GreenEyes: An Air Quality Evaluating Model based on WaveNet

Kan Huang[1,*], Kai Zhang[2] and Ming Liu[3]

[1]The Hong Kong University of Science and Technology, Clearwater Bay, Hong Kong, China
[2]Lehigh University, 27 Memorial Dr W, Bethlehem, PA, United States
[3]The Hong Kong University of Science and Technology, Clearwater Bay, Hong Kong, China

Abstract

Accompanying rapid industrialization, humans are suffering from serious air pollution problems. The demand for air quality prediction is becoming more and more important to the government's policy-making and people's daily life. In this paper, We propose GreenEyes – a deep neural network model, which consists of a WaveNet-based backbone block for learning representations of sequences and an LSTM with a Temporal Attention module for capturing the hidden interactions between features of multi-channel inputs. To evaluate the effectiveness of our proposed method, we carry out several experiments including an ablation study on our collected and preprocessed air quality data near HKUST. The experimental results show our model can effectively predict the air quality level of the next timestamp given any segment of the air quality data from the data set. We have also released our standalone dataset at this URL. The model and code for this paper are publicly available at this URL.

Keywords

deep learning, neural networks, fitting model, regression analysis, AIoT

1. Introduction

With the development of the global economy and industrialization, people's living standards have improved, in the meanwhile, environmental problems such as air pollution have become a big concern. As World Health Organization (WHO) stated [1], air pollution is the world's largest environmental health risk, which will incur many diseases including but not limited to respiratory infections, heart disease, COPD, stroke, and lung cancer. Among all kinds of pollution, air pollution has the largest impact on premature deaths annually [2]. Hence, as people's awareness of health increases, more and more smart devices such as smart bands have been developed and equipped, which can report air quality status. Moreover, a smart indoor air purifier can automatically purify the air when the resident is not at home.

The air pollution problem is widely discussed in the field of Artificial Intelligence of Things (AIoT) and Sensing Networks. Some IoT systems with variant functions are designed to monitor air quality for different application scenarios [3, 4, 5]. For instance, Ray et al. [6] built a smart air-borne PM2.5 density monitoring system based on the cloud platform. However, these systems simply execute quality detection tasks without considering future air quality to let the purifier intelligently control its power level for energy-saving purposes. To bridge this gap, we propose the GreenEyes framework to predict the trend with previous air pollution levels. The feedback control system can be illustrated as Figure 1 shows.

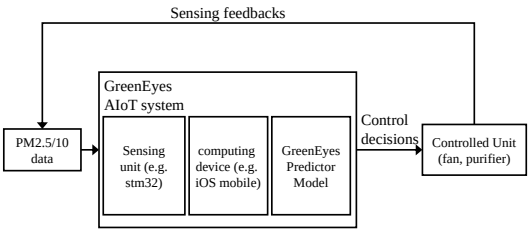

Figure 1: GreenEyes: AIoT deployment.

In this work, we firstly investigate the problem of preprocessing noisy PM2.5 sequence data and creating an appropriate supervising target sequence. We implement the GreenEyes model to predict the future air quality and evaluate it on each channel of PM2.5 data. Besides, we train our model with all channels' data together. Other works either use different kinds of data [7], or use sensors of the same model but place them at different places [6]. The former methodology is Multi-sensor Fusion [8], it is widely used in the intelligent and autonomous sys-

AMLTS'22: Workshop on Applied Machine Learning Methods for Time Series Forecasting, co-located with the 31st ACM International Conference on Information and Knowledge Management (CIKM), October 17-21, 2022, Atlanta, USA
*Corresponding author.
kan.huang@connect.ust.hk (K. Huang); kaz321@lehigh.edu (K. Zhang); eelium@ust.hk (M. Liu)
https://ai-huang.github.io/ (K. Huang); https://www.kaizhang.us/ (K. Zhang); https://ram-lab.com/people/ (M. Liu)
© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

tems [9, 10, 11, 12]. However, our approach of experiment proves that multi-sensors (of the same model, at the same place) will make the model perform better in predicting target data.

The main characteristics of this paper are summarized as follows:

- We treat WaveNet's residual layers as a feature block. This idea comes from the basic structures such as convolution-activation-pooling in computer vision. Such a design can increase reception filed and learn better representations.
- We innovatively stack several WaveNet blocks to build the model's main body. As the basic mechanism of deep learning networks is to build models brick by brick, the same module with different parameters is usually used in the same model. We borrow this idea and make it possible to parameterize our model. The model's optimal hyperparameters such as depth and filters can also be fine-tuned easily.
- We put Attention [13] and LSTM [14] at the endpoint as output layers. Ablation experiments demonstrate its necessity because this module can capture the hidden interations between features of different sequences (channels).

2. Datasets

AQI (**A**ir **Q**uality **I**ndex) is widely used for measuring the current pollution status of the air. IAQIs (**I**ndividual **A**ir **Q**uality **I**ndex) are calculated according to pollutants such as ozone, nitrogen dioxide, sulphur dioxide, and others, before final AQI is concluded. In our work, the IAQI of $PM_{2.5}$ is considered.

IAQI level data calculated from raw air quality data of sensors cannot be used directly because of high-frequency noise. As Figure 3 presents, in some intervals of the time axis, the IAQI level fluctuates very fast. This is because the air quality data is exactly fluctuating around the threshold line. In real AIoT applications, we don't need this fluctuation. Image the following module is a fan switch that takes the model's output to determine and we want this output to be relatively stable. In order to clean the data fluctuation while keeping the trend features, we innovatively brought out a method of human manually labeling. It creates an appropriated target label function that the model can learn. Also, based on the labeling tricks, the problem that the predictions on the IAQI level will fluctuate near the thresholds is much reduced.

2.1. Data Collection

We placed our 4 sensors in an office room located inside the Academic Building of the HKUST. The room is inside the academic building and has no windows, it provides a stable experimental environment for temperature and humidity. The sampling rate of the sensor is 1 Hz. We simultaneously collected around 220k data points for each sensor in a continuous period starting from 20:28 on 25th November 2019. This period is about 2 days and a half or 61 hours.

2.2. IAQI Calculation

The final AQI depends on each pollutant's IAQI, which is calculated by Equation 1

$$IAQI_p = \frac{C_p - BP_{Lo}}{BP_{Hi} - BP_{Lo}}(IAQI_{Hi} - IAQI_{Lo}) + IAQI_{Lo}, \tag{1}$$

and finally, AQI is calculated by Equation 2

$$AQI = \max\{IAQI_1, IAQI_2, IAQI_3, ..., IAQI_n\}. \tag{2}$$

In this paper, we only concern and discuss on the IAQI regarding $PM_{2.5}$.

Above equations about IAQI and AQI are universal for multi kinds of air pollution standards. Different thresholds are used when mapping air pollutants data into IAQI in different standards. Table 1 lists $PM_{2.5}$ and PM_{10} IAQI thresholds in China's and USA's standards respectively. In this paper, we use the **USA standard**.

Table 1
Concentration thresholds of IAQI w.r.t. pollutant categories, USA

IAQI	USA $PM_{2.5}$ ($\mu g/m^3$)	USA PM_{10} ($\mu g/m^3$)	China $PM_{2.5}$ ($\mu g/m^3$)	China PM_{10} ($\mu g/m^3$)
0	0	0	0	0
50	12.1	55	35	50
100	35.5	155	75	150
150	55.5	255	115	250
200	150.5	355	150	350
300	250.5	425	250	420
400	N/A	N/A	350	500
500	500.4	604	500	600

Figure 2 shows the IAQI curves corresponding to $PM_{2.5}$, with thresholds lines.

2.3. Data Polynomialization

The task of our model is to predict the IAQI level when inputting a segment of air pollutant concentration data. However, the origin IAQI level lines cannot be directly used because 1. in deep learning, a step function is very hard to learn especially on the rising and falling edges;

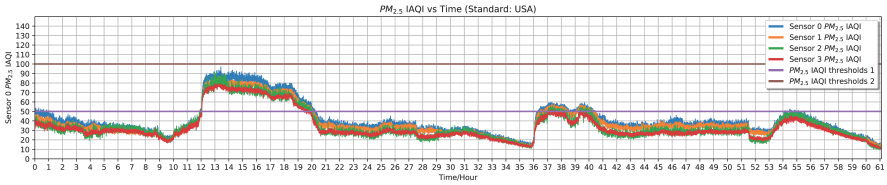

Figure 2: All $PM_{2.5}$ data with IAQI thresholds.

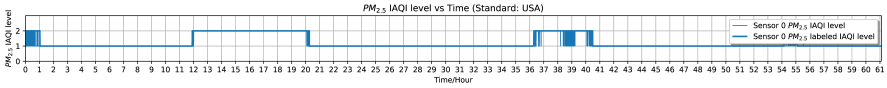

Figure 3: Sensor 0's $PM_{2.5}$ origin and labeled IAQI level.

2. in some areas the IAQI level fluctuates extremely frequently, which makes the learning even harder, as shown in Figure 3.

We're inspired by B. Rouet-Leduc's work [15] on earthquake predicting, where the earthquake events are represented as failures. The target function is designed as a series of descending ramps. Mathematically, these descending ramps form a polygonal function and count down the time to next earthquake. The problem is turned into fitting the time counting down curve with the acoustic data input.

A polygonal function, also named piece-wise linear function $f(x)$, is a continuous function mathematically defined on an interval $[a, b] \in \mathbb{R}$ such that $[a, b]$ can be divided into a set of intervals on each of which the function is a linear function, that is, there exists a subdivision

$$a = x_0 < x_1 < ... < x_n = b \tag{3}$$

such that $f(x)$ is linear on each interval $[x_{n-1}, x_n]$.

Polygonal functions can be used to generate approximations to known curves, planes, etc. Also, for unknown data, polygonal functions can also be learned by some algorithms such as decision tree, to fit the data. In our work of predicting, polygonal functions help us to eliminate the **hesitation** area, and build the target data.

2.4. Data Polygonalization: Human Labeling based on Decisions

We firstly label by hand the level step downup points, and map them into risingfalling lines. This method transfer discrete decision points into continuous target data series which have the same dimension as the time indices and corresponding $PM_{2.5}$ data. This kind of method make us get the polygonal target data as B. Rouet-Leduc, et al. [15] did. Figure 3 shows our labeling results.

Finally, the labeled target data is turned into a polygonal function. Dash lines in Figure 4 shows the results. These triangular lines will be label data for our supervised learning fitting problem.

The level polygonal lines w.r.t. their manually labeled level curves can be written as equations with form below

$$L_i(t) = k_i * t + b_i, \ where \ t \in [t_i, t_{i+1}], \tag{4}$$

where k_i is the slope of the curve, t_i and t_{i+1} are start and end time point for every interval of the polygonal line. When $k_i > 0$, the trend of the IAQI level is raising, and vice versa. The absolute value of k_i is the approximate and potential changing speed of IAQI level. Thus, every polygonal line can be divided into several segments within time interval t_i to t_{i+1}, and every segment estimates the 1-order approximate trend w.r.t. original IAQI level within corresponding time interval. For the i-th segment, k_i is

$$k_i = \frac{l_{i+1} - l_i}{t_{i+1} - t_i}. \tag{5}$$

where l_{i+1} and l_i are the original IAQI levels at end and start time t_{i+1} and t_i.

Our experiments will take these polygonalized IAQI level lines as the supervising data. The fitting problem can be described as: given a IAQI sequence of *windows size*, predict the IAQI level of the next time frame after this time window.

3. Methodology

Recently, a series of neural networks related to the autoregression model has been proposed and applied in regarding problems. DeepMind's WaveNet [16] is one of the famous and foundation work in between those [17], [18] tackle with sequence representation and generating.

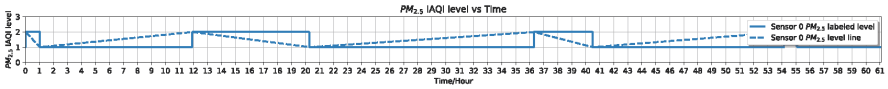

Figure 4: Sensor 0's labeled $PM_{2.5}$ IAQI level and its polygonal line.

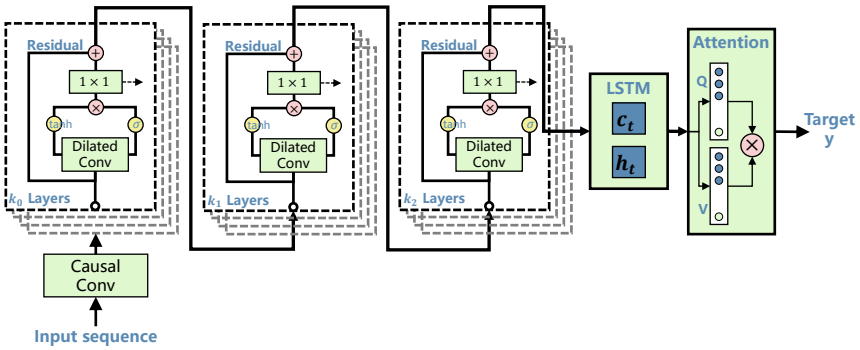

Figure 5: GreenEyes sequence to point fitting model.

For WaveNet's application scenario, the joint probability of the target sequence $\mathbf{x} = \{x_1, x_2, ..., x_T\}$ is factorized as a product of conditional probabilities as the below Equation 6. Given an input $\vec{\mathbf{x}} = \{x_1, x_2, ..., x_{T-1}\}$ and with this conditional probabilities, we can obtain the distribution of the value x_T, and make generation samples.

$$p(\mathbf{x}) = \prod_{t=1}^{T} p\{x_t | x_1, x_2, ..., x_{t-1}\} \tag{6}$$

Auto-regression models can not only be used in data generation, but also in time series prediction. In our work, every sample x_t and y_t at any time step t is conditioned on the samples at all previous timestamps, making it a multivariate auto-regression task. To limit the input length, we only consider the conditional probabilities between x_t and a sequence $x_{t-1-window\_size:t-1}$ with length $window\_size$. Different with other multivariate auto-regression tasks where sequences on all the temporal axis are modeled, we haven't used the sequence $y_{t-1-window\_size:t-1}$ to predict y, instead, we predict y only with $x_{t-1-window\_size:t-1}$.

Different from B. Rouet-Leduc's work[15], in which random forest is used to predict seismic precursors, we use WaveNet as our GreenEyes model's main part. Air pollution data has the same structure as audio data. It is pretty suitable to utilize WaveNet as air pollution data can be modeled in the same way. Also, WaveNet's dilated causal convolutions and residual and skip connections are suitable for air pollution data.

We used the original WaveNet's core part as a WaveNet Block as we believe this blockstyle configuration is more modularized, for we could change these blocks' hyper parameters more easily. Each WaveNet Block, as the same with WaveNet, contains several dilated convolution layers, called WaveNet Layer. Different dilation rates are also set on them, following DeepMind's original work.

The designing of neural networks for deep learning has always followed principles such as modularization, and expandability. Well-known networks, such as VGG [19] and ResNet [20], all have these features. VGG has two model types VGG16, and VGG19, with different model depth. And ResNet has models ResNet-18, ResNet-34, ResNet-50, etc. The cutting-edge model, Transformer [21], also obeys these designs which makes it possible to build multi variant models for various sizes and application scenarios. Our model is designed for parameterization, too. Following our constructions, finally we set 8 WaveNet layers for the first block; and 5 layers for the second, 3 layers for the third. All blocks share the same kernel size of 3, and filters of 16. This set of hyper parameters are chosen by empiric and the computational capability of a 1080 Ti GPU. There might be more optimal parameters to search in future works.

As for the Attention layer, we set up two kinds of Attention mechanism - Dot-product attention layer, a.k.a. Luong-style attention [22], as Equation 8 shows. We use the input for all value vector, key vector, and query input. Another mechanism is made by ourselves, called Temporal Attention.

$$\text{scores} = QK^T \tag{7}$$

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V \tag{8}$$

In our Attention layer, we still use the Luong's multiplicative style attention 9 to gain score, but we simply it with a FC network. Moreover, we don't use softmax function to compute the attention weight. Rather, we use the function as Equation 11 shows.

$$\text{scores}(\bar{\boldsymbol{h}}_t^T, \bar{\boldsymbol{h}}_s) = \bar{\boldsymbol{h}}_t^T \boldsymbol{W} \bar{\boldsymbol{h}}_s \tag{9}$$

$$\text{scores} = \boldsymbol{W}\boldsymbol{V} + \boldsymbol{b} \tag{10}$$

$$\text{Attention}(V) = \exp(\tanh(\text{scores}))V \tag{11}$$

The reason that we replace the softmax with a tanh function followed by an exponential function, is to better adapt our model to the temporal data set. Our data set have many temporal and periodic features to learn. Tanh function is very common in sequential models, and it is also a component in every WaveNet layer.

4. Experiments

4.1. Experimental Settings

As we sampled $PM_{2.5}$ measurements from 4 sensors, Sensor 0 to Sensor 3, so we have a 4-channels $PM_{2.5}$ IAQI data set. Each channel's data can be taken as an individual data set. The stride is set as $\{10, 5, 2\}$, respectively. Besides, we fuse data from all channels to create a new data set named $PM2.5_{All}$.

Adam[23] optimizer with an initial learning rate 0.0001 is applied in the experiments, which is multiplied by 0.1 after 20 epochs, where the total training epoch is 100. We use mean squared error (MSE) and mean absolute error (MAE) as the evaluation metrics.

4.2. Training and Validation

4.2.1. Why did We Redesign the Attention Layer?

At first, we utilized the dot-product attention layer provided by TensorFlow official. Table 2 lists all the experiments' final best metrics during training.

After we train the model with Temporal Attention, we discovery that the results on official Attention show limitations and defects. As Table 3 shows, in most experiments, Temporal Attention outperforms official Attention. When we plot the validation curve, some principles can be figured out, specifically, Figure 6 illustrates the validation MSE's curves with $stride = 10$, and Figure 7 illustrates the validation MSE's curves when applying Temporal Attention. We can conclude that when applying official Attention, the model cannot converge consistently with different data sets. Figure 6 shows that model fails to converge when it learns on $PM_{2.5}(0)$. Meanwhile, applying Temporal Attention, the model can obtain a better MSE.

Data	Stride	Minimum train MSE	Minimum validation MSE
$PM_{2.5}(0)$	10	0.0969	0.1221
	5	0.0071	0.0221
	2	0.0049	0.0148
$PM_{2.5}(1)$	10	0.0148	0.0226
	5	0.0062	0.0137
	2	0.0006	0.0039
$PM_{2.5}(2)$	10	0.0087	0.0137
	5	0.0080	0.0153
	2	0.0027	0.0154
$PM_{2.5}(3)$	10	0.0123	0.0209
	5	0.0058	0.0110
	2	0.0023	0.0037
$PM_{2.5}(\text{All})$	10	0.0182	0.0266
	5	0.0039	0.0053
	2	0.0010	0.0005

Table 2
Best metrics during training when applying official Attention.

Figure 6: Validation MSE curves when applying official Attention ($stride = 10$).

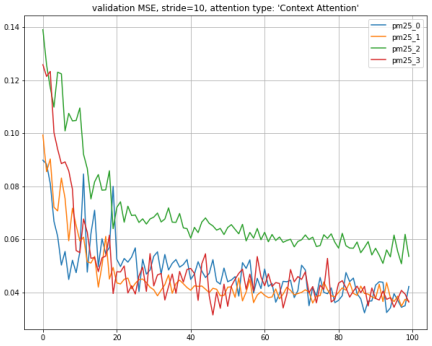

Figure 7: Validation MSE curves when applying Temporal Attention ($stride = 10$).

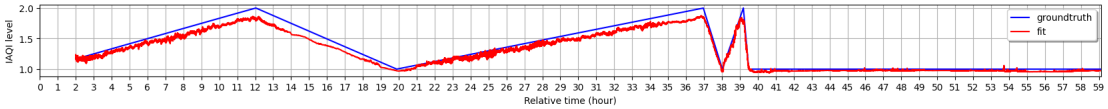

Figure 8: Evaluation of the GreenEyes model ($PM_{2.5}(3)$, stride=5).

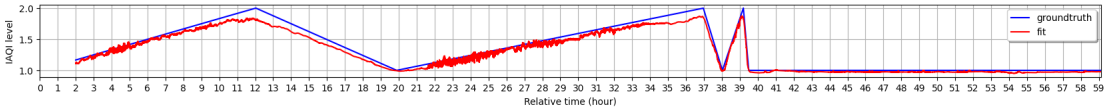

Figure 9: Evaluation of the GreenEyes model ($PM_{2.5}(3)$, stride=10).

4.2.2. Best Metrics during Training

Table 3 shows the experimental best metrics during training with the proposed Temporal Attention. It is obvious that in most cases, our Attention outperforms the official Attention. We also define a coefficient $ratio = \frac{min(train\ MSE)}{min(validation\ MSE)}$ to simply measure the generalization capability of the model.

Data	Stride	Minimum train MSE	Minimum validation MSE	ratio
$PM_{2.5}$(0)	10	0.0223	0.0234	0.96
	5	0.0034	0.0114	0.30
	2	***0.0006***	***0.0035***	0.16
$PM_{2.5}$(1)	10	0.0486	0.0510	0.95
	5	0.0058	0.0142	0.41
	2	***0.0006***	***0.0036***	0.17
$PM_{2.5}$(2)	10	0.0171	0.0187	0.92
	5	0.0024	0.0092	0.27
	2	***0.0012***	***0.0066***	0.19
$PM_{2.5}$(3)	10	0.0509	0.0468	1.09
	5	0.0074	0.0167	0.44
	2	***0.0010***	***0.0068***	0.15
$PM_{2.5}$(All)	10	0.0068	0.0103	0.66
	5	0.0014	0.0022	0.67
	2	**0.0007**	**0.0009**	0.77

Table 3
Best metrics during training when applying Temporal Attention.

4.3. Model Evaluation

Figure 8 shows that our model fits the labeled IAQI level lines well, except that its predictions differ from the ground truth a little on some parts of the lines, especially on the turning corners. Figure 9 illustrates the same evaluation performance, which presents that the model may not need much data to learn as to set stride to 2. To quantify the testing results of our model with different parameters, we test it on the whole $PM_{2.5}$ sequence by setting stride as 1. Table 4 lists the statistics of our tests.

Data	Stride	MSE	MAE
$PM_{2.5}$(0)	10	0.0266	0.13
	5	0.0144	0.11
	2	***0.0037***	***0.05***
$PM_{2.5}$(1)	10	0.0517	0.18
	5	0.0113	0.10
	2	***0.0036***	***0.05***
$PM_{2.5}$(2)	10	0.0188	0.11
	5	0.0092	0.09
	2	***0.0069***	***0.07***
$PM_{2.5}$(3)	10	0.0501	0.16
	5	0.0108	0.09
	2	***0.0070***	***0.07***
$PM_{2.5}$(All)	10	0.0118	0.09
	5	0.0026	0.04
	2	**0.0010**	**0.02**

Table 4
Test MSE and MAE under different stride parameter.

4.4. Ablation Study

In order to validate the effectiveness of the modules, we conduct an ablation study on our GreenEyes model. We remove the bidirectional LSTM module and the mutlihead attention module, respectively, and get two model variance, w/o Attention and w/o LSTM. We plot the model's (w/o LSTM) training and validation curves as Figure and Figure show respectively.

It is easily concluded that, without the LSTM layer, the model runs into the overfit status. Although it still fits well on the train set, it is rambling on the validation set.

In order to validate the Attention layer's function, we re-run the GreenEyes model with Temporal Attention on $PM_{2.5}$(0) to $PM_{2.5}$(3), and then cut off this Attention layer and run the model again on the same data sets. Table 5 shows the test MSE and MAE results of both configuration. It turns out that the model w/o Attention can

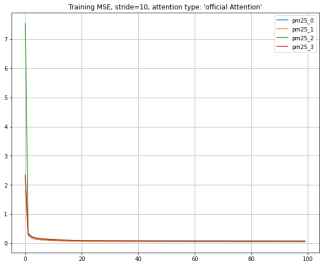

Figure 10: Model's (w/o LSTM) training plots (stride=10).

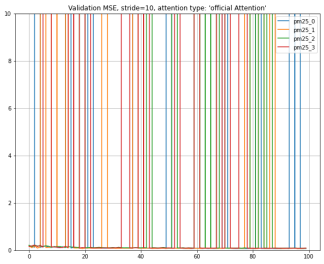

Figure 11: Model's (w/o LSTM) validating plots (stride=10).

perform better or is equivalent to the model applied with the Attention layer. However, by plotting the training curves again, we found that the model with the Temporal Attention layer can obtain smaller loss during training.

Data	Stride	Our Attention		w/o Attention	
		MSE	MAE	MSE	MAE
$PM_{2.5}(0)$	10	0.0267	0.1438	**0.0173**	**0.1189**
	5	0.0119	0.1011	**0.0092**	**0.0795**
	2	**0.0055**	**0.0661**	0.0078	0.0721
$PM_{2.5}(1)$	10	**0.0256**	0.1383	0.0262	**0.1292**
	5	0.0151	0.1093	**0.0097**	**0.0769**
	2	0.0026	0.0423	**0.0015**	**0.0305**
$PM_{2.5}(2)$	10	0.0409	0.1548	**0.0174**	**0.1170**
	5	0.0116	0.0978	**0.0053**	**0.0624**
	2	0.0057	0.0577	**0.0007**	**0.0202**
$PM_{2.5}(3)$	10	**0.0213**	**0.1338**	0.0446	0.1726
	5	**0.0064**	**0.0659**	0.0083	0.0759
	2	0.0044	0.0534	**0.0018**	**0.0305**

Table 5
Test MSE and MAE for model with and w/o Attention.

4.5. Hyper-parameter Discussion

Being inspired by the SOTA ideas of predicting the target sequence with a short sequence by using an auto-regression model such as Autoformer [24], we approach to decrease the model's input size, i.e., the data's window size. We set the window size to 3600 (which means one hour on the timeline), and train our model again. Figure 12 shows our results. Empirically, the model gains well performance as long as it reduces the training loss under 0.01. Hence, except for the result on $PM_{2.5}(3)$ when the window size is set to 3600, the model still needs optimization if we want a shorter window size. However, it is worth trying as the number of model parameters also decreases obviously as the input size is reduced. A light model saves computational costs and boosts inference.

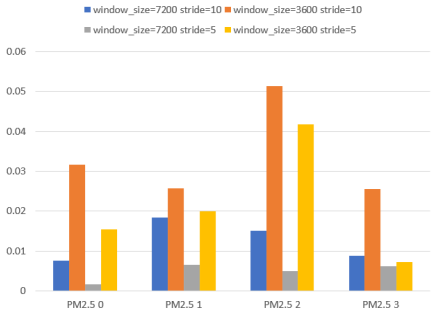

Figure 12: Window size 7200 vs 3600.

5. Conclusion

The WaveNet model designed for audio data processing is generalizable and suitable for fitting problem. Our work successfully put it into usage for IAQI level fitting and prediction. It shows that our GreenEyes model based on WaveNet has strong data fitting capability for extreme long data sequences. When given a smaller stride, fed with more data, the model can learn better. It is also found that, when trained with more channels of sensor data, the model can perform well. This can be regard as sensor data augmentation. Our innovative method that human manually label the IAQI level is useful. It creates an appropriated target label function that the model can learn and solve the threshold fluctuation problem.

It is also promising that our GreenEyes AIoT deployment design can be put into practice. Actually we've developed an iOS app to retrieve the air quality data. Mobile framework such as Tensorflow Lite [25] has been developed. A mobile phone is hopefully to be installed with our GreenEyes model and monitor the IAQI data in realtime and predict the air trend.

Due to a lack of air quality data, we only did the data fitting task. We will perform the data predicting task in the future if enough data is gathered.

6. Related Works

6.1. Statistical & Machine Learning Approaches

Except for ARIMA, ETS models mentioned in our last chapter, traditional methods such as Kalman filter [26] are also very simple and practical for time series and forecasting problems. Random forests [15], XGBoost, and SVM [27] etc are useful machine learning methods too. About method choosing, the most suitable method is highly interrelated with the data's properties and the application scenario.

In common, the essential of both traditional approaches and ML-based approaches is mining data and extracting features. Different from other feature engineering tasks, sliding windows are widely used for processing the data. Metrics such as the minimum, the maximum, the mean, and the variance of the data in the window are common features.

6.2. Deep Learning Approaches

LSTM-based deep learning methods have been developed recently to extract temporal patterns. Lai et al. proposed LSTNet [28] that encodes short-term local information into low dimensional vectors using 1D convolutional neural networks and decodes the vectors through an RNN. Shih et al. proposed TPA-LSTM [29] which processes the inputs by an RNN and employs a convolutional neural network to calculate the attention score across multiple steps.

The architecture of CNN is designed for 2D data like images. Meanwhile, recently a special variant of CNN called temporal convolutional networks (TCNs) [30] has been proposed that makes CNN capable for time series processing. Yan et al. [31] released their research work about using TCN for weather forecasting in 2020 and showed that TCN is better than the LSTM network in this application.

WaveNet related methods, including our GreenEyes model, tackle with a single sequence of time series data and show good fitting and forecasting performance concerning the prediction accuracy and data throughput capacity. Meanwhile, same with the same recent time as this thesis was being developed, new methods and approaches regarding time series forecasting have also been proposed. In recent years, graph neural networks (GNNs) have shown high capability in handling relational dependencies. Wu et al. [32] proposed a general graph neural network framework designed specifically for multivariate time series data. Their method is useful for extracts relations among variables belonging to multi sequences.

As Transformer [21] becomes great popular these years, another model based on Transforms has also been brought out. Lim et al. [33] from Google introduced the Temporal Fusion Transformer (TFT) as a novel attention-based architecture which combines high-performance multi-horizon forecasting with interpretable insights into temporal dynamics. They created gate-based networks, GRN and GLU, as new approaches for better feature selection modules.

Acknowledgments

The authors would like to thank many friends for constructive discussions and feedbacks. Special thanks to Prof. Yuan Yao who voluntarily provides GPU machine.

References

[1] W. H. Organization, et al., Ambient air pollution: A global assessment of exposure and burden of disease (2016).

[2] J. Lelieveld, J. S. Evans, M. Fnais, D. Giannadaki, A. Pozzer, The contribution of outdoor air pollution sources to premature mortality on a global scale, Nature 525 (2015) 367–371.

[3] S. Kumar, A. Jasuja, Air quality monitoring system based on iot using raspberry pi, in: 2017 International Conference on Computing, Communication and Automation (ICCCA), IEEE, 2017, pp. 1341–1346.

[4] C.-S. Oh, M.-S. Seo, J.-H. Lee, S.-H. Kim, Y.-D. Kim, H.-J. Park, Indoor air quality monitoring systems in the iot environment, The Journal of Korean Institute of Communications and Information Sciences 40 (2015) 886–891.

[5] K. Zheng, S. Zhao, Z. Yang, X. Xiong, W. Xiang, Design and implementation of lpwa-based air quality monitoring system, IEEE Access 4 (2016) 3238–3245.

[6] P. P. Ray, Internet of things cloud based smart monitoring of air borne pm2. 5 density level, in: 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), IEEE, 2016, pp. 995–999.

[7] J. Han, H. Liu, H. Zhu, H. Xiong, D. Dou, Joint air quality and weather prediction based on multi-adversarial spatiotemporal networks, arXiv preprint arXiv:2012.15037 (2020).

[8] Z. Wang, Y. Wu, Q. Niu, Multi-sensor fusion in automated driving: A survey, IEEE Access 8 (2019) 2847–2868.

[9] R. C. Luo, M. G. Kay, Multisensor integration and fusion in intelligent systems, IEEE Transactions on Systems, Man, and Cybernetics 19 (1989) 901–931.

[10] D. L. Hall, J. Llinas, An introduction to multisensor data fusion, Proceedings of the IEEE 85 (1997) 6–23.

[11] L. Wang, M. Liu, M. Q.-H. Meng, R. Siegwart, Towards real-time multi-sensor information retrieval in cloud robotic system, in: 2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), IEEE, 2012, pp. 21–26.

[12] P. Cai, S. Wang, Y. Sun, M. Liu, Probabilistic end-to-end vehicle navigation in complex dynamic environments with multimodal sensor fusion, IEEE Robotics and Automation Letters 5 (2020) 4218–4224.

[13] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[14] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[15] B. Rouet-Leduc, C. Hulbert, N. Lubbers, K. Barros, C. J. Humphreys, P. A. Johnson, Machine learning predicts laboratory earthquakes, Geophysical Research Letters 44 (2017) 9276–9282.

[16] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016).

[17] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, et al., Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 4779–4783.

[18] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al., Tacotron: Towards end-to-end speech synthesis, arXiv preprint arXiv:1703.10135 (2017).

[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[22] M.-T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, arXiv preprint arXiv:1508.04025 (2015).

[23] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017. arXiv:1412.6980.

[24] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting, in: Advances in Neural Information Processing Systems, 2021.

[25] M. S. Louis, Z. Azad, L. Delshadtehrani, S. Gupta, P. Warden, V. J. Reddi, A. Joshi, Towards deep learning using tensorflow lite on risc-v, in: Third Workshop on Computer Architecture Research with RISC-V (CARRV), volume 1, 2019, p. 6.

[26] V. Gómez, A. Maravall, Estimation, prediction, and interpolation for nonstationary series with the kalman filter, Journal of the American Statistical Association 89 (1994) 611–624.

[27] N. I. Sapankevych, R. Sankar, Time series prediction using support vector machines: a survey, IEEE Computational Intelligence Magazine 4 (2009) 24–38.

[28] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long-and short-term temporal patterns with deep neural networks, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.

[29] S.-Y. Shih, F.-K. Sun, H.-y. Lee, Temporal pattern attention for multivariate time series forecasting, Machine Learning 108 (2019) 1421–1441.

[30] C. Lea, R. Vidal, A. Reiter, G. D. Hager, Temporal convolutional networks: A unified approach to action segmentation, in: European Conference on Computer Vision, Springer, 2016, pp. 47–54.

[31] J. Yan, L. Mu, L. Wang, R. Ranjan, A. Y. Zomaya, Temporal convolutional networks for the advance prediction of enso, Scientific reports 10 (2020) 1–15.

[32] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: Multivariate time series forecasting with graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 753–763.

[33] B. Lim, S. Ö. Arık, N. Loeff, T. Pfister, Temporal fusion transformers for interpretable multi-horizon time series forecasting, International Journal of Forecasting (2021).