DEEP PARTICULATE MATTER FORECASTING MODEL USING CORRENTROPY-INDUCED LOSS

A PREPRINT

Jongsu Kim
School of Mathematics and Computing
Yonsei University
Seoul, 03722
skyobserver@yonsei.ac.kr

Changhoon Lee
Department of Mechanical Engineering
Yonsei University
Seoul, 03722
clee@yonsei.ac.kr

August 31, 2021

ABSTRACT

Forecasting the particulate matter (PM) concentration in South Korea has become urgently necessary owing to its strong negative impact on human life. In most statistical or machine learning methods, independent and identically distributed data, for example, a Gaussian distribution, are assumed; however, time series such as air pollution and weather data do not meet this assumption. In this study, the maximum correntropy criterion for regression (MCCR) loss is used in an analysis of the statistical characteristics of air pollution and weather data. Rigorous seasonality adjustment of the air pollution and weather data was performed because of their complex seasonality patterns and the heavy-tailed distribution of data even after deseasonalization. The MCCR loss was applied to multiple models including conventional statistical models and state-of-the-art machine learning models. The results show that the MCCR loss is more appropriate than the conventional mean squared error loss for forecasting extreme values.

Keywords: Air Pollution · Deep Learning · Particulate Matter · Time Series Forecasting · Heavy-Tailed Distribution · Correntropy

1 Introduction

In recent years, air pollution has become a serious problem in South Korea. Among air pollutants, particulate matter (PM) is the greatest concern. PM\textsubscript{10} (≤ 10 µm in size) and PM\textsubscript{2.5} (≤ 2.5 µm in size) are widely known to cause health problems such as cardiovascular and/or respiratory diseases [EPA [2016], Kim et al. [2018], Pope and Dockery [2006]]. At high concentrations, PM\textsubscript{2.5} not only causes health problems, but also reduces agricultural productivity and increases socioeconomic costs [OECD [2016]]. Therefore, it is necessary to develop a timely and accurate air pollution forecasting model that can operate in dynamic environments.

In South Korea, the yearly average concentrations of PM\textsubscript{10} and PM\textsubscript{2.5} are 41 and 23 µg m\textsuperscript{-3} [AirKorea [2019]], which exceed the WHO air quality guidelines of 20 and 10 µg m\textsuperscript{-3} [World Health Organization [2006]]. In addition, the yearly average concentration of PM\textsubscript{2.5} is the highest among 37 OECD countries and is almost twice the average value for OECD countries, 13.93 µg m\textsuperscript{-3} [OECD [2021]].

Despite the seriousness of the situation, there are many obstacles to the forecasting of PM\textsubscript{10} and PM\textsubscript{2.5}. First, the concentrations of PM\textsubscript{10} and PM\textsubscript{2.5} are affected by various factors, such as anthropogenic emissions and meteorological parameters [Makar et al. [2015]]. It is impossible to parameterize all of these factors, especially in real time: moreover, even though it is possible for some factors, there is still considerable uncertainty in the models and observation data [Bouarar et al. [2019]]. Second, spatiotemporal air pollution data for South Korea are limited. South Korea is located in Northeast Asia, which has undergone rapid industrialization, and it remains one of the most severely polluted regions worldwide despite numerous efforts to address this issue [OECD [2016]]. There is an urgent demand for the development of a method of forecasting the spatiotemporal concentrations of PM\textsubscript{10} and PM\textsubscript{2.5}. Moreover, recent studies showed that...
the amount of PM$_{10}$ and PM$_{2.5}$ contributed by domestic sources is also significant [Lee et al., 2019, Jordan et al., 2020], and the quantitative effect of air pollution from China is still controversial.

Many researchers have proposed various forecasting models to predict PM concentrations accurately. Deterministic and statistical methods are generally used for PM$_{10}$ and PM$_{2.5}$ prediction [Singh et al., 2012, Zhang et al., 2012]. In a deterministic model, physical or chemical equations are solved to model the creation and transport of pollutants. The most common deterministic models are CMAQ [Byun and Ching, 1999], GEOS-Chem [Bey et al., 2001], and WRF/Chem [Grell et al., 2005]. This approach has achieved great success because the models yield consistent and accurate results. However, they require extensive observation data, an accurate understanding of the physical interactions between inputs, the correct representation of meteorological processes, and appropriate assumptions regarding default parameters, in particular anthropogenic emissions. They are computationally expensive, and thus a supercomputing system is needed for daily forecasting [Baklanov et al., 2008, Zhou et al., 2017].

By contrast, statistical models have been developed to overcome these disadvantages of deterministic models. For example, classification and regression trees (CART) using decision tree [Shang et al., 2019] and fuzzy logic for parameter classification [Guo et al., 2007] are used to forecast air pollution statistically. Time series forecasting models such as autoregressive integrated moving average (ARIMA) models are also applied in air pollution prediction [Wang and Guo, 2009]. These models are supported by a well-established statistical and theoretical background and are sometimes more accurate for site-specific prediction. However, a major limitation of statistical models is that they cannot represent physical or chemical processes directly; thus, their results are sometimes unreliable.

The great success of deep learning models has attracted much attention recently. It has been proved that deep learning models can be applied to not only image processing and natural language processing, but also time series forecasting [Liu et al., 2018, Shi et al., 2019, Li et al., 2019]. On the basis of advances in deep learning methodology, several attempts have been made to apply deep learning models to air pollution prediction modeling [Choi et al., 2018, Cho et al., 2019, Franceschi et al., 2018, Bai et al., 2019]. Although these studies showed promising results, they focused on short-term prediction; consequently, long-term (≥ 12 h) results are uncertain. In addition, they used various architectures based on a multilayer per-cetron (MLP) or long short-term memory (LSTM), which cannot easily capture complex patterns in data. The increasing complexity of the models has been the subject of much research in recent years. Unfortunately, the importance of the data distribution is sometimes ignored. Deep learning models assume independent and identically distributed (i.i.d.) random variables or a Gaussian distribution; however, real-world time series data are frequently dependent data and have a non-Gaussian distribution.

To reflect the data distribution and capture complex nonlinear patterns, a model with maximum correntropy criterion induced losses [Liu et al., 2007, Feng et al., 2015] is proposed in this study. Environmental data, including air pollution and weather data, are known to have long-term seasonal variation and heavy-tailed distributions [Cichowicz et al., 2017]. We take into account the non-Gaussian characteristics of the data and develop a methodology that reflects these characteristics and offers improved prediction accuracy compared to other forecasting methods. Various machine learning and statistical models are tested. In addition, the model performance for real-world data is analyzed, and their limitations are also investigated. The data preprocessing and data collection are described in detail in Section 2. The methods of the machine learning models are explained in detail in Section 3, and Section 4 presents the model results and analysis. Conclusions are presented in Section 5.

2 Materials Analysis

2.1 Data Collection and Study Area

Two types of observation data, air quality data and weather data, are selected as explanatory variables, and PM$_{10}$ and PM$_{2.5}$ are selected as the targets of our model. Air quality data were collected in Jongno District in the central region of Seoul, the capital of South Korea. Jongno District is chosen as the target site because it is located in the center of Seoul. Weather data were measured at Seoul weather station, where the Jongno air quality observatory is located. Air quality data are publicly available at Airkorea (https://www.airkorea.or.kr/eng), and weather data are available from the Korea Meteorological Administration (https://data.kma.go.kr). Air quality data for Seoul from 2008 to 2014 are available only from the Seoul metropolitan government by request. The annual mean temperature and precipitation of Seoul from 1980 to 2010 were 12.5°C and 1450mm. However, the monthly average temperature varies from -2.4°C to 25.7°C, and 61% of the annual precipitation falls in three months (June, July, and August). South Korea is in the mid-latitudes of the Northern Hemisphere and is affected by seasonal prevailing surface winds in the southwesterly and northwesterly directions in summer and winter, respectively. The seasonal variation in air quality and weather data has been extensively analyzed [Cichowicz et al., 2017].
Deep particulate matter forecasting model using correntropy-induced loss  

Table 1: Feature information

| Input parameters | Unit       | Category            |
|------------------|------------|---------------------|
| SO₂              | ppm        | Air Quality         |
| CO               | ppm        |                      |
| NO₂              | ppm        |                      |
| PM₁₀             | µg m⁻³     |                      |
| PM₂.₅            | µg m⁻³     |                      |
| Temperature      | °C         | Weather             |
| Wind speed       | m s⁻¹      |                     |
| Wind direction (sin) |          |                     |
| Wind direction (cos) |         |                     |
| Ground level pressure | hPa     |                     |
| Relative humidity | %          |                     |
| Precipitation    | mm         |                     |

| Output parameters | Unit       | Category            |
|-------------------|------------|---------------------|
| PM₁₀              | µg m⁻³     | Air Quality         |
| PM₂.₅             | µg m⁻³     |                      |

The total data set covers January 2008 to October 2020 with hourly temporal resolution. The training set data cover January 2008 to December 2018, and the test set covers January 2019 to October 2020. All missing data were interpolated using the k-nearest neighbor algorithm implemented in scikit-learn [Troyanskaya et al., 2001, Pedregosa et al., 2011]. The wind direction in the raw data is given using 16 cardinal directions, which were converted to degrees and encoded using a trigonometric function because of the cyclical nature of the data. The input data are standardized without the standard deviation after seasonal adjustment, as described in Section 2.2.

2.2 Seasonal Adjustment of Data

Most time series are composed of signal and noise. Time series forecasting models are intended to capture the signal obscured by noise. A time series model learns signals from the past under the assumption that the variables are i.i.d., which introduces mixing and stationarity. Mixing represents asymptotic independence; that is, time series dependence between lags goes to zero as the lag increases. Stationarity indicates that random variables are identically distributed. The statistical properties of stationary time series such as mean, variance, and autocorrelation are invariant over time. However, time series are serially dependent by nature; thus, the assumptions of stationarity and mixing are, strictly speaking, impossible. Therefore, the effective sample size, which represents the same statistical quantity even for a smaller sample size, ust be determined by investigating the autocorrelation.

If a time series has a trend or seasonality, it is not considered a stationary time series. As noted in Section 1, environmental variables, air pollution, and weather variables exhibit seasonal variation. In this section, we describe how the seasonality was removed from the observation data and examine the stationarity.

Seasonal adjustment, or deseasonalization, is described by the following equation.

\[ x(t) \approx s_y,\text{smoothed} + s_w + s_h + \text{res}_h \]  

(1)

where original series \( x(t) \) is approximated by the sum of the smoothed yearly seasonality \( s_y,\text{smoothed} \), weekly variation \( s_w \), daily variation \( s_h \), and residual \( \text{res}_h \). The seasonal adjustment procedure is as follows. First, the yearly and weekly seasonality is identified by averaging the daily averaged data over 11 years for each day of the year and for each weekday of the week, respectively. The yearly seasonality is then smoothed because a single extreme value may overrepresent seasonality. Second, the day-scale smoothed yearly and weekly seasonality are subtracted from the hour-scale original data. Finally, the daily seasonality and residuals are computed. For the yearly seasonality, smoothing is done by locally weighted scatterplot smoothing [Cleveland, 1979]. Figure 1 presents a sample result of seasonality adjustment of the target variables, PM₁₀ and PM₂.₅. Figure 1B, C, D, and G, H, I correspond to \( s_y,\text{smoothed} \), \( s_w \), and \( s_h \), respectively, whereas Figure 1A, F and E, J show the original value \( x(t) \) and residual \( r \). The models used here, except for the OU process, accept input from residuals with standardization, as mentioned in Section 2.1.
Deep particulate matter forecasting model using correntropy-induced loss

The most common way to determine stationarity is by examining the autocorrelation function (ACF), which quantifies the linear interdependence between two instances of time series $x(t)$ separated by a discrete time lag $r (r > 0)$:

$$C(r) = \frac{\langle x(t)x(t+r) \rangle}{\langle x(t)^2 \rangle}$$  \hspace{1cm} (2)

If $x(t)$ and $x(t + r)$ are uncorrelated, the correlation $C(r)$ becomes zero. If two instances of a time series have a short-range correlation, the dependence is characterized by rapid decay of the ACF, $C(s) \propto e^{-r/T}$, where $T$ is the decay time or decorrelation time; that is, $C(r)$ decreases to zero within a certain time lag $s^*$.

$$C(r) \propto e^{-r/T}$$  \hspace{1cm} (3)

For long-range dependence, the autocorrelation $C(r)$ is positive even for a large time lag $r$ and follows the power law

$$C(r) \propto r^{-\xi} \quad 0 < \xi < 1$$  \hspace{1cm} (4)

If $\xi > 1$ the series is considered to have short-range dependence. These facts depend on the convergence of the mean correlation distance, $\bar{r} = \int_0^\infty C(r)dr$, to a finite value.

As shown in Figure 2, the ACF without seasonal adjustment shows a decay to nonzero autocorrelation and an hourly seasonal pattern. As mentioned in Section 1, environmental data have long-range dependence, exhibiting not only a single seasonal pattern, but also multiple complex seasonal patterns. PM$_{10}$, PM$_{2.5}$ and the other variables (except precipitation, which more closely resembles a sparse discrete event), also have similar weekly and yearly patterns. They were preprocessed according to the procedure for PM$_{10}$ and PM$_{2.5}$ before training.

After seasonal adjustment, the ACFs are closer to the confidence interval than those without seasonal adjustment. However, small-scale seasonal patterns remain. Those patterns are trained using time2vec encoding, as discussed in Section 3.3.

Quantitative analysis of the deseasonalized data was performed using detrended fluctuation analysis (DFA)\cite{Kantelhardt2001, Koscielny-Bunde2006}. DFA consists of four steps. First, the profile $Z(i)$ is defined by subtracting the mean of $x$, from $x_t$.

$$Z(i) \equiv \sum_{t=1}^{i} x_t - \langle x \rangle$$  \hspace{1cm} (5)

Figure 1: Seasonality and residuals of PM$_{10}$ and PM$_{2.5}$. A, F: raw data, B, G: yearly seasonality (red and blue represent seasonality with and without smoothing, respectively), C, H: weekly seasonality, D, I: daily seasonality, E, J: residuals
Second, the profile $Z(i)$ is divided into non-overlapping segments with size $s$. Then, the detrended series $Z_s(i)$ of each segment $m$ is computed by subtracting the local trends in the $m$th segment, $p_m(i)$, from the original series $Z(i)$:

$$Z_s(i) = Z(i) - p_m(i).$$

The local trend $p_m(i)$ is the fitting polynomial for an $n$th-order least-squares fit; $n = 2$ was used in this study. Finally, the variance function from the detrended series $Z_s(i)$ is defined for the $m$th segment, where $m = 1, \ldots, 2N_s$. In this study, quadratic polynomials ($n = 2$) are used for fitting.

$$V_s^2(m) = \frac{1}{s} \sum_{i=0}^{s} Z_s^2[(m-1)s + i]$$

(6)

and the mean fluctuation function is

$$V(s) = \left[ \frac{1}{2N_s} \sum_{m=1}^{2N_s} V_s^2(m) \right]$$

(7)

If a time series has long-range dependence, the DFA fluctuation function $V(s)$ increases for large segment size $s$ and follows another power law based on $\xi$ in Equation 4

$$V(s) \sim s^{\xi}$$

(8)

with
In Figure 3, the DFA fluctuation function is fitted using $s^h$ for large $s$. Furthermore, the fluctuation exponent $h$ and the corresponding correlation exponent $\xi$ are presented in Table 2. These two parameters approximately satisfy the condition given in [Equation 7]. The $h$ values of the deseasonalized data (Figure 3B and D) are approximately 0.5, and the $\xi$ are greater than 1, indicating short-range dependence. In conclusion, the seasonal adjustment used in this study was quantitatively proven to remove the long-range dependence of the raw data.

### Table 2: $h$ and $\xi$ values from DFA results.

|          | $h$   | $\xi$   | $1 - \xi$ |
|----------|-------|---------|-----------|
| PM$_{10}$ |       |         |           |
| With Seasonality (A) | 0.85  | 0.31    | 0.69      |
| With Seasonality (B) | 0.41  | 1.18    |           |
| PM$_{2.5}$ |       |         |           |
| With Seasonality (C) | 0.89  | 0.23    | 0.77      |
| With Seasonality (D) | 0.44  | 1.11    |           |

$h = \begin{cases} 1 - \xi & \text{for } 0 < \xi < 1 \\ 1/2 & \text{for } \xi \geq 1 \end{cases}$

2.3 Heavy-Tailed Distribution

The classical statistical modeling of time series usually describes noise as i.i.d. random variables. However, in the real world, for example, in economics, computer science, and the environment, noise has a heavy-tailed distribution,
where the tail is heavier than that of an exponential distribution. A heavy-tailed distribution is problematic because the sampling errors are large and negatively biased. When heavy-tailed data are used, the error distribution of the model can be any distribution except the normal distribution without affecting model performance. It is clearly important to understand the data distribution before training a model.

Although the most typical example of a heavy-tailed distribution is the power law distribution (Pareto distribution), the log-normal distribution should also be considered as an empirical data distribution (Clauset et al., 2009). A complementary cumulative distribution function (CCDF) \( \bar{F}(x) = 1 - F(x) \), where \( F(x) \) is the cumulative distribution function, is generally used to determine the distribution of heavy-tailed data. The CCDF of the power-law distribution is given by the following equation.

\[
\bar{F}(x) = \Pr(X > x) = \begin{cases} 
\left( \frac{x_m}{x} \right)^\alpha & x \geq x_m \\
1 & x < x_m
\end{cases}
\]  

where \( x_m \) is a scale parameter, and \( \alpha \) is the shape parameter, or tail index. The Pareto distribution is used to describe self-similar processes owing to its scale invariance and the distribution of welfare in economics. The log-normal distribution is a continuous probability distribution, where the logarithm of random variables follows a normal distribution. In a log-normal distribution, multiplication can be described as taking the product of positive random variables. The CCDF of the log-normal distribution is like that of the normal distribution, where \( \mu \) and \( \sigma \) are the mean and standard deviation of distribution, respectively.

\[
\bar{F}(x) = \frac{1}{2} \, \text{erf} \left( \frac{\ln x - \mu}{\sqrt{2} \sigma} \right)
\]
Table 3: Methodologies

| Category        | Type            | Methods                                      |
|-----------------|-----------------|----------------------------------------------|
| Univariate      | Statistical     | Ornstein-Uhlenbeck process                  |
|                 |                 | ARIMA                                        |
|                 | Machine Learning| Multi Layer Perceptron                      |
|                 |                 | Attention (Bahdanau)                        |
|                 | Multivariate    | XGBoost                                      |
|                 | Machine Learning| Multi Layer Perceptron                      |
|                 |                 | LSTNet (Skip Layer)                         |
|                 |                 | Time Series Transformer                     |

Power-law and log-normal distributions were fitted to the data with and without seasonal adjustment. In Figure 4, the CCDF is plotted with positive data and fitted by the log-normal and power-law distributions. The powerlaw package (a Python package for analyzing heavy-tailed data distribution) was used for the fitting [2009], [2014]. This result shows that positive input data follow a log-normal distribution rather than a power-law distribution. Moreover, owing to the characteristics of the distribution, the model accuracy clearly is not affected by seasonal adjustment.

This analysis clearly shows that the data used in this study follow a heavy-tailed distribution but not a power-law distribution. As shown in Figure 4 the distribution of the PM$_{10}$ data is better fitted by the heavy tailed distribution than that of the PM$_{2.5}$ data. To obtain better performance, the forecasting model should reflect the distribution. Note also that the data used in the figure are filtered positive data, although the seasonal adjustment introduces negative data naturally owing to subtraction. Other regression methods based on the multiplication of random variables cannot be applied owing to the negativity of the data and the nature of the logarithm.

3 Method

Most of the methods used in this study are divided into two categories, univariate and multivariate methods, according to the number of input parameters. These methods are selected to compare the effect of the number of input variables on the prediction results and address whether state-of-the-art methods are effective. The methods are also categorized according to the methodological approach as statistical models or machine learning models. Two statistical models are considered: the Ornstein–Uhlenbeck (OU) process model, which assumes that PM$_{10}$ and PM$_{2.5}$ are stationary stochastic processes [1930], [2012], and the ARIMA model, which is a state-space model [2018]. The machine learning models are the XGBoost model, which uses decision-tree-based ensemble machine learning [2016]; the MLP model, a fully connected hierarchical neural network [2015], [1998]; an attention model based on a recurrent neural network (RNN) [1997], [2013], [2015]; and two more complex deep learning models, the LSTNet and Time Series Transformer (TST) models [2018], [2020], [2020]. The methods are summarized in Table 3.

3.1 Univariate Models

The univariate statistical methods used in this study are the Gaussian process model and ARIMA model. The Gaussian process model is based on the OU process [1930], [2012], which evolves according to the following equation from the initial condition $x_0$:

$$dx = \frac{\mu - x}{\tau} dt + \sqrt{\frac{2\sigma^2}{\tau}} dW_t$$

(11)

where $W_t$ denotes the standard Wiener process, and $\mu$ and $\sigma$ are mean and standard deviation of the process, respectively. The correlation time scale $\tau$ is computed by integrating the ACF to the confidence interval cutoff in Figure 2. The OU process is derived assuming that the time series is a simple stochastic Gauss–Markov process. The following parameters
are used to predict the PM$_{10}$ and PM$_{2.5}$ concentrations: mean $\mu$ values of -9.47 and -1.99, stationary variance $\sigma^2$ values of 458.39 and 215.15, and correlational time scales $\tau$ of 19.02 and 20.41, respectively. The stochastic differential equation [Equation 11] was numerically solved using the Euler–Maruyama scheme Bayram et al. [2018].

The ARIMA model is a widely used statistical model for time series analysis. It is a linear combination of the AR model, which used the number of lags of dependent variables; the MA model, which uses the number of lags of the forecast errors; and an integrated component, which uses differenced time series to obtain stationary time series. It is a generalized version of the autoregressive moving average model. As described in Section 2.2, the time series data used in this study become stationary; therefore, the integration component can be ignored. The strategy for determining the order of the AR and MA terms using the ACF and partial autocorrelation function is described in Hyndman and Athanasopoulos [2018]. The resulting equation for the AR($p$) model, where $p$ is the order of the autoregressive term, is

$$Y_t = c + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p}$$

The coefficients and constant of the AR model, $\phi_i$ and $c$, are estimated using conditional and exact maximum likelihood and conditional least-squares, respectively, as implemented in stats-models (a Python package for statistical models) using the training set data Seabold and Perktold [2010].

The MLP and RNN-based attention mechanism models are used as univariate machine learning models. The MLP model is a general class of feedforward artificial neural network. MLP models show promising results for forecasting because they learn adaptively to be capable of flexible nonlinear modeling Zhang et al. [1998]. In an MLP model, all input data are serialized in one layer, and hidden layers are distributed in multiple layers. Each node receives an input signal as information, transforms it by an activation function, processes it by multiplying the local weight of each node, and then passes it to the output. Learning and changing the weights in the perceptron is performed after all the data are processed using the error of the output through back-propagation. Leaky ReLU is used as the activation function.

A disadvantage of MLP models is that they cannot handle the temporal information of the data because each node processes data independently. The RNN is designed for problems involving temporal sequences, such as machine translation. RNN models have an internal state to process input data depending on its shape and the dimensions of the model. Because an RNN uses repeated multiplication of the recurrent weight matrix, it has an exploding/vanishing gradient problem and cannot easily access very old information. LSTM and the gated recurrent unit (GRU) have been developed to resolve this problem by adding multiple gates in the internal state to store long-term information Hochreiter and Schmidhuber [1997], Cho et al. [2014]. At each LSTM or GRU cell, they learn what to learn and what to forget from a single input. Their output is called the hidden state from last cell, and it can be fed into the next cell recursively.

Another improvement in machine translation resulting from the improved single-node architecture mentioned above is the attention mechanism Bahdanau et al. [2015]. This mechanism is based on a sequence-to-sequence (S2S) technique, which translates input to output by an encoder-decoder paradigm Sutskever et al. [2014]. The encoder consumes input data and compresses them with a single latent vector; then the decoder learns to produce output data from the latent vector. This technique makes input-output translation possible in domain-to-domain mapping rather than one-to-one. Although the S2S technique translates domain-to-domain mapping, the encoder compresses the whole input data into a single latent vector. Consequently, some information may be lost during compression. In addition, the decoder also has a problem: it treats all input sequences as having the same importance because its input is the single latent vector generated by the encoder. The importance of each node in the sequence may vary, and this variation must be assessed. The attention mechanism “attends” to the input sequence when the decoder generates data. As shown in Figure 5, the alignment score is calculated using a combination of the output of each encoder, the decoder input (context vector), or the last hidden state at each decoder step. The alignment score quantifies how the model attends to its input by choosing multiple methods such as concatenation or taking the dot product. In this study, a GRU based on the concatenation score method was chosen to decrease the learning time and increase the accuracy.

### 3.2 Multivariate Models

Since the environmental variables depend on multiple factors simultaneously, a univariate model may not be sufficient for PM$_{10}$ and PM$_{2.5}$ forecasting. Multivariate models are suggested to resolve the problem. In multivariate models, it is important to capture nonlinear patterns between variables even when a variable itself exhibits a temporal pattern.

The multivariate MLP model architecture is identical to that of a univariate model, but the input data are serialized as one-dimensional. XGBoost is a fast and performant gradient boosting decision-tree model. A decision tree model constructs a probability-based tree, and prediction is performed by proceeding from a node (branch) to the target (leaves). A single decision tree is vulnerable to overfitting due to a single path to a leaf and is known as a weak learner.
Context Vector 𝑦₁, 𝑦₂, 𝑦₃, 𝑦₄
Encoder Output

Score

Figure 5: Schematic diagram of attention mechanism. Each output sequence \{𝑦₁, ⋯, 𝑦₄\} is generated by a hidden state starting from a single context vector from input sequence \{𝑥₁, ⋯, 𝑥₅\} and an attention vector. The blue-shaded vectors indicate how much the input sequence “attends” to the output sequence.

That is only slightly better than random chance. XGBoost is an ensemble model using a gradient boosting strategy with CART, and it combines multiple decision trees with weight and loss function optimization [Chen and Guestrin 2016]. It has rapidly become popular because it is faster than existing gradient-boosting implementations and is memory efficient.

LSTNet is a multivariate time series model combining a convolutional component with a convolutional neural network (CNN) without pooling and a recurrent component with a GRU [Lai et al. 2018]. In the convolutional component, the CNN is used to extract patterns involving multiple features and short-term temporal patterns. The result of the convolutional component is processed by the recurrent component to capture longer-term patterns. In LSTNet, recurrent-skip and autoregressive components are also added to memorize periodic patterns and the scale of the input. Although LSTNet outperforms other models on benchmark datasets (traffic, solar energy, and electricity datasets), which show highly periodic patterns even in input length, but not on an exchange rate dataset, which shows only nonperiodic patterns. This result indicates that more complex models are needed.

The attention mechanism allows a model to attend to a specific part of the input when producing output. Although it yields better machine translation results than typical RNN models because it enables the translation of longer sequences, the importance of differences in attention between input sequences is increased when meaning is determined from the input itself, for example, in the translation of pronouns in a sentence. Self-attention, or intra-attention, is used to extract different features within an input sequence. Furthermore, RNN-based models are vulnerable to problems with longer sequences because of the exploding/vanishing gradient problem, and they cannot be parallelized. Therefore, a self-attention-based transformer model was introduced to use attention instead of a recurrent unit [Vaswani et al. 2017]. The transformer outperforms existing models; moreover, it enables the parallelization of the model because it uses feed-forward layers instead of a recurrent network unit, which can only be processed in serial order. Owing to this success, the transformer has also been applied in time series forecasting and showed promising results [Wu et al. 2020, Zerveas et al. 2020]. The TST, an encoder-based transformer architecture [Zerveas et al. 2020], is used in this study because of its simplicity and performance. Even though the TST was not designed for forecasting problems, its encoder architecture is applicable to these problems. Here, we improved the TST model, as explained in the next section.
3.3 Proposed Models

When the given time series window is \( X_T = \{x_1, x_2, \ldots, x_T\} \), where \( x_t \in \mathbb{R}^n \), and \( n \) is the number of variables, the forecasting task for output horizon \( h \) is to predict \( Y_T = \{y_{T+1}, y_{T+2}, \ldots, y_{T+h}\} \). In this study, \( Y_T \) is a single variable, PM\(_{10}\) or PM\(_{2.5}\), and the output matrix is \( Y_T \in \mathbb{R}^n \), where \( h \) is the output horizon size, and the input matrix is \( X \in \mathbb{R}^{k \times T} \).

Most machine-learning-based models are specialized to capture nonlinear patterns in time series. We also added a learnable autoregressive component with a single fully connected layer, as in the LSTNet model.

\[
\hat{Y}_t = AR(X_{1d}^T) + \text{model}(X_T) \tag{13}
\]

where \( X_{1d}^T \) is one-dimensional target variable, and \( X_T \) is a multivariate whole sliding window input.

The overall architecture of the proposed model of the encoder-only transformer is the same as that in Zerveas et al. [2020]. The positional encoding, however, is replaced with Time2Vec [Kazemi et al. [2019], a learnable parametric positional encoding using a sine function, because periodic patterns remain, as shown in Figure 2. Time2Vec consists of the following two components.

\[
t_{2v}(\tau)[i] = \begin{cases} 
\omega_i \tau + \phi_i & \text{if } i = 0 \\
F(\omega_i \tau + \phi_i) & \text{if } 1 \leq i \leq k
\end{cases} \tag{14}
\]

where \( t_{2v}(\tau)[i] \) is the \( i^{th} \) element of \( t_{2v} \), \( F \) is a periodic activation function (sine function), and \( \omega_i \) and \( \phi_i \) are learnable parameters such as the period and phase shift in a periodic function. If the length of \( t_{2v} \) is 1, \( -i = 0 \), it is the same as the learnable positional encoding proposed in Zerveas et al. [2020]. The improved architecture is illustrated in Fig. 6.

It has been argued Zerveas et al. [2020] that batch normalization between every self-attention layer and feed-forward layer results in better performance than default layer normalization Vaswani et al. [2017]. However, a model with batch normalization was unstable and unsuitable for our data; thus, we applied layer normalization instead. The code used in this study is publicly available at GitHub (https://github.com/appleparan/mise.py).
Deep particulate matter forecasting model using correntropy-induced loss

A P

Figure 7: Actual and predicted values of MCCR for different $\beta$ corresponding to $Y_t - \hat{Y}_t$

3.4 Maximum Correntropy Criterion Induced Losses for Regression

The loss function is an objective function between the observed and predicted values. Most deep learning algorithms involve optimization problems in which a certain loss function is minimized or maximized. The most common loss function used in regression problems is the mean squared error (MSE) loss.

$$\text{MSE}(Y_t, \hat{Y}_t) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$ (15)

The MSE loss, however, tends to suffer from extreme values, which is common for heavy-tailed distributions because the error is squared. Although numerous efforts have been made to predict extreme values (for example, the use of a memory network module and a variable loss function Ding et al. [2019], Ribeiro and Moniz [2020]), these methods require a long training time because they need to access a different dataset for each batch.

It is necessary to consider the data distribution when training models based on a heavy-tailed distribution. The Kullback–Leibler (KL) divergence, $D_{KL}(P \parallel Q) = \sum_x P(x) \log P(x)/Q(x)$, is commonly used to compare the distributions of random variables $P$ and $Q$. In Qi and Majda [2020], the KL divergence is used as a loss function to predict extreme values in terms of the difference between the probability distributions in complex systems, such as turbulence modeling. It is simple to implement and can be trained more rapidly than previous functions, but as the authors of Qi and Majda [2020] mentioned, they are interested in statistical features rather than the exact trajectory of the system, and small shifts in extreme values are negligible for their purpose.

To overcome the limitations of the KL divergence, the correntropy has been proposed as a robust localized similarity measure Liu et al. [2007]. The correntropy generalizes the correlation by nonlinear mapping of random variables and measures the similarity between random variables by information theoretic learning. The maximum correntropy criterion has been used as a loss function and applied to regression problems Liu et al. [2007], Feng et al. [2015]. The maximum correntropy criterion based regression (MCCR) loss is denoted as

$$l_\beta(Y_t, \hat{Y}_t) = \beta^2 (1 - e^{-\frac{(Y_t - \hat{Y}_t)^2}{\beta^2}})$$ (16)

where $\beta$ is a positive scale parameter, and $Y_t$ and $\hat{Y}_t$ are the actual and predicted values, respectively. MCCR is plotted for various $\beta$ values in Figure 7. As suggested by Feng et al. [2015, Theorem 8], the MCCR model and least square model are equivalent if a sufficiently large $\beta$ is chosen. The authors of Feng et al. [2015] also proved that the MCCR can model heavy-tailed noise, whereas the conventional least square model handles only sub-Gaussian noise. For this reason, and owing to the data distribution used in this study, the MCCR loss is used here, and the results are compared with those of the MSE loss.

As noted in the explanation of the OU process, the confidence interval cutoff, or correlation timescale, of PM$_{10}$ and PM$_{2.5}$ is approximately 20. For training a small periodic pattern using $t2v$, the minimum input size should be at least 48; thus, this value was chosen.
4 Results

Cross-validation is another important topic in time series forecasting because of the dependence of the sliding window on the time series. Typical cross-validation splitting methods such as random splitting may cause overfitting between the training, validation, and test sets. To avoid this problem, blocked cross-validation was used, as shown in Figure 8. The training and validation sets are mixtures of multiple seasons for proper training of the seasonal variation. The ratio between the training, validation, and test sets is approximately 67%/20%/13%.

For multistep horizon prediction, a multiple-input multiple-output strategy is applied to most machine learning models, except for the OU process and XGBoost. Because the OU process is naturally unable to produce multiple outputs at once, a recursive strategy of iterative forecasting was applied. XGBoost is also unable to consume a multivariate sliding window and produce multiple outputs; thus, a single-input single-output strategy was applied by creating multiple models for each horizon length and mapping the last step of the input to each model.

4.1 Hyperparameter Optimization

For hyperparameter optimization, Bayesian optimization was used in the form of the Tree-structured Parzen Estimator sampler implemented in Optuna (a hyperparameter tuning framework) [Akiba et al., 2019]. Table 4 lists the optimized scaling parameters of the MCCR loss. Some parameters are excluded from the optimization, such as the sliding window input length, output horizon size, and batch size, which are 48, 24, and 64 h, respectively. For weight and bias optimization, Adam was used with a learning rate of $10^{-4}$ [Kingma and Ba, 2015]. Ridge regularization (L2), the most popular regularization technique for time series forecasting, was used. PyTorch is used to train the deep learning networks [Paszke et al., 2019].

Optuna selects hyperparameter by sampler and determines whether it is worth to run trial by running single epoch. If not, Optuna drops the selected hyperparameter and goes forward to next trial, which is called pruning. Because of pruning, hyperparameter optimization time varies by case. Some initial trials have fixed parameters with uniform intervals to avoid falling local minimum and each trial runs 20 epochs. After optimizing hyperparameters, our models are trained by best hyperparameter set up to 500 epochs. TST model have 160 maximum trials which is the largest among our models. Usually, TST model took a day for optimization, 4 hours for training, few minutes for testing. The models are trained and tested using NVIDIA TITAN Xp GPU and Intel Xeon Gold 6140 CPU.

The deterministic models depend on other models output such as weather and emission models, which also require extensive computing resources. In [Myoung et al., 2018], they indicate that national air quality forecasting system runs weather, emission, and air quality models and takes 3 hour 40 minutes even without preprocessing and postprocessing while our models need few minutes for forecasting with trained weights. In South Korea, air quality forecasting is done by every 6 hours and models should run on every prediction. Statistical models or machine learning models reuse trained parameters thus they can save a lot of computing resources.
Table 4: Optimized scaling parameter $\alpha$ of MCCR loss

|          | Univariate | Multi Layer Perceptron | 4.05 | Attention | 4.45 |
|----------|------------|------------------------|------|-----------|------|
| $PM_{10}$ | Multi Layer Perceptron | 4.60                     |      | LSTNet (Skip) | 2.60 |
|          | Time Series Transformer | 4.10                     |      |           |      |
|          | Univariate | Multi Layer Perceptron | 4.65 | Attention | 4.15 |
| $PM_{2.5}$ | Multi Layer Perceptron | 7.60                     |      | LSTNet (Skip) | 2.10 |
|          | Time Series Transformer | 4.10                     |      |           |      |

4.2 Evaluation Metrics

It is important to choose the best metric for evaluating models. The most widely used evaluation metrics for regression problems are the MSE, mean absolute error, and correlation coefficient (CORR).

Despite the simplicity and intuitive interpretation of those metrics, they are overly influenced by even a few extremely large values. Relative difference of evaluation metrics, such as the normalized MSE and normalized MAE are alternatives, but they are historically normalized by observed values; consequently, the metric is inflated if the observations are small values. Several metrics have been proposed for intuitive, symmetric, and unbiased model performance evaluation [Yu et al. [2006]], such as the normalized mean bias factor (NMBF) and normalized mean absolute error factor (NMAEF).

\[
NMBF = \begin{cases} 
\frac{\bar{M} - 1}{\bar{O}} & \text{if } \bar{M} \geq \bar{O} \\
1 - \frac{\bar{O}}{\bar{M}} & \text{if } \bar{M} < \bar{O} 
\end{cases}
\]

\[
NMAEF = \begin{cases} 
\frac{\sum |M_i - O_i|}{\sum O_i} & \text{if } \bar{M} \geq \bar{O} \\
\frac{\sum |M_i - O_i|}{\sum M_i} & \text{if } \bar{M} < \bar{O} 
\end{cases}
\]

where $\bar{M}$, $\bar{O}$ is the mean of the model prediction (predicted, $\bar{Y}_t$) and observation (actual, $Y_t$). These metrics are symmetric and unbiased because their values change according to the ratio of the observed and modeled quantity. The sign of the NMBF indicates overestimation and underestimation, whereas a negative NMBF indicates underestimation by a factor. The NMAEF is considered a robust normalized error metric for extreme values because the denominator of the NMAEF differs by the ratio of the observed and modeled quantities. In this study, in addition to the NMBF and NMAEF, conventional metrics, specifically, the root mean squared error (RMSE) for readability and CORR are also used. A smaller NMAEF indicates better results because it is a type of error; because the NMBF indicates bias, a value closer to 0 is better. The RMSE, like the MSE, indicates better results when the value is smaller, whereas a higher CORR is better.

\[
RMSE = \frac{1}{N} \sqrt{\sum_{i=0}^{N} (M_i - O_i)^2}
\]

\[
CORR = \frac{1}{N} \sqrt{\frac{\sum_i (M_i - \bar{M})(O_i - \bar{O})}{\sum_i (M_i - \bar{M})^2 \sum_i (O_i - \bar{O})^2}}
\]
4.3 Performance Results

In model performance evaluation, it is important to understand the prediction error. The risk, that is, the average loss across all the data, can be decomposed into the reducible error and irreducible error, which is the lowest bound of the generalization error. Moreover, the reducible error is decomposed into bias and variance. As explained in Section 4.2, the NMBF measures the bias of the results, and the NMAEF indicates variance, whereas the MSE, \( \text{MSE}(\hat{Y}) = E[(\hat{Y} - Y)^2] = \text{Var}(\hat{Y}) + \text{Bias}(\hat{Y}, Y)^2 \), measures the sum of variance and bias. For noisy data such as air pollution data, the positive irreducible error \( \epsilon \) is also represented in the MSE. In this section, we select the best models and analyze the errors by decomposing the bias and variance of each model. In addition, the performance of the loss functions is compared to provide insights into the selection of an appropriate loss function according to the data distribution.

Three main factors are compared in this study: the horizon, model, and loss function. We compare them in this order and then generalize the results. In Figure 9 and Figure 10, PM\textsubscript{10} and PM\textsubscript{2.5} model prediction results for various horizons are compared. For shorter horizons, the model shows high accuracy, as expected. However, for longer horizons, the model accuracy decreases because the model error accumulates; in addition, high peaks are not predicted. This tendency is clear when the horizon is longer. Thus, the 24 h horizon, the longest horizon considered here, was used for further analysis to ensure rigorous assessment.

Figures 11 to 14 show scatter plots for the 24 h horizon for multiple models and loss functions. The OU, ARIMA [AR(2) or AR(3)], and XGBoost models are used as baseline models for comparison. The OU process is a completely stochastic random process. The ARIMA model uses only a linear equation, and XGBoost is based on multiple decision trees. The OU process and XGBoost show high dispersion but small bias for all the loss functions and targets compared to the other models, indicating incorrect prediction results. The ARIMA model is highly biased. The OU, ARIMA, and XGBoost models do not use a gradient-descent-like method or can select only predefined loss functions such as the MSE loss.

Regardless of the loss function, the univariate attention model and multivariate models (LSTNet and TST) show better performance than the MLP models. The MLP models outperform OU, ARIMA, and XGBoost. This result is shown qualitatively by the scatter plots (Figures 11 to 14) and quantitatively by the metrics (Tables 5 and 6). The results are not surprising, because the LSTNet and transformer models are considered state-of-the-art models for time series analysis. From the attention model to the TST model, the number of parameters increases from thousands to millions, which is larger than the number of training samples (that is, the models are overparameterized). By contrast, MLP models have hundreds of parameters and do not overfit. A recent study showed that overparameterized models do not have large generalization errors under the assumption of i.i.d. variables Li and Liang [2018], but their use for non-i.i.d. data remains challenging. However, the empirical result for our data shows that overparameterized models such as the attention, LSTNet, and TST models are better than conventional or MLP models, but they exhibit limitations.

Among all the models, the LSTNet and TST models (bold text in Tables 5 and 6) show the best performance overall for quantitative analysis, as demonstrated by the evaluation metric results presented in Tables 5 and 6. The univariate attention model shows the best performance among the univariate models, but for PM\textsubscript{10}, the LSTNet models are better than the attention model, where the relative difference between them is 5.8% (12 h horizon) for the NMAEF and 9.6% (24 h horizon) for the CORR when the MSE is used as the loss function. Using the MCCCR loss does not change this tendency. The MCCCR loss of the TST and attention models of PM\textsubscript{10} is 3.7% (12 h horizon) for the NMAEF and 2.4% for the CORR (12 h horizon). For PM\textsubscript{2.5}, the attention and LSTNet model are the best univariate and multivariate models according to the MSE loss, which is the same as the PM\textsubscript{10} model result. Their relative difference is 5.8% for the NMAEF and 9.7% for the CORR.

The scatter plots for the 24 h horizon in Figures 11 and 12 show that the models are highly biased at high PM\textsubscript{10} concentrations, and only the attention, LSTNet, and TST models outperform the others. The MSE loss of most models seems to converge to its mean; thus, the models cannot predict low concentrations. The multivariate MLP model is unusual because it is less biased in terms of the MSE loss, and the predicted values also tend to converge to the mean value, although the MCCCR loss again indicates a highly biased model. The PM\textsubscript{2.5} models exhibit the same trends as the PM10 models, and these qualitative trends cannot be measured by the evaluation metrics in Figures 13 and 14. This finding indicates that changing the model improves the model results, but the improvement is not easily shown by evaluation metrics. A comprehensive analysis reveals that state-of-the-art models such as the LSTNet or TST models outperform the others. In Borovykh et al. [2019], a theoretical analysis was performed considering non-i.i.d. variables. A deep neural network for time series forecasting may be prone to overfitting to noise, in contrast to networks based on the i.i.d. assumption. According to these results, to improve the model accuracy, it is necessary to change other factors, such as the loss function, not simply the parameters or model complexity.
Deep particulate matter forecasting model using correntropy-induced loss

Figure 9: Scatter and line plot of MCCR of PM$_{10}$ TST model results for various horizons.

Figure 10: Scatter and line plot of MCCR of PM$_{2.5}$ TST model results for various horizons.

The MSE loss and MCCR loss are compared in Figures [15] and [16] for various horizons. Only multivariate models (LSTNet and TST) are used in this context because they show better results than the others. The PM$_{10}$ models using the MCCR loss outperform those using the MSE loss for the NMAEF, RMSE, and CORR (5.8%, 5.8%, and 8.9%, respectively). The PM$_{2.5}$ models, however, show better result only for CORR (8.9%), whereas the NMAEF is worse (approximately 9.6%), and the RMSE is very similar to the NMAE.

The bias ($M_i - O_i$) between two the loss functions at the 24 h horizon is plotted in Figure [17]. The median MSE loss has a positive value, whereas the MCCR loss is close to zero. This result is consistent with the scatter plots, as the mean convergence behavior results in a large bias. In addition, these tendencies are not found in the NMBF values and the shape of the scatter plot.
Mean convergence occurs because of the difference in the distributions of PM$_{10}$ and PM$_{2.5}$. As shown in Figure 4, PM$_{2.5}$ is not fully fitted by a log-normal distribution. This fact is also clearly shown in the histogram and probability density function (PDF) plot obtained by kernel density estimation (KDE) in Figure 18. Unlike that of the PM$_{10}$ data, the distribution of the PM$_{2.5}$ data is similar to a normal distribution. As mentioned in Qi and Majda [2020], the use of the MCCR loss did not address the problem of KL divergence, which sacrifices accuracy when choosing the loss function. When the horizon increases, the mean convergence became severe (Figures 9 and 10), even when the MCCR loss was used. The MCCR loss reduces the severity of this behavior, but it cannot completely remove it.

Moreover, the prediction results are shifted from the actual values on the time axis. When the horizon is longer, the shift seems to be larger. This phenomenon is known as time-warping or time distortion, and it is often studied in time
Figure 13: Scatter plot of the result of PM$_{2.5}$ models using MSE loss. Equations give the best fit line.

Figure 14: Scatter plot of the result of PM$_{2.5}$ models using MCCR loss. Equations give the best fit line.

series similarity analysis and time series classification [Yi et al., 1998]. To the best of our knowledge, time warping in time series forecasting is an unexplored topic and should studied further.

5 Conclusion

We presented a novel TST-model-based air pollution forecasting model. We also showed that the prediction result can be improved by simply varying the loss function depending on the data distribution. The proposed approach exhibits not only better performance but also simplicity. Air pollution data such as PM$_{10}$ and PM$_{2.5}$ data involve multiple factors and highly complex seasonality. The consideration of complex seasonality for different time scales, such as daily or hourly, was also proposed. Our results suggest that setting the input length to 48 h, using multivariate models,
applying state-of-the-art models such as LSTNet or TST, and using the MCCR loss function provides the best results. Importantly, our results provide further evidence for the importance of the data distribution in machine learning research. If the data characteristics are not considered, even state-of-the-art models cannot give better results.

Future research should focus on the development of methods of handling long-range dependence and extreme values. As mentioned in Section 3.4, multiple approaches have been proposed recently to train memory over a longer horizon on the basis of the well-established extreme value theory [Ding et al., 2019], [Ribeiro and Moniz, 2020].

In addition, theoretical research on the generalization of non-i.i.d. data such as time series data is also needed. Time series forecasting is known as a challenging area owing to the possibility of overfitting the dataset. In [Borovykh et al., 2019].
Figure 17: Violin plot of bias ($M_i - O_i$) in 24 h horizon. NMBF is displayed at the top of each plot. The short-dashed lines indicate first and third quantiles, and the long-dashed line indicates the median value.

In [2019], the generalization capabilities of fully connected neural networks were extensively analyzed using a Hessian matrix. A thorough search of the relevant literature showed that this is the only paper to provide a theoretical foundation for the generalization of non-i.i.d. data. Time series data are a classic example of non-i.i.d. values, and noise is easily overfitted to training data with an overparameterized model. Moreover, a difference in distributions between the training and test data might degrade the out-of-sample prediction performance. Although insights on the generalization of time series models are presented in [Borovykh et al., 2019], studies of more complex models such as RNNs are needed. In addition, as mentioned in Section 4.3, the generality of time warping in time series should be investigated further.

Acknowledgments

This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (2017R1E1A1A03070282).

Nomenclature

- $\alpha$: Pareto Index
- $\bar{F}$: Complementary Cumulative Distribution Function
- $\beta$: MCCR scale parameter
- $\hat{Y}_t$: Predicted value
- $\text{res}_h$: Residuals
- $\xi$: Long-range dependence power law exponent
Figure 18: Histogram (blue bars) and KDE plot (orange lines) of raw and deseasonalized data distribution.

$C(r)$  Autocorrelation
$F$  Cumulative Distribution Function
$h$  DFA fluctuation exponent
$M_i$  Predicted value
$O_i$  Actual value
$s_h$  Daily seasonality
$s_w$  Weekly seasonality
$s_{y,\text{smooth}}$  Smoothed yearly seasonality
$V(s)$  DFA fluctuation function
$x(t)$  Single input time series
$Y_t$  Actual value

References

EPA. Health and Environmental Effects of Particulate Matter (PM), 2016.

Hong Bae Kim, Jae Yong Shim, Byoungjin Park, and Yong Jae Lee. Long-term exposure to air pollutants and cancer mortality: A meta-analysis of cohort studies. *International Journal of Environmental Research and Public Health*, 15(11):2608, 2018. ISSN 16604601. doi [10.3390/ijerph15112608](https://doi.org/10.3390/ijerph15112608).

C. Arden Pope and Douglas W. Dockery. Health effects of fine particulate air pollution: Lines that connect. *Journal of the Air and Waste Management Association*, 56(6):709–742, 2006. ISSN 21622906. doi [10.1080/10473289.2006.10464485](https://doi.org/10.1080/10473289.2006.10464485).
Table 5: Summary of results for all models PM$_{10}$ with 48 h of input MCCR or MSE loss. Each row shows the results for a metric and model, whereas each column compares the loss function and horizon size. Bold text indicates that the value is highest for that metric, horizon size, and loss function. The metrics of OU, AR(2) and XGBoost are the same for both loss functions because they are not affected by the loss function or cannot be changed.

| Loss function | MSE | MCCR |
|---------------|-----|------|
| **Metric**    | 3   | 6    | 12   | 24   | 3   | 6    | 12   | 24   |
| **Model**     |     |      |      |      |     |      |      |      |
| NMAEF         |     |      |      |      |     |      |      |      |
| OU            | 0.31160 | 0.43167 | 0.53694 | 0.61026 | 0.31160 | 0.43167 | 0.53694 | 0.61026 |
| AR(2)        | 0.20658 | 0.28436 | 0.36907 | 0.44366 | 0.20658 | 0.28436 | 0.36907 | 0.44366 |
| MLP (Uni.)   | 0.31160 | 0.43167 | 0.53694 | 0.61026 | 0.31160 | 0.43167 | 0.53694 | 0.61026 |
| Attention    | 0.20696 | 0.27927 | 0.35710 | 0.43374 | 0.20000 | 0.27004 | 0.33744 | 0.38906 |
| XGBoost      | 0.23436 | 0.29161 | 0.36583 | 0.42970 | 0.20455 | 0.26496 | 0.32889 | 0.39465 |
| LSTNet (Skip)| 0.19949 | 0.26763 | 0.33630 | 0.40943 | 0.20058 | 0.26478 | 0.33992 | 0.38197 |
| TST          | 0.20826 | 0.28890 | 0.37691 | 0.42262 | 0.19674 | 0.26380 | 0.32507 | 0.37876 |
| RMSE         |     |      |      |      |     |      |      |      |
| OU            | 14.62660 | 20.44270 | 25.43694 | 28.72156 | 14.62660 | 20.44270 | 25.43694 | 28.72156 |
| AR(2)        | 11.23659 | 14.92856 | 18.42945 | 21.02014 | 11.23659 | 14.92856 | 18.42945 | 21.02014 |
| MLP (Uni.)   | 11.39708 | 14.91436 | 18.08831 | 20.47184 | 11.47124 | 15.04905 | 17.91509 | 19.92637 |
| Attention    | 11.22181 | 14.79134 | 18.18526 | 20.74852 | 11.14113 | 14.72053 | 17.81423 | 19.66952 |
| XGBoost      | 12.05999 | 15.64123 | 19.20367 | 22.67406 | 12.05999 | 15.64123 | 19.20367 | 22.67406 |
| LSTNet (Skip)| 10.96963 | 14.29854 | 17.49411 | 20.69443 | 11.15474 | 14.60806 | 17.79352 | 19.80228 |
| TST          | 11.25933 | 14.98114 | 18.54122 | 20.41276 | 11.15219 | 14.63372 | 17.45747 | 19.56047 |
| CORR         |     |      |      |      |     |      |      |      |
| OU            | 0.80755 | 0.62727 | 0.42931 | 0.27401 | 0.80755 | 0.62727 | 0.42931 | 0.27401 |
| AR(2)        | 0.88081 | 0.78134 | 0.65179 | 0.53814 | 0.88081 | 0.78134 | 0.65179 | 0.53814 |
| MLP (Uni.)   | 0.87526 | 0.77890 | 0.65876 | 0.55302 | 0.87453 | 0.77841 | 0.65252 | 0.53576 |
| Attention    | 0.88089 | 0.78308 | 0.66262 | 0.55321 | 0.88112 | 0.78211 | 0.65675 | 0.55157 |
| XGBoost      | 0.85994 | 0.75357 | 0.60975 | 0.41355 | 0.85994 | 0.75357 | 0.60975 | 0.41355 |
| LSTNet (Skip)| 0.88502 | 0.79557 | 0.67414 | 0.49977 | 0.88502 | 0.79557 | 0.67414 | 0.49977 |
| TST          | 0.87931 | 0.77791 | 0.65392 | 0.54980 | 0.88282 | 0.78749 | 0.67248 | 0.56119 |
| NMBF         |     |      |      |      |     |      |      |      |
| OU            | -0.00323 | -0.00309 | -0.00028 | 0.00463 | -0.00323 | -0.00309 | -0.00028 | 0.00463 |
| AR(2)        | 0.04033 | 0.07405 | 0.12661 | 0.19097 | 0.04033 | 0.07405 | 0.12661 | 0.19097 |
| MLP (Uni.)   | 0.01702 | 0.06066 | 0.10342 | 0.16357 | -0.00163 | 0.00440 | -0.03813 | -0.03663 |
| Attention    | 0.03545 | 0.06052 | 0.10803 | 0.18671 | -0.00687 | -0.01279 | -0.02345 | -0.04751 |
| XGBoost      | 0.02536 | 0.04296 | 0.07382 | 0.08280 | 0.02536 | 0.04296 | 0.07382 | 0.08280 |
| LSTNet (Skip)| 0.01764 | 0.03158 | 0.05536 | 0.09990 | 0.01848 | 0.02684 | -0.02924 | 0.00358 |
| TST          | 0.02888 | 0.06234 | 0.14025 | 0.15299 | 0.01167 | 0.01944 | -0.00590 | -0.01241 |

OECD. *The Economic Consequences of Outdoor Air Pollution*. Organisation for Economic Co-operation and Development, 2016. doi:10.1787/9789264257474-en

AirKorea. Annual Report of Air Quality. Technical report, AirKorea, 2019. URL: https://www.airkorea.or.kr/web/detailViewDown?pMENU_NO=125

World Health Organization. Air Quality Guidelines. *Air Quality Guidelines*, 2006. ISSN 0944-1344.

OECD. Air pollution exposure indicator, (2021).
Table 6: Summary of results for all models PM$_{2.5}$ with 48 h of input MCCR or MSE loss. Each row shows the results for a metric and model, whereas each column compares the loss function and horizon size. Bold text indicates that the value is highest for that metric, horizon size, and loss function. The metrics of OU, AR(3) and XGBoost are the same for both loss functions because they are not affected by the loss function or cannot be changed.

| Metric | Loss function | MSE | MCCR |
|--------|---------------|-----|------|
|        | Horizon size  | 3   | 6    | 12   | 24   | 3   | 6    | 12   | 24   |
| NMAEF  | OU            | 0.33740 | 0.46857 | 0.60018 | 0.68653 | 0.33967 | 0.46773 | 0.59131 | 0.68730 |
|        | AR(3)         | 0.21508 | 0.29414 | 0.37558 | 0.43095 | 0.21508 | 0.29414 | 0.37558 | 0.43095 |
|        | MLP (Uni.)    | 0.21828 | 0.29117 | 0.36862 | 0.43095 | 0.21753 | 0.29193 | 0.37206 | 0.46097 |
|        | Attention      | 0.22568 | 0.29492 | 0.36727 | 0.43198 | 0.21453 | 0.29735 | 0.39183 | 0.48990 |
|        | XGBoost        | 0.23096 | 0.29453 | 0.37795 | 0.45343 | 0.23096 | 0.29453 | 0.37795 | 0.45343 |
|        | MLP (Multi.)   | 0.22659 | 0.29086 | 0.36495 | 0.43857 | 0.20924 | 0.27786 | 0.39393 | 0.48344 |
|        | LSTNet (Skip) | 0.20757 | 0.27602 | 0.35062 | 0.42858 | 0.20747 | 0.27770 | 0.37823 | 0.45436 |
|        | TST            | 0.21458 | 0.28792 | 0.36576 | 0.43434 | 0.20901 | 0.28004 | 0.36093 | 0.47585 |
| RMSE   | OU            | 9.42310 | 13.10905 | 16.74847 | 19.28485 | 9.47383 | 13.16762 | 16.65591 | 19.33624 |
|        | AR(3)         | 6.93497 | 9.20198 | 11.48500 | 13.39068 | 6.93497 | 9.20198 | 11.48500 | 13.39068 |
|        | MLP (Uni.)    | 7.05789 | 9.25539 | 11.44019 | 13.16623 | 6.97408 | 9.18980 | 11.30530 | 13.16732 |
|        | Attention      | 7.33964 | 9.37080 | 11.41769 | 13.06500 | 6.92724 | 9.22089 | 11.56166 | 13.56691 |
|        | XGBoost        | 7.55752 | 9.37789 | 11.99462 | 14.24905 | 7.55752 | 9.37789 | 11.99462 | 14.24905 |
|        | MLP (Multi.)   | 6.99418 | 9.05431 | 11.32332 | 13.45415 | 6.70583 | 8.78049 | 11.81080 | 13.97028 |
|        | LSTNet (Skip) | 6.69026 | 8.71840 | 11.20545 | 12.99081 | 6.92550 | 9.08245 | 11.20545 | 12.99081 |
| CORR   | OU            | 0.81975 | 0.65019 | 0.42922 | 0.23856 | 0.81864 | 0.65106 | 0.43967 | 0.24084 |
|        | AR(3)         | 0.89937 | 0.81549 | 0.69832 | 0.56058 | 0.89937 | 0.81549 | 0.69832 | 0.56058 |
|        | MLP (Uni.)    | 0.89364 | 0.80890 | 0.68903 | 0.55275 | 0.89687 | 0.81209 | 0.69759 | 0.56876 |
|        | Attention      | 0.88970 | 0.80709 | 0.69212 | 0.56860 | 0.88957 | 0.81279 | 0.68962 | 0.54083 |
|        | XGBoost        | 0.87710 | 0.80211 | 0.64844 | 0.45717 | 0.87710 | 0.80211 | 0.64844 | 0.45717 |
|        | MLP (Multi.)   | 0.90058 | 0.82361 | 0.70043 | 0.53116 | 0.90481 | 0.82975 | 0.68050 | 0.49566 |
|        | LSTNet (Skip) | 0.90760 | 0.83255 | 0.71840 | 0.54388 | 0.90900 | 0.82169 | 0.70240 | 0.56993 |
|        | TST            | 0.89831 | 0.81675 | 0.70713 | 0.57669 | 0.90009 | 0.82127 | 0.70663 | 0.57703 |
| NMBF   | OU            | -0.00202 | -0.00720 | -0.00745 | 0.00006 | 0.00275 | 0.00527 | 0.00916 | 0.00181 |
|        | AR(3)         | 0.01556 | 0.02749 | 0.04589 | 0.06797 | 0.01556 | 0.02749 | 0.04589 | 0.06797 |
|        | MLP (Uni.)    | 0.02095 | 0.02734 | 0.05274 | 0.06668 | -0.02503 | -0.04517 | -0.10505 | -0.16068 |
|        | Attention      | 0.01346 | 0.02096 | 0.03643 | 0.06301 | -0.02102 | -0.04517 | -0.10505 | -0.16068 |
|        | XGBoost        | -0.00995 | 0.00255 | 0.01059 | 0.02108 | -0.00995 | 0.00255 | 0.01059 | 0.02108 |
|        | MLP (Multi.)   | 0.06821 | 0.07098 | 0.07403 | 0.09475 | 0.00009 | 0.02031 | 0.01059 | 0.02108 |
|        | LSTNet (Skip) | 0.01879 | 0.01997 | 0.03186 | 0.06406 | -0.00422 | -0.08282 | -0.10908 | -0.19098 |
|        | TST            | 0.01678 | 0.01718 | 0.04882 | 0.07764 | -0.01225 | -0.07278 | -0.1614 | -0.1614 |
Deep particulate matter forecasting model using correntropy-induced loss

Yang Zhang, Marc Bocquet, Vivien Mallet, Christian Seigneur, and Alexander Baklanov. Real-time air quality forecasting, part I: History, techniques, and current status. *Atmospheric Environment*, 60:632–655, 2012. ISSN 13522310. doi:10.1016/j.atmosenv.2012.06.031

D. W. Byun and J. K. S. Ching. Science Algorithms of the EPA Models-3 Community Multiscale Air Quality (CMAQ) modeling system. *EPA/600/R-99/030*, 1999. ISSN 1939-1404.

Isabelle Bey, Daniel J. Jacob, Robert M. Yantosca, Jennifer A. Logan, Brendan D. Field, Arlene M. Fiore, Qinbin Li, Hongy G. Liu, Loretta J. Mickley, and Martin G. Schultz. Global modeling of tropospheric chemistry with assimilated meteorology: Model description and evaluation. *Journal of Geophysical Research Atmospheres*, 106 (D19):23073–23095, 2001. ISSN 01480227. doi:10.1029/2001JD000807.

Georg A. Grell, Steven E. Peckham, Rainer Schmitz, Stuart A. McKeen, Gregory Frost, William C. Skamarock, and Brian Eder. Fully coupled "online" chemistry within the WRF model. *Atmospheric Environment*, 39(37):6957–6975, 2005. ISSN 13522310. doi:10.1016/j.atmosenv.2005.04.027

A. Baklanov, P. G. Mestayer, A. Clappier, S. Zilitinkevich, S. Joffre, A. Mahura, and N. W. Nielsen. Towards improving the simulation of meteorological fields in urban areas through updated/advanced surface fluxes description. *Atmospheric Chemistry and Physics*, 8(3):523–543, 2008. ISSN 16807324. doi:10.5194/acp-8-523-2008

Guangqiang Zhou, Jianming Xu, Ying Xie, Luuy Chang, Wei Gao, Yixuan Gu, and Ji Zhou. Numerical air quality forecasting over eastern China: An operational application of WRF-Chem. *Atmospheric Environment*, 153:94–108, 2017. ISSN 18732844. doi:10.1016/j.atmosenv.2017.01.020

Zhigen Shang, Tong Deng, Jianniang He, and Xiaohui Duan. A novel model for hourly PM2.5 concentration prediction based on CART and EELM. *Science of the Total Environment*, 651:3043–3052, 2019. ISSN 18791026. doi:10.1016/j.scitotenv.2018.10.193

Danni Guo, Renkuan Guo, and Christien Thiart. Predicting air pollution using fuzzy membership grade Kriging. *Computers, Environment and Urban Systems*, 31(1):33–51, 2007. ISSN 01989715. doi:10.1016/j.compenvurbsys.2005.07.006

Weiqiang Wang and Ying Guo. Air pollution PM2.5 data analysis in Los Angeles long beach with seasonal ARIMA model. In *2009 International Conference on Energy and Environment Technology, ICEET 2009*, pages 7–10, 2009. ISBN 9780769538198. doi:10.1109/ICEET.2009.468

Guokun Lai, Wei Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018*, 2018. ISBN 9781450356572. doi:10.1145/3209978.3210006

Shun Yao Shih, Fan Keng Sun, and Hung yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441, 2019. ISSN 15730565. doi:10.1007/s10994-019-05815-0

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhui Chen, Yu-Xiang Xiang Wang, and Xifeng Yan. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *arXiv*, jun 2019. ISSN 23318422. URL http://arxiv.org/abs/1907.00235

Ji Eun Choi, Hyesun Lee, and Jongwoo Song. Forecasting daily PM10 concentrations in Seoul using various data mining techniques. *Communications for Statistical Applications and Methods*, 25(2):199–215, mar 2018. ISSN 23834757. doi:10.29220/CSAM.2018.25.2.199 URL https://doi.org/10.29220/CSAM.2018.25.2.199

Kyunghak Cho, Byoung-young Lee, Myeongheum Kwon, and Seogcheol Kim. Air Quality Prediction Using a Deep Neural Network Model. *Journal of Korean Society for Atmospheric Environment*, 35(2):214–225, 2019. ISSN 1598-7132. doi:10.5572/kosae.2019.35.2.214

Fabiana Franceschi, Martha Cobo, and Manuel Figueredo. Discovering relationships and forecasting PM10 and PM2.5 concentrations in Bogotá, Colombia, using Artificial Neural Networks, Principal Component Analysis, and k-means clustering. *Atmospheric Pollution Research*, 9(5):912–922, 2018. ISSN 13091042. doi:10.1016/j.apr.2018.02.006

Yun Bai, Yong Li, Bo Zeng, Chuan Li, and Jin Zhang. Hourly PM 2.5 concentration forecast using stacked autoencoder model with emphasis on seasonality. *Journal of Cleaner Production*, 224:739–750, jul 2019. ISSN 09596526. doi:10.1016/j.jclepro.2019.03.253

Weifeng Liu, Puskal P. Pokharel, and Jose C. Principe. Correntropy: Properties and applications in non-Gaussian signal processing. *IEEE Transactions on Signal Processing*, 55(11):5286–5298, 2007. ISSN 1053587X. doi:10.1109/TSP.2007.896065

Yunlong Feng, Xiaolin Huang, Lei Shi, Yuning Yang, and Johan A.K. Suykens. Learning with the maximum correntropy criterion induced losses for regression. *Journal of Machine Learning Research*, 16(30):993–1034, 2015. ISSN 15337928.
Deep particulate matter forecasting model using correntropy-induced loss

Robert Cichowicz, Grzegorz Wielgosiński, and Wojciech Fetter. Dispersion of atmospheric air pollution in summer and winter season. Environmental Monitoring and Assessment, 189(12), 2017. ISSN 15732959. doi:10.1007/s10661-017-6319-2

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays. Bioinformatics, 17(6):520–525, 2001. ISSN 13672959. doi:10.1093/bioinformatics/17.6.520

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011. ISSN 15324435.

William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. Journal of the American Statistical Association, 74(368):829–836, 1979. ISSN 1537274X. doi:10.1080/01621459.1979.10481038

Jan W. Kantelhardt, Eva Koscielny-Bunde, Henio H.A. Rego, Shlomo Havlin, and Armin Bunde. Detecting long-range correlations with detrended fluctuation analysis. Physica A: Statistical Mechanics and its Applications, 295(3-4): 441–454, 2001. ISSN 03784371. doi:10.1016/S0378-4371(01)00144-3

Eva Koscielny-Bunde, Jan W. Kantelhardt, Peter Braun, Armin Bunde, and Shlomo Havlin. Long-term persistence and multifractality of river runoff records: Detrended fluctuation studies. Journal of Hydrology, 322(1-4):120–137, 2006. ISSN 00221694. doi:10.1016/j.jhydrol.2005.03.004.

Aaron Clauset, Cosma Rohilla Shalizi, and M. E.J. Newman. Power-law distributions in empirical data. SIAM Review, 51(4):661–703, 2009. ISSN 00361445. doi:10.1137/070710111.

Jeff Alstott, Ed Bullmore, and Dietmar Plenz. Powerlaw: A python package for analysis of heavy-tailed distributions. PLoS ONE, 9(1):e85777, 2014. ISSN 19326203. doi:10.1371/journal.pone.0085777.

G. E. Uhlenbeck and L. S. Ornstein. On the theory of the Brownian motion. Physical Review, 36:823, 1930. ISSN 0031899X. doi:10.1103/PhysRev.36.823.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794, 2016. ISBN 9781450342322. doi:10.1145/2939672.2939785.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT press, 2015. ISBN 9780262035613.

Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting, 14(1):35–62, 1998. ISSN 01692070. doi:10.1016/S0169-2070(97)00044-7

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997. ISSN 08999767. doi:10.1162/neco.1997.9.8.1735.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, 2014. ISBN 9781937284961. doi:10.3115/v1/d14-1179.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015.

Neo Wu, Bradley Green, Xue Ben, and Shawn O’Banion. Deep transformer models for time series forecasting: The influenza prevalence case, 2020. ISSN 23318422.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning, 2020. ISSN 23318422.

Mustafa Bayram, Tugcem Partal, and Gulsen Orucova Buyukoz. Numerical methods for simulation of stochastic differential equations. Advances in Difference Equations, 17, 2018. ISSN 16871847. doi:10.1186/s13662-018-1466-5.
Deep particulate matter forecasting model using correntropy-induced loss

Skipper Seabold and Josef Perktold. Statsmodels: Econometric and Statistical Modeling with Python. In Proceedings of the 9th Python in Science Conference, 2010. doi:10.25080/majora-92bf1922-011

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pages 3104–3112, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010, 2017.

Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2Vec: Learning a vector representation of time, 2019.

Daizong Ding, Mi Zhang, Xudong Pan, Min Yang, and Xiangnan He. Modeling extreme events in time series prediction. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019. ISBN 9781450362016. doi:10.1145/3292500.3330896

Rita P. Ribeiro and Nuno Moniz. Imbalanced regression and extreme value prediction. Machine Learning, 109(9):1083–1835, 2020. ISSN 15730565. doi:10.1007/s10994-020-05900-9

Di Qi and Andrew J. Majda. Using machine learning to predict extreme events in complex systems. Proceedings of the National Academy of Sciences of the United States of America, 117(1):52–59, 2020. ISSN 10916490. doi:10.1073/pnas.1917285117

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019. ISBN 9781450362016. doi:10.1145/3292500.3330701

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach and H. Larochelle and A. Beygelzimer and F. d’Alché-Buc and E. Fox and R. Garnett, editor, Advances in Neural Information Processing Systems, pages 8024–8035, 2019.

Jisu Myoung, Tahee Kim, Yonghee Lee, Insuk Suh, and Lim Suk Jang. Optimization of the computing environment to improve the speed of the modeling (WRF and CMAQ) calculation of the National Air Quality Forecast System. Journal of Environmental Science International, 27(8):723–735, 2018. ISSN 1225-4517. doi:10.5322/jesi.2018.27.8.723

Shaocai Yu, Brian Eder, Robin Dennis, Shao-Hang Chu, and Stephen E. Schwartz. New unbiased symmetric metrics for evaluation of air quality models. Atmospheric Science Letters, 7(1):26–34, 2006. ISSN 1530-261X. doi:10.1016/j.asl.125

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In Advances in Neural Information Processing Systems, 2018.

Anastasia Borovykh, Cornelis W. Oosterlee, and Sander M. Bohté. Generalization in fully-connected neural networks for time series forecasting. Journal of Computational Science, 2019. ISSN 18777503. doi:10.1016/j.jocs.2019.07.007

Byoung Kee Yi, H. V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. In Proceedings - International Conference on Data Engineering, 1998. doi:10.1109/icde.1998.655778.