PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: MATHEMATICS AND INFORMATICS
DEPARTEMENT : CMPUTER SCIENCE
N° :……..

DOMAIN : MATHEMATICS AND COMPUTER SCIENCE
FIELD : COMPUTER SCIENCE
OPTION : SIGL

Dissertation submitted in partial fulfillment of the requirements for the degree of MASTER

By: Dilmi Abdelbaqi
Chikouche aboubaker

Subject

Failure prediction in cloud environment using deep learning

Publicly defended before the jury composed of :

Nacer Amroun	University of M'sila	Chair
Dr. Debbi hichem	University of M'sila	Supervisor
Meftah Lekhal	University of M'sila	Examiner

College year : 2019/2020

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: MATHEMATICS AND INFORMATICS
DEPARTEMENT : CMPUTER SCIENCE
N° :……..

DOMAIN : MATHEMATICS AND COMPUTER SCIENCE
FIELD : COMPUTER SCIENCE
OPTION : SIGL

Dissertation submitted in partial fulfillment of the requirements for the degree of MASTER

By: Dilmi Abdelbaqi
Chikouche aboubaker

Subject

Failure prediction in cloud environment using deep learning

Publicly defended before the jury composed of :

Nasser Amroun	University of M’sila	Chair
Dr. Debbi hichem	University of M’sila	Supervisor
Meftah Lekhal	University of M’sila	Examiner

College year : 2019/2020

Dedication

We would like to dedicate this modest work to our families especially our parents for their encouragement and support from day one until this moment.

For all the working staff of our department from administration to all the instructors.

Finally to all the people that we know through the years colleagues and friends.

Acknowledgements

In the name of Allah the most Merciful and Beneficent

First and Foremost, praise is to ALLAH, the Almighty, the greatest of all, on whom ultimately, we depend for sustenance and guidance. We would like to thank Almighty Allah for giving us opportunity, determination and strength to do our research. His continuous grace and mercy was with me throughout my life and ever more during the tenure of my research.

Now, we would like to thank and express our deep and sincere gratitude to our supervisor Dr. Hichem Debbi for his continuous support, guidance and encouragement. We appreciate all his contributions of time, support and ideas.

We would also like to thank all the Jury Members, who have agreed to review this work.

We thank all the teachers who guided us throughout our journey. Thanks to all who helped us.

Thanks

Contents

Lists of Figures

Table List

Abbreviation List

RNN: recurrent neural network

LSTM: long short-term memory

IOT: internet of things

MLP: multilayer perceptron

CNN: convolutional neural network

Tanh: hyperbolic tangent

RELU: rectified linear unit

FC: fully connected

ANN: artificial neural networks

BRNN: bidirectional recurrent neural network

BP/USD: British Petroleum/ united State dollar

US: united state

F: Fahrenheit

Mph: Miles per hour

Cov: covariance

ACF: autocorrelation function

AR: autoregressive

MA: moving average

ARMA: autoregressive Moving Average

ARIMA: autoregressive Integrated Moving Average

ARFIMA: autoregressive Fractionally Integrated Moving Average

SARIMA: seasonal autoregressive Integrated Moving Average

ARCH: autoregressive Conditional Heteroskedasticity

GARCH: Generalized autoregressive Conditional Heteroskedasticity

EGARCH: Exponential Generalized autoregressive Conditional Heteroskedasticity

GRU: gated research unit

IT: Information Technology

AWS: amazon web services

Iaas: Infrastructure as a Service

Paas: Platform as a Service

Saas: Software as a Service

VMs: multiple virtual machines

ITIC: Information Technology Intelligence consulting

LANL: los Alamos National Laboratory

SMART: self-monitoring analysis and reporting technology

SVM: support vector machine

FAR: failure detection rate

BP: Backpropagation

DOC: decentralized online clustering

CPU: center processing unite

SLLE: Supervised Local Linear Embedding

KNN: K-Nearest Neighbours

I/O: input/output

GB: Gigabyte

HQ: High Quality

RAM: Random Access Memory

GTX: Graphics Technology Xtreme

API: Application programing interface

SciPy: scientific python

D: dimension

RDP: remote desktop protocol

Introduction

Cloud computing has become one of the most interesting topics in the world of IT, many software industries migrated to cloud computing platforms to benefit from the offered services for a low cost such as Microsoft Azure, Google Cloud and Amazon AWS, where it offers many service models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These cloud services are used every day by millions of users, which requires a high service availability because a small failure could cost a lot. Cloud service providers have spent a lot of resources and made extraordinary effort to maintain a high reliability and availability. However, cloud computing services still facing many challenges, which affects the satisfaction of the user requests, these problems can be caused by a failure in software or hardware components, which leads to node and application failures. In this regard, a quite number of research works attempted to deal with these failures. Among these techniques, we mention predictive methods, which are based on predicting the failure before it happens. Although predictive methods could be very effective, we noticed that there is a low number of works that focused on node failure. A failure in cloud services can be caused by many reasons such as disk failure, network outage, misconfigurations, memory leak, hardware breakdown, etc.

Since cloud companies usually record every detail of their running machines, it is important to rely on these historical data, which is time series data to build and efficient techniques and methods in order to improve the reliability of a cloud system, as it is important for a reliable cloud system to understand the observed failures and predict these failures before it happens, which allows to allocate the task to another healthy machine.

In our project we build a prediction model based on LSTM, which is an extension of RNN to predict failures on Google cluster data set. Our approach is based on applying deep learning techniques on time series data to predict failures that can occurs in the future in a cloud environment, ANN made a revolution in the field of artificial intelligence by providing “deep learning” models, these models shown an efficient performance in many stages in real life problems, the model we proposed is evaluated based on many metrics and showed a good result.

CHAPTER 1:

MACHINE LEARNING

1 Introduction

This chapter is organized as follows, the first section defines machine learning and its basics, the next section contains deep learning definition, basics, architecture and applications, the last section of this chapter talks about deep neural networks, more specifically, recurrent neural networks (RNN) and long short-term memory (LSTM).

2 Machine learning

2.1 What is machine learning?

According to Cambridge dictionary [1], machine learning is “the process of computers changing the way they carry out tasks by learning from new data, without a human being needing to give instructions in the form of a program”.

It is a search field that came from the combination of statistics, artificial intelligence, and computer science, machine learning techniques gives computers the ability to do complex tasks that usually done by humans or animals, and going even further by doing tasks out of human capabilities.

Inspired by human learning with the time, let’s take a simple example of you trying to shoot a basketball through the circle, at first attempt you realize you didn’t put that much of power in it, at second attempt, you are closer to the circle but you have to change the angle slightly, basically here, after every throw you learn and improve the final result, that’s how computers learn and improve with machine learning.

2.2 Types of machine learning

Depending on the nature of received data and the desired output, machine learning types splits into four major learning categories [2], which are as follows:

2.2.1 Supervised learning

Supervised learning works with labeled training data, the algorithm learns from each example in this data trying to predict the correct output object, the learning phase continue until reaching a good performance result, it's the most favorite type of learning because it is a supervision learning which is easier to learn under it, the major disadvantage of this type is that the dataset has to be labeled by a machine learning engineer which is very costly process.

Supervised learning problems can be divided into regression and classification problems.

Classification problem

Basically when we have a known classes and the goal is to assign new inputs to one or more of these classes, famous example of classification is mail spams, a good classification filter can work greatly to catch it, the inputs are the email messages and the output assigning classes are "spam" or "not spam".

Regression problem

It is a supervised problem, where the outputs are real values or continuous quantity.

2.2.2 Unsupervised learning

In unsupervised learning, the data is not labeled nor classified, the algorithm try to determine the hidden patterns on its own, there is no correct or wrong output, it's more like a data mining as it is an actual learning, because this type of algorithm tend to restructure and provide new features, that might come in handy in data visualization and representation.

Unsupervised learning problems can be divided into regression and classification problems.

Clustering problem

Unlike classification, classes are not known, and the data meant to be divided into groups, clustering is based on similarity when organizing the groups, social media is a good example on applying clustering to determine communities of users, you definitely faced a specific

advertising your Facebook account, they used some techniques to make advertising go to specific groups of users.

Association problem

Association rules are basically just if/then statements used to discover relationships between data in databases or repositories.

2.2.3 Semi-supervised

A mixt of both above types, where some of training data has desired outputs while the other does not have it, this type used to counter the disadvantages of both previous types.

2.2.4 Reinforcement learning

Considered the most ambitious type of machine learning, while supervised learning trained with the correct output while in reinforcement learning there is no answer, but the algorithm who decides what to do to complete the given task, in this type the algorithm learns from its errors in each previous output where the current output depends on the input, and the next input depends on the previous output.

2.3 Machine learning applications

Machine learning algorithms at the moment are spreading in so many fields, and the achieved results are highly great, this is some application fields where machine learning can be applied:

Stock trading

Many platforms available to help make better stock trading, where the platform tend to do a big amount of analysis and processing to give recommendations, a machine learning algorithm can learn from some information like the historical opening and closing prices and the buy and sell of the stock, where 70 percent of trades are performed by machine learning not by humans.

Voice recognition

Siri service on apple’s devices is famous example of machine learning, you ask Siri, and it process what you want, Siri uses a device and cloud-based statistical model to perform the above process.

Medicine and healthcare

The use of machine learning with the big data can be used in healthcare analytics, which a number of companies are looking for to provide a well-informed data to help make better decisions.

Information like weight, heart rate, pulse, pedometers, blood pressure and blood glucose levels are now available on smartphones, which allows to track and trace user health regularly, then machine learning systems can recommend some healthy tips via the device.

The internet of things

IOT is basically a connected computing device that can transfer data over a network via a device-to-device communication, now these devices are being used at home as much in industry.

Technologies like Arduino and raspberry Pi computers are used in measuring of motion, temperature and sound then use these extracted data for analysis.

Many other fields are already integrated the machine learning algorithms to automate and help perform various of needed tasks, robotics, social networks, finance and others and we showcased some of them in this section.

2.4 Machine learning languages

There is a several languages that can be used in the field of machine and there is no absolutely better language than another, it all back to the picking of the right tool for a specific job, and the following are some of languages can use for machine learning

Python

Starting with python because it is the most famous one in machine learning due to its syntax simplicity, which make it easy to learn, easy to read, python has a large ecosystem, support object-oriented programming, and has a large community, that is why python is growing very fast.

R

R is an open source statistical programming language, it has a large number of machine learning packages and visualization tools, R syntax is not that easy to learn.

MATLAB

The MATLAB language is widely used for technical computing and algorithm creation; it has some features like R, such as plotting and graphs.

Java

A widely used open source language that has a great number of libraries, java existed for a long time; it is a good choice for build complex and large machine learning applications.

2.5 Machine learning cycle

Basically, a machine learning project is a cycle of actions needs to be performed.

Starting by collecting data, then accept that data to be cleaned and checked for quality, after that done, the processing phase performed by machine learning techniques you create.

Finally, the results are presented, the reporting can happen in a variety of ways.

3 Deep learning

3.1 What is deep learning?

Deep learning has been used in many tasks and applications today, especially with the amount of data generated every day that allows to use deep learning techniques properly, because it requires a huge amount of data to learn from it, that's what differ deep learning from other approaches.

Deep learning is a subfield of machine learning, inspired from human brain, where non-biological neural networks used to handle a large amount of data, and learn from it, the architecture of artificial neural networks is similar to the biological neural networks, where the simplest neural network referred to as the perceptron, that has a single input layer and an output node that use an activation function.

Considering a simple neuron:

$$y = \sum(weight * input) + bias \quad (1.1)$$

Where "y" is the output value of given neuron.

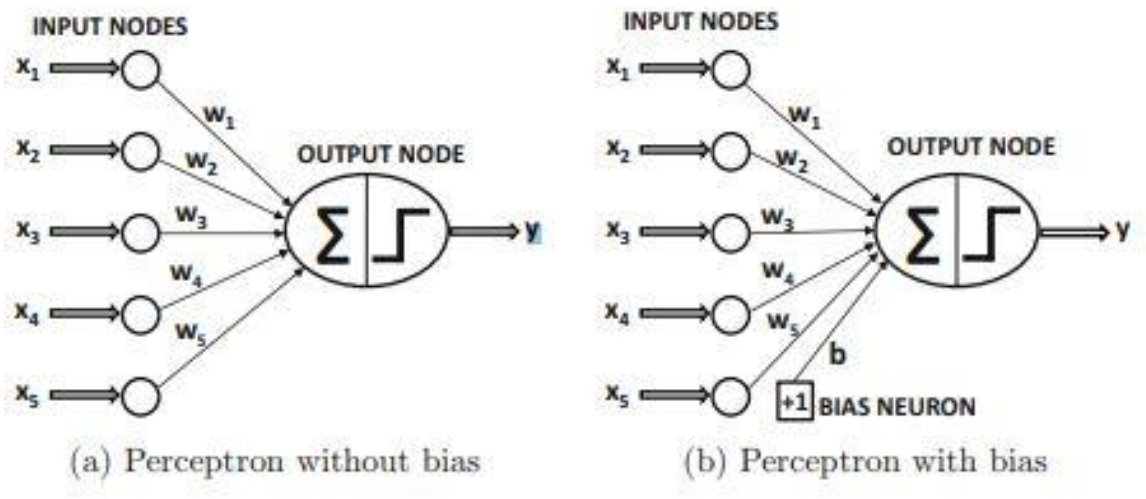

Figure 1.1: perceptron architecture

Furthermore, artificial neural networks can have many layers and more than one output, where neural networks are arranged in multiple layers, every layer consists of a number processing units ”nodes”, each node have its own modifiable value in it called “bias” represented by “b”, , and between every two connected nodes from different layers there is a weight, the output of each node is the product of the weight and the received value from the previous layer, where every layer is connected to the next one, the learning process in the neural networks is the modification of weights value which effect the initial value will be forwarded to a given neuron, the modification of weights in each iteration is based on backpropagation approach, this layer architecture called “feedforward network” , the deep neural networks showed a great results in many fields including speech recognition, machine translation, bioinformatics, prediction and many others.

3.2 Backpropagation

Backpropagation is the core of learning for deep learning, it’s a method used to train artificial neural networks, basically it regulates the weights of neurons to get better output results until reaching satisfying results, and this approach minimizes the error function based on derivative calculations.

Mathematically backpropagation is something like this [3]:

$$W_{updated} = W_{old} - \eta \nabla E \quad (1.2)$$

Where W is the weight, η is the learning rate, and E is the cost function measuring overall performance, the learning rate is controlled by how much we should update, backpropagation relays on derivatives of the function (E), measuring how E changes when the weights changes, that's mean we should find the derivatives of E.

$$E = \frac{1}{2}\sum_{o \in Output}(t_0 - y_0)^2 \quad (1.3)$$

Where t is the target and y is the predicted output, then we turn the difference between them into an error derivation:

$$\frac{\partial E}{\partial y_0} = -(t_0 - y_0) \quad (1.4)$$

This chain of derivatives can be repeated through many layers we want.

There are many other algorithms to train artificial neural networks, the commonly used ones are the optimization algorithms like the gradient descent and Adam optimizer.

3.3 Activation function choice

As we have seen above, a neuron calculates the weighted sum of its inputs, then adds the bias before decides whether it should be activated or not, the output is a value and the neuron does not know the bounds of the value to decide based on it, to do so the activation function is used to make a decision whether to activate the neuron or not, there is many activation functions and the choice is critical, and it depends on what we want from the application to do.

Some activation functions explanation:

1/linear function:

A linear function has an equation of form: A= cx, rarely used because the derivative is always a constant, which not practical for backpropagation learning to train the model.

2/sigmoid function:

Equation: $A = \frac{1}{(1+e^{-x})}$

A non-linear function, generally used in output layer, it gives a clear prediction where the results bounded between 0 and 1, the problem of this function is vanishing gradient in case of very low values of x.

3/tanh function:

Often used in the hidden layers because its values are between -1 and 1, the key feature of tanh function that is a zero centered, which bring the mean close to 0, making learning for the next layer much easier.

4/ReLu function

A non-linear function that gives x as an output if x is positive and 0 otherwise, ReLu is computationally efficient by allowing the network to converge rapidly, meaning that ReLu learns much faster than the functions mentioned above.

5/SoftMax function:

A special type of sigmoid function, good for classification problems, because it's able to handle multiple classes, SoftMax is generally used in the output layer of the classifier.

3.4 Hyperparameters of a neural network

Hyperparameters are basically a set of variables that determine the network structure, and cannot be learned by the model, example of it is the number of unites in a hidden layer, while backpropagation is used to adjust weights by the model itself, the numbers of layers and units of each layer and even the learning rate and the dropout called hyper parameters, and they configured manually, there is no scientific way to adjust these parameters.

A standard way to find a good set of hyper parameters is by splitting the data into training and testing data, for example let's take 10% of data as testing set and 10% for validation and the rest 80% for training [3], the idea here is to train the model on the train set and test it on the validation set, if the results not good enough we re-train the model again and test on validation until we get a good results, then we test on the test set, but there is a problem can occur here which the "overfitting", it means that the model perform very poorly on unseen test data even if the model predict perfectly in train data, to face this problem we need to choose the right hyper parameters to fit the model on the data, but it not that simple to get it in that easy, avoiding the

problem of overfitting and get a right hyper parameters can be done by using some techniques such as regularization and cross validation.

3.5 Neural networks types and applications

Neural network nowadays is widely used in so many applications such as computer vision, speech recognition, natural language processing, and other application areas where deep learning solved many problems, there is many types of neural networks each type used for a specific problem, let's take a look at the most interesting types and its applications in real-world problems:

3.5.1 Multilayer perceptron

MLP are feed forward neural networks with many hidden layers, where each node in the current layer is connected to all the nodes in the next layer, which makes it a fully-connected neural networks, the learning is based on the backpropagation method, MLP uses a nonlinear activation function, usually the sigmoid function.

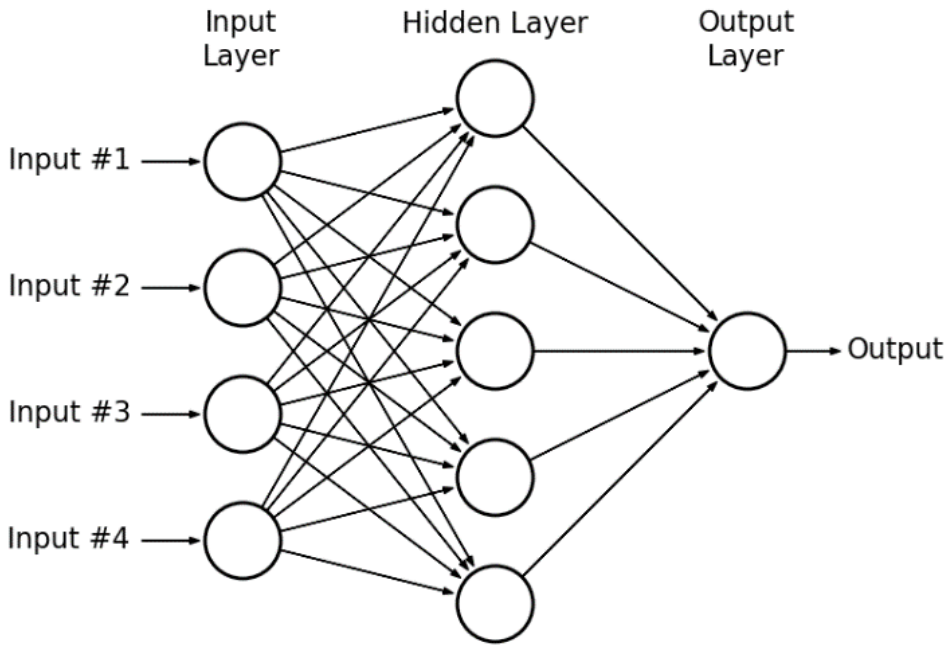

Figure 1.2: basic MLP architecture

And these are some applications of feed forward neural networks:

-complex classification

-machine translation

-speech recognition

3.5.2 Convolutional neural network

A CNN uses a variation of the multilayer perceptron's, it contains one or more than one convolutional layer, the convolutional layer processes the inputs then feed the results to the next layer, which is a fully connected neural network, CNNs shows efficient results in many fields such as:

-image and video recognition

-natural language processing

-text classification

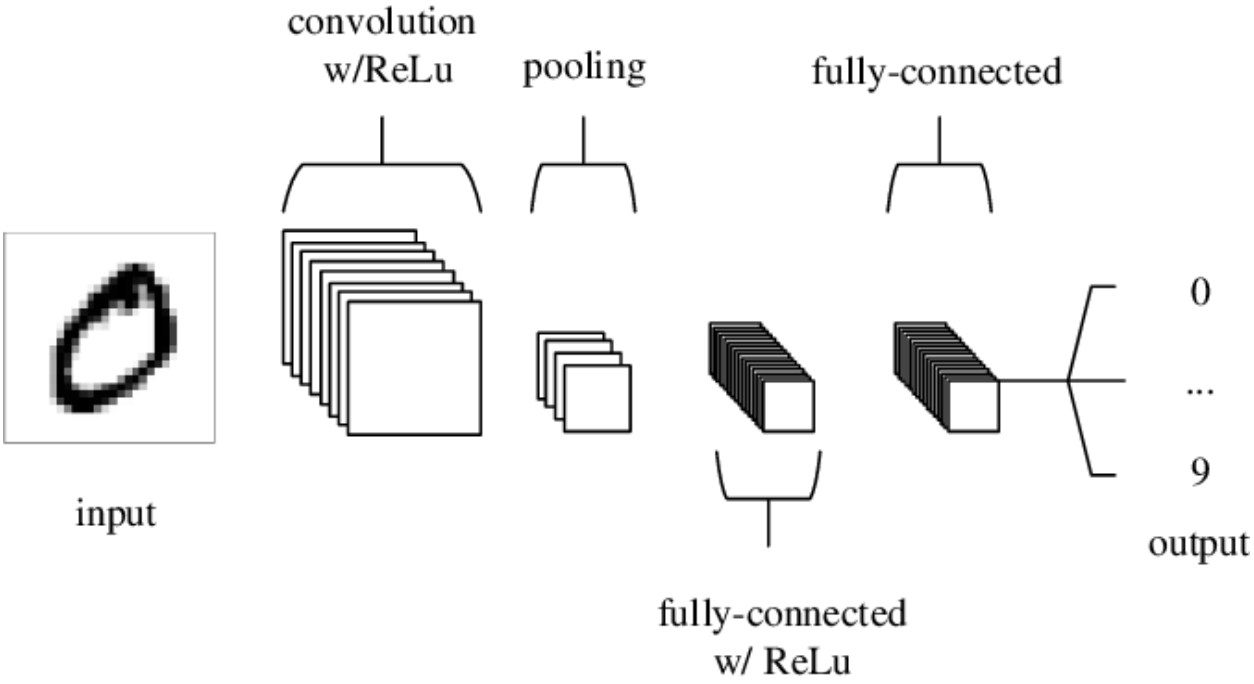

Figure 1.3: CNN basic architecture

Input: the input data for example if the input is an image, then the input consists of the raw pixel values of this image.

Convolutional layer: compute the output of neurons that are connected to local regions in the input.

RELU layer: apply an elementwise activation function, such as the max (0, x) thresholding at zero.

POOL layer: perform a down sampling operation to reduce the spatial size of the representation and the amount of parameters and computation in the network.

Fully connected layer: the output of pooling layer acts as input to the FC layer, which is a traditional ANN containing an activation function that produce a prediction.

3.5.3 Recurrent neural networks

RNNs are a specific type of neural networks where the output of a layer fed back to the input, RNNs used on sequential data where in each timestamp is dependent on the previous, examples of using RNNs:

-text processing

-machine translation

-anomaly detection

In the upcoming section, we will take an in-depth look at RNNs.

3.5.4 long-short term memory

A special type of RNN, used to solve the problem of forgetting long term dependencies in RNNs, where each node in the network has a specific cell to store these kinds of dependencies, and these are various applications of LSTM:

-time series prediction

-handwriting recognition

-stock prediction

-text prediction

The last section of this chapter will provide a detailed view about LSTM.

4 Recurrent neural networks

Generally speaking, typical neural networks architectures are designed for multi-dimensional data where attributes largely independent one another and the learning based on feedforward approach, but what about other types of data like time series? How to deal with it?

Time series data values are sequential and closely related to each other on successive timestamp, for these specific data, and if we treat the data at independent timestamps, then the information about relationships of values is lost, RNNs proposed to handle this problem, in recurrent neural networks there is a one-to-one correspondence between the layers in the network and the specific positions in the sequence.

4.1 What are recurrent neural networks?

RNNs are a special type of neural networks that has an extra twist, which is that the output fed back to itself, we generally called it a self-loop, furthermore the output of a layer can be also fed back to the previous layer, in other words an output at given time (t) is depended on the previous output (t-1), so we can think that RNNs have a "memory".

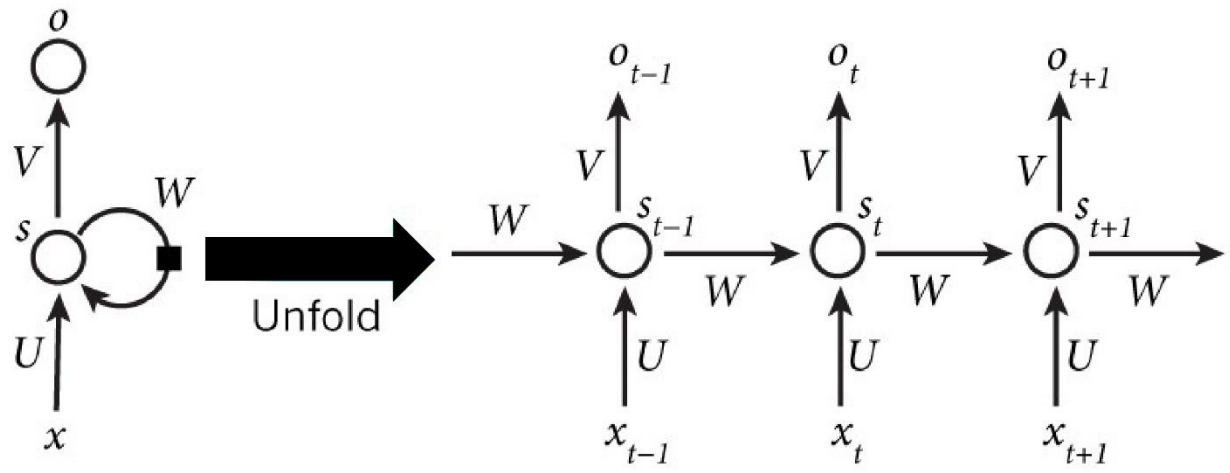

Figure 1.4: RNN architecture

The current state can be formulated like this:

$$S_t = f(U_{x_t} + W_{s_{t-1}})$$

4.2 Bidirectional Recurrent Networks

The problem with simple RNNs is that the state at a given time has an information only about past output, in some cases we need the knowledge about both past and future, BRNN have two hidden states for backward and forward directions, and they do not interact with each other, the backward states interacts only with each other and the same for forward, which makes training similar to RNNs, starting first with computing the hidden states in the forward direction, and then run in the backward direction, after that a backpropagation algorithm is applied.

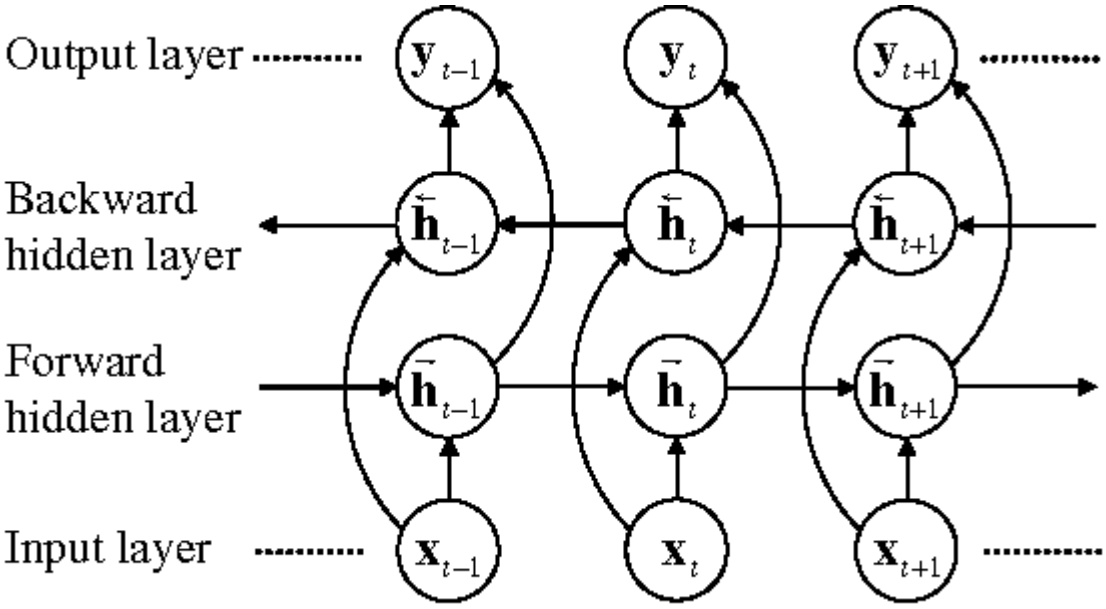

Figure 1.5: BRNN architecture

BRNNs widely used in applications where the prediction based on bidirectional context, an example of it is handwriting where the properties of one individual element depend on both sides of it, we could replace BRNN with two separate RNNs, one for forward and the other use the reverse input, and achieve almost the same result, but the difference here is that the two types trained combined, which means the two types do not interact directly one another, that's results a weak integration.

5 Long short-term memory

5.1 What is a long short-term memory?

A long short-term memory is a special kind of RNN have the capacity to learn long term dependencies, LSTMs were introduced to the world in 1997 by Hoch Reiter & Schmid Huber. LSTMs have the ability to avoid the long-term dependency, remembering information for long periods of time, usually LSTMs outperform the typical RNNs because the raising problem with RNNs is that they forget very quickly while the goal is to remember long term dependencies, while LSTMs can memorize them, the feature that gives LSTMs this capability is the long-term memory.

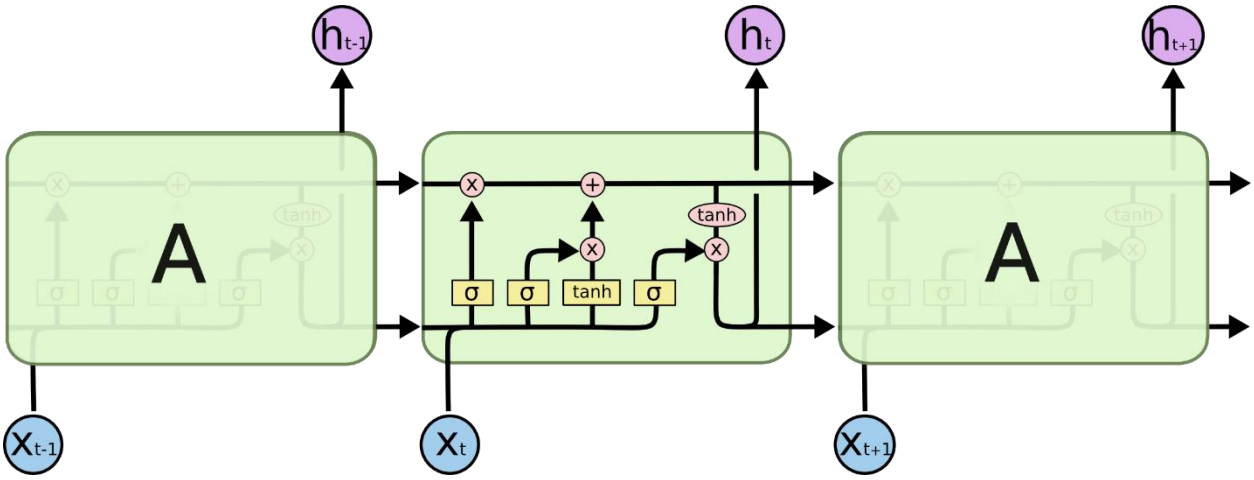

Figure 1.6: LSTM architecture

The long term memory called cell state, this cell can be modified by the forget gate, the previous cell state multiplied with the forget gate and then adds the new information coming from the input gates, the forget gate basically decides which information to forget by multiplying a specific position in the matrix with 0, it uses a sigmoid function, if the output is 1 the information is kept in the cell state, the modification of a cell state based on two functions, sigmoid for update and tanh to create a vector, with the cell state updated, the output is corresponding to the updated cell state, choosing a fraction of cell to go through a tanh function to be the output, the change in the state cell effects the state passed from a time unit to the next, the following equations illustrates step by step LSTM walk through:

Cell state is modified by the forget gate and the input gate.

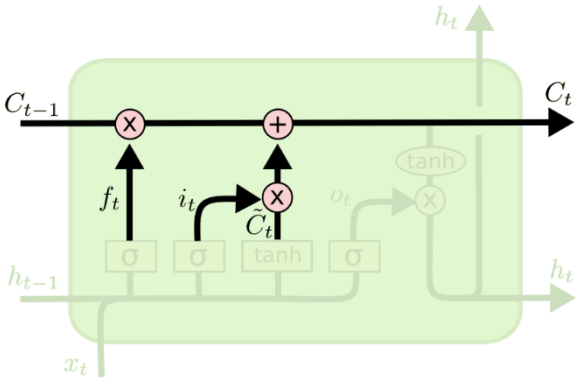

Figure 1.7: cell state architecture

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

Cell state equation

The forget gate output decides which information to forget using a sigmoid function.

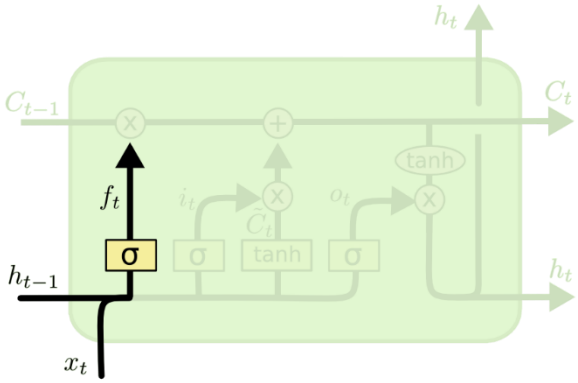

Figure 1.8: forget gate architecture

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Forget gate equation

The input gate determines which information to enter the cell state, the input gate is a sigmoid function, and the input modulation gate is a tanh function to allow the state cell to forget memory.

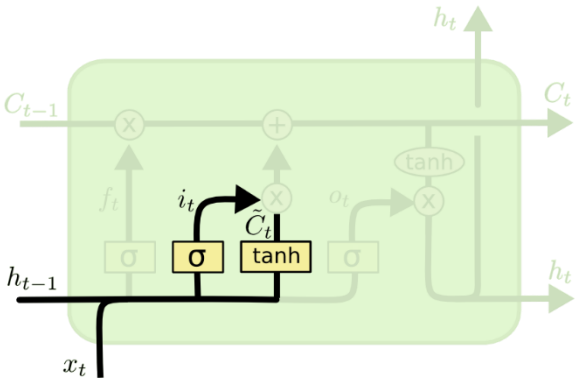

Figure 1.9: input gate architecture

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

Input gate function

$$\tilde{c}_{t=}\tanh(W_c[h_{t-1}, x_t] + b_c)$$

Input modulation gate function

The output gate choose value from the matrix should be pass through to the next hidden state, the hidden state decides which information should take to the next sequence.

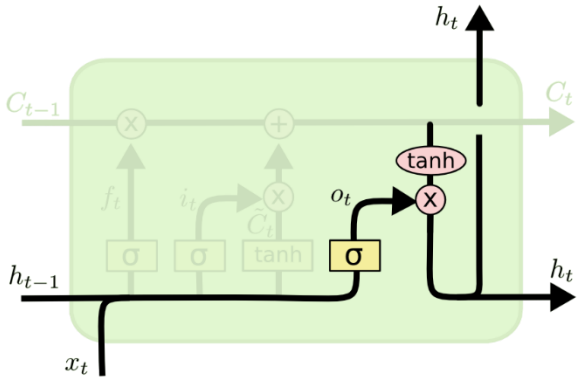

Figure 1.10: output state architecture

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

Output gate function

$$h_t = o_t * \tanh(c_t)$$

Hidden state equation

Chapter two:

Time Series

1 Introduction

Time series is a wide search area that has a big importance in many practical domains, the main purpose of time series is to collect and study the past observations to come with a model which describes the structure of the series. Then use this model to predict new values of this series. This act of predicting can be termed as time series forecasting, which means predicting the future depending on understanding the past observations. Over many years a lot of exertion has been done by researchers to develop a proper and efficient model to improve the forecasting performance.

In this chapter we will present a definition of time series, components, characteristics, and the most famous proposed models.

2 Definition of time series

A time series is a sequential set of data points, observed over successive times, mathematically, time series is a sequence of vectors x (t), t = 0,1,2,... where t represents the time, and the variable x(t) is the random variable. A time series that records a single variable is termed as univariate, but if the records consider more than one variable, it is termed multivariate.

A time series can be continuous or discrete, where in continuous time series observations measured in all points of time, whereas in discrete time series observations measured only at discrete points of time. For example, heart rate readings can be recorded in continuous time series, while population of a particular city represented in discrete time series. [4]

3 Components of a Time Series

The values of an observation in a time series can be affected generally by four main components, these components are: trend, seasonal, cyclical and irregular components, and this is a description of each one them.

Trend:

The trend can be defined as the general tendency of a time series to increase, decrease or stagnate over a long period of time, in other words a trend is a long term movement in a time series, for

example human population growth time series show upward trend, while time series of mortality rates observation show downward trend.

Seasonal:

Seasonal variations in time series are periodic fluctuations over a year during the season, seasonal variations caused by many factors such as: climatic conditions, traditional habits, etc. for example sales of ice-cream increase in the summer.

Cyclical:

The cyclical variation in time series usually last for a year or more, it is a notable changes in the series due to physical causes, which repeat in cycles, in financial and business time series there is a kind of cyclical variation as the business cycle consist of four phases:

i) Prosperity, ii) Decline, iii) Depression and iv) Recovery. [4]

Irregular:

The irregular variations are totally unpredictable, and caused by incidents such as earthquake, war, flood, etc. these influences are not regular and do not repeat in cycles, there is no defined technique to for measuring these random fluctuations.

4 Time series examples

Usually to visualize patterns such as trends and cycles, time series plots are used, depending on the nature of analysis needed there can be different kinds of time series, below we show some time series plots:

Figure2.1: Weekly BPS/USD exchange rate series (1980-1993)

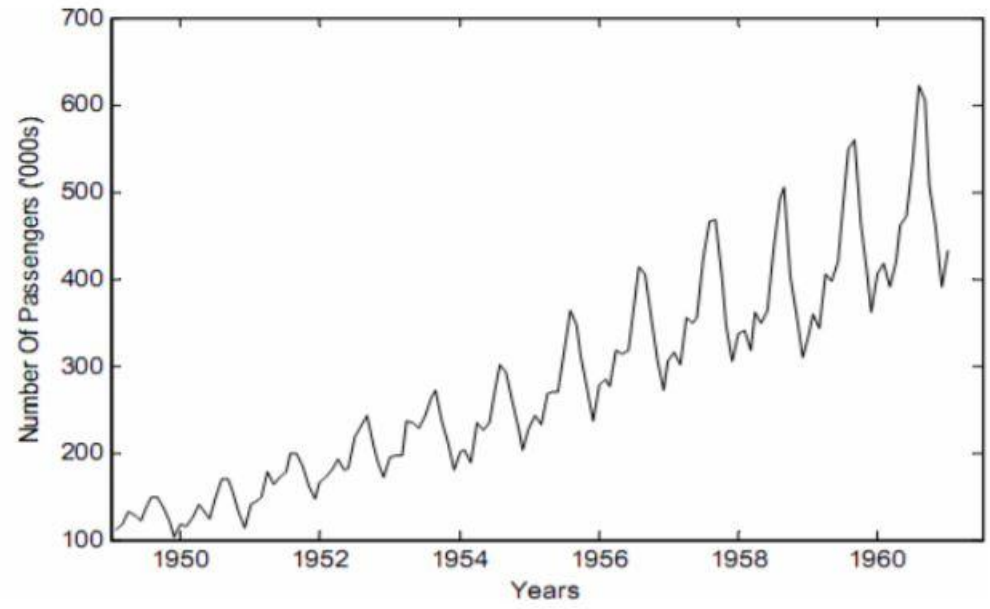

Figure 2.2: Monthly international airline passenger series (Jan. 1949-Dec. 1960)

The figure 1.1 shows a weekly exchange rate between British pound and US dollar from 1980 to 1933.

The figure 1.2 is a seasonal time series, shows the number of international airline passengers (in thousands) between Jan. 1949 to Dec. 1960 on a monthly basis.

5 Time series analysis

The procedure of fitting a time series to a proper model is termed as Time Series Analysis, it consists of methods that attempt to understand the nature of time series and it's useful for future forecasting and simulation.

To develop a suitable model to use it in forecasting, it requires collecting and analyzing past observations, then future events are predicted using the model, time series forecasting has been used in many applications, important decisions are taken based on predicted results, to make a good prediction, fitting an adequate model to time series is very important, over the past decades many research have been done to develop a suitable models.

5.1 Univariate vs Multivariate time series

In time series analysis, time series data can be univariate or multivariate depending on the number variables that vary over time, univariate analysis is the simplest one where the time series consists of one single variable, thus the past observations over time are only of one variable and it does not deal with causes nor relationships, a typical example of univariate time series is the temperature readings below, where the forecasting in this case based only on the historical values of temperature.

Time	Temperature
5:00 am	59 °F
6:00 am	59 °F
7:00 am	58 °F
8:00 am	58 °F
9:00 am	60 °F
10:00 am	62 °F
11:00 am	64 °F
12:00 pm	66 °F
1:00 pm	67 °F
2:00 pm	69 °F
3:00 pm	71 °F
4:00 pm	71 °F
5:00 pm	71 °F
6:00 pm	69 °F
7:00 pm	68 °F
8:00 pm	65 °F
9:00 pm	64 °F

Table 2.1: temperature readings [12]

Univariate time series analysis has some limitations especially in the determination of relationship between two or more variables, correlations, comparisons, causes, explanations, and contingency between variables., that is multivariate time series analysis used to cover these limitations, where multivariate time series analysis deals with two or more inputs, it incorporate not only the target variable's past observations but also other variables observed simultaneously over time, it focuses on studying the relationships and the correlation between variables over time, taking the above example, if we include other factors that can affect the temperature then then there is multiple variables in consider to predict the temperature, the table below illustrates this case.

Time	Temperature	cloud cover	dew point	humidity	wind
5:00 am	59 °F	97%	51 °F	74%	8 mph SSE
6:00 am	59 °F	89%	51 °F	75%	8 mph SSE
7:00 am	58 °F	79%	51 °F	76%	7 mph SSE
8:00 am	58 °F	74%	51 °F	77%	7 mph S
9:00 am	60 °F	74%	51 °F	74%	7 mph S
10:00 am	62 °F	74%	52 °F	70%	8 mph S
11:00 am	64 °F	76%	52 °F	65%	8 mph SSW
12:00 pm	66 °F	80%	52 °F	60%	8 mph SSW
1:00 pm	67 °F	78%	52 °F	58%	10 mph SW
2:00 pm	69 °F	71%	52 °F	54%	10 mph SW
3:00 pm	71 °F	75%	52 °F	52%	11 mph SW
4:00 pm	71 °F	78%	52 °F	52%	11 mph SW
5:00 pm	71 °F	78%	52 °F	52%	12 mph SW
6:00 pm	69 °F	78%	52 °F	54%	11 mph SW
7:00 pm	68 °F	87%	53 °F	60%	12 mph SW
8:00 pm	65 °F	100%	54 °F	66%	11 mph SSW
9:00 pm	64 °F	100%	55 °F	72%	13 mph SSW

Table 2.2: temperature readings depending on other variables [13]

5.2 Stochastic Process

Stochastic Process is often used in modeling time series data, a time series with random variables x1, x2, x3, . . ., where x1 is the random variable taken at the first time point, and x2 is the random variable taken at the second time point, and so on, a collection of random variables, {xt}, indexed by t is referred to as a stochastic process.

Thus, the sequence of observations of the series is actually a sample realization of the stochastic process that produced it. [4]

5.3 Stationary Time Series

The concept of stationarity is a stochastic process, which is a special assumption about the behavior of the time series, the statistical properties such as mean and variance of a stationary process do not depend upon time. It is a necessary condition for building a time series model that is useful for future forecasting. Furthermore, the mathematical complexity of the fitted model reduces with this assumption. [4]

There are two types of stationary processes which are defined below:

A process {x(t),t = 0,1, 2,...} is **Strongly Stationary or Strictly Stationary** if the joint probability distribution function of {xt−s , xt−s+1 ,..., xt ,...xt+s−1 , xt+s} is independent of t for all s.[4].

If a time series is strictly stationary, then all the distribution functions for subsets of variables from the process is independent of time, in other words, if the time series properties are not affected by shifting the time, we called it **Strongly stationary.**

A stochastic process is said to be **Weakly Stationary** of order k if the statistical moments of the process up to that order depend only on time differences and not upon the time of occurrences of the data being used to estimate the moments [5, 6, 7]. For example a stochastic process {x (t), t = 0, 1, 2,...} is second order stationary [5, 7] if it has time independent mean and variance and the covariance values $Cov(x_t, x_{t-s})$ depend only on s.[4]

To develop an adequate model for future forecasting the time series is expected to be stationary, but it's not always the case, As stated by Hipel and McLeod [7], the greater the time span of historical observations, the greater is the chance that the time series will exhibit non-stationary characteristics. While in short time span, model the time series using a stationary stochastic process is possible, if the time series is not stationary, differencing and power transformations are often used to make it stationary by removing the trend.

5.4 Autocovariance and Autocorrelation Functions

Plotting a scatter diagram of pairs x_t, x_{t+k} separated by the same interval k, can provide useful information about the nature of the time series, the pairs of adjacent observations x_t, x_{t+k} are

positively correlated if x_t can provides useful information about value that will be observed in the next period.

The covariance between x_t and its value at another time period, say, x_{t+k} is called the autocovariance at lag k, defined by:

$$\gamma k \ = \ Cov(xt\ , xt + k) \ = \ E[(xt \ - \ \mu)(xt + k \ - \ \mu)]$$

The collection of the values of γk, k = 0, 1, 2… is called the autocovariance function, the autocorrelation coefficient at lag k for a stationary time series is

$$\rho k \ = \frac{E[(yt \ - \ \mu)(yt + k \ - \ \mu)]}{\sqrt{E[(yt \ - \ \mu)^2]E[(yt + k \ - \ \mu)^2]}} \ = \ \frac{Cov(yt, yt + k)}{Var(yt)} \ = \ \gamma k \, \gamma 0$$

The collection of the values of ρk, k = 0, 1, 2,… is called the autocorrelation function (ACF).

6 Time Series Forecasting

6.1 What is the meaning of forecasting

A forecast is a prediction of some future event or events. As suggested by Neils Bohr, “making good predictions is not always easy”. [8]

Most forecasting problems requires the use of time series, as time is an important factor in many businesses that utilize daily, weekly, monthly, quarterly, or annual data, and to make a good forecasting accuracy it is highly recommended to select a proper model that fit the time series.

Today the world just wants to know what is going to happen tomorrow, we can see that in all fields such as business and industry, government, economics, medicine, politics, and finance. As suggested by Neils Bohr Forecasting is the prediction of some future event or events using models built using premade knowledge and previews data (time series). Forecasting is so important for future planning and decision-making processes, for areas like the ones suggested in literature [8] such as:

- Operations Management: business organization use forecasting to produce a production schedule of products depending on the demand and the sales, it also controls the inventories and manage the supplies. Forecasting may also be used to define witch mix of products and services to offer depending on the area.

- Marketing: marketing decisions are now based on forecasting, using the results experts can determine the advertising expenses, promotions or change in pricing, to enable businesses to increase their effectiveness.
- Finance and Risk Management: investors are interested in forecasting due to it abilities of assisting in decisions making through the prediction of the interest rate, options, and the risk associated with the investment.
- Economics: governments, financial institutions demand forecasting for major economic variables such as the total of domestic product, population growth, unemployment, consumption.
- Industrial Process Control: each production process has important controllable variables; forecasting can be used to determine when the changed are made or if the process should be turned off and when to move forward with it.
- Demography: demographers forecast births, deaths and migration pattern of populations by regions or country, this study helps governments planning policy and social actions including health care, retirement programs, and antipoverty programs.

6.2 Single-step vs Multi-step forecasting

A forecasting problem that requires the prediction of the next time step is called single step forecasting, while a forecasting problem that requires the prediction of more than one time step is called multi-step [9].

It is important to know the nature of prediction problem dealing with, because the more time steps must be predicted the more challenging the problem will be.

6.3 The forecasting process

A process is a series of connected activities that transform one or more inputs into one or more outputs. [8] These activities are:

1. Problem definition

2. Data collection

3. Data analysis

4. Model selection and fitting

5. Model validation

6. Forecasting model deployment

7. Monitoring forecasting model performance

The figure shows the activities

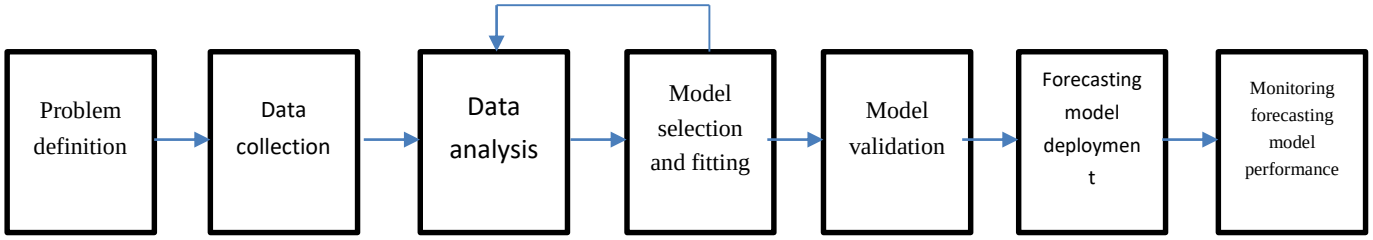

Figure 2.3: forecasting process

6.3.1 Problem definition

It is basically making a good understanding on how the forecasting will be, and reaching the customer expectations. The main question to ask at this phase is how the forecast is going to be, either monthly, year, weekly …… etc. the amount needed for the forecasting interval to be revised. The accuracy of the forecasting it's also a critical feature in this phase because if we make good forecasting, as result we will have good business decisions.

6.3.2Data collection

in this phase our main concern is to gather the relevant data for our forecasting variable(s), after this we need to make sure that data collection is useful for the problem in hand, it is important to deal with some problems that occurs during this phase such as missing values of some variables, handling these problems and planning how to deal with storage issues will preserve the reliability and integrity of the data.

6.3.3Data analysis

A preliminary step to selecting the model, patterns like trends, seasonal or other components should be revealed using time series plots, this preliminary step can provide important information about patterns, these information helps choosing models to work with.

6.3.4 Model selection and fitting

The selection of a model is very important as it reflects the underlying structure of the series and this fitted model in turn is used for future forecasting [4], that is this phase consists of choosing one or more model and fit it to the time series, various methods used to fit a model properly like the least squares, also there exist methods to evaluate the candidates models to choose the best efficient one.

6.3.5 Model validation

Evaluating the fit of model on historical data is not sufficient as the application is intended to new data, usually the fitting errors are smaller than the forecast errors [8], data splitting is a well-known method used for validating a model, it based on dividing the data into two segments, where the first segment called fitting segment used to fit the model, the other segment called forecasting segment used by the model to make the forecasts.

6.3.6 Model deployment

Consists of getting the model and forecasts in use, an important issue may rise during the deployment that is providing data source and making sure that will continue to be available.

6.3.7 Monitoring forecasting model performance

After deploying the model, the conditions change over and may affect the performance of the model, to ensure that the model is always performing well, monitoring of forecasts errors is an essential part of a good forecasting system.

7 Time Series Forecasting models

As we mentioned in the previous section about forecasting process, it’s important to choose the right model that suits the required task, there is a variety of models used and showed a great result, these are the most common ones listed below.

7.1 Time Series Forecasting Using Stochastic Models

Stochastic models can be divided to types linear or non-linear depending on the value of the series whether it is linear or non-linear[4], generally there are two widely used linear time series models, Autoregressive(AR)[…] and Moving Average(MA)[…], furthermore the combination of these two, produced the Autoregressive Moving Average (ARMA) […] and Autoregressive Integrated Moving Average (ARIMA) […], The Autoregressive Fractionally Integrated Moving Average (ARFIMA) [...] model generalizes ARMA and ARIMA models. For seasonal time series forecasting, a variation of ARIMA, viz. the Seasonal Autoregressive Integrated Moving Average (SARIMA) […] model is used.[4]

On the other hand, with many time series reveals some non-linear patterns, non-linear models are widely used in the economy and finance fields, Autoregressive Conditional Heteroskedasticity (ARCH) […] model and its variations like Generalized ARCH (GARCH) […], Exponential Generalized ARCH (EGARCH) [..], are the famous non-linear models.

7.1.1 The Autoregressive Moving Average (ARMA)

A combination of AR (p) and MA (q), suitable for univariate time series modeling, basically future values of a variable in an AR model are a combination of p past observations and a random error together with a constant term [4], mathematically AR (p) expressed as […]:

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-1} + \varepsilon_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \varepsilon_t$$

AR (p) equation

y_t And ε_t are respectively the actual value and random error (or random shock) at time period t, φi (i = 1, 2,..., p) are model parameters and c is a constant. The integer constant p is known as the order of the model.

an MA(q) model is a linear regression of the current observation of the time series against the random shocks of one or more prior observations.[4]

$$y_t = \mu + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

MA (q) equation

Where μ is the mean of the series, θi (i = 1, 2,..., p) are the model parameters and q is the order of the model. The random shocks are assumed to be a white noise

Combining Autoregressive (AR) and moving average (MA) models can form an effective model called ARMA, mathematically ARMA is expressed as:

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

According to Box and Jenkins [4] the necessary and sufficient condition for the AR(p) process to be stationary is that all the roots of the characteristic equation must fall outside the unit circle, While MA(q) is always stationary regardless of values of the parameters. An ARMA (p, q) process is stationary if all the roots of the characteristic equation ϕ (L) = 0 lie outside the unit circle.

7.1.2 Autoregressive Integrated Moving Average (ARIMA)

The feature of ARIMA model is making non-stationary time series to be stationary by applying finite differencing of the data points, ARIMA (p,d,q) models mathematically formulated using the lag polynomials [4]:

$$\varphi(L)(1-L)^d y_t = \theta(L)\varepsilon_t$$

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)(1-L)^d = \left(1 + \sum_{j=1}^{q} \theta_j L^j\right)\epsilon_t$$

The p and q are integers refers to the order of the autoregressive and moving average parts of the model respectively. While the d is the level of differencing.

7.1.3 Seasonal Autoregressive Integrated Moving Average (SARIMA)

While ARIMA model is for non-stationary and non-seasonal time series data, Box and Jenkins [7] proposed a model known as the Seasonal ARIMA (SARIMA) model to handle the seasonality, to remove non-stationarity from the series, a seasonal differencing of appropriate order is used, A first order seasonal difference is the difference between an observation and the corresponding observation from the previous year and is calculated as. $z_t = y_t - y_{t-s}$ For monthly time series s = 12 and for quarterly time series s = 4. This model is generally termed as the s SARIMA (p,d,q) × (P, D,Q) model[4].

7.2 Nonlinear Time Series Models

Linear models discussed above are widely used due to their simplicity, however many time series show nonlinear patterns especially in finance and economy, many nonlinear models proposed to deal with this kind of time series, Some of the famous models are Autoregressive Conditional Heteroskedasticity (ARCH) [10, 11] model and its variations like Generalized ARCH (GARCH) [10,11], Exponential Generalized ARCH (EGARCH) [10] etc..

7.3 Time Series Forecasting Using artificial neural networks

Artificial neural networks techniques have been proposed for time series forecasting and it gained a lot of attention in recent years, the architecture of ANNs is inspired for biological neural networks and it simulate the work of the brain, ANNs learn by experience which means it extract patterns from input data then produce an output based on the previous knowledge it has from the past experiences, in other words ANNs are self-adaptive, which makes it suitable for time series forecasting, many neural networks methods showed a promising results especially deep learning neural networks, in this section we took the most famous neural networks methods applied successfully in time series problems.

7.3.1 MLP for time series forecasting

MLP stands for multi-layer perceptron, a feed forward neural network that has three layers, an input, hidden and an output layer, each layer connected to the adjacent, MLP has some features that can make it powerful for time series forecasting, first of all it has the ability to approximate

nonlinear functions and work with multivariate time series with any number of features given, also it can perform multi-step forecasts, the MLP model can be used to predict univariate as well as multivariate time series data as it has the ability to learn a mapping from inputs to outputs.

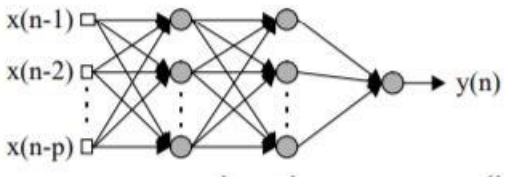

Figure 2.4: MLP architecture for one-step time series prediction

The network input vector shown in figure 1.4 consists of past observations of the time series as follows x = [x (n-1), x (n-2),..., x(n-p)], The output y(n) of the MLP network equals one-step prediction, The actual value x(n) of the series represents the desired output.

MLPs requires an input/output mapping or it will perform poorly taking an example of random walk [14], to achieve this, inputs and outputs must be fixed and the mapping function has to be static, many studies used MLPs for time series forecasting and it gives a satisfied results [15][16].

7.3.2 CNN for time series forecasting

CNN stands for convolutional neural networks, a special type of feedforward neural networks that contains multiple layers, it is widely used in image processing tasks, generally it consist of three layers, the convolutional layer that processes the input, a pooling layer used to perform down sampling along the spatial dimensionality of the given input, finally a fully connected layer that act as an output layer , recently CNNs have been showing great results for time series forecasting taking advantage of its ability to extract features from the input, this ability can be applied to time series, where CNN treats the inputs as 1D sequence that operated by the convolutional layer, then a pooling layer that perform a down sampling to the outputs of the convolutional layer, after that a fully connected layer interprets the features extracted by the convolutional part of the model.

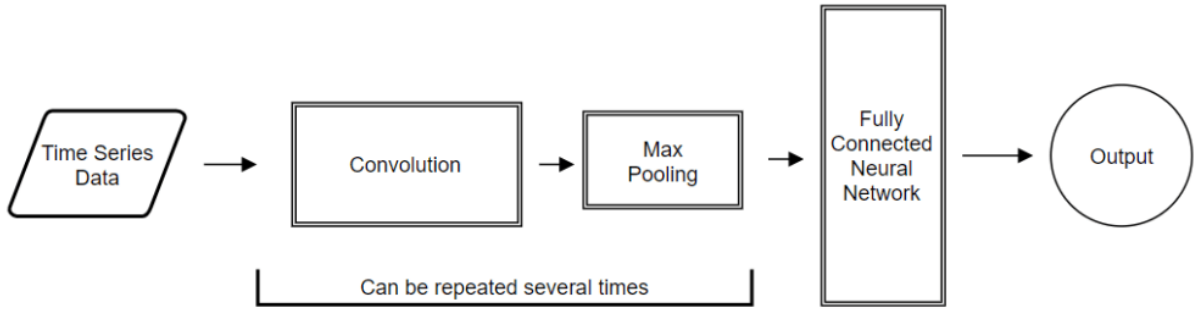

Figure 2.5: CNN architecture for time series forecasting

There are many types of CNN models, each type is used for a specific problem, CNNs can deal with time series, whether it was univariate or multivariate, single-step or multi-step forecasting.

7.3.3 RNN for time series forecasting

RNN stands for recurrent neural networks, it is a type of artificial neural networks, RNNs are powerful in prediction problems as they make use of sequential data, where the previous output fed back to the current inputs, making it act as it has a memory that captures past outputs, although it is limited to memorize just a few steps back, therefor some RNN extensions proposed to solve this problem, the most popular models recently are LSTM and GRU, which are basically an RNN with a feature of memorizing long dependencies, where LSTM has 3 output gates and GRU has 2 output gates, these models are widely used in time series forecasting with many variations showing a good results unlike the traditional RNN.

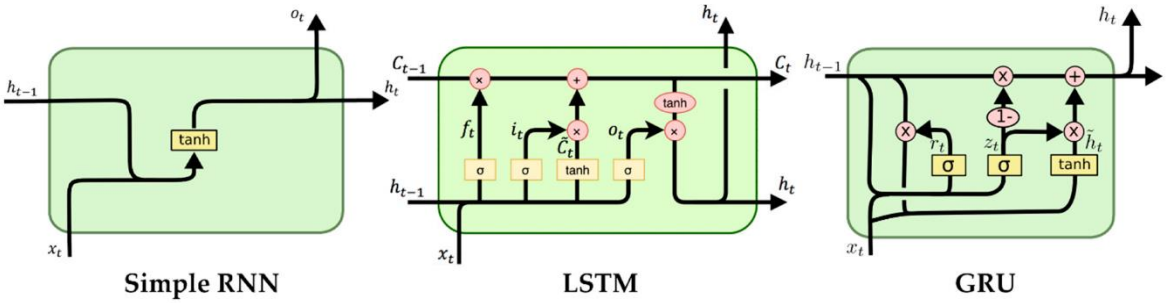

Figure 2.6: simple RNN vs LSTM vs GRU architecture

7.3.4 LSTM for time series forecasting

LSTM stands for long short term memory, a kind of RNN that has been very popular recently, it has basically the same architecture of RNN but it has one difference that make it solve the RNN problem of forgetting long dependencies which is the function of the hidden state that decides what to keep and what to forget, LSTMs are a great deep learning technique for time series forecasting.

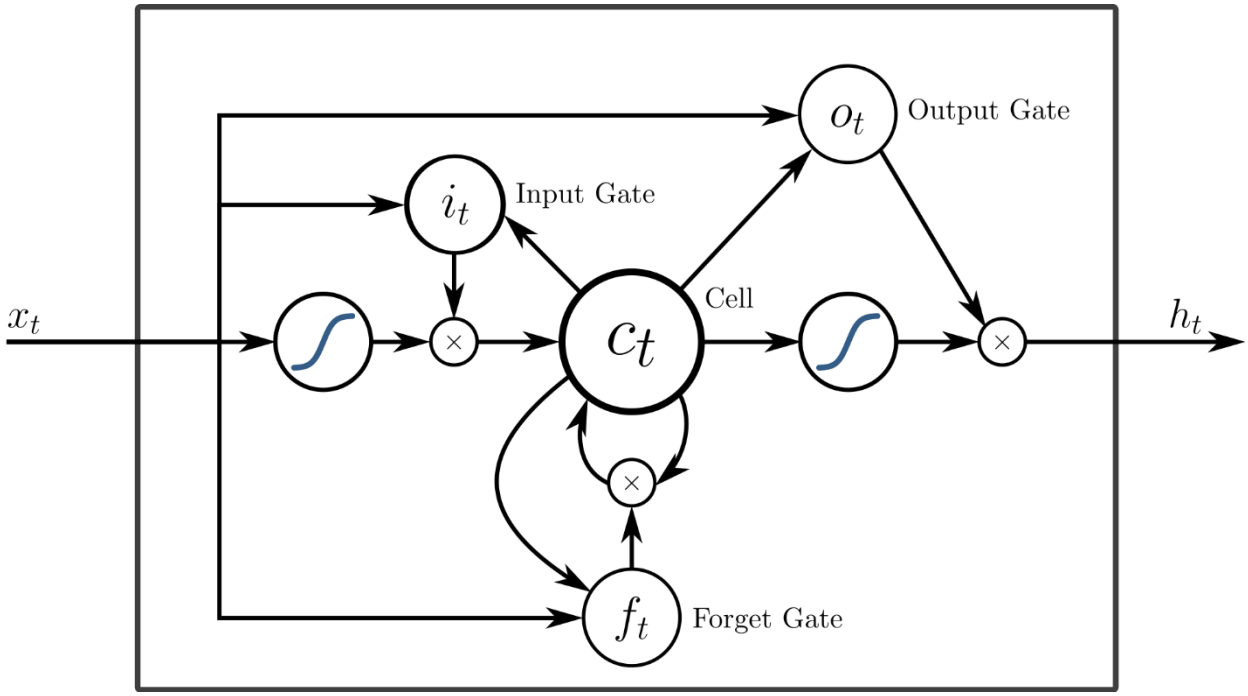

Figure 2.7: LSTM architecture

There are many types of LSTMs that can be used in time series forecasting, each one specialized in a specific problem, we can mention from them:

-Stacked LSTM: an LSTM model that has multiple hidden LSTM layers where each layer contains multiple memory cells, while the traditional LSTM has one single hidden LSTM layer followed by a simple feedforward output layer, the addition of layers adds levels of abstraction of input observations over time.

-Bidirectional LSTM: In problems where all time steps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide

additional context to the network and result in faster and even fuller learning on the problem, this type of LSTM can be applied to time series forecasting and according to some studies it improves slightly the results comparing to the traditional LSTMs.

-CNN LSTM: a hybrid model of combining CNN with LSTM, this type specifically used for sequence prediction problems with spatial inputs, like images or videos, it based on using CNN layers for feature extracting of input data combined with LSTM for prediction, CNN-LSTM developed for visual time series like images.

-Encoder-Decoder LSTM: a special type of LSTM used in sequence to sequence problems also called seq2seq, Sequence-to-sequence prediction problems takes a sequence as input and requires a sequence prediction as output. This model can be used for multi-step time series forecasting. This architecture is comprised of two models: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence. [9]

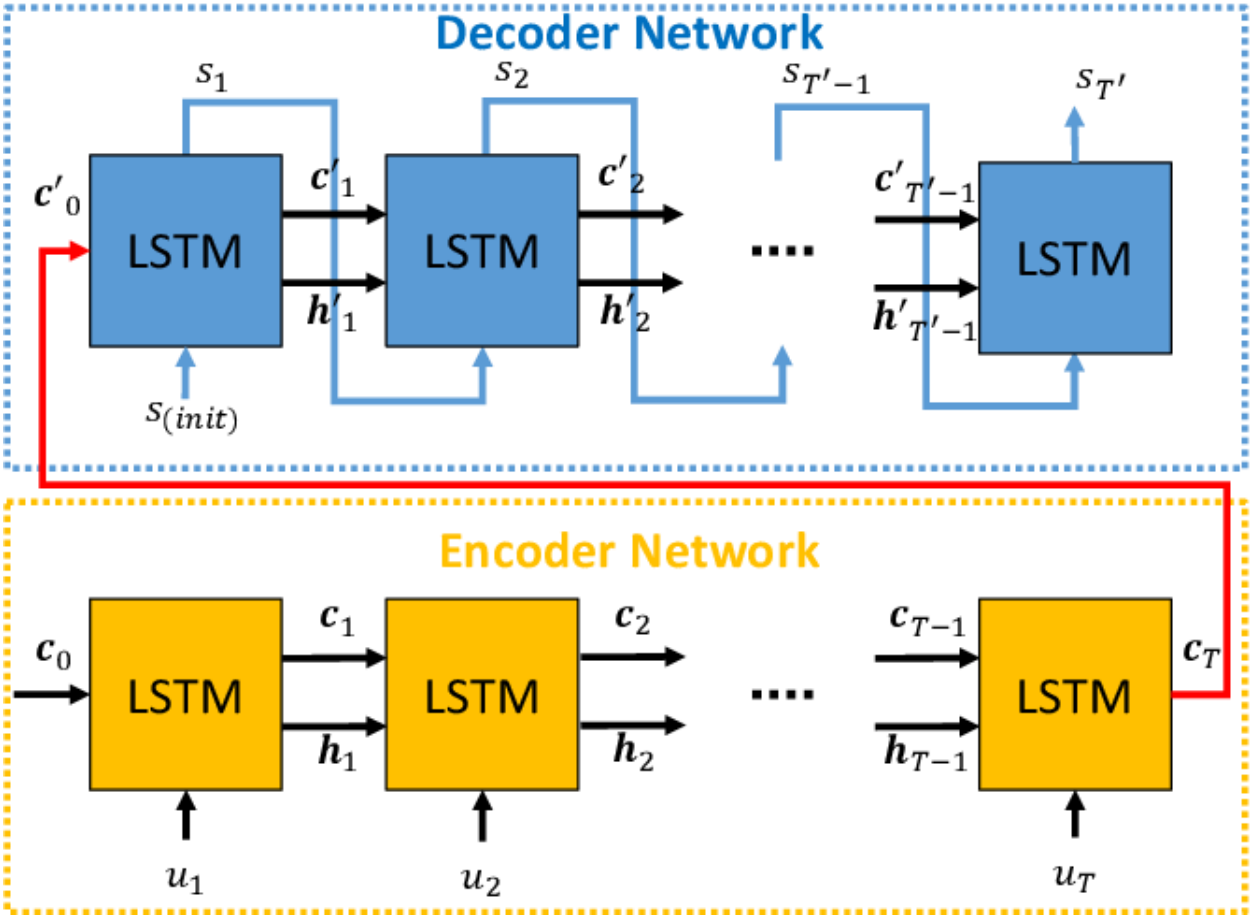

Figure 2.8: The LSTM encoder-decoder architecture. [17]

Each one of these types can be adapted and used according to the time series concerned whether it is univariate or multivariate.

Chapter three:

Failure Prediction in Cloud System

1 Introduction

Cloud computing has become one of the most interesting topics in the world of IT, many software industries migrated to cloud computing platforms to benefit from the offered services for a low cost such as Microsoft Azure, Google Cloud and Amazon AWS, where it offers many service models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These cloud services are used every day by millions of users, which requires a high service availability because a small problem could cost a lot, cloud service providers have a lot of efforts to maintain a high reliability and availability, although, cloud computing services still facing many problem which affect the satisfaction of the user requests, these problems can be caused by a failure in software or hardware components which leads to node and application failures, quite a number of studies attempt to deal with these failure by predicting the failure before it happens, noticing that there is a little number of them focused on node failure.

In this chapter we will discuss cloud computing and cloud failure prediction, and present the main works that dealt with this problem.

2 Cloud computing

According to amazon [18]" Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS)", and to Microsoft [19] "cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale"

A cloud service consists of a large number of computing nodes, in modern cloud computing, virtualization is the used technology to better scalability, maintainability, and reliability. A physical node can host multiple virtual machines, a physical node can host multiple virtual machines (VMs). If a node fails, all VMs hosted on it will also fail.

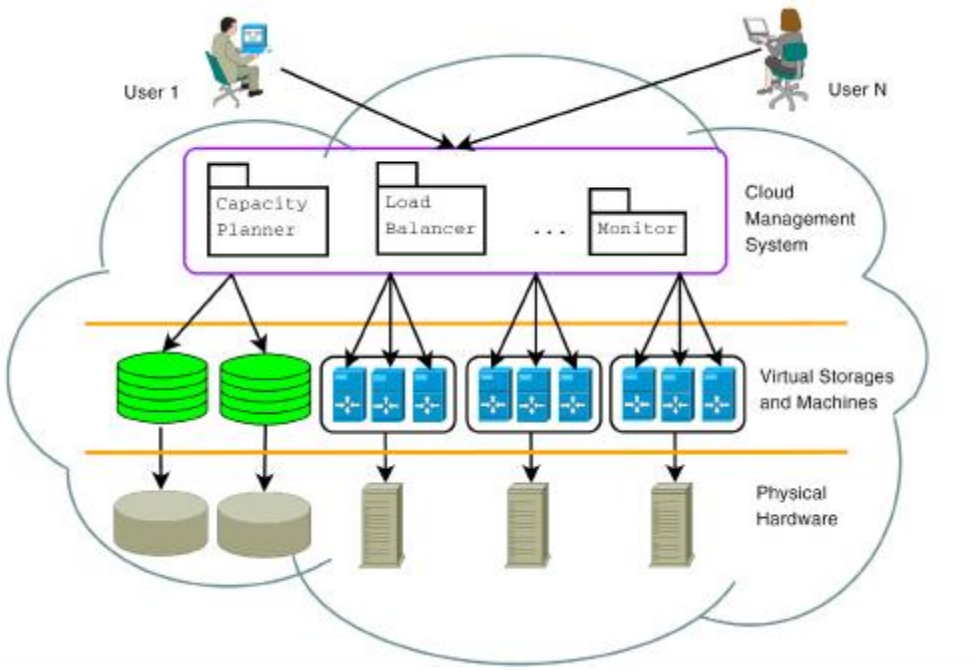

Figure3.1: cloud architecture [20]

3 Failure in cloud service

A failure is an event that affect the service provided to a user, Failure can be caused by many reasons: a node failure for example refers to the unavailability of a physical node for any reason like disk failure, network outage, misconfigurations, memory leak, hardware breakdown, etc.

Failures in cloud could seriously affect users and business and cost a huge amount of money, many famous failures happened to various cloud service providers, for example, in February 2017, AWS experienced a massive outage of its S3 Storage services, causing a majority of websites which relied on AWS S3 unresponsive. On November 18, 2014, the Azure Storage Service was hit by a massive outage as a result of software updates for performance increases. A similar one followed in December, 2015. To give a general view on how much is the cost of these kind of failures, According to ITIC found that one hour of downtime costs 98% of businesses at least $100,000, 86% of businesses $300,000 or higher and 34% of businesses between $1 million - $5million [21].

Efficient and successful failure prediction in cloud service is challenging due to the dynamics of runtime cloud states, heterogeneity of configuration, non-linearity of failure occurrences, and overwhelming volume of performance data in production environments. There are a lot of efforts

to maintain availability of services, several studies proposed to predict the failure before it's happened based on historical trace using different approaches (mathematical, probability, deep learning etc.).

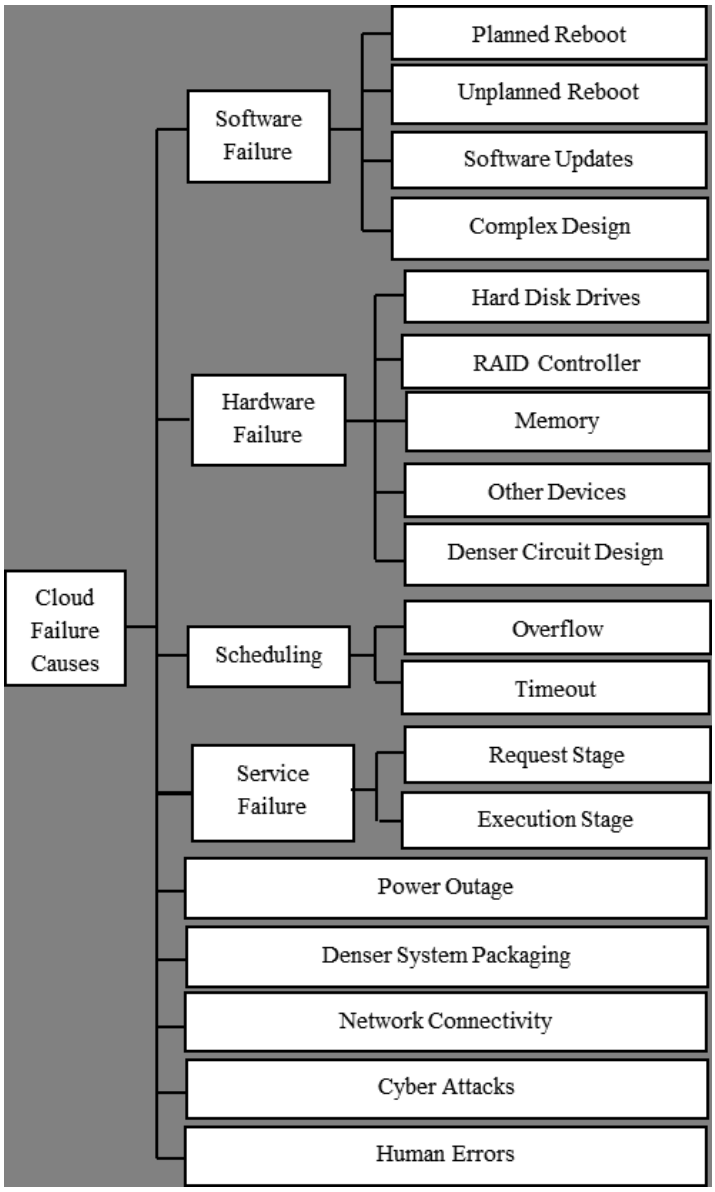

Figure3.2: failure causes in cloud environment

4 Failure prediction survey

In the following we will discuss the main works done to deal with this problem, we present their approach and the specific failure type handled.

4.1 Decision tree and Forests

Decision trees use a tree data structure to store classification rules. Based on the analysis of a set of instances, interior nodes are generated, which contain decision rules, and exterior nodes (leaves), which include the predicted outcome. Starting with the root node, each node directs to one of its subtrees, depending on the outcome of the rule/test that was mapped to the specific node. Once a leaf is reached, the result mapped to that leaf is returned as a result, Random forest is just an improvement over the top of the decision tree algorithm. The core idea behind Random Forest is to generate multiple small decision trees from random subsets of the data.

Nakka et al. [22] build a decision tree to predict node failure up to one hour in advance. They use the dataset from Los Alamos National Laboratory (LANL), The raw failure and usage data from a system is used to generate instances (records) containing node usage and failure information at a particular time instant. Every record contains both past and future information relative to the current instant of time. The authors did some cleaning, curating and merging logs, then applied some machine learning techniques like random trees, random forests and REPTree, The Random Forest approach gives the best result when including both the root cause of the failure as well as usage and failure information.

Chalermarrewong et al. [23] use the Autoregressive-Moving-Average Model (ARMA) and the Fault Tree analysis to predict the possible system failures in data centers. starting by building and training ARMA model from a set of data collected from each computing node, to increase the accuracy, the authors designed a self-adjusted ARMA where the model will re-train itself, The ARMA predictions are then used to conduct a Fault Tree Analysis, the authors define a threshold, values for the outputs of the ARMA model, and according to these thresholds they assign 0 or 1 to the leaves of the tree. Then these Boolean values fed towards the root node, where, a binary prediction is calculated. For evaluation, the authors use the simulator system Simics to simulate eight virtual machines, where their model yields a precision of 100% and a recall of 93%.

Guan et al. [24] propose two learning approaches that use Bayesian methods and decision trees to predict failure dynamics in cloud computing systems. In order to build the decision tree, the algorithm define a gain function $G(xi, n) = H(n) - H(xi, n)$, where $H(n)$ is the entropy of node n; and $H(xi, n)$ is the sum of entropies of all child nodes of node n when splitting based on attribute

xi, since the model may face an overfitting, the authors set a pruning dataset, unused in training, for the test, the environment consists of 11 Linux server clusters on campus. Two clusters contain 116 servers each, while others host 32, 16, 22, 20, 10 and 8 servers in them. In total, there are 362 high-performance servers in the cloud. No quantitative statement for precision and recall are given.

Sîrbu and Babaoglu [25] use one month of Google job failure data to predict node failures within 24 hours, starting by collecting features from multiple tables of google workload trace, the data points were separated into two classes: safe (negatives) and fail (positives). To face the fact that the safe class is much larger than the fail class and classifiers have difficulty learning patterns from very imbalanced datasets. The authors used a random subsampling approach, Sîrbu and Babaoglu claim that the method is suitable for online use, Results could be improved by changing the subsampling procedure.

4.2 classification

Classification is design to classifying observation into a specific pre known category with mutual specification. Classification require predetermined categories.

There are many works that uses classification to try making the most effective solution to this problem, and we are going to highlight few of them:

Zhu et al [26]:

They used a backpropagation neural network model based on SMART attributes, and also developed an SVM model to make even a better prediction accuracy. SVM support vector machine is a supervised machine learning method for classification, it takes two set of training samples and will find the best decision hyperplane that separates the two classes. Artificial neural networks they take as input a vector and convert it into another vector of variables, in this case, the chosen method to train it is Backpropagation algorithm. The SMART stand for self-monitoring analysis and reporting technology, a SMART dataset was collected from a single running data center of Baidu Inc, the dataset get through feature selection to get rid of useless attributes that will not change during operation and Feature normalization so we can get fair comparison between different feature values. The SVM model gets a detection rate of 80% with

0.3% FAR stand for failure detection rate. The BP network model achieve higher failure detection rate the SVM model, and slightly higher FAR.

Pelaez et al [27]:

Their approach is much different from the solution that already exist, the common idea is to analyze traces of system events to find correlations between anomalies and node failure, and use patterns to identified failure at run time, but their approach is way different , they have used decentralized online clustering algorithm DOC to detect anomalies in resource usage logs. Their solution has 3 main layers firs one is DOC, and DOC is a clustering algorithm designed to run in a distributed setting in data that is also distributed. Clustering algorithm use cluster detection and it is based on evaluating the relative density of points within the information space .In DOC the information space is subdivided dynamically into regions, and each region is assigned to a particular processing node .so basically the DOC is responsible for handling how the points get distributed to the nodes, and implements the clustering algorithm. Built in the top of the DOC is anomaly detection layer, which take as inputs the cluster generated from the DOC layer and use it to extract the anomalies from them, and also keep track the logs and extract the relevant features from them, and invoke DOC. the top layer is in charge of making predictions based on the anomalies extracted from the below layer and enhance our precision accuracy , using two techniques, multiple time bins and multiple clustering , using the first one we found out that using 5 to 10 minutes can increase the precision, and using multiple clustering just we make sure of not leaving any important information behind , changing every time feature sets. The test was made on several thousand nodes harvest acceptable results of approximately 2% CPU utilization.

LU et al [28]:

Their proposition was like follow, focusing on failure feature extraction without manually filtering or compressing. They have made an online prediction by using supervised locally linear embedding algorithm to manage the affected matrices. An autonomic failure prediction framework is shaped of three function, performance monitoring, feature extraction and failure prediction. The first step is collecting data of different performance metrics under different condition, faulty and non-faulty. After it feature extraction to subtract dimensionality and gain features, and in this case in order to do that we preform SLLE, its main principle is to keep local neighborhood relation of data in both the embedding space and the intrinsic one, it only have two

parameters to be set and what make it easy to use it for failure prediction. In order to achieve good SLLE mapping there are three SLLE parameters need to be set carefully, the intrinsic dimensionality m, the number of neighbor k and generalization parameter α. The experiment was implemented in collection of 874 metrics from 200 nodes plus the router's metrics gutting us a total of 1274 metrics. The final results were SLLE method can automatically predict more than 50% of the central processing unit (CPU) and memory failures, and around 70% of the network failure getting better result better then both KNN and SVM.

4.3 Neural networks

Neural networks have become a famous topic in recent years, due to its important features which is, Neural networks are data-driven self-adaptive methods, universal functional approximators in that they can approximate any function with arbitrary accuracy also they are a non-linear models, in this section we will discuss the main work on failure prediction in cloud using neural networks models.

Chen et al [29]:

The authors Chen et al use a neural network model to predict job failure from google cluster dataset, they used specifically the RNN model and the prediction is based on resource usage measurements from the Google cluster traces, the authors applied recurrent neural networks to the resource usage measures, and generate features to categorize the input resource usage time series into different classes. They found that the proposed model could save 6% to 10% of resources. The authors started by analyzing the dataset, preprocessing it and extract the important features, the next step is defining the model then train it and test it with time series produced from google dataset.

Zhu et al [30]:

In this paper, Zhu et al, build a drive failure prediction model based on Backpropagation (BP) neural network in comparison with an improved Support Vector Machine (SVM), the authors test a three layer neural network with a sigmoid activation function on SMART dataset, a hard disk failure data from 20,000 drives. The BP learning algorithm has two phases: a feed-forward stage and a back-propagation stage. The model trained and tested on SMART dataset after the

preprocessing, feature selection and normalization, the proposed neural network achieved a detection rate more than 95%, also comes with a higher false alarm rate of 0.48% to 2.26%.

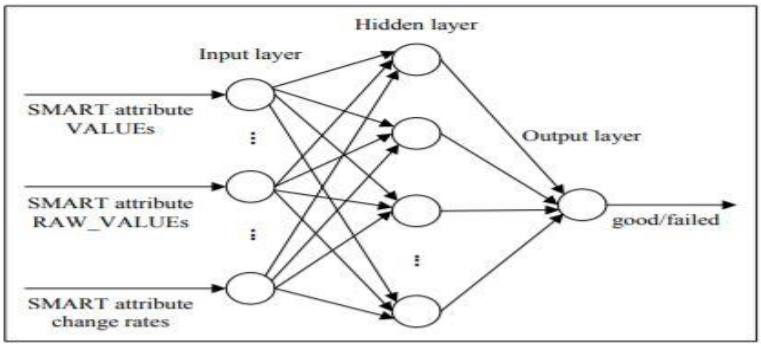

Figure3.3: the neural network model

Islam et al [31]:

The authors Islam et al, present a long short term memory (LSTM) neural network in order to predict job failures using the Google dataset. LSTM is a special type of RNN especially suitable for long-term temporal dependencies. Islam et al fed the raw resource measures and task attributes to the network after performing a failure characterization study on the Google cluster workload trace to understand the key characteristics of task/job failures in a production cloud. this analysis aims to extract the relevant features, the proposed failure prediction model consists of five stages: (1) monitoring and storing the system and application metrics, (2) processing data to structured formats containing their spatial and temporal information, (3) extracting relevant features from data, (4) predicting failures using machine learning model, and (5) failure remediation management based on the predicted results.

The model achieves around 87% of accuracy, 85% of sensitivity and 89% of specificity at task level, in the job level the model achieves an accuracy of 81%, true positive rate of 83%. However, false positive rate is 20%, which is a bit higher compared to the task level results. The authors state that up to 10% of CPU time and other resources can be saved using their predictive approach.

Lin et al [32]:

In recent study, Lin et al, Microsoft engineers, proposed MING a novel model to predict node failure in cloud environment, this model combines: 1) a LSTM model to incorporate the

temporal data, 2) a Random Forest model to incorporate spatial data; 3) a ranking model that embeds the intermediate results of the two models as feature inputs and ranks the nodes by their failure-proneness, 4) a cost-sensitive function to identify the optimal threshold for selecting the faulty nodes. During building the model, the authors identifies three difficulties, **complicated failure causes:** Due to the complexity of the large-scale cloud system, node failures could be caused by many different software or hardware issues. **Complex failure-indicating signals:** Failures of a node could be indicated by many temporal signals produced by the node locally. And could also be reflected by spatial properties that are shared by nodes that have explicit/implicit dependency among them in different global views of the cloud. **Highly imbalanced data:** Node failure data is highly imbalanced as most of the time the cloud service system has high service availability.

The proposed approach MING includes two phases of training, **phase 1:** two base classification models are trained: a LSTM model for temporal data and a Random Forest model for spatial data. **Phase 2:** the intermediate results of the two base learners are embedded as features and fed as input to a ranking model. The ranking model ranks the nodes by their failure-proneness. MING identifies the top r ones that minimize the misclassification cost as the predicted faulty nodes. The below figure describes this model.

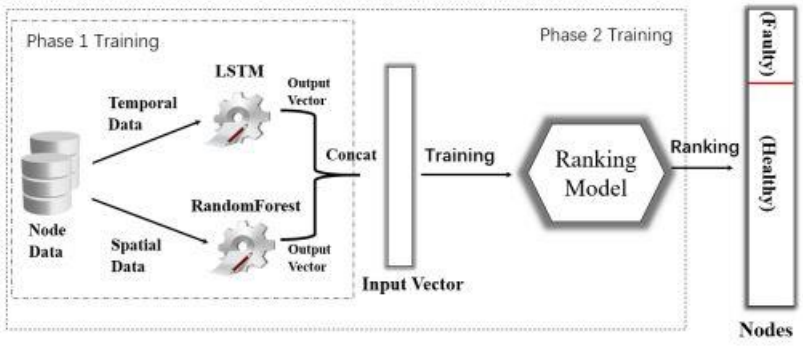

Figure 3.4: The overview of MING

In phase 1, LSTM applied for temporal features, the authors used bi-directional LSTM, for spatial features, they applied the Random Forest learner, It builds a multitude of decision trees at training time and outputs the class of the voting result from the individual trees. Random Forest splits the trees based on the information gain, therefore it can better reflect the impact of discrete values.

In phase 2, the intermediate output vectors produced by the previous learners becomes the input vector to the ranking model. Then concatenate the two output vectors and form a 256 x 1 input vector V for the ranking model. To evaluate the proposed approach, they collect data from a production cloud service system. The data are from part of the data centers, containing over half a million of physical cloud computing nodes. MING achieves 92.4% of precision, 63.5% of recall, according to these results MING outperforms the traditional approaches.

5 Conclusion

In this chapter, we have discussed cloud failure, causes and consequences and failure prediction in cloud, and presented the main approaches (correlation probability, classification, decision tree and forests, and neural networks) and the works done in every approach in detail.

Chapter four:

Implementation and Experimental Results

1 Introduction

In this chapter, we will talk about the proposed model, the implementation on google cluster dataset and evaluate the results, at the beginning we will describe and introduce the difficulties faced with the dataset used to train the model which is the google cluster dataset, then we will explain the steps followed to preprocess this dataset, after that we will show the model built to train the dataset, in the end we will discuss the results found.

2 Google Cluster dataset

2.1 Overview

The dataset released by google in 2011 contains data collected from 12500 machine[33], the trace includes a describe of jobs submitted by users, machines description, each job consists of one or more tasks, each of which came with resource requirements and resource usage, each task represents a Linux program that run on a single machine, each task has records of resource usage(CPU, memory, disk I/O…), and a termination status(event type), which allows us to analyze and build a model to predict machine failures using the resource consumption of each task, the figure below illustrates the state(event type) transition for jobs and tasks:

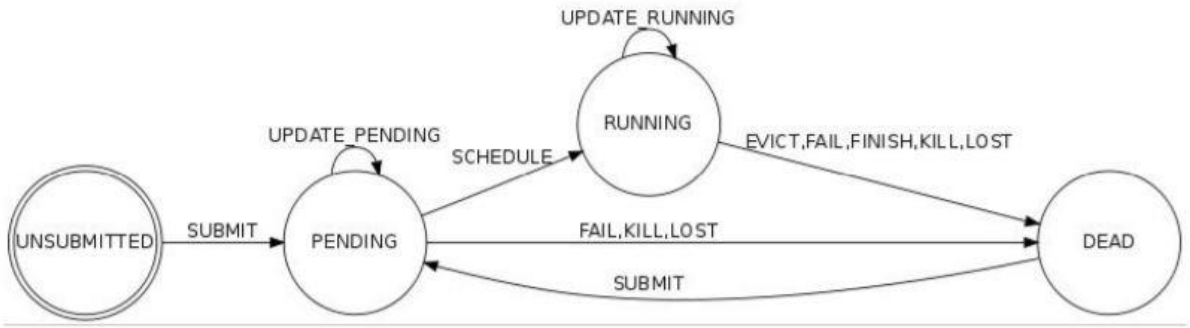

Figure4.1: state transition for tasks and jobs [33]

Each job and task event has a value representing the type of event. The state of the job or task after the event can always be determined from this event type.

The following are the event type codes:

● SUBMIT (0): A task or job became eligible for scheduling.

● SCHEDULE (1): A job or task was scheduled on a machine. (It may not start running immediately due to code-shipping time, etc.) For jobs, this occurs the first time any task of the job is scheduled on a machine.

● EVICT(2): A task or job was descheduled because of a higher priority task or job, because the scheduler overcommitted and the actual demand exceeded the machine capacity, because the machine on which it was running became unusable (e.g. taken offline for repairs), or because a disk holding the task's data was lost.

● FAIL (3): A task or job was descheduled (or, in rare cases, ceased to be eligible for scheduling while it was pending) due to a task failure.

● FINISH (4): A task or job completed normally.

● KILL (5): A task or job was cancelled by the user or a driver program or because another job or task on which this job was dependent died.

● LOST (6): A task or job was presumably terminated, but a record indicating its termination was missing from our source data.

● UPDATE_PENDING (7): A task or job's scheduling class, resource requirements, or constraints were updated while it was waiting to be scheduled.

● UPDATE_RUNNING (8): A task or job's scheduling class, resource requirements, or constraints were updated while it was scheduled.

In our work, we are focusing only on event types of fail (3) and finish (4) to predict the failed tasks to prevent from future failures that could be caused by the unhealthy allocated machines due to many reasons, we observed that the tasks has a strong correlation with resource usage, where higher resource usage tasks are the most likely to fail.

2.2 Dataset challenges

Cloud computing offers many services, google provide these services via google cloud, during the analysis of the google cluster dataset, we found some challenging points, the first one is that the data are separated into many data tables which requires a data analysis work, the second challenge we faced is the size of the data(The total size of the compressed trace is

approximately 41GB), to handle all this amount of data it needs a cloud service to do the work in it, the last thing is that the time series data is irregular which make it challenging to the model to train.

3 Experimental environment

In order to implement our model, we used many tools and frameworks we will define it briefly, and the we did the experiment on a i7-6700HQ pc with 16GB of RAM and GEFORCE GTX 1060 6GB graphic card:

Jupyter notebook:

The Jupyter Notebook [34] is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

Python:

Python [35] is an object oriented high level programming language, it is a multi-purpose programming language used in many fields like artificial intelligence, machine learning, data science and automation…, in recent years it becomes the most used programming language due to its ease of use and it has a large ecosystem (frameworks, libraries and tools).

TensorFlow:

TensorFlow [36] is an open source software library for numerical computation using data-flow graphs. It was originally developed by the Google Brain Team within Google's Machine Intelligence research organization for machine learning and deep neural networks research

Keras:

Keras [37] is a high-level neural network API, run on top of TensorFlow and other machine learning libraries, it was developed to make implementing deep learning models as fast and easy as possible and with less code for research and development.

Pandas:

Pandas [38] is an open source Python library designed for data manipulation and analysis, it offers data structures and operations for manipulating numerical tables and time series which make it easy to deal with data in python.

NumPy:

NumPy [39] is a scientific package for computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), it offers functions for fast manipulation of objects, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Scikit-learn:

Scikit-learn [40] is a library in Python that provides many unsupervised and supervised learning algorithms. The library is built upon the SciPy (Scientific Python)

Matplotlib:

Matplotlib [41] is a python library for 2D plotting, used to produce high quality figures, it can be used in all Python environments of developing, these figures can be saved in different format to make their use easy for scientific researches.

Seaborn:

Seaborn [42] is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. And it is closely integrated with pandas data structures.

4 Implementing BiLSTM deep learning model on google dataset

In this section we provide the project source code and the explanation of each step we did in the process after preparing the environment by installing jupyter notebook and TensorFlow, keras and all the required tools.

4.1 installing the packages and frameworks

In this cell we imported all the necessary frameworks and packages we will work with.

```
Entrée [1]: import numpy as np
            import tensorflow as tf
            from tensorflow import keras
            import numpy as np
            import matplotlib.pyplot as plt
            import pandas as pd
            import glob
            import seaborn as sns
            pd.set_option('display.max_columns', None)
            import os
            from keras.models import Sequential
            from keras.layers import Dense, Dropout, LSTM, Activation
            from sklearn import preprocessing
            from sklearn.metrics import confusion_matrix, recall_score, precision_score
            from keras import utils
            from keras.utils import to_categorical

            Using TensorFlow backend.
```

Figure4.2: import packages and frameworks

4.2 data preprocessing

In this phase we load the data into the work environment, then apply some data preprocessing techniques to get a formulated data before building the model, in our case we will use the following rules to get the desired structured data:

-joining and merging data tables

-data cleaning and remove missing values

-feature extraction

-split data into train and test

-data normalization

-create sequences for input

4.2.1 Load the data tables

The original data is divided into multiple tables, we load the task tables (task usage, task events) into pandas dataframes.

```
Entrée [5]: #Load the files
            df=pd.concat( [pd.read_csv(f, compression="gzip", header=None)
                           for f in glob.glob(r'D:/datasets/google cluster data/test1/task usage/part-00*-of-00500.csv.gz')])
            df.columns=['time', 'end_time', 'job_ID', 'task_index' , 'machine_ID', 'CPU_rate', 'canonical_memory_usage',
                        'assigned_memory_usage', 'unmapped_page_cache','total_page_cache','max_memory_usage','disk_I/O_time',
                        'local_disk_space_usage','max_CPU_rate','max_disk_IO_time','cycles/instruction','memory_accesses/instruction',
                        'sample_portion','aggregation_type','sampled_CPU_usage']
            df.shape

   Out[5]: (3922515, 20)
```

Figure4.3: load task usage data

```
Entrée [12]:
te=pd.concat( [pd.read_csv(f, compression="gzip", header=None,dtype={ "machine_ID": 'int64'})
               for f in glob.glob(r'D:/datasets/google cluster data/test1/task events/part-00*-of-00500.csv.gz')])
te.columns = ['time','missing info','job_ID','task_index','machine_ID','event_type','user','scheduling class',
              'priority','CPU request','memory request','disk space request','different machines restriction']

te.shape

Out[12]: (1519276, 13)
```

Figure4.4: load task events data

4.2.2 Joining and merging data

In order to get the full measures usage and the termination status of each task we need to join and merge task usage and task events dataframes using the **merge** and **groupby** functions.

```
Entrée [20]:
c=pd.merge(te,df,on = ['job_ID','task_index','machine_ID'])
c.head()
```

Out[20]:

	time_x	job_ID	task_index	machine_ID	event_type	user	scheduling class	priority	CPU request	memo reque
0	1970-01-01 00:00:00.000110846	6221861800	73	3605497855	1	huEqgBuTsgIbw1KmYjlUw/N6rUCEeIqRUY51E4M8icU=	0	0	0.0125	0.01
1	1970-01-01 00:00:00.000110846	6221861800	73	3605497855	1	huEqgBuTsgIbw1KmYjlUw/N6rUCEeIqRUY51E4M8icU=	0	0	0.0125	0.01
2	1970-01-01 00:00:00.000110846	6221861800	73	3605497855	1	huEqgBuTsgIbw1KmYjlUw/N6rUCEeIqRUY51E4M8icU=	0	0	0.0125	0.01
3	1970-01-01 00:00:00.000110900	6221861800	73	3605497855	3	huEqgBuTsgIbw1KmYjlUw/N6rUCEeIqRUY51E4M8icU=	0	0	0.0125	0.01
4	1970-01-01 00:00:00.000110900	6221861800	73	3605497855	3	huEqgBuTsgIbw1KmYjlUw/N6rUCEeIqRUY51E4M8icU=	0	0	0.0125	0.01

Figure4.5: merge dataframes

```
Entrée [22]:
c = c.groupby(['time_x','job_ID', 'task_index','machine_ID','event_type'], as_index=False).agg('mean')
```

Figure4.6: groupby function

4.2.3 Data cleaning and remove missing values

In our work we consider two termination status (finish, fail) and the dataset has 8 different termination status, we will drop all the rows that has termination status 0 or 1, and collect the other termination status as failed to get rid of the highly imbalanced time series and to better train the model, next we will drop the missing values (nan),

```
Entrée [24]: c1 = c.query('event_type !=0 & event_type!=1')
             c1
```

Out[24]:

	time_x	job_ID	task_index	machine_ID	event_type	scheduling class	priority	CPU request	memory request	disk space request	different machines restriction	CPU_rate	canonica
0	1970-01-01 00:00:00.000110846	6221861800	54	6562993	3	0	0	0.01250	0.015900	0.000404	0	0.000000	
7	1970-01-01 00:00:00.000110846	6221861800	185	4820075950	3	0	0	0.01250	0.015900	0.000404	0	0.000000	
15	1970-01-01 00:00:00.000110846	6221861800	448	3349108012	3	0	0	0.01250	0.015900	0.000404	0	0.000000	
16	1970-01-01 00:00:00.000110846	6221861800	581	638356	3	0	0	0.01250	0.015900	0.000404	0	0.000000	
17	1970-01-01 00:00:00.000110846	6221861800	585	410753191	3	0	0	0.01250	0.015900	0.000404	0	0.000000	
...	...	...	...	...	...	...	...	...	...	...	...	...	
988877	1970-01-01 00:00:00.000115857	6264852738	221	988439	2	0	0	0.03125	0.003109	0.000386	0	0.032996	
988878	1970-01-01 00:00:00.000115857	6264858031	302	257337203	2	0	0	0.06250	0.031800	0.000771	0	0.000453	
988879	1970-01-01 00:00:00.000115857	6264881627	4	336021530	4	0	4	0.06250	0.015900	0.000386	0	0.003243	
988880	1970-01-01 00:00:00.000115857	6264881627	65	590641088	4	0	4	0.06250	0.015900	0.000386	0	0.002421	
988881	1970-01-01 00:00:00.000115857	6264881627	96	4820096633	4	0	4	0.06250	0.015900	0.000386	0	0.002659	

502875 rows × 25 columns

Figure4.7: remove rows with termination status 0 or 1

```
Entrée [26]: c1.event_type[c1.event_type == 3] = 1
             c1.event_type[c1.event_type == 4] = 0
             c1.event_type[c1.event_type == 5] = 1
             c1.event_type[c1.event_type == 2] = 1
             c1.event_type[c1.event_type == 8] = 1
```

Figure4.8: collect rows with different termination status as failed

```
Entrée [30]: c1.isna().sum()
```

Out[30]:

```
time_x                          0
machine_ID                      0
event_type                      0
scheduling class                0
priority                        0
CPU_rate                        0
canonical_memory_usage          0
assigned_memory_usage           0
unmapped_page_cache             0
total_page_cache                0
max_memory_usage                0
local_disk_space_usage          0
max_CPU_rate                    0
max_disk_IO_time            12912
cycles/instruction         277982
memory_accesses/instruction    295747
sampled_CPU_usage               0
dtype: int64
```

```
Entrée [31]: c1 = c1.dropna()
```

Figure4.9: remove missing values

4.2.4 Feature extraction

to construct an effective failure prediction model, we collect data for each machine and task and identify features from the data, we choose task features that is highly correlated with the termination status, thus we can construct a reliable prediction model, table () show the features we selected to build the model.

Feature	Description
scheduling class	represents how latency-sensitive task is
priority	Task priority (the importance)
CPU rate	mean CPU usage rate
canonical_memory_usage	Canonical memory usage measurement
assigned_memory_usage	memory usage based on the memory actually assigned to the container
unmapped_page_cache	Linux page cache (file-backed memory) not mapped into any userspace process
total_page_cache	total Linux page cache (file-backed memory)
max_memory_usage	the maximum value of the canonical memory usage
disk_I/O_time	Disk time usage measured by blkio subsystem for Linux containers
local_disk_space_usage	runtime local disk capacity usage
max_CPU_rate	Max CPU usage rate
max_disk_I/O_time	Max disk i/o time measurement
cycles/instruction	processor performance counter
memory_accesses/instruction	processor performance counter
sampled_CPU_usage	The average CPU usage during a one-second period

Table4.1: features description

Using the correlation plot provided by seaborn framework, we can see the correlation between the features:

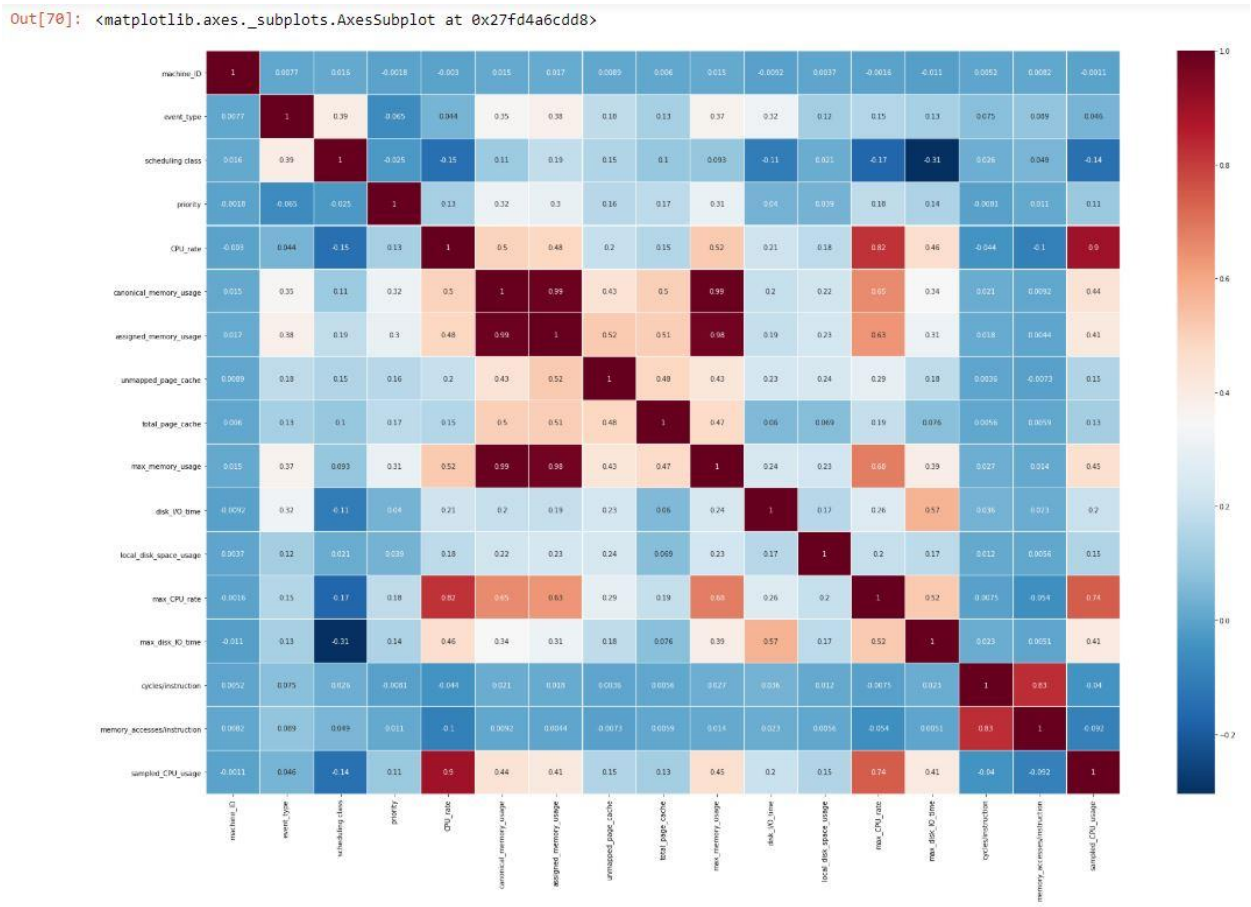

Figure4.10: correlation table

4.2.5 Data split train-test

In this phase we split the data into train and test then, we used 70% of data for training and 30% for testing.

```
Entrée [71]: train_size = int(len(c1)*0.7)
             test_size = len(c1) - train_size
             train , test = c1[0:train_size], c1[train_size:len(c1)]
             print(train.shape, test.shape)

             (93259, 17) (39969, 17)
```

Figure4.11: train test split

4.2.6 Data normalization

To get an effective results from the model, it is important to normalize the data into values that range between 0 and 1, we used a MinMaxscaler from scikitlearn to do that.

```
Entrée [73]: # MinMax normalization
cols_normalize = train.columns.difference(['machine_ID','event_type','priority','scheduling class'])
min_max_scaler = preprocessing.MinMaxScaler()
norm_train_df = pd.DataFrame(min_max_scaler.fit_transform(train[cols_normalize]),
                             columns=cols_normalize,
                             index=train.index)
join_df = train[train.columns.difference(cols_normalize)].join(norm_train_df)
train = join_df.reindex(columns = train.columns)
train.head()
```

Figure4.12: train data normalization

```
Entrée [74]: norm_test_df = pd.DataFrame(min_max_scaler.transform(test[cols_normalize]),
                            columns=cols_normalize,
                            index=test.index)
test_join_df = test[test.columns.difference(cols_normalize)].join(norm_test_df)
test = test_join_df.reindex(columns = test.columns)
test = test.reset_index(drop=True)
test.head()
```

Out[74]:

	machine_ID	event_type	scheduling class	priority	CPU_rate	canonical_memory_usage	assigned_memory_usage	unmapped_page_cache	total_page_cache	max_
0	5494745986	1	0	4	0.008627	0.001438	0.002070	0.002439	0.000826	
1	367886057	0	0	4	0.001802	0.001120	0.002070	0.002235	0.000755	
2	1429192957	1	0	4	0.008335	0.001343	0.002060	0.002178	0.000766	
3	3437368011	0	0	4	0.002446	0.001086	0.002078	0.002434	0.000785	
4	294771759	1	0	4	0.014572	0.001444	0.002060	0.002291	0.000805	

Figure4.13: test data normalization

4.2.7 Split data to X_train, y_train, X_test, y_test

After normalizing the data we split the train and test data in order to get features-target pair, to do that, we created a custom function, we set the time step to 100 which is sufficient for backward observations.

```
Entrée [90]: def create_dataset(X, y ,time_steps=1):
                 Xs, ys = [],[]
                 for i in range(len(X)- time_steps):
                     v = X.iloc[i: (i + time_steps)].to_numpy()
                     Xs.append(v)
                     ys.append(y.iloc[i + time_steps])
                 return np.array(Xs) , np.array(ys)

Entrée [91]: TIME_STEPS = 100
X_train, y_train = create_dataset(train, train.event_type, time_steps= TIME_STEPS)
X_test, y_test = create_dataset(test, test.event_type, time_steps= TIME_STEPS)
```

Figure4.14: create_dataset function

4.3 build the deep neural network model

4.3.1 Model configuration and building

In order to build and efficient model and avoid the overfitting and gradient descent problems, it is important to have the best configurations(hyperparameters), we have built

different models with different hyperparameters (#neurons, activation function, #number of layers, optimizers...), We made a summary of the last model that we developed, see **figure .**

```
Entrée [89]: model.summary()

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_2 (Bidirection (None, 100, 200)          90400
_________________________________________________________________
dropout_3 (Dropout)          (None, 100, 200)          0
_________________________________________________________________
lstm_4 (LSTM)                (None, 50)                50200
_________________________________________________________________
dropout_4 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 51
=================================================================
Total params: 140,651
Trainable params: 140,651
Non-trainable params: 0
_________________________________________________________________
```

Figure4.15: our BiLSTM model summary

```
Entrée [92]: # build the network


model = Sequential()

model.add(Bidirectional(LSTM(
        input_shape=(X_train.shape[1], X_train.shape[2]),
        units=100,
        return_sequences=True)))
model.add(Dropout(0.2))

model.add(LSTM(
         units=50,
         return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(units=1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure4.16: our BiLSTM model building

4.3.2 Model training

We trained the model using our prepared data, we use binary-cross-entropy loss function because we have a binary classification problem (fail, finish) rather than categorical classification problem (categorical-cross-entropy).

```
Entrée [94]: %%time
# fit the network
model.fit(X_train, y_train, epochs=10, batch_size=200, validation_split=0.05, verbose=1,
          callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=0, verbose=0, mode='auto')])

Train on 88501 samples, validate on 4658 samples
Epoch 1/10
88501/88501 [==============================] - 117s 1ms/step - loss: 0.1770 - accuracy: 0.9437 - val_loss: 0.5194 - val_accurac
y: 0.9725
Epoch 2/10
88501/88501 [==============================] - 115s 1ms/step - loss: 0.1454 - accuracy: 0.9564 - val_loss: 0.3194 - val_accurac
y: 0.9742
Epoch 3/10
88501/88501 [==============================] - 113s 1ms/step - loss: 0.1417 - accuracy: 0.9585 - val_loss: 0.2846 - val_accurac
y: 0.9732
Epoch 4/10
88501/88501 [==============================] - 113s 1ms/step - loss: 0.1375 - accuracy: 0.9593 - val_loss: 0.1807 - val_accurac
y: 0.9751
Epoch 5/10
88501/88501 [==============================] - 113s 1ms/step - loss: 0.1349 - accuracy: 0.9598 - val_loss: 0.1535 - val_accurac
y: 0.9753
Epoch 6/10
88501/88501 [==============================] - 117s 1ms/step - loss: 0.1338 - accuracy: 0.9604 - val_loss: 0.2367 - val_accurac
y: 0.9757
Wall time: 11min 32s
```

Figure4.17: model training with accuracy and loss values

5 Experimental results and model evaluation

5.1 Test Metrics

Most of existing classification-based prediction models use Precision/Recall/F1 measure to evaluate the effectiveness of a model. In our experiments, we also use these metrics as evaluation metrics (although there are more metrics for evaluating a classification-based model).in addition to these metrics, we used the confusion matrix to evaluate our model, Precision measures the percentage of identified failed machines that are actually faulty. Recall measures the percentage of failed machines that are correctly identified over all the failed machines. F1 measure is the harmonic mean of precision and recall, which weights recall and precision equally. A confusion matrix is a summary of prediction results on a classification problem.

5.2 Evaluation results

We evaluated the performance of our model, we used the validation set to compute the accuracy of the model, we also computed the confusion matrix, precision and recall of our model.

Training accuracy

```
Entrée [95]: # training metrics
scores = model.evaluate(X_train, y_train, verbose=1, batch_size=200)
print('Accurracy: {}'.format(scores[1]))

93159/93159 [==============================] - 23s 250us/step
Accurracy: 0.9591129422187805
```

Figure4.18: training accuracy

Confusion matrix of training data

```
Entrée [96]: # make predictions and compute confusion matrix
y_pred = model.predict_classes(X_train,verbose=1, batch_size=200)
y_true = y_train
print('Confusion matrix\n- x-axis is true labels.\n- y-axis is predicted labels')
cm = confusion_matrix(y_true, y_pred)
cm

93159/93159 [==============================] - 23s 244us/step
Confusion matrix
- x-axis is true labels.
- y-axis is predicted labels

Out[96]: array([[76991,   662],
       [ 3147, 12359]], dtype=int64)
```

Figure4.19: confusion matrix of training data

Precision and recall of training data

```
Entrée [97]: # compute precision and recall
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
print( 'precision = ', precision, '\n', 'recall = ', recall)

precision =  0.9491590507641502
 recall =  0.7970463046562621
```

Figure4.20: precision and recall of training data

5.3 Test results

Table 4.2 shows the evaluation results of our model in identifying failed machines, our model achieves a good results, the accuracy of the test reached 86.6%, the average precision, recall, and F1-measure values are 89.3%, 69.8%, and 78.4%, respectively. Considering the highly imbalanced nature of google cluster data and the complexity of the problem, it is very challenging to achieve both high recall and high precision.

```
Entrée [98]: # test metrics
scores_test = model.evaluate(X_test, y_test, verbose=2)
print('Accurracy: {}'.format(scores_test[1]))

Accurracy: 0.8669141530990601
```

Figure4.21: test accuracy

```
Entrée [101]: # compute precision and recall
precision_test = precision_score(y_true_test, y_pred_test)
recall_test = recall_score(y_true_test, y_pred_test)
f1_test = 2 * (precision_test * recall_test) / (precision_test + recall_test)
print( 'Precision: ', precision_test, '\n', 'Recall: ', recall_test,'\n', 'F1-score:', f1_test )

Precision:  0.8939211136890951
 Recall:  0.6982239942007974
 F1-score: 0.784045584045584
```

Figure4.22: the effectiveness of our model

Metrics	Precision	Recall	F1-score
Results	89.3%	69.8%	78.4%

Table4.2: test results

We also used the confusion to visualize the effectiveness of our model, figure () shows the results we got.

```
Entrée [100]: # make predictions and compute confusion matrix
y_pred_test = model.predict_classes(X_test)
y_true_test = y_test
print('Confusion matrix\n- x-axis is true labels.\n- y-axis is predicted labels')
cm = confusion_matrix(y_true_test, y_pred_test)
cm

Confusion matrix
- x-axis is true labels.
- y-axis is predicted labels

Out[100]: array([[24931,  1143],
       [ 4163,  9632]], dtype=int64)
```

Figure4.23: confusion matrix of test results

6 Discussions

The first attempts of building the model we have got bad results, for that we tried to identify the problems and solve them, we will discuss these problems below:

Data collection:

At the beginning when we were looking for a dataset of cloud computing service, we experienced a hard time to find an accurate dataset because there a little few of them and most of them have so many errors during the collecting phase, and when we finally land with the choice on google dataset, it was a big size dataset so we have downloaded a small portions of it so that in the training phase the computer can handle big amount of data, the next problem with google dataset, is the highly imbalanced data, where there is labels occurs rarely, when other labels are strongly present which makes it tough to the model to predict the failure, to overcome this issue we combined all the different labels as failed rather than doing an oversampling process.

Hardware limitation:

Our setup consists of i7hq processor with 16 GB of ram and NVidia gtx1060 graphics card, which not enough to train a huge amount of data, we limited with a small amount of data that the computer can handle, to handle this amount of data it requires a cloud computing power, like the RDP.

Conclusion and Future Work

In this dissertation we investigated a new research area, which is cloud failure). The work developed in this dissertation is about applying deep learning techniques to predict failures in a cloud environment, which can help improve service availability, resource savings and maintain a good service quality. In this field of cloud, failure prediction based on deep learning techniques are being used recently, and have shown effectiveness over other traditional prediction techniques.

In this context, we have implemented a deep learning technique BilSTM, which is a special type of LSTM, on a dataset provided by google cloud, in order to predict failures in cloud using this dataset. Before choosing google dataset we spent a lot of time searching for an available dataset of cloud computing service, we faced a difficulty to find one to rely on. One important reason for not availability of datasets, is that most companies providers like Microsoft and Amazon do not offer it for public use. After a long search we arrived to use public google dataset. After that, we faced another problem which, is the big size of this dataset, which is impossible for us to handle all at once, so we ended up with using small chunks of it but sufficient, after that we ran through the preprocessing phase which the important step, we cleaned the data and extracted the features, after that we normalized the data, so that the model can learn effectively. After the preprocessing phase, we have split the data and feed to the model we built, the first results were not good enough, which led us to look for the problems encountered and solve them. Finally, after tuning the hyperparameters of the model and changing some features, we got perfect results.

As future works, we want to implement other deep learning techniques and do a comparative study for failure prediction.

References

[1] https://dictionary.cambridge.org/ .

[2] William Sullivan “Machine Learning for Beginners Guide Algorithms: Decision Tree & Random Forest Introduction” 2017.

[3] Sandro Skansi “Introduction to Deep Learning from Logical Calculus to Artificial Intelligence” https://doi.org/10.1007/978-3-319-73004-2 2018.

[4] Ratnadip Adhikari “an introductory study on time series modeling and forecasting”

[5] J. Lee, “Univariate time series modeling and forecasting (Box-Jenkins Method)”, Econ 413, lecture 4.

[6]John H. Cochrane, “Time Series for Macroeconomics and Finance”, Graduate School of Business, University of Chicago, spring 1997.

[7]K.W. Hipel, A.I. McLeod, “Time Series Modelling of Water Resources and Environmental Systems”, Amsterdam, Elsevier 1994.

[8]WALTER A. SHEWHART and SAMUEL S. WILKS "INTRODUCTION TO

TIME SERIES ANALYSIS AND FORECASTING"

[9] J. Brownlee. Deep learning for Time series.

[10] H. Park, “Forecasting Three-Month Treasury Bills Using ARIMA and GARCH Models”, Econ 930, Department of Economics, Kansas State University, 1999.

[11]R. Parrelli, “Introduction to ARCH & GARCH models”, Optional TA Handouts, Econ 472 Department of Economics, University of Illinois, 2001.

[12] https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/

[13] https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/

[14] https://en.wikipedia.org/wiki/Random_walk

[15] C Voyant, M L Nivet, C Paoli, M Muselli and G Notton "Meteorological time series forecasting based on MLP modelling using heterogeneous transfer functions" Journal of Physics: Conference Series, Volume 574, 3rd International Conference on Mathematical Modeling in Physical Sciences (IC-MSQUARE 2014) 28–31 August 2014, Madrid, Spain

[16] M.Akhoondzadeh "A MLP neural network as an investigator of TEC time series to detect seismo-ionospheric anomalies" **Advances in Space Research** Volume 51, Issue 11, 1 June 2013, Pages 2048-2057

[17] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung and Jun Won Choi "Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture" arXiv:1802.06338v3 [cs.LG] 22 Oct 2018

[18] https://aws.amazon.com/what-is-cloud-computing/

[19]. https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

[20] https://www.researchgate.net/figure/Overview-of-a-cloud-architecture-for-e-Learning_fig2_261502986

[21] https://www.nccgroup.com/uk/about-us/newsroom-and-events/blogs/2019/november/the-hidden-cost-of-cloud-failure/

[22] Nakka, N., Agrawal, A., and Choudhary, A. Predicting node failure in high performance computing systems from failure and usage logs. In 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (2011), IEEE, pp. 1557–1566

[23] Chalermarrewong, T., Achalakul, T., and See, S. C. W. Failure prediction of data centers using time series and fault tree analysis. In 2012 IEEE 18th Int'l Conf. on Parallel and Distributed Systems (Dec 2012), pp. 794–799

[24] Guan, Q., Zhang, Z., and Fu, S. Proactive failure management by integrated unsupervised and semi-supervised learning for dependable cloud systems. In 2011 Sixth Int'l Conf. on Availability, Reliability and Security (Aug 2011), pp. 83–90.

[25] Sîrbu, A., and Babaoglu, O. Towards operator-less data centers through datadriven, predictive, proactive autonomics. Cluster Computing 19, 2 (Jun 2016), 865–878.

[26] = Zhu, B., Wang, G., Liu, X., Hu, D., Lin, S., and Ma, J. Proactive drive failure prediction for large scale storage systems. In2013 IEEE 29th Symp. on Mass Storage Systems and Technologies (MSST) (May 2013), pp. 1–5

[27] = Pelaez, A., Quiroz, A., Browne, J. C., Chuah, E., and Parashar, M. Online failure prediction for HPC resources using decentralized clustering. In2014 21stInt'l Conf. on High Performance Computing (HiPC) (Dec 2014), pp. 1–9

[28] = LU, X., qiang WANG, H., jie ZHOU, R., and yu GE, B. Autonomic failure prediction based on manifold learning for large-scale distributed systems. The Journal of China Universities of Posts and Telecommunications 17, 4 (2010), 116 –124.

[29] Chen, X., Lu, C. D., and Pattabiraman, K. Failure prediction of jobs in compute clouds: A google cluster case study. In 2014 IEEE Int'l. Symp. on Software Reliability Engineering Workshops (Nov 2014), pp. 341–346.

[30] Zhu, B., Wang, G., Liu, X., Hu, D., Lin, S., and Ma, J. Proactive drive failure prediction for large scale storage systems. In 2013 IEEE 29th Symp. on Mass Storage Systems and Technologies (MSST) (May 2013), pp. 1–5.

[31] Islam, T., and Manivannan, D. Predicting application failure in cloud: A machine learning approach. In 2017 IEEE Int'l Conf. on Cognitive Computing (ICCC) (June 2017), pp. 24–31

[32] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, Murali Chintalapati, and Dongmei Zhang. 2018. Predicting Node failure in cloud service systems. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018). Association for Computing Machinery, New York, NY, USA, 480–490. DOI:https://doi.org/10.1145/3236024.3236060

[33] https://github.com/google/cluster-data

[34] https://threathunterplaybook.com/tutorials/jupyter/introduction.html#what-is-a-jupyter-notebook

[35] https://www.python.org/

[36] https://github.com/kiteco/tensorflow

[37] https://keras.io/

[38] https://pandas.pydata.org/

[39] https://numpy.org/

[40] https://scikit-learn.org/stable/

[41] https://matplotlib.org/

[42] https://seaborn.pydata.org/

ملخص

في الاعوام الاخيرة نماذج التعلم العميق أظهرت نتائج جيدة في العديد من المجالات (مثل معالجة اللغات الطبيعية و معالجة الصور الرقمية)، الباحثون بدؤوا باستخدام هاته النماذج على بيانات السلاسل الزمنية للتنبؤ بالفشل ، التنبؤ بالفشل في البيئة السحابية هو مجال بحث جديد حيث يحاول الباحثون تحقيق نتائج جيدة للحفاظ على توافر الخدمة للخدمة السحابية ، في هذا المشروع قمنا بتطبيق إحدى تقنيات التعلم العميق و هي الشبكات العصبية الاصطناعية بالتحديد الذاكرة طويلة المدى LSTM على مجموعة بيانات قوقل للتنبؤ بالفشل ، أظهر نموذجنا نتائج جيدة بعد تقييمه واختباره.

Abstract

In recent years deep learning techniques shown a great results in many fields (such as natural language processing, computer vision). Among their new applications, researchers started to apply these techniques on time series data to predict failures. Predicting failures in cloud environment is a recent research area where researchers try to achieve good results to maintain the service availability of a cloud service. In this project we applied one of deep learning techniques which is neural networks, specifically LSTM on google cluster dataset to predict failures, our model shown good results after extensive evaluation and experimental test.

Résumé

Au cours des dernières années, les techniques de deep learning ont donné des bons résultats dans de nombreux domaines (tels que le traitement du langage naturel, la vision par ordinateur). Les chercheurs ont commencé à appliquer ces techniques sur des données de séries chronologiques pour prédire les défaillances. La prédiction des défaillances dans les environnements de cloud est un nouveau domaine de recherche dans lequel les chercheurs essayer d'obtenir de bons résultats pour maintenir la disponibilité du service. Dans ce projet, nous avons appliqué l'une des techniques de deep learning qui est les LSTM sur la base de données du Google cluster pour prédire les défaillances, notre modèle a montré de bons résultats après l'avoir évalué et testé.