# Solve Air: Air Pollution Forecasting using Deep Attentive Sequence Learning

Abinash Sinha* and Dhananjay Muddappa

Department of Computer Science and Engineering, University of Minnesota

*Abstract* - **Urbanization has led to alarming amount of pollution. Controlling air pollution is paramount to survival of earth and thus should be one of the top priorities. With the advent of highly efficient computers in the twenty-first century, artificial intelligence techniques to forecast air pollution are gaining traction. In particular, deep learning could be exploited to solve this time-series forecasting problem. This paper summarizes my venture in understanding sequence modeling using recurrent neural networks and corresponding deep learning optimization techniques to apply suitable techniques for air pollution forecasting. We explore and implement existing state-of-the-art attention mechanism utilizing a combination of convolutional neural networks and recurrent neural networks which attends to temporal patterns across time series of each feature of data. Since change in weather conditions possesses a seasonal nature there seems to be a pattern in features used for predicting air pollution (PM2.5) level. Also, comparison is being made with traditional mechanism of attention and encoder-decoder RNNs.**

*Keywords* - **PM2.5, Time-series, Attention, Convolutional Neural Network, Recurrent Neural Network**

## 1. INTRODUCTION

Air pollution is shortening lifespans by months and in some cases years. Indian metropolitan cities such as Bangalore [1] and Delhi [2] have dangerously high levels of air pollution. In a study conducted by the Harvard International Review, it was found that 40% of the population in Delhi suffer from respiratory ailments. This problem could be tackled with proper planning of resources and execution of smart policies by governments. In order to be able to plan things for future, past data is required. This is where artificial intelligence, more precisely machine learning comes into the picture. It isn't possible for humans to understand the vast complex datasets of pollution and all its covariates. Machine learning could help us predict pollution levels while considering all the various features that are available. For example, IBM partnered with Beijing, one of the highly polluted cities in the world, to use machine learning and big data to forecast air pollution and accordingly update and implement air quality control policies of the city. Protecting our environment implies protecting its inhabitants. The definition of development should involve enacting smart measures to protect our environment from over-exploitation and thus pollution, using technology.

Features like wind direction, wind speed, air temperature, air pressure, dew point, cumulative rain hours, pollution levels in the past could help us define the state of the environment. Therefore, using the past values of these features at one-hour interval, we configured the number of hours as our input sequence and the output as the pollution levels at every hour for configured number of hours in advance.

Also, with the motive to gain in-depth understanding of sequential modelling we reviewed several related well-cited papers being heavily referred to for doing our research. Sequential modelling had been the first choice in terms of solving problems involving sequential nature of data like translation text in one language to text in another language (neural machine translation). This mainly involves a recurrent neural network which involves neurons or so to say cells that

are interconnected to each other in such a way that each cell takes as input elements of given input sequence. This idea really fascinated us to work towards understanding this far-reaching concept popular for its effectiveness in broad practical applications like time-series forecasting.

Forecasting air pollution falls under the category of multi-variate time series forecasting with the features or variable as described above. There are various techniques available to approach such a time-series prediction problem and we would be reviewing some of the best techniques used considering only the temporal aspects of the problem since we are just focusing on one location. For this work, we are not considering the aspects of spatial variability in the model. We just focus per location basis like Beijing or Delhi. First, we provide the results for the city of Beijing where prior research had been done. Later on, we train and test our models for India dataset and analyse the effectiveness of the models. As per our research, several techniques involving recurrent neural networks have been applied with sequence-to-sequence learning. Encoder-decoder RNN is the baseline model with which we compare the other attention-based sequence learning models. There wasn't much work found in using attention mechanisms of Bahdanau [14] on Beijing dataset. Here, we are using aforementioned attention mechanisms and also explore existing state-of-the-art attention mechanism which incorporates attention towards temporal patterns prevailing in features of the dataset for time-series forecasting indirectly by attending to features of hidden state outputs of encoder network used. The main idea behind time-series forecasting is to be able to understand the relevant contributions of each feature towards prediction. Also, comparison is made with a naive baseline method of auto-regressive integrated moving average (ARIMA).

## 2. THEORY

Particulate matter (PM) is defined as a mixture of solid and liquid particles found in air. PM2.5 in particular is of particles with less than 2.5 micrometers in diameter. These particles are very fine and studies have found that a there is a close link between PM2.5 exposure and death due to heart and lung disease. Thus, accurately predicting the PM2.5 values is of utmost concern in order to prevent disease and death. With current available data is possible to obtain hourly PM2.5 values which could be used to predict the values up to 24 hours in the future. This would be a valuable prediction as governments could enact both temporary and permanent measures to reduce pollution for the safety of both the citizens and environment around. For example, this information could be used to issue warnings to the citizens to stay indoors or even use it to subsidize public transport on high PM2.5 level days in order to incentivize citizens to not use their personal vehicles, which could exacerbate the problem.

We considered many techniques such as regression, neural networks, recurrent neural networks, long short term memory (LSTM) and time-series models to do these predictions. What we were looking for was a model that had high accuracy and also would run fast.

A brief overview of the methods that precede LSTM-based methodology are listed below.

### 2.1 Feed Forward Neural Network

Neural Networks are excellent at perceiving things from data which is not clearly perceptible to humans. While it was originally motivated biologically by the neural network of the human brain they do not really function the same way as biological neural networks. A simple neural network is represented in Figure 1. It consists of an input X with each row labelled as $x_1$, $x_2$, etc. A weight is applied in the next step to each of these rows and next an activation function such as sigmoid or rectified linear unit (ReLU) or tanh is used to introduce non-linearity to the input. A full neural network is shown in Figure 2.

The above described transformations are commonly applied in the hidden layers until the final step where another function such as the softmax function is used to obtain the output. What is amazing about a neural network is that they are capable of learning any function when given an input that can map it to the output.

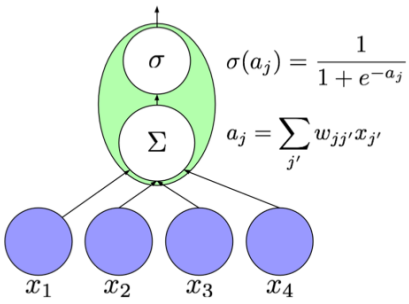



**Figure 1**

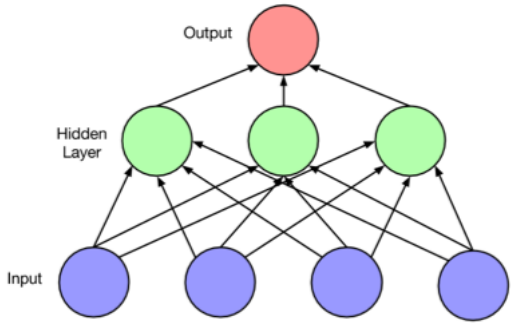



**Figure 2**

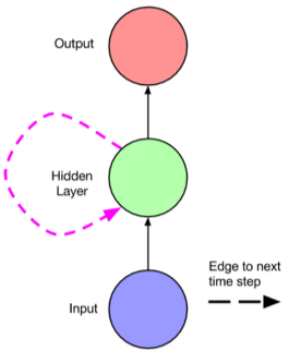



**Figure 3**

Thus, neural networks have been developed with varying architectures – using different number of input nodes and hidden layers and different activation functions – to perform a variety of tasks such as image classification and character recognition. This is called a feed forward network where there are no cycles present in the computations. Each layer will apply a function on the data and then pass it to the next layer in a forward direction.

However, despite their strengths, the standard neural networks have weaknesses. When each row is independently generated i.e. they are not related to one other – feed forward neural networks perform splendidly. When they do have relationships in time or space neural networks do not work as well as this independence assumption fails. Prediction of PM2.5 is heavily temporally dependent and so feed forward neural networks would not be a good fit to predict PM2.5 values. In order to overcome this shortcoming, we then considered recurrent neural networks (RNN).

## 2.2 Recurrent Neural Network

They can be defined as connectionist models which have the ability to selectively pass information across sequence steps, while processing sequential data one element at a time. For practical purposes such as forecasting or natural language processing RNNs are very useful as they can handle time interdependencies. RNNs do this by making use of the architecture shown in Figure 3. It is an augmented feed-forward neural network which has edges that can span adjacent time steps thus introducing a notion of time to the model. We can unfold the network to understand this concept better as seen in Figure 4. Each hidden layer sends information to the next subsequent time step and thus the interdependency of the data is maintained. Vanilla RNNs however have some glaring issues.

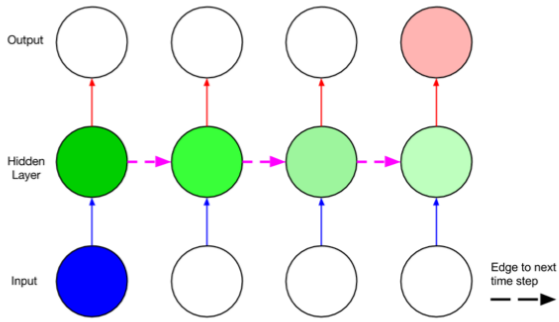



**Figure 4.**

One problem is that as the gap increases between relevant information and the point at which we are predicting RNNs have problems in connecting the information. These long range dependencies also tend to have a problem which is known as vanishing/exploding gradients. Gradient descent is the most commonly used technique in optimization for neural networks. It is basically a derivative that is applied on the error between output of the network and the actual expected result. This information is passed on from one hidden layer to the next in order to perform updates. In RNNs this value has a tendency to quickly approach zero or infinity which would tend to slow the training or cause errorneous updates. For our dataset, PM2.5 values are quite reliant on past values and hence we moved on to find other better architectures. Many modifications were applied to RNNs to mitigate their problems and one of the most successful architectures is the long short term memory (LSTM). They are specifically designed to avoid the problems of vanilla RNN.

## 2.4 Convolutional Neural Network

It is a special type of neural network known for handling grid-like topological data. Convolution leverages three important ideas that can help improve a machine learning system: sparse interactions,

parameter sharing and equivariant representations. A pure CNN layer has lesser number of parameters than usual feed forward neural network since it is not fully connected. Examples include time-series data, which canbe thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Instead of matrix multiplication convolution operation is performed here.

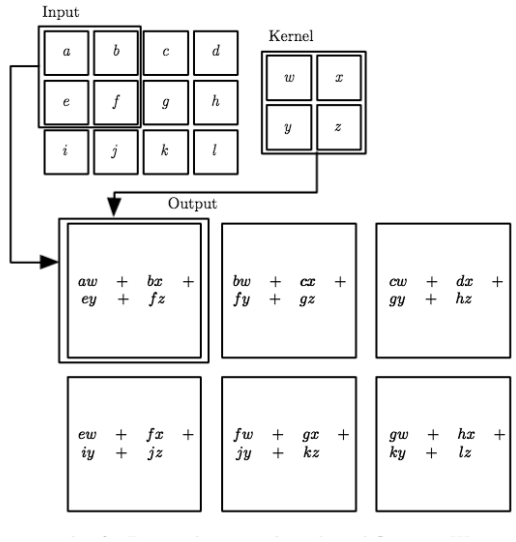



**Figure 7.**

One simple use case where convolution is used is edge detection in an image.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n).$$

An equation representing convolution operation is shown above.

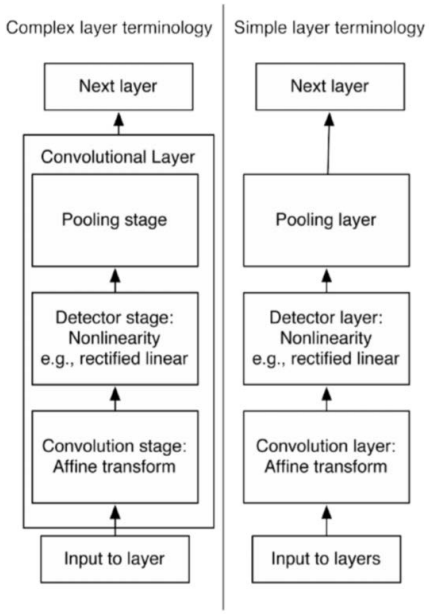



**Figure 8.**

Convolution basically tries to capture local cross-correlation ingrained in the amongst the data being fed. A data where we can apply convolution is which fails the pixel permutation test. For example, if pixels in image is permuted then we would loose the original image. Similaryly, for time-series data as well where if we permute the sequences then it wouldn't be aligned chronologically anymore. We would lose the temporal characteristics and that data would be something else.

Pooling is done in order to decrease the number of features that we get from the convolved data with the intuition that local values are correlated to each other so we could take either maximum/mean of those neighbouring values.

## 2.3 Long Short-Term Memory

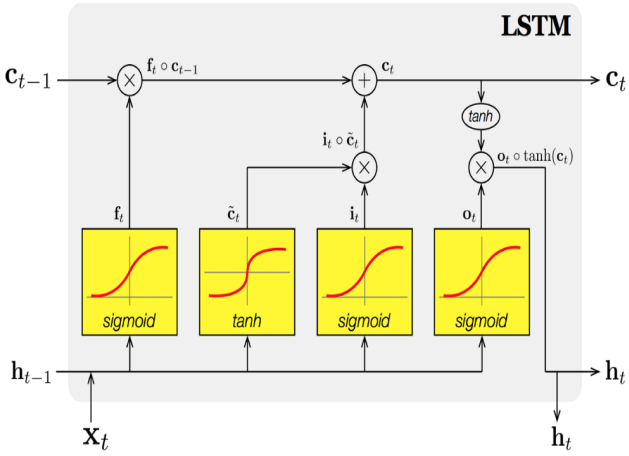



**Figure 5. LSTM Architecture**

The architecture of LSTM can be seen in Figure 5. It utilizes the following equations,

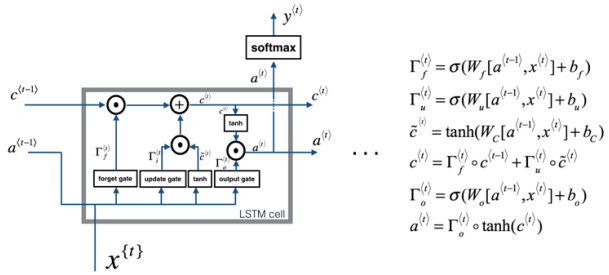



**Figure 6. LSTM Equations**

$c_{t-1}$ is the previous cell state, $c_t$ is the current cell state, $h_{t-1}$ is the output of the previous timestep's hidden layer, $h_t$ is the output of the current cell and $x_t$ is the input. To decide what information needs to be discarded, the first unit $f_t$ – the forget gate – is used. It looks at $h_{t-1}$ and current input xt and chooses to either forget the entire $c_{t-1}$ with an output of 0 or to remember

it with an output of 1. The next step is to calculate the current cell state. It makes use of both the input gate it and candidate cell state $\check{c}_t$. The input gate is a unit which takes activation from the current data point $x_t$ and the previous time step output from a previous hidden layer $h_{t-1}$. Next a tanh layer is used to create a candidate cell $\check{c}_t$ which is used to finally update the current state of the cell. Then both these values are used along with the forget gate to finally find the current cell state given by the formula for $c_t$. Lastly to decide what the output of this cell is we make use of $c_t$ and the output gate $o_t$. The output gate is a sigmoid function which decides what part of cell state $c_t$ to update and then we finally can obtain $h_t$ which will be sent to the next cell and the cycle will continue.

## 3. BACKGROUND / RELATED WORK

Being able to forecast air pollution using machine learning was a goal that had been tried over the several years. Earliest we could find was IBM's Green Horizon research initiative launched in the year 2014 which tried to control alarming air pollution levels in China. They employed machine learning to train several deterministic models in order to help adapt to changing situations and eventually select the best amongst those. As and when deep learning had been gaining popularity research seems to be progressing towards solving this problem.

Lately, research on utilizing statistical methods over deterministic models have gained traction given the rapidly growing research in deep learning. In this respect, we focused ourselves more on getting acquainted with deep learning techniques in order to be able to experiment its use to understand and approach the underlying problem statement.

Air pollution data is typically cyclical (seasonal) and hence general linear methods such as regression limit the accuracy of the predictions. Feed-forward neural networks that can handle non-linear data perform poorly as they do not take into consideration the sequential inter-dependencies existing in the dataset. For temporal sequences recurrent neural networks instead are popularly used. They however have two glaring weaknesses: firstly, they create derivatives that vanish when many layers are present in the architecture and secondly, they are not so efficient with high training cost and struggle to remember data across a large sequence.

[3] has talked about learning sequence of targets from sequence of inputs using stacked layers of LSTM giving an empirical understanding of this technique. It talks about encoding the sequence of inputs into a single vector that represents the inter-dependencies ingrained in sequential input. This vector is then fed into decoder network which provides output in the form of sequence. LSTM has this ability to capture relevant information over longer sequences better than vanilla RNN could as per the understanding from [4].

Following the far-reaching notions stated in [3] and [4], [6] talks about using LSTM-based sequence-to-sequence to create a model for air pollution forecasting. In [7], encoder-decoder framework using multiple RNN layers with LSTM cells stacked together to achieve better accuracy. It tries to represent the temporal inter-dependencies in input sequence by computing the mean of these hidden vectors of encoder RNN. This mean vector is the context vector fed into each cell of decoder RNN.

It is quite prevalent to conclude by conducting literature review that LSTM-based RNN have been the basic model structure used for air pollution prediction. Even [15] talks about the same by using it for classification purpose in terms of Air Quality Index and makes comparison with Support Vector Regression using non-linear RBF kernel. Apart from LSTM, GRU and vanilla RNN cell configuration have also been tried and compared with each other in [17] on a portion of a large publicly available dataset of AirNet where in GRU seems to outperform the other cell architectures considering GRU has a smaller number of units and dataset is large and training happens faster than LSTM could.

All this research into using statistical methods for air pollution forecasting have commonly used RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) as the core loss functions or to make comparisons with baseline models they use. There is also a comparison done between the different RNN models made using plain RNN cell, LSTM cell and GRU cell. A simple summarization is done for performances of these different models by varying the number of layers in network and also varying the length of input sequence fed into the network of layers. It is a use case of many-to-one where length of input sequence is one and only a scalar value is predicted.

[21] has a very interesting conclusion that setting the forget gate bias to 1 improves the performance of LSTM greatly. Even more modern techniques such as gated recurrent unit do not out-perform LSTM by much when they set $b_f$ to 1. The paper tested over

10000 different architectures and found that none of the architectures dramatically increase performance and various test data. Thus, this paper gave us more confidence as to selecting LSTM for use in model over any other RNN architecture.

Next, we wanted to know more about various LSTM variants and their performance. For this we made reviewed paper [9]. The paper tested 8 different variants – no input gate, no forget gate, no output gate, no input activation function, no output activation function, coupled input and forget gate (CIFG), no peepholes (NP) and full gate recurrence (FGR). CIFG is a modified GRU. FGR adds nine additional recurrent weight matrices. This paper also found that vanilla LSTM tends to have very comparable performance to all its variants. CIFG and NP were found to not decrease performance while also reducing the number of parameters and computational complexity so these two variants might be used to vanilla LSTM if modeling data is too complex. Through its empirical results, it could be concluded that learning rate can be tuned first using a fairly small network. Then later on we can play with other hyper parameters.

## 4. APPROACH

### 4.1 Data Collection and Pre-Processing

There was a lot of experiments done on Beijing dataset available on UCI repository. As followed in [6], we used the years 2010 - 2012 as our training set, 2013 as our validation set and 2014 as our test set to use for the following threee models - vanilla encoder-decoder, encoder-decoder with bi-directional encoder and decoder using Bahdanau attention mechanism as explained in [14] and lastly the state-of-the-art attention feature-wise temporal pattern attention mechanism. The pollution level is given on an hourly basis here. It has 8 features - PM2.5, cumulative dew point, air temperature, air pressure, wind direction, wind speed, cumulative snow hours and cumulative rain hours.

Along with testing on the Beijing data, we also wanted to try on a new dataset. We found a dataset which had features across the years 2012 to 2015 for the city of Delhi, India on an hourly basis with the following features: air temperature, cumulative precipitation, wind speed and wind direction.

Next searched for more features that are closely linked with PM2.5 values. Found 21 total datasets. 7 for each of the years 2013, 2014 and 2015: CO levels, NO levels, NO2 levels, O3 levels, SO2 levels, PM10, and PM2.5

The 22 total datasets were then joined using their respective data/time columns to create a consolidated database which had ~18,000 rows with 10 features and 1 column PM2.5 which is the predicted variable. Due to devices malfunctioning and poor data collection methodologies we found that many of the rows had to be dropped as they had widely inconsistent PM2.5 values which would cause improper predictions. Blank values were imputed using the last valid value for the 11 features of the dataset. After this min-max scaling was applied to the dataset to ensure all the data features are on a similar scale for ease of comparison and also ease of network training. The final dataset that we made use of had 13612 rows and 11 columns considering the PM2.5 value too as a feature to be fed in to our input sequence to RNN models.

For Beijing dataset, input to each cell of LSTM RNN had 8 features while for India dataset it had 11 features. The output had 1 feature for both Seq2Seq encoder-decoder and Bahdanau attention-based RNN models which indicated PM2.5 value.

### 4.2 ARIMA Model

As the dataset is one that consists of date/time and forecasted value – time series analysis is the perfect way to analyze the dataset. At first, we only considered the date/time column and the PM2.5 column to see the predictive power of just past PM2.5 values. The time series plot of the PM2.5 values and the logged time series are given below:

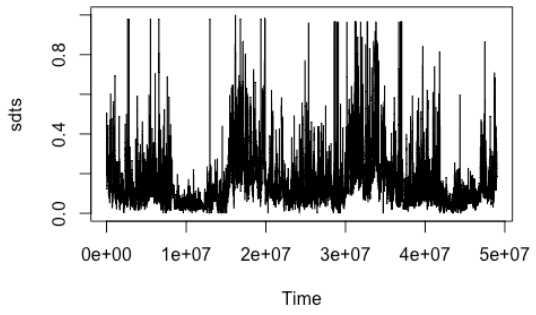



**Figure 6. PM2.5 vizualised as time series**

The dataset was checked for stationarity. Series has a constant mean, variance and auto-covariance that does not depend on time. The statistical test used for this is the Dicky Fuller test. Running the test informed us that the time series was indeed stationary (p-value < 0.01). Next, the dataset was split into training set with 13501 rows and test set with 88 rows. ARIMA

modelling was used to fit the train set and then we calculated RMSE on the test set. Best fit was found using an ARIMA(1,1,2) model. The fit is visualized below:

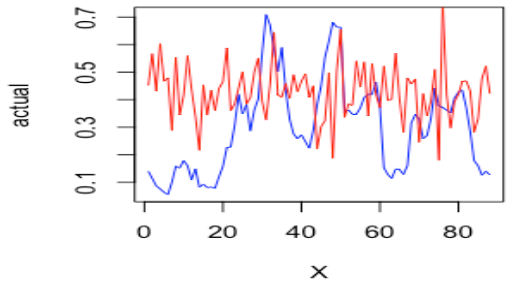



**Figure 7.**

Test RMSE (calculate on min-max scaled data) for the model is 2.226034. We thought the prediction was naïve because it only made use of one feature to do the prediction so this motivated us to try Vector Auto Regression (VAR). This is a method for multi-variate time series analysis. Hence, in this case all of the features were used. We attempted VAR analysis to obtain following fit for PM2.5:

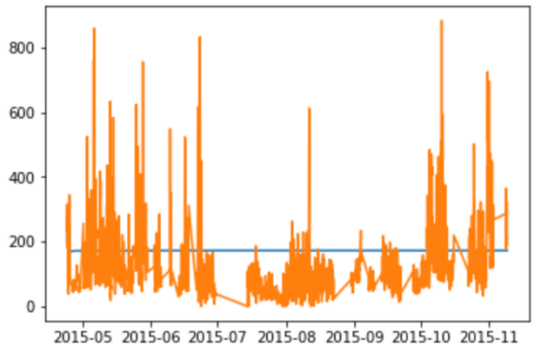



**Figure 8**

The RMSE on min-max scaled data for this fit was 0.112 however, was once again spectacularly naïve. This motivated us to use deeper methods which are detailed as follows.

### 4.3 RNN Model

Secondly, we make use of recurrent neural network Following the empirical results of [9], we try tuning the learning rate first and then tune others. Then later on we tune other hyper parameters like batch size, number of hidden units, input sequence length and number of layers stacked in encoder network.

Focus is being given on the newly articulated temporal pattern attention mechanism since the results of it being better than traditional attention mechanism and plain RNN models for multi-variate time-series forecasting has been empirically proven in [10]. Thus, we try to get the best achievable performance by tuning the hyper parameters. Number of epochs for training had been selected low so that we could see if better performance could be achieved even with less training as compared to models which we encountered in several other research projects on commonly used Beijing dataset. This fact is validated using validation data and later on testing the trained model on test data. Mean Absoluter Error is the loss function that is being used in all our models.

#### 4.3.1. Seq2Seq Encoder-decoder RNN Model

Secondly, encoder-decoder RNN is being used to model this time-series problem by taking data of configured number of hours for Beijing dataset and the result is as shown below. The concept of guided or teacher training is used which is basically utilized during the time of training. According to it, the true output of each of the cells in decoder network is fed as input to next cell. This enables faster training. While testing we feed the predicted output as input to next cell decoder network. Figure 9 gives an idea of how vanilla seq2seq encoder-decoder works. The final hidden state output of the last cell of encoder network basically represents the input sequence provided. The decoder network takes this final hidden state output as initial hidden state input to its first cell. This is as formulated in [3] by Sutskever et al.

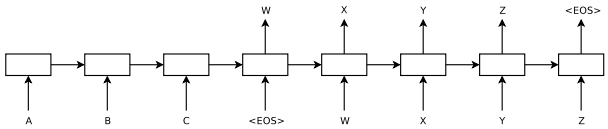



**Figure 9. Vanilla Seq2Seq Encoder-Decoder**

For example, this figure illustrates having as input a sequence A-> B -> C and giving output sequence as W -> X -> Y -> Z. For Beijing dataset, we could think each of A, B and C being hourly basis consecutive 8-D input where those 8 dimensions represent the 8 features of dataset. Here, each of W, X, Y and Z is 1-D output where that value represents the PM2.5 level.

#### 4.3.2 Bahdanau Attention-based Encoder-decoder RNN Model

This attention mechanism is based on the idea suggested by Bahdanau in which bi-directional encoder is used in order to consider the dependencies not only of the past but also of the future time steps. Using bi-directional LSTM would help get a rich set of features in terms of concatenated hidden state output vectors which includes the hidden state output vectors generated from both the directions of our

encoder bi-directional LSTM network [14]. For obvious reasons, bi-directional LSTM doesn't share weights since it needs to learn different patterns - one in forward direction and other in backward direction.

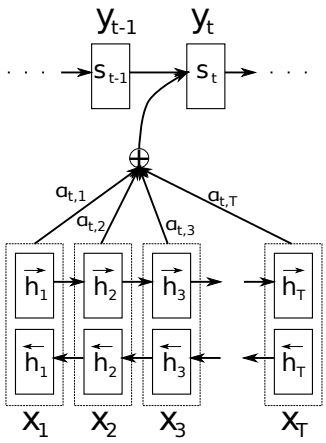



**Figure 9**

The attention mechanism is basically used to weigh in the contributions of each time step in the input time-series input towards predicting correct output. Below are the equations for the same.

$$\alpha_i = \frac{\exp(f(h_i, h_t))}{\sum_{j=1}^{t-1} \exp(f(h_j, h_t))}$$

$$v_t = \sum_{i=1}^{t-1} \alpha_i h_i.$$

This attention mechanism will capture dependencies across time steps. It is seen here that if we take last 10 hours data it is sufficient enough to capture

4.3.3 Temporal Pattern Attention-based RNN

In this newly formulate technique, convolutional neural network is used to capture the time-invariant temporal patterns, basically the periodic nature of change in values of each feature as can be seen by plotting the values of features of data from 2010 - 2014. We see cyclic trends in all features as plotted in Figure 10. Therefore, this problem makes it very much a valid use case for the state-of-the-art attention mechanism formulated in [10]. We have made use of the same network architecture designed for temporal pattern attention mechanism.

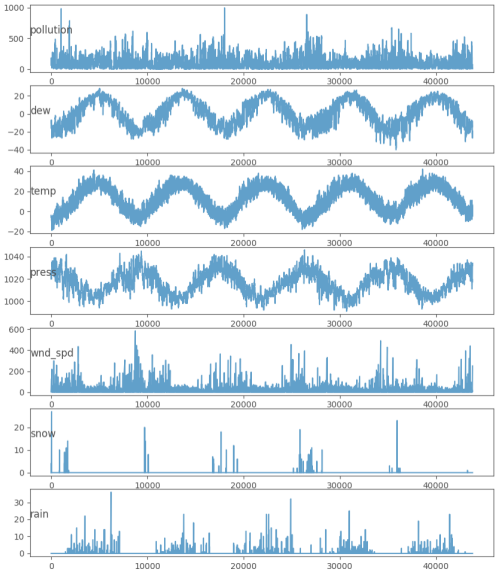



**Figure 10 For Beijing Data Set**

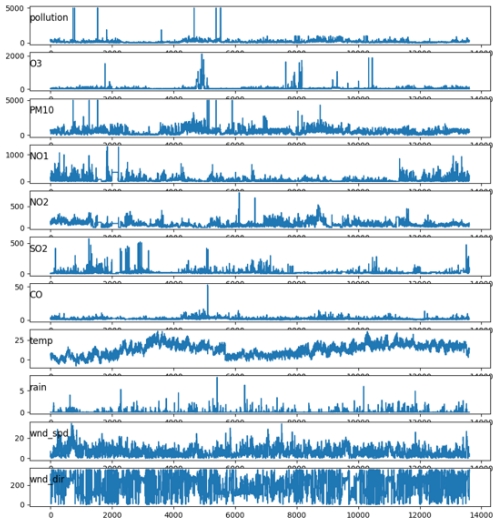



**Figure 11 For Delhi Data Set**

This is the architecture used as show in Figure 13.

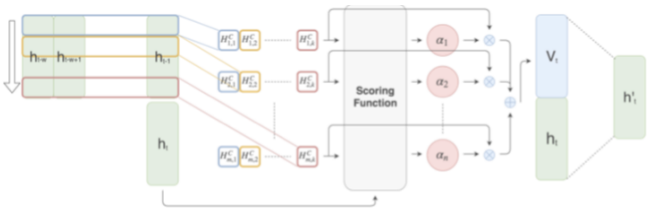



**Figure 12**

Basically, green coloured hidden states of encoder network on the left in Figure 12.

Here **m** value represents the number of hidden units of encoder network. Here the features of the hidden states, **h**, of LSTM cells in encoder are being attended to for deciding indirectly which feature of the input sequence has more contribution towards the predicting output. Two things are happening here. First, **k** number of convolution kernels (C) are being used via convolutional neural network that takes input

values of each dimension of **w** number of hidden state vectors of encoder network. This **w** is our attention length.

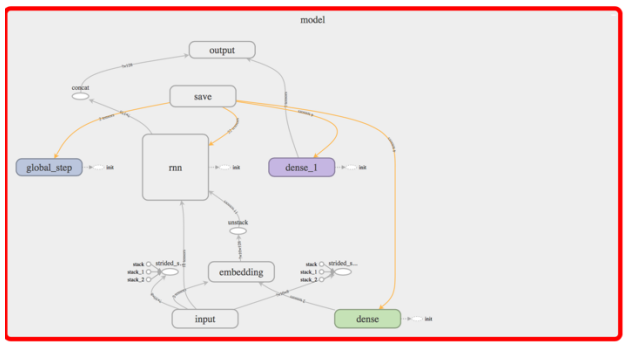



**Figure 13. Overview of computational graph of model used**

**Figure 14**

Figure 14 gives an overview of the computational graph of attention mechanism used. This below equation represents the conv2d layer in the Figure 14 above.

$$H_{i,j}^{C} = \sum_{l=1}^{w} H_{i,(t-w-1+l)} \times C_{j,T-w+l}.$$

This term $W_a h_t$ in the below equation represents the first pink-coloured dense layer and shown in the Figure 14 above. The output is later multiplied with $H_i^C$ as shown below.

$$f(H_i^C, h_t) = (H_i^C)^\top W_a h_t,$$

$$\alpha_i = sigmoid(f(H_i^C, h_t)).$$

Sigmoid activation is used to calculate attention values because it would enable being attentive to multiple time-series belonging to dimension of hidden state vectors of encoder network.

$$v_t = \sum_{i=1}^{m} \alpha_i H_i^C.$$

This equation above represents the context vector that we will be concatenated with previous hidden state vector and fed to a dense layer which is represented by the equation below:

$$h_t' = W_h h_t + W_v v_t,$$

Finally, another dense layer is used outside **attention** block which is named **attn_output_projection** as represented by equation below:

$$y_{t-1+\Delta} = W_{h'} h_t',$$

The Δ here represents horizon which is basically how many time steps ahead we want to predict the value of. Here in our model we have taken Δ = 1 in order see the immediate next pollution level, PM2.5. The output using this architecture actually predicts all the feature values. We take the feature value corresponding to pollution level from that multi-variate single output.

## 5. RESULTS AND ANALYSIS

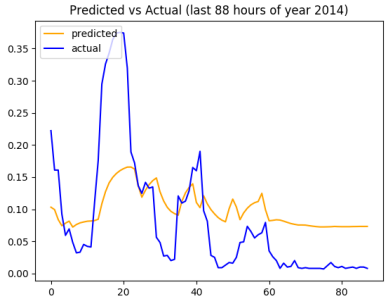
**Note: Our baseline parameters are as stated. Until and unless stated if we talk about changing a parameter then rest of the parameters will have have values as stated below. Here in each epoch, we ensure all batches (number of batches = size of data set divivded by batch size) of data of size, batch_size is seen and Adam optimization is performed and then move on to next epoch for training.**

1. **epochs = 15**
2. **learning rate, lr = 0.005**
3. **input sequence length = 10**
4. **output sequence length = 1 (basically just the PM2.5 value in the next hour given the past 10 hours of data**

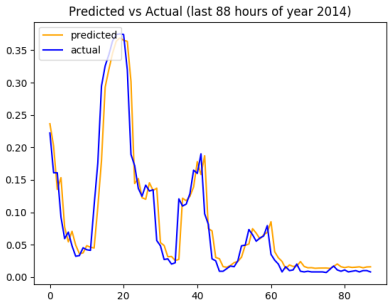
### 5.1. Beijing Dataset

| **Method (LSTM cell used)** | **Test RMSE** |
|---|---|
| Plain Enc - Dec RNN | 90.532 and number of parameters = 138369<br>(40.234 when epochs = 50)<br>(34.075 when epochs = 100)<br>**(23.384* when epochs = 5475)**<br>(23.558 when epochs = $15x10^3$) |
| Bahdanau attention-based RNN | **24.251* and # of trainable parameters: 1023000**<br>(31.592 when epochs = 100)<br>(25.710 when epochs = 20)<br>(24.230 when lr = 0.002)<br>(24.637 when lr = 0.01) |
| Temporal Pattern attention-based RNN | **23.121***<br>(23.5 when epochs = 15) |

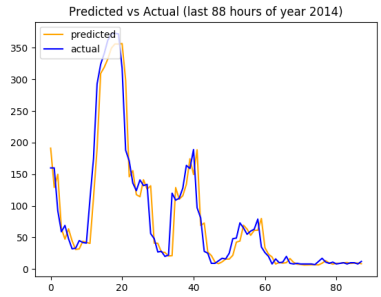
*** indicates best possible value achieved after experimentation**



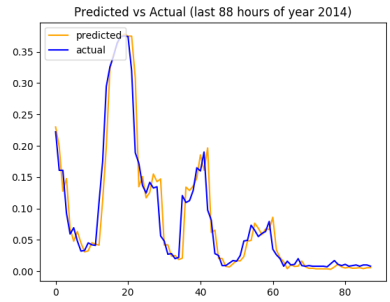

**Plot for LSTM seq2seq encoder-decoder model (epochs = 10 where in each epoch we take gradient of one batch size of data)**




**Plot for LSTM seq2seq encoder-decoder model (epochs = 5475 where in each epoch we take gradient of one batch size of data)**



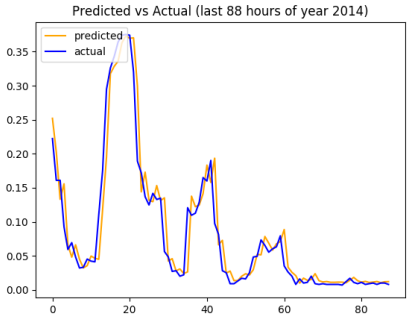

**Plot for LSTM Bahdanau attention-based RNN model (epochs = 10)**



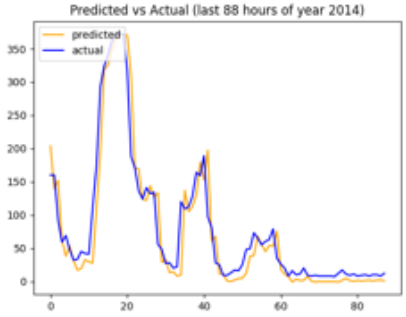

**Plot for LSTM temporal pattern attention-based RNN model (epochs = 10)**

| Method (GRU cell used) | Test RMSE |
|---|---|
| Seq2Seq Enc - Dec RNN | **(23.841* when epochs = 5475)** |
| Bahdanau attention-based RNN | **25.512*** |
| Temporal Pattern attention-based RNN | **23.5*** |

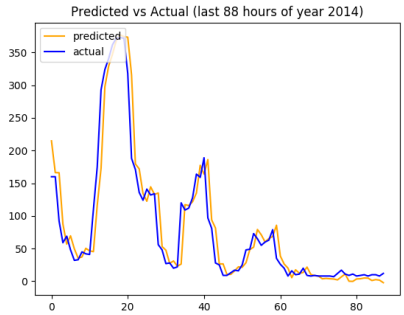
*** indicates best possible value achieved after experimentation**



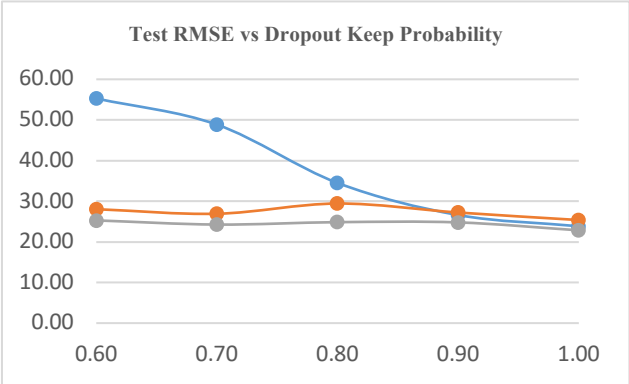

**Plot for LSTM seq2seq encoder-decoder model (epochs = 5475 where in each epoch we take gradient of one batch size of data)**




**Plot for GRU Bahdanau attention-based RNN model (epochs = 10)**




**Plot for LSTM temporal pattern attention-based RNN model (epochs = 15)**




**Blue curve: Seq2Seq Enc-Dec RNN, Orange curve: Bahdanau Attention RNN and Grey line: Temporal Pattern Attention RNN**
**Figure 15**

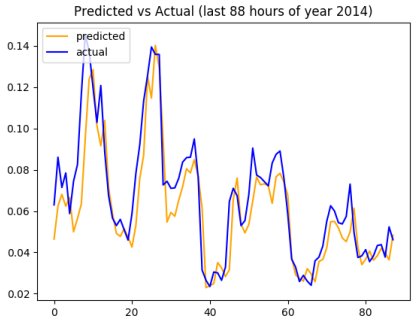
Also, when the keep probability for drop out is increased a bit we see an increase in error for temporal pattern attention-based model. This can be explained by the fact that the model overfits a bit on the training set losing some generalization ability. When GRU unit is used instead of LSTM unit we also see a decrease in time taken to train the model. As per

Figure 15, as the keep probability in context of drop out increases, we see there is decrease in error for all models. This implies, there as such no requirement of drop out regularization to do. Also, it can be seen that temporal attention mechanims performs better than the other two models at all value of keep probability and has almost constant error rate implying more generalization ability of this attention mechanism. And also encoder-decoder performs better when Bahdanau attention mechanism.

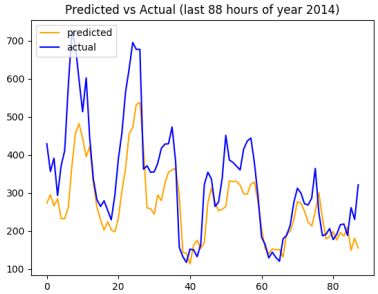
## 5.2. Delhi Dataset

Here to the values of parameters are started with the same values as for Beijing dataset. However, limitations here were there wasn't continous data avaialable on an hourly basis like it was available for Beijing dataset. So, there would be some added error involved in modelling.

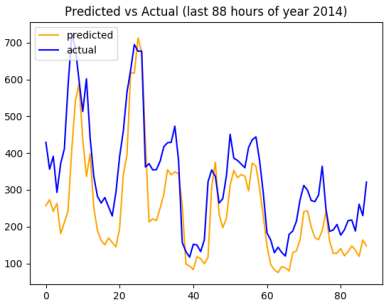
| Method (LSTM cell used) | Test RMSE |
|---|---|
| Plain Encoder-decoder RNN | (**60.701*** when epochs = 5000 and number of parameters = 138369) |
| Bahdanau attention-based RNN | **(69.071 # epochs= 43 and # parameters: 1023000)** |
| Temporal Pattern attention-based RNN | **67.954*** when epochs = 43 and number of parameters = 245867 |




**Plot for seq2seq encoder-decoder model (epochs = 5000 where in each epoch we take gradient of one batch size of data)**




**Plot for temporal pattern attention-based RNN model (epochs = 43)**




**Plot for Bahdanau attention-based RNN model (epochs = 43)**

Initialization are supposed to be done in such a way that we decrease the variance of the weights involved for each layer so that they don't get saturated while training. Otherwise, it might cause slowdown in training. Biases can be initialized to zero since asymmetry breaking is provided by small random numbers in weights.

Adam optimizer is being used and the reasons behind it are as follows:

Adam optimizer takes care of per-parameter learning rate improving performance on non-convex problems like the air pollution forecasting with sparse gradients and each rate is adapted based on average of recent gradients ensuring that it does well on noisy problems (concept of momentum). It also makes use of second moments of gradients for adapting each parameter's learning rate.

**He initialization** technique is used for weights' initializations of a layer whenever **ReLu** activation functions are used in that layer as per [22]. Here we are using truncated normal version of it.

$$\sqrt{\frac{2}{size^{[l-1]}}}$$

$W^{[l]}$ = *np.random.randn(size_l, size_l-1) * np.sqrt(2/size_l-1)*

**Glorot initialization** technique is used for weights' initializations of a layer whenever **tanh**, **sigmoid** and **softmax** activation functions are used in that layer as per [23]. Here we are using truncated normal version of it. Also, it is used in the case whenever a dense or CNN type layer is used.

$$\sqrt{\frac{2}{size^{[l-1]} + size^{[l]}}}$$

$W^{[l]}$ = *np.random.randn(size_l, size_l-1) * np.sqrt(2 / (size_l-1 + size_l) )*

## 5.3 Temporal Pattern Attention-based RNN

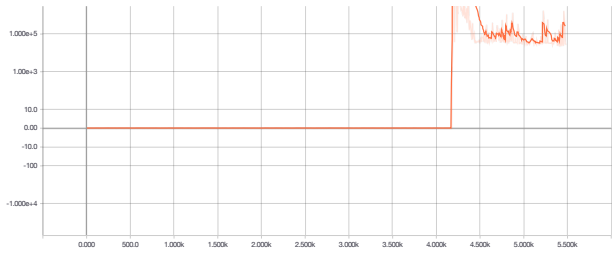
### 5.3.1 Beijing Dataset

For, temporal pattern attention based RNN model, when we increase the number of layers we see that we need to decrease the value of learning rate implying that it becomes more sensitive to changing gradients. If we use the same learning rate of 0.005 when number of layers is increased to two and keep all other hyper parameters same, the loss function varies a lot indicating that learning rate needs to be reduced further. However, there wasn't any significant change in RMSE value of test data even after increasing the number of layers. This indicates that the architecture itself is complex enough to capture the inter-dependencies across time-series of features of dataset. There seems to be slight improvement in accuracy over what seq2seq encoder-decoder without attention could do.

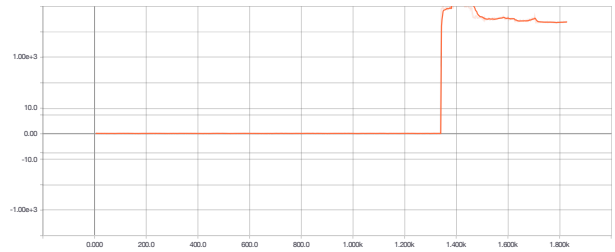
Also, when decreasing the batch-size it causes our temporal pattern attention RNN model to not perform well since we have taken fewer number of epochs and larger batch size is able to average the varying gradients to have a better movement towards optimum. If we had taken smaller batch size and tried with larger number of epochs we could have achieved probably the same results.

In the paper [10], encoder-decoder without attention was giving better accuracy than Luong attention-based model in terms of RMSE and other metrics like MAE. It could be observed here that encoder-decoder without attention basically performs better than Bahdanau attention-based RNN model as well.

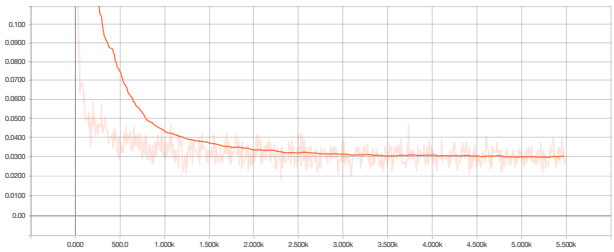
Also, to validate their original architecture, where gradient clipping was used, first testing was done without gradient clipping and if epochs is increased we get graph of loss function as below. It has reached minimum already but when further epoch is run then it tends to go far away from optimum. This implies that we need gradient clipping in order to ensure we do not move out of the position of minimum in loss function. Graphs below are smoothened versions of original graphs as generated in TensorBoard.



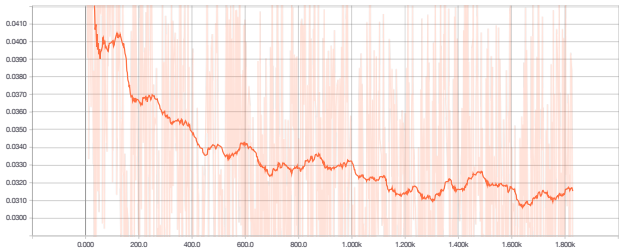
**Training loss when epochs = 15**



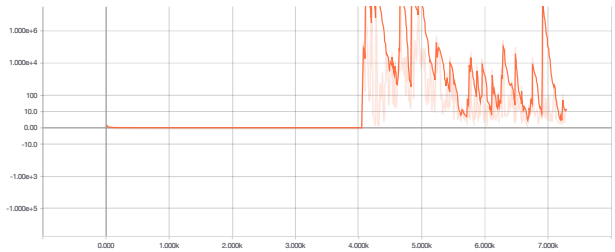
**Validation loss when epochs = 15**



**Training loss when epochs = 15 and gradient clipping is done**



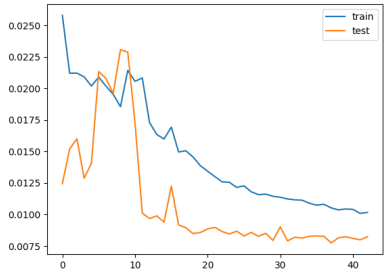
**Validation loss when epochs = 15 and gradient clipping is done**



**Training loss when epochs = 20 and gradient clipping is done**

Gradient clipping tried to control the overshoot in the movement from optimum value of loss function. Also, increasing the number of epochs has no significant effect on getting to a further minimum value indicating that we had to stop early. Running 10 - 15 epochs was sufficient enough. Also, if we increase the number of epochs too much it goes out of optimum and error increases.

5.3.2 Delhi Dataset




**Training versus validation loss for Bahdanau attention-based RNN model (epochs = 43)**

There is a need to increase number of epochs here with the same hyper parameters for India dataset and if we look closely there is less of periodicity in India dataset so performs worse than encoder-decoder one.

## 6. CONCLUSION

It can be concluded from the comparison of the different RNN models that the state-of-the-art temporal pattern attention mechanism gives the best performance amongst all models tested here for air pollution forecasting situation when there is pattern across time steps of each feature. This validates the results implied in [10]. Also, encoder-decoder without attention was giving better accuracy than Luong attention-based model in terms of RMSE and other metrics like MAE. It could be observed here that encoder-decoder without attention basically performs better than Bahdanau attention-based RNN model as well. One reason could be that Bahdanau attention tends to over fit the training dataset a little bit creating a little more variance when being tested its generalization on test data. The results that we achieved using seq2seq encoder-decoder are better than what is being talked about in [6]. It is likely because of the change in how we initialized our weights using glorot initializer for the dense layer which is stacked on top of each cell of decoder network. If no gradient clipping had been done then it is very likely not to return to the optimum value of loss function because gradients are huge near optimum value as could be concluded from conducted experiments.

For India dataset however, which seems less periodic as compared to Beijing's dataset, temporal pattern attention RNN doesn't perform better on test data as compared to how seq2seq encoder-decoder without attention has performed. Possibly, because of lack of periodicity which is clearly observable in Beijing. The results for India dataset show that attention being given to time steps rather than on temporal patterns of features has effectively benefitted in prediction accuracy. However, the resutls for Delhi dataset contradicts the conclusion of [10] that even for non-pattern based time-series data temporal pattern attention-based model performs better than seq2seq encoder-decoder model without attention.

## 7. ACKNOWLEDGEMENT

A lot was learnt during the project given we had to start learning the concepts of rurrent neural network from scratch. We first had to understand it as much as possible by reading the concerned theory. Mostly important to learn was how to create models. It is necessary to develop those inuitions when deciding upon an architecture of the model one creates and how to go about hyper-parameter tuning. We had tried to give our best with the same. The paper [10] which talks about this attention mechanism we heavily discussed in this report really helped us get a good hang of how to go about coding and also learning the framework of tensorflow and conducting hyper-parameter tuning.

## 8. FUTURE WORK

We would conduct similar modeling experiments extensively for India dataset going ahead and also study about including spatial attention as well for air pollution forecasting. On this line, the paper we would begin to study and understand is **Cheng, Weiyu et al. "A Neural Attention Model for Urban Air Quality Inference: Learning the Weights of Monitoring Stations." AAAI (2018).** Morever, air pollution forecasting as a problem for machine learning to solve seems to have gain traction just recently.

## 9. REFERENCES

[1] Chinnaswamy, Anitha & Naguib, Raouf & T. Nguyen, Q & Trodd, Nigel & Marshall, Ian & Yaacob, Norlaily & Santos, Gil Nonato & Vallar, Edgar & Galvez, Maria Cecilia & H. Shaker, M. (2014). Air Pollution in Bangalore, India: A Six-Year Trend and Health Implication Analysis.

[2] Delhi world's most polluted city: Study, May 08 2014 from https://www.hindustantimes.com/india/delhi-world-s-most-polluted-city-study/story-Kqiz2WDZ8muWya6MJpbGPM.html

[3] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, "Sequence to sequence learning with neural

networks", *Advances in neural information processing systems, 3104-3112,* 2014

[4] Sepp Hochreiter, Jurgen Schmidhuber, "Long Short-Term Memory", *Neural Computation 9(8):1735-1780,* 1997

[5] Alex Shertinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network", Available: https://arxiv.org/pdf/1808.03314.pdf

[6] Vikram Narasimha Reddy, Shrestha Mohanty, "Deep Air : Forecasting Air Pollution in Beijing, China", Available: https://www.ischool.berkeley.edu/sites/default/files/sproject_attachments/deep-air-forecasting_final.pdf, 2017

[7] Tien-Cuong Bui, Van-Duc Le, Sang-Kyun Cha, "A Deep Learning Approach for Forecasting Air Pollution in South Korea Using LSTM", Available: https://arxiv.org/pdf/1804.07891v3.pdf, 2018

[8] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation", *Empirical Methods in Natural Language Processing*, 2014

[9] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber, "LSTM: A Search Space Odyssey", *IEEE Transactions on Neural Networks and Learning Systems,* vol. 28, issue: 10, pp: 2222 - 2232, Oct. 2017

[10] Shun-Yao Shih, Fan-Keng Sun, Hung-yi Lee, "Temporal Pattern Attention for Multivariate Time Series Forecasting", *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases,* 2019

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research 15 1929-1958,* 2014

[12] Guoqiang Zhang, B. Eddy Patuwo, Michael Y. Hu, "Forecasting with artificial neural networks: The state of the art", *International Journal of Forecasting* vol. 14, issue 1, pp 35–62, 1998

[13] Yasin Akın Ayturan, Zeynep Cansu Ayturan, Hüseyin Oktay Altun, "Air Pollution Modelling with Deep Learning: A Review", *Int. J. of Environmental Pollution & Environmental Modelling,* vol. 1(3), pp 58-62, 2018

[14] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, "Neural machine translation by jointly learning to align and translate", *International Conference on Learning Representations*, 2015

[15] Ibrahim Kok, Mehmet Ulvi Simsek, Suat Ozdemir, "A deep learning model for air quality prediction in smart cities", *IEEE International Conference on Big Data (Big Data)*, 2017

[16] Minh-Thang Luong, Hieu Pham, Christopher D. Manning, "Effective approaches to attention-based neural machine translation", *Empirical Methods in Natural Language Processing,* 2015

[17] Athira Va, Geetha Pb, Vinayakumar Rab, Soman K P, "DeepAirNet: Applying Recurrent Networks for Air Quality", *International Conference on Computational Intelligence and Data Science*, 2018

[18] Sebastian Ruder, "An overview of gradient descent optimization algorithms" Available (Blog version): http://ruder.io/optimizing-gradient-descent, 2017

[19] Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep learning", *The MIT Press*, 2016

[20] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, Yoshua Bengio, "On the properties of neural machine translation: encoder-decoder approaches", *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Empirical Methods in Natural Language Processing,* 2014

[21] Rafal Jozefowicz, Wojciech Zaremba, Ilya Sutskever, "An empi rical exploration of recurrent network architectures", *International Conference*

*on Machine Learning*, Francis Bach and David Blei (Eds.), vol. 37., pp 2342-2350

[22] Kaiming He et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", *IEEE International Conference on Computer Vision*, 2015

[23] Xavier Glorot, Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks", *International Conference on Artificial Intelligence and Statistics*, PMLR, vol. 9, pp: 249-256, 2010

## 10. DIVISION OF WORK

**Our objective was to explore the sequence modelling using different RNN architectures in this paper for air pollution forecasting.**

1. Abinash Sinha - https://github.com/abinashsinha330/Air-Pollution-Forecasting-using-Machine-Learning-APF-Model The portion related to RNN (Basically whole sections of results, analysis, conclusion, references, theory related to RNN, CNN, LSTM, Background / Related Work related to RNN and LSTM)

2. Dhananjay Muddappa - The portion related to ARIMA