paper, by using Convolutional LSTM network to leverage the spatiotemporal correlation of weather and air quality data, we successfully build forecasting model to achieve high-level performance compared to other machine learning approaches. With convolution operation instead of full matrix multiplication used in FC-LSTM, ConvLSTM can be trained much faster and achieve significant performance improvement. In addition, by comparing models with different hyperparameters, the experiment shows that having an excessively large model can harm performance because it makes the model harder to train and take a longer time to converge. For future work, we will investigate combining ResNet architecture with RNN [15] to try to improve the performance over long sequences. We will also try to add convolution and deconvolution layers before and after the ConvLSTM component in our forecasting model, in order to leverage image super-resolution methodologies and reduce computational cost.

7 Contributions

Shuo Sun: Worked on model ideas, model coding, data processing coding, data visualization, paper writing, and poster design.

Gaoxiang Liu: Worked on model ideas, baseline model, data processing coding, data visualization, and paper writing.

References

[1] P. I. C, B. RT, T. MJ, and et al, "Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution," *JAMA*, vol. 287, no. 9, pp. 1132–1141, 2002.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[3] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, pp. 802–810, 2015.

[4] T. Gneiting, A. E. Raftery, A. H. Westveld III, and T. Goldman, "Calibrated probabilistic forecasting using ensemble model output statistics and minimum crps estimation," *Monthly Weather Review*, vol. 133, no. 5, pp. 1098–1118, 2005.

[5] G. Finzi and G. Tebaldi, "A mathematical model for air pollution forecast and alarm in an urban area," *Atmospheric Environment (1967)*, vol. 16, no. 9, pp. 2055–2059, 1982.

[6] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-air: When urban air quality inference meets big data," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1436–1444, ACM, 2013.

[7] A. Kurt, B. Gulbagci, F. Karaca, and O. Alagha, "An online air pollution forecasting system using neural networks," *Environment International*, vol. 34, no. 5, pp. 592–598, 2008.

[8] Y. Zheng, X. Yi, M. Li, R. Li, Z. Shan, E. Chang, and T. Li, "Forecasting fine-grained air quality based on big data," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2267–2276, ACM, 2015.

[9] X. Li, L. Peng, X. Yao, S. Cui, Y. Hu, C. You, and T. Chi, "Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation," *Environmental Pollution*, vol. 231, pp. 997–1004, 2017.

[10] Biendata, "https://biendata.com/competition/kdd_2018/data/," *KDD Competition*, 2018.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[12] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–.

[13] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[14] PyData, "Seaborn: statistical data visualization. https://seaborn.pydata.org/."

[15] Y. Wang and F. Tian, "Recurrent residual learning for sequence classification," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 938–943, 2016.