Machine Learning Based Heuristic BBU-RRH Switching Scheme for C-RAN in 5G

by Jiamo Liu

Prepared for O. Falowo
Department of Electrical Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Masters of Science degree in Telecommunication Engineering.

February 2019

Declaration

1. This report is my own work.
2. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.
3. I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor

Signature: Signed by candidate

Jiamo Liu

Date: 11/06/2018

Acknowledgements

I would like to express my gratitude to my parents who support me through my studies and serve as great role models. Furthermore, I would like to thank my supervisor for providing funds as well as guiding me through my masters programme. I also like to show my appreciation for my friend and colleague Udit Paul who has been helpful and make my masters journey enjoyable. Lastly, I would like to thank my girlfriend for always being on my side.

Abstract

The immense increase in bandwidth demand by various services such as high definition video streaming, online gaming, and virtual reality have made it increasingly challenging for operators to be able to provide satisfactory services to the end users while making a profit. Cloud Radio Access Network (C-RAN) is a new architecture that has been proposed to facilitate the mobile networks' ability to meet the increase in bandwidth demand.

C-RAN consists of three parts, namely Remote Radio Head (RRH), the front haul link, and Baseband Processing Units (BBU) pool. Many RRHs are associated with one BBU pool, and all RRHs within the pool are logically connected to every BBU in the pool. Thus, a BBU-RRH switching algorithm needs to be developed as it is able to enhance the performance of such architecture while managing the resource efficiently. This work mainly focuses on providing a traffic profile prediction based BBU-RRH switching algorithm using a real life dataset. There are relevant works that propose algorithms to achieve this purpose, however some of these algorithms suffer from high switching complexity while others fall short in QoS provisioning. Therefore, this work aims to develop a BBU-RRH algorithm that will be able to enhance the QoS while reducing the switching complexity with the aid of machine learning techniques. This algorithm consists of three parts. The first part proposes an efficient RRH clustering mechanism that determines which RRHs are associated with a specific BBU pool. The second part utilises recurrent neural networks (RNN) to predict the daily traffic profile of RRHs, so that a relatively accurate traffic profile prediction can be obtained to facilitate the switching algorithm. Finally the third part is the BBU-RRH switching scheme that works in conjunction with the predicted traffic profile to make an informed decision about the associations between RRHs and BBUs within the BBU pool.

The simulations have suggested that the proposed algorithm is able to reduce the number of BBUs used and therefore save on energy. In addition, the algorithm also reduces the occurrence of congestion and failure states, and thus improves the quality of the service of the network. Finally, this switching algorithm also reduces the switching complexity in comparison to an existing algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to the study

Mobile networks play a crucial role in data transferring and communication. It is envisioned in [1] that the monthly mobile network data demand will be approximately 50 Exabytes in 2021, and the increase is almost 7-fold in comparison to the demand in 2016. Such demands can no longer be satisfied with the current paradigm of mobile network. Therefore, Fifth Generation (5G) network is introduced as the next generation mobile network solution that offers excellent quality of service (QoS) to its subscribers. Cloud Radio Access Network (C-RAN) is proposed as one of the potential RAN architecture for 5G mobile networks due to the following reasons [2]:

- It aligns with the cloud-computing paradigm of 5G
- It reduces energy consumption of the network
- It is suitable for non-uniform traffic
- It increases throughput while decreasing delay
- It is easy to maintain and upgrade

In C-RAN, multiple Remote Radio Heads (RRHs) are associated with one Baseband Processing Unit (BBU) pool. The RRHs merely provides radio coverage while the BBU provides computational power needed for the network. Each

RRH is logically connected to every BBU in the pool and the connections can be dynamically adjusted. Therefore an efficient BBU-RRH switching algorithm is required to manage the computational resource, and such algorithm can potentially affect QoS provided by the network.

Machine Learning is a technique that allows the computer to learn to perform a certain task without being explicitly programmed [3], and such technique is currently the state-of-art in many fields that involve complex non-linear data treatment [4]. In this work, machining learning is applied to identify potential clusters of RRHs that will be served by one BBU pool, as well as predicting the daily traffic profile of cells. This work focuses on developing a BBU-RRH switching algorithm that attempts to minimise the number of active BBUs while improving QoS.

1.2 Objectives of this study

1.2.1 Problems to be investigated

The problem under investigation is to develop an efficient BBU-RRH switching algorithm that dynamically adjusts the associations between RRHs and BBUs in C-RAN. This algorithm will be developed with the knowledge of daily traffic profile and clustering of RRHs, which is obtained by machine learning on a real world dataset.

1.2.2 Purpose of the study

This study contains the following as its purposes:

- Process a real world big data set
- Cluster RRHs that will be served by a single BBU
- Predict the traffic profile of each RRH or cell
- Establish a BBU-RRH switching algorithm that will reduce energy consumption and switching complexity while improving QoS

Finally the algorithm is validated by method of simulation.

1.3 Scope and limitations

The scope and limitations are:

- 5G standards are not clearly defined at the moment
- The RRHs of only one operator is considered
- Due to lack of details of RRHs (such as coverage, power, *etc.*), the RRHs are arranged according to hexagonal convention, which indicates no overlapping between coverage of RRHs
- Aggregate distance is the only parameter used in RRH clustering
- The increase or decrease is assumed to be linear within the sampling period of the traffic profile model
- The volume of traffic is scaled in the dataset
- The front haul link capacity is assumed to be sufficient
- One RRH can only belong to one BBU within the BBU pool

1.4 Plan of development

The plan of development of this report is as follows:

- Chapter 2 - Presents relevant literatures that will be useful in fulfilling the purposes of study
- Chapter 3 - Presents the dataset used in the study
- Chapter 4 - Presents and validates the implementation of the RRH clustering method
- Chapter 5 - Presents and validates the implementation of the traffic profile prediction model for the RRHs
- Chapter 6 - Presents and validates the implementation of the BBU-RRH switching algorithm

- Chapter 7 - Draws conclusion based on the obtained results and discussion
- Chapter 8 - Suggests future work to improve on the current study

1.5 Author's Publication

Part of the work presented in this thesis is published by the author in:

- J. Liu and O. Falowo, Traffic-aware heuristic bbu-rrh switching scheme to enhance QoS and reduce complexity," in 2018 IEEE 29th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Sep 2018.
- S. T. O. F. G. M. Jiamo Liu, Udita Paul, K-means based spatial base station clustering for facility location problem in 5g," in 2018 The annual Southern Africa Telecommunication Networks and Applications Conference, pp. 44{49, Sept 2018.

Chapter 2

Literature Review

2.1 The Fifth Generation of Wireless Network

This section gives an overview of the progress and status of fifth generation of wireless networks.

2.1.1 Shortcomings of Traditional Mobile Networks

The volume of mobile traffic is ever-increasing [1] and such increase will only become even more explosive, as more use cases such as Internet of Things (IoT), virtual reality (VR) are proposed from the demand side. According to [5], an average mobile user is expected to consume around 1 terabyte annually by the year 2020. Thus, it becomes an almost impossible task for the current LTE system to accommodate such explosive growth in data demand and the ever-expanding use cases. To put it in perspective of LTE networks [6], an ideal LTE network with a theoretical downlink data rate of 150 Mbps and a 2×2 MIMO can only support less than 40 full HD video streaming if the rate of streaming is 4 Mbps. In addition, LTE network is designed to accommodate 600 RCC-connected users within a cell [6], however this is far from sufficient for use cases such as IoT or Machine to Machien Communications (M2M), where each service could potentially require thousands of devices connected to a single cell. Even though there are many research that aims to enhance the capacity of the current LTE networks, this is however only a temporary solution. Thus, a new mobile network technology is required to sustain these new use cases in the long run.

2.1.2 5G Overview

5G is considered to be the next major technological advancement from the current 4G wireless networks. In addition to the enhanced QoS, the 5G networks are also expected to accommodate a wider range of vertical industries and play a more significant role in the daily life. However, 5G is still in the rudimentary stage and its standard is not yet available at the time of writing. The performance requirements or specifications of 5G mobile networks can be characterised as the following [6]:

- Data rates of 1 Gbps or more: This envisioned downlink data rate is at least 10 times of the theoretical maximum data rate (150 Mbps) of LTE network
- Low round trip delay: The delay of the 5G networks is required to be less than 1 ms, which is one tenth of the delay of LTE networks
- Massive number of connections: One of the main objectives of 5G networks is to enable use cases such as IoT and M2M, and this requires 5G networks to provide connections to thousands of devices
- Ultra-reliability: Due to some mission-critical use cases of 5G, the network is required to be available for connection almost ubiquitously. The degree of reliability is envisioned to be 99.999%
- Reduced energy consumption: The energy consumption of 5G network is predicted to be a tenth of that of LTE networks

It can be easily observed from above that 5G indeed has some demanding specifications, and fulfilling these demands is not trivial. Therefore, a set of 5G facilitators are proposed to aid the implementation of 5G, and some of these facilitators will be discussed in the following sections. Figure 2.1 shows an schematic overview of 5G networks.

2.1.3 5G Facilitators

Numerous technologies have been proposed to be incorporated as part of the 5G network. Motivations and brief descriptions of these facilitators will be presented in the following sections.

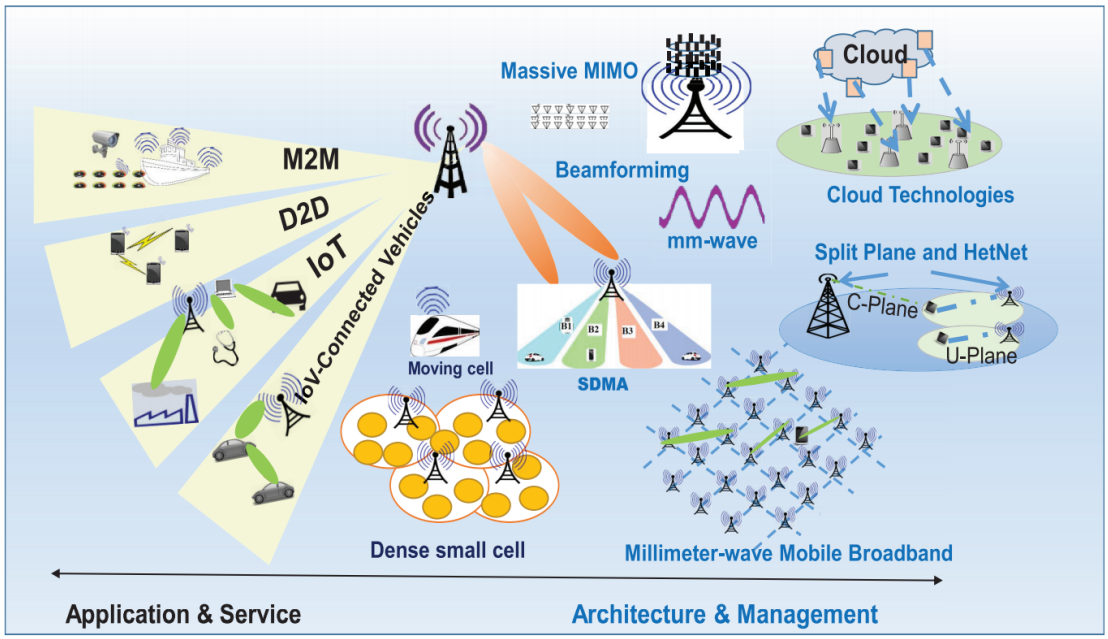

Figure 2.1: Schematic diagram of 5G [6]

Software Defined Networks

The high demand for data rate and connectivity of 5G network results in a more dense cell deployment, and this leads to an increased number of network elements. Configuration and management of these elements becomes a complex problem and software defined networks (SDN) is proposed as one of the potential solution. SDN separates the data plane from the user plane [7], and this allows more flexibility and scalability to the network. This idea is shown in Figure 2.2. It should be noted that this plane separation enables the control signals to be transmitted on a different band that is independent of user or data plane, which leads to an decrease in control signal overhead [8]. Such segregation is achieved by programming various off-the-shelf components in software and thereby reducing the reliance on task-specific hardware. Due to the software based design of the network, the configuration and management of the network can now be performed remotely at a central controller, which greatly simplifies the process [7].

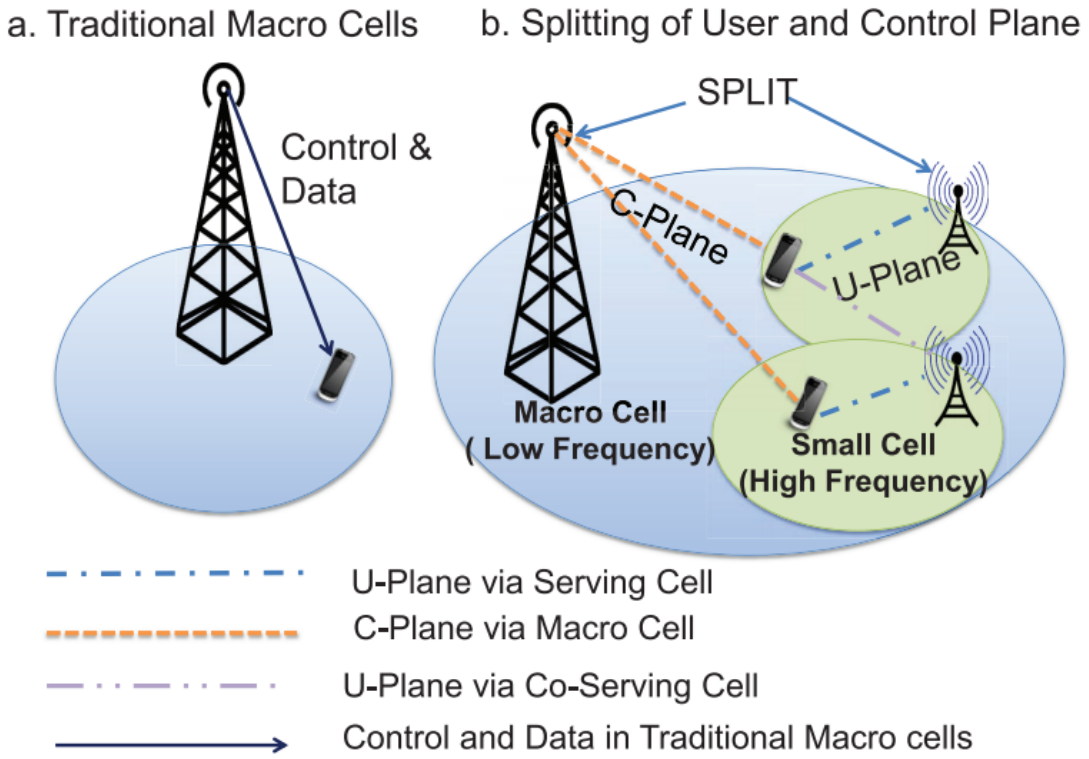

Figure 2.2: Separation of user plane and control plane [6]

Network Slicing

5G is expected to host many use cases or vertical industries and it is obvious that the QoS requirements of these vertical industries differ from each other. For example, a health monitoring service favours reliability over throughput, whereas mobile broadband services prefer networks with high data rate. It is therefore of great interest to be able to provide differentiated services to different use cases as an operator [9]. The use cases can be classified into the following categories based on their QoS requirements [9]:

- Enhanced mobile broadband (eMBB): It generates significant amount of data, and requires high data rate
- Critical communication (CriC): It requires the network to be ultra-reliable while having a very low latency, the volume of data transferred is usually relatively less

- Massive IoT (mIoT): This use case is characterised by a massive number of connections as well as lack of mobility
- Vehicle-to-X (V2X) communications: This service has the characteristics of CriC with additional demands such as high speed accuracy

In contrast with the traditional one-size-fits-all approach, network slicing provides multiple end-to-end logical networks on a common physical infrastructure, and this is depicted in Figure 2.3. All these logical networks or slices are isolated from each other, and they can be created and managed on demand using software. Each slice can serve subscribers with similar demands and characteristics, therefore each slice can be optimised for the specific task and this leads to an enhanced QoS while saving on network resources [10].

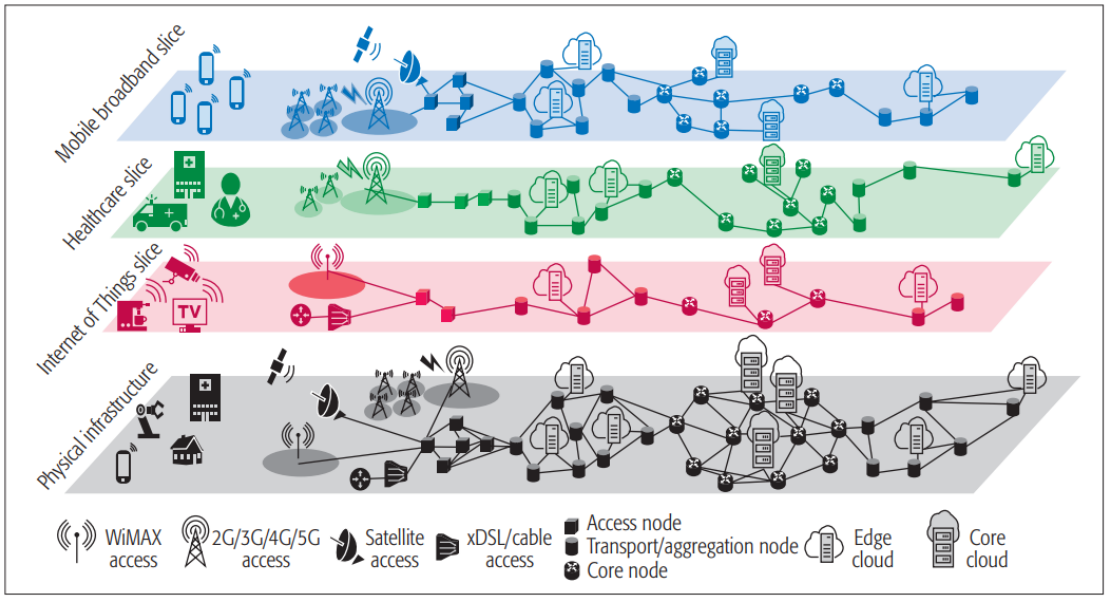

Figure 2.3: Multiple task-specific independent logical networks on a shared physical hardware [10]

The mechanisms used to achieve this highly flexible networks are SDN and network function virtualisation (NFV). NFV essentially allows a network function to be deployed on a generic hardware using software programming, whereas SDN allows the network to be easily managed via software. These techniques allow operators to lease physical infrastructure from an infrastructure provider (InP)

and run their services on those generic hardware. This new approach (shown in Figure 2.4) will empower the network with more scalability and flexibility while saving the operators from providing and managing the physical hardware.

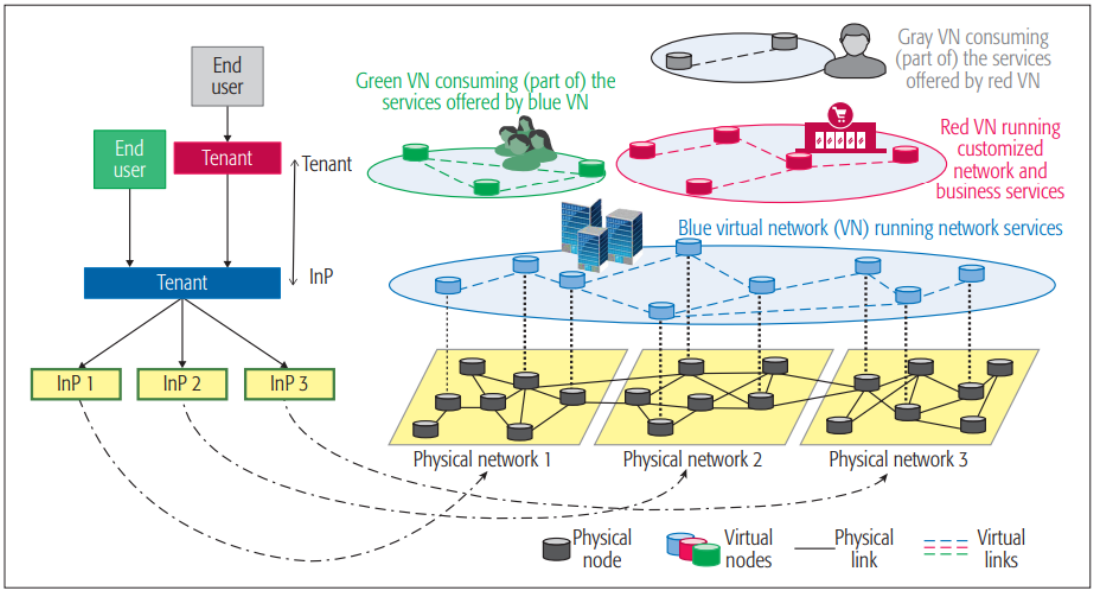

Figure 2.4: Interactions between principal participants in network slicing [10]

2.1.4 Cloud Radio Access Network

One of challenges in mobile network is the non-uniformity of volume of mobile traffic due to spatial and temporal behaviours of subscribers. For example, there will be more mobile traffic in a residential area at night in comparison to that of an office area. This temporal and spatial related trend in volume of traffic, shown in Figure 2.5, is known as the tidal effect [11]. It can be observed in Figure 2.6 that the non-uniformity is reduced and smoothed out by aggregating different traffic patterns or profiles together. Thus, C-RAN is proposed as the potential solution to address tidal effect and other challenges of mobile networks in the 5G context.

In the traditional approach, each base station is equipped with computation units that are responsible for processing signals. However this approach is clearly inefficient, because the capacity of these units has to be designed according to peak traffic to ensure ubiquitous connection for subscribers, which leads to

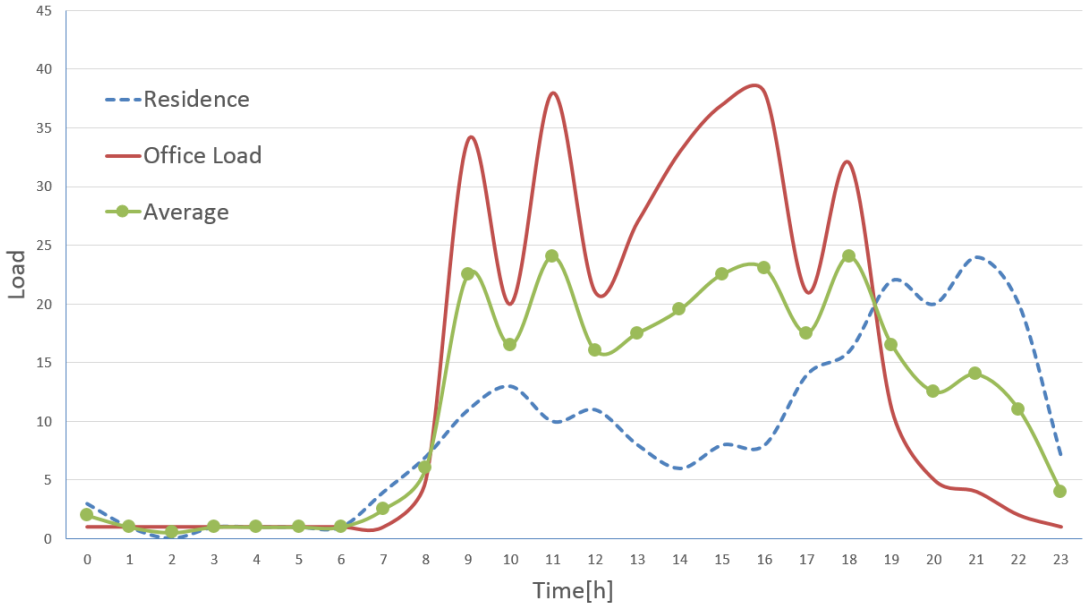

Figure 2.5: The non-uniformity of mobile traffic load of different regions during a day

a significant amount of wastage in computation resources because each BBU is under-utilised most of the time. Furthermore, it also becomes extremely challenging to manage and maintain these units as the network scales, and this is caused by the de-centralised placements of these computational units. In order to mitigate some of these challenges, the computational units are now placed at a more convenient and centralised place that could be as far as 40 kilometres from the RRH [13] [12]. This, of course, makes management and maintenance simpler while reducing energy usage for cooling and etc. However, the associations between RRH and computational units remain static, which still leads to a significant waste in computing power. C-RAN is then designed to address the shortcomings of the previously mentioned approaches. C-RAN also has remote RRHs that allow simpler maintenance and management of the network while collecting computational resources into a pool called BBU pool. In addition, all computational units or baseband units within pool are logically connected to every RRH that the pool serves, which allows RRHs within under-utilised BBUs to be switched to other BBUs and thereby reducing the total number of active BBUs. This feature improves BBU resource utilisation while reducing the energy

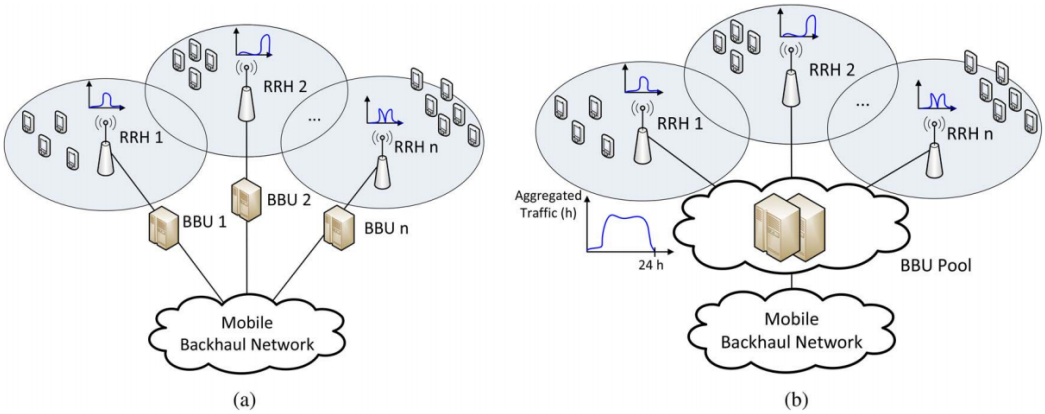

Figure 2.6: Aggregation of different traffic profiles produce a much smoother aggregate traffic profile [12]

consumption by shutting down un-used BBUs [12]. In addition to the RRHs and BBUs mentioned above, C-RAN also requires a high-performance fronthaul link to transmit data between the BBU pool and the RRHs, and optical fiber is envisioned to be the most suitable candidate for this purpose [14]. A general C-RAN network is illustrated in Figure 2.7.

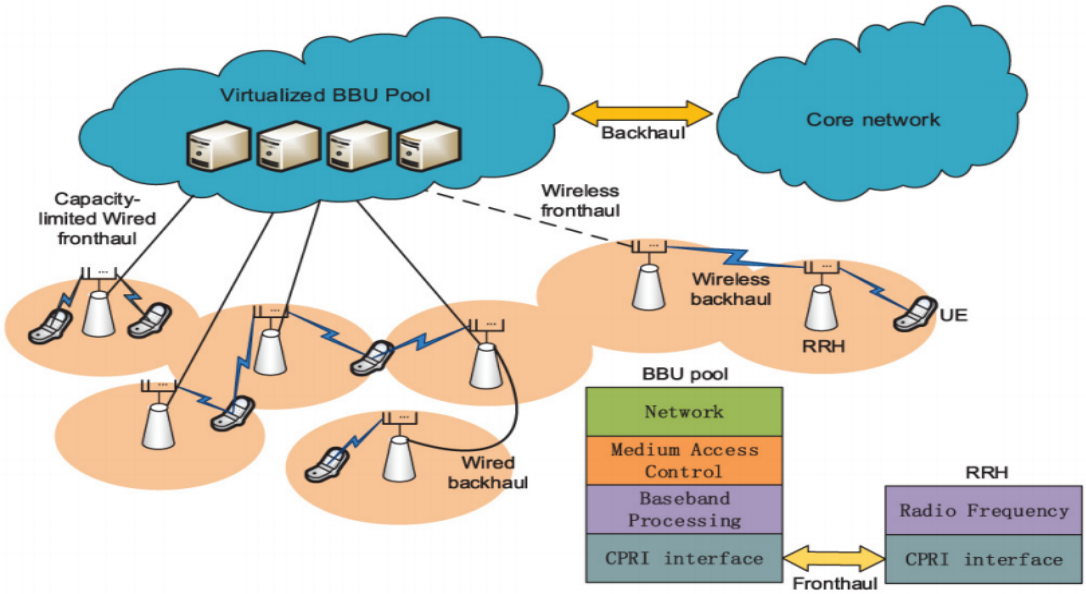

Figure 2.7: A general C-RAN network [14]

2.2 Traditional Time Series Prediction

This section gives an overview of two classical methods used in regression problems.

2.2.1 ARIMA

ARIMA stands for Auto-Regressive Integrated Moving Average, it is one of the most widely adopted statistical method for time series prediction. It is a generic statistical process that can be transformed into different auto-regressive models when its parameters are modified. An ARIMA model consists of two processes, namely auto-regressive (AR) model and moving average (MA) model.

In an AR model, the predicted value or variable depends on a linear combination of its past values, and a p^{th} order AR(p) model is denoted by the following equation [15]:

$$y_t = c + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p} + e_w \tag{2.1}$$

Where c is the mean of the process or data samples, y_t is the predicted sample of data, α is the weight of past observations, and e_w is the white noise. The α values are determined using various non-linear error minimisation or log likelihood maximisation methods, and these methods usually exist as a black box within different statistical software packages.

It should be noted that ARIMA performs well when the process is stationary or $c = 0$. Therefore the differencing method is applied to all data samples to ensure that the process is stationary and properties of the time series are independent of the time of observation. This transformation is governed by the following equation:

$$\hat{y_t} = (y_t - y_{y-1}) - (y_{t-1} - t_{t-2}) - \ldots - (y_{t-d+1} - y_{t-d}) \tag{2.2}$$

where d is the degree of differencing.

The MA model predicts the current value y_t by linearly combining the past forecast errors, and a q^{th} order MA(q) model is represented by the following equation:

$$y_t = c + e_t + \beta e_{t-1} + \ldots + \beta_q e_{t-q} \tag{2.3}$$

where e_t is the error of past predictions, and β is the weight of past errors. Combining these equations, a complete ARIMA(p, d, q) is obtained as the

following:

$$\hat{y}'_t = c + \alpha_1 \hat{y}'_{t-1} + \alpha_2 \hat{y}'_{t-2} + \ldots \alpha_1 \hat{y}'_{t-p} + e_t + \beta e_{t-1} + \ldots + \beta_q e_{t-q} \tag{2.4}$$

Where $\hat{y}'_t$ is the predicted value of differenced data samples. This ARIMA process can be transformed into the following auto-regression models given specific parameters:

Model Name	ARIMA model
White Noise	$ARIMA(0,0,0)$
Random Walk	$ARIMA(0,1,0)$
Auto Regression	$ARIMA(p,0,0)$
Moving Average	$ARIMA(0,0,q)$

2.2.2 Kalman Filter

Kalman filter recursively operates on a linear state space, and this filter is able to predict the future states in addition to estimating the current states which deviate from true values due to noises. This filter can be further decomposed into prediction phase and correction phase. This filter initially predicts the state variables in the linear state space, and then the state predictions are corrected during the second phase. This method iterates until the error covariance arrives at a local minima in the optimisation procedure [16]. The state space model used in Kalman Filter is given as:

$$y_{k|k-1} = Ay_{k-1|k-1} + Bu_{k-1|k-1} + e_{k-1|k-1} \tag{2.5}$$

The measured states are related to the predicted states by the following relationship:

$$z_{k|k-1} = Hy_{k-1|k-1} + n_{k-1|k-1} \tag{2.6}$$

y is a matrix that contains all state variables of interest. A is the transformation matrix that relates current states to future states in the absence of externalities e. u is a controlled input and B describes the system reaction to input. H maps the true states into the observation states, e, n are modelled as white noise with a mean of 0 and covariance P, Q respectively. The correction phase is governed by the following equations:

$$\hat{y}^- = A\hat{y}_{k-1} + Bu_{k-1} \tag{2.7}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.8}$$

$\hat{y}^-$ is the *a priori* estimation, $\hat{y}$ is the *a posteriori* estimation, P^- denotes the *a priori* estimation of the error covariance matrix and P denotes the *a posterori* error covariance matrix. Furthermore, the relationship between the *a priori* estimation and the *a posteriori* estimation is defined as the following:

$$\hat{y_k} = \hat{y_k^-} + K(z_k - H\hat{y_k^-}) \tag{2.9}$$

The difference represents the residual or the error, which can be interpreted as how much the measured states deviate from the observed states. K denotes the Kalman gain which should minimise the *a posteriori* error covariance matrix P. After the above mentioned prediction phase, correction phase is introduced to enhance the prediction accuracy. The correction phase is depicted by the following equations:

$$\hat{K} = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.10}$$

$$\hat{y_k} = \hat{y_k^-} + \hat{K}(z_k - H\hat{y_k^-}) \tag{2.11}$$

$$P_k = (\hat{I} - \hat{K}H)P_k^- \tag{2.12}$$

This correction phases calculates the *posteriori* error covariance matrix and estimates based on the *a priori* error covariance matrix and estimates. The updated matrix are then used to calculate the next states in the state space model, and this iterative approach can be used to extrapolate the future states.

2.3 Machine Learning Methods

Machine learning is currently the state-of-art technique for regression and classification problems, because it is able to produce a much more accurate result for a highly complex and non-linear dataset in comparison to its traditional counterparts [17]. This section gives an overview of a few popular machine learning techniques for regression problems.

2.3.1 Linear Regression

Linear regression is the simplest form of machine learning regression techniques, and this section serves to illustrate some common key concepts in machine learning techniques [18]. Consider a scenario that we have a 2-dimensional (1 feature) data set D in the form of:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)\} \tag{2.13}$$

We are trying to approximate a relationship between x and y based on the existing data, furthermore we also have the additional insight that the relationship should be approximately linear. We can then fit a regression line L for the data set, and L can be written as the following form:

$$L : y = b_0 + b_1 x \tag{2.14}$$

The initial values of both b_0 and b_1 are chosen arbitrarily, which means that the regression line most likely fits poorly for the given dataset D. It is obvious that by changing parameters b_0 and b_1, we can adjust the shape of the line L, and we should adjust it until a good fit occurs. Thus, we can measure how well the line fits the data by computing a cost function $e(b_0, b_1)$:

$$e(b_0, b_1) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2.15}$$

Where $\hat{y}$ is the predicted or fitted value, and y is the actual value in the data set. It can be observed that the cost function is essentially the mean squared error of the regression. The objective is then to minimise the cost function iteratively so that L can fit the dataset well, and the technique used for this is called gradient descent.

Gradient Descent

When the cost function e reaches a local minima then the partial derivative with respect to all variables should be 0. This basic idea allows us to traverse the cost function in the direction of the gradient until a saddle point is reached, thus the gradient descent is represented as the following:

$$\hat{b_0} = b_0 - \alpha \frac{\partial}{\partial b_0} e(b_0, b_1) \tag{2.16}$$

$$\hat{b_1} = b_1 - \alpha \frac{\partial}{\partial b_1} e(b_0, b_1) \tag{2.17}$$

Where b_0, b_1 are the updated parameters and α is called the learning rate, which modifies how much we descent in the direction of the gradient. These steps should be repeated until a saddle point is reached, however there are a few shortcomings with this simple implementation. The first shortcoming is that the cost function is not necessarily convex, therefore the saddle point could be either a local maxima or a local minima, and this local minima and maxima does not guarantee a satisfactory result. More sophisticated stochastic methods, such as Adam optimiser, are implemented to overcome these challenges. Another very important challenge is the choice of learning rate α, inappropriate α value could lead to failure to converge, as a result some fine tuning and heuristic insight is required. This linear regression can be visualised in Figure 2.8

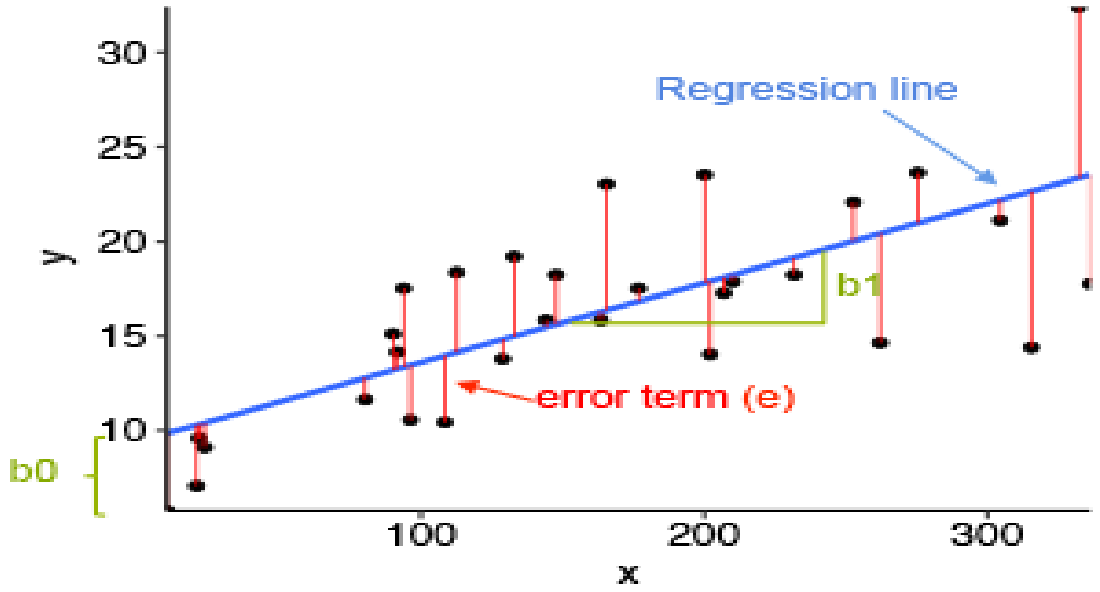

Figure 2.8: Graphical Visualisation of Linear Regression [18]

2.3.2 Neural Networks and Deep Learning

Linear regression and similar supervised learning methods all require prior insight of the shape of the function that would fit the data well. However, a lot of

regression and classification problems in reality are highly complex and non-linear, therefore these methods are not the most ideal approaches [19]. Artificial neural networks, which is inspired by the operation of biological neutral networks, are the state-of-art techniques that are proven to produce satisfactory results for these problems. A single hidden layer artificial neural network architecture can be represented in Figure 2.9.

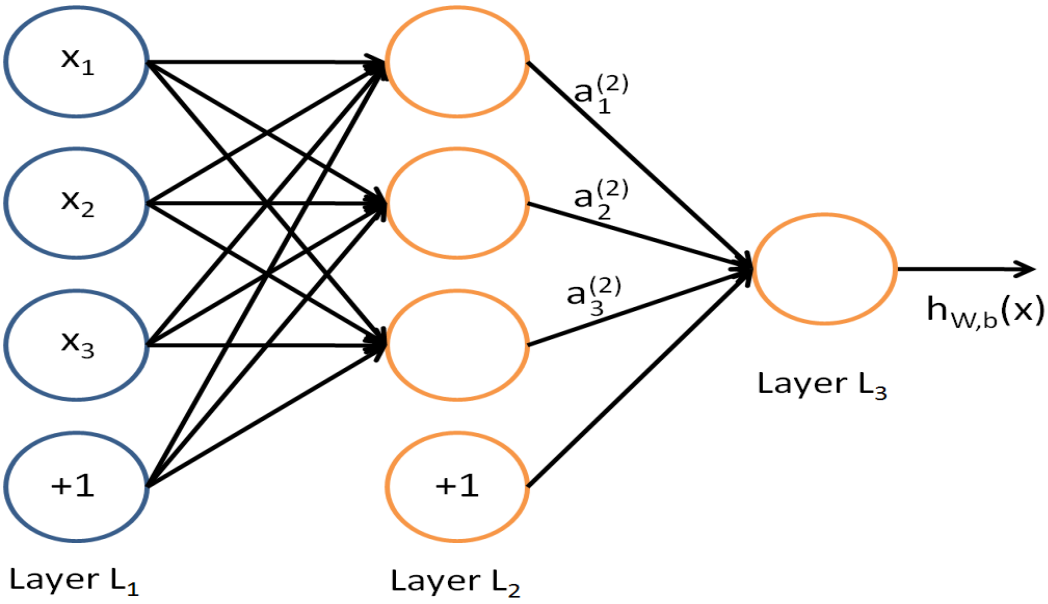

Figure 2.9: Single Hidden Layer Neural Network Architecture [20]

In Figure 2.9, 3 features are fed into the neural networks, which are depicted by x_1, x_2, x_3 in the input layer L_1. The $+1$ term is called the bias term, which is used to offset any constant term that potentially exists in the dataset. The connections in the form of black arrows between $L1$ and $L2$ are essentially weights, w, that are applied to the computation that each neuron performs. a are outputs computed by neurons in $L2$, and they are fed as inputs into the output layer $L3$. h in this case is the output of the entire neural network. This architecture can be written as $4X4X1$ to denote the number of neurons in each layer.

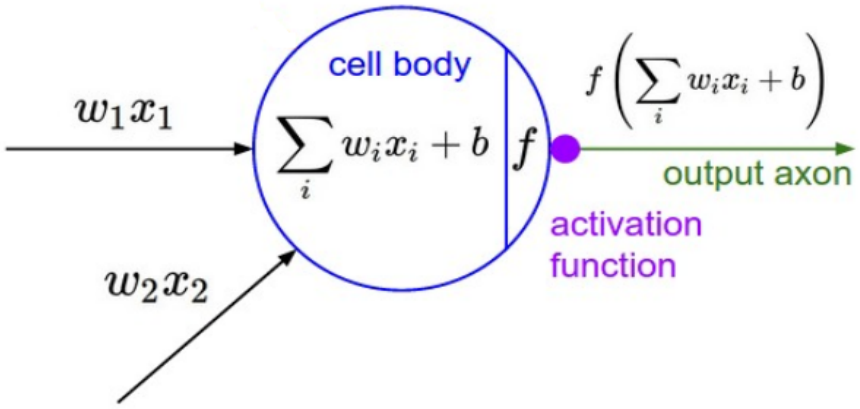

Figure 2.10: Single Neuron [20]

Single Neuron

Assume that $x_1, x_2, \ldots x_n$ are fed into a single neuron. The output, a, of the i^{th} neuron will be governed by the following equation:

$$a_i(w, b) = f(\sum_{j=1}^{n} w_j x_j + b_i) \tag{2.18}$$

f is called an activation function, which is used to induce some non-linearity to the sum of weighted inputs, and b is a constant term which attempts to offset the constant values. An overview of a single neuron is given in Figure 2.10.

Activation Function

Activation functions are used to map the linear output to a non-linear space, as well as limiting the output to a given range of values. This section shows the plots and mathematical expressions of a few popular activation functions which are shown in Figure 2.11. The formula of $tanh(z)$ is given by:

$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.19}$$

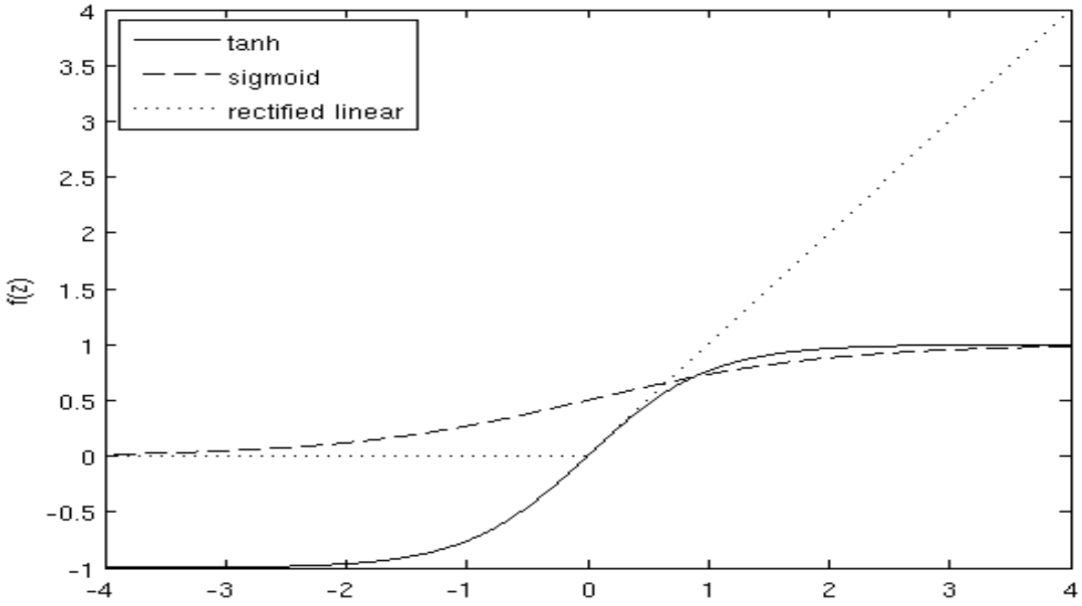

Figure 2.11: Examples of Activation Functions [20]

The sigmoid activation function is governed by:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.20}$$

Finally the rectified linear function is given by:

$$f(z) = \max(0, z) \tag{2.21}$$

z in our example is the sum of weighted inputs into one neuron. The activation functions should be chosen with care, as it plays an important role in the performance of the neural networks. For example, the *tanh* has a range of $[0, 1]$, which makes it an ideal choice when dealing with probabilities. The best activation function choice is usually not obvious, therefore one may have to experiment with different activation functions before finding the ideal one.

Forward Propagation

At this stage, the governing equations of each neuron in the hidden or intermediate layers are established, and it is possible to compute the final output h using above defined rules if the weights that each link represents are known. The final output

h_w, b in Figure 2.9 can then be represented as the following:

$$h(w,b) = f(\sum_{j=1}^{4} a_j f(\sum_{i=1}^{4} x_i w_i + b)) \tag{2.22}$$

However, this approach, which is called forward propagation, would require manually or iteratively defining each link with a weight value, which is obviously not scalable and sub-optimal when the size of the neural networks gets larger. Thus, an automatic weight tuning mechanism, called back propagation, is introduced to overcome this challenge.

Back Propagation

Similar to the idea of gradient descent shown in the previous section, it would be of great interest to study how the weights, w, would affect the performance of the neural network, and this study enables automatic tuning of weights in neural networks. Firstly, the cost function of the neural networks in Figure 2.9 for m training examples is defined as:

$$e(w,b;x,y) = \frac{1}{2}||h_{w,b}(x) - y||^2 \tag{2.23}$$

$$e(w,b) = [\frac{1}{m}\sum_{i=1}^{m} e(w,b;x,y)] + \frac{\lambda}{2}\sum_{l=1}^{2}\sum_{i=1}^{4}\sum_{j=1}^{4}(w_{ji})^2 \tag{2.24}$$

Equation 2.23 is similar to the cost function defined for gradient descent, which essentially computes the squared difference between the hypothesis and the target value. The first term in Equation 2.24 computes the average error for m training examples under the current w, b and the second term is called a weight decay, which penalises larger weights and allows the model to change more smoothly and generalise better. λ is a constant that controls how severe the penalty is. Equation 2.24 is a function of a vector of w and b, therefore it is possible to minimise the error with respect to different w and b, and this idea resonates with that of gradient descent, the equations are as follows:

$$\hat{w_{ij}^l} = w_{ij}^l - \alpha\frac{\partial}{\partial w_{ij}^l}e(w,b) \tag{2.25}$$

$$\hat{b_i^l} = b_i^l - \alpha\frac{\partial}{\partial b_i^l}e(w,b) \tag{2.26}$$

Where l denotes the layer involved in the calculation, i, j denotes the neurons involved, α is the learning rate and $\hat{w}, \hat{b}$ denotes the updated values after back propagation. However, it should be noted that this method is not only computationally expensive, but also facing similar problems of that of gradient descent.

Over-fitting Prevention

One of the biggest challenges of neural networks model is over-fitting. Over-fitting occurs when the neural networks work extremely well on the training set, however fails to generalise and produce reasonable output when a new batch of similar data is fed in. Examples of under-fitting, over-fitting and desired fitting are given in Figure 2.12.

Figure 2.12: Examples of Under-fitting, Desired and Over-fitting [21]

One of the most popular technique to combat over-fitting is called early stopping. Assume a data set of size n is provided for the neural network model, early stop mechanism divides the data set into 3 sections, namely training set, validation set and test set. The convention of the ration between the three sets are usually 70%/15%/15%. The neural network will train on the given training set and monitor the error of the model on the validation set. If the error on the training set is extremely low in comparison to that of validation set, then it is possible that over-fitting has occurred, and this can be shown in Figure 2.14. Therefore we need to stop the training early enough so that the generalisation is preserved during the training process. This idea can be illustrated in Figure 2.13.

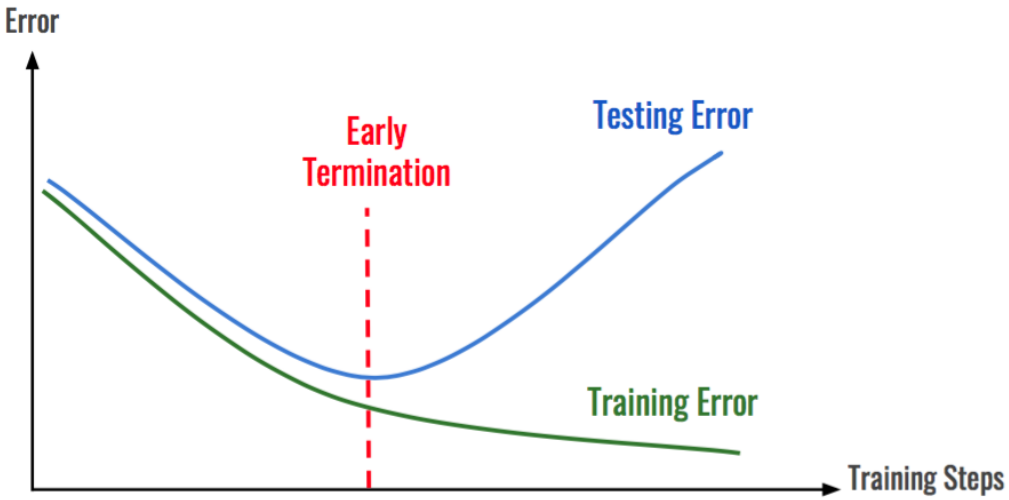

Figure 2.13: Early Stopping for Training [21]

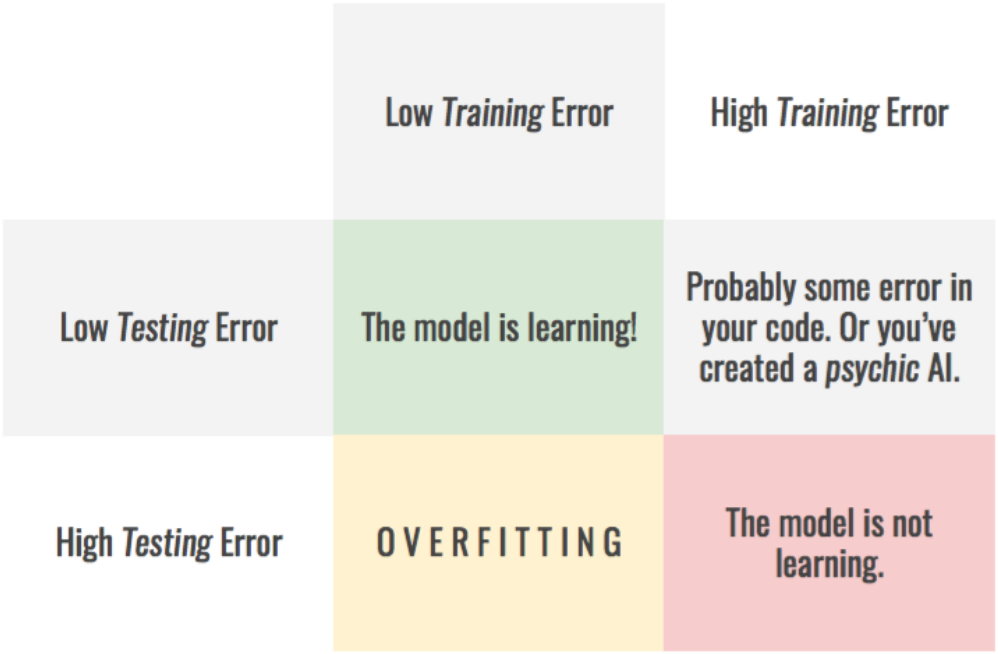

Figure 2.14: Over-fitting detection [21]

2.3.3 Recurrent Neural Network Models (RNN)

The models introduced above are feed-forward models, which means no feedback loops are present in the architectures. Those models are widely adopted in pattern

recognition as they attempt to directly relate the inputs to the outputs without taking consideration of the possible impacts of previous inputs. By introducing feedback or loops into feed forward architectures, the neural network model is able to memorise states that are related to previous inputs, and the final output is obtained by taking current as well as past inputs into consideration. This is particularly useful when the order of data is relevant. Because of this unique feature of recurrent neural network model, it is generally regarded as the state-of-art method to solve time-series regression problems. The architectural differences between the two models are illustrated in Figure 2.15. RNN models can be further

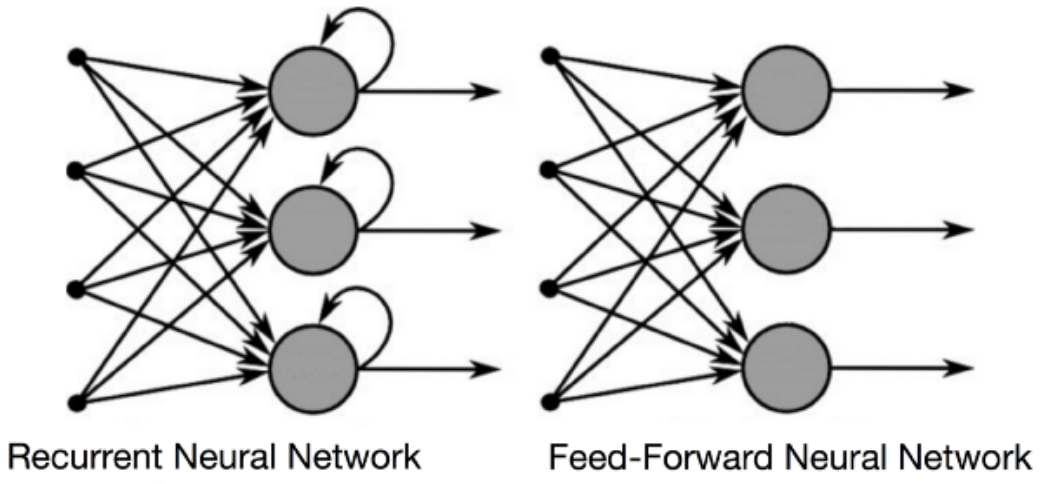

Figure 2.15: Feed-forward VS RNN architectures [22]

expanded or unrolled into a series of neural networks whose size depends on the number of past inputs deemed relevant. This can be shown in Figure 2.16. In Figure 2.16, the left shows the compact representation of RNN model which can be unrolled into a sequence of 3 neural network models if 2 past inputs are of interest. x_t is the input at the current time instant, U, W, V are the weights of the RNN model, s_t is the hidden state or memory at the current time instant, and o_t is the output at the current time instant. Therefore the hidden state s_t can be

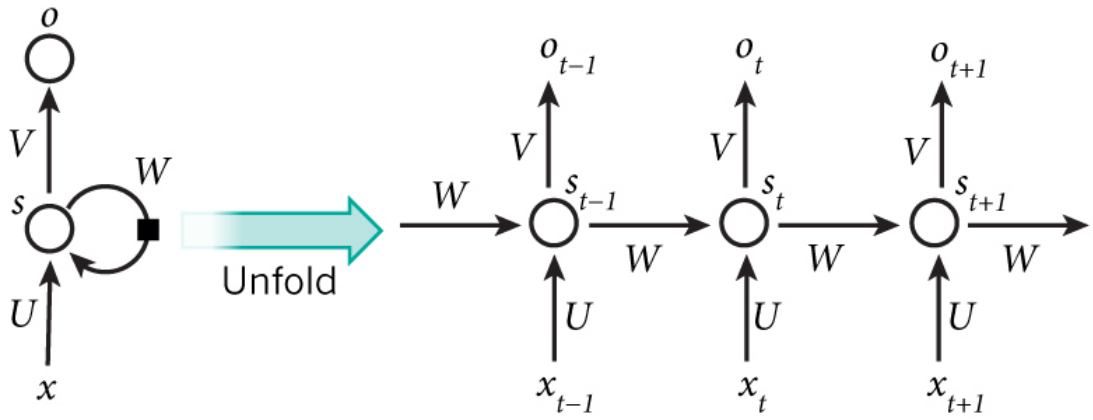

Figure 2.16: Unrolled RNN model [23]

calculated as the following:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{2.27}$$

Where f is the activation function which usually is a non-linear function such as $tanh$ or $ReLU$. The output o_t is therefore governed by the following equation:

$$o_t = softmax(Vs_t) \tag{2.28}$$

$softmax$ function normalises the output value to the range of $[0, 1]$ and it can be inverse-transformed to give real-life interpretation. Although a simple RNN model is thought to be a novel model that incorporates past inputs into learning, however it suffers from exploding gradient and vanishing gradient problems if no amendments are made to the model.

Exploding and Vanishing Gradients

Using the unrolled diagram of Figure 2.16, we can formulate the Back Propagation Through Time (BPTT) as the following:

$$E_t(o_t, \hat{o}_t) = -o_t \log \hat{o}_t \tag{2.29}$$

The above equation calculates the error or cross entropy loss at time instant t. o denotes the target value and $\hat{o}$ denotes the hypothesis value. The overall error of

the RNN model is therefore:

$$E(o, \hat{o}) = \sum_{t} E(o_t, \hat{o_t}) \tag{2.30}$$

According to the back propagation defined in Equation 2.25, the derivative of error at time instant n with respect to U, V, W can then be represented as the following:

$$\frac{\partial E_n}{\partial V} = \frac{\partial E_n}{\partial \hat{o_n}} \frac{\partial \hat{o_n}}{\partial V} \tag{2.31}$$

According to Equation 2.28 and 2.29, this derivative can be computed using parameters from the current time instant only. However this is not the case for both $\frac{\partial E_n}{\partial U}$ and $\frac{\partial E_n}{\partial W}$, because function s is related to all current and previous U, W as shown in Equation 2.27. This can therefore be shown as the following:

$$\frac{\partial E_n}{\partial W} = \frac{\partial E_n}{\partial \hat{o_n}} \frac{\partial \hat{o_n}}{\partial s_n} \frac{\partial s_n}{\partial W} \tag{2.32}$$

It should be noted that s_n is related to all past s_n, therefore chain rule needs to be applied to that partial derivative term, thus the final form is obtained as the following:

$$\frac{\partial E_n}{\partial W} = \sum_{k=1}^{n} \frac{\partial E_n}{\partial \hat{o_n}} \frac{\partial \hat{o_n}}{\partial s_n} \frac{\partial s_n}{\partial s_k} \frac{\partial s_k}{\partial W} \tag{2.33}$$

$$\frac{\partial E_n}{\partial U} = \sum_{k=1}^{n} \frac{\partial E_n}{\partial \hat{o_n}} \frac{\partial \hat{o_n}}{\partial s_n} \frac{\partial s_n}{\partial s_k} \frac{\partial s_k}{\partial U} \tag{2.34}$$

In [24], Pascanu *et al* pointed out that Equation 2.33 and 2.34 could either be extremely large (exploding) or small (vanishing). Fortunately, exploding gradient problem can be detected easily as the outputs or hypothesis will simply be NaN. One simple but effective way to combat this challenge is called gradient clipping [24], which basically limits the gradient to a pre-defined threshold. On the other hand, it is also possible for any one component of these gradient terms to approach 0 and creates a vanishing gradient problem. If weights for one of the neuron to be 0, it then drives all previous layers' gradients to be 0 and this will cause the neural network model unable to learn the long-term dependencies. Unfortunately this is a lot more difficult to solve, new architectures and thinkings are needed to address this challenge.

Long Short Term Memory (LSTM)

LSTM is introduced in [25] in 1997 as one of the early attempt to address the vanishing gradient problem. It has gained tremendous popularity after the introduction, and it is probably the most classic RNN model used for time-series prediction. Different from the traditional RNN model, in which the neuron only includes one activation function such as *tanh*, LSTM has introduced additional components in the neuron to produce a much more plausible result. The LSTM architecture has forget, input, and output gates which assigns weights to the respective parameters, and this gives an indication of how important historic data is in comparison to the output. An overview of the LSTM architecture is given in Figure 2.17

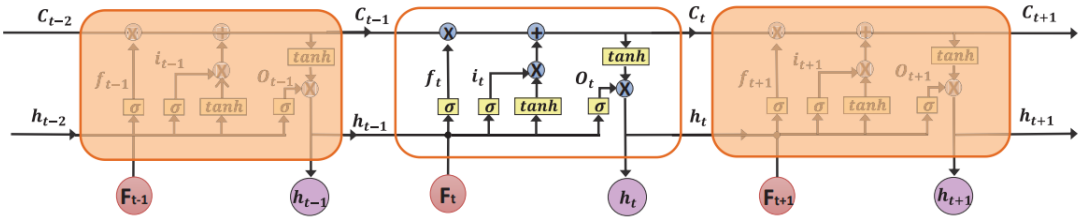

Figure 2.17: LSTM architecture [26]

In figure 2.17, σ denotes a sigmoid function that outputs a value between 0 and 1, c is the memory or cell state of the neuron, F is the input at a certain time instant,each rectangular block is a neuron in the neural network model and h is the hypothesis or output. Based on the above diagram the following governing equations of gates of LSTM can then be obtained:

$$f_t = \sigma(W_f \cdot [h_{t-1}, F_t] + b_f) \tag{2.35}$$

Where f is the output value of forget gate, W_f is the weight vector of forget gate, and b is the constant term. It is possible that certain past inputs are no longer relevant to the current output, therefore forget gate is used to enable the network to forget such past inputs by attenuating its value. The new cell state, which is modified by input gate, is updated according to the following rules:

$$i_t = \sigma(W_i \cdot [h_{t-1}, F_t] + b_i) \tag{2.36}$$

$$C_t = f_t * C_{t-1} + i_t * tanh(W_c \cdot [h_{t-1}, F_t] + b_c) \tag{2.37}$$

Where i_t is the input gate. Finally the output, which is modified by output gate, are obtained by the following equations:

$$O_t = \sigma(W_o \cdot [h_{t-1}, F_t] + b_o) \tag{2.38}$$

$$h_t = O_t * tanh(C_t) \tag{2.39}$$

Where O_t represents the output gate and weights of different gates are tuned using back propagation. Based on the above equations, it can be observed that LSTM introduced a few more hyper-parameters such as weights of different gates, and this addition allows the network to establish a more accurate model for data with complex long-term dependencies. Furthermore, LSTM is also able to prevent the vanishing gradient problem as it selectively memories the useful information from the previous state, which means multiplication of many derivative terms in back-propagation, such as that of Equation 2.33, is avoided because only the last state is required in the process of back propagation. However this accuracy is achieved at the cost of complexity, the computational time of LSTM model is usually much longer in comparison to a simpler architecture.

Gated Recurrent Units (GRU)

GRU is a LSTM-based model proposed by Cho *el al.* in 2014 [27]. It basically aggregates the forget and input gates into a update gate. This modification simplifies the original LSTM architecture, and therefore it has proven to be less computationally costly during training. Apart from the low complexity, empirical evidence also suggests that it is able to achieve similar performance to LSTM [28]. The overall architecture of GRU is illustrated in Figure 2.18. The governing equations of a GRU unit is then:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{2.40}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2.41}$$

$$\hat{h_t} = tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{2.42}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h_t} \tag{2.43}$$

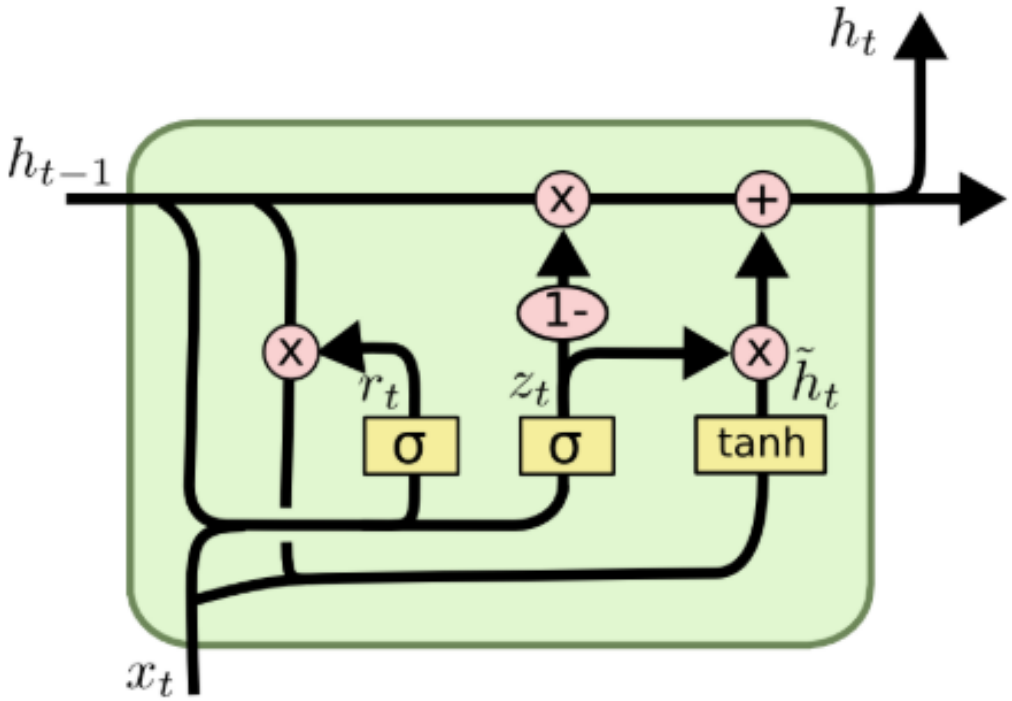

Figure 2.18: GRU architecture [29]

Deep Learning

The traditional learning techniques mentioned above requires the prior knowledge of the features suitable for the model to perform a certain task. For example, if the neural network model is to perform a task of predicting the price of a house based on certain parameters, these parameters or features (area, location, etc.) have to be manually determined before feeding into the networks, and this process is called feature engineering. However, it is possible for a certain task to have many features or features that can not be identified easily, and such challenge could servilely degrades the performance of the neural networks. Deep learning was then proposed to address this challenge by allowing the neural network to extract the features automatically, and this is achieved by stacking multiple layers of neural networks together to form a deep or hierarchical architecture [30]. An example of deep learning architecture is represented in Figure 2.19.

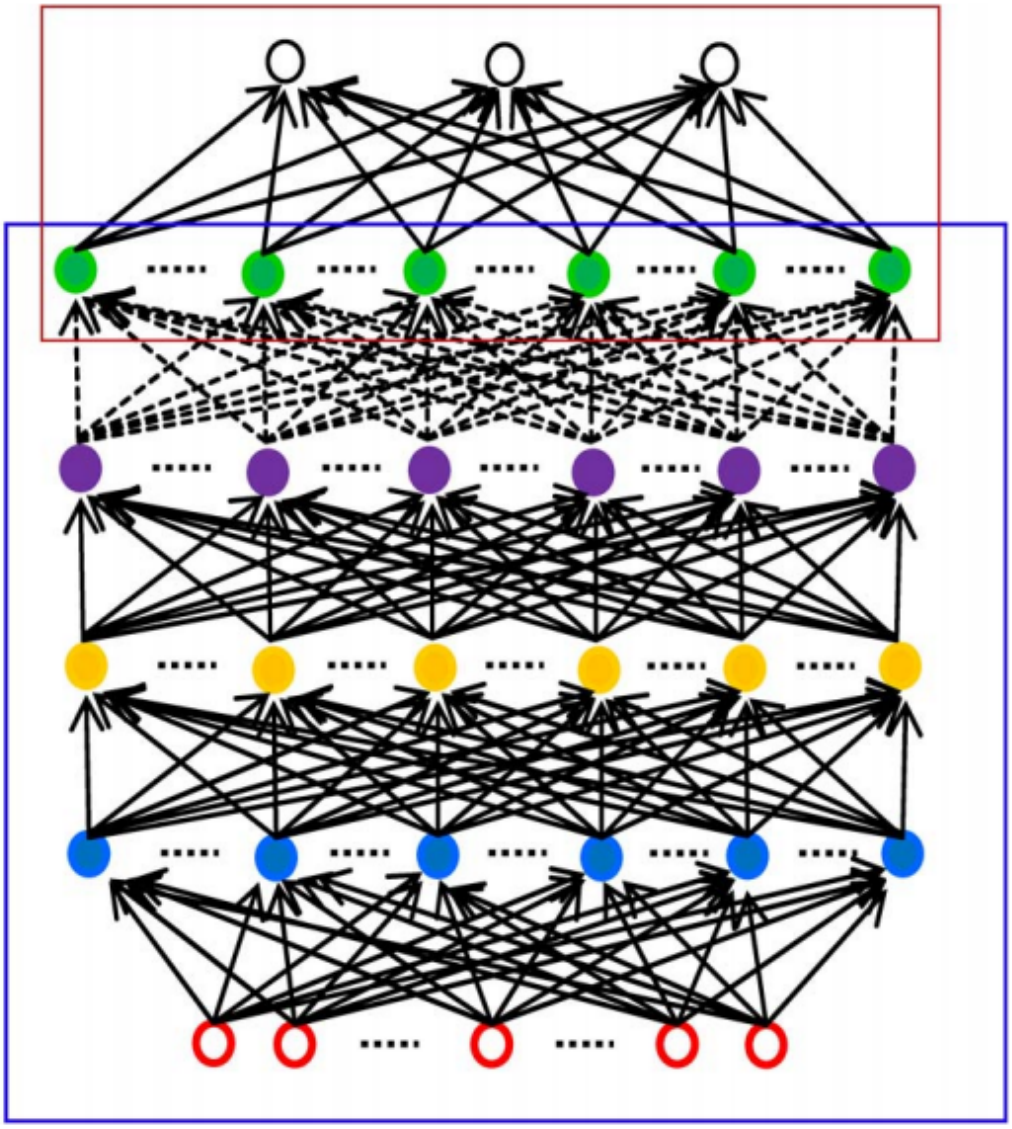

Figure 2.19: Deep learning architecture [30]

In addition to feature extraction, deep learning models are also superior in comparison to shallow learning in scalability. Deep learning models benefit significantly from training on big data which is now available thanks to the advancement in data science and storage technologies. Performance of traditional

shallow neural networks converge, which means they do not benefit from additional data in the training set, whereas deep learning neural networks scale well with increase in size of the training set, and such relationship is shown in Figure 2.20. Due to the above mentioned advantages, deep learning has really taken off as one of the most popular architecture or technique when supervised learning is required.

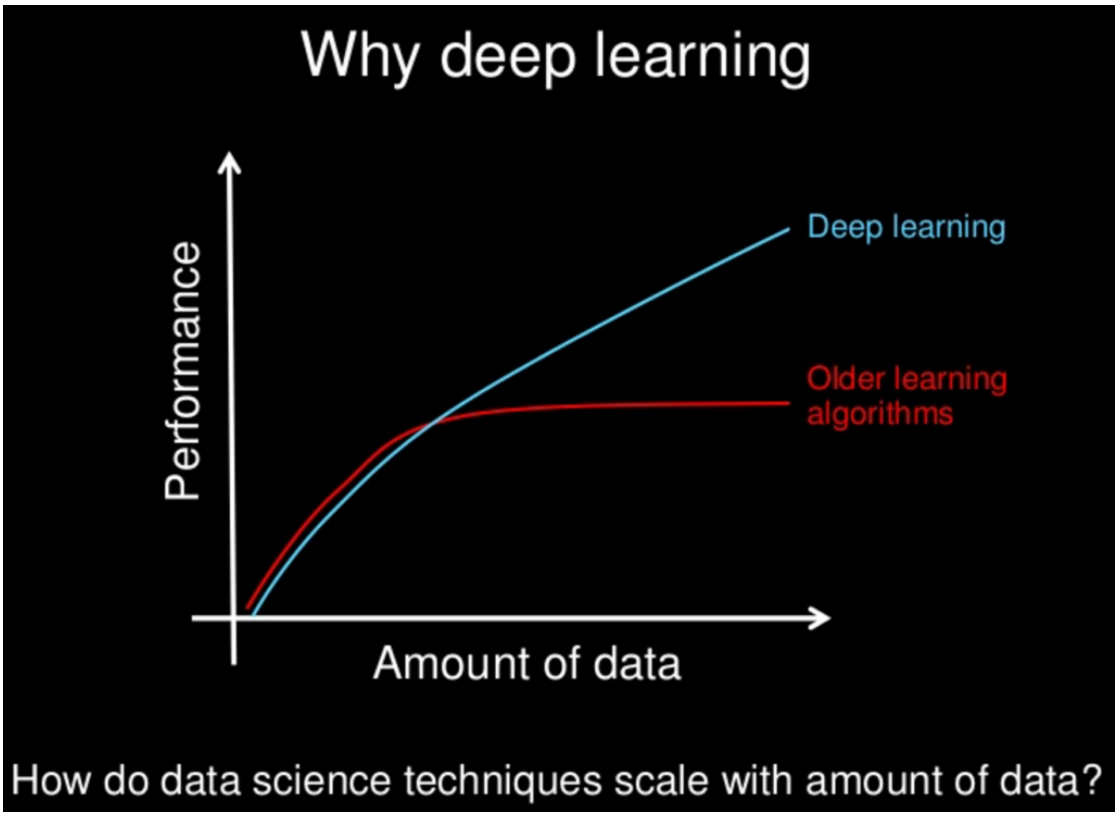

Figure 2.20: Deep learning scalability [31]

2.4 Clustering Methods

The objective of clustering is to identify elements with similar features of interest and group them accordingly. Machine learning techniques are the most popular approaches to deal with such problems. This section gives an overview of a few widely adopted machine learning techniques in the context of clustering problems.

2.4.1 Expectation Maximisation(EM)

EM is a widely adopted unsupervised learning algorithm that aims to cluster data based on their characteristics. This method can be divided into two steps, namely expectation and maximisation. The expectation step calculates the expectation of a certain data element belonging to a certain probability distribution, and then the probability distribution parameters are updated to maximise the sum of expectation of all data elements (Maximum Likelihood). This method is repeated until convergence when the expectation can no longer be increased [32]. The logic of EM method is shown in Figure 2.21

Gaussian Mixture Models

This section gives an overview of the Gaussian Mixture Model (GMM) which is the most popular probability density function or model used in EM method [32]. In this model, we assume that there are k Gaussian sources and each is associated with a certain mean μ and standard deviation σ. Each Gaussian source can be represented as in Equation 2.44.

$$N(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{\sigma^2}} \tag{2.44}$$

The linear combination of k Gaussian sources can then be represented as in Equation 2.45.

$$p(x|\mu, \sigma) = \sum_{i=1}^{k} \pi_i N(x, \mu_i, \sigma_i) \tag{2.45}$$

It should be noted that μ and σ denotes vectors of means and standard deviations of the k Gaussian sources. π is the hidden variable or weight of each Gaussian source. Assume that k clusters are formed to classify the given data points, then the probability, r_{ic}, of the i^{th} data point belonging to a certain Gaussian source c is then computed as in Equation 2.46, and this step is called **expectation**:

$$r_{ic} = \frac{\pi_c N(x_i|\mu_c, \sigma_c)}{\sum_{j=1}^{k} \pi_j N(x_i|\mu_j, \sigma_j)} \tag{2.46}$$

It should be noted that all parameters such as mean, standard deviations are treated as constants during the expectation steps, the parameters are then updated

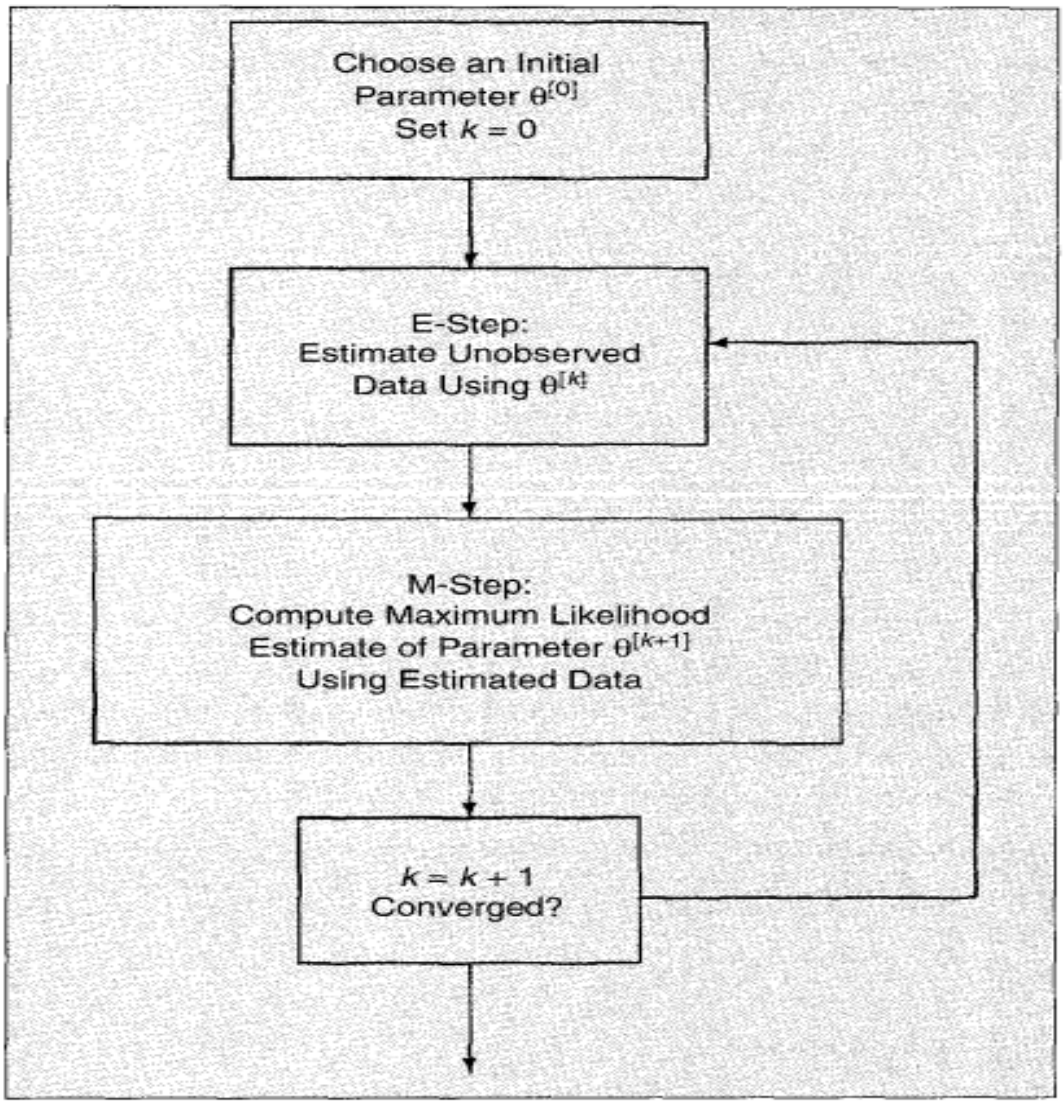

Figure 2.21: Logic of EM method [32]

to maximise the sum of log likelihood of all data points belonging to the Gaussian sources, and the update equations of the **maximisation** step are shown in Equations 2.47, 2.48, 2.49, 2.49 :

$$\mu_c^{new} = \frac{1}{N_c} \sum_i r_{ic} x_i \tag{2.47}$$

$$N_c = \sum_i r_{ic} \tag{2.48}$$

$$\sigma_c^{new} = \frac{1}{N_c} \sum_i r_{ic}(x_i - \mu_c^{new})^2 \tag{2.49}$$

$$\pi_c = \frac{N_c}{n} \tag{2.50}$$

Where n is the total number of data points. These update procedures are repeated until the Gaussian sources are no longer altered due to change in the membership of each Gaussian source, and this procedure produces a soft membership as each data point belongs to a Gaussian source with a certain probability.

2.4.2 K-Means Clustering

K-means clustering can be considered as a variant of the EM method mentioned above, as its procedure is quite similar to EM method. K-means clustering also has two distinctive phases, the first phase assigns all data elements to the centroid or clusters closest to it, after which it calculates the total aggregate distance of all members belonging to a centroid, and the second phase re-locates the centroid according to the locations of its members. Assume that a n dimensional data set X is to be classified into k clusters denoted by K. The aggregate distance is then calculated as in Equation 2.51 [33]:

$$e = \sum_{m=1}^{k} \sum_{X \in K_m} ||X - K_m|| \tag{2.51}$$

It should be noted that all k centroids are initialised at random locations, therefore K-means algorithm is inherently non-deterministic if the problem is not convex. Furthermore, different kinds of distances, such as Euclidean distance, Manhattan distance and etc. can also be used in the calculation, and this will play an important role in the outcome of the algorithm. The locations of the centroids are then updated by computing the mean value of the positions of its l members, and this is represented in Equation 2.52:

$$K_m = (\frac{\sum_{i=1}^{l} x_{i1}}{l}, \frac{\sum_{i=1}^{l} x_{i2}}{l} \dots \frac{\sum_{i=1}^{l} x_{in}}{l}) \tag{2.52}$$

These two steps are repeated until no membership change occurs as a result of the updated centroid locations, and at this point the algorithm is declared to be converged. One of the challenges of implementing K-means and EM method is the prior knowledge of the number of clusters needed. Furthermore, these algorithms also do not produce global optimal solutions as they are non-deterministic in non-convex problems, however these problems can usually be solved to a satisfactory degree with heuristic methods.

Elbow Method

The elbow method is a heuristic method used to determine the number of suitable clusters for the above mentioned clustering algorithms [34]. In this method, the distortion of the output is compared against the number of clusters. A distortion could be defined in the form of aggregate distance, inverse likelihood, and etc. In essence, distortion is inversely related to the performance of the clustering algorithm. It should be noted that it is possible for distortion to reach a value of 0 when the number of clusters is equal to the size of the data set, however this is not practical in real life. Thus, the number of clusters is heuristically chosen when an increase in number of clusters does not result in a significant decrease in distortion. An example of the elbow method is shown in Figure 2.22

The Silhouette Method

This provides a more analytical method to determine the suitable number of clusters. This method basically measures how well a point is assigned to a cluster by measuring its similarity to its cluster against the dissimilarity against other clusters. The Silhouette index s is governed by the following equation:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{2.53}$$

$a(i)$ is the average distance between i^{th} data element to all the other elements that belong to the same cluster of data element i. $b(i)$ is the smallest average distance between data element i to all the other clusters that data element i is a not a member of. It can be observed that $s(i)$ is a number between -1 and 1, if $s(i)$ is equal to 1, then it is theoretically perfectly clustered, and the reverse is also true at the other extreme value -1. By taking the average of $s(i)$ of the entire

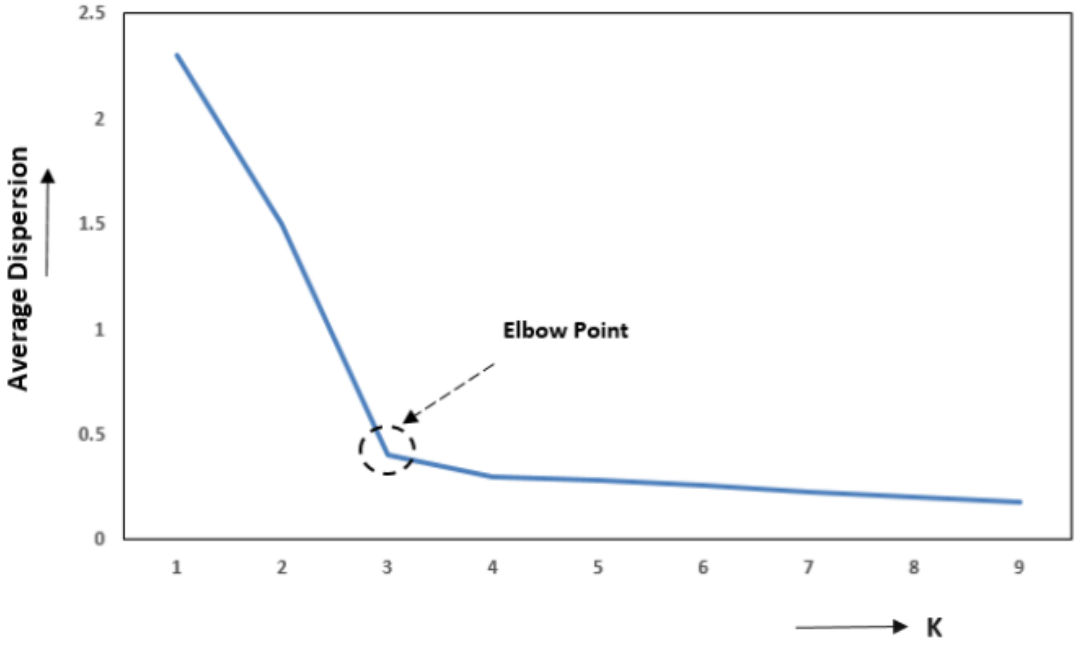

Figure 2.22: Illustration of elbow method [34]

dataset, it is then possible to obtain an indication of how well the clustering is performed. If an inappropriate number of clusters are chosen then it is possible that certain groups of datums may have a low Silhouette index in comparison to others, therefore the number of clusters should then be altered.

2.5 Current BBU-RRH Switching Schemes

Based on the C-RAN model described in Section 2.14, a user can only get access to the network if he is connected to one of the RRHs that are associated with a BBU within the BBU pool, it is then obvious that this access problem can essentially be modelled as two separate components. One component is the association between users and RRH, while the other is the association between RRHs and BBUs within the BBU pool. It should be noted that RRHs that are associated with a BBU may need be switched over to other BBUs when there is insufficient capacity at the original BBU, therefore a switching algorithm needs to be developed to perform this task efficiently. This section only gives an overview of some of the BBU-RRH

switching schemes available at the time of writing as the association or switching between users and RRHs is not of interest in this study.

2.5.1 Semi-Static and Adaptive Switching Schemes

Namba *et al.* have proposed a semi-static and an adaptive algorithm in [35] and this work is considered as the basis of many other switching algorithms. In addition to the algorithms they have contributed, they also outlined two widely adopted principals in the switching algorithm design. The principals introduced are as follows:

- BBU reduction: The number of active BBUs in the BBU pool should be as few as possible after the switching process
- BBU aggregation: Neighbouring RRHs should be assigned to the same BBU if possible, as this will facilitate the Coordinated Multi-Point transmission or reception (CoMP).

Furthermore, the author also abstracted the resource available at a certain BBU to a number between 0 and 1, where 0 denotes inactivity or fully empty and 1 denotes fully utilised. The author also established two thresholds on the capacity of a BBU, namely upper threshold and lower threshold. If the BBU is operating above the upper threshold then a congestion occurs, and if the BBU is operating below the lower threshold then the BBU is under-utilised and should be shut down.

Semi-Static Switching Scheme

This algorithm associates and switches based on the peak traffic load of all RRHs in a relatively long time interval (24 hours in their simulation). This algorithm will iterate over every RRH and attempt to assign all RRHs to their neighbours' BBUs if the destination BBU is able to accommodate the incoming RRH and operate under the pre-defined upper threshold. The pseudo code of the algorithm is shown in Algorithm 1.

RRH$_i$: i^{th} RRH $(1 \leq i \leq I)$
BBU$_j$: j^{th} BBU $(1 \leq j \leq J)$
$R(RRH_i), R(BBU_j)$: Required resource of RRH_i or used resources in BBU_j
$j = 0$
for $i = 0; i \leq I; i{+}{+}$ **do**
  if *RRH_i is not associated with a BBU* **then**
    insert_queue(RRH$_i$) //Insert RRH$_i$ in Queue
  end
  while *Queue is not empty* **do**
    RRH$_t$ = search_RRH(); //Find RRH with the most required resource in queue
    if $R(BBU_j) + R(RRH_t) \leq UpperLimit$ **then**
      BBU_assignment(RRH_t, BBU_j); // RRH$_t$ is assigned to BBU$_j$
      $R(BBU_j) += R(RRH_t)$; //update resources of BBU$_j$;
      insert_que(neighbour RRHs of RRH$_t$); // insert unassociated neighbours of RRH$_t$
    end
    remove_queue(RRH$_t$); //Remove RRH$_t$ from the queue
  end
  j++;
end

Algorithm 1: Semi-Static Switching Algorithm [35]

Adaptive Switching Scheme

The authors have used a much shorter switching time interval for this algorithm (1 hour in their simulation), and a switch will occur at every time interval if the BBU usage is above its upper threshold (Case A) or below its lower threshold (Case B). For both cases, the algorithm will attempt to switch RRHs, which have either most neighbour or least traffic load, to other BBUs so that congested BBU can operate below upper threshold and under-utilised BBUs can be shut down. The RRH is switched to BBUs with sufficient capacity in the following order of preference:

- BBUs with most neighbours of the switching RRH

- Active BBUs with sufficient capacity
- Inactive BBUs

The overview of this process can be represented in Figure 2.23 and Figure 2.24.

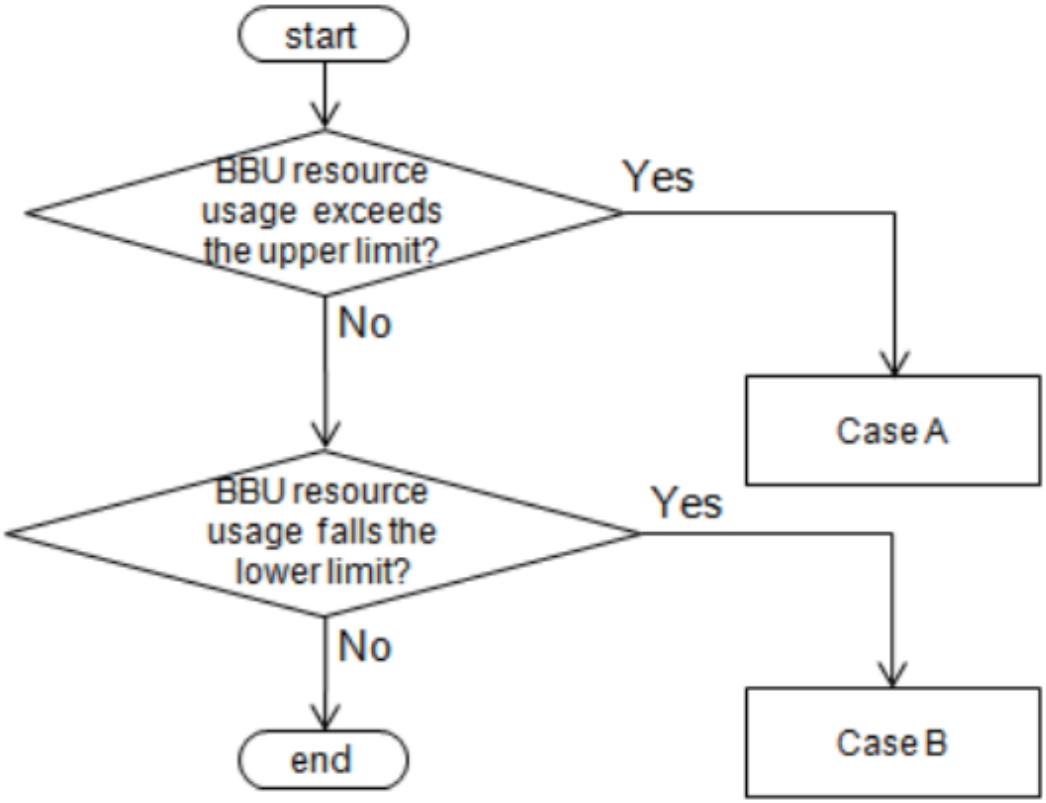

Figure 2.23: Adaptive switching scheme overview [35]

Results & Discussion

The paper [35] has simulated the performance of these two algorithms according to the parameters in Figure 2.25. It should be noted that the traffic profile used is the same as the one shown in Figure 2.5 The results have shown that they BBU usage is significantly reduced in comparison to the traditional cell deployment which would require 100 BBU to function. Semi-static algorithm reduces the BBU usage by 26% while adaptive scheme reduces the usage by 47%. The simulation result

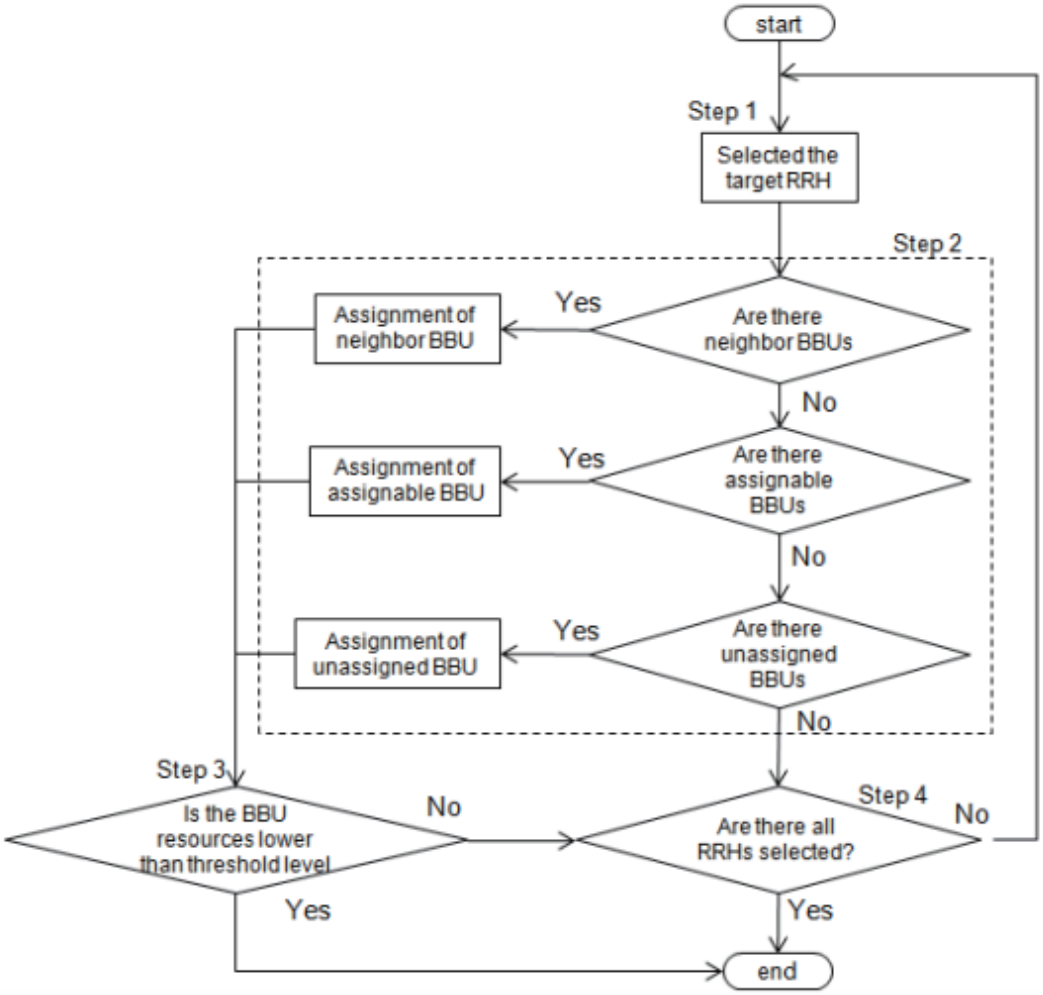

Figure 2.24: BBU-RRH switching logic for case A [35]

is shown in Figure 2.26. Based on the results given, the advantages of the two algorithms can be listed as follows:

- Semi-static algorithm achieves moderate BBU reduction while adaptive algorithm reduces the BBU usage significantly
- Both algorithms adhere to the BBU aggregation principal when possible
- Semi-static algorithm has a low complexity in theory
- Adaptive algorithm can achieve BBU reduction close to optimum level at

Number of RRHs	100
Coverage area of a RRH	Hexagonal layout
Target time	24 hours
Traffic profiles	a) office, b) residence
Time interval of BBU-RRH switching for semi-static and adaptive scheme	24 hours for semi-static 1 hour for adaptive
Upper limit for BBU resource	0.9
Lower limit for BBU resource	0.5

Figure 2.25: Simulation parameters [35]

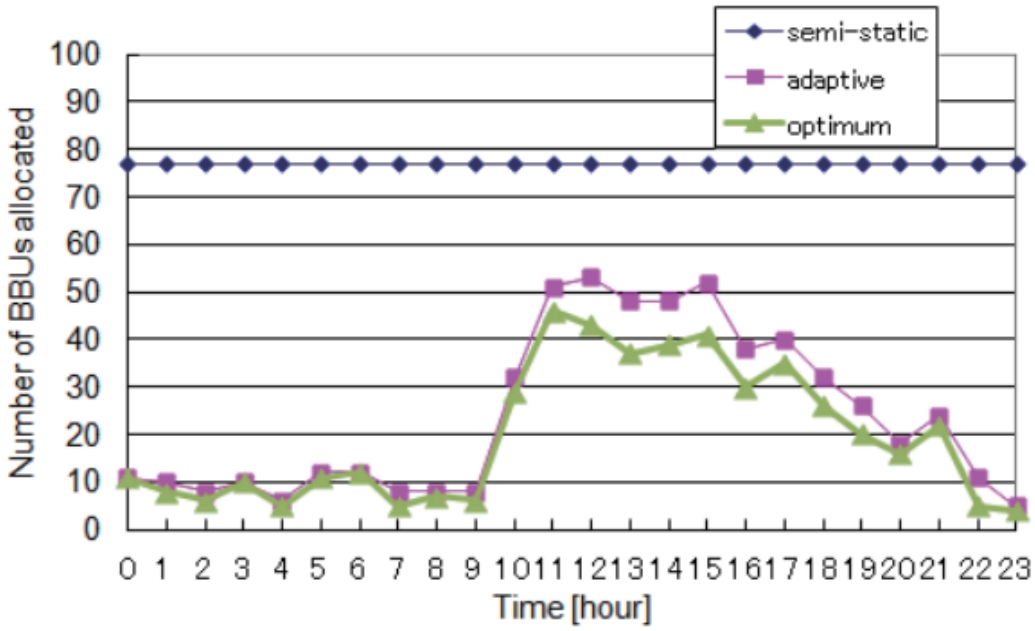

Figure 2.26: Simulation results for an office area [35]

certain time instants

Despite the advantages mentioned above, these algorithms also have a few shortcomings listed below:

- Semi-static algorithm has a long switching time interval, and it allocates according to the peak traffic in the traffic profile, which makes it relatively inefficient.
- Both algorithms do not consider the continuous nature of traffic generation as they work with snapshots of networks on a hourly or 24-hourly basis. It is possible that, especially in the adaptive scheme case, the resource allocated may not be sufficient very shortly after the allocation during the period of rising traffic, and also the resource allocated may become inefficient during the period of falling traffic.

2.5.2 Optimisation Approaches

Different from the above mentioned heuristic methods, some authors have taken an optimisation approach to this problem. In these work, the authors identified parameters that they would like to maximise against some of the physical constraints, and the optimisation framework is usually established based on different angles of the problem. Chen *et al* [36] have proposed an optimisation framework that aims to select a maximum number of BBUs that need to be switched to a minimum number of BBUs if the BBU is operating above the upper threshold. Ha *et al* [37] has proposed an optimisation framework to minimise the total transmission power in C-RAN. Guo *et al* [38] has proposed another optimisation framework that jointly minimises the total energy consumption and number of active BBUs within C-RAN.

Evaluation

The simulation of these algorithms have all shown that they are able to achieve their objectives under certain constraints. However, not all results of interest are shown in the papers, for example, [36] and [37] do not show the level of BBU reduction of the algorithms. It should also be noted that these switching

schemes all demonstrate some sort of trade-off amongst their parameters and complexity. As these algorithms all work with snapshots of network without taking into consideration of the continuous nature of call arrival and traffic. It is possible that these complex algorithms may need to be performed many times due to reasons mentioned in Section 2.5.1. This would then increase the difficulty of implementation as these tasks may need to be finished within a short time interval. Furthermore, frequent switching could incur high signalling cost within the network. Thus it is argued that these algorithms may need additional mechanism to reduce the complexity and become more traffic aware.

Chapter 3

Dataset

An open big telecommunication dataset released by the Telecom Italia [39] is studied to obtain more practical insight on the traffic profiles of base stations within a city. This section gives an overview of the compositions of this dataset.

3.1 Master Dataset Overview

This dataset includes the Call Detail Records (CDR) of different areas in the vicinity of Milan, Italy, and these CDRs record the traffic generated by SMS, voice and data from November 2013 to December 2013 (22 holidays and 40 workdays). This dataset divides the area of interest into 1000 equal squares and each square has a dimension of 235 metres by 235 metres. Figure 3.1 shows how the area of interest is divided, and Figure 3.2 shows the area of interest in the map.

This dataset contains the following fields:

- Cell ID: The ID number ranging from 1 to 10000 shown in Figure 3.1.
- Time Stamp: The amount of time elapsed in milliseconds since 1 January, 1970 at UTC. The granularity of the time record is 10 minutes in the dataset.
- Country Code: The country code of Italy
- SMS Reception: The SMS reception activity within the square during the time interval, and the country code of the sender is also recorded

9901	9902	9903	...	...	...	...	...	...	10000
9801	9802	9803	...	...	...	...	...	...	9900
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
101	102	103	...	...	...	...	...	...	200
1	2	3	...	...	...	...	...	...	100

Figure 3.1: Area division of the dataset

- SMS Sending: The SMS sending activity within the square during the time interval, and the country code of the recipient is also recorded
- Call Reception:The call reception activity within the square during the time interval, and the country code of the caller is also recorded
- Outgoing Call:The Outgoing call within the square during the time interval, and the country code of the callee is also recorded
- Internet Traffic: The record of traffic usage within the square during the time interval, each CDR is generated if (a) A user initiates a connection or (b) A user ends a connection or (c) 15 minutes have elapsed since last CDR or 5MB is used since the last CDR

To simplify the model, only internet traffic is used in the studies as it consumes significantly more data in comparison to other services. It should also be noted that all values in the dataset are scaled to avoid possible privacy breach. The scaling factor is estimated to be proportional to 1000000 according to [40]

3.2 Derived Base Station Dataset

Based on the methods and dataset provided by the author in [40], it is possible to approximate geographical locations and traffic volume of each base station within the area of interest based on the open big dataset. The geographical locations of

Figure 3.2: The area of interest shown in map

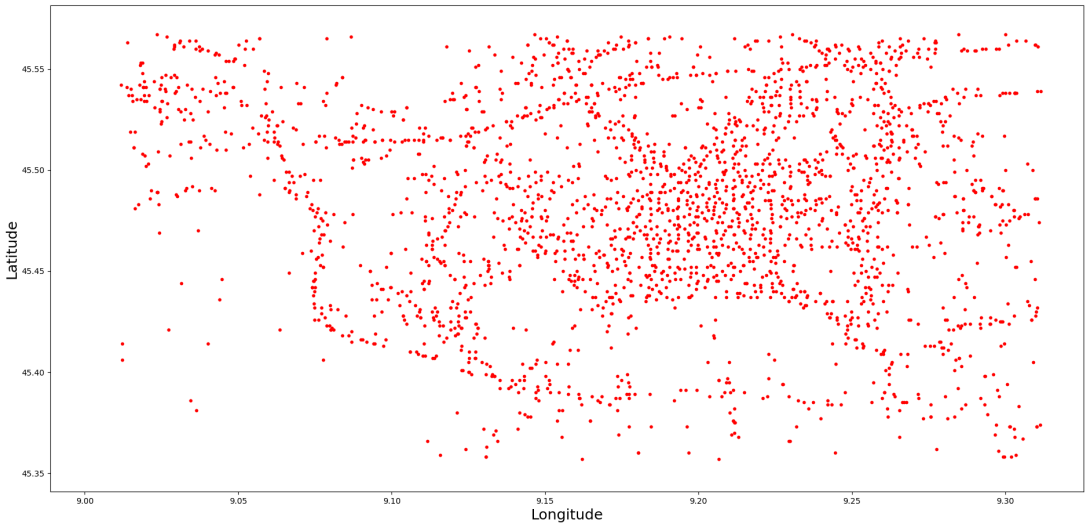

Figure 3.3: The geographical distribution of base stations

the base stations are represented in Figure 3.3. This area is estimated to have 2554 base stations in total, and the traffic of each base station is recorded on an hourly basis. This results in 62 matrices of the dimensionality of $2554X24$, where each matrix represents the traffic for all base stations in one day. A snapshot of the given dataset is given in Figure 3.4.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	9.23	45.536	180.56	153.52	144.41	119.6	118.35	118.57	116.61	153.95	248.23	271.77	238	322.82	275.16	322.39
2	9.223	45.508	276.65	245.58	211.44	191.69	158.07	153	144.13	196.58	208.85	233.12	286.01	360.27	302.92	247.03
3	9.2086	45.499	138.76	59.212	54.741	44.36	48.753	34.718	51.669	48.715	80.906	85.493	121.03	89.498	69.631	68.776
4	9.2154	45.507	379.55	371.29	310.74	335.67	284.37	192.88	189.56	252.43	254.85	232.94	330.62	301.38	317.09	268.67
5	9.2128	45.499	140.45	62.635	58.151	47	50.999	37.997	53.796	50.864	81.842	89.447	123.21	92.047	76.947	73.542
6	9.2108	45.519	162.91	101.45	99.792	80.164	70.44	77.221	71.522	89.329	89.369	103.67	158.55	124.42	112.97	122.97
7	9.2171	45.533	156.39	84.079	55.684	53.854	51.283	51.008	101.95	65.978	129.12	238.34	244.92	291.07	305	309.27
8	9.2267	45.528	123.24	169.52	122.59	98.429	72.291	69.225	69.486	79.974	87.708	116.23	116.85	123.61	106.89	119.82
9	9.1787	45.474	454.18	406.44	479.26	316.11	249.99	150.31	180.09	113.54	140.06	217.14	232.37	276.46	321.96	274.62
10	9.1832	45.468	674.46	660.28	1028.4	607.58	968.8	689.4	300.66	360.96	379.4	712.82	578.41	608.14	866.58	884.62
11	9.1762	45.466	445.08	337.7	305.9	356.74	306.59	237.32	244.14	251.15	265.7	341.45	328.6	400.29	383.03	552.48
12	9.1846	45.471	1365.3	1200.3	833.15	823.61	656.64	654.06	656.59	674.91	1068	1338.8	1541.4	1408.1	1764.9	1739.3
13	9.183	45.466	453.71	465.66	490.7	375.01	246.09	194.24	190.69	190.42	173.13	251.76	295.34	299.48	374.8	483.32
14	9.2254	45.515	668.65	440.32	365.68	312.32	327.46	291.98	581.82	1221.4	620.03	468.9	530.96	595.74	593.18	568.1
15	9.1844	45.481	1621.5	1599.4	1713.4	1229.3	976.71	603.8	518.45	609.95	693.16	980.88	1110.5	1272.7	1361.6	1348.4
16	9.1881	45.479	796.64	643.72	479.42	309.21	362.17	247.57	242.32	271.71	395.05	427.38	523.39	642.26	615.09	513.01
17	9.1917	45.48	458.1	367.07	395.85	224.29	243.64	270.75	270.23	363.12	383.01	468.42	404.98	465.41	562.41	492.98
18	9.1971	45.481	1092	282.33	207.21	191.8	164.03	527.28	254.67	307.23	243.75	317.41	391.94	492.25	315.71	347.14
19	9.2293	45.524	199.09	154.16	161.83	117.68	85.086	81.238	97.435	147.28	134.62	259.47	180.6	180.1	243.63	190.86
20	9.1308	45.358	59.3	54.355	53.217	41.034	41.519	46.609	49.109	106.81	60.928	74.183	78.887	95.429	326.25	83.388
21	9.2151	45.561	506.17	368.11	307.85	287.18	280.24	286	309.74	401.27	445.35	565.05	602.87	682.1	611.48	593.24
22	9.219	45.564	207.27	175.29	139.11	130.01	123.49	135.27	147.32	173.42	280.92	325.41	335.29	339.15	351.7	398.68
23	9.252	45.445	212.86	194.43	181.93	176.73	144.24	167.32	203.93	226.26	224.46	345.64	259.71	347.02	308.17	290.44

Figure 3.4: Snapshot of the derived base station dataset

Chapter 4

Clustering

Before BBU-RRH switching schemes can be developed, it is essential to identify the RRHs or base stations that will be associated with a specific BBU pool in a real life scenario, and this problem is usually overlooked or out of scope for many BBU-RRH switching schemes. This section aims to provide a solution to cluster the base stations shown in Figure 3.3 according to the geographical locations of base stations, so that the aggregate distance of all base stations to the centroid is minimised. K-means and EM methods are investigated and compared in this section. This work is published by the author in [41].

4.1 K-means Problem Formulation

All base stations of interest are denoted by a set B that contains b base stations, where $b = 2554$ in this case. The k centroids of the clusters are denoted by set K:

$$B = \{B_1, B_2, B_3 \dots B_b\} \tag{4.1}$$

$$K = \{K_1, K_2, K_3 \dots K_k\} \tag{4.2}$$

Each element in B contains (x, y) components which represent the longitude and latitude of the base station. The objective of this problem is then to find locations of k centroids so that the aggregate Euclidean distances of its members to their centroids are minimised:

$$e(K) = \sum_{m=1}^{k} \sum_{B \in K_m} ||B - K_m||^2 \tag{4.3}$$

$$\underset{K}{\text{minimize}} \quad e(K)$$

The algorithm of K-means is described in section 2.4.2.

4.2 EM Problem Formulations

All base stations are also denoted as B shown in Equation 4.1. Gaussian Mixture model is used for the EM method, and all relevant parameters and algorithms can be found in Section 2.4.1. It should be noted that EM method only produces soft membership as each member has a probability to belong to a certain Gaussian source, thus the members are hard assigned to the Gaussian source with the highest probability so that the results can be compared to K-means algorithm.

4.3 Elbow Method

As mentioned in literature review, one of the challenges of these methods is to determine a suitable number of centroids or clusters. Elbow method is used in this work to address this challenge. Distortion d, which is defined as the ratio between error e and number of clusters k is used as the evaluation metric:

$$d = \frac{e}{k} \tag{4.4}$$

Figure 4.1 shows an exponentially decaying function for the ratio between d and k for K-means clustering, which means after around 20 clusters, the performance no longer significantly improves by increasing the number of clusters. 20 clusters are also used for EM for comparison purposes.

4.4 Baseline Method

The baseline method or arbitrary grouping essentially divides the area of interest into 20 equal height rectangles, and the geometric median of the rectangle is the centroid. This method is only shown to juxtapose how much clustering methods can improve.

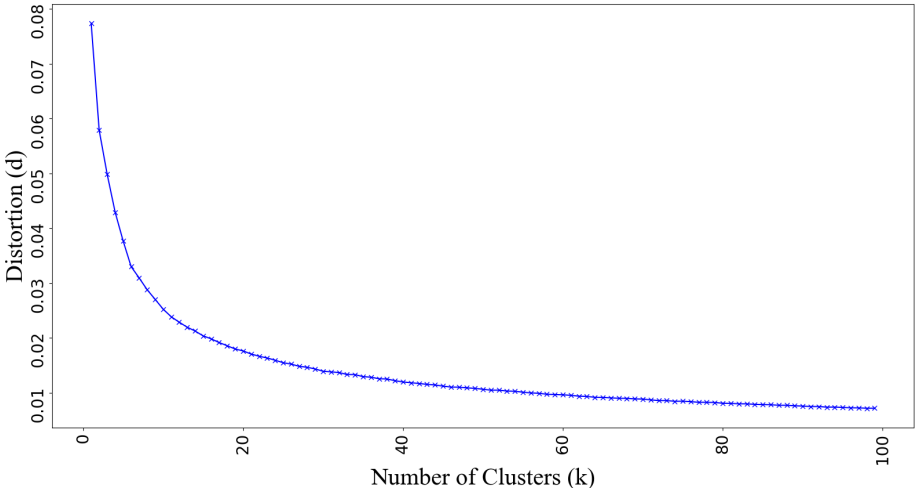

Figure 4.1: Distortion VS number of clusters for K-means

4.5 Global Minima Approximation

It is known that the clustering methods under investigation do not necessarily produce solutions that are deemed as global optimal. Some of popular heuristic methods to solve this problem include Monte Carlo initialisation, multiple starts and etc. This work proposes to establish a threshold value based on the number of iterations and previous results of the algorithm, and the global minima is approximated by comparing the obtained result to this threshold value. The threshold value is described as in Equation 4.5 [41] :

$$\hat{e}(k,\alpha,n,z) = (1 - n\frac{\alpha}{z})\frac{\sum_{i=2}^{t=k-1}\beta(i) - \beta(i-1)}{t-1} + \beta(k-1), \quad (4.5)$$
$$\beta(i) \leq \beta(i-1)$$

β is the result obtained for a given k value, $\hat{e}$ denotes the threshold that should be compared to, $\frac{\alpha}{z}$ is a normalised constant, n is the number of times the algorithm has already been executed and $\alpha = 0.01$, and this is the learning rate that determines how fast the threshold is relaxed. If the clustering algorithm is able find e less than or equal to $\hat{e}$ for a given number of clusters then the solution should be recorded, and the algorithm will move onto the next k value.

This global minima approximation method hypothesis that β will decrease at a rate of the rolling average of all previous gradients of β where positive gradients are ignored. Then it will continue to increase the threshold value $\hat{e}$ until either an acceptable solution is found or it has reached 100 iterations. The result with the smallest e is then recorded if it has been executed for 100 times.

4.6 Results & Discussion

This section will present and evaluate the results of both K-means method and EM method. Figure 4.2 is the visualisation of the clustering results of K-means algorithm when $k = 20$. Figure 4.3 is the visualisation of the EM clustering with GMM model when there are 20 Gaussian sources.

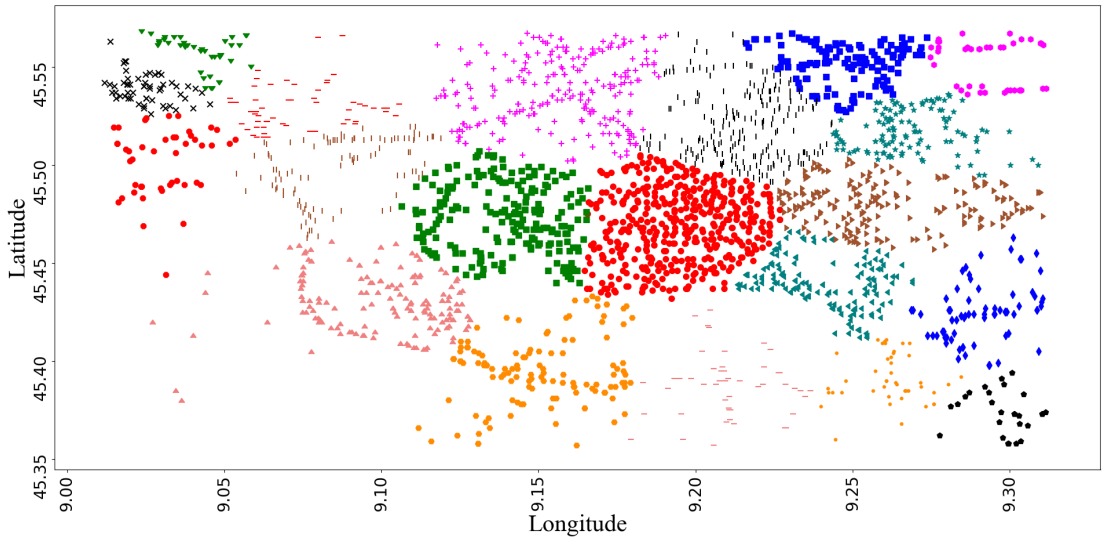

Figure 4.2: Visualisation of K-means clustering

It can be observed in Figure 4.2 and Figure 4.3 that both methods have achieved visually reasonable clustering results based on the geographical locations of the base stations. Therefore it is difficult to determine the more suitable method based on the visualisation. Thus, the aggregate distances of the members to its centroids of the two methods are compared with the baseline to determine the most suitable method for this study. As discussed previously, both methods are

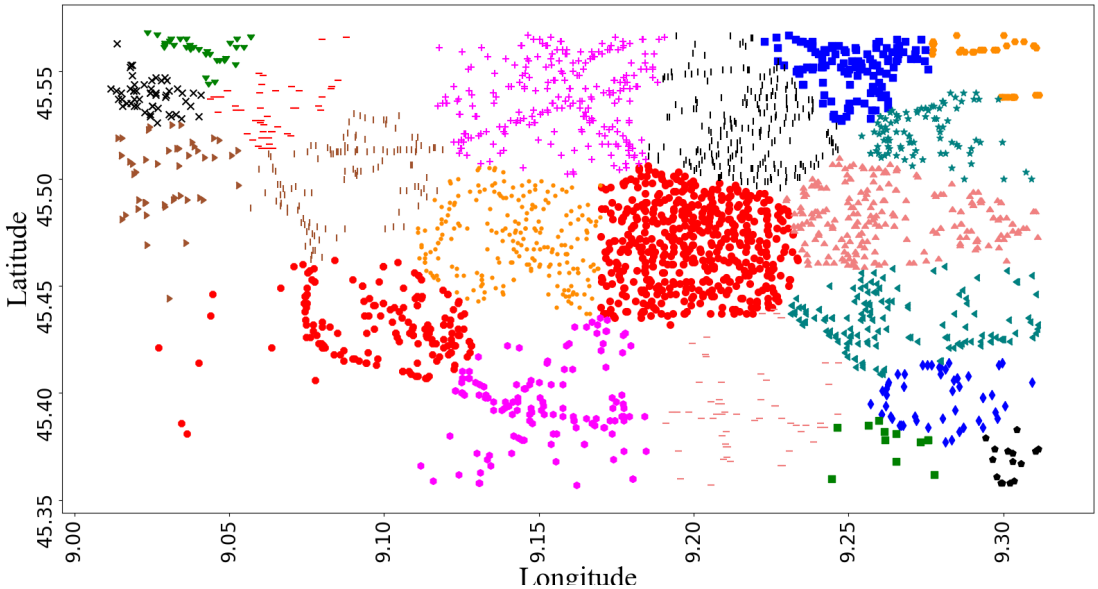

Figure 4.3: Visualisation of EM clustering with Gaussian Mixture Model

non-deterministic and highly dependent on the initialisation of the centroids, thus each algorithm is run for 50 times and the average of the performance is recorded.

Figure 4.4 shows the performance of different clustering methods investigated in the studies. It can be clearly observed that both EM and K-means methods outperformed the baseline method by a significant margin. It is evident that K-means slightly outperformed EM when no approximation is done, however the K-means solution is not exactly ideal as e is occasionally increased despite the increase in the number of clusters. Thus, the approximation is applied to K-means and it can be observed that this technique clearly enhances the performance of the clustering algorithm at the cost of computation complexity. The reason why K-means outperforms EM method is that K-means method is more biased towards a circular cluster while EM method is more biased towards an ecliptic cluster. Furthermore, Euclidean distance is a clustering metric in K-means while it is not the case for EM method, and all these features of K-means enable it to outperform EM in geographical location clustering. Each cluster is assumed to be served by one BBU pool and each base station can be viewed as a RRH in the context of study. K-means method is therefore the method of choice in this study.

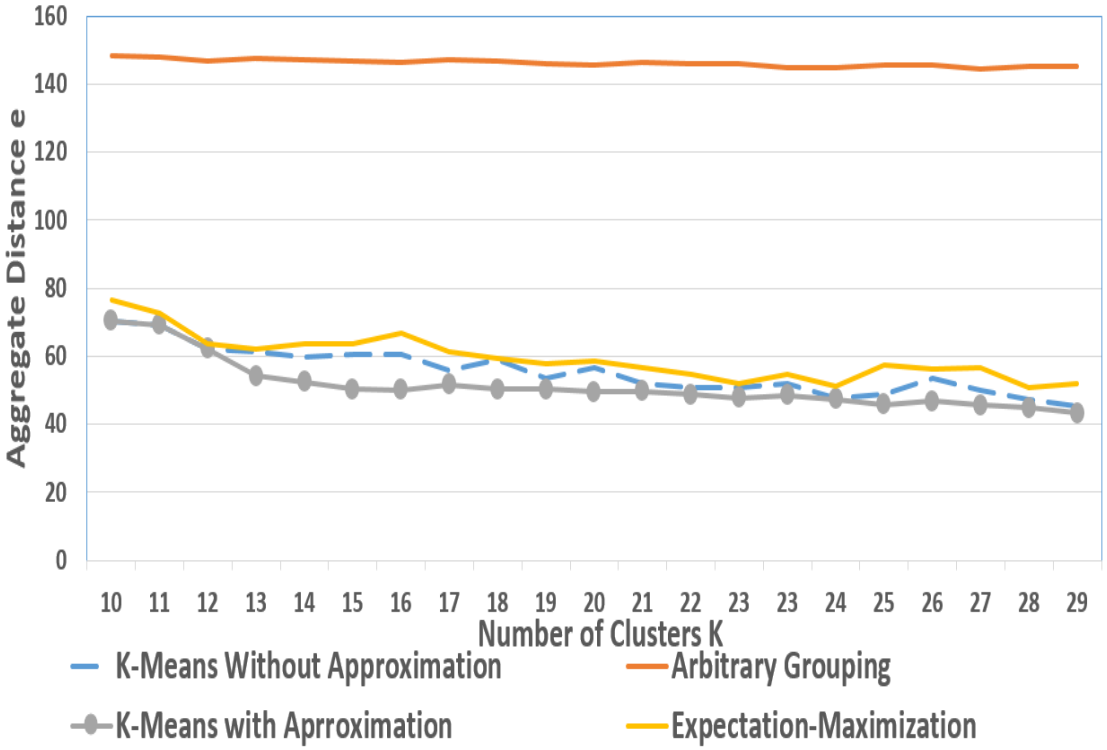

Figure 4.4: Aggregate distance of different methods

Chapter 5

Traffic Profile

This section gives an analysis of the traffic profiles of RRHs within a BBU pool, and the traffic profiles of RRHs are predicted using deep recurrent neural network models.

5.1 Traffic Profile Analysis

Based on the results obtained in Section 4, each cluster is assumed to be served by one BBU pool. Figure 2.5 shows the typical traffic profiles of two types of areas, however traffic profiles in real life is a lot more diverse and detailed analysis of traffic profiles of RRHs are then required to facilitate traffic profile prediction.

5.1.1 Traffic Profiles in a BBU pool

According to the clustering results, the number of RRHs within a cluster ranges from 31 to 329 base stations or RRHs depending on the locations of the cluster. Due to the large number of clusters and large number of base stations within a cluster, it is not possible to show the traffic profiles of all base stations within all cluster. Thus, selected few traffic profiles of clusters with most, median, and least number of RRHs are studied.

Figure 5.1, 5.2, 5.3 show the the traffic profiles of 6 randomly selected RRHs within the cluster. It can be clearly observed that there are traffic profiles of different nature and magnitude within a BBU pool and each one has some form of seasonality, however the seasonality is highly complex and non-linear. This

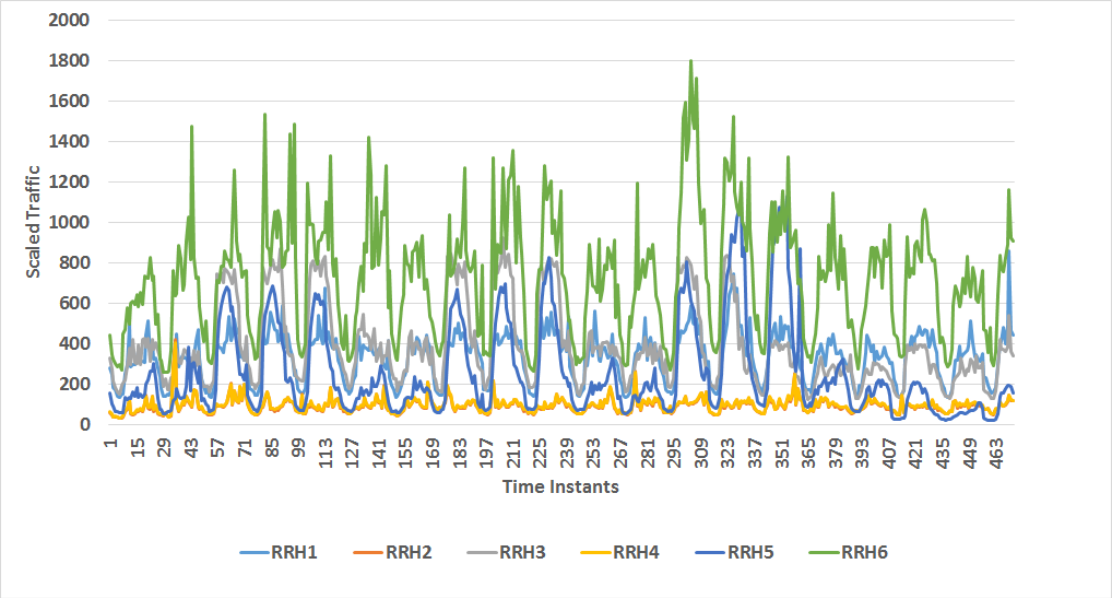

Figure 5.1: Traffic Profiles of 6 RRHs within cluster that has 329 RRHs

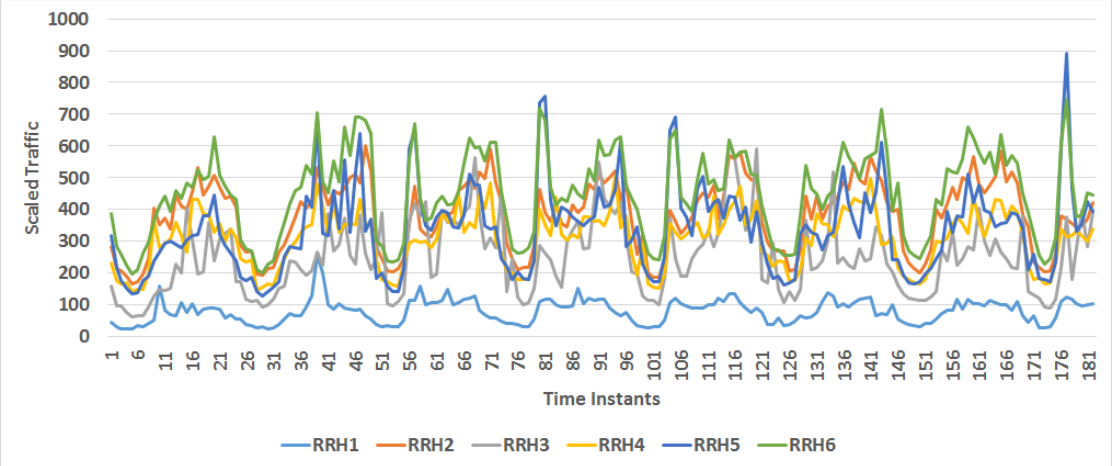

Figure 5.2: Traffic Profiles of 6 RRHs within cluster of that has 76 RRHs

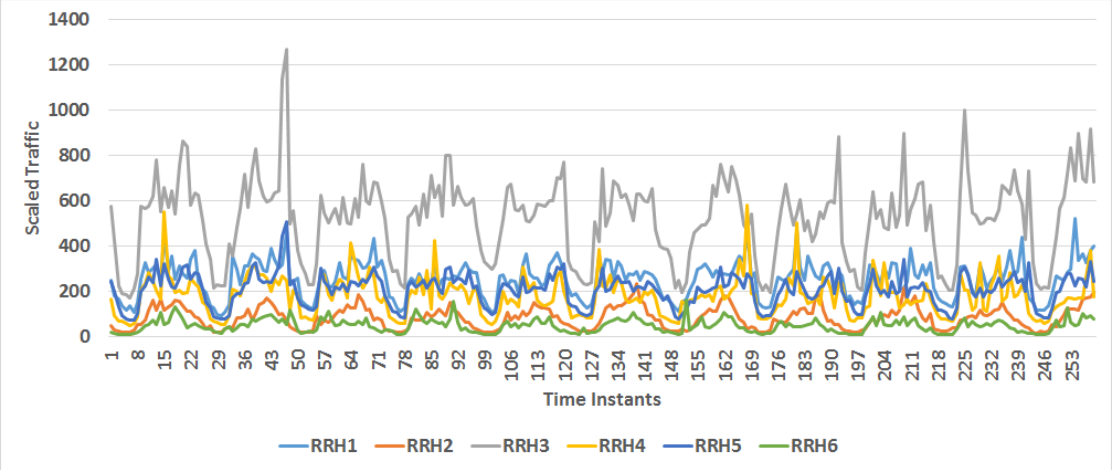

Figure 5.3: Traffic Profiles of 6 RRHs within cluster of that has 31 RRHs

irregularity or complexity could be attribute to the complex temporal relationship of this data, for example, the day of the week can play a role in the number of traffic generated at a certain time instant. Furthermore, such temporal features are neither obvious nor easy to engineer, therefore a deep learning model is highly motivated for predicting such time series.

5.1.2 Traffic Profiles of BBU pools

It can be observed that RRHs within a BBU pool have traffic of various magnitude and nature, and it was previously hypothesised that by aggregating RRHs' traffic profiles, the overall traffic profile will become smoother and more predictable, and this section serves to confirm such hypothesis. Figure 5.4, 5.5, and 5.6 show the aggregated traffic profiles of the clusters of study, it can be inferred that aggregation of traffic profiles indeed make the pattern more observable and predictable, therefore it is hypothesised that the prediction accuracy can then be enhanced.

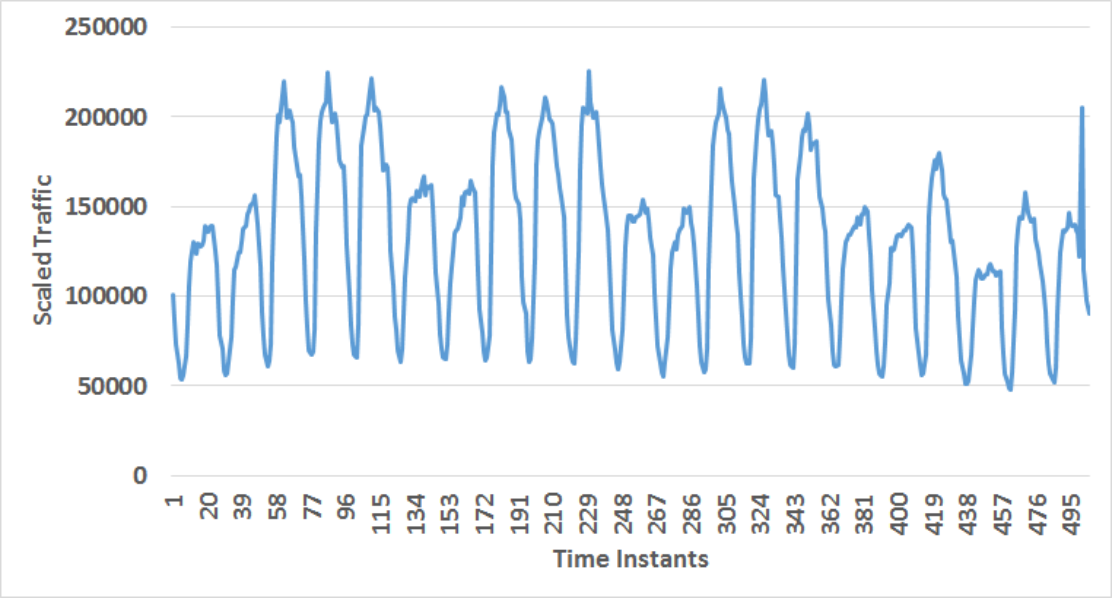

Figure 5.4: Aggregated Traffic Profiles of RRHs within cluster that has 329 RRHs

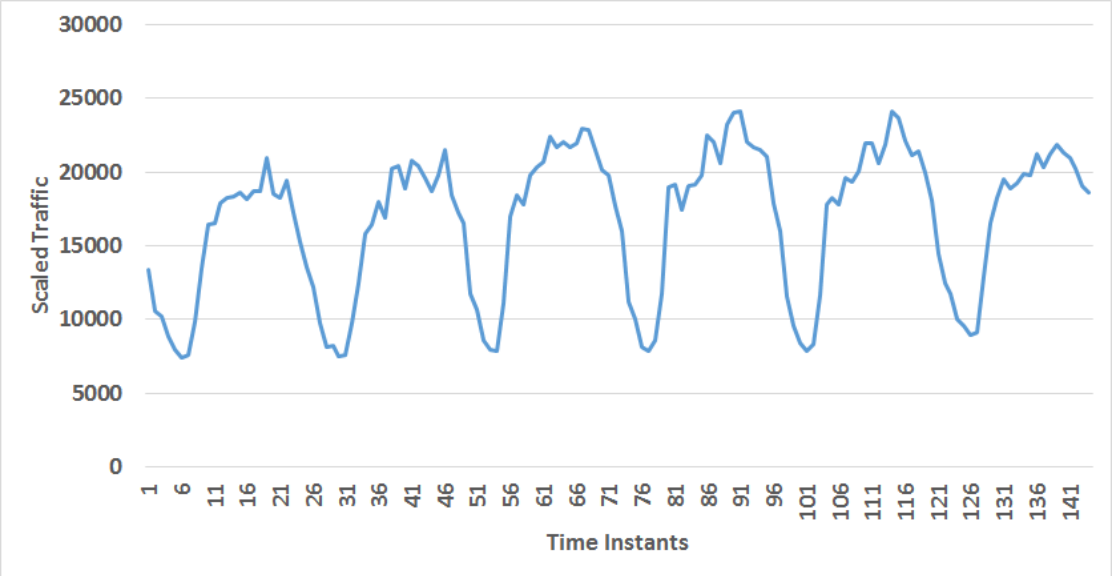

Figure 5.5: Aggregated Traffic Profiles of RRHs within cluster that has 76 RRHs

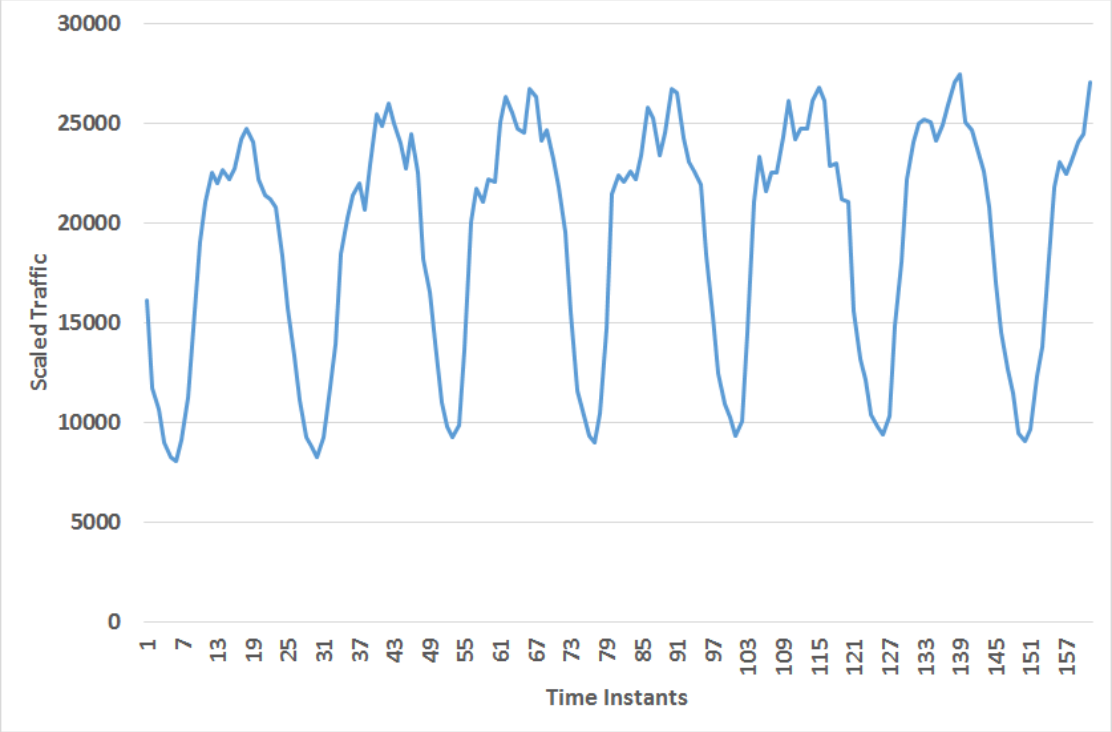

Figure 5.6: Aggregated Traffic Profiles of RRHs within cluster that has 31 RRHs

5.2 Traffic Profile Prediction

This section shows the prediction accuracy of the traffic profiles of RRHs and BBU pools using deep recurrent neural networks. The hyper parameters of the neural networks are determined through experimentation and various activation functions are experimented to find the optimal one. In order to further improve the usability of the dataset, the days are divided into workdays and holidays as some of them exhibit traffic profiles of different nature for a given RRH or BBU pool.

5.2.1 BBU Pool Traffic Prediction

The hyper parameters of the deep recurrent neural network is shown in Table I. The

Table I: Neural Network Hyper Parameters

Parameters	Value
RNN Models	LSTM,GRU
Training Set	70%
Validation Set	15%
Test Set	15%
Initial Learning Rate	0.001
Architecture	1X50X50X1
Gradient Descent Technique	Adam Optimiser
Activation Functions	RELU, Tanh,
Iterations	100

neural network has 1 input and 1 output, and there are two hidden layers which have 50 neurons each, and the number of iteration model attempts at learning is 100. Figure 5.7 demonstrate a few examples of the effectiveness of the RNN models for some BBU pool traffic profiles, and similar trends are observed for traffic profiles of other BBU pools. The orange section denotes the how well the model works against the training set while the green part shows how well the model works against the validation and testing sets. It can be seen that this RNN model is successful in identifying and neglecting the abnormalities occurred during the day and the results suggest that this deep learning model is plausible with a good prediction accuracy. In addition to the visualisation, the suitability of this

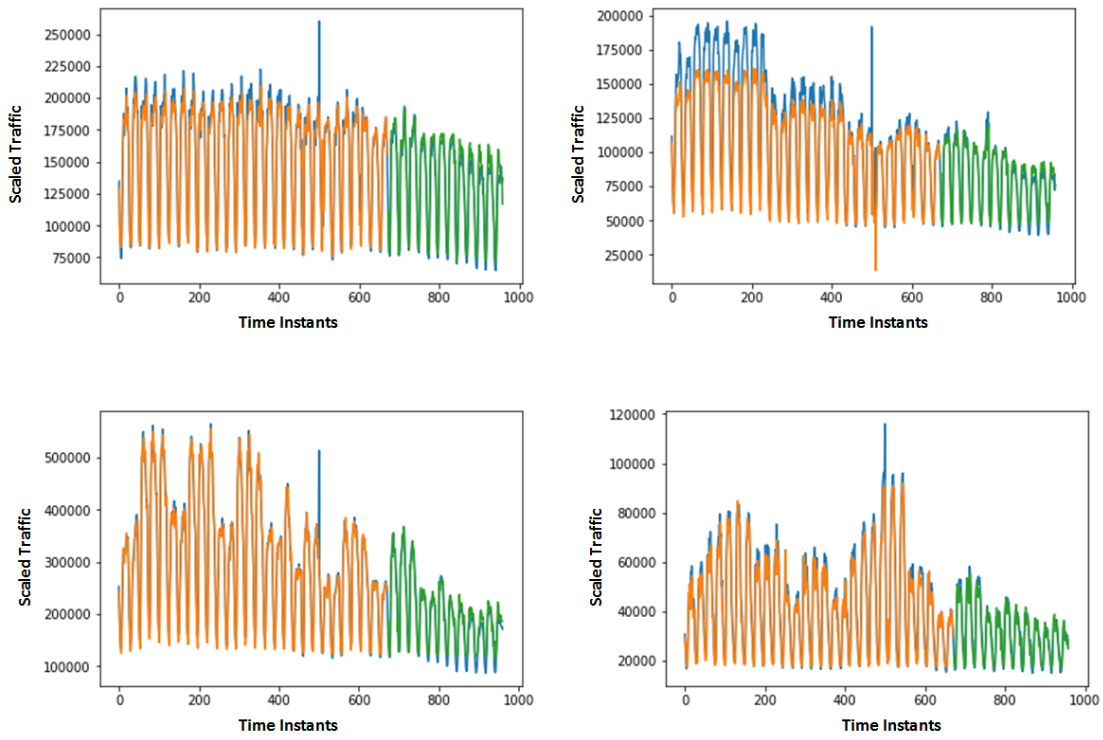

Figure 5.7: Training and Testing Visualisation

model is also analysed against metrics such as Root Mean Square (RMSE) and Symmetric Mean Absolute Percent Error (SMAPE) of the validating and testing sets as in Figure 5.8 and Figure 5.9.

It can be observed from Figure 5.8 and Figure 5.9 that the optimal combination of activation functions and RNN model depends on the type of days as well as the type of region, therefore it could be argued that each RNN model should be tailored for the BBU pool to achieve optimal performance. RMSE errors are measured in the units of scaled traffic in this case, thus SMAPE metric is used to allow some interpretation on these results. Figure 5.10 and Figure 5.11 shows the percentage error of SMAPE metric. It can be observed that SMAPE rescales the magnitude of the error of RMSE due to normalisation, and it is also evident that most of these combinations produce accurate predictions as the percentage error is small in general.

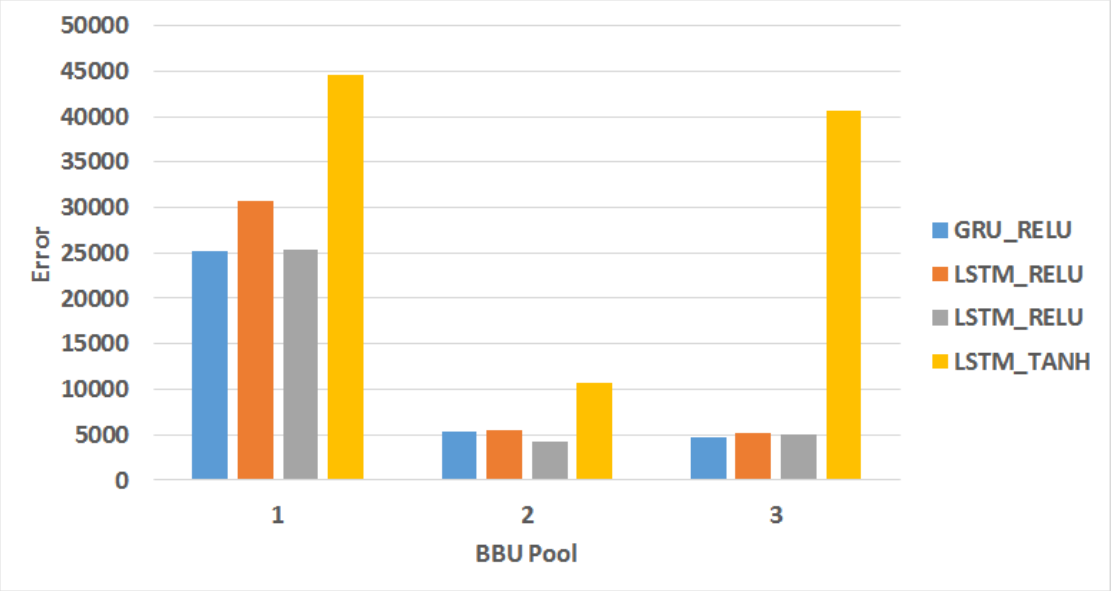

Figure 5.8: RMSE for holiday traffic of BBU pools

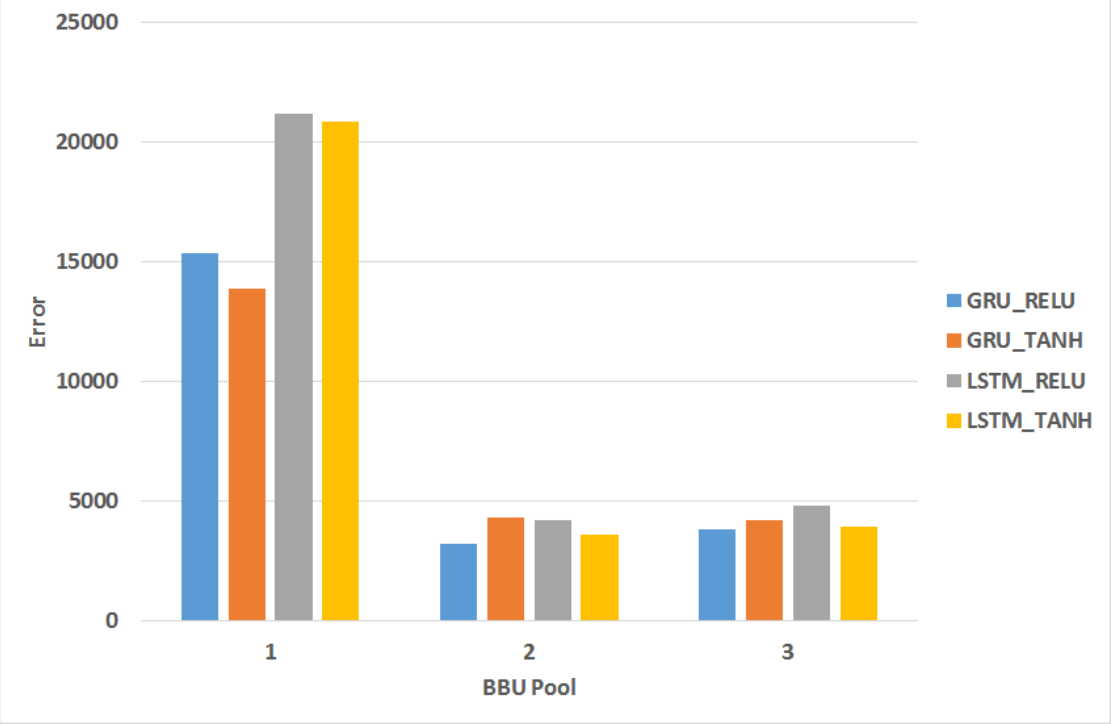

Figure 5.9: RMSE for workday traffic of BBU pools

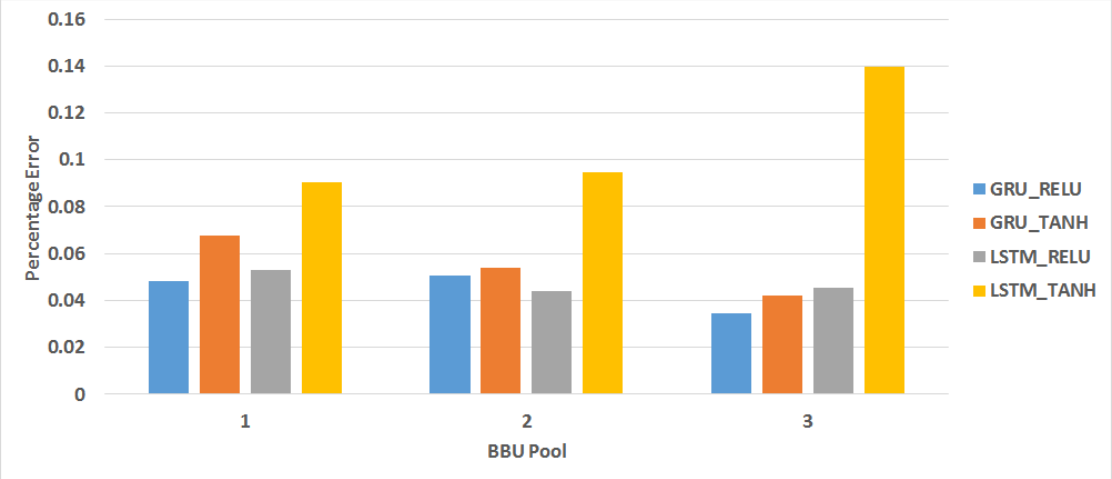

Figure 5.10: SMAPE for holiday traffic of BBU pools

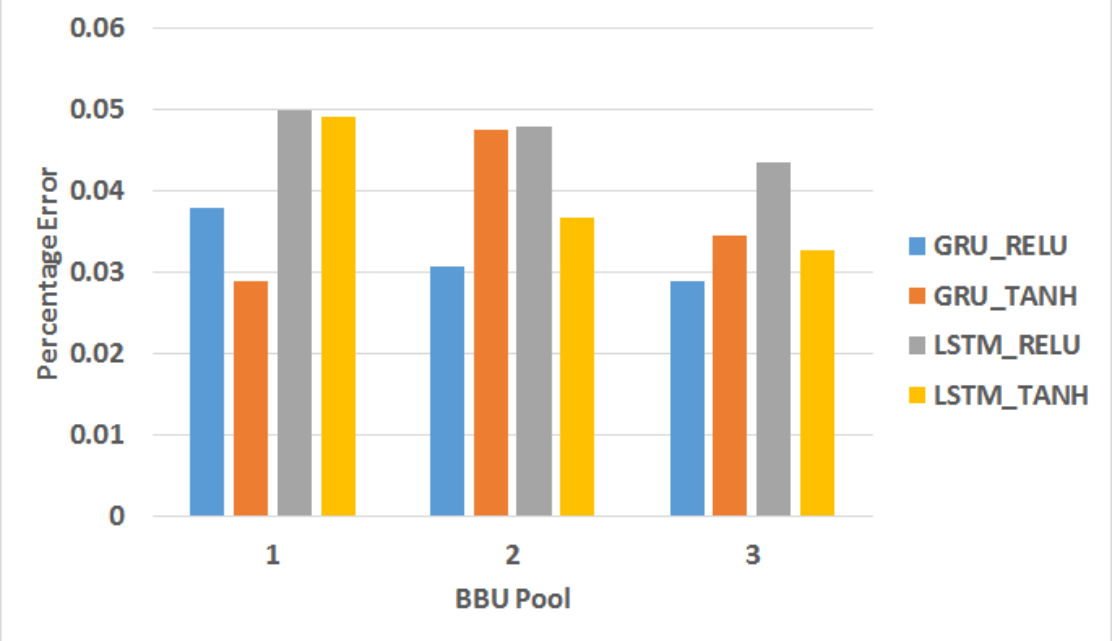

Figure 5.11: SMAPE for workday traffic of BBU pools

5.2.2 RRH Traffic Prediction

Due to the large number of RRHs, only a selected few RRHs' traffic profiles are presented in this section. The relevant hyper-parameters are the same as in Table I. The accuracy of the prediction of the training, validation and testing sets are shown

in Figure 5.12. It can be observed in Figure 5.12 that a reasonable accuracy is

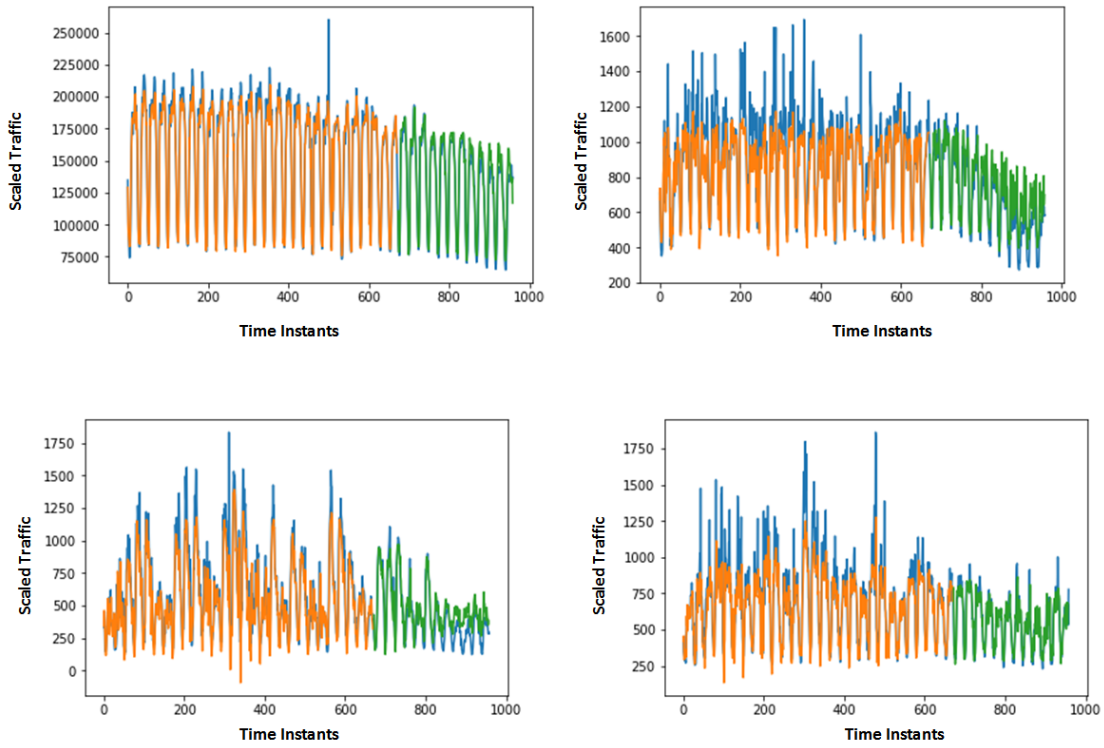

Figure 5.12: Training and testing visualisation

achieved for the prediction task despite the highly complex nature of the data. The SMAPE and RMSE metrics are also used to evaluate the prediction accuracy of the selected RRHs, and visualisation of these metrics can be found in Figure 5.13, and 5.14, it can be observed that certain combinations outperform others for the chosen RRH, however further investigation is needed to validate if such trend extends to all RRHs. However, it should also be kept in mind that each RRH can have its own type of recurrent neural network models, therefore it is not of great interest to show such fine tuning.

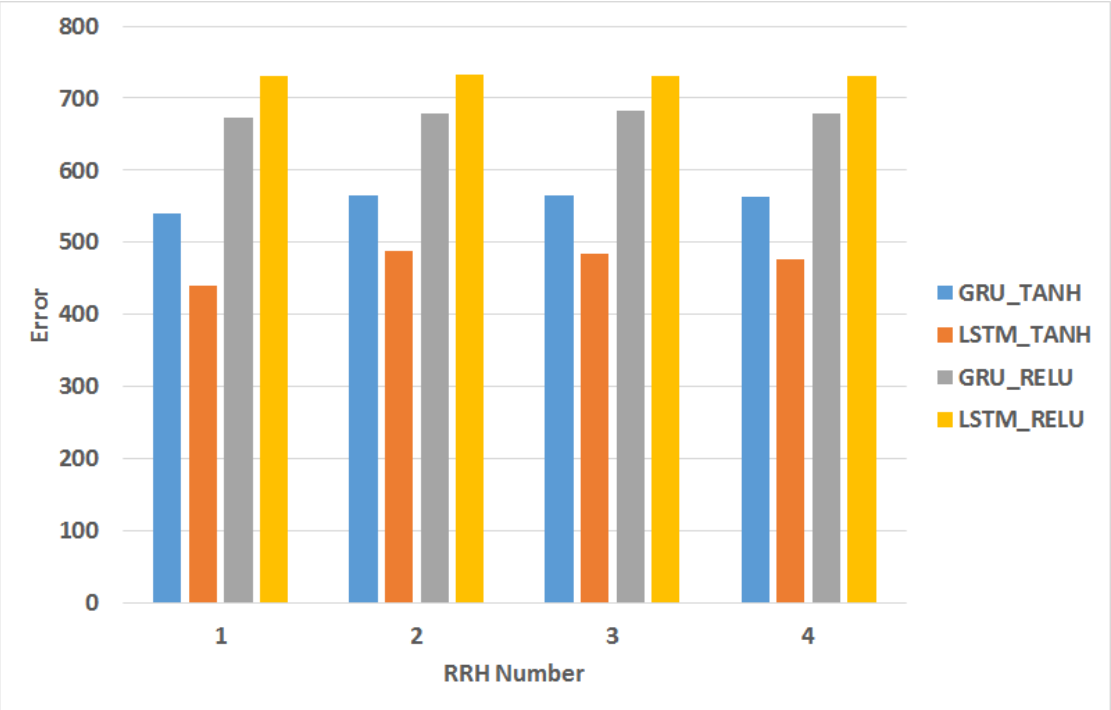

Figure 5.13: RMSE for arbitrary RRHs

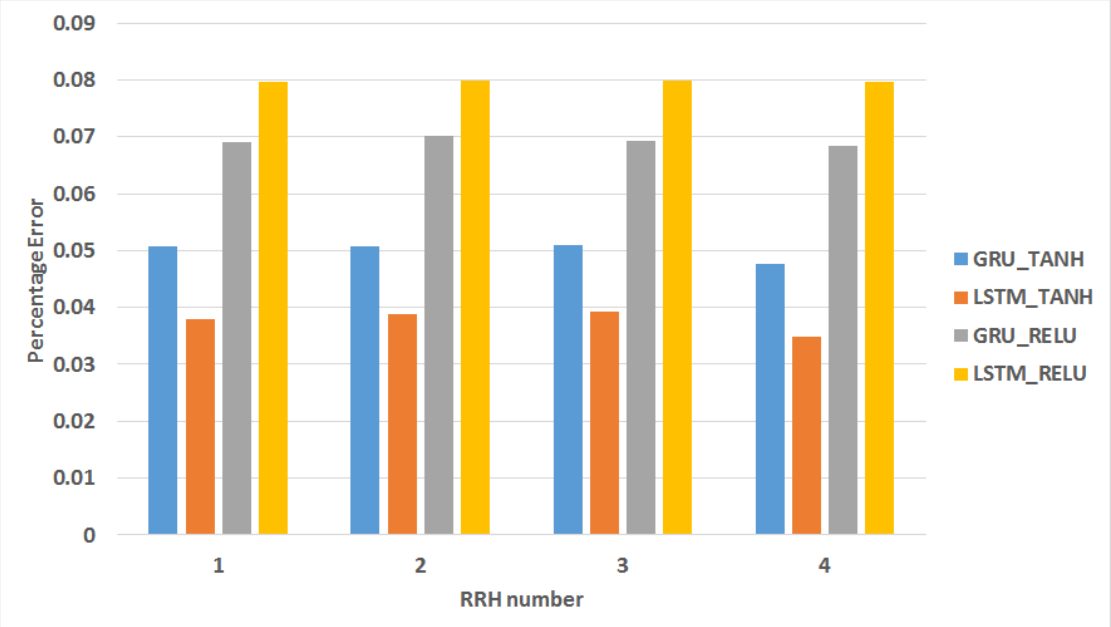

Figure 5.14: SMAPE for arbitrary RRHs

Chapter 6

Switching Algorithm

This section proposes a traffic-aware BBU-RRH switching algorithm that is able to reduce the switching complexity while improving the QoS provision. The existing algorithm that serves as a comparison baseline is described in Section 2.5.1. The proposed algorithm is designed and published by the author in [42].

6.1 Motivation

As suggested in literature review, existing switching algorithms may have achieved significant BBU usage reduction, but almost all of them may encounter the problem of high switching complexity. This high switching complexity is attribute to the philosophy of their resource allocation schemes. These algorithms only allocate resource at a fixed time interval or when there is insufficient capacity at a certain BBU within the pool. It can be observed in previously studied traffic profiles that traffic increases at a fast rate during certain certain time intervals (e.g. start of work hours), and resource allocation performed during that time period may become inadequate shortly. The same idea can be applied to the period of time when traffic decreases, because the resource allocation performed then will become inefficient in a short time. It is therefore reasonable to propose a switching algorithm that is able to perform resource allocation with the knowledge of the traffic profile. This will allow the resource allocation to be more conservative during rising traffic period and more aggressive during decreasing traffic period. Eventually, this algorithm can achieve low switching complexity as well as close-

to-optimum BBU usage without compromising on the QoS provision.

6.2 System Modelling & Assumptions

In C-RAN, every RRH is logically connected to every BBU within the BBU pool, which means the RRH can be dynamically switched over to another BBU within the pool if necessary.

6.2.1 RRHs' Arrangement and Traffic Profiles

The arrangement of RRHs follows a conventional hexagonal cell approach, and the typical traffic profiles of office and residence, described in Figure 2.5, are used for simulation purposes. A visual illustration is found in Figure 6.1. A set which contains all RRHs is denoted as R.

6.2.2 BBU Modelling

Each BBU within the BBU pool is modelled as a container that has a maximum capacity of 1. Each BBU has 4 thresholds, and these thresholds are named as upper threshold, soft upper threshold, lower threshold, and soft lower threshold. The soft thresholds indicate how conservative the algorithm is when a BBU-RRH association or resource allocation is required, and these soft thresholds are dynamically updated. If the usage of a BBU is above soft upper threshold, then the algorithm starts to offload some of its RRHs to other BBUs. If the usage of a BBU is below soft lower threshold, then the algorithm will try to shut down this BBU by offloading all its RRHs to other BBUs. The BBU capacity model is shown in Figure 6.2.

In order for a BBU to be aware of the traffic profiles, the traffic of the i^{th} RRH within the BBU is denoted by the following equations:

$$C^i_{rrh} = C^i_{used} + C^i_{trend} + \beta_i$$
$$\triangle C = \sum C^i_{trend}$$

Where:

- C^i_{rrh} is the amount of resource required by the i^{th} RRH.

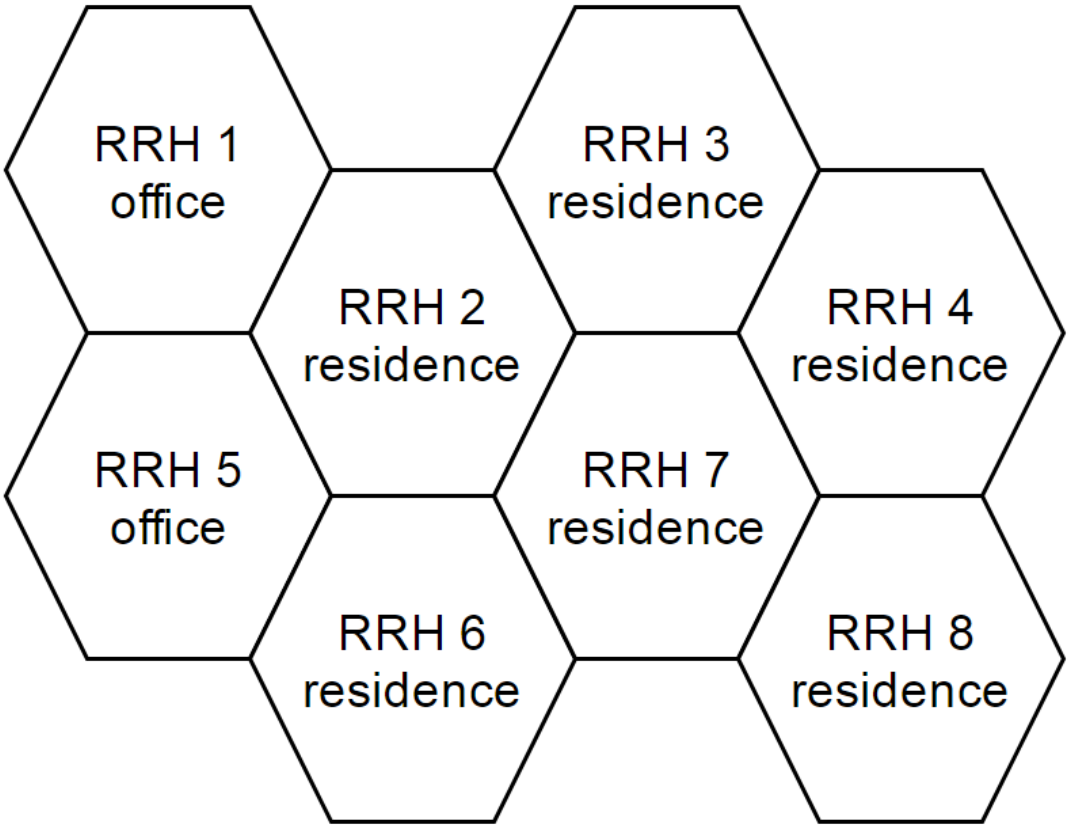

Figure 6.1: Arrangement and Traffic Profiles of RRHs

- C^i_{used} is the amount of resource a BBU actually needs to accommodate the i^{th} RRH.
- C^i_{trend} is the predicted increase or declination of traffic of the BBU associated with i^{th} RRH.
- β_i is the amount of resource reserved for uncertainty in the traffic profile of the i^{th} RRH in a BBU.
- $\triangle C$ is the sum of trends of all RRHs within the associated BBU.

The soft upper and lower thresholds of a BBU are updated according to the

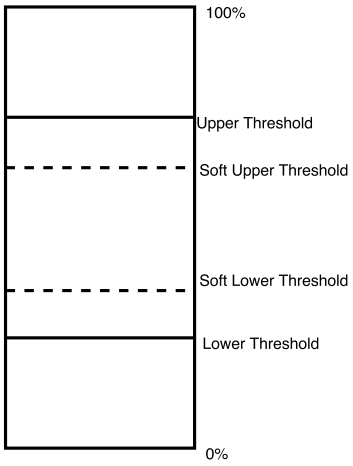

Figure 6.2: BBU Capacity Model

following rules:

$$T_{su} = T_u - \sum C^i_{trend} - \bar{\beta}$$
$$T_{sl} = T_l - \sum C^i_{trend}$$

Where:

- T_{su} is the soft upper threshold.
- T_u is the upper threshold.
- $\bar{\beta}$ is the mean of resource reserved for uncertainty of all RRHs within the BBU.
- T_{sl} is the soft lower threshold
- T_l is the lower threshold

Furthermore, soft lower threshold cannot be less than lower threshold and the soft upper threshold cannot be more than upper threshold. A set which contains all BBUs in the pool is denoted by B.

6.2.3 Fronthaul Link

The Fronthaul link is assumed to be a high throughput fiber optical network with low latency, and this link is capable of handling the traffic between BBU pools

and RRHs.

6.2.4 Assumptions

The following assumptions are made when simulating the algorithm:

1. The change of traffic is linear within one hour.
2. The BBU is in congestion state if the load is more than its upper threshold, and this will result in poor QoS.
3. The BBU will is in a failure state if the load is more than its physical capacity, and this will result in inadequate service provision.
4. Accommodating neighbouring RRHs within the same BBU will enhance the Coordinated Multipoint (CoMP) technology. [35]

6.3 Problem Formulation

This traffic profile aware switching algorithm that will attempt to switch the RRHs to its neighbours' BBU if the current one has too much load , otherwise this RRH will be switched to other suitable BBU. RRHs in under-utilised BBUs will be switched or offloaded to other BBUs to decrease the number of active BBUs. After executing this algorithm, the BBUs will have the following states:

- The soft upper threshold is less than the upper threshold.
- The soft lower threshold is greater than the lower threshold.
- The aggregate usage of all RRHs within a BBU is between the soft upper threshold and soft lower threshold.
- The trend within a BBU is as close to 0 as possible.

6.4 Basic Idea

The prior knowledge of RRHs' traffic profiles are used to alter the amount of load a BBU accepts, and the switching is done according to the following principals introduced in [35].

1. A switch is required if the BBU is operating above its soft upper threshold or below the soft lower threshold.
2. If a switch is required, the algorithm should try to switch RRHs to their neighbouring RRHs' BBU when possible.
3. when it is not possible to switch the RRH to its neighbour's BBU, then it should switch other BBUs that will result in a minimum number of BBUs
4. If the load in a BBU is altered in any way, the soft thresholds should be adjusted or updated for the BBU.

6.5 Traffic-Aware BBU-RRH Switching Algorithm with Dynamic Thresholds

This section provides an overview of the proposed algorithm, and this algorithm can be classified into the following phases:

1. Learning Phase: In this phase, the past traffic profiles are studied and analysed via machine learning techniques, so that an accurate traffic profile prediction can be obtained for each RRH.
2. Switching phase: In this phase, the algorithm will attempt to offload RRHs for BBUs that operate above their soft upper threshold, and shut down under-utilised BBUs that operate below their soft lower thresholds.
3. Readdress phase: This phase occurs when there are BBUs with positive $\triangle C$ and negative $\triangle C$ within the BBU pool. The algorithm then tries to switch the RRHs between BBUs with positive $\triangle C$ and negative$\triangle C$, in order for the $\triangle C$ to be as close to 0 as possible for each BBU in the BBU pool.

6.5.1 Learning Phase

A window period M is needed for learning phase, M determines how many days of past traffic profiles are relevant, e.g., when M is 5, it means that the traffic profiles of the past 5 days are relevant. By applying machine learning techniques

demonstrated in previous sections, a predicted hourly traffic profile that is similar to Figure 2.5 can be established. In addition to the predicted traffic profile, it is also important to determine the standard deviation of the traffic load of each hour to enhance the effectiveness of the algorithm. These information can be collected into the following sets:

$$P^i = \{P_0^i, P_2^i \dots P_{23}^i\}, \quad RRH_i \in R$$
$$D^i = \{D_0^i, D_1^i \dots D_{23}^i\}, \quad RRH_i \in R$$

Where P^i is the volume of traffic of the i^{th} RRH at every hour, and D^i denotes the standard deviation of traffic load of the i^{th} RRH at each hour.

6.5.2 Switching phase

There are two types of switching mode in the switching phase, namely offload and shut-down. The details of these two modes are as follows.

Offloading

In the case of offloading, the BBU will continue switching RRH with the least traffic to other BBUs until the usage of BBU is no longer more than its soft upper threshold. The destination BBU is chosen in following order of preference:

- Neighbouring RRHs' BBUs which can accommodate the load of the RRH being offloaded.
- Active BBUs which can accommodate the load of the RRH being offloaded.
- Unused BBUs.

The above idea can be represented in Figure 6.3.

Shut-Down Logic

The algorithm attempts to shut down BBUs whose usage of is less than its soft lower threshold. The algorithm offloads the RRHs to other BBUs with the following order of preference:

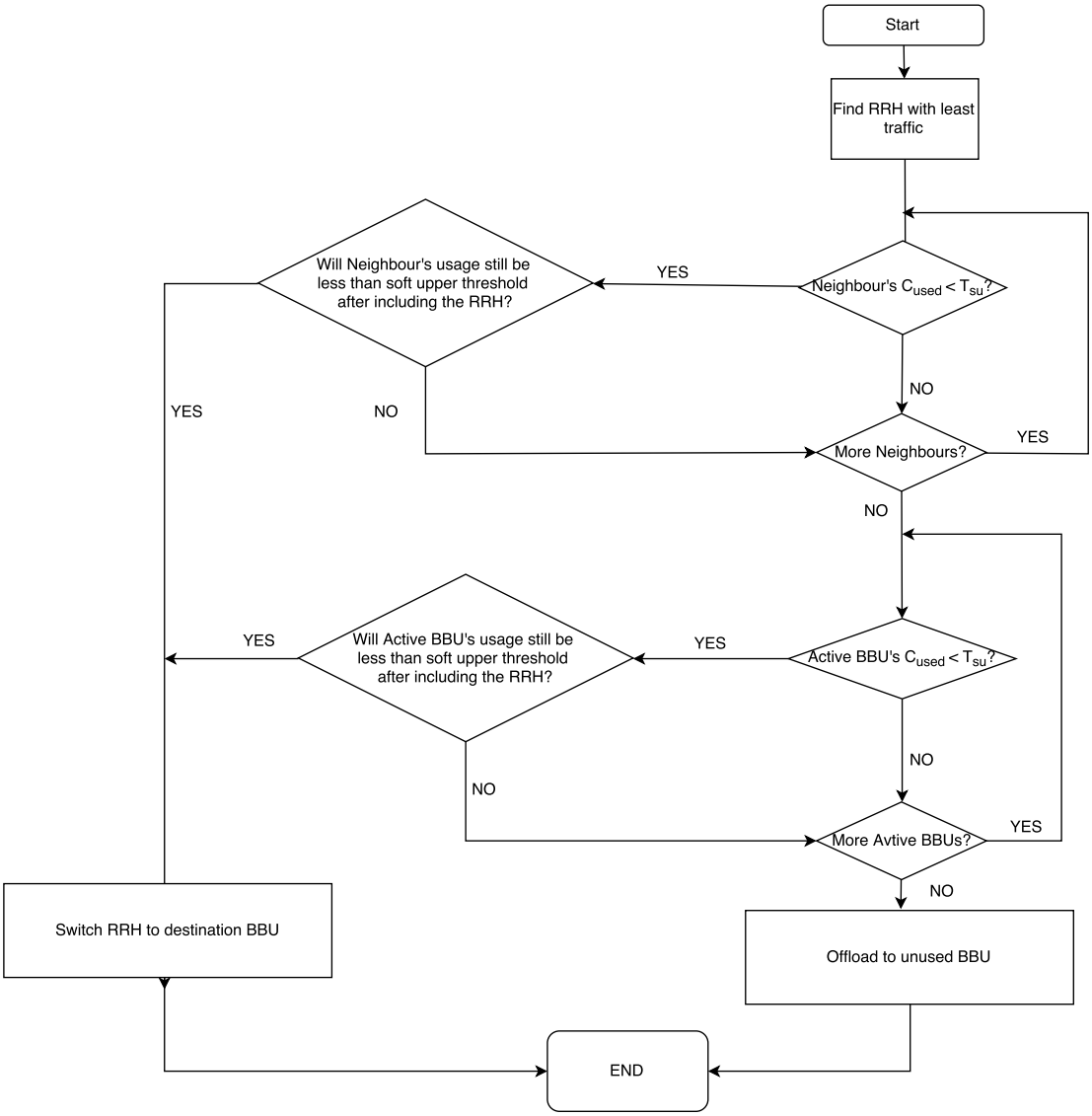

Figure 6.3: Offload Logic

- Neighbouring RRHs' BBUs which can accommodate the load of the RRH being offloaded.
- Active BBUs which can accommodate the load of the RRH being offloaded.

After the switching phase, all BBUs' usage should be between soft upper and soft lower thresholds, and this should reduce the probability of BBU having insufficient capacity or inefficient usage until the next resource allocation interval, a flow chart

of the shut-down logic can be found in Figure 6.4.

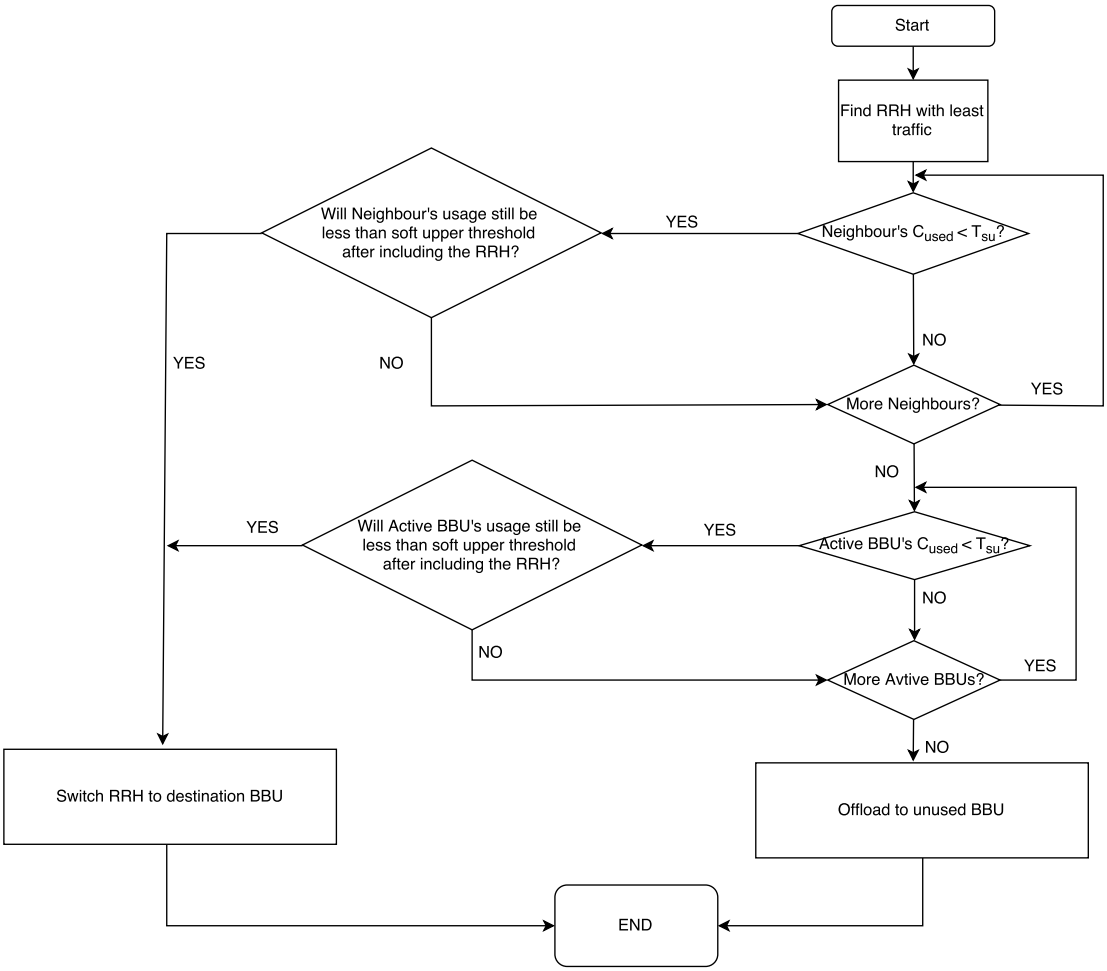

Figure 6.4: Shut-down Logic

6.5.3 Redress Phase

During this phase, the algorithm will attempt to swap the RRHs of different BBUs in order to keep the value of $\triangle C$ as close to 0 as possible. After redressing phase, the rate of utilisation of each BBU is effectively improved as less resource is occupied due to uncertainty. Let all active BBUs be classified as the following sets:

$$B_+ = \{B_{1+}, B_{2+}, \ldots B_{m+}\}, \quad B_+ \in B$$
$$B_- = \{B_{1-}, B_{2-}, \ldots, B_{n-}\}, \quad B_- \in B$$

Where B_+ is a set of active BBUs whose $\triangle C$ value is larger than 0, and B_- is a set of active BBUs whose $\triangle C$ value is less than 0. During this phase, the algorithm will iterate over the set whose sum of all $\triangle C$ of BBUs is larger, and then keep offloading RRHs with least $|C_{trend}|$ to BBU that has most unused capacity in the other set. This is repeated until one set is empty. It should be noted that it may not be possible to offload RRHs to BBUs in the other set due to insufficient capacity. Therefore a trading algorithm, shown in Figure 6.5, is introduced to facilitate the redress phase. The complete logic flow of redress phase is shown in Figure 6.6. Offloading is given preference over this phase as they pose less complexity.

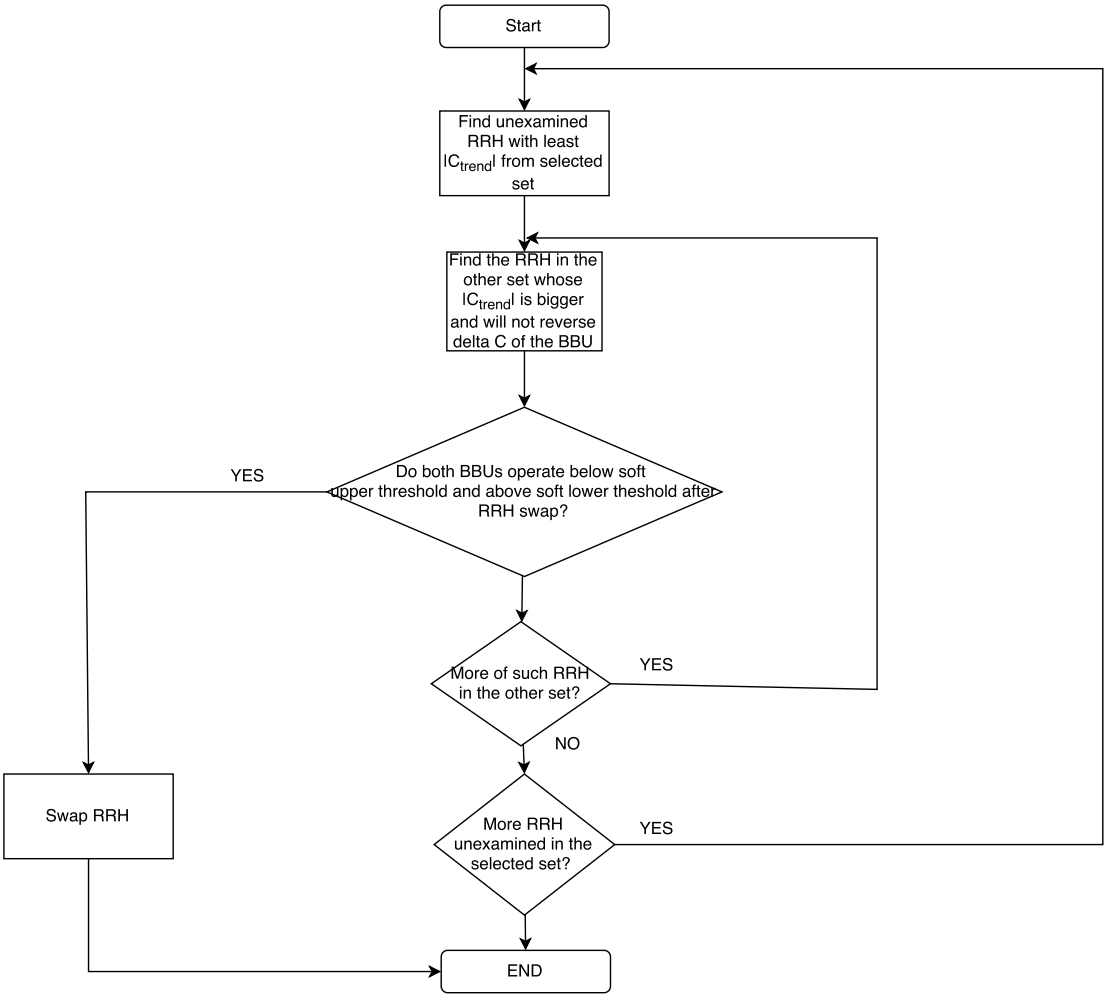

Figure 6.5: Trading Logic

6.5.4 Overview of Algorithm

This section shows how different phases or parts of the algorithm relate to each other. This algorithm is run after every resource allocation interval is elapsed or when the usage is more than soft upper threshold or less than soft lower threshold. These ideas can be represented in Figure 6.7.

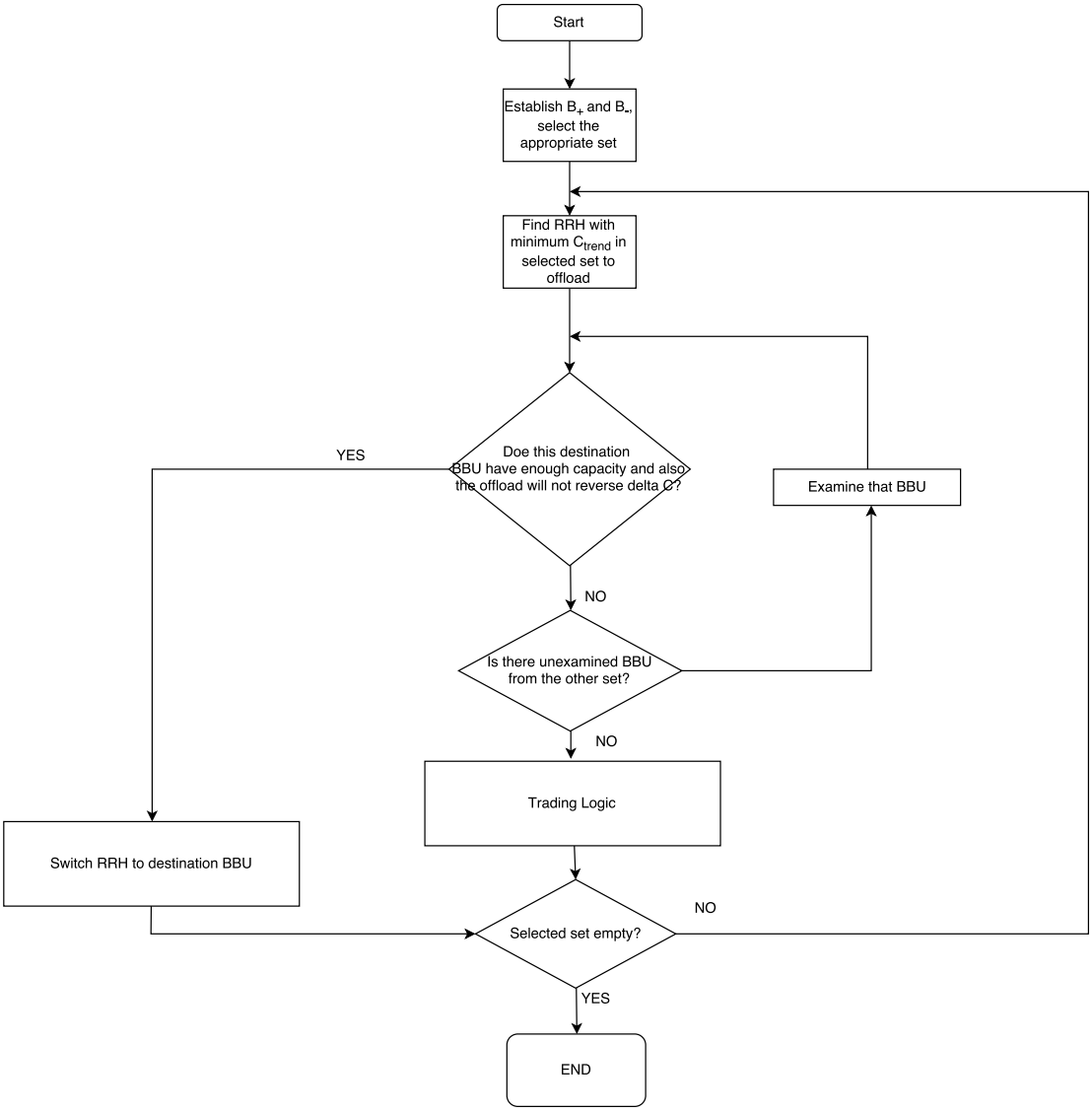

Figure 6.6: Readdress Logic

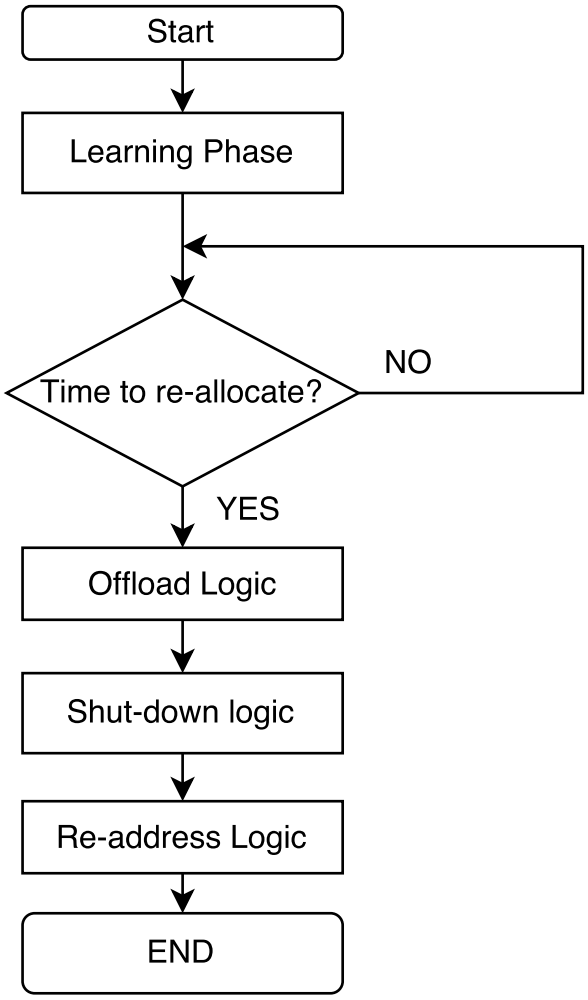

Figure 6.7: Overview of all phases

6.6 Simulation Results

This section shows and compare the simulation results of the proposed algorithm to the baseline algorithm [35] which does not consider traffic profiles in the switching process.

6.6.1 Simulation Parameters

The following parameters are used during the simulation process:

Parameters	Value
Re-allocation Interval	60 minutes
C_{trend} Considered	60 minutes
Number of RRHs	10 X 10
Traffic Profiles	Office and residence shown in Figure 2.5
Simulation Duration	24 hours
Upper Limit	0.9
Lower Limit	0.5
Standard Deviation of Traffic Profiles	Randomly generated, at most 20%

6.6.2 Simulation & Results

This section presents results of the proposed algorithm on BBU usage reduction, QoS and switching complexity. The results of 3 cases of this algorithm is shown to demonstrate the effectiveness. The algorithm that the proposed algorithm compared to is discussed in [35].

Case 1

In this case, the algorithm is simulated for the duration of one day. This allows the interpretation on the logic behind the decision making process of the algorithm. It should be noted that 100 BBUs would be required in a traditional architecture, which is significantly more than the C-RAN counterpart. The performance of this algorithm is shown in Figure 6.8.

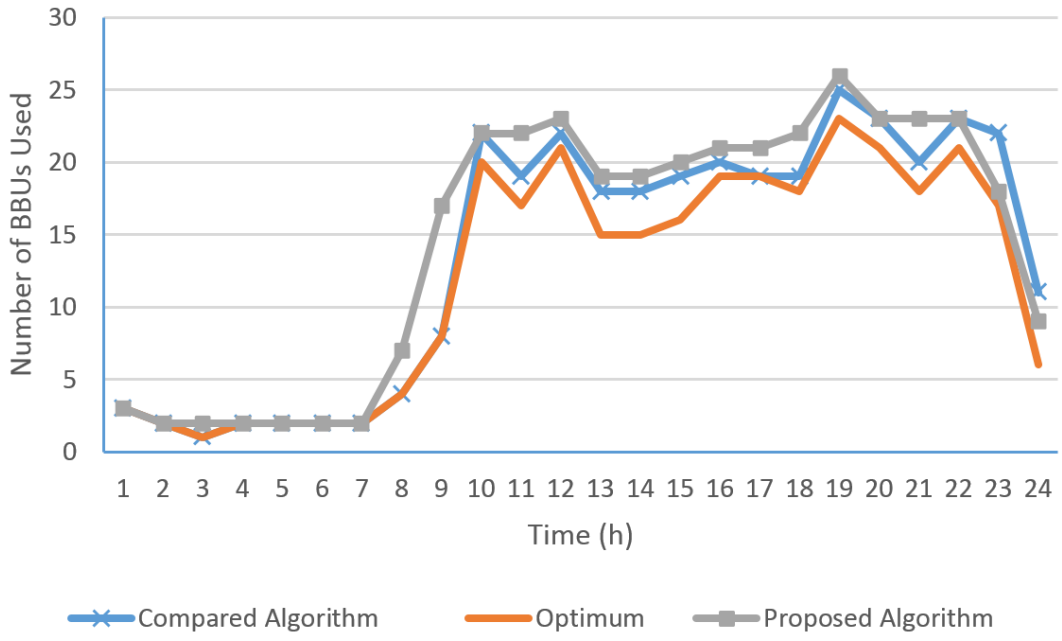

Figure 6.8: Algorithm Performance VS Optimum

It can be observed in Figure 6.8 that both performs close to optimum level of BBU usage reduction. It may seem that the proposed algorithm is less efficient between 7 to 10 AM as the proposed algorithm requires more BBUs to operate. This however is not true in reality, the traffic between these hours rise at a very fast rate. If the resource allocation does not take this into consideration, then it is almost inevitable for BBUs to run into capacity problems, such as congestion or failure. Furthermore, if the compared algorithm reallocates the resource every time a capacity problem occurs, then this will generate additional switching traffic (Control traffic) that could further aggravate the problem. If the algorithm reallocates the resource whenever BBU runs into capacity problem, it still has no guarantee that the new solution can remain effective for a long enough period because the compared algorithm does not take future traffic into consideration. This could then impose a very high switching complexity if such resource allocation changes are made rapidly. It should also be noted that the proposed algorithm outperforms its counterpart when the traffic is decreasing, and this is attributed to the more aggressive shut-down strategy that is enabled by traffic-awareness.

Figure 6.9 shows the number of occurrence of congestion state or failure state if no resource allocation is done between the hour interval. The BBU is in congestion if its usage is more than upper threshold, and it is in failure state if its demand is more than the physical capacity limit.

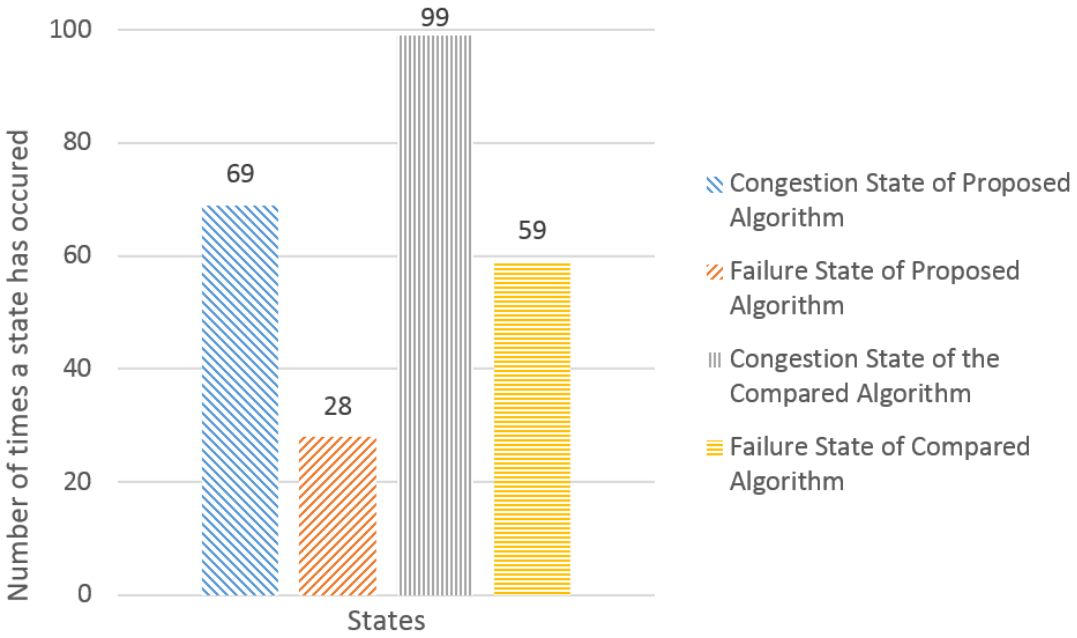

Figure 6.9: Comparison of BBU States

It can be observed from Figure 6.9 that the proposed algorithm has reduced the number of occurrence of congestion and failure as the result of its traffic-awareness. The proposed algorithm also operates at a high efficiency of 83.4% shown in Figure 6.10

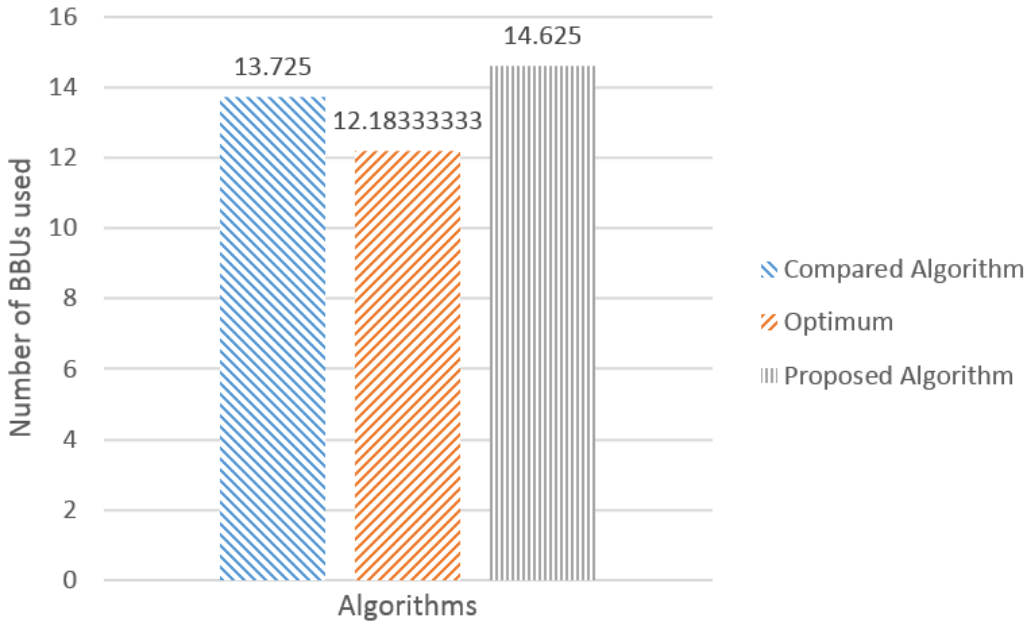

Figure 6.10: Average number of BBUs used by Algorithms within a Day

Case 2

This section presents results of 100 simulation iterations, the performance and comparison is shown in Figure 6.11. The results are similar to that of a single run. Figure 6.12 shows that the proposed algorithm has an efficiency of 83.3%

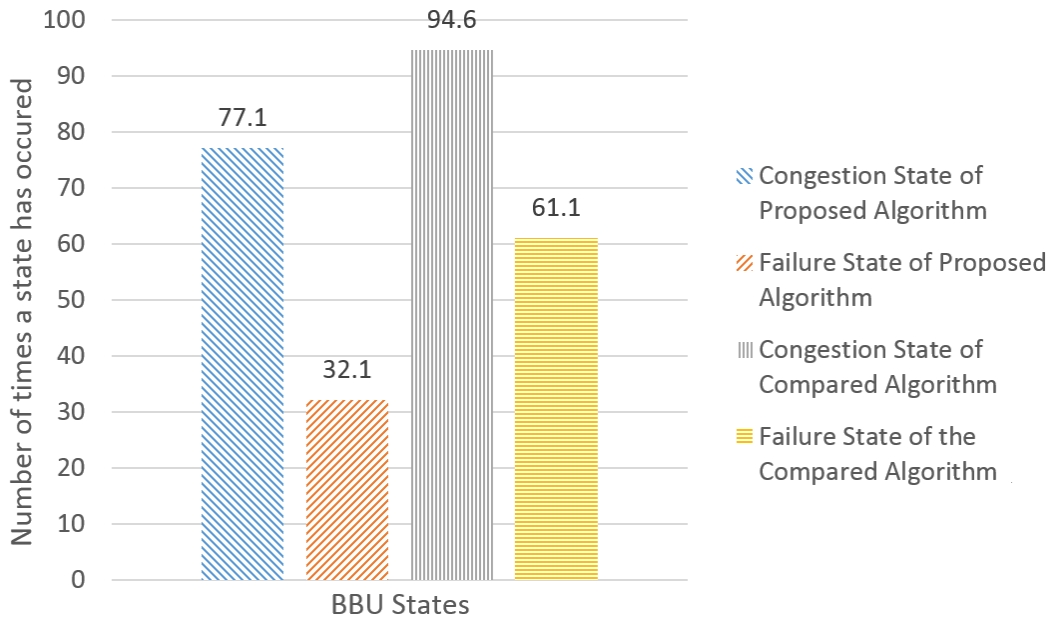

Figure 6.11: Comparison of BBU States for Multiple Runs of Simulation

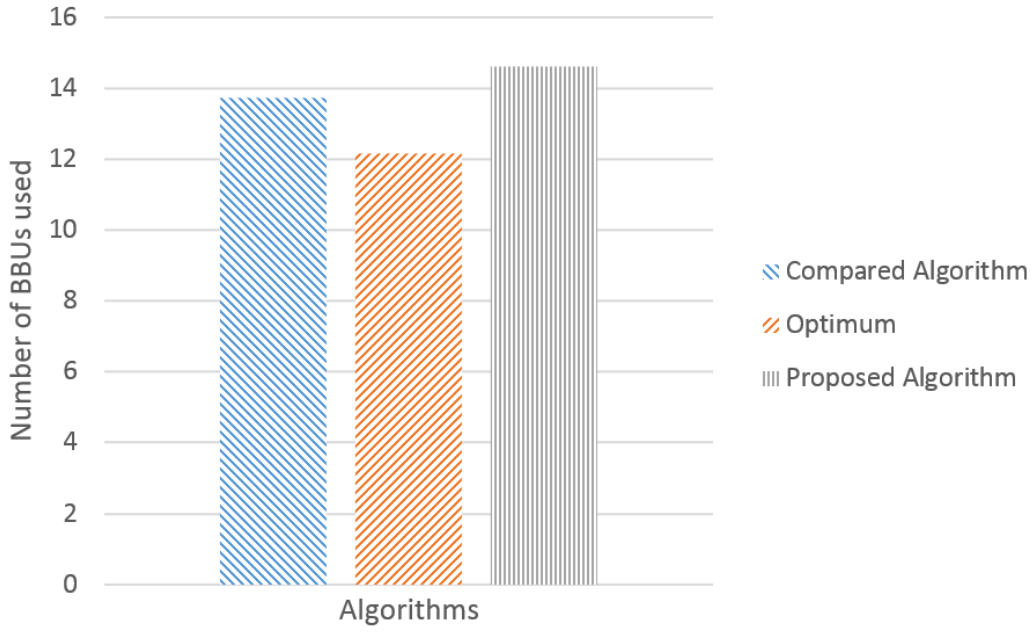

Figure 6.12: Average number of BBUs used by Algorithms within a Day for Multiple Runs of Simulation

Case 3

Both algorithms are simulated for 100 times and the granularity of the simulation is now changed to 1 minute, which means the states are monitored per minute. The average number of switches is shown in Figure 6.13. It can be observed that the proposed algorithm has reduced the average number of switches in comparison to that of [35].

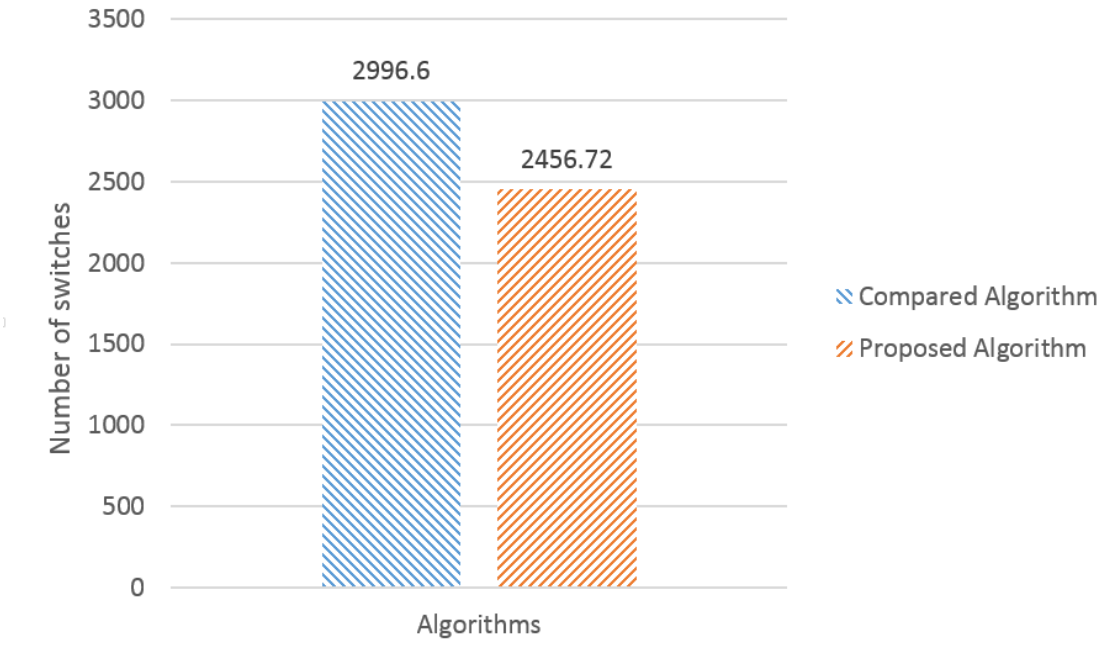

Figure 6.13: Average number of switches by Algorithms within a Day for Multiple Runs of Simulation

Chapter 7

Conclusion

This work has investigated three inter-related problems, namely base station geographical clustering, traffic profile prediction, and BBU-RRH switching algorithm. A comprehensive approach for BBU-RRH switching process can be realised by combining these three components.

The results of the clustering have shown that K-means is indeed a suitable method for such geographical location clustering task, and the proposed modification is able to reduce its error and therefore enhance its performance. The traffic profile section has demonstrated the effectiveness of recurrent neural network on both BBU pool and RRH traffic profile predictions. It is realised that different BBU pools and RRHs may need different combinations of activation functions and models to achieve an optimal performance. The proposed BBU-RRH switching algorithm is able to reduce the number of active BBUs close to optimum level. The result have also shown that the complexity of the proposed algorithm is reduced compared to others while the QoS provision is also improved.

Chapter 8

Future Work

In order to further improve the this work, the following is recommended for further research.

8.1 Integrate with 5G

5G standards are not yet finalised and commercialised at the time of writing, the algorithms may need further investigation to be fully compatible to 5G mobile networks.

8.2 More Detailed Dataset

The dataset used even though is meaningful and suitable for this work. It however can improve this work further by including more details such as the coverage of RRHs, power rating, etc.

8.3 Weights in Clustering

The clustering section in this work considers aggregate distance as the only parameter, it could be more meaningful if other parameters such as traffic of each RRH are included in the process.

8.4 Multi-homing RRHs

In this work, each RRH only belongs to one BBU, however it is also possible for RRH to belong to multiple BBU in the future. This could inspire different and more efficient switching schemes.

References

[1] CISCO, "Cisco visual networking index (vni) update global mobile data traffic forecast 2016-2021," tech. rep., CISCO, 2017.

[2] *C-RAN:The Road Towards Green RAN.*

[3] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, July 1959.

[4] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, "A brief survey of machine learning methods and their sensor and iot applications," in *2017 8th International Conference on Information, Intelligence, Systems Applications (IISA)*, pp. 1–8, Aug 2017.

[5] T. Rappaport, W. Roh, and K. Cheun, "Wireless engineers long considered high frequencies worthless for cellular systems. they couldn't be more wrong," vol. 51, pp. 34–+, 09 2014.

[6] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 1617–1655, thirdquarter 2016.

[7] C. f. Lai, R. h. Hwang, H. c. Chao, M. M. Hassan, and A. Alamri, "A buffer-aware http live streaming approach for sdn-enabled 5g wireless networks," *IEEE Network*, vol. 29, pp. 49–55, Jan 2015.

[8] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5g network architecture," *IEEE Communications Magazine*, vol. 52, pp. 65–75, Nov 2014.

[9] A. Kaloxylos, "A survey and an analysis of network slicing in 5g networks," *IEEE Communications Standards Magazine*, vol. 2, pp. 60–65, MARCH 2018.

[10] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, pp. 80–87, May 2017.

[11] S. Troia, G. Sheng, R. Alvizu, G. A. Maier, and A. Pattavina, "Identification of tidal-traffic patterns in metro-area mobile networks via matrix factorization based model," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 297–301, March 2017.

[12] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud ran for mobile networks ;a technology overview," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 405–426, Firstquarter 2015.

[13] G. Kardaras and C. Lanzani, "Advanced multimode radio for wireless ; mobile broadband communication," in *2009 European Wireless Technology Conference*, pp. 132–135, Sept 2009.

[14] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent advances in cloud radio access networks: System architectures, key techniques, and open issues," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 2282–2308, thirdquarter 2016.

[15] A. Rath, S. Samantaray, K. S. Bhoi, and P. C. Swain, "Flow forecasting of hirakud reservoir with arima model," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 2952–2960, Aug 2017.

[16] C. Pandey and S. Nemade, "Enhancement of the speech quality by the implementation of second order fast adaptive kalman filter algorithm," in *2014 Annual IEEE India Conference (INDICON)*, pp. 1–6, Dec 2014.

[17] M. Li, J. Zhang, and S. Zheng, "Time series forecasting for density of wood growth ring using arima and neural networks," in *2007 International Conference on Machine Learning and Cybernetics*, vol. 5, pp. 2816–2820, Aug 2007.

[18] M. Usuelli, *R Machine Learning Essentials*. Packt Publishing, 2014.

[19] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, pp. 568–576, Nov 1991.

[20] "Multi-layer neural network." `http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/`.

[21] "Memorizing is not learning!6 tricks to prevent overfitting in machine learning.." `https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting`.

[22] "Recurrent neural networks and lstm." `https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5`.

[23] "Recurrent neural networks tutorial, part 1 introduction to rnns." `http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/`.

[24] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. III–1310–III–1318, JMLR.org, 2013.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, pp. 1735–80, 12 1997.

[26] Y. Wei, Z. Wang, M. Xu, and S. Qiao, "An lstm method for predicting cu splitting in h.264 to hevc transcoding," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, Dec 2017.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using

rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[28] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[29] "Understanding lstm networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[30] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 865–873, April 2015.

[31] "Andrew ng: Deep learning, self-taught learning and unsupervised feature learning." `https://www.youtube.com/watch?v=n1ViNeWhC24`.

[32] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, pp. 47–60, Nov 1996.

[33] S. Banerjee, A. Choudhary, and S. Pal, "Empirical evaluation of k-means, bisecting k-means, fuzzy c-means and genetic k-means clustering algorithms," in *2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pp. 168–172, Dec 2015.

[34] R. L. Thorndike, "Who belongs in the family," *Psychometrika*, pp. 267–276, 1953.

[35] S. Namba, T. Warabino, and S. Kaneko, "Bbu-rrh switching schemes for centralized ran," in *7th International Conference on Communications and Networking in China*, pp. 762–766, Aug 2012.

[36] Y.-S. Chen, W.-L. Chiang, and M.-C. Shih, "A dynamic bburrh mapping scheme using borrow-and-lend approach in cloud radio access networks," *IEEE Systems Journal*, vol. 12, pp. 1632–1643, 2018.

[37] V. N. Ha, L. B. Le, and N. Do, "Energy-efficient coordinated transmission for cloud-rans: Algorithm design and trade-off," in *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, March 2014.

[38] H. Guo, K. Wang, H. Ji, and V. C. M. Leung, "Energy saving in c-ran based on bbu switching scheme," in *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pp. 44–49, Sept 2016.

[39] "Open big data." `https://dandelion.eu/datamine/open-big-data/`.

[40] "Machine learning-based traffic prediction and pattern extraction for dynamic optical routing in sdn mobile metro networks." `http://hdl.handle.net/10589/126134`.

[41] S. T. O. F. G. M. Jiamo Liu, Udita Paul, "K-means based spatial base station clustering for facility location problem in 5g," in *2018 The annual Southern Africa Telecommunication Networks and Applications Conference*, pp. 44–49, Sept 2018.

[42] J. Liu and O. Falowo, "Traffic-aware heuristic bbu-rrh switching scheme to enhance qos and reduce complexity," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sep 2018.