TRANSFER LEARNING FOR TIME SERIES IMPUTATION IN WIRELESS SENSOR NETWORK

A dissertation report submitted in partial fullfilment of the requirement for the award of the degree of

MASTER OF TECHNOLOGY
in
ARTIFICIAL INTELLIGENCE AND ROBOTICS

SUBMITTED BY:
VISHAL PANDEY
15/ICS/076

SUPERVISED BY:
Dr. VIDUSHI SHARMA
(Assistant Professor)

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
GAUTAM BUDDHA UNIVERSITY
GREATER NOIDA - 201312, UTTAR PRADESH, INDIA

MAY, 2020

SCHOOL OF INFORMATION AND COMMUNIOCATION TECHNOLOGY
GAUTAM BUDDHA UNIVERSITY
GREATER NOIDA - 201312

Candidate's Declaration

I here by declare that the work embidied in this dissertation entitled "TRANSFER LEARNING FOR TIME SERIES IMPUTATION IN WIRELESS SENSOR NETWORK" is submitted for the partial fulfillment of the requirements to award the degree of Integrated M.Tech (Artificial Intelligence and Robotics) to School of Information and Communication Technology(ICT), Gautam Buddha Univesity, Freater Noida. It is an authentic record and my own bonafied work has been carried out under the supervision of Dr. Vidushi Sharma, School of ICT. This work is correct to the best of my knowledge and belief and has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma af any university or other institute of higher learning, exceptt where due acknowledgement has been made in the text. Responsibility for any plagiarism related issue stands solely with me.

.

Name of student: Vishal Pandey
Roll No. 15/ICS/076
Signature:
Date:
Place: Greater Noida

SCHOOL OF INFORMATION AND COMMUNIOCATION TECHNOLOGY
GAUTAM BUDDHA UNIVERSITY
GREATER NOIDA - 201312

Certificate

This is to certify that work entitled "TRANSFER LEARNING FOR TIME SERIES IMPUTATION IN WIRELESS SENSOR NETWORK" by "Vishal Pandey" Roll No. 15/ICS/076 Integrated M.Tech final year with specialisation "ARTIFICIAL INTILLEGENCE AND ROBOTICS" has been carried under my supervision.

.

Dr. Vidushi Sharma
Assistant Professor
School Of ICT
Gautam Buddha University
Greater Noida, 201312
(Uttar Pradesh India)

ACKNOWLEDGMENT

Throughout my dissertation, I have been very fortunate to get help and support from many people. I feel immense pleasure in expressing my profound gratitude to my supervisor, Dr. Vidushi Sharma, SOICT, for her guidance, useful suggestion, and for permitting me to carry out this work, which helped me complete the work on time. She inspired me greatly to work on this dissertation. Her willingness to motivate me contributed tremendously to work. I would like to thank her for showing me a few examples related to the topic in my dissertation.

I would like to acknowledge Dr. Anurag Singh Baghel, for his encouragement that boosted my morale and confidence, I would like to express my gratitude and heartfelt thanks to Dr. Neeta Shing, for her excellent supervision and support in his research.

Besides, I would like to thank the authority of the School of Information and Communication Technology (ICT), Gautam Buddha University, for providing a good environment and facilities that helped me complete this dissertation.

I Would like to express my heartfelt thanks to my beloved parents for their blessings and whiches for the successful completion of this work. Last but not least, Almighty God has been the invisible divine force behind my accomplishments.

Vishal Pandey
15/ICS/076

ABSTRACT

IoT is a field that combines the capability of two domains: computer science and electronics. IoT devices are basically sensor nodes (usually wireless) connected to each other and to the central control system through the internet. Various applications IoT is in healthcare, weather monitoring, remote control, collecting data from sensors. Missing data in the wireless sensor network is inevitable and is a major field of study since the start of IoT and WSN. IoT operates in technologies which are lossy networks. There may be many reasons for the loss of data while communication one of the reason may be confession the other reason being a loss in tunning of duty cycles of sender and receiver and other environmental condition in which they operate. Recovering the lost data is important in mission-critical applications such as health care, defense etc. In IoT networks to overcome these issues and regenerate the data machine learning techniques can prove to be beneficial since the IoT networks have constraint devices so robust and complex techniques do not fit in their communication model.

List of Figures

List of Tables

List of abbreviation

SSIM	Sequence to Sequence Imputation Model
LSTM	Long Short Term Memory
MSE	Mean Squared Error
WSN	Wireless Sensor Network
GPS	Global Positioning System
LED	Light Emiting Diode
RF	Radio Frequency
AI	Artificial Intelligence
DL	Deep Learning
RNN	Recurrent Neural Network
TL	Transfer Learning
EMG	Electromyographic
EEG	Electroencephalographic
ARIMA	Autoregressive integrated moving average
VLSW	Variable Length Sliding Window
MQTT	Message Queueing Telementary Transport
COAP	Constrained Application Protocol
PM2.5	Particulate Matter diameter less than 2.5 micrometer
DNN	Deep Neural Network

Contents

Chapter 1

Introduction

1.1 Overview

Wireless sensor network is a type of network which is wireless used for sensing purpose. It is a collections of various amount of sensors available in a network which are connected wirelessly. The main component of the wireless sensor network are sensor nodes. Sensor nodes play an important role in the working of this entire network system. Sensor nodes act as a powerful component for doing various activities like communication, sensing and data interpretation. The wireless network system is a remarkable tool in technology which ease the computational work to a great extend. The use of the wireless sensor network technology helps the work of data interpretation so fast and accurate. The errors causes by human intervention in the process at communication and networking can be completely sabotaged by the use of this technology. Using the concept of wireless communication, network system and various sensor theories makes this entire system work efficiently and with remarkable accuracy.

1.2 Wireless Sensor Network

In recent time their is more ease with communication, and computation because of advances in processor, memory, and radio technology enable small and cheap nodes capable of sensing,. Distributing sensing of environmental having networks of such nodes can coordinate to perform phenomena. With a wide range of applications in areas such as traffic monitoring[1], medical care[2], in hospitable terrain, robotic exploration[3], and agriculture surveillance[4]. The advent of efficient wireless communications and advancement in electronics has enabled the development of low-power, low-cost, and multifunctional wireless sensor nodes that are characterized by miniaturization and integration.

The network uses location information to reduce redundant transmissions, thereby saving energy. The sensor network is divided into virtual grids and each sensor node associates itself with a virtual grid based on its location. Sensor nodes within a virtual grid are classified as either gateway nodes or internal nodes. While gateway nodes are responsible for forwarding the data across virtual grids, internal nodes forward the data within a virtual grid. In order to make communications both reliable and energetic efficient Deep learning approch is used.

Wireless Sensor Networks (WSNs) can be defined as a self-configured and infrastructure less wireless networks to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location or sink where the data can be observed and analysed. A sink or base station acts like an interface between users and the network. One can retrieve required information from the network by injecting queries and gathering results from the sink. Typically a wireless sensor network contains hundreds of thousands of sensor nodes. The sensor nodes can communicate among themselves using radio signals. A wireless sensor node is equipped with sensing and computing devices, radio transceivers and power components. The individual nodes in a wireless sensor network (WSN) are inherently resource constrained: they have limited processing speed, storage capacity, and communication bandwidth. After the sensor nodes are deployed, they are responsible for self-organizing an appropriate network infrastructure often with multi-hop communication with them. Then the onboard sensors start collecting information of interest. Wireless sensor devices also respond to queries sent from a "control site" to perform specific instructions or provide sensing samples. The working mode of the sensor nodes may be either continuous or event driven. Global Positioning System (GPS) and local positioning algorithms can be used to obtain location and positioning information[5]. Wireless sensor devices can be equipped with actuators to "act" upon certain conditions.

Wireless sensor networks (WSNs) enable new applications and require non-conventional paradigms for protocol design due to several constraints. Owing to the requirement for low device complexity together with low energy consumption (i.e. long network lifetime), a proper balance between communication and signal or data processing capabilities must be found. This motivates a huge effort in research activities, standardization process, and industrial investments on this field.

Wireless sensor networks are used in environmental trackings, such as forest detection[6], animal tracking[7], flood detection[8], forecasting and weather prediction[9], and also in commercial applications like seismic activity prediction and monitoring.Irrespective of the application, Wireless Sensor Networks in general can be classified into the following categories[10].

- Static and Mobile WSN.
- Deterministic and Nondeterministic WSN.
- Single Base Station and Multi Base Station WSN.
- Static Base Station and Mobile Base Station WSN.
- Single-hop and Multi-hop WSN.

The components of WSN system are sensor node, rely node, actor node, cluster head, gateway and base station. a. Sensor node: Capable of executing data processing, data gathering and communicating with additional associated nodes in the network.

A sensor node, also known as a node in a sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. A mote is a node but a node is not always a mote.

A Wireless sensor network can be defined as a network of devices that can communicate the information gathered from a monitored field through wireless links[11]. The data is forwarded through multiple nodes, and with a gateway, the data is connected to other networks like wireless Ethernet.

A Wireless sensor network can be defined as a network of devices that can communicate the information gathered from a monitored field through wireless links. The data is forwarded through multiple nodes, and with a gateway, the data is connected to other networks like wireless Ethernet.

There are various features one of them is linearity which represents the relationship between input variations and output variations.Second is frequency response: The frequency response is the range i to the input remains relatively high.Third is reliability.Fourth is accuracy.Fifth is repeatability and other features are size, weight and volume.

A sensor consists of three main components in which the sensing section contains the sensor itself which is based on a particular technology and the processing circuitry converts the physical variable into an electrical variable in which the signal output contains the electronics connected to a control system.The base stations are one or more components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user as they typically forward data from the WSN on to a server.

Sensor nodes are used for constant sensing, event ID, event detection & local control of actuators. They convert physical or monitored signals like temperature, motion or light intensity to electrical signals. They are composed of a transistor, a resistor and an LED. Connect the resistor and LED in series from the positive supply to the collector of the transistor. Connect the emitter of the transistor to the negative terminal of the supply. Create two wires with exposed ends.

Most sensors use wireless RF signals, a hardwired connection or WIFI connectivity. These devices communicate primarily using RF signals. The electrical signals are transmitted to reciever either by wired communication or wireless communication.

Through sensors there is an imporvement in many fields like diagnostics in medical applications; improved performance of energy sources like fuel cells and batteries and solar power; improved health and safety and security for people; sensors for exploring space and the known university; and improved environmental monitoring.

They are classified into three categories: passive, omnidirectional sensors; passive, narrow-beam sensors; and active sensors. Passive sensors sense the data without actually manipulating the environment by active probing. They are self powered; that is, energy is needed only to amplify their analog signal.The most frequently used different types of sensors are classified based on the quantities such as Electric current or Potential or Magnetic or Radio sensors, Humidity sensor, Fluid velocity or Flow sensors, Pressure sensors, Thermal or Heat or Temperature sensors, Proximity sensors, Optical sensors, Position sensors.These sensors connect to a microcontroller (uC), then connect uC to the PC through a USB or RS-232 or Bluetooth.

Challenges in such WSN include high bandwidth demand, high energy consumption, quality of service (QoS) provisioning, data processing and compressing techniques, and cross-layer design. physical environment[12]. Mobile nodes have the ability to sense, compute, and communicate like static nodes.

Wireless network is basically based on sensory nodes. These sensory nodes act as the information carrier in the whole network system . these sensory nodes are scattered in the field called sensory field from where the network system does the work of data collection. The sensory field is the place from where the sensory nodes collect the data required for the designed system. The data after collection is transmitted and requires actions are taken on it and final processing takes place over the transmitted data. The data after the processing from the sensory field is directed toward the sink which is called as 'gateway'. This is how the compelet wireless sensorey network is designed in this arcitectural pattern. This architecture is uses used in building the model to get the required task executed.

1.3 Deep Learning and Data Science

Data Science approach is used to solve the problems related to data. It tells automatically which features to use for calculation. A subset of it is AI. A subset of it is Deep Learning where the same function is applied again and again. Figure 1.1 Describes the classificiation of deep learning.

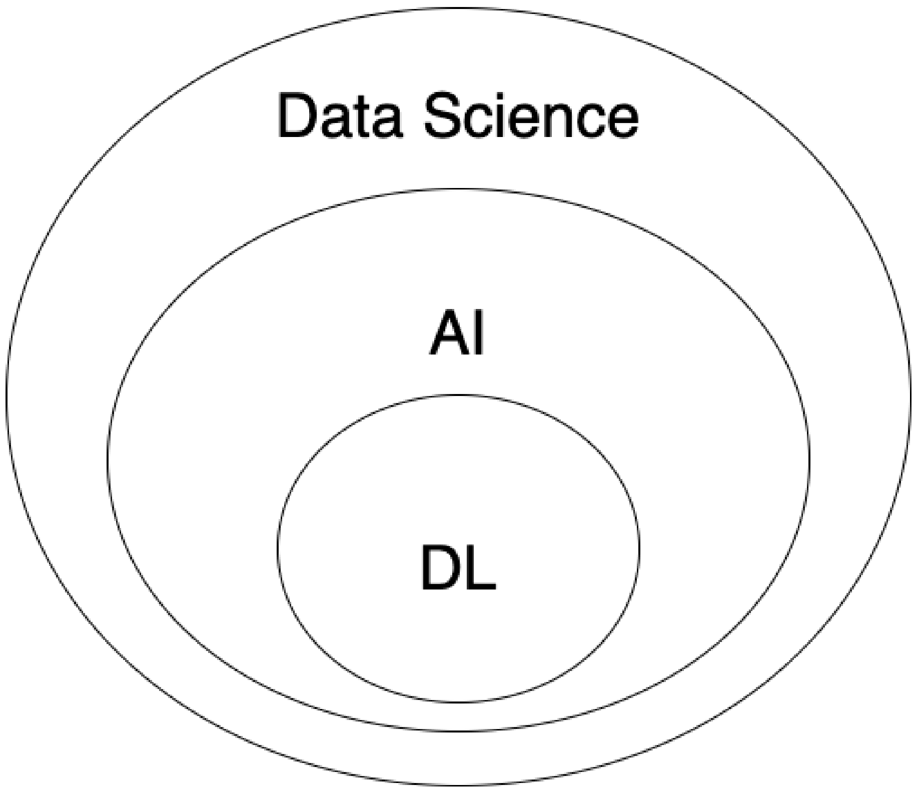


Figure 1.1: Deep Learning Hirarchy

Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain[13]. They used a combination of algorithms and mathematics they called "threshold logic" to match the thought process.

Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data[14]. Deep learning allows machines to solve complex problems even when using a data set that is very diverse, unstructured and inter-connected. If there are more than three layers (including input and output) it will qualify as “deep” learning.

Deep learning systems represent the first time computers can understand images at a useful level and reasonable cost. The majority of jobs involve some form of visual perception. ie detection and classification of visual information.

Deep Learning uses a Neural Network to imitate animal intelligence. There are three types of layers of neurons in a neural network: the Input Layer, the Hidden Layer(s), and the Output Layer. Connections between neurons are associated with a weight, dictating the importance of the input value.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound.

Deep learning networks can be successfully applied to big data for knowledge discovery, knowledge application, and knowledge-based prediction. In other words, deep learning can be a powerful engine for producing actionable results

A Deep feature is a consistent response of a node or layer within a hierarchical model to an input that gives a response that’s relevant to the model’s final output. One feature is considered “deeper” than another depending on how early in the decision tree or other framework the response is activated.

Deep learning is an increasingly popular subset of machine learning. Deep learning models are built using neural networks. A neural network takes in inputs, which are then processed in hidden layers using weights that are adjusted during training. Then the model spits out a prediction. The number of layers of neurons stacked is greater than what’s done in Machine Learning, so you now have deep neural networks. Deep Learning is a subset of Machine Learning. Deep learning consists of neural networks with many hidden layers which makes the layers deep. Deep learning then can be defined as neural networks with a large number of parameters.

The layers in one of four fundamental network architectures[15]:

- Unsupervised Pre-trained Networks.
- Convolutional Neural Networks.
- Recurrent Neural Networks.
- Recursive Neural Networks.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation. Its applications are used in industries from automated driving to medical devices. Automated Driving: Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents. he ’deep’ in deep learning simply refers to having multiple ’hidden’ layers; that is layers that aren’t the input, or the final output, but somewhere in between.

One of the Deep learning algorithm CNN is suitable for spatial data such as images[16]. RNN is suitable for temporal data, also called sequential data[17]. CNN is considered to be more powerful than RNN. They unlike feed-forward neural networks - can use their internal memory to process arbitrary sequences of inputs.

More layers can be better but also harder to train. As adding a second layer only improves the accuracy by approx 0.2% (0.9807 vs. 0.9819) after 10 epochs.

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning[18]. LSTM networks are well-suited to classifying, processing and making predictions based on time series data since there can be lags of unknown duration between important events in a time series.

Human mankind always wanted to have the machines which would have the ability to do the reasoning aspect by it itself. Building such a intellectual machine which could do all reasoning work by itself was a big challenge. Having a machine which could function like a human brain and have the capacity to do the logical operations all by itself was huge challenge. A machine which could actually behave like a human brain. Well, that desire of human mankind to have such a computing machine lead to emergence of the concept of AI i.e artificial intelligence. AI is a concept which acted as a great tool to give a huge strength to the computing machines these days. AI has a capacity to work exactly like how human neural system work. Well, the alogrithum used to design any AI concept was inspired by human neural system.

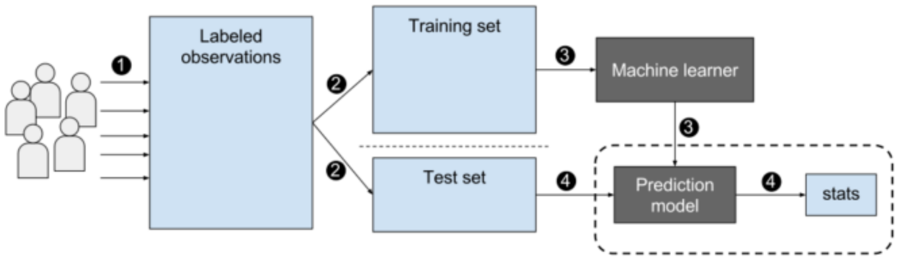


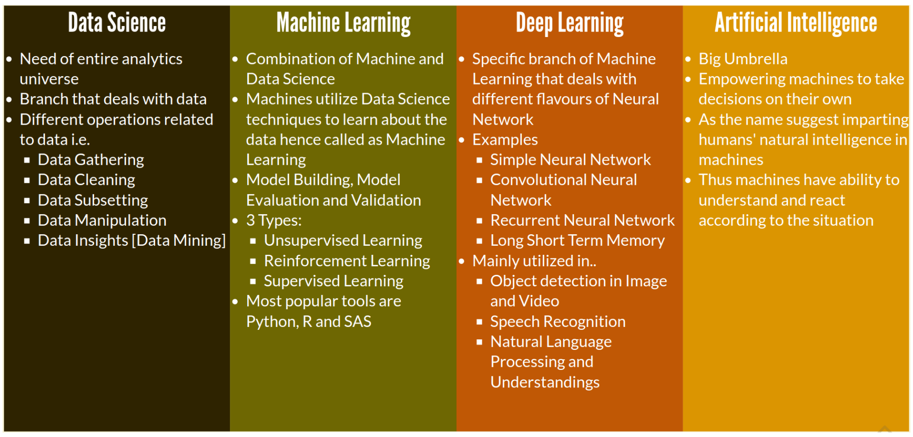
Figure 1.2: Supervised Machine Learning

As the research progressed in the field of AI which lead to have remarkably improved piece of technology , the concept of machine learning came into existence alongside with all of this. Machine learning is an important part of AI.machine learning is a concept where the machine has the capacity to learn by itself from the given set of data in a particular manner.

Earlier the information stored in the system and how the system used to work was based on hard core computational language. The language was more in the numeric form which not giving the freedome of any other sort of interpretation. The language used for these machinery was more scientific. To have a reasoning capability in the machine, there was a huge need of thee development of some different sort of language system. Due to this need, the concept of AI came into existence.

AI gave the computers the reasoning ability and analyise the data using thise reasoning logics.

But later feeding the data everytime to the syste to check the result became a proble. So as to solve such problem, people needed something which could actually self learn the data around the world. So as to accomplish that task, the concept of machine learning came ino existence. Where the concept of machine learning opened the gateway for the machine to self identify and learn from the data available to them by them selves and act accurately over it.



Data Science	Machine Learning	Deep Learning	Artificial Intelligence
• Need of entire analytics universe • Branch that deals with data • Different operations related to data i.e. ▪ Data Gathering ▪ Data Cleaning ▪ Data Subsetting ▪ Data Manipulation ▪ Data Insights [Data Mining]	• Combination of Machine and Data Science • Machines utilize Data Science techniques to learn about the data hence called as Machine Learning • Model Building, Model Evaluation and Validation • 3 Types: ▪ Unsupervised Learning ▪ Reinforcement Learning ▪ Supervised Learning • Most popular tools are Python, R and SAS	• Specific branch of Machine Learning that deals with different flavours of Neural Network • Examples ▪ Simple Neural Network ▪ Convolutional Neural Network ▪ Recurrent Neural Network ▪ Long Short Term Memory • Mainly utilized in.. ▪ Object detection in Image and Video ▪ Speech Recognition ▪ Natural Language Processing and Understandings	• Big Umbrella • Empowering machines to take decisions on their own • As the name suggest imparting humans' natural intelligence in machines • Thus machines have ability to understand and react according to the situation

Figure 1.3: Difference between Data Science, ML, DL and, AI

After this, to solve more complexed problem using AI and using machine learning, we need more sophisticated tools. From where exactly the concept tof deep learning came into existence. Deep learning is the part of machine learning you can say the eep learning is the subset of the machine learning.

Deep learning is majorly used to interpret the data and come up with the conclusions where there is huge lack of segregated data. Where the data available to us is not in a presentable format and completely unorganized, there deep learning act a strong tool to do our analysis over those type of data and retrieve something valuable out of it

1.4 Transfer Learning

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem[19]. For example, knowledge gained while learning to recognize cats could apply when trying to recognize tigers. This area of research bears some relation to the long history of psychological literature on transfer of learning but from the practical standpoint, reusing or transferring information from previously learned tasks for the learning of new tasks has the potential to significantly improve the sample efficiency of a reinforcement learning agent.

Algorithms are available for transfer learning in Markov logic networks and Bayesian networks. Transfer learning has also been applied to cancer subtype discovery, general game playing, text classification, digit recognition and spam filtering.

In 2020 it was discovered that, due to their similar physical natures, transfer learning is possible between Electromyographic (EMG) signals from the muscles when classifying the behaviours of Electroencephalographic (EEG) brainwaves from the gesture recognition domain to the mental state recognition domain[20]. It was also noted that this relationship worked vice-versa, showing that EEG can likewise be used to classify EMG in addition. The experiments noted that the accuracy of neural networks and convolutional neural networks were improved through transfer learning both at the first epoch and at the asymptote.So algorithms are improved by exposure to another domain.

The main features are: What transfer learning is and how to use it. Common examples of transfer learning in deep learning.When to use transfer learning on your own predictive modelling problems.

Transfer learning is related to problems such as multi-task learning and concept drift and is not exclusively an area of study for deep learning. The transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained. Transfer learning is an optimization, a shortcut to saving time or getting better performance. It is not obvious that there will be a benefit to using transfer learning in the domain until after the model has been developed and evaluated. These features are required for Transfer learning:

Higher start. The initial skill (before refining the model) on the source model is higher than it otherwise would be. Higher slope. The rate of improvement of skill during the training of the source model is steeper than it otherwise would be. Higher asymptote. The converged skill of the trained model is better than it otherwise would be.

Two models of Transfer Learning

- Develop a Model Approach
- Pre-trained Model Approach

Develop a Model Approach

- Select Source Task. You must select a related predictive modelling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.
- Develop Source Model. Next, you must develop a skilful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.
- Reuse Model. The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modelling technique used.
- Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

Pre-trained Model Approach

- **Select the Source Model.** A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
- **Reuse Model.** The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modelling technique used.

- **Tune Model.** Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

Transfer learning is a different type of machine learning method. As the concept of machine learning is really important to have good AI based programme and to have good analytical software. As explained in the starting section about the emergence of machine learning. Elaborating the importance of the machine learning here.

It is very important to understand the kind of technique we are using for the machine learning process. As the technique we use for the machine learning process plays a very big part at how quality results we will receive and how fast the execution of the work will take place.

So there are various type of machine learning techniques in the market.

Where 1. Supervised learning – in this learning the machine learns by flashing the certain set of information infront of it. It is like learning like a small kid where the kid learns by having flash cards.

Some common application of supervised learning are as follows :-

Advertisement clickablity Identification of spam Facial recoginition

Unsupervised learning –in this sort of learning the machine is feed with unorganised data. After feeding the system with the unorganized , the system is allowed to identify the difference among them by itself.

Some common application of unsupervised learning are as follows :-

Recommendation system Buying behaviour Common issue rectifier

Reinforcement learning- In this type of learning the machine learn by iterating the given data again and again. The machine learning by through same type of data various times and becoming more accurate about it.

Some common application of reinforcment learning are as follows :-

Video games Industrial management Management of resources

Here we will talk about following things about transfer learning What is exactly transfer learning Real time application of transfer learning Productivity impact by using transfer learning.

1.5 Challenges and Problem Identification

Limitations in applying the prediction model in the production environment.

1. Sensor nodes are resource constraint so the prediction model cannot be deployed on sensor nodes.

2. Cluster head can have more resources in comparison to the individual nodes but the machine learning model also cannot be applied there because a cluster head needs to process data collected from a number of different sensor nodes, which intern leave no option to deploy on the cluster head.

3. Machine learning algorithm needs heavy computing generally GPUs are used for the training of the machine learning models, and to be applied in the production environment you have to continue training and updating the model to better adapt for the upcoming missing values, so remote devices are not the choice for the deployment.

4. For training a supervised deep-learning algorithm we need a huge amount of labeled data, which is rare in the IoT and WSN environment.

1.6 Motivation

Cloud computing is a branch of computer science where a network of computers connected to the internet (called cloud) and available for use as per demand.

As there is no limitation in the computing capacity and resources in the cloud and IoT devices are connected to the internet the IoT devices can send their data to these cloud in real-time for processing and get a response (corrective measure) simultaneously.

1.7 Research Objective

The specific objective of this dissertation are as follows:

1. To find an approach to apply deep learning for imputing missing time series data generated by wireless sensor network and also for time series forecasting when there is a scarcity of data produced by the wireless sensor network.

2. Applying and studying different deep learning algorithm for time series imputation and forecasting in wireless sensor network.

3. Applying transfer learning in wireless sensor network and study the improvement.

1.8 Methodology

To complete this task the jupyter (a free and open source integrated developement environment (IDE) is being used for python). The resutls have been obtained by a set of code containing predefined functions in various packages or libraries available in python.

1.9 Thesis Organisation

This thesis consists of 6 chapters. Inside the Chapter 1 an overview of machine learning, wireless sensor network, transfer learning followed by problem statement and research objective. Chapter 2 consists of related work in this field. Chapter 3 and 4 explains the various steps which make up the proposed mechanism. Chapter 5 presents the results and analysis of the proposed model followed by conclusion and future scope.

1.10 Summary

In this chapter various component of machine learning, transfer learning, deep learning and wireless sensor network are discussed. Various challenges are also identified in this chapter.

In the next chapter various different literature present on this topic are aggregated.

Chapter 2

Literature Survey

2.1 Overview

Various articles and research paper has been studied to find the applicability of time series imputation and time-series forecasting in the wireless sensor network. Also, various techniques and algorithms available to deal with this problem are studied. It gave a clear picture of what the problem is and what are the measures that are taken and can be taken to solve the problem. It gave a clear insight into the problem statement and helped in shaping the organisation and direction of the thesis.

2.2 Related Literature

A lot of literature is present on missing time series data imputation. Only closely related is mentioned here. T et al. [21], the author proposed a fusion long short-term memory-convolutional neural network model to predict the stock prices by combining features from different representations of data. This model outperforms here in the comparison of single models because here, LSTM+CNN both used to extract temporal features and image features.[22] Verma et al. [23], the author worked on RNN based model and used Generic Sliding Window (GSW) to produce more training samples, i.e., more time-series data so that model can be trained easily.

The traditional methods, like ARIMA, SARIMA, etc., remove the non-stationary parts in the time series data and try to fit a parameterized stationary model. Further combination of ARIMA and Kalman Filter [24, 25] is used by the space model and provides better results. Multivariate Imputation by Chained Equations [26] initializes the values which are missing in time series data randomly, and then the elimination of each missing variable is done using the chain equations. Yi et al. [27] proposed a multi-view learning-based method to impute the missing time series values in air quality data and attain a mean relative error of about 17% for PM2.5. Wang et al. [28] applied a collaborative filtering method to the recommendation system in order to fill the missing time series values.

Yuan et al. [29], the author applied the LSTM model on air pollution time series data and proves the LSTM model outperforms the statistical method like mean, moving average method, etc., and achieves better accuracy in predicting PM2.5 values of time series data. Che et al.[30], the author proposed a method called GRU-D, which is applied to health care data, on which it gives a remarkable performance. This method is based upon Gated Recurrent Unit for missing time series data imputation. Sutskever et al. [31], the author invented a seq2seq learning approach that used multi-layer LSTM to map the input sequence to a fixed dimensional vector and a deep LSTM to decode the target sentences from that vector. The authors used the WMT'14 dataset for English to French translation tasks. The Authors achieved a BLEU score of 34.8 on the entire test set. The authors also analyzed that reversing the source sentences can lead to an increase in the performance of LSTM by making optimization easier[25].

Leke et al. [32], the author proposed an auto-encoder neural network to predict missing data in the forest fire data set from the UCI data repository by considering that a wireless sensor network usually monitors multiple parameters and used a common approach is to reconstruct the missing parameter based on other available parameters in the data sets. Cho et al. [33], the author invented the Recurrent Neural Network-based Encoder-Decoder novel model. This model consisted of two Recurrent Neural Networks. One of the RNN would be used to encode input sentences into a fixed-length vector, and the other would be used to decode the fixed-length vector. The proposed model is jointly trained to maximize the probability of target sequences conditioned on source sentences.

In the above discussion, various literature in the field of missing time series data imputation has been discussed. With the study of these works, it is analyzed that machine learning or deep learning techniques can play an important role in the imputation of missing time series data[34]. It is also analyzed and understood that this work would be done by implementing Encoder-Decoder architecture of sequence two sequence learning [35, 36, 37]

Data loss is a major problem in IoT devices because of various issues like fault in the devices. [21]
Various methods for predicting missing data are developed to overcome this problem. Machine learning is a branch of computer science gaining much popularity these days and deep learning is a subset of machine learning which can be used for finding un evident pattern in the data which can be used to find the relation in the time series data and then that relationship can be used for predicting the missing data collected [21].

Predicting the missing data in time series data collected by the sensor would help in data analysis forecast prediction, corrective measures can be taken.[24] Machine learning and deep learning algorithms are proved to be best in predicting the missing data in the time-series data. Fig 1 depicts the machine learning techniques used for data imputation.

Artificial neural network (ANN) - ANN is a machine learning technique where the model is trained by feeding samples to the model with the label. Training samples are generated using various sliding window methods available. It gives a fair result. But the resource overhead is more as we go deep for increasing the accuracy of the model.[25]

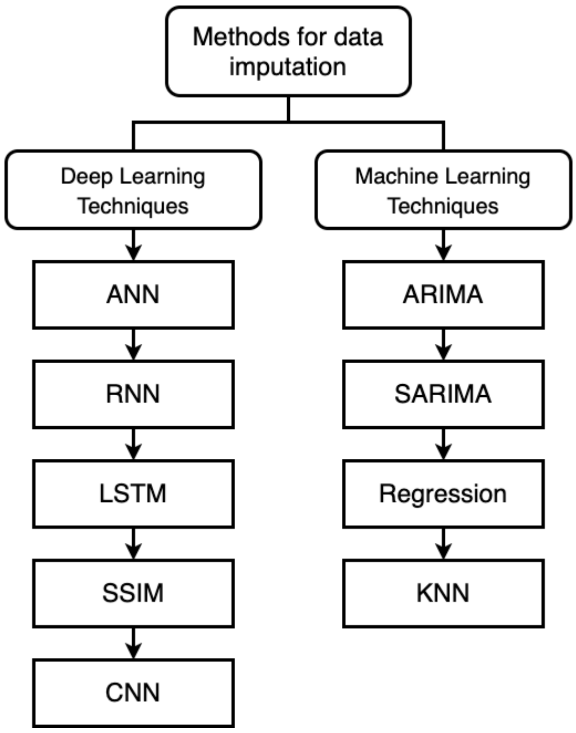


Figure 2.1: Available methods of imputation

Recurrent Neural Network (RNN) - RNN is a machine learning technique in which temporal sequence is considered while making forecasting. Because of this reason it performs better than ANN. But still it is not capable to model upon large sequences of temporal data.[26]

Long Short Term Memory (LSTM) - The problem of remembering large sequence data and inferring upon it is overcome by LSTM network.[27]

Sequence to Sequence Imputation Model (SSIM) - In SSIM [37] instead of predicting just the next data point in the sequence, it is capable of forecasting several data points next to the current end. Attention mechanism and (Variable length sliding window) VLSW are used for further improving the model accuracy/efficiency.[28]

Convolutional Neural Network (CNN) - The convolutional neural network is a technique in machine learning which is very famous in the image processing and video processing field. As it tries to learn the spatial arrangement of the pattern it proves to be useful in time series forecasting. But as it very resources heavy it may not be suitable for IoT.[29]

2.3 Summary

Many machine learning and deep learning methods are available in the literature and can be used for making time series imputation and forecasting. Deep learning requires a large dataset to solve the problem effectively. WSN doesn't provide much data in the real-time scenario. There is also the limitation of applying deep learning of the scratch on end nodes due to low computing power and low memory available in the sensing nodes.

Chapter 3

Prelimnary Techniques

3.1 Overview

This chapter provides various tools and techniques that are used in this thesis like VLSW protocol used, SSIM model used and other language and framework details and platform used for realisation of the techniques proposed by this thesis.

3.2 VLSW Protocol

Variable-length sliding window protocol is a technique to generate time-series data for training machine learning or deep learning model. In this technique, a window of variable length is moved over the data, with zero-padding the remaining unaddressed bits of the window. For time series imputation of five consecutive timestamps, the variable-length window before and after that timestamp is collected and stacked into an array of data, and the actual values of that five timestamp data are used as the output of the algorithm. Figure 3.1 describes the working of variable length sliding window protocol.

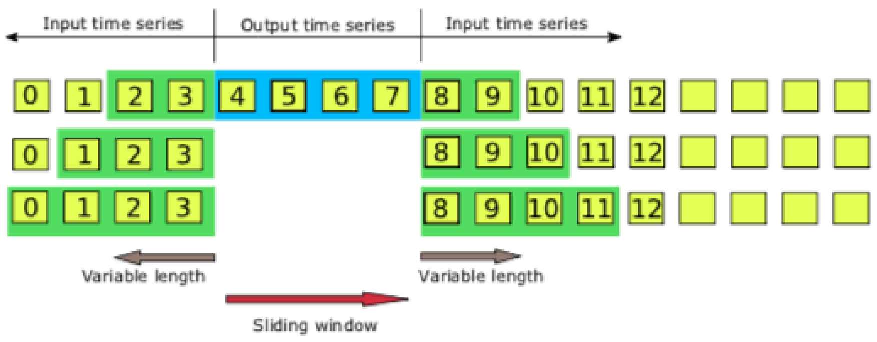


Figure 3.1: VLSW Protocol

3.3 Sequence to Sequence Imputation Model

SSIM model module consist of 4 processes which are:

- Encoder

- Attention layer
- Decoder
- Dense layer

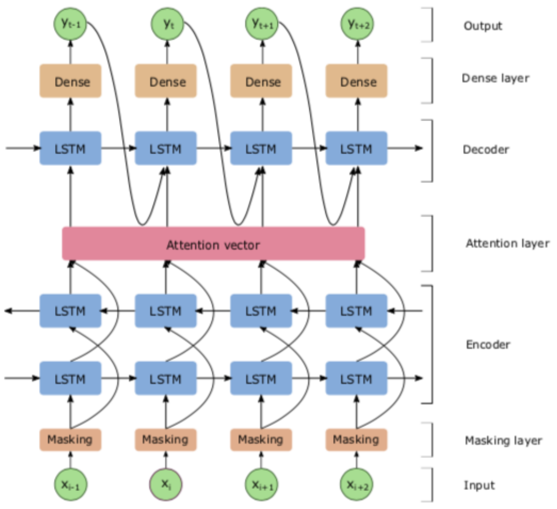


Figure 3.2: SSIM Model Architecture

BiLSTM is used as the encoder which processes the data sequence in both directions i.e in forward direction and in backward direction. It checks for the pattern and learns its attention mechanism that is used to focus on a specific part of the sequence which is necessary to generate the output sequence. Figure. 3.2 describes the SSIM model architecture.

3.4 Implementation Tools

Following is the disscussion about the tools that are used for implementing the models and training and testing the approach.

3.4.1 Python

Python is an interpreted, high-level, general-purpose programming language. It is widely used for programming machine, and deep learning algorithms, many famous machine learning, and deep learning libraries are built into python. Data scientists and data analysts widely use them. Prototyping is very easy in this language, and it is easy to learn and master. It is best suited for training deep learning models. The very famous deep learning framework Tensorflow is built using this language.

3.4.2 Scikit Learn

Scikit learn is a free opensource machine learning library written in python language. There may machine learning algorithms are implemented into this library like classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. There are many helper functions and machine learning useful methods are implemented that can be used out of the box.

3.4.3 Tensorflow

Tensorflow is a software application popular for implementing machine learning algorithms, particularly neural networks. It was developed by Google it was released as an open-source platform in 2015. It is called Tensorflow because it takes input as a multidimensional array. Multidimensional arrays are also known as tensors you can construct a sort of flow chart of operation you want to perform on that input so input goes in at one end, and then it flows through this system of operation and comes out the other end as output, and that is why it is called TensorFlow. It's extremely versatile; it can be run on many diff platforms (computers, pc, laptops, raspberry pi, etc.). It can be trained on multiple machines. It can also run on GPU as well as CPU. There is tensorboard to help monitor the operation of machine learning algorithms visually.

3.4.4 Pandas

Pandas is a python library that gives a great set of tools for data analysis. With pandas you can load prepare manipulate model and analyze data, can join data, reshape data, merge data, Combine datasets. It is very useful library for data manipulation. In this thesis the pandas is used to wrangel the dataset remove missing values and padding and careing the timestamps so that can be made in sequence manner so that deep learning algorithms can be applied to it.

3.4.5 Matplotlib

Matplotlib is a tool for data visualization in python strongly coherred with numpy and pandas. It helped in the thesis for generating the result graphs and other graphs needed for the analysis.

3.4.6 Google Colab

Google Colab is a Jupyter notebook in the google cloud pre installed with many machine learning and deep learning libraries generally needed for the data analysis data visualization, machine learning and deep learning. Beside that other not installed libraries can also be installed and used. It also provide the GPU (Graphics Processing Unit) support for training the models very fast as compared to CPU (Centeral Processing Unit) which is also available and can be used. Along with CPU and GPUs it comes with TPU (Tensor Processing Unit) which are special kind of GPU specialy designed to work with the very famous tensorflow framework. Currently it is only available in the google cloud hosted platforms like colab or for custom google compute cloud one can easily build and run machine learning algorithms in the cloud.

3.5 Summary

In this chapter various tools and techniques are described that uare used for the development of the proposed work. The next chapter describes the solution to the problem using the tools and techniques.

Chapter 4

Critical Analysis of Imputation Techniques

4.1 Overview

In this chapter various timeseries imputation and forecasting techinques are compared and results are compiled along with their implementation details. The results of the SSIM model is also described here which is an encoder decoder model the type of model normally used in machine translation techniques.

Model	MSE Score
Regression	0.342
LSTM	0.094
CNN	0.0763
SSIM	0.0944

Table 4.1: Comparision of time series techniques

4.2 Regression

Regression is a the core of most of the machine learning algorithm. For time series imputation regression technique can be used although it may not give the best results but it gives the fairly satisfiable results based on time taken by the algorithm to make a prediction model.

Figure 4.1 shows an actual vs predicted graph of the a regression model trained on the PM2.5 values of Beijing city. Table 4.1 contains the MSE score achieved by the model.

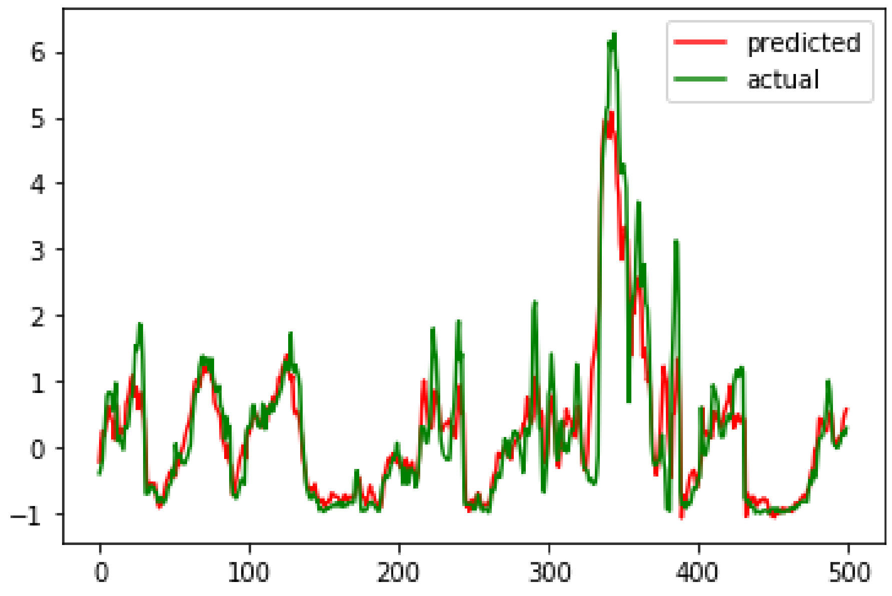


Figure 4.1: Regression PM2.5 Actual vs Predicted graph

4.3 LSTM

Long Short Term Memory(LSTM) Network is a special kind of recurrent neural network where it maintain a cell state variable which helps it to deal with the vanishing gradient problem in RNNs. It proves to be making it very feasible for time series imputation.

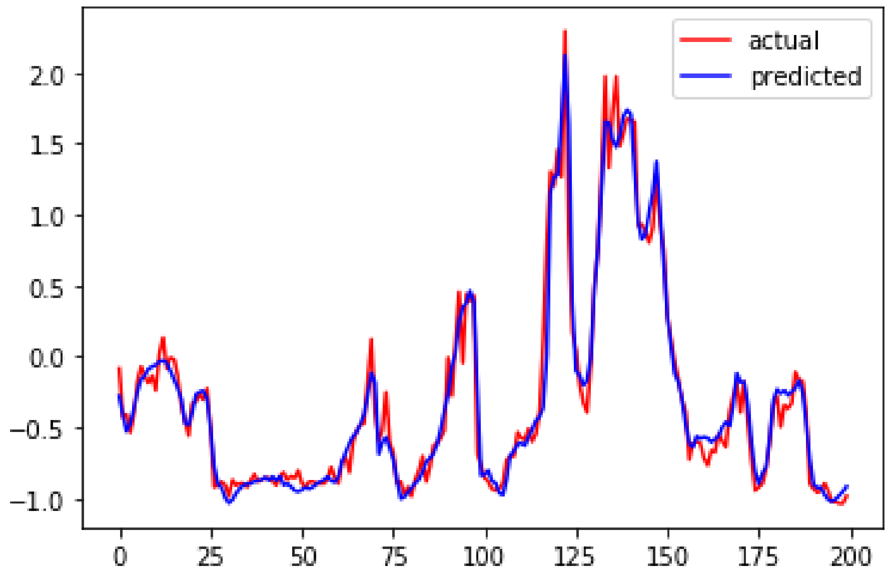


Figure 4.2: LSTM PM2.5 Actual vs Predicted graph

An LSTM model is trained on PM2.5 its MSE score is listed in table 4.1. Figure 4.2 shows the actual vs predicted graph of the LSTM model trained on the PM2.5 values of Beijing city.

4.4 CNN

Convolutional Neural Network is very famous neural network for the field of image recognition and computer vision. It has feature of pattern detection which makes it very feasible in time series forecasting as in time series forecasting and imputation certain pattern in the data need to be exploited for making the prediction. It is generaly used with non covolutional layers in the deep learning but the main core of it is the convolutional layer which are responsible for exploiting the pattern in the data.

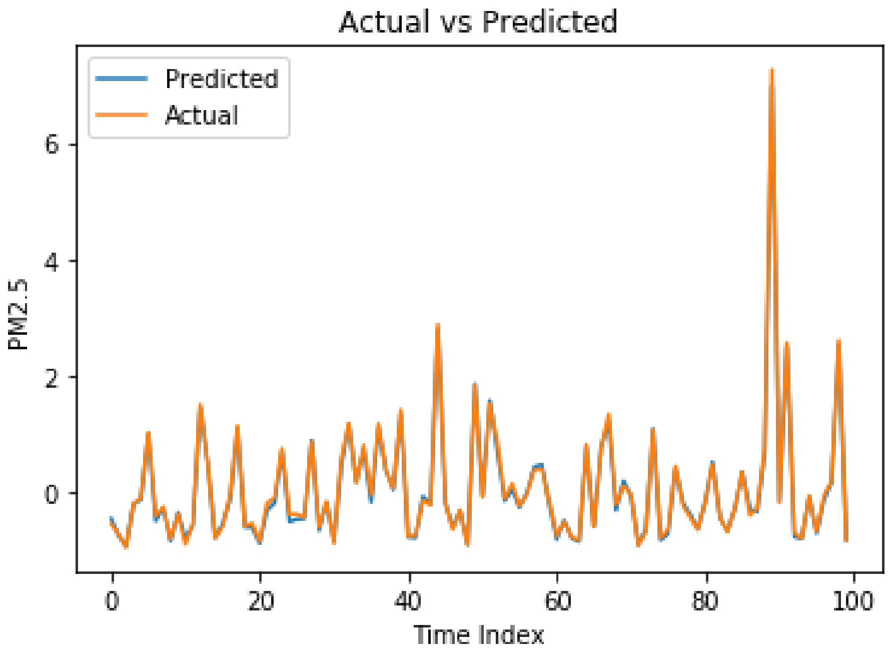


Figure 4.3: CNN PM2.5 Actual vs Predicted graph

Figure 4.3 show the actual vs predicted graph of the CNN model trained on PM2.5 values of beijing city. Table 4.1 show list the MSE score of the model.

4.5 SSIM

Sequence to Sequence Imputation Model (SSIM) is an encoder decoder model using the LSTM model as encoder attention vector is used which the encoder convert the input sequence into and another LSTM model is used as the decoder model after that a dense layer is also used for softmax values of the sequence to be generated. The working of the encoder decoder model is such that a variable length input is given to the network it converts the given input sequence to a fixed length vector known as attention vector, and the decoder convert the attention vector into the desired length sequence vector for the output.

Figure 4.3 show the actual vs predicted graph of the SSIM model trained on PM2.5 values of beijing city. Table 4.1 show list the MSE score of the model.

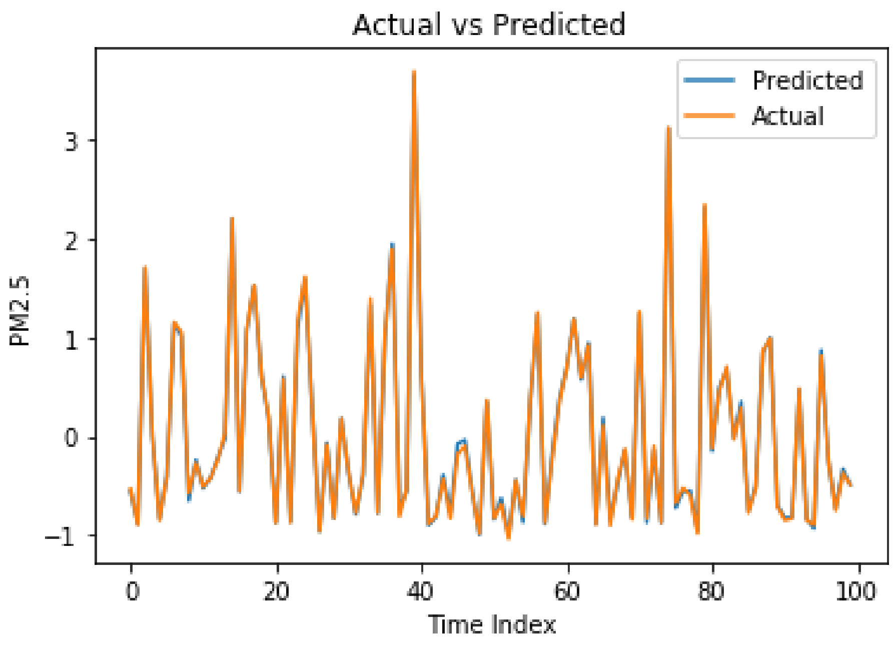


Figure 4.4: SSIM PM2.5 Actual vs Predicted graph

4.6 Summary

In this cahpter various available machine learning and deep learning models are implemented and their mse score and actual vs predicted graphs are compared. All these model works well when they are applied on some problems where there are properly large amount of data available. In the next chapter a new approach of dealing with problem when not much data is available with hand the how deep learning models can be applied. The Transfer Learning For Time Series Imputation is applied on the two deep learning models which works well with time series problem i.e., CNN and SSIM.

Chapter 5

Proposed Work

5.1 Overview

In this chapter the proposed work of **Transfer Learning For Time Series Imputation** is described with the system models, analytical models, flow diagrams, model architecture, trained model symmary along with the discussion about every components in detail.

5.2 System Model

System model is such that once we achive the desired level of model accuracy and loss value that model are deployed in the real environment so that it get used and also trained in online environment which is that it self correct and make the system self sufficient .Our system model consists of the following components.

5.2.1 Sensing Layer

Sensors and actuators in this layer collect data from the environment in this layer. Also, it provides the interface to the real world to interact with the environment and provide impact in the physical world. It helps in both collecting data from the environment and to provide action that results in the making system work properly with in the sync.

5.2.2 Network Layer

Data collected by the sensing layer is transported to the computing devices and processing and control system using Network Layer. This layer is used for transporting the processed and to be processed data to and fro from on end to the other end. Many protocols and networks like MQTT COAP can be used easily.

5.2.3 Data Processing Layer

The data processing layer is the layer of IoT Architecture where all the computing of the system occurs. Our paper tries to solve the problem all of which lies here. It is the part of processing unit as shown in Figure. 5.1

Nature of IoT data - IoT systems generates high-volume, streaming, location, and time-specific this type of data is called time-series data. We would be deploying our fault-tolerance system in this layer only.

Fault tolerance layer

In this layer, we predict the missing values that our sensors don't able to collect. This may be due to many various reasons like sensor failure, network disturbance, battery discharge etc. this is the part where the action according to the predictions is taken. It may be some alarm initiation some improvement logic or simply update the record in the database. This is also the part of the processing unit as shown in Figure. 5.1

5.2.4 Application layer

The application layer is the layer where the user interacts with the system or the machine interface takes action based on the output of the IoT system.

Application layer can be made in any framework that working and nature of problem. In the current simulation scenario as the pollution data is taken for the reference it is more feasble to make a web interface where an user can see the real time monioring of the various weather and pollution parameters and if required can take action based on the results shown in the dashboard. For making the dashboard any front end framework can be used like Angular or React where it is very easy to build user interfaces with single page application. For the backend python framework like django or flask can be used as it will help in predicting and implementing the machine learning models on the cloud environment.
Server side application programmable interface and front end dashboard can be connected with the any real time network interface. The one which would work feasible is socket. Other protocols like MQTT or COAP can also be used.

In the fault tolerance layer, we have generated a machine learning model in that we can generate inferences from and also train the model in real-time.

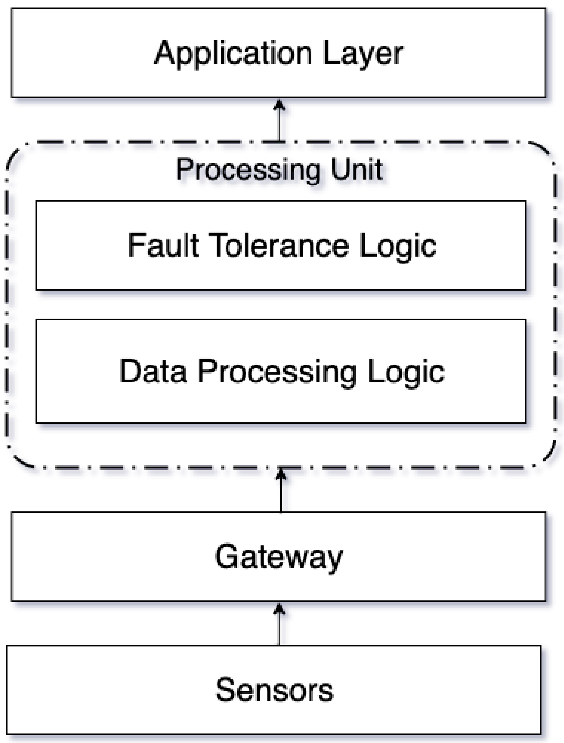


Figure 5.1: System Model

5.3 System Flow

In this section the flow of the data and precessing and imputation processes are described. The figure 5.2 dipicts the system flow diagram in the form of data flow diagram.

5.3.1 Data Collection

For this demonstration and testing the approach proposed PM2.5 Data of two chinese are taken. This data is taken from UCI machine learning repository. It is very famous source for dataset for academic purposes. It contains many famous machine learning dataset widely used for academic and research perposes. Some of them are IRIS Flower Dataset, Titanic Survival dataset, etc. The two cities PM2.5 values data are taken from a dataset containing PM2.5 values of five chinese cities named "PM2.5 Data of Five Chinese Cities Data Set". Description of the dataset used is given below.

Dataset abstract This hourly data set contains the PM2.5 data in Beijing, Shanghai, Guangzhou, Chengdu and Shenyang. Meanwhile, meteorological data for each city are also included.

The table 5.1 describe the nature of the dataset.

The table 5.2 describe attributes and features of the attributes of the dataset.

The two cities selected out of 5 are Beijing and Shanghai. Beijing data is used as the data set with large values and the Shanghai data is considered as the realtime data for which deep learning model need to be developed. This data is traind both without using transfer learning and with the transfer learning applied.

Data Set Characteristics	Multivariate, Time-Series
Number of Instances	52854
Area	Physical
Attribute Characteristics	Integer, Real
Number of Attributes	86
Date Donated	2017-07-18
Associated Tasks	Regression
Missing Values?	Yes
Number of Web Hits	81692

Table 5.1: Dataset Characteristics

No	row number
year	year of data in this row
month	month of data in this row
day	day of data in this row
hour	hour of data in this row
season	season of data in this row
PM	PM2.5 concentration
DEWP	Dew Point (Celsius Degree)
TEMP	Temperature (Celsius Degree)
HUMI	Humidity (%)
PRES	Pressure (hPa)
cbwd	Combined wind direction
Iws	Cumulated wind speed (m/s)
precipitation	hourly precipitation (mm)
Iprec	Cumulated precipitation (mm)

Table 5.2: Dataset Attributes

5.3.2 Data Cleaning

Data cleaning is a process of sanitising the data available with us so that its usabilituy may increase. There is a very famous phrase in data science garbage in garbage out. So data cleaning is very important step in any data related work.

Dataset contains many missing values which currently cannot make use of as it is so the missing values columns are dropped from the dataset managing the consistency in the dataset and the cleaned datase is used for the further processing.

Following method of the pandas is used for the cleaning the data.

```
pm25_beijing = df1.iloc[:, 5].dropna().values[:].reshape(-1, 1)

pm25_shanghai = df2.iloc[:, 9].dropna().values[:600].reshape(-1, 1)
```

5.3.3 Data Preprocessing

Loading the data

The dataset is stored on a github repository so that it can be easily accessed through a link anywhere the code is running. The only need would be internet access while runing the code.

Matplotlib library is used for loading the data in the python environment.

```
import matplotlib.pyplot as plt
URL_beijing = "LINK_OF_GITHUB_CSV_FILE"
df_beijing = pd.read_csv(URL_beijing)
```

Removing unnecessary data

Actually the dataset consisted of various featue about the weather but for simplicity all other attributes except PM2.5 were removed.

Scaling the data

The PM2.5 values are not in the range of -1 to 1 so these values are scaled down to -1 to 1 range so that the can be fed to the machine learning algorithm later.

Scikit learn StandardScaler function is used for scaling the data down.

```
from sklearn.preprocessing import StandardScaler
scaler_beijing = StandardScaler()
scaler_beijing.fit(pm25_beijing)
pm25_beijing = scaler_beijing.transform(pm25_beijing)
```

Making dataset as input output pair for training

There are several ways of doing this the one best technique of doing this. One method is discussed earlier. For this sliding window protocol is used for making this data as supervised learning dataset so that it can be trained.

```
N = 20
O = 5
P = 20

X_beijing = []
y_beijing = []

for i in range(len(pm25_beijing) - (N + O + P)):
    temp = []
    temp1 = pm25_beijing[i : i+N]
    temp2 = [0 for _ in range(O)]
    temp3 = pm25_beijing[i+N+O : i+N+O+P]
    X_beijing.append(np.append(np.append(temp1, temp2), temp3))
    y_beijing.append(pm25_beijing[i+N : i+N+O].reshape(O))
```

Here N is the length of the left slide length, O is the output length, i.e., for how many timestamp we want to predict the output, and P is the length of the right slide length.

The X data is prepared as to consist of left slide length from the sequence of data available for the output lenth 0 is filled in the sequence and at last the right slide length data is attached.

The y of the dataset is prepared by taking the actual values of the timestamp values of the O length datapoints.

5.3.4 Finding Similar Dataset

As the transfer learning to work for the WSN or IOT data it is necessary to find some dataset similar to the problem at hand that need to be solved. This type of cases can be very simmilar to real world such that a new thermal plant is need to be installed in any area and there is need for some fault toulerence logic for that but not have enough data to train the deep learning model for that. For that similar dataset can be picked from any similar thermal plant sensors.

For training for the shanghai PM2.5 value prediction Beijing data is used as training point for applying the transfer learning.

5.3.5 Training deep neural network

Spliting the dataset into training and test sets

Before feeding the data to the model the first step is to split the data into training and the test set so that model performance can be measured.

Scikit-Learn library model selection model train test split function is used for spliting the dataset into the training and testing values.

Code for this splitting is provided below

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_beijing, y_beijing)
```

Training

The first step of applying transfer learning is to train the model with huge amount of data available from any source which give the huge amount of data. The dataset for transfer learning training is picked in the previous section, that is Beijing city data. The deep learning model is first trained on the Beijing PM2.5 values dataset that contains data from previous several years.

5.3.6 Removing last layer of DNN

For applying the transfer learning to with the existing model the first step is to remove the last layer of the trained model so that new model with its features and attributes can be coded. the main motive of removing the last layers of the DNN to apply transfer learning is to incorporate the feature of the dataset one we are trying to train now.

5.3.7 Adding more layer to the DNN

The next step of applying the trasfer learning is to add more layer to the model from the previous step so that the new feature can be incorporated and trained again for the new dataset.

*Note: The last two steps can be ignored if nature and nature of data is ditto same.

5.3.8 Retraining the model

When new layer is added to the model now it can be trained on the data available now for new sensor on new location. Training can be in any manner, i.e., online training or batch training.

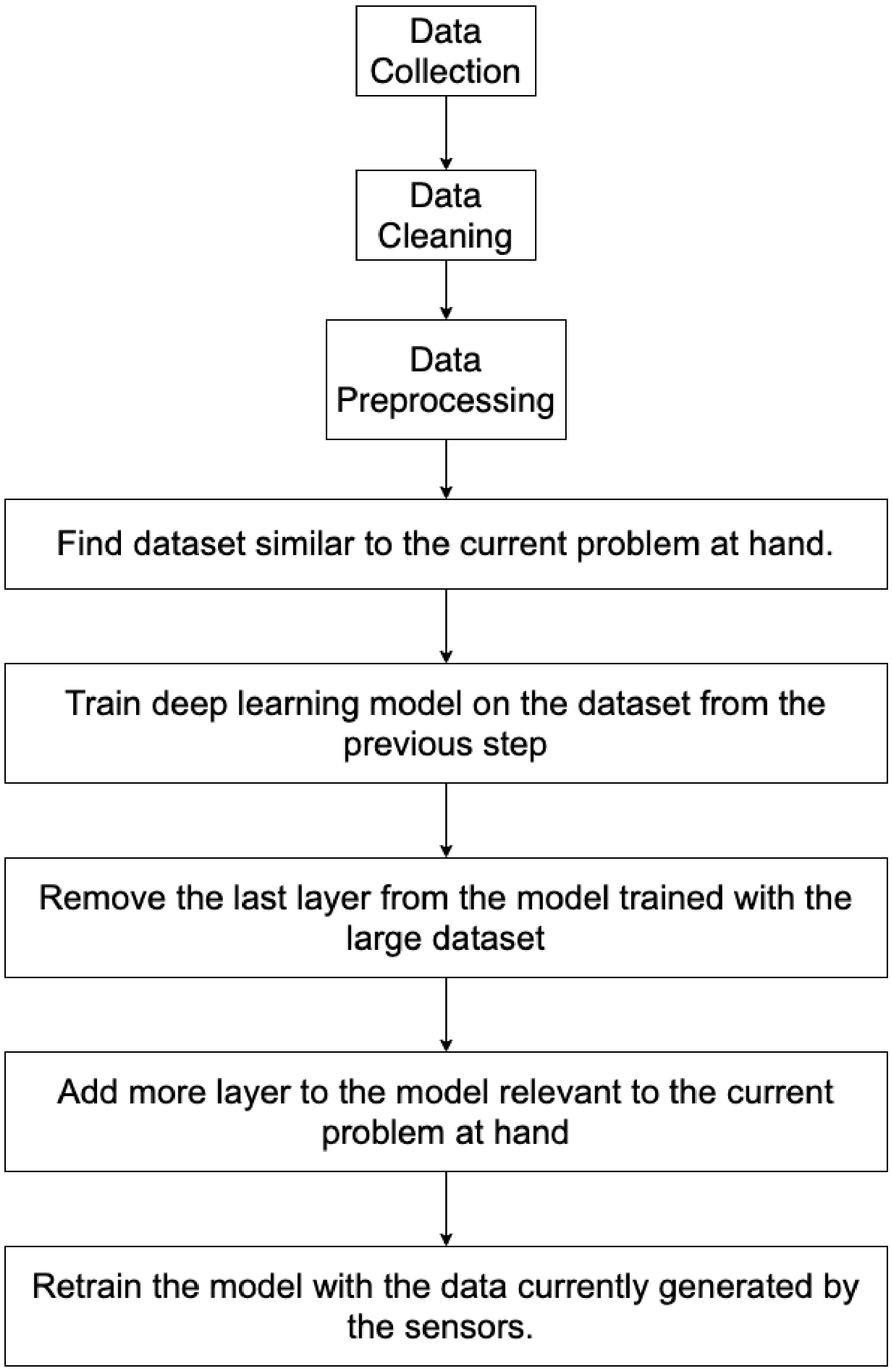

Data Collection
Data Cleaning
Data Preprocessing
Find dataset similar to the current problem at hand.
Train deep learning model on the dataset from the previous step
Remove the last layer from the model trained with the large dataset
Add more layer to the model relevant to the current problem at hand
Retrain the model with the data currently generated by the sensors.

Figure 5.2: System Flow Diagram of Transfer Learning For Time Series Imputation

5.4 Model Architecture

The two models is selected for applying transfer learnig for this proposed work. Figure 5.4 describes the model summary of the two models.

5.4.1 SSIM (Sequence to Sequence Imputation Model)

LSTM is a special kind of RNN(Recurrent Neural Network) which exploit the pattern in the data which makes. It has a memory unit which remember the previous sequence in the memory. SSIM is an encoder decoder model which is generally used in natural language processing for machine translation and it can be used here for time series imputation. Figure 5.4a Describe this architecture of the SSIM model.

The SSIM model architecture contains the followin layers.

Bidirectional LSTM layer

It is a bidirectional LSTM layer to exploit the data in both the direction in forward along with backwards.

There are 128 neurons used in the Bidirectional LSTM layer.

Attenttion Layer

This is a layer used in Encoder Decoder model to focus on certain specific part of the sequence which are more relevant to the application and provides better results. An analogy of this is found in the image processing where the task is to identify the object in the image. Since the object in the image can be found in certain part of the photograph it need to focus on that part of the photo to identify the object. Figure. 5.3 describes this proces of attention mechanism in terms of computer vision.

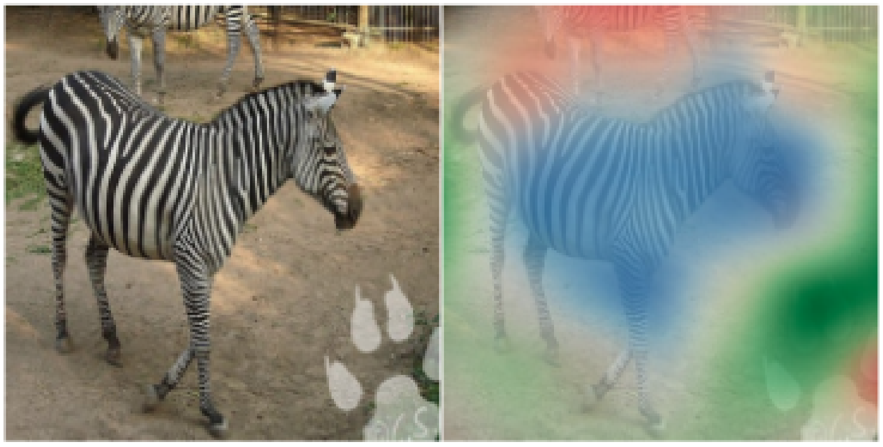


Figure 5.3: Attention Mechanism in terms of computer vision

LSTM Layer

This is a simple LSTM layer in the SSIM model comparising the decoder of the encoder decoder system. This layer is used for decoding the sequence generated by the input.

Dense Layer

This layer is used for maping the output sequence to the desired number of output timestamp values needed by the model.

5.4.2 CNN (Convolutional Neural Network)

Convolution neural network is basically the applied in the field of image recognition and computer vision to find pattern in the image and label according to the labels given to it. Figure 5.4a describes the model summary of the CNN model.

Conv1D Layer

This is the first layer of the CNN model containing for learning the pattern in the data. It contains with 64 filters, kernal size is 3 and relu activation function is used.

MaxPooling Layer

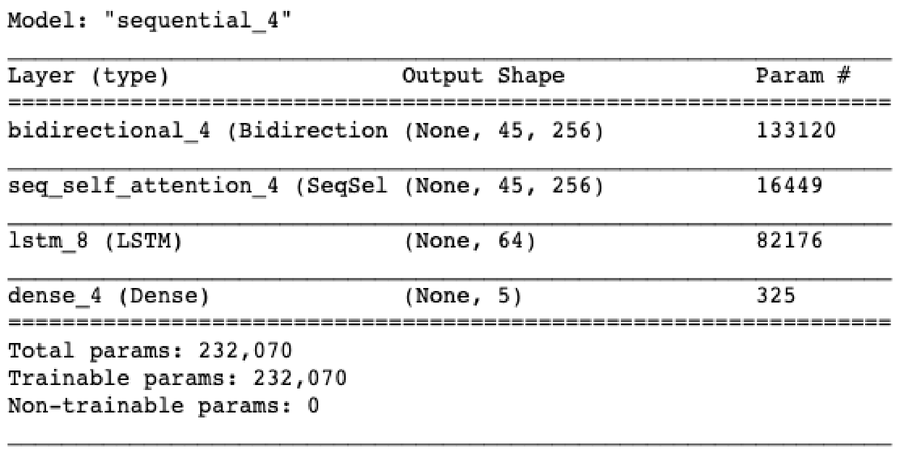
Maxpooling layer is used to down sample input to eleminate the overfitting the training data.

Flatten Layer

The flatten layer is used to change the shape of the input to the 1D sequence of data.

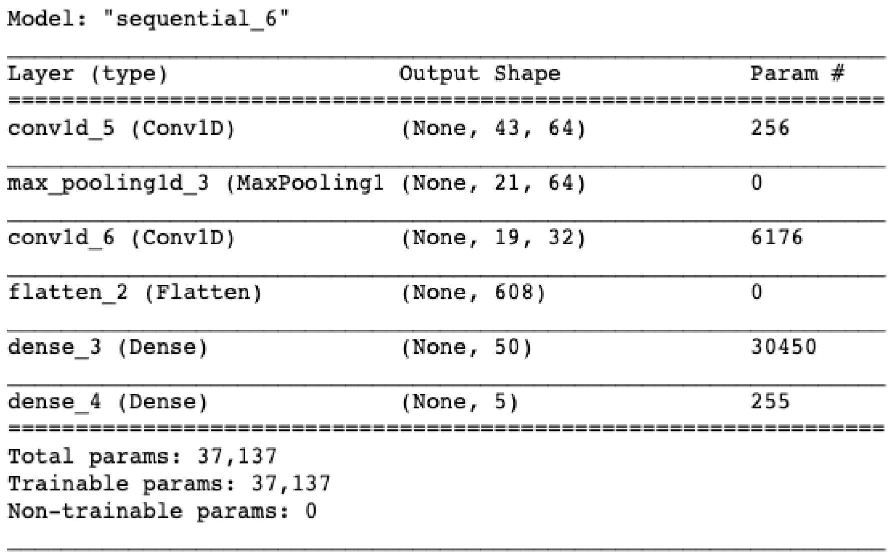
Dense Layer X 2

Two Dense Layer is used one with 50 neurons and another with the 5 neurons to generate the output sequence of the desired length (5 timestamp)



```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_4 (Bidirection (None, 45, 256)           133120
_________________________________________________________________
seq_self_attention_4 (SeqSel (None, 45, 256)           16449
_________________________________________________________________
lstm_8 (LSTM)                (None, 64)                82176
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 325
=================================================================
Total params: 232,070
Trainable params: 232,070
Non-trainable params: 0
_________________________________________________________________
```

(a) Shanghai SSIM Model Summary



```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_5 (Conv1D)            (None, 43, 64)            256
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 21, 64)            0
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 19, 32)            6176
_________________________________________________________________
flatten_2 (Flatten)          (None, 608)               0
_________________________________________________________________
dense_3 (Dense)              (None, 50)                30450
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 255
=================================================================
Total params: 37,137
Trainable params: 37,137
Non-trainable params: 0
_________________________________________________________________
```

(b) Shanghai CNN Model Summary

Figure 5.4: Model summary of transfer learning

5.5 Summary

In this chapter the working of the whole proposed system is discussed in details. In the next chapter the results generated by the system is analysed and described in details.

Chapter 6

Results and Analysis

In the previous chapter the proposed work is explained in details. In this chapter the results generated by applying transfer learning on the shanghai data with two models, SSIM and CNN is analysed.

	Beijing data	Shanghai data	Shanghai with TL	Improvement
CNN	0.0763	0.031	0.015	52 %
SSIM	0.0944	0.0656	0.0215	67 %

Table 6.1: Comparision of MSE Loss of various model with transfer learning

MSE Loss MSE (Mean Squared Error) is the loss function that is used for the training the models on the sensor data. It is used to measure the efficiency of the model trained. Lesser the value of MSE score more the model is efficient.

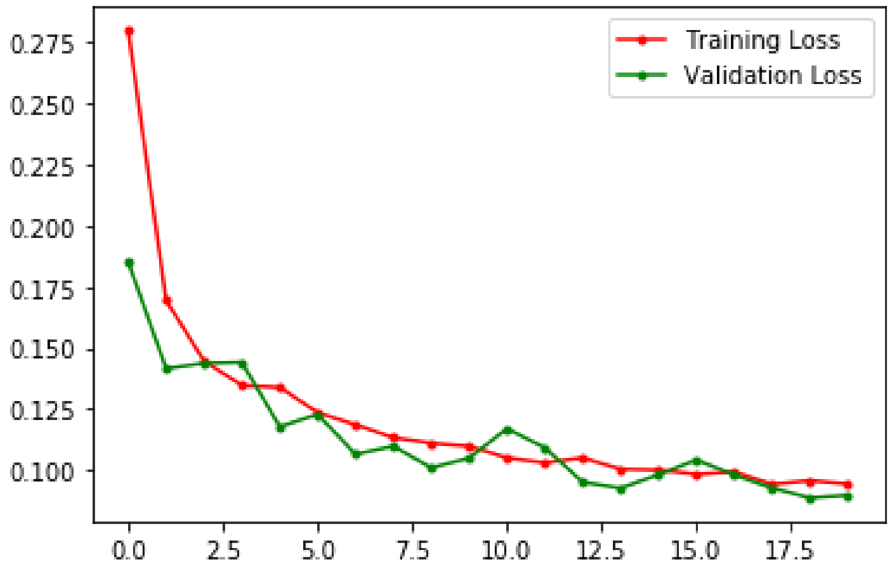


Figure 6.1: Beijing SSIM

Beijing data on CNN model On training CNN model with bejing data mse score achived is 0.763

Beijing data on SSIM model On training SSIM model with beijing data mse score achieved is 0.0944

This results that SSIM model achieved more efficiency in comparision to CNN model for time series frecasting problem.

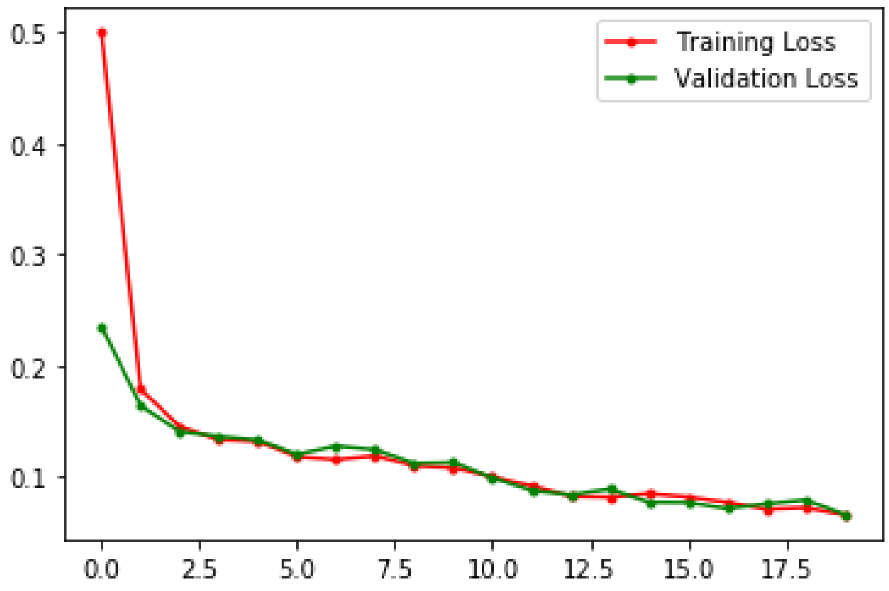


Figure 6.2: Shanghai SSIM

Shanghai data on SSIM without Transfer Learning On training SSIM model with shanghai data without Transfer learning mse score achieved is 0.0656

Shanghai data on SSIM with Transfer Learning On training SSIM model with shanghai data with Transfer learning with using pretrained model of beijing data mse score achieved is 0.0215

This results that SSIM model with transfer learning shows 67% improvement.

Figure 6.1 shows the training vs validation loss graph of Beijing SSIM training

Figure 6.2 show the training vs validation loss graph of Shanghai SSIM training

Figure 6.3 shows the training vs validation loss graph of Shanghai SSIM training with transfer learning with using pretrained bejing ssim model.

Table 6.1 Summarises all the models losses and imporvements.

While using SSIM model there is 67% imporvement with transfer learning.

Also

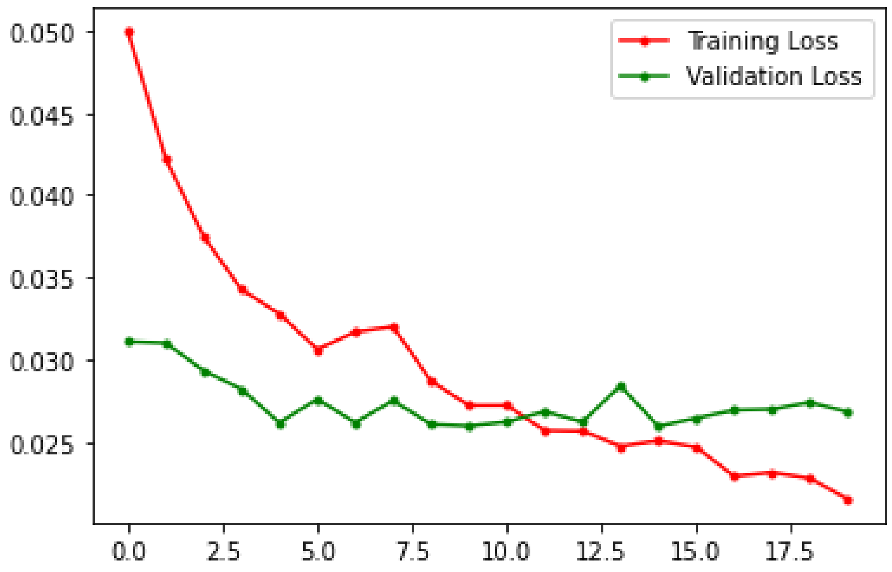


Figure 6.3: Shanghai SSIM Transfer Learning

While using CNN model there is 52% improvement with transfer learning.

Chapter 7

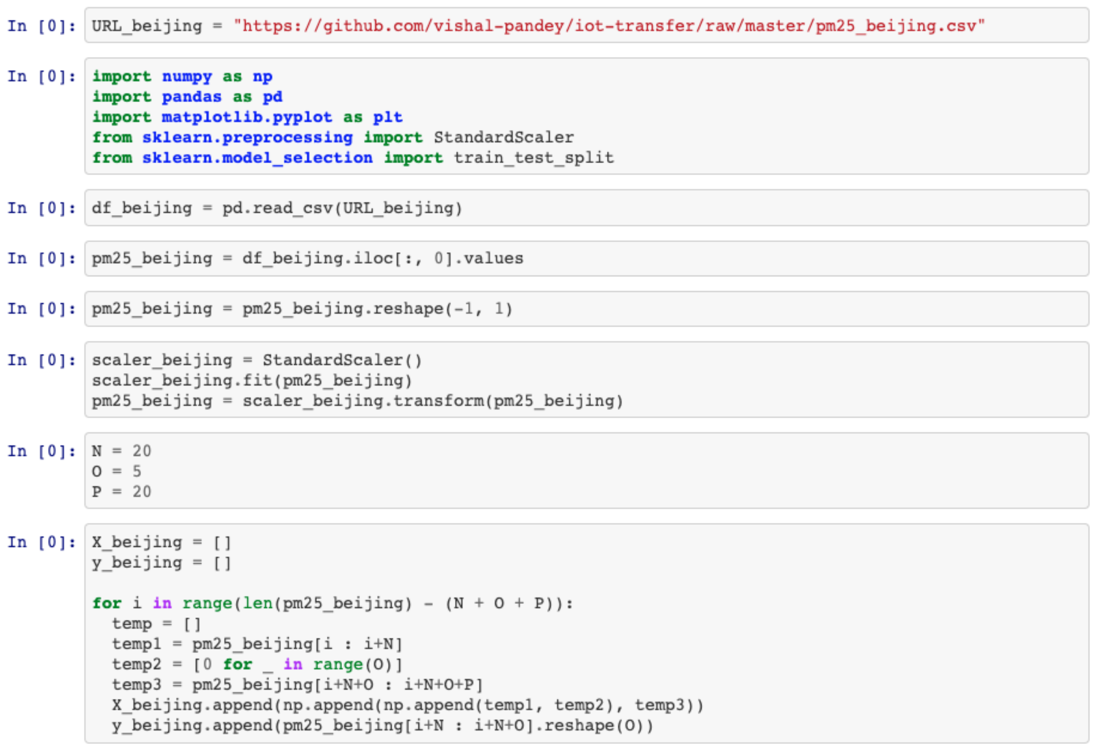
Conclusion

The proposed model of using transfer learning in wireless sensor network data increases the effeciency of the forecasting and imputation. It also reduces the training time which makes it more feasible for online training and for real-time scenario.

The result in the previous chapter shows that it make more feasible to train such model with transfer learning in comparision to the building one from scratch. MSE loss shows that more accuracy can be achieved with less data where WSN is not generating much data for applying the deep learning.

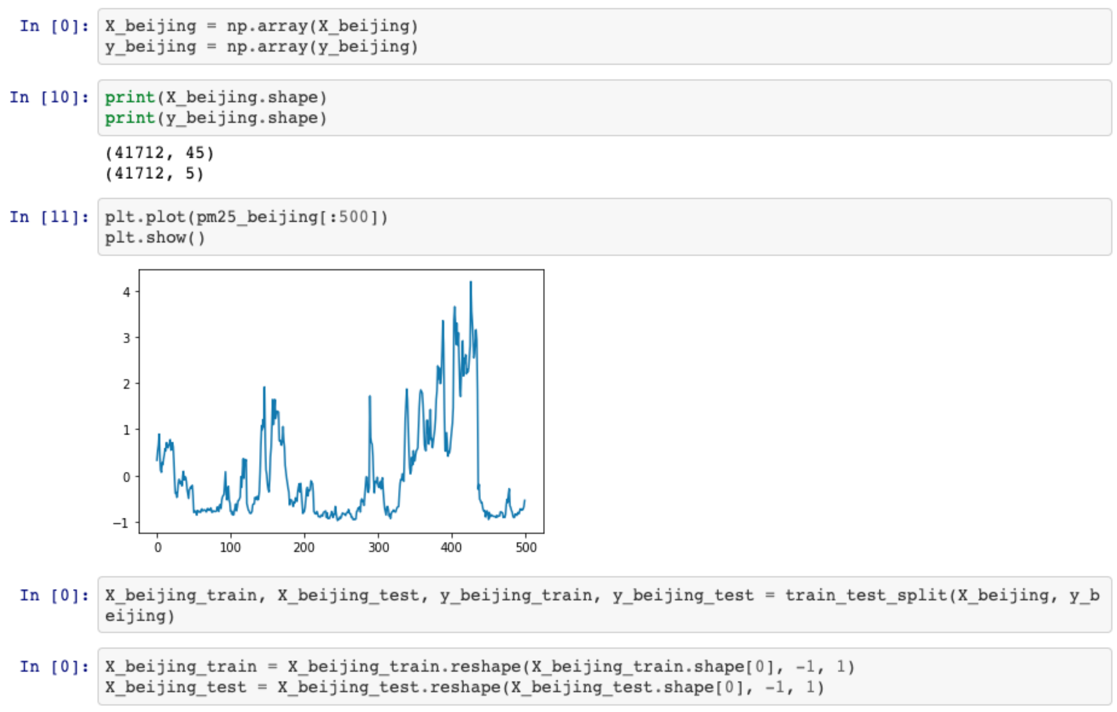
Appendix A

Screenshots of Implementation

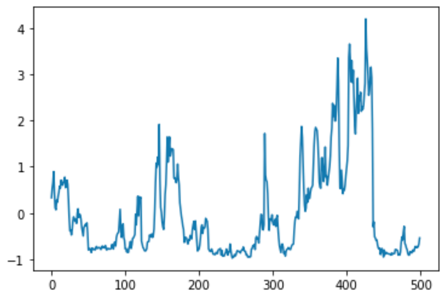


```
In [0]: URL_beijing = "https://github.com/vishal-pandey/iot-transfer/raw/master/pm25_beijing.csv"

In [0]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import StandardScaler
        from sklearn.model_selection import train_test_split

In [0]: df_beijing = pd.read_csv(URL_beijing)

In [0]: pm25_beijing = df_beijing.iloc[:, 0].values

In [0]: pm25_beijing = pm25_beijing.reshape(-1, 1)

In [0]: scaler_beijing = StandardScaler()
        scaler_beijing.fit(pm25_beijing)
        pm25_beijing = scaler_beijing.transform(pm25_beijing)

In [0]: N = 20
        O = 5
        P = 20

In [0]: X_beijing = []
        y_beijing = []

        for i in range(len(pm25_beijing) - (N + O + P)):
          temp = []
          temp1 = pm25_beijing[i : i+N]
          temp2 = [0 for _ in range(O)]
          temp3 = pm25_beijing[i+N+O : i+N+O+P]
          X_beijing.append(np.append(np.append(temp1, temp2), temp3))
          y_beijing.append(pm25_beijing[i+N : i+N+O].reshape(O))
```

Figure A.1: Code Screenshot 1

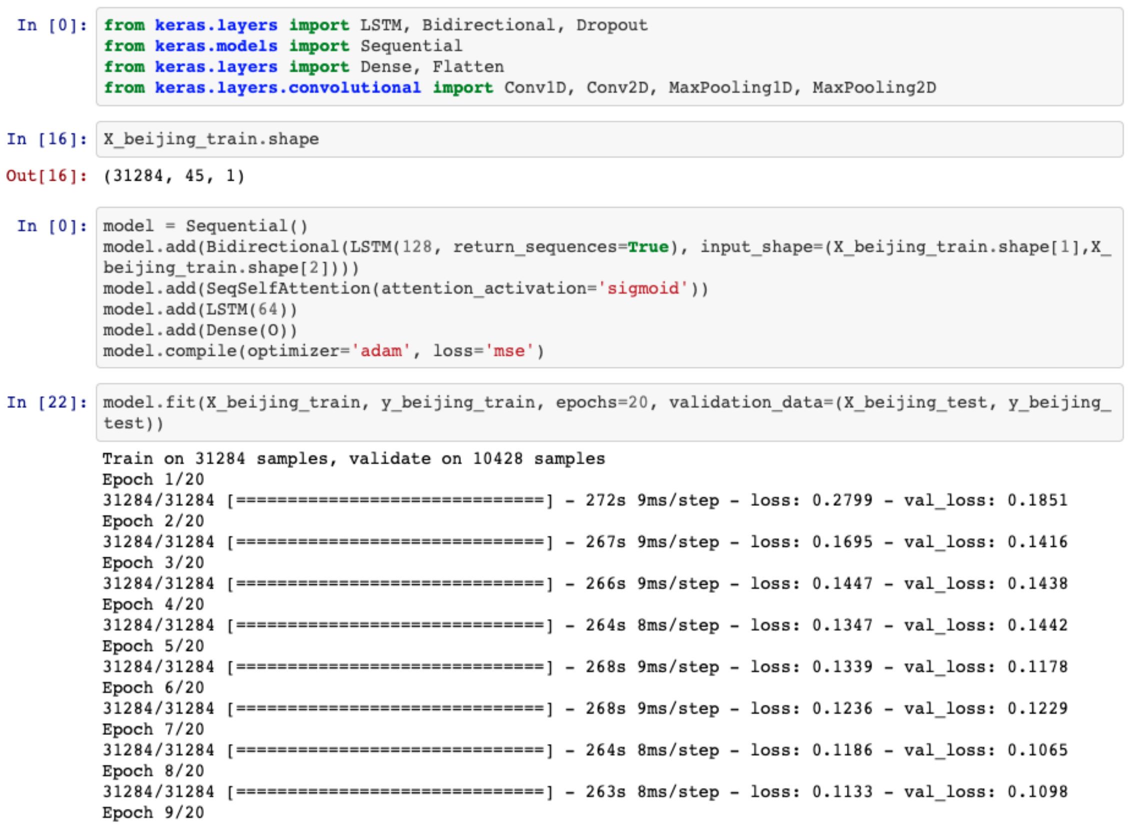


```
In [0]:  X_beijing = np.array(X_beijing)
         y_beijing = np.array(y_beijing)

In [10]: print(X_beijing.shape)
         print(y_beijing.shape)

         (41712, 45)
         (41712, 5)

In [11]: plt.plot(pm25_beijing[:500])
         plt.show()
```



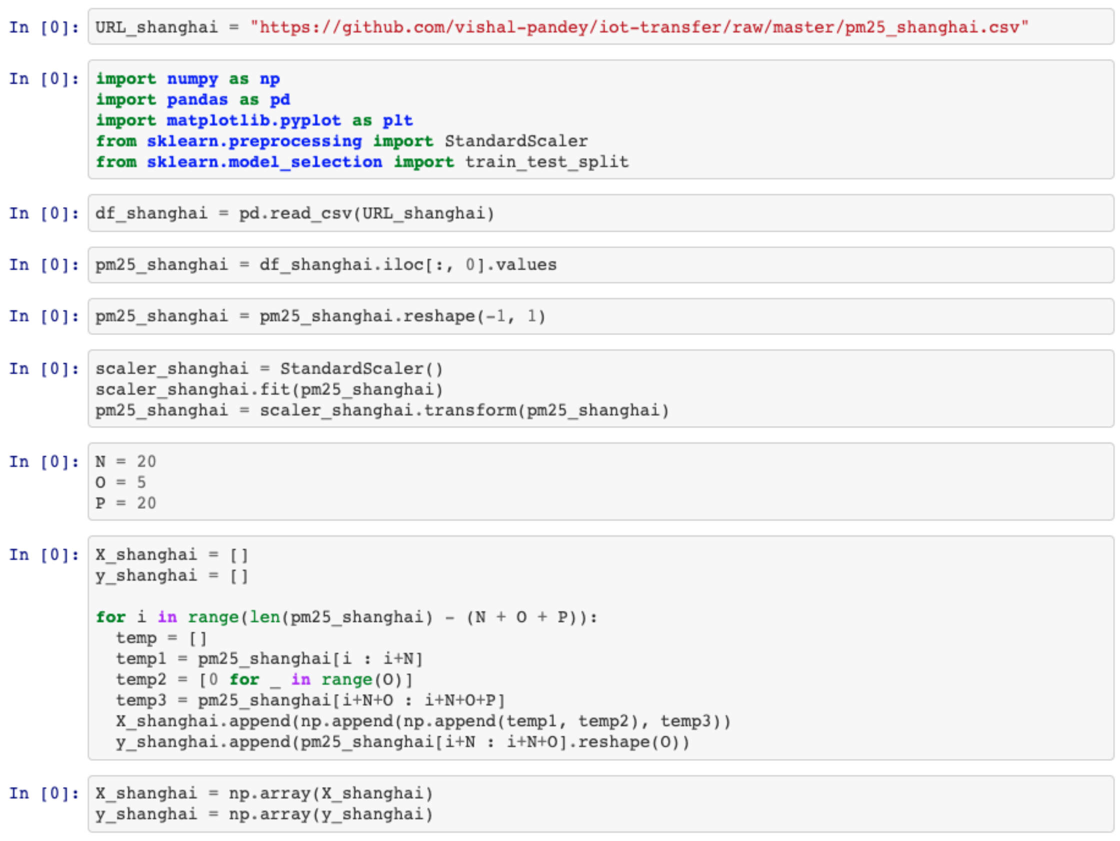
```
In [0]:  X_beijing_train, X_beijing_test, y_beijing_train, y_beijing_test = train_test_split(X_beijing, y_b
         eijing)

In [0]:  X_beijing_train = X_beijing_train.reshape(X_beijing_train.shape[0], -1, 1)
         X_beijing_test = X_beijing_test.reshape(X_beijing_test.shape[0], -1, 1)
```

Figure A.2: Code Screenshot 2



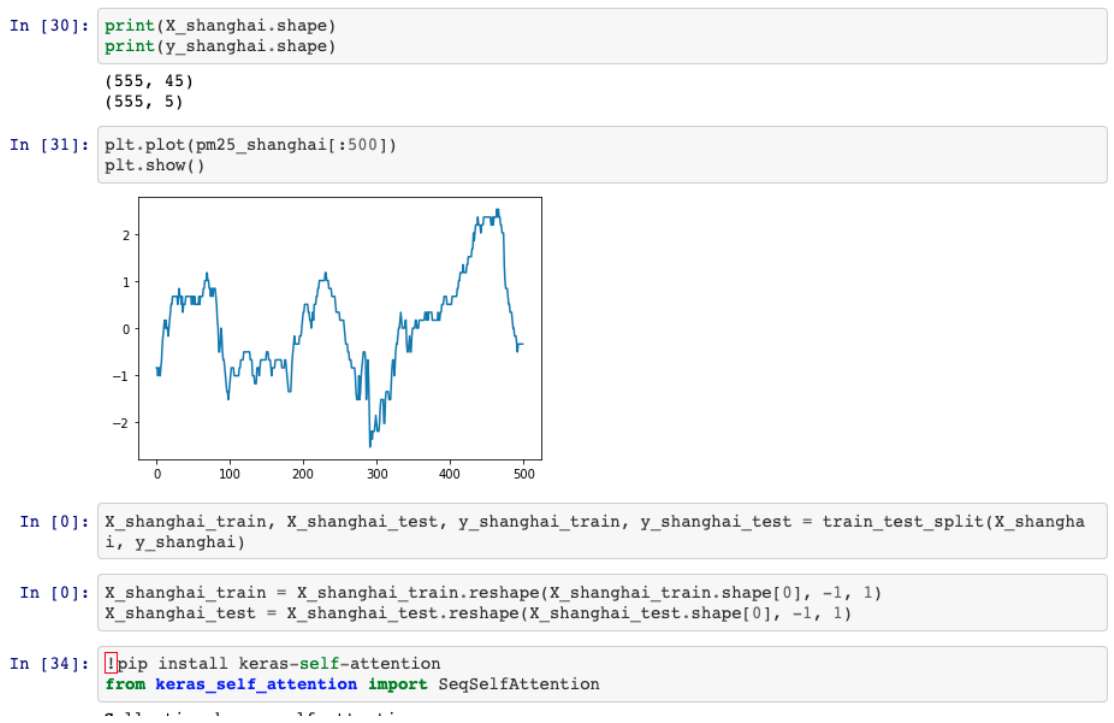
```
In [0]:  from keras.layers import LSTM, Bidirectional, Dropout
         from keras.models import Sequential
         from keras.layers import Dense, Flatten
         from keras.layers.convolutional import Conv1D, Conv2D, MaxPooling1D, MaxPooling2D

In [16]: X_beijing_train.shape

Out[16]: (31284, 45, 1)

In [0]:  model = Sequential()
         model.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=(X_beijing_train.shape[1],X_
         beijing_train.shape[2])))
         model.add(SeqSelfAttention(attention_activation='sigmoid'))
         model.add(LSTM(64))
         model.add(Dense(O))
         model.compile(optimizer='adam', loss='mse')

In [22]: model.fit(X_beijing_train, y_beijing_train, epochs=20, validation_data=(X_beijing_test, y_beijing_
         test))

         Train on 31284 samples, validate on 10428 samples
         Epoch 1/20
         31284/31284 [==============================] - 272s 9ms/step - loss: 0.2799 - val_loss: 0.1851
         Epoch 2/20
         31284/31284 [==============================] - 267s 9ms/step - loss: 0.1695 - val_loss: 0.1416
         Epoch 3/20
         31284/31284 [==============================] - 266s 9ms/step - loss: 0.1447 - val_loss: 0.1438
         Epoch 4/20
         31284/31284 [==============================] - 264s 8ms/step - loss: 0.1347 - val_loss: 0.1442
         Epoch 5/20
         31284/31284 [==============================] - 268s 9ms/step - loss: 0.1339 - val_loss: 0.1178
         Epoch 6/20
         31284/31284 [==============================] - 268s 9ms/step - loss: 0.1236 - val_loss: 0.1229
         Epoch 7/20
         31284/31284 [==============================] - 264s 8ms/step - loss: 0.1186 - val_loss: 0.1065
         Epoch 8/20
         31284/31284 [==============================] - 263s 8ms/step - loss: 0.1133 - val_loss: 0.1098
         Epoch 9/20
```

Figure A.3: Code Screenshot 3



```
In [0]: URL_shanghai = "https://github.com/vishal-pandey/iot-transfer/raw/master/pm25_shanghai.csv"

In [0]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import StandardScaler
        from sklearn.model_selection import train_test_split

In [0]: df_shanghai = pd.read_csv(URL_shanghai)

In [0]: pm25_shanghai = df_shanghai.iloc[:, 0].values

In [0]: pm25_shanghai = pm25_shanghai.reshape(-1, 1)

In [0]: scaler_shanghai = StandardScaler()
        scaler_shanghai.fit(pm25_shanghai)
        pm25_shanghai = scaler_shanghai.transform(pm25_shanghai)

In [0]: N = 20
        O = 5
        P = 20

In [0]: X_shanghai = []
        y_shanghai = []

        for i in range(len(pm25_shanghai) - (N + O + P)):
          temp = []
          temp1 = pm25_shanghai[i : i+N]
          temp2 = [0 for _ in range(O)]
          temp3 = pm25_shanghai[i+N+O : i+N+O+P]
          X_shanghai.append(np.append(np.append(temp1, temp2), temp3))
          y_shanghai.append(pm25_shanghai[i+N : i+N+O].reshape(O))

In [0]: X_shanghai = np.array(X_shanghai)
        y_shanghai = np.array(y_shanghai)
```

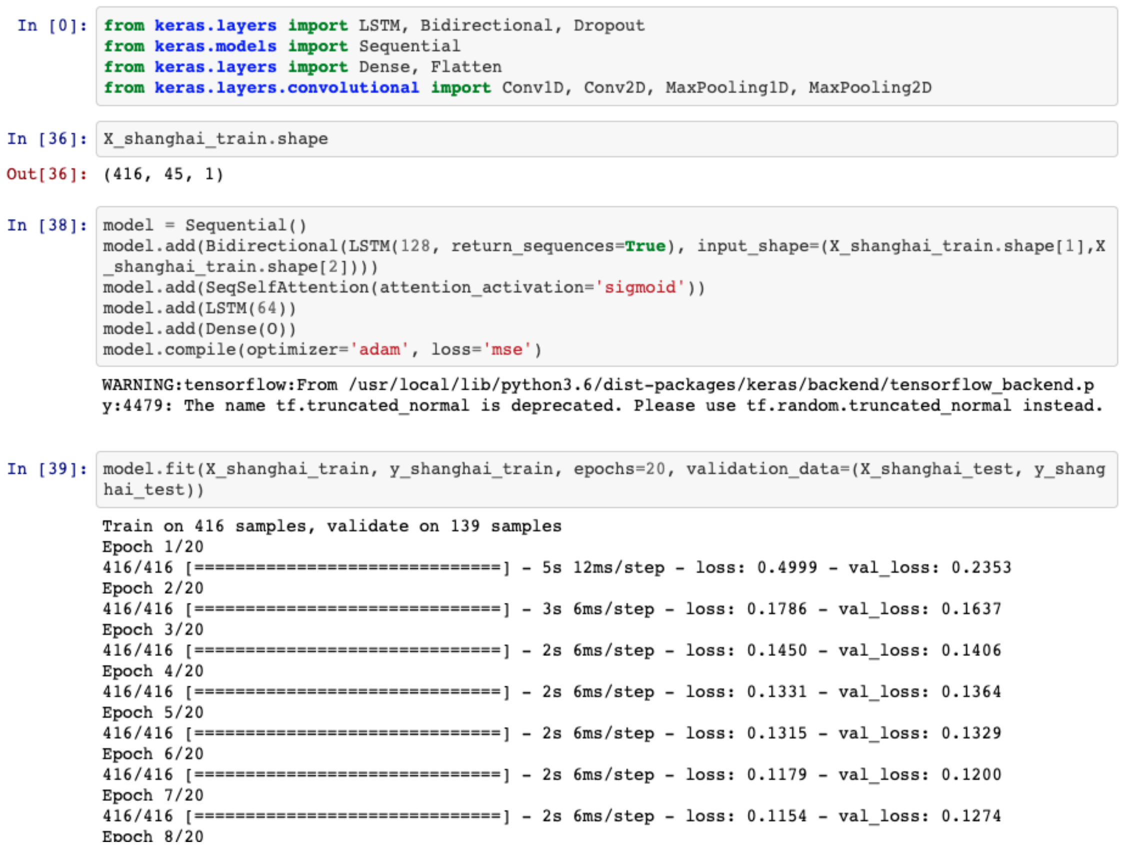
Figure A.4: Code Screenshot 4



```
In [30]: print(X_shanghai.shape)
         print(y_shanghai.shape)

         (555, 45)
         (555, 5)

In [31]: plt.plot(pm25_shanghai[:500])
         plt.show()
```

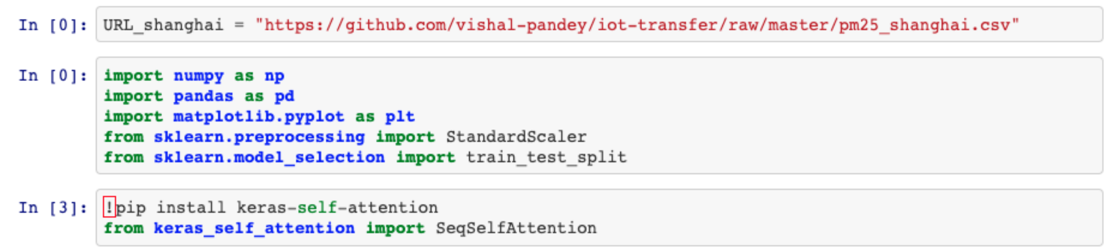
```
In [0]: X_shanghai_train, X_shanghai_test, y_shanghai_train, y_shanghai_test = train_test_split(X_shangha
        i, y_shanghai)

In [0]: X_shanghai_train = X_shanghai_train.reshape(X_shanghai_train.shape[0], -1, 1)
        X_shanghai_test = X_shanghai_test.reshape(X_shanghai_test.shape[0], -1, 1)

In [34]: !pip install keras-self-attention
         from keras_self_attention import SeqSelfAttention
```

Figure A.5: Code Screenshot 5



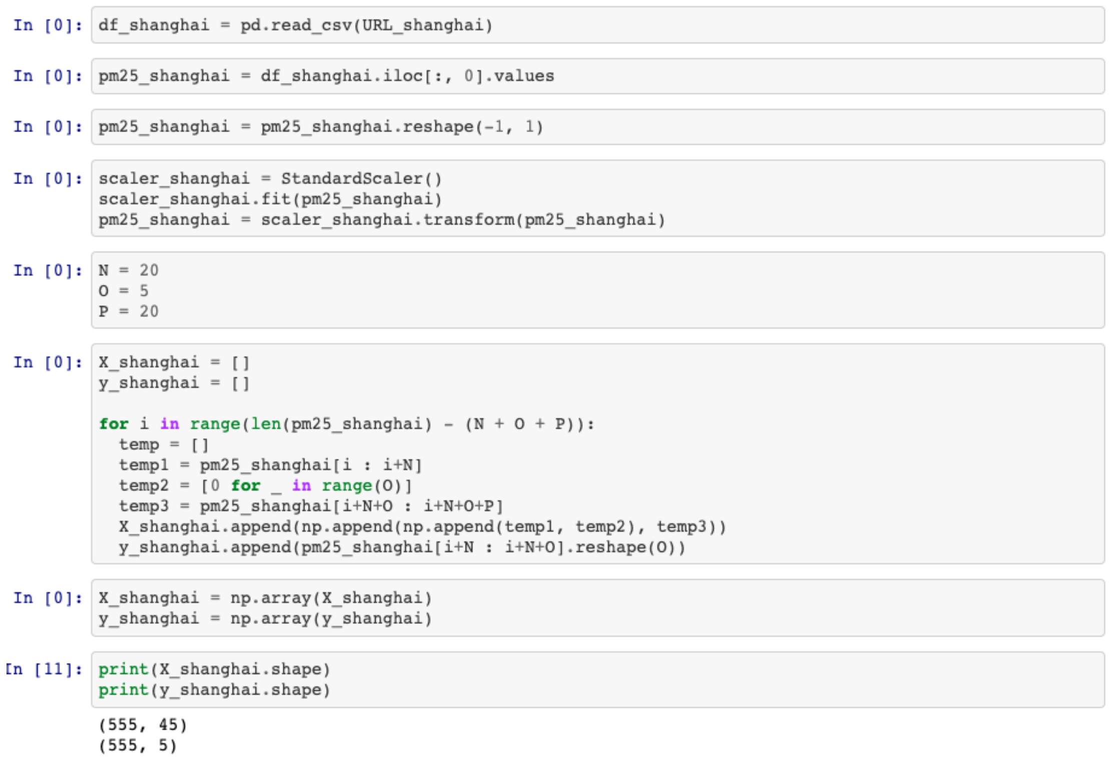
```
In [0]: from keras.layers import LSTM, Bidirectional, Dropout
        from keras.models import Sequential
        from keras.layers import Dense, Flatten
        from keras.layers.convolutional import Conv1D, Conv2D, MaxPooling1D, MaxPooling2D

In [36]: X_shanghai_train.shape

Out[36]: (416, 45, 1)

In [38]: model = Sequential()
         model.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=(X_shanghai_train.shape[1],X
         _shanghai_train.shape[2])))
         model.add(SeqSelfAttention(attention_activation='sigmoid'))
         model.add(LSTM(64))
         model.add(Dense(0))
         model.compile(optimizer='adam', loss='mse')

         WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.p
         y:4479: The name tf.truncated_normal is deprecated. Please use tf.random.truncated_normal instead.

In [39]: model.fit(X_shanghai_train, y_shanghai_train, epochs=20, validation_data=(X_shanghai_test, y_shang
         hai_test))

         Train on 416 samples, validate on 139 samples
         Epoch 1/20
         416/416 [==============================] - 5s 12ms/step - loss: 0.4999 - val_loss: 0.2353
         Epoch 2/20
         416/416 [==============================] - 3s 6ms/step - loss: 0.1786 - val_loss: 0.1637
         Epoch 3/20
         416/416 [==============================] - 2s 6ms/step - loss: 0.1450 - val_loss: 0.1406
         Epoch 4/20
         416/416 [==============================] - 2s 6ms/step - loss: 0.1331 - val_loss: 0.1364
         Epoch 5/20
         416/416 [==============================] - 2s 6ms/step - loss: 0.1315 - val_loss: 0.1329
         Epoch 6/20
         416/416 [==============================] - 2s 6ms/step - loss: 0.1179 - val_loss: 0.1200
         Epoch 7/20
         416/416 [==============================] - 2s 6ms/step - loss: 0.1154 - val_loss: 0.1274
         Epoch 8/20
```

Figure A.6: Code Screenshot 6



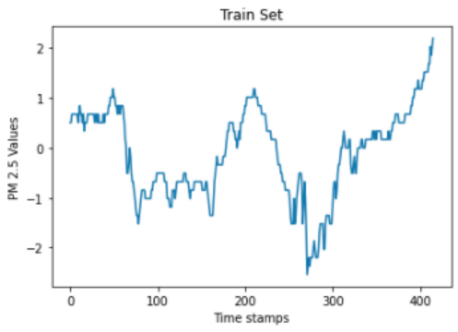
```
In [0]: URL_shanghai = "https://github.com/vishal-pandey/iot-transfer/raw/master/pm25_shanghai.csv"

In [0]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import StandardScaler
        from sklearn.model_selection import train_test_split

In [3]: !pip install keras-self-attention
        from keras_self_attention import SeqSelfAttention
```

Figure A.7: Code Screenshot 7



```
In [0]: df_shanghai = pd.read_csv(URL_shanghai)

In [0]: pm25_shanghai = df_shanghai.iloc[:, 0].values

In [0]: pm25_shanghai = pm25_shanghai.reshape(-1, 1)

In [0]: scaler_shanghai = StandardScaler()
        scaler_shanghai.fit(pm25_shanghai)
        pm25_shanghai = scaler_shanghai.transform(pm25_shanghai)

In [0]: N = 20
        O = 5
        P = 20

In [0]: X_shanghai = []
        y_shanghai = []

        for i in range(len(pm25_shanghai) - (N + O + P)):
          temp = []
          temp1 = pm25_shanghai[i : i+N]
          temp2 = [0 for _ in range(O)]
          temp3 = pm25_shanghai[i+N+O : i+N+O+P]
          X_shanghai.append(np.append(np.append(temp1, temp2), temp3))
          y_shanghai.append(pm25_shanghai[i+N : i+N+O].reshape(O))

In [0]: X_shanghai = np.array(X_shanghai)
        y_shanghai = np.array(y_shanghai)

In [11]: print(X_shanghai.shape)
         print(y_shanghai.shape)

        (555, 45)
        (555, 5)
```

Figure A.8: Code Screenshot 8



```
In [13]: plt.plot(y_shanghai[int(len(y_shanghai)*0.75):, 0])
         plt.xlabel("Time stamps")
         plt.ylabel("PM 2.5 Values")
         plt.title("Test Set")
         plt.show()
```

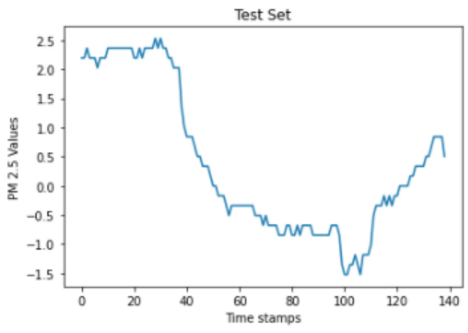


Figure A.9: Code Screenshot 9

```
Epoch 12/20
31284/31284 [==============================] - 276s 9ms/step - loss: 0.1031 - val_loss: 0.1092
Epoch 13/20
31284/31284 [==============================] - 275s 9ms/step - loss: 0.1051 - val_loss: 0.0951
Epoch 14/20
31284/31284 [==============================] - 276s 9ms/step - loss: 0.1004 - val_loss: 0.0927
Epoch 15/20
31284/31284 [==============================] - 277s 9ms/step - loss: 0.1001 - val_loss: 0.0982
Epoch 16/20
31284/31284 [==============================] - 277s 9ms/step - loss: 0.0984 - val_loss: 0.1043
Epoch 17/20
31284/31284 [==============================] - 274s 9ms/step - loss: 0.0993 - val_loss: 0.0980
Epoch 18/20
31284/31284 [==============================] - 280s 9ms/step - loss: 0.0943 - val_loss: 0.0927
Epoch 19/20
31284/31284 [==============================] - 280s 9ms/step - loss: 0.0958 - val_loss: 0.0887
Epoch 20/20
31284/31284 [==============================] - 276s 9ms/step - loss: 0.0944 - val_loss: 0.0898
Out[22]: <keras.callbacks.History at 0x7fde30c5b8d0>

In [23]: plt.plot(model.history.history['loss'], 'r.-', label="Training Loss")
         plt.plot(model.history.history['val_loss'], 'g.-', label="Validation Loss")
         plt.legend()
         plt.show()
```

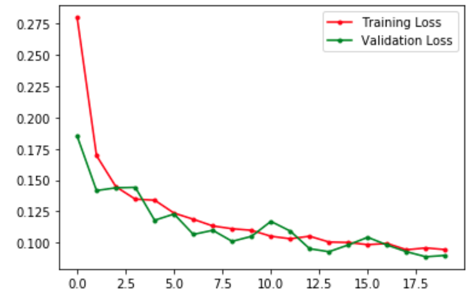


Figure A.10: Code Screenshot 10

```
416/416 [==============================] - 2s 6ms/step - loss: 0.0998 - val_loss: 0.0989
Epoch 12/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0916 - val_loss: 0.0874
Epoch 13/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0823 - val_loss: 0.0836
Epoch 14/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0816 - val_loss: 0.0887
Epoch 15/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0847 - val_loss: 0.0770
Epoch 16/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0818 - val_loss: 0.0767
Epoch 17/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0769 - val_loss: 0.0711
Epoch 18/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0706 - val_loss: 0.0755
Epoch 19/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0720 - val_loss: 0.0790
Epoch 20/20
416/416 [==============================] - 2s 6ms/step - loss: 0.0656 - val_loss: 0.0660
Out[39]: <keras.callbacks.History at 0x7f4300815c50>

In [40]: plt.plot(model.history.history['loss'], 'r.-', label="Training Loss")
         plt.plot(model.history.history['val_loss'], 'g.-', label="Validation Loss")
         plt.legend()
         plt.show()
```

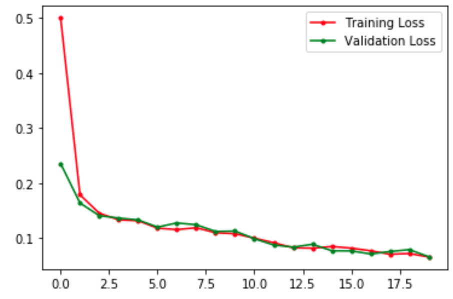
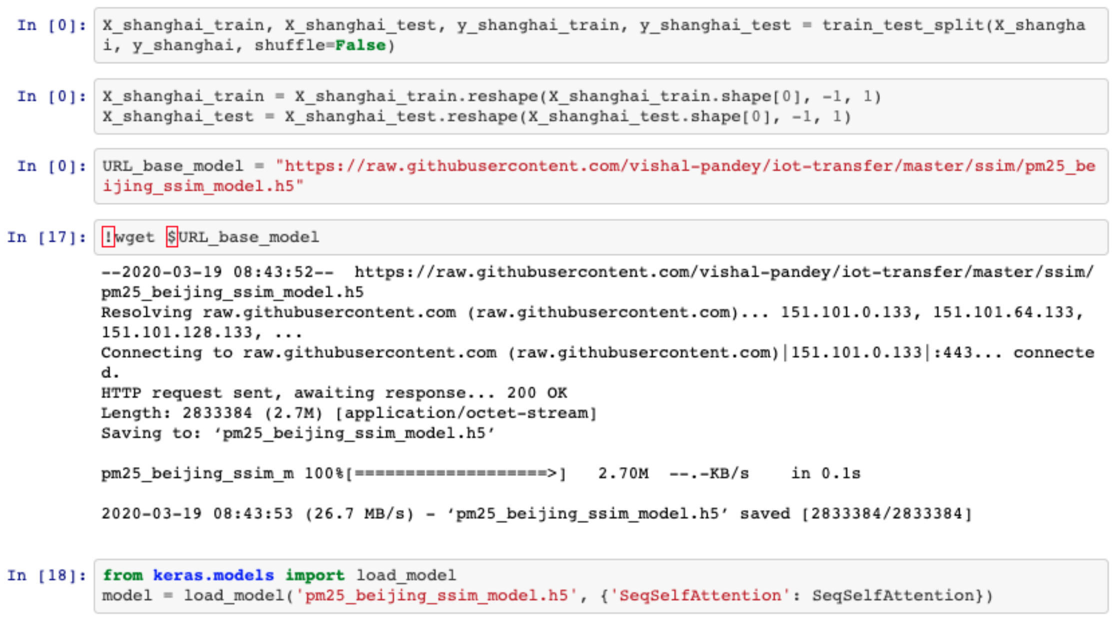
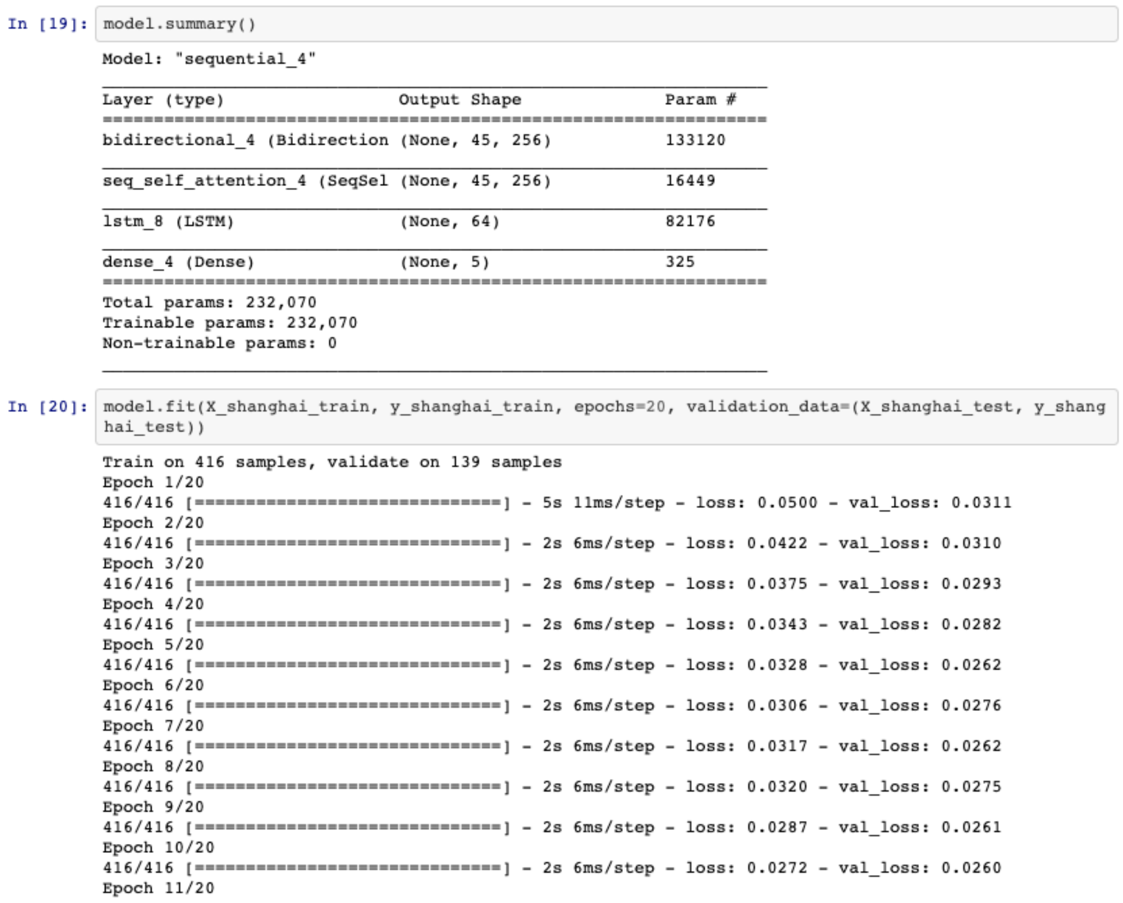


Figure A.11: Code Screenshot 11



```
In [0]: X_shanghai_train, X_shanghai_test, y_shanghai_train, y_shanghai_test = train_test_split(X_shangha
        i, y_shanghai, shuffle=False)

In [0]: X_shanghai_train = X_shanghai_train.reshape(X_shanghai_train.shape[0], -1, 1)
        X_shanghai_test = X_shanghai_test.reshape(X_shanghai_test.shape[0], -1, 1)

In [0]: URL_base_model = "https://raw.githubusercontent.com/vishal-pandey/iot-transfer/master/ssim/pm25_be
        ijing_ssim_model.h5"

In [17]: !wget $URL_base_model

        --2020-03-19 08:43:52--  https://raw.githubusercontent.com/vishal-pandey/iot-transfer/master/ssim/
        pm25_beijing_ssim_model.h5
        Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133,
        151.101.128.133, ...
        Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connecte
        d.
        HTTP request sent, awaiting response... 200 OK
        Length: 2833384 (2.7M) [application/octet-stream]
        Saving to: ‘pm25_beijing_ssim_model.h5’

        pm25_beijing_ssim_m 100%[===================>]   2.70M  --.-KB/s    in 0.1s

        2020-03-19 08:43:53 (26.7 MB/s) - ‘pm25_beijing_ssim_model.h5’ saved [2833384/2833384]

In [18]: from keras.models import load_model
         model = load_model('pm25_beijing_ssim_model.h5', {'SeqSelfAttention': SeqSelfAttention})
```

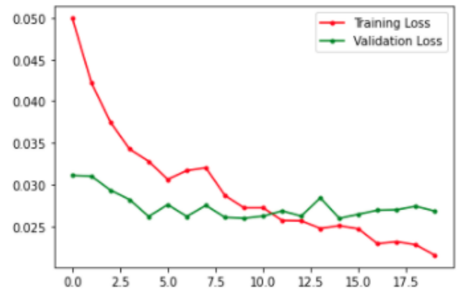
Figure A.12: Code Screenshot 12



```
In [19]: model.summary()

        Model: "sequential_4"
        _________________________________________________________________
        Layer (type)                 Output Shape              Param #
        =================================================================
        bidirectional_4 (Bidirection (None, 45, 256)           133120
        _________________________________________________________________
        seq_self_attention_4 (SeqSel (None, 45, 256)           16449
        _________________________________________________________________
        lstm_8 (LSTM)                (None, 64)                82176
        _________________________________________________________________
        dense_4 (Dense)              (None, 5)                 325
        =================================================================
        Total params: 232,070
        Trainable params: 232,070
        Non-trainable params: 0
        _________________________________________________________________

In [20]: model.fit(X_shanghai_train, y_shanghai_train, epochs=20, validation_data=(X_shanghai_test, y_shang
         hai_test))

        Train on 416 samples, validate on 139 samples
        Epoch 1/20
        416/416 [==============================] - 5s 11ms/step - loss: 0.0500 - val_loss: 0.0311
        Epoch 2/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0422 - val_loss: 0.0310
        Epoch 3/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0375 - val_loss: 0.0293
        Epoch 4/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0343 - val_loss: 0.0282
        Epoch 5/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0328 - val_loss: 0.0262
        Epoch 6/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0306 - val_loss: 0.0276
        Epoch 7/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0317 - val_loss: 0.0262
        Epoch 8/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0320 - val_loss: 0.0275
        Epoch 9/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0287 - val_loss: 0.0261
        Epoch 10/20
        416/416 [==============================] - 2s 6ms/step - loss: 0.0272 - val_loss: 0.0260
        Epoch 11/20
```

Figure A.13: Code Screenshot 13

```
In [21]: plt.plot(model.history.history['loss'], 'r.-', label="Training Loss")
         plt.plot(model.history.history['val_loss'], 'g.-', label="Validation Loss")
         plt.legend()
         plt.show()
```



```
In [0]: y_hat = model.predict(X_shanghai_test)
```

```
In [23]: plt.title("Actual vs Predicted")
         plt.plot(y_shanghai_test[:, 0], label="Actual")
         plt.plot(y_hat[:, 0], label="Predicted")
         plt.xlabel("Time Index")
         plt.ylabel("PM2.5")
         plt.legend()
         plt.show()
```

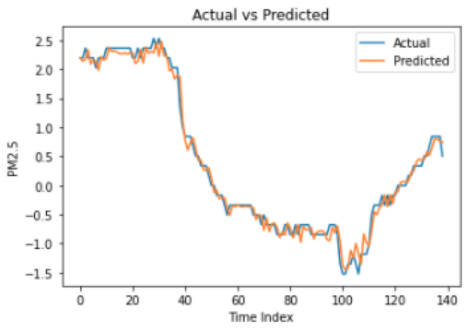


Figure A.14: Code Screenshot 14

Bibliography

[1] [M. Mousa, M. Abdulaal, S. Boyles and C. Claudel, "Wireless Sensor Network-Based Urban Traffic Monitoring Using Inertial Reference Data," 2015 International Conference on Distributed Computing in Sensor Systems, Fortaleza, 2015, pp. 206-207, doi: 10.1109/DCOSS.2015.21.

[2] [H. Furtado and R. Trobec, "Applications of wireless sensors in medicine," 2011 Proceedings of the 34th International Convention MIPRO, Opatija, 2011, pp. 257-261.

[3] [Bardella, A., Danieletto, M., Menegatti, E. et al. Autonomous robot exploration in smart environments exploiting wireless sensors and visual features. Ann. Telecommun. 67, 297–311 (2012). https://doi.org/10.1007/s12243-012-0305-z

[4] [M. R. M. Kassim and A. N. Harun, "Applications of WSN in agricultural environment monitoring systems," 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2016, pp. 344-349, doi: 10.1109/ICTC.2016.7763493.

[5] [@book10.5555/1643457, author = Mao, Guoqiang and Fidan, Baris and Mao, Guoqiang and Fidan, Baris, title = Localization Algorithms and Strategies for Wireless Sensor Networks, year = 2009, isbn = 1605663964, publisher = Information Science Reference - Imprint of: IGI Publishing, address = Hershey, PA

[6] [J. Zhang, W. Li, Z. Yin, S. Liu and X. Guo, "Forest fire detection system based on wireless sensor network," 2009 4th IEEE Conference on Industrial Electronics and Applications, Xi'an, 2009, pp. 520-523, doi: 10.1109/ICIEA.2009.5138260.

[7] [Susanto, Hengky & Chang, Chorng & Lalooses, Francine. (2007). AN APPROACH FOR TRACKING WILDLIFE USING WIRELESS SENSOR NETWORKS.

[8] [T. M. Thekkil and N. Prabakaran, "Real-time WSN based early flood detection and control monitoring system," 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kannur, 2017, pp. 1709-1713, doi: 10.1109/ICICICT1.2017.8342828.

[9] [R. Das, S. Ghosh and D. Mukherjee, "Bayesian Estimator Based Weather Forecasting using WSN," 2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, India, 2018, pp. 1-4, doi: 10.1109/ICRAIE.2018.8710410.

[10] https://www.electronicshub.org/wireless-sensor-networks-wsn/

[11] Matin, M.A. & Islam, M.M. Overview of Wireless Sensor Network; IntechOpen: London, UK (2012)

[12] S. Sharma, R. K. Bansal and S. Bansal, "Issues and Challenges in Wireless Sensor Networks," 2013 International Conference on Machine Intelligence and Research Advancement, Katra, 2013, pp. 58-62, doi: 10.1109/ICMIRA.2013.18.

[13] Palm G. (1986) Warren McCulloch and Walter Pitts: A Logical Calculus of the Ideas Immanent in Nervous Activity. In: Palm G., Aertsen A. (eds) Brain Theory. Springer, Berlin, Heidelberg

[14] F. Q. Lauzon, "An introduction to deep learning," 2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA), Montreal, QC, 2012, pp. 1438-1439, doi: 10.1109/ISSPA.2012.6310529.

[15] https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html

[16] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[17] T. Liu, T. Wu, M. Wang, M. Fu, J. Kang and H. Zhang, "Recurrent Neural Networks based on LSTM for Predicting Geomagnetic Field," 2018 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES), Bali, 2018, pp. 1-5, doi: 10.1109/ICARES.2018.8547087.

[18] A. Pulver and S. Lyu, "LSTM with working memory," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 845-851, doi: 10.1109/IJCNN.2017.7965940.

[19] Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. J Big Data 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6

[20] Bird, Jordan J.; Kobylarz, Jhonatan; Faria, Diego R.; Ekart, Aniko; Ribeiro, Eduardo P. (2020). "Cross-Domain MLP and CNN Transfer Learning for Biological Signal Processing: EEG and EMG". IEEE Access. Institute of Electrical and Electronics Engineers (IEEE). 8: 54789–54801. doi:10.1109/access.2020.2979074. ISSN 2169-3536

[21] Kim, T., & Kim, H. Y. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. PLOS ONE, 14(2), e0212320.

[22] Kyunghyun Cho, Bart van Merri¨ enboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

[23] H. Verma and S. Kumar, "An accurate missing data prediction method using LSTM based deep learning for health care," in Proc. 20th Int. Conf. Distrib. Comput. Netw., 2019, pp. 371–376

[24] D. S. Fung. Methods for the estimation of missing values in time series. 2006.

[25] A. C. Harvey. Forecasting, structural time series models and the Kalman filter. Cambridge university press, 1990.

[26] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf. Multiple imputation by chained equations: what is it and how does it work? International journal of methods in psychiatric research, 20(1):40–49, 2011.

[27] X. Yi, Y. Zheng, J. Zhang, and T. Li. St-mvl: filling missing values in geosensory time series data. 2016.

[28] J. Wang, A. P. De Vries, and M. J. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 501–508. ACM, 2006.

[29] H. Yuan, G. Xu, Z. Yao, J.Jia, and Y.Zhang, "Imputation of missing data in time series for air pollutants using long short-term memory recurrent neural networks," in Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput., 2018, pp. 1293–1300.

[30] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. Scientific reports, 8(1):6085, 2018.

[31] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014a. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Sys-tems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pages 3104–3112.

[32] C. Leke, B. Twala, and T. Marwala, "Missing data prediction and classification: The use of auto-associative neural networks and optimization algorithms," arXiv preprint arXiv:1403.5488, 2014

[33] Kyunghyun Cho, Bart van Merri¨ enboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

[34] Moghar, A., & Hamiche, M. (2020). Stock Market Prediction Using LSTM Recurrent Neural Network. Procedia Computer Science, 170, 1168–1173. doi:10.1016/j.procs.2020.03.049

[35] Yang, Y., Guizhong, L.: Multivariate time series prediction based on neural networks applied to stock market. In: 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236), vol. 4, IEEE (2001)

[36] Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing 50, 159–175 (2003)

[37] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in Ijcai, vol. 15, 2015, pp. 3995–4001.

[38] Zhang, Y.-F., Thorburn, P., Xiang, W., & Fitch, P. (2019). SSIM -A Deep Learning Approach for Recovering Missing Time Series Sensor Data. IEEE Internet of Things Journal, 1–1. doi:10.1109/jiot.2019.2909038