
**Addis Ababa University**

**Addis Ababa Institute of Technology**

**School of Electrical and Computer Engineering**

# Optimization of Artificial Neural Network Using Ant Colony for Workload Load Balancing in Cloud Computing

**By: Wondewosen Feleke**

**December 9, 2020**


# Addis Ababa University

# Addis Ababa Institute of Technology

# School of Electrical and Computer Engineering

## Optimization of Artificial Neural Network Using Ant Colony for Workload Balancing in Cloud Computing.

A thesis submitted to the School of Electrical and Computer Engineering in partial fulfillment of the requirements for the Degree of Master of Science in Computer Engineering.

**By: Wondewosen Feleke**

**Advisor: Dr. Surafel Lemma**

**December 9, 2020**

# Declaration

I, the undersigned, certify that this research work titled Optimization of Artificial Neural Network Using Ant Colony for Workload Balancing in Cloud Computing is my own work. The work has not been presented elsewhere for assessment. Where material has been used from other sources, it has been properly acknowledged.

**Wondewosen Feleke Endeshaw** _____

Signature

Date of submission: December 9, 2020

Place: Addis Ababa

This thesis has been submitted for examination with my approval as a University advisor.

Advisor: **Dr. Surafel Lemma** _____

Signature

# Addis Ababa University

# Addis Ababa Institute of Technology

# School of Electrical and Computer Engineering

## Optimization of Artificial Neural Network Using Ant Colony for Workload Balancing in Cloud Computing

**By: Wondewosen Feleke**

**Approval by Board of Examiners**

| | |
|---|---|
| Dr. Bisrat Derebssa | _____ |
| Dean, School of Electrical and Computer Engineering | Signature |
| Dr. Surafel Lemma | _____ |
| Advisor | Signature |
| Dr. Fitsum Assamnew | _____ |
| Internal Examiner | Signature |
| Dr. Sosina Mengistu | _____ |
| Internal Examiner | Signature |

# Abstract

Cloud computing is an emerging technology and it is growing exponentially. Cloud computing delivers shared computing resources such as servers, storage, databases, networking, applications, platforms and other resources over the internet. These services are provided in the form of Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS). The biggest challenge for cloud data centers and servers is how to handle and service user requests that are arriving frequently from end users.

Load balancing enables to achieve a maximum resource utilization, providing high quality of services, improving performance and prevents the system from failure by distributing workload across one or more computing resources. Existing load balancing approaches could be categorized as **static** or **dynamic** load balancing algorithms. Both static and dynamic load balancing algorithms suffers from virtual machine migration during load balancing process which causes significant performance degradation and overhead.

Recently, resource management driven by forecasting the future workload is proved to be efficient approach to achieve effective dynamic resource scaling, automatic resource provisioning, energy efficient computing, virtual machine migration and scheduling.

Artificial neural network (ANN) trained by Backpropagation (BP) has been used as a load balancing mechanism in cloud computing to distribute workloads. However, ANNs trained with gradient descent algorithms such as popular BP are associated with four distinct limitations; i.e., (1) easily trapped in local minima; (2) slow convergence speed; (3) convergence depends on choice of momentum, learning rate and weights; and, (4) vanishing and exploding gradient problems.

To avoid the above problems a nature inspired swarm intelligence algorithms has been used to train ANN as an alternative training methods. Continuous Ant Colony Optimization ($ACO_R$) is a non-gradient, meta-heuristic swarm intelligence algorithm that has been used to train ANNs.

In this research a load balancing mechanism based on workload forecasting which utilizes historical dataset is proposed. ANN optimized by $ACO_R$ algorithm is used to predict future workload based on historical workload information.

To evaluate the proposed approach, we designed and implemented a system based on two GWA-T-12 Bitbrains **FastStorage** and **Rnd** workload trace datasets. FastStorage, consists of the individual traces of 1,250 VMs and the **rnd** datasets consist of individual traces of 500 virtual machines. The dataset includes information about CPU utilization, memory usage, network usage and storage usage of virtual machine.

The system is tested using four types of information extracted from the datasets using Random forest feature selection algorithm. Based on their scores the top six important features, i.e. (CPU provisioned, CPU usage in megahertz, CPU usage in percentage, memory usage, disk read, and network transmitted throughout) are selected. Using this dataset ten second, fifteen second, and twenty second window size dataset are prepared for evaluation. With respect to prediction window size the performance of the model is tested and the effect of different observation window sizes on the performance of the model is evaluated. The best performance achieved by the proposed system is 96.3% accuracy on Faststorage CPU dataset and 95.2% accuracy on rnd CPU dataset. Finally, we have used other swarm intelligence training mechanism and tested the performance using all versions of the datasets. The best performance is achieved when utilizing Particle swarm (ANN-PSO) on FastStorage CPU datasets. Particle swarm (ANN-PSO) achieved an accuracy of **99.19%** on training and **98.9%** on testing while using FastStorage CPU datasets.

***Keywords:*** Cloud Computing, Load Balancing, Workload Prediction, Artificial Neural Network, Back-propagation, Ant Colony Optimization, Swarm Intelligence.

## Acknowledgement

First and foremost, I would like to thank God Almighty for giving me all the strength and wisdom through the process of researching and writing this thesis.

I express my deepest gratitude to my advisor Dr. Surafel Lemma for his support, guidance and encouragement throughout the period this work was carried out.

Last but not least, I express my very profound gratitude to my family and friends for providing me with continuous support, prayers, and encouragement. This achievement would not have been possible without them. Thank you!

**Table of Contents**

**List of Figures**

**List of Tables**

**List of Algorithms**

**List of Acronyms**

| | |
|---|---|
| $ACO_R$ | Continuous Ant Colony Optimization |
| ANN | Artificial Neural Network |
| IaaS | Infrastructure as a Service |
| PaaS | Platform as a Service |
| SaaS | Software as a Service |
| MSE | Mean Squared Error |
| NIST | United States National Institute for Standards and Technology |
| CSV | Comma Separated Files |
| VMM | Virtual machine monitors |
| CPU | Central Processing Unit |
| ACO | Ant Colony Optimization |
| PSO | Particle Swarm Optimization |
| GA | Genetic Algorithm |
| SLBA | Static Load Balancing Algorithm |
| DLBA | Dynamic Load Balancing Algorithm |

# Chapter One

# Introduction

With the advent and growth of internet technologies (i.e., Networking, computing and processing capacity, storage technologies, mobile technologies and software technologies) cloud computing has increased exponentially in recent years. A widespread adoption across most industries, academia, the government sector and rapidly maturing market are contributing massively to a rapid growth of cloud computing.

With this growth the number of requests processed by the cloud are also increasing rapidly. For example, *as of 2019 Google processes more than 3.5 billion requests per day* and stores *10 exabytes* of data on its cloud datacenters. Cisco predicts the annual global data center traffic will reach 15.3 zettabytes (ZB) by 2020, with 92 percent of all workloads being processed in the cloud by 2020. Similarly, Cisco also forecasts the global consumer cloud storage traffic per user will be 1.7 gigabytes per month by 2020, compared to 513 megabytes per month in 2015 [1]–[3].

The biggest challenge for cloud datacenters and servers is to efficiently handle and service millions of user requests that are arriving frequently from end users. The distribution of resource in the cloud is greatly variable and mapping and assigning user requests to proper cloud resources is a very challenging task which affects the performance of the cloud system[4].

Inappropriately distributed load might result in inadequate resource utilization and causes certain computing nodes being overwhelmed while the other computing resources are underutilized. This problem might cause service degradation and system failure.

Load balancing is the process of distributing workload across one or more servers, datacenters, or other computing resources, in order to achieve a maximum resource utilization and avoid overload.

Efficient load distribution mechanism is vital in order to fairly distribute incoming workloads among computing nodes. It enables to achieve a maximum resource utilization, providing high quality of services, improving performance and prevents the system from failure[5]. Existing load balancing approaches could be categorized as **static** or **dynamic** load balancing algorithms[6]–[9].

During load balancing process these types of algorithms primarily consider current node workload and decide where to execute the task based on workload threshold calculation. Workload threshold indicate the maximum percentage of utilization. If the utilization of a resource in a server exceeds its corresponding threshold values the server is considered as overloaded.

Both static and dynamic load balancing algorithms suffers from virtual machine migration during load balancing process which causes significant performance degradation and overhead. These problems consequently lead a cloud system to become inefficient and costly.

Recently, resource management driven by forecasting the future workload is proved to be efficient way of achieving effective dynamic resource scaling, automatic resource provisioning, energy efficient computing, virtual machine migration and scheduling[10]–[13]. Machine learning algorithms predict future workloads of computing nodes by learning from past workload information with a great accuracy. ANNs are one class of machine learning algorithm.

ANN trained by backpropagation has been used as a load balancing mechanism in cloud computing and to distribute workloads equally across all nodes based on workload prediction[14][15].

However, ANNs trained with gradient descent algorithms such as popular back propagation are associated with four distinct limitations; i.e., (1) easily trapped in local minima; (2) slow convergence speed; (3) convergence greatly depends on choice of momentum, learning rate and

weights; and, (4) vanishing and exploding gradient problems. This motivated the need for alternative training methods.

A swarm intelligence algorithms are nature inspired, non-gradient heuristic algorithms which have taken inspiration from the collective behavior of insects, flocks of birds and fishes. One such swarm intelligence based meta-heuristic approach is $ACO_R$. $ACO_R$ has been used to train artificial neural network to solve the limitations of gradient descent training methods [16], [17]. Ant colony optimization has a powerful ability to search for a globally optimal solution.

This thesis contributes towards the development of load balancing algorithms based on workload prediction utilizing ANN optimized by $ACO_R$. The model analyses previous workloads to predict future load on the computing nodes. We used two dataset from GWA-T-12 Bitbrains **FastStorage** and **Rnd** to create the model of the proposed system and evaluate the model performance[18].

### 1.1 Statement of the Problem

Load balancing is the process of distributing workload across one or more servers, datacenters, or other computing resources. Efficient load distribution mechanism is vital in order to fairly distribute incoming workloads among computing nodes. It enables to achieve a maximum resource utilization, providing high quality of services, improving performance and prevents the system from failure[5]. Existing load balancing approaches could be categorized as **static** or **dynamic** load balancing algorithm.

During load balancing process both static and dynamic algorithms primarily consider current node workload and decide where to execute the task based on workload threshold calculation. Workload threshold indicate the maximum percentage of utilization. Both static and dynamic load balancing algorithms suffers from virtual machine migration problem during load balancing process which

causes significant performance degradation and overhead. These problems consequently lead a cloud system to become inefficient and costly.

Recently, resource management driven by forecasting the future workload is proved to be efficient way of achieving effective dynamic resource scaling, automatic resource provisioning, energy efficient computing, virtual machine migration and scheduling[10]–[13]. Machine learning algorithms predict future workloads of computing nodes by learning from past workload information with a great accuracy.

ANNs are one class of machine learning algorithm. Artificial neural network (ANN) trained by Backpropagation (BP) has been used as a load balancing mechanism in cloud computing to distribute workloads equally across all nodes based on workload prediction [13].

However, ANNs trained with gradient descent algorithms such as popular back propagation (BP) are associated with four distinct limitations; i.e., (1) easily trapped in local minima; (2) slow convergence speed; (3) convergence greatly depends on choice of momentum, learning rate and weights; and, (4) vanishing and exploding gradient problems.

A swarm intelligence algorithms are nature inspired, non-gradient heuristic algorithms which have taken inspiration from the collective behavior of insects, flocks of birds and fishes. One such swarm intelligence based meta-heuristic approach is $ACO_R$. $ACO_R$ has been used to train artificial neural network to solve limitations of gradient descent training methods [16], [17]. Thus, in this research, we propose a load balancing mechanism based on workload forecasting which utilizes ANN optimized by $ACO_R$ and historical resource utilization information. This thesis provides an answer to the following two main questions:

**RQ1**: Is ANN optimized by $ACO_R$ prediction effective to solve the cloud workload balancing problem compared to other machine learning techniques?

**RQ2**: What is the effect of using different window size on the accuracy of the proposed system?

## 1.2 Objectives

### 1.2.1 General objective

The general objective of this research is to design and develop a load balancing algorithm based on ANN optimized using $ACO_R$ prediction method.

### 1.2.2 Specific Objectives

To accomplish the above general objective, the following specific objectives are identified.

- To collect a suitable dataset and identifying suitable data prepossessing and cleaning techniques.
- To design and model the system
- To implement the designed model and train the system
- To evaluate the performance of the designed model
- To compare the performance with related optimization methods.
- To select the better predictor after testing and comparing the different predictors.

## 1.3 Research Methodology

In order to complete this thesis, we followed the following procedures.

### 1.3.1 Literature Review

Literature related to this work have been reviewed to assess the efforts that have been made so far in cloud environment load balancing, workload prediction, artificial neural network, ant colony optimization and other swarm intelligence optimization techniques.

### 1.3.2 Dataset and Preprocessing

Two datasets from GWA-T-12 Bitbrains (FastStorage and Rnd) needed for the experiment were collected and prepared. The data contains information about CPU, memory, disk, and network utilization of virtual machines from a distributed datacenter.

### 1.3.3 System Design and Implementation

The architecture, module, and components of the proposed system were defined. Performance metrics, training approach and parameter tuning were specified. Required tools, algorithms, programing languages were identified, chosen, and studied.

### 1.3.4 Experimentation and Discussion

The actual experiments and the result were analyzed and provided in the form of tables and graphs. The result is discussed in detail and compared with related approaches.

## 1.4 Contribution of the Thesis

Cloud computing is an emerging technology that has significantly transformed the means of computing. Load balancing is an essential component of cloud environment. It prevents any single server from getting overloaded and the system from failure. In cloud environment the problem of load balancing remains challenging. This thesis contributes towards the development of load balancing algorithm based on neural network optimized using $ACO_R$. The proposed approach predicts the workload and distribute the load fairly among computing nodes. The performance of the model is evaluated empirically with the use of two datasets and compared against related swarm intelligence optimization algorithms.

## 1.5 Thesis Organization

The rest of this thesis is organized as follows. Chapter Two describes the background of artificial neural network, history and approaches of ant colony optimization and the basics of cloud

computing. Chapter Three presents a literature review, which gives a detailed overview of the previous work and techniques employed in cloud computing load balancing problems. Chapter Four presents proposed methodology and the architectural framework used to develop the algorithm. Chapter Five briefly presents result obtained through experimental evaluation. Finally, Chapter Six concludes the thesis with discussion about the result attained through experimental evaluation and concisely presents the future research direction.

# Chapter Two

## Theoretical Background

In this chapter, the architecture of cloud computing, essential characteristics, deployment models and service models of cloud are discussed. The details of load balancing problems, different load balancing algorithms and techniques are also elaborated. Next, the particulars of artificial neural network, their success, limitation and problems are also presented. Finally, swarm intelligence as an optimization techniques are also discussed in details.

### 2.1 Cloud Computing

Basically cloud computing is the delivery of shared computing resources such as servers, storage, databases, networking, applications, platforms and more over the cloud, i.e., the internet[19], [20]. It enables ubiquitous access to shared pools of configurable system resources and higher-level services that can be rapidly provisioned. These services are often offered by cloud computing Companies (e.g., Amazon, Google, Microsoft, etc.) and charge for services based on usage.

There is no standard definition of cloud computing. Companies offering cloud service define cloud computing in different ways. A comprehensive definition of cloud computing is provided by United States National Institute for Standards and Technology (NIST), Department of Commerce. According to NIST cloud computing is defined as [21]:

*"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models".*

The NIST definition of cloud computing identifies five essential characteristics as illustrated in Figure 2.1.

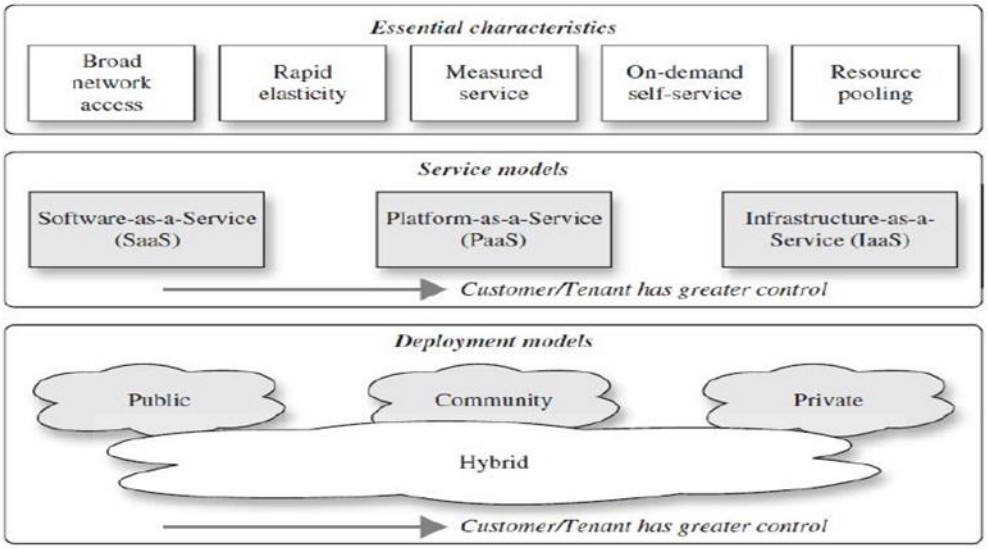


Figure 2.1: NIST Cloud Computing Model (adopted from [20])

### 2.1.1 Essential Characteristics of Cloud Computing

As stated in the NIST cloud definition, every cloud environment exhibits the following five characteristics [21].

i. ***On-demand self-service***: Customers can unilaterally provision computing resources such as server time and network storage automatically according to their demand and pay for it without sales agent.

ii. ***Broad network access.*** Customers access cloud services over the network through different platforms (e.g., mobile phones, tablets, laptops, and workstations) from anywhere and at any time.

iii. ***Resource pooling.*** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

iv. ***Rapid elasticity.*** Resource usage can be elastically provisioned and released automatically according to customer needs.

v. ***Measured service.*** Cloud system providers automatically control, monitor and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

### 2.1.2 Cloud Deployment Model

Cloud computing resources can be deployed in four different ways, namely; private, public, hybrid, and community as shown in Figure 2.1.

(a) ***Private Cloud***: This deployment model has been built, is maintained and operated specifically to provide the services solely for a single organization. It may be managed by the organization or a third party and may exist on premise or off premise[21]. Requires a large amount of investment, needs allocation of space, hardware, and environmental controls. This deployment model is suitable for large enterprises[22]–[24].

(b) ***Public Cloud:*** In this deployment model the cloud infrastructure is available to the general public on a commercial basis by a cloud service provider (an organization selling cloud services). With a public cloud, all hardware, software, and other supporting infrastructure are owned and managed by the cloud provider. The client access these services and manage their own account using a web browser. It is very cost effective for small and midsize business to deploy IT solutions [20-22]. E.g. Amazon EC2, Google clouds, Microsoft Azure.

(c) ***Community cloud:*** Community cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns such as mission, security requirements, policy, and compliance considerations [22]. The costs are spread over fewer users than a public cloud (but more than a private cloud). It may be managed by the organizations or a third party and may exist on premise or off premise[23], [25].

(d) ***Hybrid Cloud***: The hybrid cloud is a composition of two or more clouds (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability[21]. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service[22], [23].

### 2.1.3 Types of Cloud Service Delivery Models

Cloud computing service models always offered through three different models, IaaS, SaaS and PaaS.

(a) ***Infrastructure as a Service (IaaS)***: IaaS is the most basic form of cloud computing service. In this service model the customers can rent IT infrastructure such as servers and virtual machines (VMs), datacenter space, storage, networks, firewalls, operating systems, and various utility applications from a cloud provider on a pay-as-you-go basis. Afterwards, it is a customer responsibility to manage every aspect of the hardware, from the operating system (OS) through to the applications that are built and run on it. Applications are developed either by the customer, or by another vendor [23], [25].

(b) ***Software as a Service (SaaS)***: It is a delivery of software applications over the internet. In this service model a cloud service provider hosts and manage the software applications and necessary infrastructures, and implement maintenance in case of service failure. Customers

don’t need to install, manage or buy hardware for software applications, all they have to do is connect and use it. It is usually on demand and can be accessed with a web browser on a phone, tablet, or PC from anywhere at any time (e.g., email)[21].

(c) ***Platform as a service (PaaS)***: PaaS is a cloud computing service model that delivers an on demand programming environment or platform for developing, testing, delivering, and managing software applications [26]. This platform is generated for the programmers to create, test, run and host the applications. Amazon Web Service, Google Apps Engine (GAE), Windows Azure, IBM BlueMix are the examples of PaaS.

### 2.1.4 Virtualization and Hypervisor in Cloud Computing

Virtualization plays a very important role in a cloud computing. Most clouds are built on virtualized infrastructure technology and cloud computing deeply relies on virtualization technology[27]. Without virtualization technology the implementation of cloud computing could not be possible. In computing, virtualization refers to a technique of creating virtual (rather than actual) version of computing resources such as hardware (e.g. servers, storage devices), software platforms (operating system, applications), network resources and other more computing resources [28].

The main aim of the virtualization is to run multiple operating systems and applications on a single machine by sharing all the resources that belong to the hardware; mainly the server [29]. This enables the cloud more scalable, efficient and economical [30].

One of the most important operation performed in cloud computing is load balancing. Load balancing is done using virtual machines so that each node has similar load.

### 2.1.5 Hypervisor

Hypervisor also referred to as virtual machine monitor (VMM) is an important component in the virtualization, and plays a key role in a cloud system as a whole. It is a program that allows multiple operating systems to share a single hardware host. It provides system resources to virtual machines and enforces access control policies[31]. There are two types of hypervisor: Type 1 and Type 2 hypervisors. As of 2020, the most popular Type1 hypervisors include, Citrix Xen Server, Microsoft Hyper-V, Red Hat KVM, and VMware vSphere hypervisors. Figure 2.2 shows the architecture of virtualization with hypervisor [32], [33] .

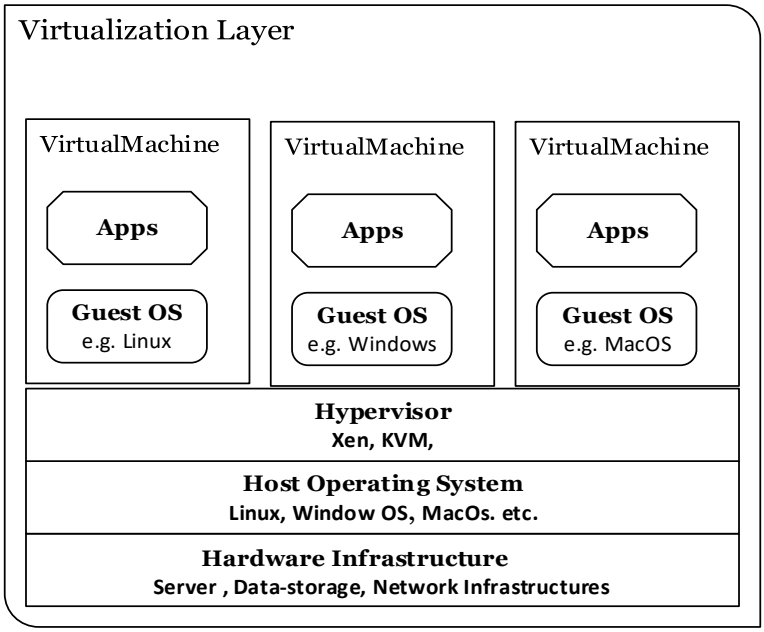


Figure 2.2: Virtualization in Cloud Computing

Commonly, cloud virtualization is categorized into four. These are: hardware, software, storage and network virtualization[32]–[34]. Table 2.1. Show summary of virtualization types.

| Hardware Virtualization | Software Virtualization | Storage Virtualization | Network Virtualization |
|---|---|---|---|
| ▪ Full<br>▪ Para<br>▪ Emulation | ▪ Application<br>▪ OS<br>▪ Service | ▪ Block<br>▪ File | ▪ External<br>▪ Internal |

Table 2:1: Virtualization Types

### 2.1.6 Benefits and Issues of Cloud Computing

Cloud computing offers many benefits both for cloud clients (industries, IT related businesses, consumers) and for companies that provide cloud services. According to survey by Dell, "Companies that invest in cloud and big data enjoy up to 53% faster growth than their competitors"[34]. For consumers, the following are a list of the major advantages offered by cloud computing;

- Cost saving
- Efficiency
- Flexibility
- Performance and scalability.
- Backup and recovery
- Increase Security
- Mobility

Despite the advantage gained using cloud computing there are also issues related to cloud computing. In general, these concerns are classified into six categories[35]:

- Data management and resource allocation,
- Security and privacy
- Load balancing
- Scalability and availability
- Migration and Compatibility
- Communication and interoperability

## 2.2 Load Balancing in Cloud Computing

In cloud computing, load balancing refers to the process of virtual machine workload distribution across multiple servers in order to improve overall system performance and minimize response

time[36]. Load balancing is one of performance enhancement technique employed in a cloud computing. Load balancer sits in between client and backend servers monitoring user requests. Figure 2.3 shows a diagram of load balancer in cloud computing. Load balancing provides a mechanism to distribute the load of virtual machines across all nodes to improve resources and service utilization, and provide high satisfaction to users[37], [38]. In cloud computing, the following are some of the main goals of load balancing [38], [39].

- To improve overall system performance
- To increase quality of service and high user satisfaction
- Prevent degradation of service and system failure
- Maintain system stability in emergency situations such as a sudden surge of arrivals.
- Optimize resource utilization.
- Cost effectiveness
- Improve scalability and flexibility

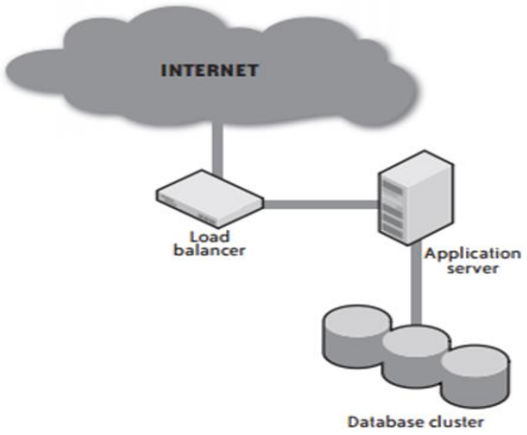


Figure 2.3: Block diagram of cloud load balancer

There are two categories of load balancing : **static** and **dynamic** load balancing algorithms [25], [37].

### i. Static Load balancing Approaches

This algorithm depends on prior knowledge of the system resources. It does not require information about the present state of the node. Therefore, the decision of load distribution doesn't depend on the current state of the system[25]. The problem is, this type of algorithms are not flexible and cannot match the dynamic changes to the workload during execution time. They are suitable for homogenous and stable environment.

### ii. Dynamic Load Balancing Algorithms

The present state of the system is required to make a decision on load distribution. They are adaptable and adjust their performance in accordance to the system state and allow to move tasks from overloaded machine to under loaded machine in real time[25]. Dynamic load balancing are further categorized as **distributed** and **centralized** (non-distributed) algorithms. Figure 2.4 shows a general classification of cloud load balancing algorithms.

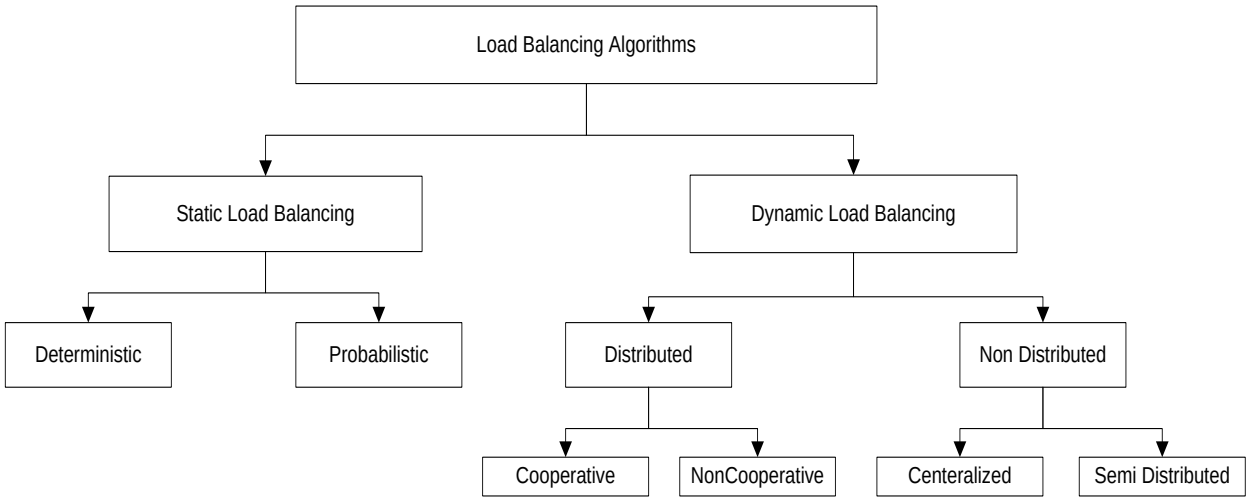


Figure 2.4: Classification of Load Balancing Algorithms

### 2.2.1 Policy in Load Balancing Algorithm

A load balancing algorithm has four policy components: - transfer, selection, location, and information polices[40]. Figure 2.5 shows the interaction among components of dynamic load balancing algorithms.

**(a)** ***Transfer Policy*****:** transfer policy determines if the task should be executed on the local node or remotely. Based on the threshold, the transfer policy decides whether or not a processor or node is a sender or a receiver. If the load on a node is above the threshold, then it acts as a sender for new tasks. If the machine's load is below the threshold then it acts as a receiver.

**(b)** ***Selection Policy*****:** based on the transfer policy decision, if the node is chosen to be the sender; the selection policy selects the task to be transferred.

**(c)** ***Information Policy:*** The information policy is accountable for collecting information about the state of the nodes in the network[36], [41], [42]. It decides what and when the information has to be collected. The information can be gathered in three techniques: demand driven, periodic and state change driven.

**(d)** ***Location Policy:*** Location policy is responsible for selecting the best node among all available nodes in the system to share the load. To find a suitable node, the policy either selects a node randomly, uses polling, or negotiation techniques.

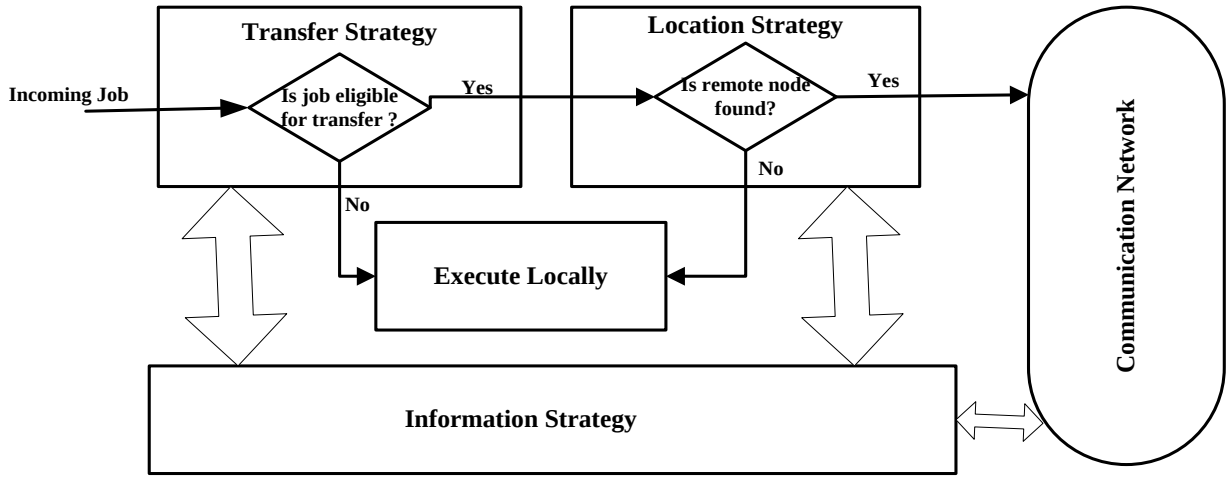


Figure 2.5: Interaction between dynamic load balancing algorithms([37])

### 2.2.2 Comparison between Static and Dynamic algorithms

Static and dynamic load balancing algorithm can be compared based on the qualitative actors listed in Table 2.2.

Table 2:2: Summary of Static and Dynamic Algorithms

| S.NO | Factor | Load Balancing | |
|---|---|---|---|
| | | **Static load balancing algorithm** | **Dynamic load balancing algorithm** |
| 1 | Nature | Workload is assigned at compile time | Workload is assigned at run time |
| 2 | Overhead involved | Little overhead | Greater overhead due to process redistribution |
| 3 | Resource utilization | Lesser utilization | Greater utilization |
| 4 | Processor thrashing | No thrashing | Considerable thrashing |
| 5 | Predictability | Easy to predict | Difficult to predict |
| 6 | Adaptability | Less adaptive | More adaptive |
| 7 | Reliability | Less | More |
| 8 | Response time | Short | Longer |
| 9 | Stability | More | Less |
| 10 | Complexity | Less | More |
| 11 | Cost | Less | More |

## 2.3 Artificial Neural Network

Artificial Neural Network is a computational model that imitates the human brain system. The original concept of ANNs dates back to 1943 and was initiated after the introduction of a simplified model of artificial neural network by McCulloch and Pitts[43]. They describe the concept of a neuron, a single cell living in a network of cells that receives inputs, processes those inputs, and generates an output. ANN have been developed as generalizations of mathematical models of biological nervous systems.

### 2.3.1 Biological Neuron

The brain is principally composed of about **10 billion** neurons and trillions of synaptic interconnections. Each neuron receives **electrochemical inputs** from other neurons through a web of input and output terminals, or channels called **dendrites**. A **neuron** is a special biological cell that accepts, processes and transmits information through electrical and chemical signals. Figure 2.6 shows a schematic diagram of a biological neuron.

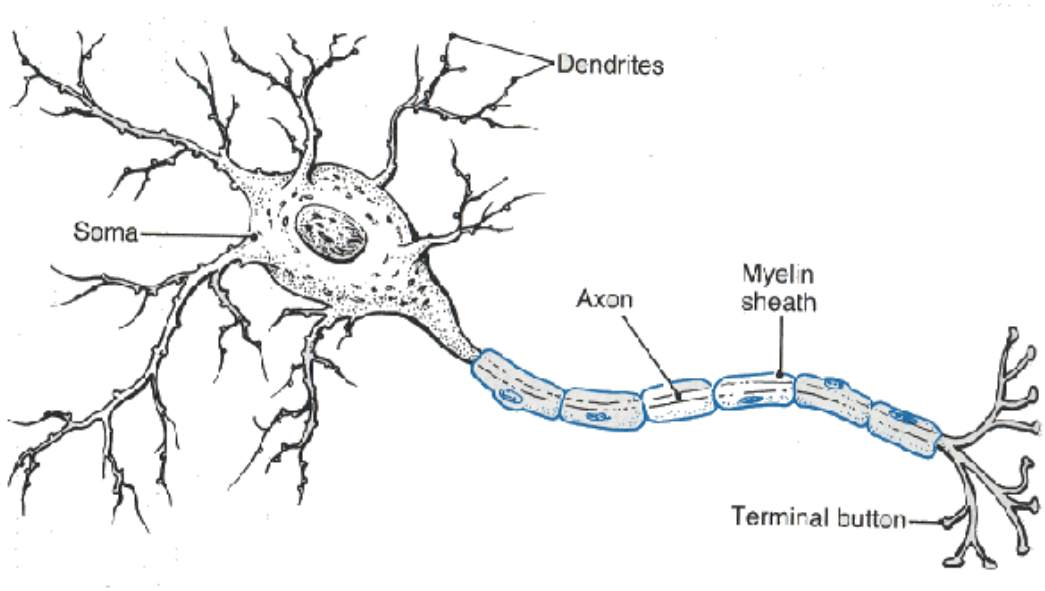


Figure 2.6: Neuron Diagram (Carlson, 1992)

This is the model on which ANNs are based. Although they are inspired by biological neurons they do differ from their biological counterparts in many ways. The human brain is extremely complex, with billions of neurons and trillions of synaptic interconnections. Modeling the sophisticated human brain with artificial neural network is very difficult.

ANNs tries to approximate the behavior of human brain in much smaller and simpler scale. Even though they mimic the behavior of the human brain is such small scale, they have shown a great success in solving problems which are easy for human but difficult for traditional computer, such as image recognition and predictions. Recently, they are being extensively employed in real world decision making domains including business applications like data mining, pattern identification, sales prediction and forecasting, risk management etc. Figure 2.7 represents the comparison between biological neural network and artificial neural network.

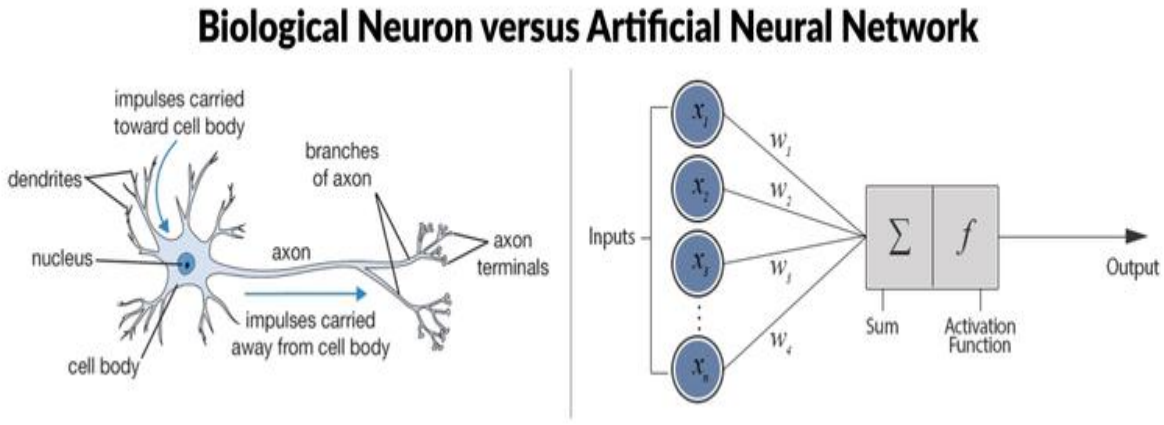


Figure 2.7: Biological Neuron vs. Artificial Neural Network ([43])

### 2.3.2 Neural Network

The fundamental processing element of a neural network is a neuron. A neuron receives one or more inputs $x_1 \ldots x_n$, from other neurons and **sums** them to produce an **output**. Each of these

inputs are multiplied by a connection **weight** *w*. In the simplest case, these products are simply **summed**, fed through a **transfer function** *f* to generate a result, and then **output** ***(O)*** [44]–[46]. The neuron output signal **O** is given by the following relationship:

$$O = f(net) = f\left(\sum_{j=1}^{n} w_j x_j\right) \qquad (1)$$

Where $w_j$ is the weight vector, and the function ***f(net)*** is referred to as an **activation** (transfer) function. The variable ***net*** is defined as a scalar product of the weight and input vectors,

$$\text{net} = w^T x = w_1 x_1 + \cdots + w_n x_n \qquad (2)$$

Where $T$ is the transpose of a matrix, and, in the simplest case, the output value $O$ is computed as

$$O = f(net) = \begin{cases} 1 & if \;\; w^T \geqslant \theta \\ 0 & otherwise \end{cases}$$

Where $\theta$ is called the threshold level; and this type of node is called a *linear threshold unit*. Figure 2.8 shows a fundamental representation of an artificial neuron

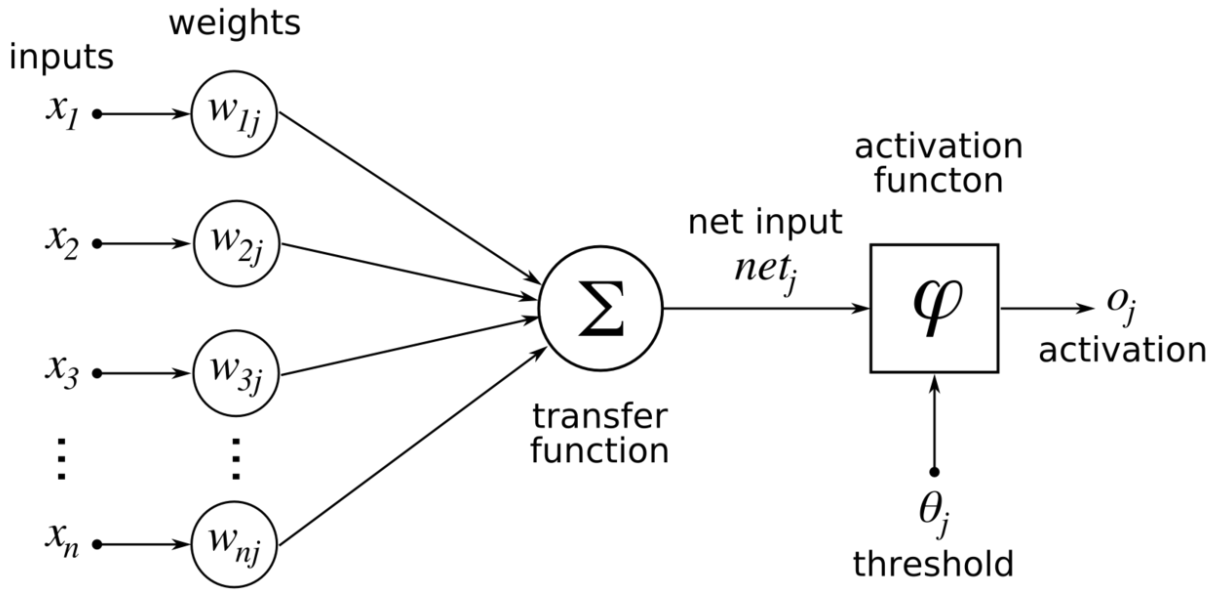


Figure 2.8: Fundamental representation of an artificial neuron ([43])

### 2.3.3 Neural Network Architecture

At its foundation, the architecture of artificial neural network consists of a single neuron. One of the first neural networks to be developed was the **perceptron**. As shown in the Figure 2.9 the perceptron is a very simple neural network with only **one neuron** and a single layer neural network. A single neuron layer is insufficient for many practical problems, and networks with a large number of nodes are frequently used. Obviously, more than two artificial neurons are used in solving real life problems [44].

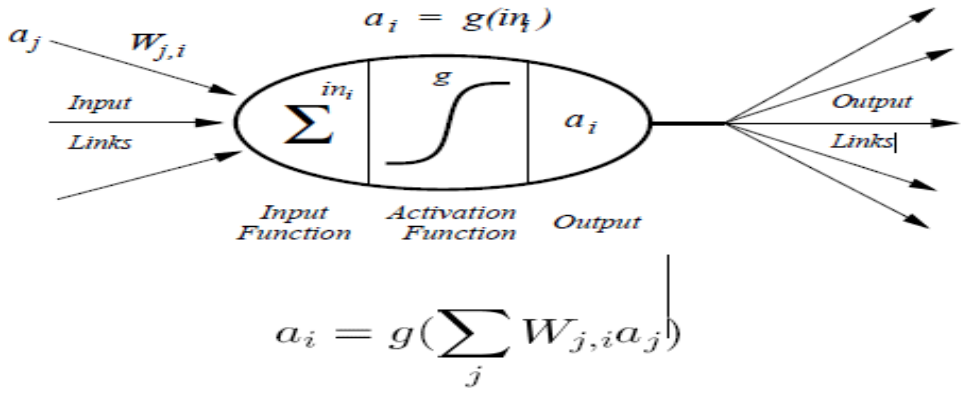


Figure 2.9: Basics of artificial neuron

When two or more artificial neurons combined together they create an artificial neural network. Usually, the architecture of artificial neural network structured in neuron layers: input, hidden, and output layers. Figure 2.10. Illustrate a multilayer artificial neural network[44].

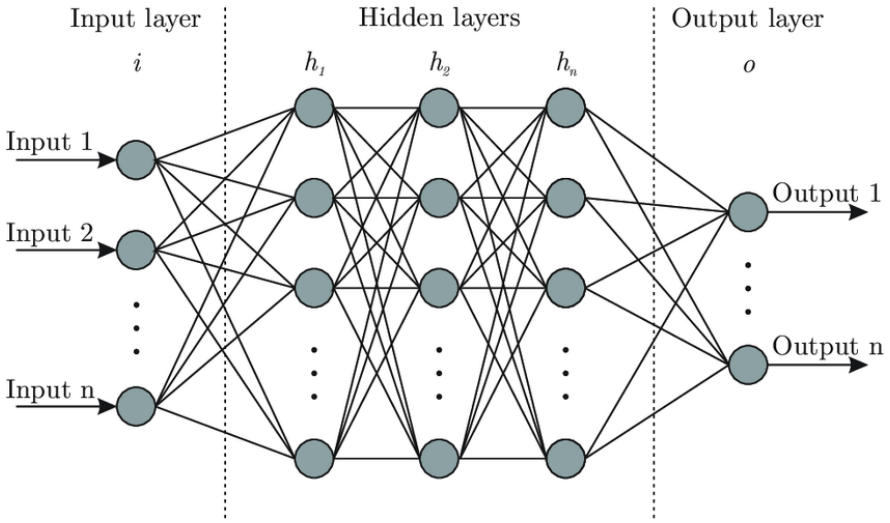


Figure 2.10: Multi-layer artificial neural network

The section below will explain the components of a perceptron. These components are also valid for other arrangements (e.g. feed forward, recurrent topology) of an artificial neural network. Generally, the artificial neuron is made up of the following main components.

i. **Input and output node**
ii. **Connections**
iii. **Weight matrix and bias**
iv. **Summation function**
v. **Transfer or activation function**

The input nodes receive a certain number of inputs from some other nodes, or from an external source and pass on the information to the hidden nodes. The output nodes are collectively referred to as the output layer and are responsible for computation and transferring information from the network to the outside world.

The network connections, connect one neuron in one layer to another neuron in another layer or the same layer. A connection always has a weight value associated with it. The goal of the training is to update this weight value to decrease the loss (error). Weights in an ANN are the most

important factor in converting an input to impact the output. Weights are numerical parameters which determine how strongly each of the neurons affect the other. A weight is multiplied to the input to add up to form the output. A bias (offset) is an extra input to neurons, and it is always 1, and has its own connection weight. The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$\boldsymbol{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}$$

The inputs and corresponding weights are vectors which can be represented as ($i_1$, $i_2$ . . . in) and ($w_1$, $w_2$ . . . $w_n$).

**Activation (Transfer) Function** is a very important component of artificial neural network. Activation functions are mathematical equations that determine the output of a neural network. The result of the summation function can be any value ranging from negative infinity to positive infinity. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

There are many activation functions used in artificial neural network. Figure 2.11 shows the most common types of activation function used in neural networks. **Sigmoid** activation function is the most widely used activation function in neural networks. The following are the most popular type's activation function:

- Sigmoid or logistics activation function
- Tanh- Hyperbolic tangent activation function
- Softmax activation function
- Rectified linear unit – ReLU

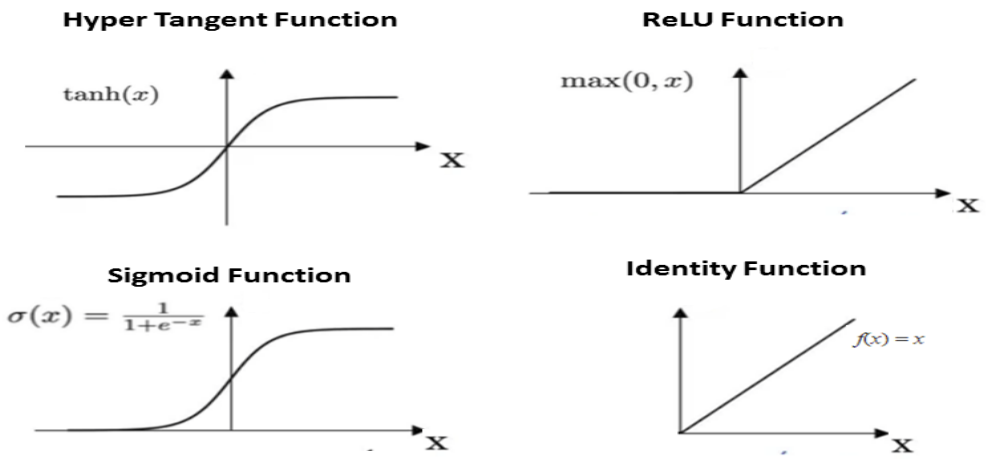


Figure 2.11: Types of Activation Functions

## 2.4 Training in Artificial Neural Network

The computational process of neural network begins by receiving multiple inputs from the outside world or from other neurons. Each of these inputs is multiplied by a connection weight. These products are summed, fed to a transfer function (activation function) to generate the final result, and this result is sent as output. At the beginning, the initial weight values are chosen randomly. These initial weight values can be any value. In order to get the desired output, choosing optimal set of weights value is vital. Training a neural network is actually a process of finding the optimal set of weights for the links between neurons, which will make the neural network produce the correct output corresponding to the given input. After this training, the network will be expected to exhibit 'intelligence' by performing what it has been trained to do, based on the input patterns. There are two approaches to training;

- Supervised and
- Unsupervised

**(a) Supervised Training:** In this type of training method, both the inputs and their prior known corresponding desired outputs are presented to the network. The network then

processes the inputs and during the running process the resulting outputs are continuously compared with the desired data. The errors between the actual output and desired output are then propagated back to adjust the connection weights. These errors finally used to measure the difference between the actual output and the target output. After a certain number of iterations, if the result obtained is the closest match to the desired output, the final connection weights are then established. This process is called training. During training, the set of data which enables the training is called the training set. Back propagation training algorithm is the most popular neural network training algorithm which belongs to this category.

**(b) Unsupervised Training:** in this type of training, only the set of inputs is presented to the network. Unlike supervised training, the desired output associated with each input parameter is unknown and not provided to the network. The neural network adjusts their weighs without external influence. The network explores, try, learn, organize, adjust network's weights and biases and derive conclusions by itself without external influence. The Kohonen neural network training algorithm belongs to this category.

**(c) Reinforcement Learning:** The reinforcement learning algorithm also called as graded learning mimic in a way the adjusting behavior of humans which interacting with a given physical environment. The network connections are modified according to feedback information provided to the network by its environment.

### 2.4.1 Backpropagation Training Algorithm

The backpropagation (BP), also known as the error back propagation algorithm, is a gradient descent, supervised training approach for artificial neural network. It is the most commonly and

widely used training approach in neural networks. The algorithm learns by calculating the errors of the output layer and adjusting the weights of the network in order to minimize the mean square error (MSE) between the actual output and the desired output. The performance (prediction accuracy) of a neural network is proportion to the training method used to train the network. The sequence of steps implemented in the training is presented below.

i. Provide inputs to the input layer and assign weights and biases to the inputs randomly
ii. For each input, compute the activation of internal and output units (Forward propagation)
iii. For each output, Compute error deviation as derivative between output activation and target output
iv. Backpropagate summed product of the weights and errors to hidden layer(Backpropagation)
v. Adjust the connection weights according to error and after every such epoch, compute the error.
vi. Stop when the error falls below a predetermined threshold or number of iteration exceeds maximum number of epochs.

### 2.4.2 Problems with Backpropagation

Although back propagation algorithm is flexible, has high learning capabilities and mathematical elegance, it suffers from various problems. The main problems associated with BP includes network paralysis, local minima in their learning instances and slow convergence rate[17], [47], [48].

### 2.4.3 Swarm Intelligence algorithm for training artificial neural network

To overcome the problem of BP training, many researchers have adopted several meta-heuristic global optimization algorithms. Meta-heuristic algorithms aim to find global or near-optimal solutions at a reasonable computational cost to overcome the local minima problem [49]. Meta-heuristic algorithms are nature-inspired algorithms that aim to find a good optimal solution in a

reasonable computational cost without guaranteeing feasibility or optimality. The most popular meta-heuristic algorithms are simulated annealing, ant colony optimization, Genetic algorithm (GA), Particle swarm intelligence (PSO), Tabu search, and Ant colony optimization algorithm (ACO)[47], [50], [51].

- **Genetic algorithm**

Genetic algorithm (GA) is a population based algorithm inspired by Charles Darwin natural or genetic selection and rejection process[52]. It imitates the natural evolution process of reproduction, mutation, crossover and selection of genetic chromosomes. GA is a meta-heuristic global search algorithm and has been shown to perform well in obtaining global solutions [53]. GA has been used in artificial neural network training and weight optimization[54][48].

- **Particle Swarm Optimization**

PSO is another population based global search algorithm and has been used in artificial neural network training[49]. The general idea of PSO is inspired by a social behavior of fish schooling or large flying flocks of birds searching for food sources through cooperative work of the population. In PSO algorithm, a flock of birds, known as particles (candidate solutions) search the best solution in a search space given the particles initial random position and velocities[55].

- **Hybrid Algorithms**

In these hybrid backpropagation-genetic algorithm (BP-GA) and backpropagation-particle swarm optimization (BP-PSO), the GA and PSO are applied to initialize and adjust the weights of BP network to escape from the local minima[47], [56]. The meta-heuristic algorithm has more potential than the local search algorithms for avoiding local minima since it is capable to explore different area of search space simultaneously.

### 2.4.4 Training Artificial Neural Network with Continuous Ant Colony Optimization ($ACO_R$)

The original Ant Colony Optimization (ACO) is a discrete met-heuristic optimization approach inspired by a foraging behavior of biological ants. Biological ants utilize a chemical called **pheromone** which enables ants to find the shortest paths between food source and their nest.

ACO has been used to solve difficult optimization problems such as: Traveling salesman problem, routing in telecommunication networks, traffic control, forex forecasting and many more which are a problem with a discrete solution space [50]. The original ant colony optimization was proposed by Marco Dorigo in 1992 in his PHD thesis[51]. Following the original ACO (Ant System) many variants of ACO have been developed, including AntNet, Ant Colony System, Max-Min Ant System and the ANTS Algorithm, Continuous ACO ($ACO_R$) and many more[57], [58]. The original ACO system consist of three main components (ant, pheromone, and daemon action)[50]. Figure 2.16 shows the framework and components of ACO.

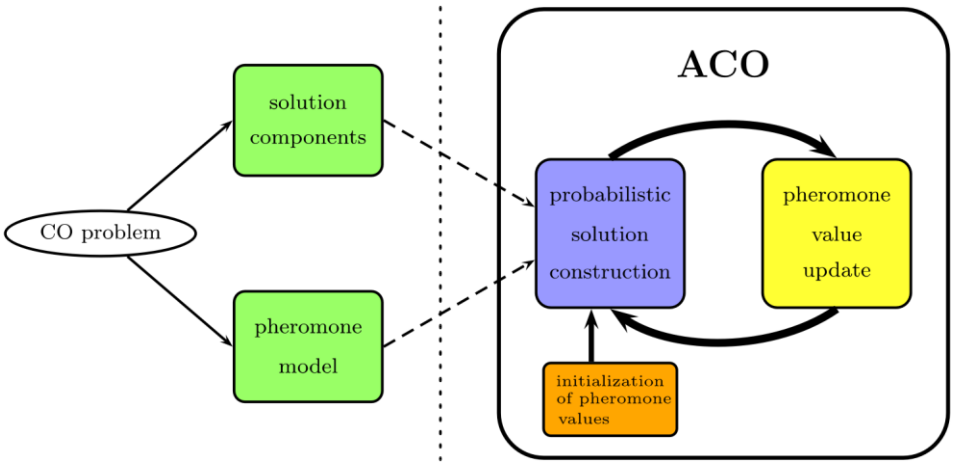


Figure 2.12: The working of ACO framework [59]

The idea of employing $ACO_R$ for neural network optimization was proposed by Krzysztof Socha and Christian Blum after adapting discreet ACO to continuous optimization. The authors applied $ACO_R$ to train feed-forward neural networks to solve the problem of patter classification in the medical field[55]. Training process in a neural network is a continuous optimization problem[17].

In the continuous optimization problem, there are infinite solution components to choose from. Generally, the $ACO_R$ is used to find optimal initial weight value for backpropagation which is applied when training is terminated. $ACO_R$ does not use gradient information at all, and the only problem-dependent information that it has access to is the fitness function.

Many variants of $ACO_R$ are used to solve a continuous optimization problems. In this research, the original $ACO_R$ is utilized and investigated for the problem of load balancing in cloud computing.

# Chapter Three

# Literature Review

This chapter presents a survey of existing load balancing algorithms in cloud computing. Commonly they are categorized as ***static*** and ***dynamic*** load balancing algorithms.

***Round Robin (RR)*:** is commonly used and the simplest load balancing algorithm used in cloud computing. It is a static algorithm, which distributed the load to the nodes (VM) in a ring manner [60]. This algorithm divides the incoming traffic equivalently between nodes without any other consideration such as current load of the node. Although RR is a simplest algorithm to implement it has many drawbacks. It assigns tasks to the node irrespective of whether it is heavily loaded or lightly loaded which causes load imbalance in the system.

B.Radojevic and M. Zagar [61] proposed a static algorithm called **CLBDM (Central Load Balancing Decision Model).** This algorithm was proposed to resolve the drawback of the Round Robin algorithm. CLBDM is an upgrade of the Round Robin Algorithm which is based on session switching at the application layer.

To achieve better executing efficiency, in S.-C. Wang et al. Proposed a two-phase load balancing that merged OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms. The OLB algorithm puts each and every node in working condition so as to achieve the goal of cloud computing whereas LBNN scheduling algorithm is used to minimize the execution time of the tasks on a node which reduces overall completion time. Combining these two algorithms help achieve efficient utilization of all resources and improves the performance efficiency of the network of multiple processors as whole.

Dhinesh Babu L.D. and P. Venkata Krishna [62] proposed a honeybee foraging behavior algorithm. This algorithm is inspired by the biological foraging behavior of honey bees for finding

and reaping food. The algorithm group the servers under virtual servers (VS) wherein each virtual server has its own virtual service queue. Each server estimates a profit corresponding to the amount of time that the CPU spends on the processing of a request. The performance of the system is enhanced with increased system diversity. The main problem is that throughput is not increased with an increase in system size.

***Biased random sampling*** load balancing algorithm is a distributed and scalable load balancing approach that uses a random sampling approach to distribute workload across multiple nodes in the system [63]. The performance of the system is improved with high and similar population of resources, thus resulting in an increased throughput by effectively utilizing the increased system resources.

***Equal Spread Current Execution (ESCE):*** is an algorithm that maintains an index, table of VMs and the current work load assigned to the VM [63]. It assigns a client request to the least loaded virtual machines after it scans the index table for the virtual machine with the minimum number of current allocations.

***Throttled Load Balancing Algorithm (TLB)*** load balancing algorithm also track down virtual machines state by maintain an index table of virtual machines as well as their states[64]. The load balancer allocates the task to the virtual machine after scanning the index table, and update the index table of virtual machines.

GA is an efficient meta-heuristic algorithm known to solve various combinatorial optimization problems. Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S [65] proposed a load balancing algorithm based genetic algorithm for cloud computing optimization. GA uses a mechanism of natural selection strategy which composed of three operations: selection, crossover, and mutation operation to efficiently traverse the solution search space.

**Nada M.Sallami, A. daoud and S. Alousi**. [14] proposed load balance technique based on artificial neural network. The algorithm distributes workload across cloud nodes using the basic feed forward artificial neural network (ANN) trained by back-propagation (BP) learning method. ANN predicts the demand and thus allocates resources according to that demand. However, this approach suffers from local optima problem caused by the back propagation learning method.

**Ren Gao and Juebo Wu** [66] proposed a dynamic load balancing approach via ant colony optimization (ACO) for balancing the workload in a cloud computing platform. The advantage of this algorithm is its suitability for dynamically changing cloud environment and achieve better performance than round robin approach. However, it suffers from network overhead caused by a number of ants generated by the algorithm.

**Other Load Balancing Algorithms**

- ***Weighted Round Robin Algorithm*:** This algorithm tried to fix the problem of round robin by assigning weigh to nodes. During task allocation the weighted round considers the resource capabilities of the VMs and assigns based on the weight given to each of the virtual machines.
- ***Min-Min Load Balancing:*** Based on prior information the job is mapped to the appropriate node. The limitation of this algorithm is it causes starvation since the priority always give to the shortest jobs first.

- ***Max-Min Load balancing:*** works on contrary to min-min approach. This algorithm also causes starvation because the priority is always given to the longest jobs first.

- ***Opportunistic Load Balancing algorithm (OLB):*** OLB is a static load balancing algorithm that has the goal of keeping each node in the cloud busy regardless of the current workload of each node however it causes the tasks to be processed in slower manner and will cause bottlenecks.

Table 3.1 below shows a comparison of different load balancing algorithms based on different metrics

Table 3:1: Comparison of Load Balancing Algorithms

| Algorithm | Nature | Response Time | Network-Overhead | Complexity | Fault Tolerance |
|---|---|---|---|---|---|
| Round Robin | static | Slow | No | Low | no |
| LBMM | static | Slow | No | Low | no |
| Opportunistic LB | static | Slow | No | Low | no |
| Two-Phase (OLB+ LBMM) | static | Fast | No | High | Yes |
| Fuzzy logic | dynamic | Fast | No | Yes | yes |
| Stochastic Hill Climbing | static | Fast | No | Low | no |
| Genetic Algorithm | dynamic | Moderate | No | moderate | no |
| Throttled (TLB) | dynamic | Fast | No | Low | no |
| Biased Random Sampling | dynamic | Fast | No | Low | no |
| Honeybee Foraging | dynamic | Fast | Yes | Low | Yes |
| Active Clustering | static | Yes | No | Low | No |
| CLBDM | static | Slow | Yes | Low | No |
| Ant Colony | dynamic | Fast | Yes | NO | Yes |

- **Workload Prediction in Cloud computing**

Many of the researchers working in the area of workload prediction proposed different workload prediction approaches for different problems. Prediction can be done using Linear Regression, ANN, or K-NN. This predicted value of utilization is further used to find out the overloaded computing nodes, if predicted value is larger than a certain threshold then node is considered as overloaded otherwise it can accommodate new VMs. Cloud workload prediction can be used for different purposes load balancing, energy efficient computing, virtual machine migration, resource provisioning and profiling more generally, to manage and predict performance and Quality of Service (QoS).

Islam et al. [67], applied neural networks for resource allocation and implemented adaptive management of resources in the cloud system. They trained the neural network using the back-propagation algorithm and their experimental results shown that Neural Network-based predictions have less percentage error than LR.

Linear regression (LR) was applied to develop LiRCUP for prediction, which took historical CPU utilization as input and established a linear relationship between the future and current CPU utilization. Coefficient parameters of regression equations were initialized randomly and dependent variables were considered as expected utilization while independent variables represented current utilization values[68] .

K-NN was applied for the forecasting of resource requirements using historical data. K-NN–UP algorithm fetched historical dataset as an input and predicted the CPU utilization. The algorithm used the past and current CPU utilization as the input dataset. The CPU utilization was the output data in the next time instance to forecast CPU usage in each PM. This predicted CPU utilization values were further used for detecting overloaded and underutilized PMs [69].

RL based learning agent observed the PM status and collects the current total utilization of PM[70]. An approach was presented to monitor and collected PM resource utilization using the local agent and the global agent respectively to develop migration mechanisms [71].

Duggan et al.[72] Proposed a live migration strategy, Reinforcement Learning network aware Live Migration (RLLM) that was network-aware to monitor the BW demand and performed proper action based on experience when network congestion occurs. Their system performed as a decision support system in which an agent was made able to schedule VM migration by learning an optimized time.

The migration process depends on the BW usage in a cloud datacenter. Their experiments' results described that an agent can learn available BW at the peak network saturation and be capable to schedule VMs for migration from underutilized PM at the appropriate time using available BW in a cloud datacenter.

In some research, support vector regression (SVR) was also used as a learning approach and SVM classify data and make predictions effectively on overlap regions of two classes, it is widely used for forecasting and classification [73].

CPU utilization values were used for training prediction models to forecast the upcoming resource utilization. The network received current and previous CPU utilization values as input and calculated the output as predicted utilization. In the training phase, this output was compared with the real CPU traces and the difference was considered as error. That error was propagated back and weights were adjusted to minimize the prediction error[74].

A resource monitor was used to collect the utilization of PM resources on a different time interval. Further, these are stored in a buffer and read using the preprocessing unit to make them smooth and sent for the normality test. If data passes the normality test, then Autoregressive Neural

Network (AR-NN) was used for the prediction and prediction results were stored in a buffer. AR-NN consists of three layers, the input layer takes eighteen-lagged inputs, the hidden layer to perform processing of data and forwarding to the output layer of the single neuron. The workload traces were collected from randomly selected VMs from fastStorage data[75].

Abdelsamea et al., proposed a method that uses multiple regression where CPU, RAM, and network BW were used for the detection of PM overloading and significantly reduced EC. They used multiple factors to perform VM management while maintaining the consumption of energy and SLA [76].

[27] Khoshkholghi et al., proposed a dynamic and adaptive energy-efficient management of VMs using iterative weighted linear regression (IWLR) to detect overloaded and underutilized PMs. They reduced the consumption of energy in the cloud datacenter and guarantee the system performance regarding CPU, RAM, and BW. IWLR was applied for predicting the PM utilization which determines the upper and prethreshold. If the predicted utilization of the resource is greater than or equal to one, then PM will not accept new VM and moved to under pressure list, PM will be moved to the overloaded list if the upper threshold is greater than or equal to one[77] .

Shaw et al., presented an RL method Advanced Reinforcement Learning Consolidation Agent (ARLCA) for optimum VM allocation which was capable to optimize the VM distribution in the data center with significant improvement in energy saving and reduction in SLA violation. They developed a state-action space, in which state was defined as the total active PMs as a percentage of the total PMs. The action was a combination of the utilization rate of any PM and the size of the VM to be placed. ARLCA took VM placement list as an input and performed the PM–VM mapping mechanism in the datacenter. The agent determined the global state then each PMs

CPU utilization rate and VM size is computed to generate possible actions to map all VMs on to PMs in the datacenter[78].

SVM was used to develop dynamic and adaptive VM scheduling algorithms based on resource utilization traces of VMs and PMs. The resource analyzer prepared the data set in a regular interval using PMs and VMs utilization history.

This dataset was used to predict future resource utilization for VMs scheduling and placement methods. VMs scheduling optimization was done by defining the weights of the PMs according to the VMs resource utilization using SVM. Historical data of PMs and VMs were evaluated and classified according to the overall resource utilization[79].

[Mason et al.[80], developed a mechanism for predicting the CPU usage of PM using evolutionary Neural Networks and Particle Swarm Optimization (PSO), Differential Evolution (DE), and Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) optimization techniques were used for network training. The experiments results demonstrated that CMA-ES performs best in comparison to other optimization techniques used and trained networks accurately to predict CPU utilization.

Ajay Rawat [81] in his paper, used the ARIMA model to predict the failure of VMs. Using this model, prediction of VM failure is done on non-stationary failure trace. ARIMA i.e. auto-regressive moving average is applicable to both stationary as well as non-stationary data.

Workload prediction based energy efficient virtual machine consolidation in cloud computing. The proposed algorithm considers future load prediction before any allocation of VMs on the PMs and for allocation, modified Best Fit algorithm[82] is applied to reduce energy consumption. This prediction must be faster to make quicker allocations/migrations so that it

does not put an overhead on the energy efficient scheme, thus improving the overall QoS and performance.

# Chapter Four

# Proposed Methodology

This chapter presents the methodology used in conducting this research. It covers the architectural components of the proposed methodology and the datasets selected for this research.

As mentioned in the previous section, the problem of unfair load distribution in cloud system can be resolved either using static or dynamic algorithms[83]–[85]. In this research we propose a dynamic load balancing algorithm through ANN optimized by $ACO_R$. The algorithm predict the future workload of the cloud computing nodes based on historical data and assign the incoming user requests according to the prediction result.

The primary reason for selecting artificial neural network for this problem is because of their ability in modeling complex problems and decision making[86]–[88]. The cloud environment is complex and obscure to predict. The nature of user request is dynamic and unpredictable. ANN's are intelligent enough to deal with obscure systems where the input is not clearly known or measurable[89].

This research utilizes artificial neural network, i.e. multilayer feed-forward network optimized using $ACO_R$ algorithm. After it predict the workload of the virtual machine it can assign the incoming tasks based on the prediction result.

The architectural diagram of proposed model is shown in Figure 4.1. The basic structure of our novel model encompasses three different modules, these are ***workload prediction module***, ***load balancing module***, and ***resource monitoring module***. The resource monitoring module is composed of two components: resource analyzer and resource information collector. The accuracy of the model is evaluated using statistical performance evaluation measure, ***Mean Squared Error (MSE)***.

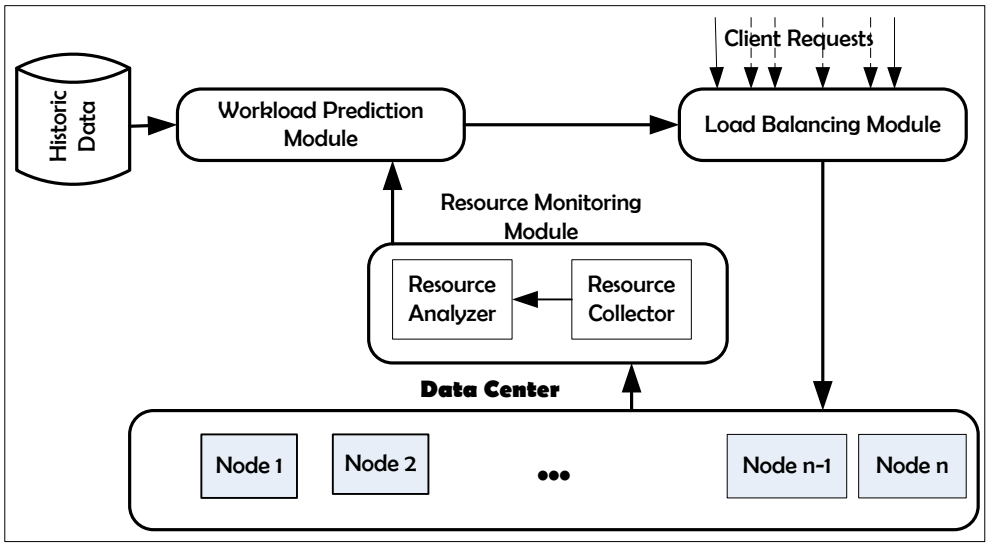


Figure 4.1: Architectural diagram of proposed methodology

## 4.1 Workload Prediction Module

The core module of our system is load prediction module. This module plays very important role in our model. It predicts the amount of workload available on each cloud computing nodes. In order to make accurate prediction of load of each node, the datasets and the network structure of the module should be constructed. The historical dataset is the number of requests stored over time and used as input for the prediction. Figure 4.2(a) shows the work flow diagram of workload prediction module. It starts by accepting historical dataset of computing nodes which followed by preprocessing of datasets. The preprocessing includes aggregation and normalization of dataset that required for training and testing purpose. The next step is training the module to predict the workload of computing nodes. At this stage we perform training optimization to ANN with $ACO_R$ algorithm. Figure 4.2(b) and Figure 4.3 shows the detail process of training optimization using $ACO_R$. After training of model, the accuracy of the model is evaluated by conducting an empirical study. Generally, the above processes can be summarized in following four major steps.

(i) Gathering historical data,

(ii) Building and configuring relevant network,

(iii)Initializing weights and biases, and

(iv)Training and validation step.

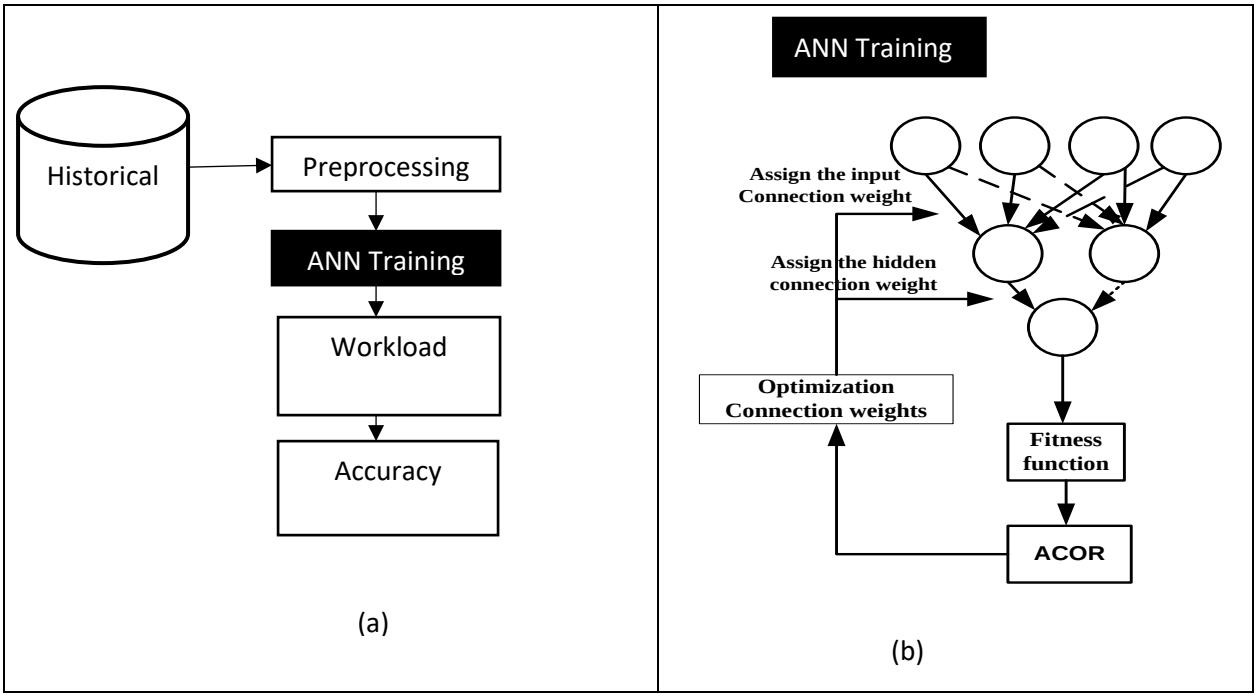


Figure 4.2: Workload prediction Workflow

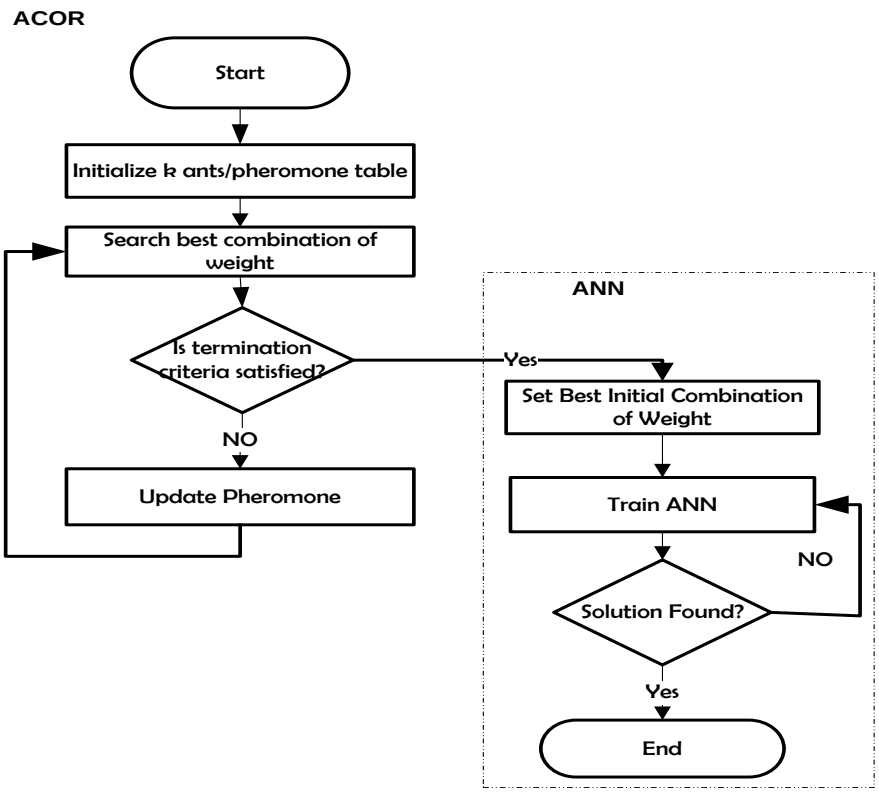


Figure 4.3: Flow chart of ANN training with ACOR

## 4.2 Resource Monitoring Module

The purpose of resource monitoring module is to explore how the nodes are functioning and gather the node information periodically. The gathered information is saved as log. The log contain information about usage of resources such as node's CPU utilization, storage, and bandwidth usage. The time period to grasp load information must be set reasonably to avoid unnecessary overhead and enormous load information. If the node load information gathered in a time series way, there will be too much data to process. If the node load information collection frequency is too high, it may not be able to reflect the node load condition. It is suggested that 5-10 second time interval is a reasonable time to collect node load information[90]. Then this information is sent to

resource analyzer for further processing to filter needed information. The data can be collected by using centralized or decentralized architecture.

## 4.3 Resource Analyzer

Resource monitoring module collects different types of information like processing time, CPU utilization, memory utilization, energy consumption, energy utilization, bandwidth, delay, latency etc. from different nodes. This information may contain redundant, invalid, conflict and irrelevant data. Before this information is used by the prediction module, it is essential to filter the data in order to match the arrangement and dynamics of prediction module. Analysis involves filtering some data out, removing unwanted data and aggregating similar types of data together in order to extract the desired information. Finally, this information will be sent to the prediction module and used as an input to neural network for prediction.

## 4.4 Load Balancing Module

The load balancing architecture shown in Figure 4.4 consists of four main components. These are the user, the job selector, the load balancer and the algorithm to be used. Algorithm I shows how the load balancing module searches and selects a suitable VM from all available hosts to assign user requests. The following steps are carried out for the execution of a request by the user.

i. Every request from a user arrives at Data Centre Controller.
ii. Data Centre Controller queues all the incoming requests and queries the Central Load Balancer for allocation of requests.
iii. Central Load Balancer consist of a database selects the most suitable VM and returns the id of the selected VM to the Data Center Controller.
iv. Finally, Data Center Controller assigns the request to VM whose id is provided by Central Load Balancer.

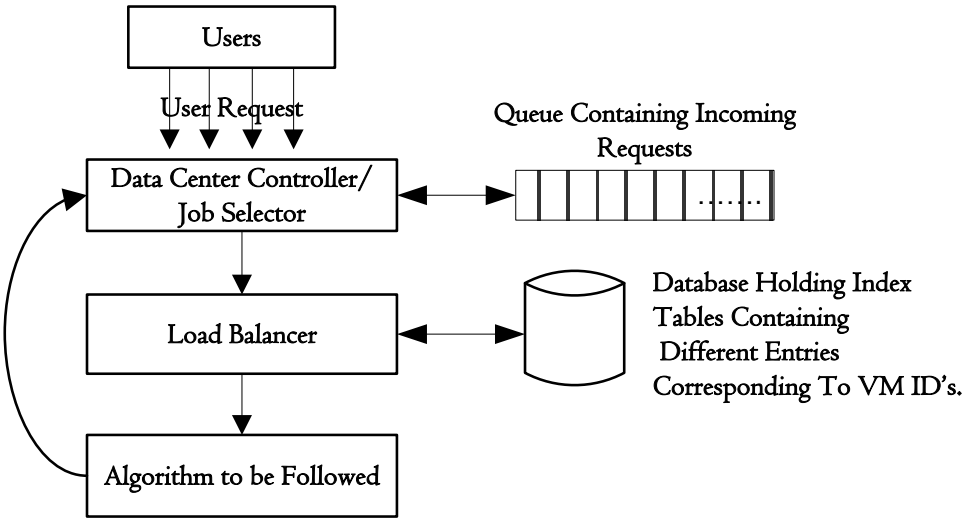


Figure 4.4: Components of load balancing module

**Algorithm I : Load Balancing Module Process**

Input : Host list with *n* hosts, VM list with *m VMs*.

Output: VM with smalles workload

1. **Begin**
2. **for each** n in host list **do**
3. **for each** m in VM list **do**
4. Apply prediction model to predict future workloads for virtual machine m
5. Calculate the average of the predicted workloads
6. **if** (predicted utilization > threshold) **then**
7. vmUnderloadedlist.add
8. **end if**
9. **end for**
10. **end for**
11. Sort vmUnderloadedList in the order of decreasing utilization
12. Apply VM selection policy
13. Select the VM with the smallest workload and
14. **Return** selected Vm to return the id of the selected VM to the Data Center Controller
15. **End procedure**

## 4.5 $ACO_R$ for Training Artificial Neural Network

As mentioned earlier, in this research we use continuous ant colony optimization algorithm as a training mechanism. The difference between discrete ACO and $ACO_R$: ACO uses a finite set of variables chosen with a discrete probability distribution, while $ACO_R$ uses a continuous probability distribution, specifically a Gaussian probability density function[17]. Finding the optimal set of neural network weights is continuous, as the solution space is continuous. So, in this thesis the ACO algorithm for continuous optimization as defined by Krzysztof Socha and Christian Blum[17] is used. The schematic diagram of the algorithm is presented in the Figure 4.4. The algorithm solves the optimization problem by iteratively implementing the following two steps:

1. Construct candidate solutions probabilistically using a probability distribution over the search space.
2. Use the candidate solutions to modify the probability distribution in such a way that future sampling is biased towards high quality solutions.

In general, $ACO_R$ follows the following step by step procedure to produce optimal solution. First, the solution archive is initialized. Then, at each iteration a solution is constructed probabilistically by the ants. These solutions may be improved by any improvement mechanism. Finally, the solution archive is updated.

### 4.5.1 $ACO_R$ Framework Components

$ACO_R$ consist of an initialization step and three algorithmic components. Figure 4.5 shows the general outline of $ACO_R$. The detail outline of the components are described below. Simply put, training neural network is the process of finding the optimum set of weight values for the links between neurons allocated in input, hidden and output layer of a neural network.

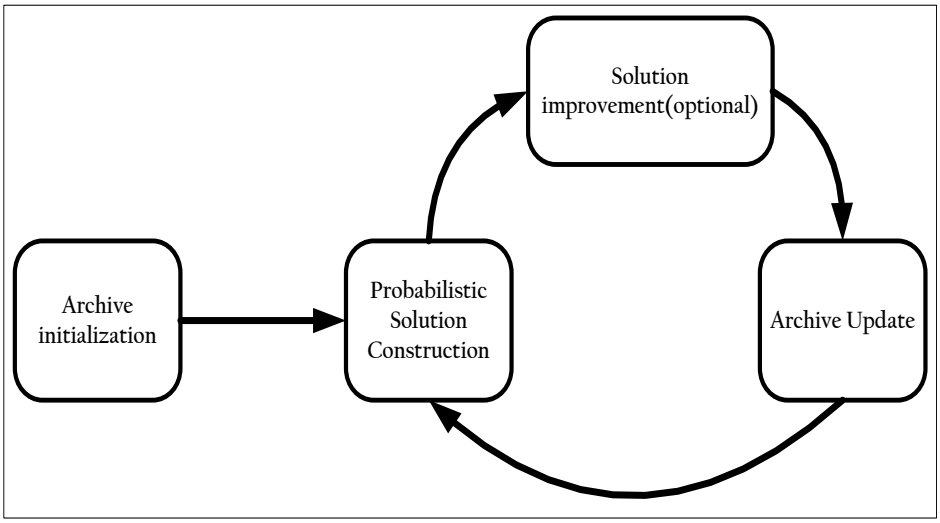


Figure 4.5: The process of ACOR algorithm

### Archive structure, initialization, and update

The **archive** is a table T generated by the algorithm to store all candidate solutions. Initially, the algorithm generate a random number of possible solution called candidate solutions. In our case this candidate solutions represent a combination of connection **weight** values needed for the neural network. The candidate solutions are possible solutions, i.e., they are not the final optimal solution. The structure of the solution archive is shown in the Figure 4.6.

The solution archive or table maintains all the candidate solutions, $S_i$; each candidate solution contains the values for all the 'n' variables that define the search space. In our case 'n' is equal to the number of weights in the neural network that is trained. The archive contains 'k' solutions ordered according to their quality. At each algorithm iteration, a set of m solutions is

probabilistically generated and added to T. The same number of the worst solutions are removed from T. This biases the search process towards the best solutions found during the search.

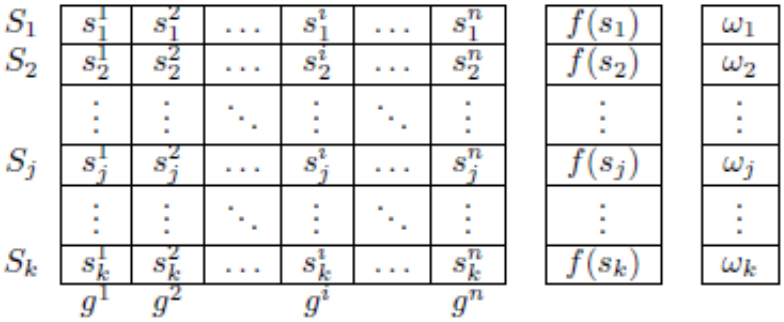

Figure 4.6: Structure of Archive, Fitness vector and Weight Vector

### i. Fitness vector

The $\boldsymbol{n}$ dimensional optimization problem contains $S_i$ *solution* and $f(S_i)$ *objective function*. The fitness vector contains the result of the objective function for all the solution in the archive.

This is the function that we are trying to minimize. In this case we are trying to minimize the sum squared error for our training set. The archive is sorted against the values of this vector. Like GA and PSO approaches, $ACO_R$ does not use gradient information at all, and the only problem-dependent information that it has access to is the fitness function (error function).

### ii. Wight Vector $(\omega)$

Each solution added to the solution archive has a weight value. The weight vector contains the weight value associated to each solution. These values are evaluated and ranked based on their quality. The weight vector is calculated according to the following formula.

$$\omega l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}}$$

Where $k =$ number of solutions in the archive, $l =$ index of the solution, $i =$ dimension and $q =$ a parameter of the algorithm. When $q$ is small, the best ranked solutions are strongly preferred, and when it is large, the weight distribution is almost equal.

### iii. Probability Solution Construction

The probabilistic construction of solution is performed by sampling Gaussian probabilistic distribution functions (PDFs) over the search space. An ant performs $\boldsymbol{n}$ construction steps and at each construction step the ant chooses a value for decision variable. This is done by sampling a Gaussian kernel PDF, which is derived from the solution stored in archive T. Gaussian Kernel PDF is a weighted sum of several one dimensional Gaussian functions. The Gaussian kernel PDF is computed using the following equation.

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i^2}}}$$

Where $k$ = number of solutions in the archive, $l$ = index of the solution and $i$ = dimension. The kernel PDF helps us to create an $n$ dimensional probability distribution space as show in Figure 4.7. From this we can sample to generate a set of biased weights for the neural network under training.

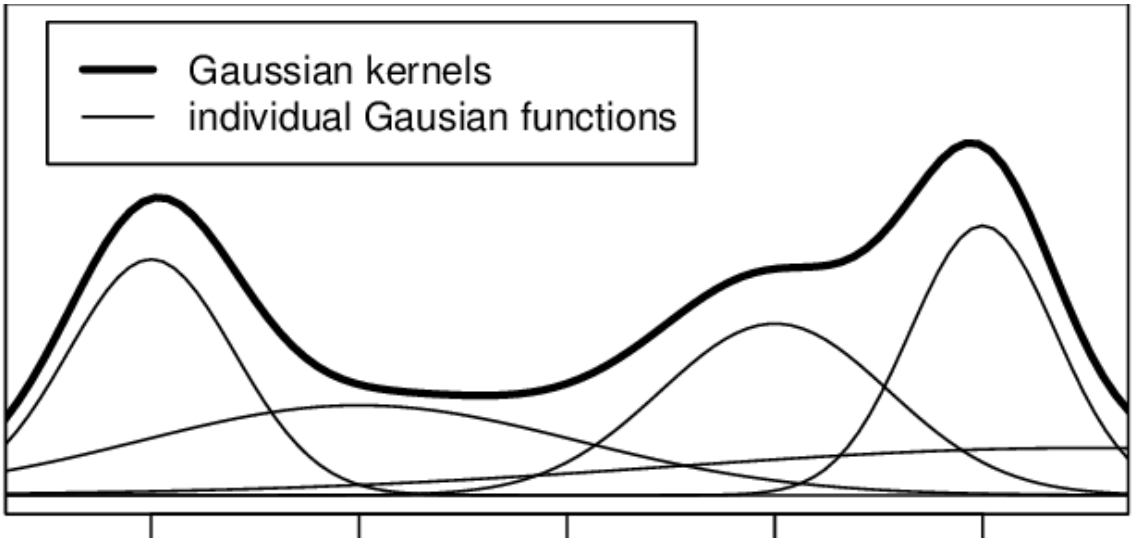


Figure 4.7: Gaussian Kernel PDF

# Chapter Five

# Experiment and Result

This chapter presents the experiment performed with the proposed methodology and the findings. In the first part of this section, the experiments environment specification and the experimental parameters will be discussed. Then experimental results are presented and discussed. In the result section, the performance of the proposed algorithm is compared against related methodologies with different parameters.

## 5.1 Development Environment and Tools

We used 64-bit Microsoft Windows 10 operating system to perform all our experiments. The machine is installed with 2.4GHz Intel(R) core™i5 processor and 8GB random access memory (RAM). To implement the algorithms java programming language is used and the Apache Commons Math3 library has been used for random number generation and other mathematical operation [91]. NetBeans 8.2 software is used as a development environment tool.

## 5.2 Neural Network Structure

The benchmark network is multiple input multiple output (MIMO) feed-forward neural network. The model architectures in abbreviated notation are shown in Figures 5.1. Using this model the algorithm tested with different parameters. All of the network models share common features on normalization, number of hidden layer neurons, performance function, training algorithm, generalization and training stopping criteria.

The number of nodes in the hidden layer can influence the overall ANN performance considerably. Too few hidden layer nodes can lead to under fitting, and using too many nodes can lead to over-fitting problems. There is no specified rule and optimal number in theory to follow while

specifying the number of hidden layer neurons. The rule of thumb says the hidden layer neuron should not be greater than twice the number of inputs[92].

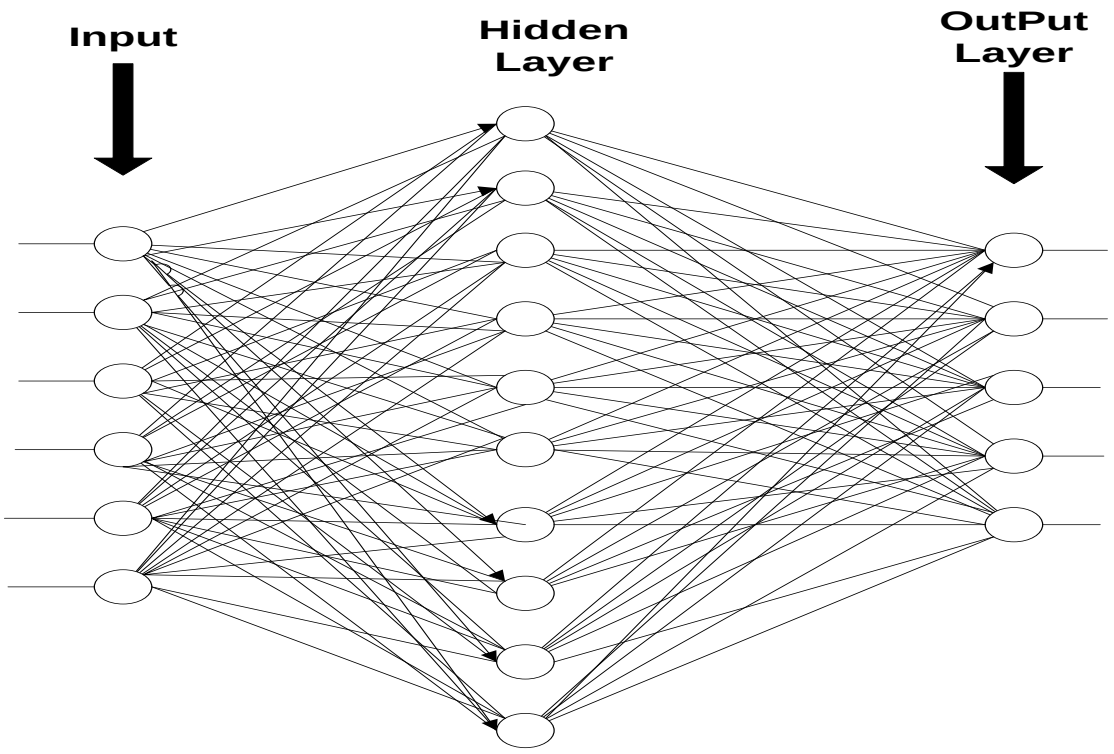


Figure 5.1: Neural Network Model

**Training process and stoppage criteria**

The computational process of neural network begins by receiving multiple inputs from the outside world or from other neurons. Each of these inputs are multiplied by a connection weight. These products are summed, fed to a transfer function (activation function) to generate the final result, and this result is sent as output.

Figure 5.1 shows the proposed multilayer feedforward network structure used in this research. It consist of one input layer with 6 inputs, one hidden layer with a number of nodes and an output layer. Training begins by feeding the dataset to the input layer. Each of these inputs are multiplied

by connection weight, summed and fed to activation function. The outputs are calculated at the output layer and compared with the desired loads, errors are computed and weights are adjusted. This training process continues until the network achieves the acceptable error rate or the iteration limit is reached. Once the network is trained within a tolerable error, the network is tested with different dataset. The parameters to train the network are shown in Table 5.1 and Table 5.2.

Table 5:1: Summary of parameters used to train the network

| Parameters | Values |
|---|---|
| Maximum number of epochs | 1000 |
| Performance goal(mse) | 0.05 |
| Maximum time to train in seconds | ∞ |
| Maximum performance gradient | 1e-6 |

Table 5:2: Experimental parameter values

| Parameter | Value |
|---|---|
| Input nodes(n) | 6 |
| Hidden nodes(p) | 10 |
| Maximum iteration | 1000 |
| Training data size | 60% |
| Test data size | 40% |
| Learning period | 0.01 |

## 5.3 Dataset Description

To evaluate our methodology we used a **faststorage** and **rnd** datasets from GWA-T-12 Bitbrains workload trace datasets [18]. Bitbrains is a service provider that specializes in managed hosting and business computation for enterprises. The dataset includes a collection of data from many major banks, credit card operators and insurers. Many computers, data centers and storage systems

are linked together to analyze data produced by these companies. FastStorage, consists of the individual traces of 1,250 VMs that are used in the software applications hosted within the Bitbrains datacenter. The **rnd** datasets consist of individual traces of 500 VMs.

Both traces were stored in .csv (comma separated values) files and each of these .csv files contained approximately 43 minutes' worth entries, each row separated by 300 ms and each .csv files consist of roughly 8600 rows of data. The datasets have eleven attributes as shown in the Table 5.3. For our purpose we used only six attributes contained in these preprocessed .csv files and classified into classes that represented the upper and lower bounds of the average CPU utilization. The dataset is divided into testing and training datasets. We used 60% of them as training set and the remaining 40% sets as test set. Table 5.3 shows the summary of the features included in the dataset.

Table 5:3: Schema of the GWA-T-12 Bitbrains dataset

| Schema | | |
|---|---|---|
| Index | Name | Description |
| 0 | Timestamp | The number of milliseconds since the start of the trace |
| 1 | CPU cores | Number of virtual CPU cores provisioned |
| 2 | CPU capacity provisioned (MHZ) | The capacity of the CPUs in terms of MHZ |
| 3 | CPU usage (MHZ) | Utilization of CPU in terms of MHZ |
| 4 | CPU usage (%) | Utilization of CPU in terms of percentage |
| 5 | Memory capacity provisioned (KB) | The capacity of the memory of the VM in terms of KB |
| 6 | Memory usage (KB) | The memory that is actively used in terms of KB |
| 7 | Disk read throughput (KB/s) | Disk read throughput in terms of KB/s |
| 8 | Disk write throughput (KB/s) | Disk write throughput in terms of KB/s |
| 9 | Network in (KB/s) | Network received throughput in terms of KB/s |
| 10 | Network out (KB/s) | Network transmitted in terms of KB/s |

### 5.3.1 Data Preprocessing and Feature Selection

The raw data from both datasets are time-series data. In order for the datasets to match our purpose, we performed a series of filters onto the raw dataset. Initially we changed the continuous raw data to discrete data by performing an averaging filter onto the raw data and cleaned unnecessary values such as zeros and null values. This process is done by taking the 15 second (50 rows) average resource utilization on all of the VMs. Then, the average resource utilization for CPU, memory, disk read, and network out contained in these preprocessed .csv files are classified into classes. The class represent the upper and lower bounds of the average CPU utilization. This classification was performed with uniform classification. We use CPU utilization as an output or target values. Tables 5.4 below shows the mapping between classes to average CPU utilization percent range for uniform classification method. It classifies the outputted average CPU utilization into 5 separate classes, each representing a uniformly sized range of CPU utilization. Class 0 starts from 0 - 20% and each corresponding class is incremented in ranges of 20%.

Table 5:4: Average CPU utilization ranges for the uniform classification method

| **Uniform Classification** | |
|---|---|
| **Class#** | Percentage Range |
| **0** | 0-20% |
| **1** | 20-40% |
| **2** | 40-60% |
| **3** | 60-80% |
| **4** | 80-100% |

## Random Forest Feature Selection

Random forest (RF) algorithm for variable selection was adapted to this problem and implemented. We do not use all the features to train a model. We may improve our model with the features correlated and non-redundant, so feature selection plays an important role. We use the ranked features list as produced by random forests for searching a minimal feature set to train a proposed model.

RFs are among the most popular machine learning methods with relatively good accuracy, a good predictive performance, robustness and ease of use. RFs have been widely used as a powerful classification method with the randomization in feature selection. RFs have shown excellent performance for both classification and regression problems. RF model works well even when predictive features contain irrelevant features or noise. They also provide two straightforward methods for feature selection: mean decrease impurity and mean decrease accuracy.

The mean decrease in accuracy (MDA) and the mean decrease in Gini (MDG) are commonly used statistical measures of variable importance for determining which predictor variables are best suited to differentiate the classes of interest and for reducing the dimensionality of large data sets. Figure 5.2 shows list of features and their score according to their importance selected by RF.

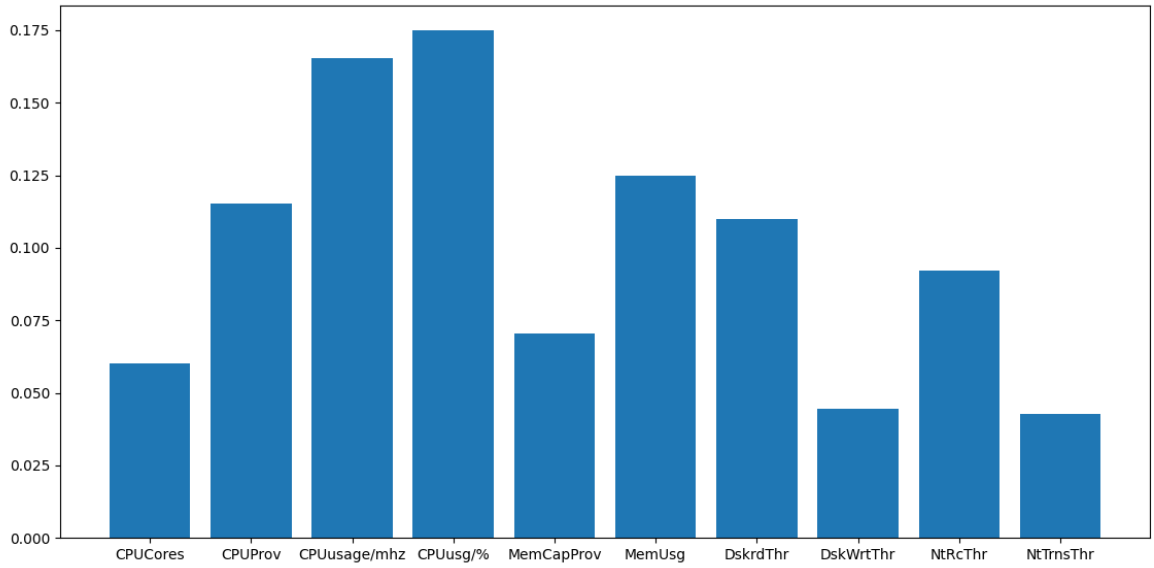


Figure 5.2: List of features and their scores

Based on z-scores, for our purpose we used the top six important features, i.e. (CPU provisioned, CPU usage in megahertz, CPU usage in percentage, memory usage, disk read, and network transmitted throughout).

### 5.3.2 Normalization

Before the data used by the network, it is important for it to be normalized. Commonly, the data is normalized in the range of [0, 1]. This process eliminates the problem of overshadowing, so the most important variables with small magnitudes will not be overshadowed by less important variables having larger magnitudes. The following equation is used to normalize data.

$$y = \frac{(x - xmin)}{xmax - xmin}$$

Where $x$ is the original data and $y$ is normalized data

## 5.4 Experimental Metrics and Evaluation

To demonstrate the efficiency of the proposed methodology, the algorithm is tested and evaluated using the Mean squared error (*MSE*) with varying the important parameters of the model. MSE is the most commonly used evaluation method in machine learning for regression problems. The mean squared error is essentially the mean of the squares of the individual difference between network output and known outcome in each training dataset. The result of MSE is always positive. The following equation is used to calculate the mean squared error.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (yi - \hat{y}i)^2$$

Where:

$y$ : Target output

$\hat{y}$ : Actual output

$n$: Training data sample size

## 5.5 Experiments

To answer our research questions stated in Chapter One, different experiments are conducted. The performance of the algorithm has a great correlation with a selection of precise network parameters. The model is tested with different parameters. Some of the system parameters and their values are fixed based on previous investigations while the other parameter values are set based on experimental analysis[93]. The summary of parameters selected for both neural network and $ACO_R$ are presented in Table 5.5 and Table 5.6. In order to insure a fair comparison between algorithms, an equal amount of effort is required in the parameter tuning process for each of the algorithms. The number of nodes in the hidden layer, learning rate and training epoch are the same for all experiments. Additionally, **sigmoid** and **tanh** function are used as activation function.

Table 5:5: Experimental parameter values used by neural network

| **Parameter** | **Value** |
|---|---|
| Input nodes(n) | 6 |
| Hidden nodes(p) | 10 |
| Maximum iteration | 1000 |
| Training data size | 60% |
| Test data size | 40% |
| Learning rate | 0.01 |
| Performance goal(mse) | 0.005 |

Table 5:6: Experimental parameter values used by ACOR

| **Parameter** | **Symbol** | **Value** |
|---|---|---|
| No. of ants used in an iteration | $m$ | 6 |
| Speed of convergence | $\varepsilon$ | 0.85 |
| Locality of the search process | $q$ | $10^{-4}$ |
| Archive size(k) | $k$ | 100 |

The proposed methodology was tested on both ***FastStorage*** and ***Rnd*** dataset. Four experiments are done on each dataset with respect to prediction window size. We study the effect of prediction performance of the proposed model using different fixed sliding window sizes. Three different formats of input data for observation window sizes 10, 15, and 20 with five second increments and the accuracy of the prediction for the time frame was evaluated. The dataset used in experiments are selected based random forest feature selection. We perform this test to observe the effect of varying training set length on the forecasting performance. We also tested the model with input data contains only ***CPU*** utilization of twenty second window size time frame. So at each time step only CPU utilization is provided to the model and the prediction accuracy is evaluated.

**Experiment I: FastStorage-Rnd – 10 second window size data frame dataset**

The first experiment is performed with the observation of ten second time frame dataset selected with random forest feature selection. This experiment is conducted by feeding both ***FastStorage*** *and* ***Rnd dataset*** for the proposed algorithm in different runs. The aim of the experiment is to evaluate the performance of the model by providing ten second time frame resource utilization of virtual machines data to the algorithm. Table 5.7 and 5.8 shows the training and testing accuracy obtained by the algorithm employing both datasets.

Applying *FastStorage - dataset*, the best performance achieved by the model is **84.4%** on training dataset and **83.3%** on testing dataset. Again, the model was able achieve **86.1%** training accuracy and **85.4** % testing accuracy when Rnd *dataset* is employed. Although, the difference in performance degradation is small on both training and testing accuracy, compared to FastStorage-CPU dataset the model performed less (**2.3%** training accuracy and **2.1%** testing accuracy difference) on *Rnd- CPU dataset.* This may indicate that *FastStorage - CPU* dataset does represent well all possible patterns than *Rnd- CPU dataset*.

The classical backpropagation neural network training algorithm is used as a reference algorithm to compare the performance of $ACO_R$. Backpropagation algorithm is the most popular, widely used training mechanism and workhorse of learning in artificial neural network [93]. The results are illustrated in Table 5.7 and 5.8. $ACO_R$ achieved reasonably good result, however, it is outperformed by backpropagation (ANN_BP) training on both datasets and training time.

Table 5:7: Accuracy of ten seconds window size Faststorage dataset**.**

| | **Dataset** | **Train Accuracy** | **Test Accuracy** |
|---|---|---|---|
| **ANN_ACO$_R$** | FastStorage | 84.4 | 83.3 |
| **ANN-BP** | | 92 | 93.1 |

Table 5:8: Accuracy of ten seconds window size Rnd dataset

| | Dataset | Train Accuracy | Test Accuracy |
|---|---|---|---|
| **ANN_ACO$_R$** | Rnd | 86.1 | 85.4 |
| **ANN-BP** | | 95 | 94.4 |

The performance of the proposed method is compared against some related ANN training methods. The result achieved on both versions of the dataset is shown in Figures 5.3 and 5.4. Yet again the proposed method is outperformed by Particle swarm (ANN-PSO), Genetic algorithm (ANN-GA), and hybrids of ACO$_R$-BP neural network training methods. The hybrid of ACO$_R$-BP based neural network obtained promising prediction result and outperforms back propagation based neural network in training accuracy. A hybrid of ACO$_R$-BP also outperforms ANN-GA and ANN-ACO$_R$ models in both training and testing prediction accuracy.

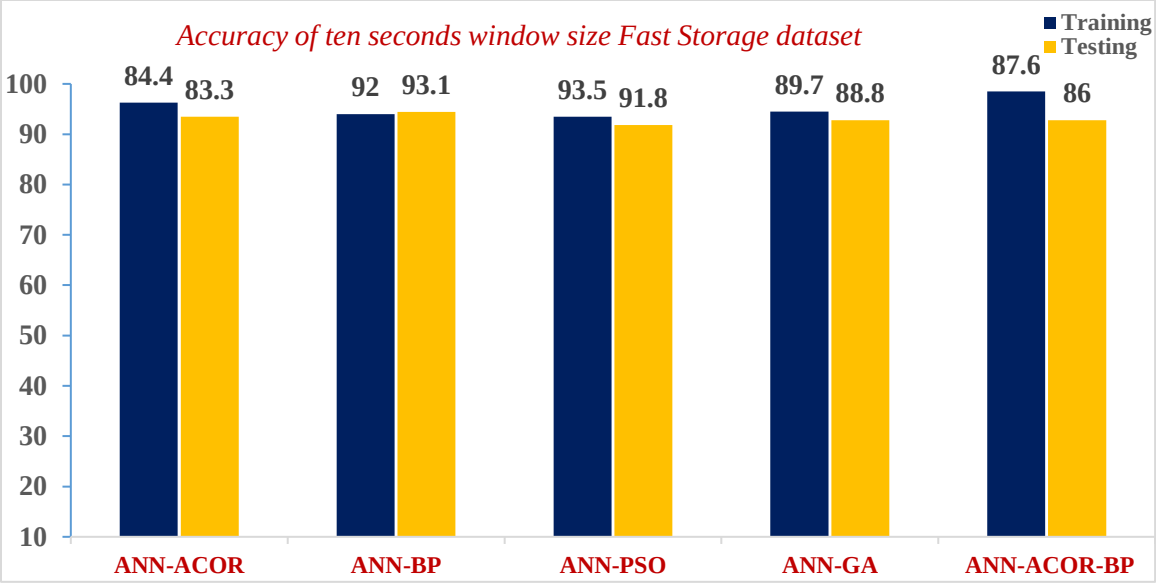


Figure 5.3: Accuracy of different algorithms on FastStorage ten seconds window size dataset

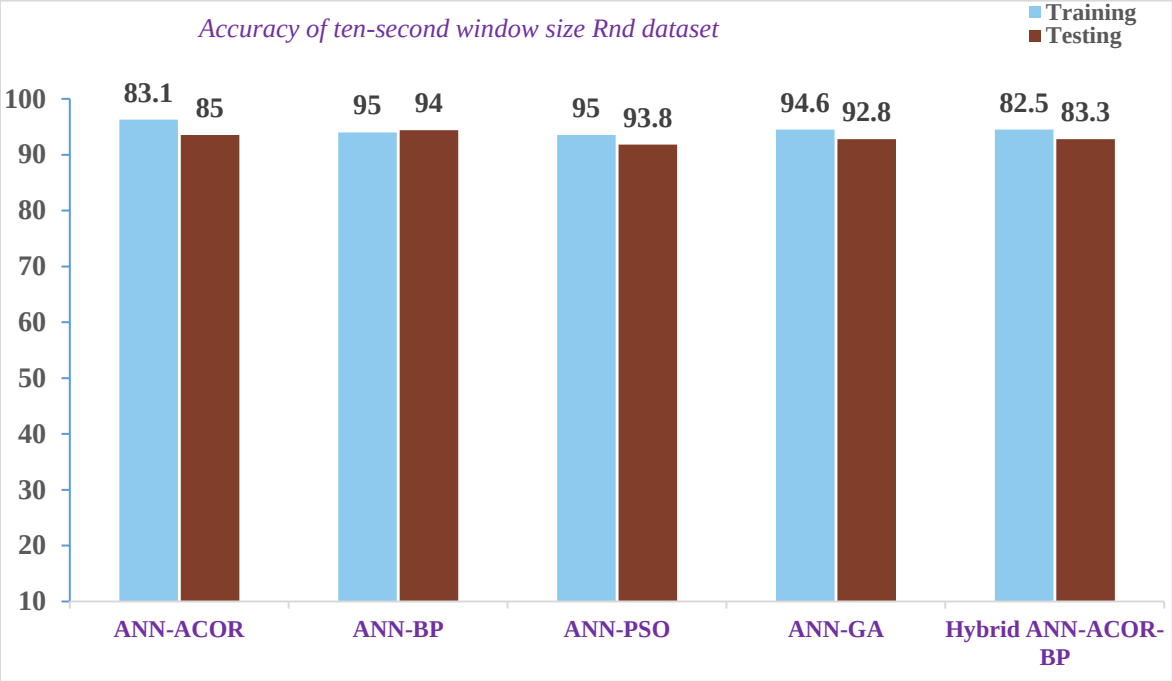


Figure 5.4: Accuracy of different algorithms on Rnd ten-second window size dataset

**Experiment II: FastStorage-Rnd – fifteen seconds window size time frame dataset**

The aim of this experiment is to see if the sliding window has any positive effect on the forecasting accuracy of the fitted models. In this experiment, at each time step fifteen second time frame resource utilization input data is provided to the model and CPU utilization of the system is predicted. Tables 5.9 and 5.10 below show the train and testing accuracy for both FastStorage and Rnd dataset. As shown in the tables, for both models i.e. ANN-ACOR and ANN-BP, We find that prediction errors tend to decrease when the window size time frame of training data increases. BP achieved better result compared to the previous experiment, also the performance of BP is better than proposed methodology. This indicates that the classical back propagation based neural network is suitable than proposed method for this kind of problems.

Table 5:9: Accuracy of fifteen second window size Faststorage dataset

| | Dataset | Train Accuracy | Test Accuracy |
|---|---|---|---|
| **ANN_ACO$_R$** | Faststorage | 91 | 89.5 |
| **ANN-BP** | | 96 | 94.4 |

Table 5:10: Accuracy of fifteen second window size Rnd dataset

| | Dataset | Train Accuracy | Test Accuracy |
|---|---|---|---|
| **ANN_ACO$_R$** | Rnd | 89 | 89 |
| **ANN-BP** | | 95.33 | 91.6 |

We also achieved a better result with all other training methods except Hybrid ANN-ACO$_R$-BP training methods in the Rnd dataset than ANN-ACOR. Figures 5.7 and 5.8 show the training and testing accuracy achieved by related ANN training methods. These results show that the appropriate observation window size can play an important role to minimize the estimation accuracy for different estimation method.

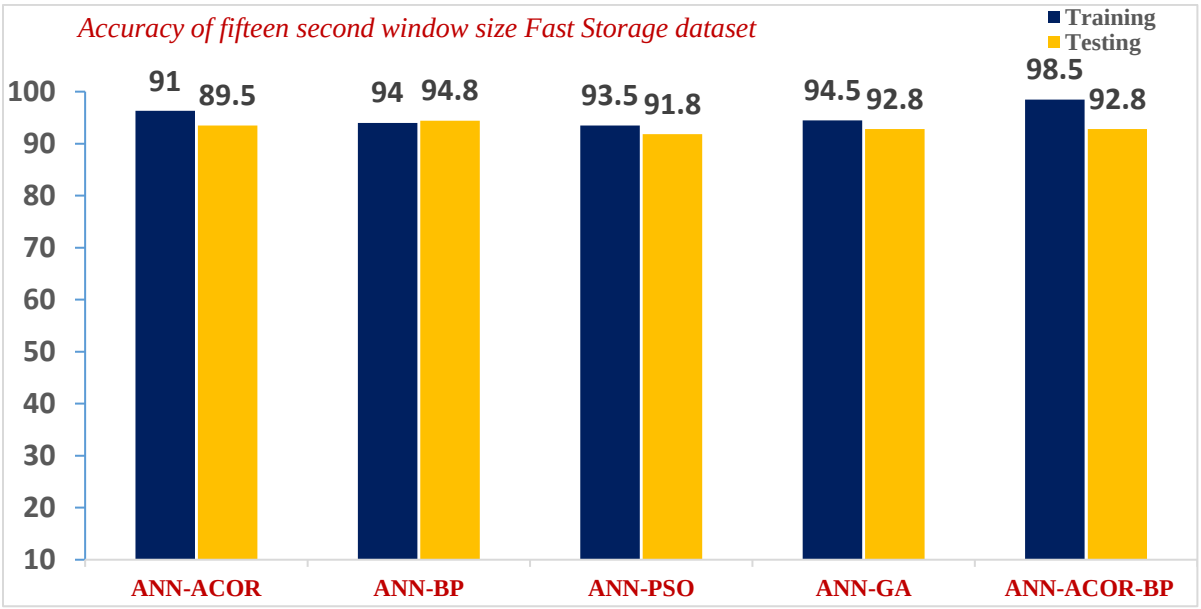


Figure 5.5: Accuracy of different algorithms on fifteen second window size FastStorage dataset

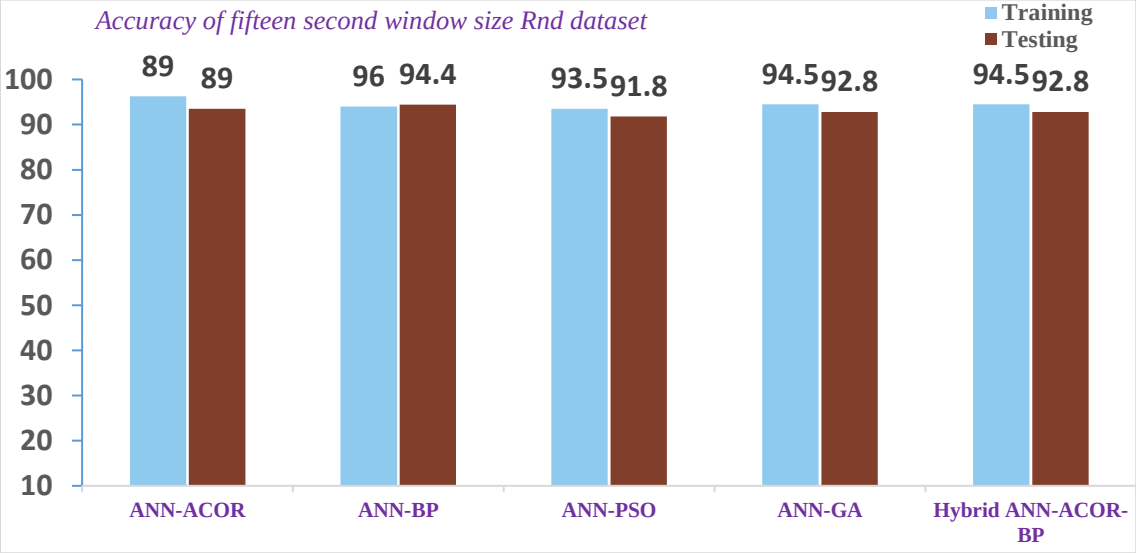


Figure 5.6: Accuracy of different algorithms on fifteen second window size Rnd dataset

**Experiment III: FastStorage-Rnd – twenty second window size time frame dataset**

In this experiment the performance of the proposed model is studied by providing twenty seconds window size time frame dataset to the algorithm. At each time step we fed the model twenty seconds window size time frame dataset and examined the result. Tables 5.11 and 5.12 below show the train and testing accuracy for both FastStorage and Rnd dataset. The best performance achieved by the model is 90 % training accuracy 92.5% test accuracy and 91.5% training and 92.3 % testing on FastStorage and Rnd dataset respectively. The model performed better when we utilize the Rnd dataset, and the result clearly shows that accuracy of prediction model improves as the size of the training window increases. Generally, accuracy of prediction model improves as the size of the training window increases i.e. it is better to have large historical data for training the prediction model so that it covers all possible patterns. In addition, our model does not perform better than the typical BP based neural network on both versions of datasets.

Table 5:11: Accuracy of Faststorage twenty second window size dataset

| | Dataset | Train Accuracy | Test Accuracy |
|---|---|---|---|
| **ANN_ACO$_R$** | Rnd | 91.5 | 92.3 |
| **ANN-BP** | | 96 | 94.4 |

Table 5:12: Accuracy of Rnd twenty second window size dataset

| | Dataset | Train Accuracy | Test Accuracy |
|---|---|---|---|
| **ANN_ACO$_R$** | FastStorage | 90 | 92.5 |
| **ANN-BP** | | 96 | 94.4 |

Obviously, we compared the performance of the model with related nature inspired neural network training methods. The result shows that the model is outperformed by Particle swarm (ANN-PSO), Genetic algorithm (ANN-GA), and Hybrid of $ACO_R$-BP neural network training methods. But the result obtained by a hybrid of $ACO_R$–BP is better than the result achieved by all other algorithms on both versions of datasets. Figures 5.9 and 5.10 show the training and testing accuracy achieved by nature inspired ANN training methods.

These results show that the appropriate observation window size can play an important role to minimize the estimation accuracy for different estimation method. In addition, our model does not perform better than the typical BP based neural network on both versions of datasets.

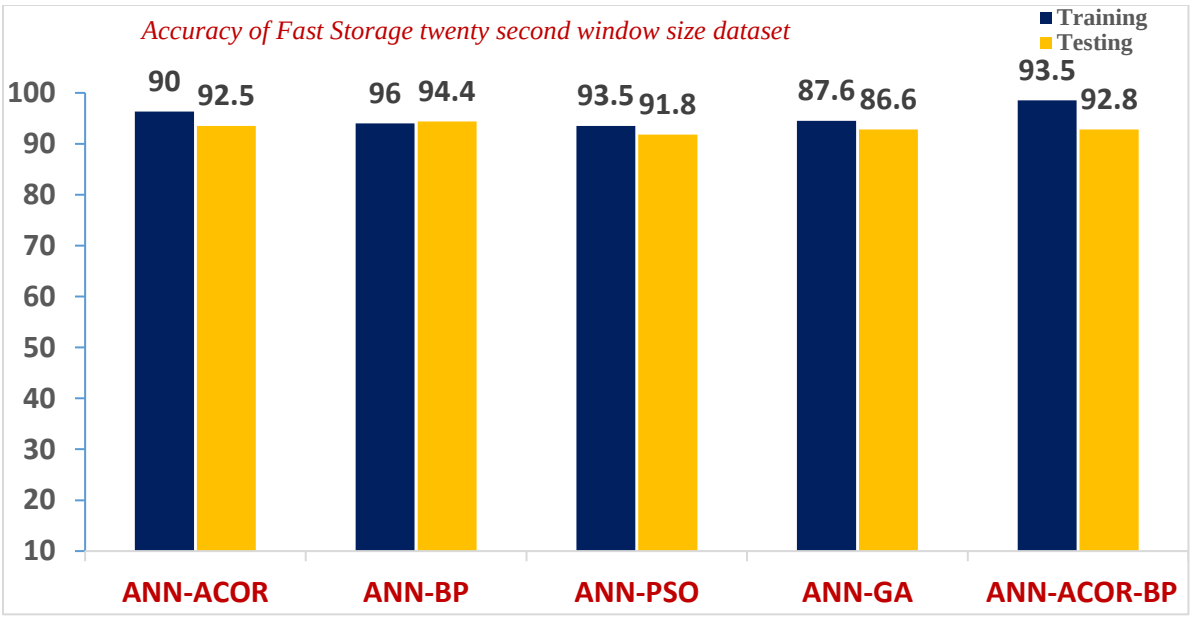


Figure 5.7: Accuracy of different algorithms on Faststorage twenty second window size dataset.

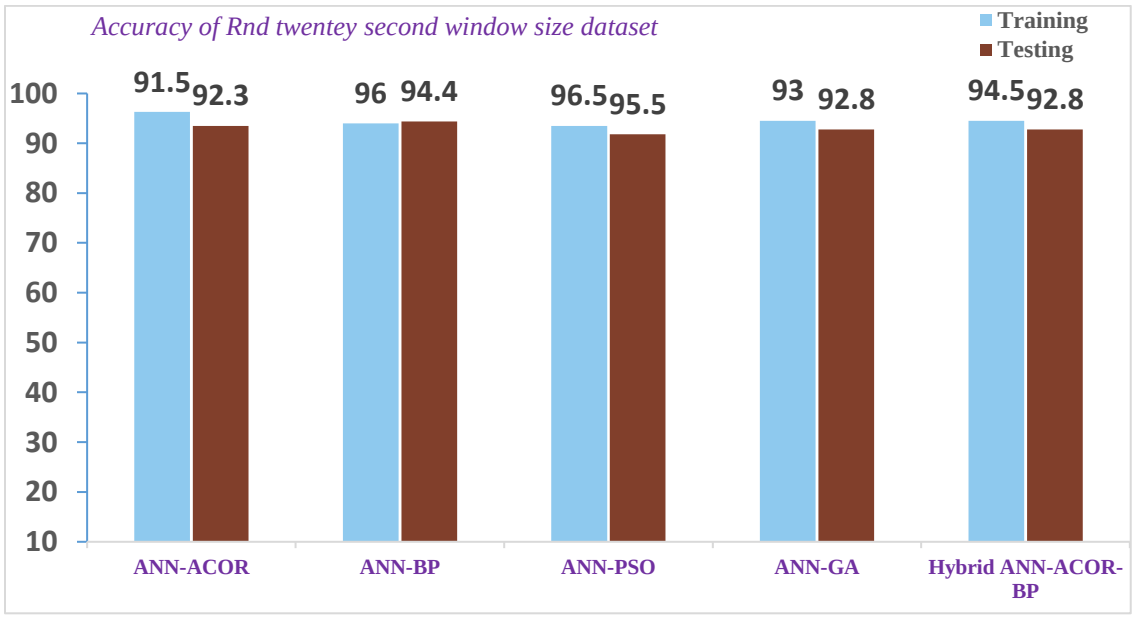


Figure 5.8: Accuracy of different algorithms on Rnd twenty second window size dataset

**Experiment IV: FastStorage-Rnd – twenty second window size time frame CPU dataset**

This experiment is conducted by feeding both ***FastStorage and Rnd - CPU dataset*** for the proposed algorithm in different runs and the performance of the model is evaluated. Table 5.13 shows the training and testing accuracy obtained by the algorithm employing both datasets.

Applying *FastStorage - CPU dataset*, the best performance achieved by the model is 96.3% on training dataset and 93.5% on testing dataset. Again, the model was able achieve 95.2% training accuracy and 92.9% testing accuracy when Rnd-*CPU dataset* is employed. Although, the proposed model outperformed by other training methods, compared to the previous three experiments the model yield better result using this dataset.

| CPU_Dataset | Training Accuracy | Test Accuracy |
|---|---|---|
| Fast_Storage | 96.3% | 93.5% |
| Rnd | 95.2% | 92.9% |

Table 5:13: Training and Testing Accuracy of Faststorage and Rnd CPU dataset

The performance of the proposed method is also compared against some related ANN training methods. The result achieved on both versions of the dataset is shown in Figures 5.11 and 5.12. Yet again the proposed method is outperformed by Particle swarm (ANN-PSO), Genetic algorithm (ANN-GA), and hybrids of $ACO_R$-BP neural network training methods.

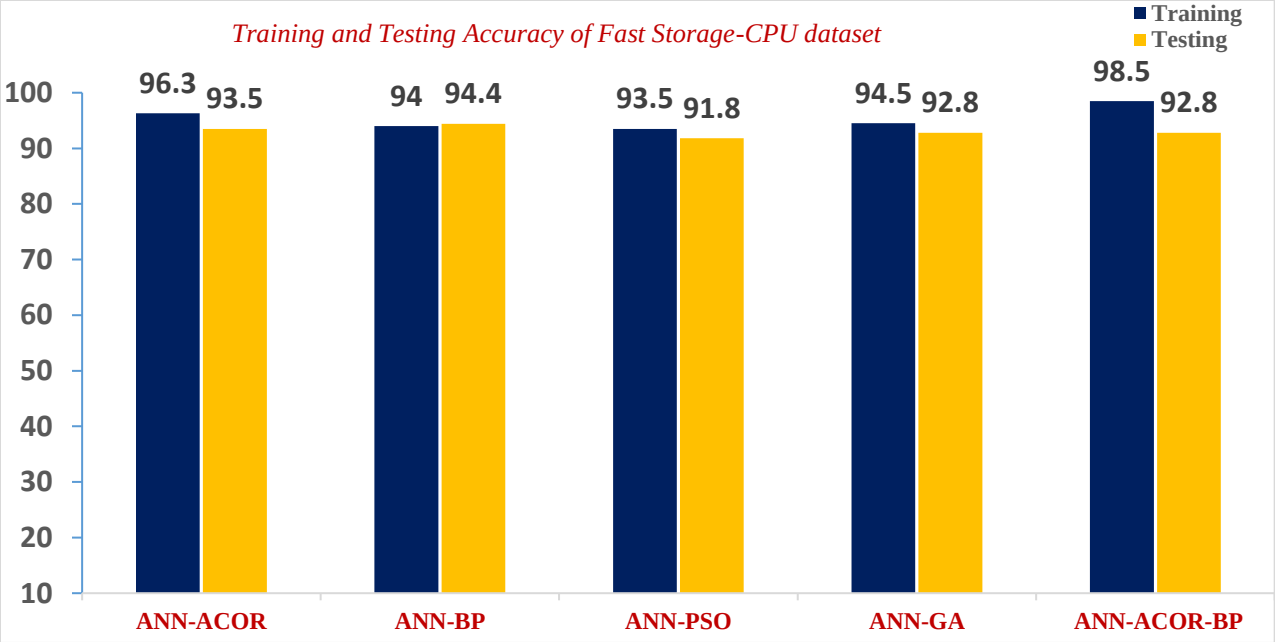


Figure 5.9: Training and Testing Accuracy of different algorithms on Faststorage CPU dataset

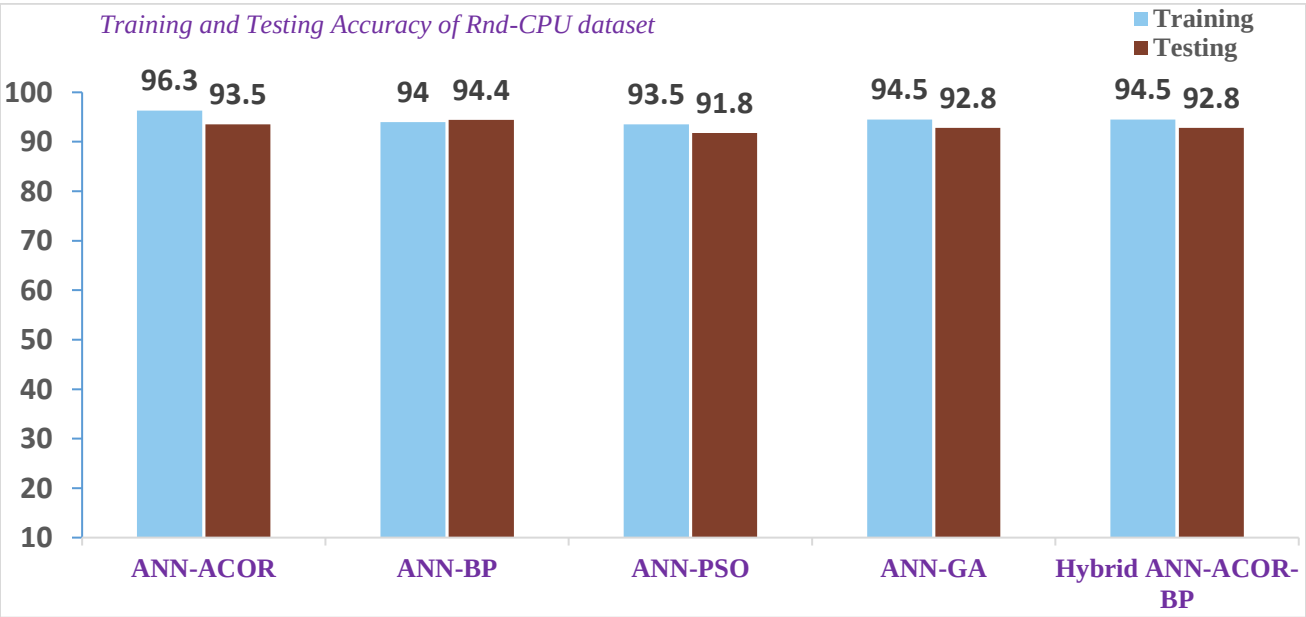


Figure 5.10: Training and Testing Accuracy of different algorithms on Rnd CPU dataset

If the system allocates the user request according to this prediction it might cause waste of CPU resources. Finally, this problem may lead to unfair load distribution, poor quality of service and system failure.

- **Training Time and Accuracy**

The proposed method is not par with the classical back propagation training method in both training time and performance or accuracy. There might be several reasons for that, First, ANN-$ACO_R$ explore a larger area which means larger search space which helps to avoid local optima's, however it will take longer to train.

The other factor for unsatisfactory overall performance of the algorithms may be due to parameter settings. The optimal performance and efficiency of $ACO_R$ often depends on suitable parameters. Appropriate tuning of the underlying parameters can drastically improve the performance of $ACO_R$. $ACO_R$ algorithm with respect to its various parameters (solution construction strategy, distance measure metric, and pheromone evaporation rate) showed that its performance and rate of convergence are sensitive to the chosen parameter values, and especially to the value of the pheromone evaporation rate. The mechanism of construction of new solution may also presented considerable challenge for $ACO_R$ to achieve better performance. $ACO_R$ depends on the construction of new solution, variable by variable basis using Gaussian sampling with a selected variables from an archive solutions. Gaussian distribution lacks non uniformity and wider coverage of selection in terms of selection of a solution for sampling.

To address the first part of our research question (***RQ1: Is ANN optimized by $ACO_R$ prediction effective to solve the cloud workload balancing problem compared to other machine learning techniques?***), based on the result obtained from the three experiments and using MSE as performance criteria, our model, **ANN- $ACO_R$** is outperformed by all algorithms and scored relatively high MSE. So, it is difficult to say that our model is competent enough compared to other models since the result shows low prediction power. But, the hybrid of **$ACO_R$-BP** out

performs all algorithms and achieved better MSE results. As a result, it is better to use a hybrid of **$ACO_R$-BP** as training mechanism to solve the problem of load balancing in cloud computing than our model**.**

To answer the second part of our question (***RQ2: What is the effect of using different window size on the accuracy of the proposed system?***), four experiment were conducted with different window size dataset on the training and testing datasets. We observed that window size 20 yield best prediction accuracy. Window size 10 and 15 gives minimum prediction accuracy compared to window size 20. These results show that the appropriate observation window size can play an important role to maximize the estimation accuracy.

To answer the second part of our question (***RQ2: What is the effect of using different window size on the accuracy of the system?***), three experiment were conducted with different dataset formats on the training and testing datasets. In each experiment we provided CPU, CPU-memory and all, i.e., memory, disk utilization and network traffic, formats of data to the system. Based on the result presented in the above experiment the best performance is achieved when the CPU utilization dataset is provided to the model. As more information is provided to the model the accuracy gets worse. From this we conclude that the more information we provide to the model the more the algorithm get confused and the extra information to the model act as noises for CPU utilization prediction.

# Chapter Six

## Conclusion and Future Work

Load balancing is the process of distributing workload across one or more servers, data centers, or other computing resources, in order to achieve maximum resource utilization, enabling better system response time and improving performance. Workload balancing in cloud computing is a challenging process and requires an efficient method to enable a ration distribution of load among available computing nodes. Inappropriately distributed load might result in inadequate resource utilization and causes certain computing nodes being overwhelmed while the other computing resources being underutilized.

In this research, we have presented a work load balancing approach utilizing **ANN** optimized by continuous ant colony optimization (**$ACO_R$**) algorithm for cloud computing. The performance of the model has been investigated and compared against related methodologies. The system is evaluated using BitBrains faststorage and rnd datasets.

Four experiments have been conducted on the proposed system using both types of datasets and the performance of the system is evaluated. These experiments investigate the performance of the model with respect to different window size datasets. In the first experiment, for the model, we provide a ten second time frame dataset selected with random forest feature selection of both versions (i.e. faststorage and rnd) datasets. The best performance is an accuracy of 84.4% for training and 83.3 % for testing on Faststorage dataset and 86.1% for training and 85.4% for testing on rnd dataset.

In the second experiment, at each time step fifteen second time frame resource utilization input data is provided to the model and CPU utilization of the system is predicted. The aim of this experiment was to see if the sliding window has any positive effect on the forecasting accuracy of

the models. The best performance achieved by the model is 91% training accuracy 89.5% test accuracy and 89% training and 89% testing on FastStorage and Rnd dataset respectively.

In the third experiment, at each time step twenty second time frame resource utilization input data is provided to the model and CPU utilization of the system is predicted. The aim of this experiment was also to see if the sliding window has any positive effect on the forecasting accuracy of the models. The best performance achieved by the model is 90% training accuracy 92.5% test accuracy and 91.5% training and 92.3 % testing on FastStorage and Rnd dataset respectively. The model performed better when we utilize the Rnd dataset, and the result clearly shows that accuracy of prediction model improves as the size of the training window increases.

The final experiment was conducted by feeding both twenty second window size CPU dataset of both FastStorage and Rnd CPU dataset. Applying FastStorage CPU dataset, the best performance achieved by the model was 96.3% on training dataset and 93.5% on testing dataset. The model was able achieve 95.2% training accuracy and 92.9 % testing accuracy when Rnd-CPU dataset is employed. The model achieved better result when CPU dataset is provided.

Finally, we compared the result from this model with Particle swarm (ANN-PSO), Genetic algorithm (ANN-GA), and Hybrid of $ACO_R$-BP neural network training methods. The result obtained by a hybrid of ANN-$ACO_R$ –BP is better than the result achieved by all algorithms on both versions of datasets except Particle swarm (ANN-PSO). The best performance achieved by Particle swarm (ANN-PSO) is an accuracy of 99.19% on training and 98.9% on testing while using FastStorage CPU datasets.

## 5.6 Future Works

- Future extensions to this work includes, a prediction model based on other machine learning approaches such as RNN, LSTM, and BLSTM with different attributes of ACO parameters.
- It is interesting to investigate the efficiency of the proposed method by combining other static or dynamic load balancing algorithms.
- How this method effect the process of scheduling and workload migration needs further studies.

## References


[1] CR. Chaudhary, G. S. Aujla, N. Kumar and J. J. P. C. Rodrigues, haudhary, G. S. Aujla, "Optimized Big Data Management across Multi-Cloud Data Centers: Software-Defined-Network-Based Analysis," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 118–126, 2018, doi: 10.1109/MCOM.2018.1700211.

[2] https://www.oberlo.com/blog/google-search-statistics, "google-search-statistics," Jan. 2020.

[3] "Cisco Global Cloud Index: Forecast and Methodology," *2015–2020 white Pap. Cisco Public; http//www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf,.*

[4] G. Soni and M. Kalra, "A novel approach for load balancing in cloud data center," in *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, 2014, pp. 807–812, doi: 10.1109/IAdCC.2014.6779427.

[5] N. & J. Mazher, "Efficient load balancing algorithm in cloud computing," *Indian J. Sci. Technol.*, vol. 10, no. 11, pp. 1–8, 2017, doi: 10.17485/ijst/2017/v10i11/110107.

[6] S. G. Domanal and G. R. M. Reddy, "Load Balancing in Cloud Environment Using a Novel Hybrid Scheduling Algorithm," in *Proceedings - 2015 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2015*, Mar. 2016, pp. 37–42, doi: 10.1109/CCEM.2015.31.

[7] G. Rastogi and R. Sushil, "Analytical literature survey on existing load balancing schemes in cloud computing," in *Proceedings of the 2015 International Conference on Green Computing and Internet of Things, ICGCIoT 2015*, Jan. 2016, pp. 1506–1510, doi: 10.1109/ICGCIoT.2015.7380705.

[8] M. and M. A. Katyal*, “A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment,” *Int. J. Distrib. Cloud Comput.*, vol. 1, no. 2, pp. 5–14, 2013.

[9] G. Ruhi, “Review on Existing Load Balancing Techniques of Cloud Computing,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 2, pp. 168–171, 2014.

[10] R. Hu, J. Jiang, G. Liu, and L. Wang, “Efficient resources provisioning based on load forecasting in cloud,” *Sci. World J.*, vol. 2014, 2014, doi: 10.1155/2014/321231.

[11] N. Roy, A. Dubey, and A. Gokhale, “Efficient autoscaling in the cloud using predictive models for workload forecasting,” in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 500–507, doi: 10.1109/CLOUD.2011.42.

[12] Y. Jiang, C. S. Perng, T. Li, and R. Chang, “ASAP: A self-adaptive prediction system for instant cloud resource demand provisioning,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2011, pp. 1104–1109, doi: 10.1109/ICDM.2011.25.

[13] A. Khan, X. Yan, S. Tao, and N. Anerousis, “Workload characterization and prediction in the cloud: A multiple time series approach,” in *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, 2012, pp. 1287–1294, doi: 10.1109/NOMS.2012.6212065.

[14] A. Al Sallami, N., “Load balancing with neural network.,” *IJACSA) Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 10, 2013.

[15] P. Liu, “Using Random Neural Network for Load Balancing in Data Centers,” in *ICOMP 2015*, 2015, pp. 3–8.

[16] M. Ghanem and A. Jantan, “Swarm intelligence and neural network for data classification,” in *Proceedings - 4th IEEE International Conference on Control System, Computing and*

*Engineering, ICCSCE 2014*, Mar. 2014, pp. 196–201, doi: 10.1109/ICCSCE.2014.7072714.

[17] C. Blum and K. Socha, "Training feed-forward neural networks with ant colony optimization: An application to pattern classification," in *Proceedings - HIS 2005: Fifth International Conference on Hybrid Intelligent Systems*, 2005, vol. 2005, pp. 233–238, doi: 10.1109/ICHIS.2005.104.

[18] " http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains, 2019.," 2019.

[19] W. Qing Li, Cheng Wang, Jing Wu, "Towards the business information technology alignment in cloud computing environment: anapproach based on collaboration points and agents," *Int. J. Comput. Integr. Manuf.*, vol. 24, no. 11, pp. 1038–1057, 2011.

[20] R. Prashant Gupta, A. Seetharaman, "The usage and adoption of cloud computing by small and medium businesses," *Int. J. Inf. Manage.*, vol. 33, no. 5, pp. 861–874, 2013.

[21] P. Mell and T. Grance, "The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology," in *Public Cloud Computing: Security and Privacy Guidelines*, Nova Science Publishers, Inc., 2012, pp. 97–101.

[22] Z. Sajid, Mohammad & Raza, "Cloud Computing: Issues & Challenges," *Int. Conf. Cloud*, pp. 35–41, 2013.

[23] A.Nandgaonkar, S.V. and Raut, "A Comprehensive Study on Cloud Computing," *A Compr. Study Cloud Comput.*, vol. 3, no. 4, pp. 733 – 738, 2014.

[24] L.Devasena, "Impact Study of Cloud Computing On Business Development. Operations Research and Applications," *An Int. J.*, vol. 1, no. 1, pp. 1–7, 2014.

[25] U. Joshi, Shalini & Kumari, "Load balancing in cloud computing: Challenges & issues," vol. 10, no. 1109, pp. 120–125, 2016.

[26] T.V. Anthony T. Velte, Robert Elsenpeter, *Cloud Computing: A Practical Approach*. 2009.

[27] Intel.com.2020, "cloud-computing-virtualization-building-private-iaas-guide," *www.intel.com/content/dam/www/public/us/e n/documents/guides/cloud-computing-virtualization-building-private-iaas-guide*, 2020. .

[28] K. Koganti1, E. Patnala, S. S. Narasingu, "Virtualization technology in cloud computing environment," *Intl. J. Emerg. Technol. Adv. Eng.*, vol. 3, no. 3, pp. 771–773, 2013.

[29] R. K. and S. Kolesnikova, "Evaluation of Hypervisor Stability Towards Insider Attacks," *J. Electron. Sci. Technol.*, vol. 14, no. 1, 2016.

[30] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, "A View of Cloud Computing, Communications of the ACM," vol. 53, no. 4, pp. 50–58, 2010.

[31] I. Takouna., "Infrastructure as a service security: Challenges and solutions," *7th Intl. Conf. Informatics Syst.*, 2015.

[32] M. T. Nader Benmessaoud, CJ Williams, Uma Mahesh Mudigonda, "Microsoft System Center: Network Virtualization and Cloud Computing," vol. 94, 2014.

[33] D. from T. (n.d.), "What is Network Virtualization?, "*https://www.techopedia.com/definition/655/network-virtualization.*, Jan.2020.

[34] Www.salesforce.com/products/platform/best-practices/benefits-of-cloud-computing, "12 Benefits of Cloud Computing and Its Advantages."

[35] F. Moghaddam, M. Ahmadi, S. Sarvari, M. Eslami, and A. Golkar, "Cloud computing challenges and opportunities: A survey," in *2015 International Conference on Telematics and Future Generation Networks, TAFGEN 2015*, Oct. 2015, pp. 34–38, doi:

10.1109/TAFGEN.2015.7289571.

[36] R. Mukhopadhyay, D. Ghosh, “A study on the application of existing load balancing algorithms for large, dynamic, heterogeneous distributed systems,” *Proc. WSEAS Internatinal Conf. Softw. Eng. Parallel Distrib. Syst. (SEPADS),* pp. 238--243, 2010.

[37] S. A. and M. A. Shah, “Load balancing algorithms in cloud computing: A survey of modern techniques,” *Natl. Softw. Eng. Conf.*, vol. 10, no. 1109, pp. 30–35, 2015.

[38] A. M. Alakeel, “A Guide to Dynamic Load Balancing in Distributed Computer Systems,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 10, no. 6, 2010.

[39] J. Uma, V. Ramasamy, and A. Kaleeswaran, “Load Balancing Algorithms in Cloud Computing Environment - A Methodical Comparison,” *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 3, no. 2, pp. 272–275, 2014, [Online]. Available: http://ijarcet.org/wp-content/uploads/IJARCET-VOL-3-ISSUE-2-272-275.pdf.

[40] S. B. and M. Rana, “A study on load balancing in cloud computing environment using evolutionary and swarm based algorithms,” *Int. Conf. Control. Instrumentation, Commun. Comput. Technol.*, pp. 245–250, 2014.

[41] P. Werstein, H. Situ, “Load balancing in a cluster computing,” *Proc. Int. Conf. Parallel Distrib. Comput. Appl. Technol.*, vol. IEEE Compt, pp. 569--577, 2006.

[42] M. Beltran, A. Guzman, “Dealing with heterogeneity in load balancing algorithms,” *Proc. Int. Symp. Parallel Distrib. Comput. Comput. Soc. Washingt.*, pp. 123--132, 2006.

[43] R. Chakraverty S., Sahoo D.M., “McCulloch–Pitts Neural Network Model. In: Concepts of Soft Computing,” *Springer, Singapore. https://doi.org/10.1007/978-981-13-7430-2_11*, vol. 10, no. 1007, 2019.

[44] I. N. da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves,

*Artificial neural networks: A practical course*. Springer International Publishing, 2016.

[45] H. Hassoun, "Fundamentals of artificial neural networks," *MIT Press,* 1995.

[46] K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3. pp. 31–44, Mar. 1996, doi: 10.1109/2.485891.

[47] J. R. Zhang, J. Zhang, T. M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training," *Appl. Math. Comput.*, vol. 185, no. 2, pp. 1026–1037, Feb. 2007, doi: 10.1016/j.amc.2006.07.025.

[48] C. U. Joy, "Comparing the Performance of Backpropagation Algorithm and Genetic Algorithms in Pattern Recognition Problems," *Int. J. Comput. Inf. Sstems*, vol. 2, no. 5, pp. 7–12, 2011.

[49] A.Kulluk, Sinem, Lale Ozbakir, "Training neural networks with harmony search algorithms for classification problems.," *Eng. Appl. Artif. Intell.*, vol. 25, no. 1, pp. 11–19, 2019.

[50] A. Wahab, Mohd Nadhir Ab, Samia Nefti-Meziani, "A comprehensive review of swarm optimization algorithms," *PLoS One*, vol. 10, no. 5, pp. 1–36, 2015.

[51] T. S. "Ant colony optimization. Dorigo, Marco, Mauro Birattari, "Ant colony optimization," *EEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006.

[52] H. Goldberg, David E., "Genetic algorithms and machine learning.," *deepblue.lib.umich.edu*, 1988.

[53] L. Er, Meng Joo, "Genetic algorithms for MLP neural network parameters optimization," *2009 Chinese Control Decis. Conf. IEEE*, 2009.

[54] A. Zanchettin, Cleber, Teresa B. Ludermir, "Hybrid training method for MLP: optimization of architecture and training," *IEEE Trans. Syst. Man, Cybern. Part B*, vol.

41, no. 45, pp. 1097–1109, 2011.

[55] A. Shayeghi, Hossein, H. A. Shayanfar, “A hybrid particle swarm optimization back propagation algorithm for short term load forecasting.,” *Int. J. Tech. Phys. Probl. Eng.*, vol. 1, no. 3, pp. 12–22, 2010.

[56] S. Kartheeswaran and D. D. C. Durairaj, “A hybrid genetic algorithm and back-propagation artificial neural network based simulation system for medical image reconstruction in noise-added magnetic resonance imaging data,” Apr. 2016, doi: 10.1109/GET.2015.7453863.

[57] Dorigo, Marco, “Ant colony system: a cooperative learning approach to the traveling salesman problem.,” *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, 1997.

[58] H. Stützle, Thomas, “MAX–MIN ant system.,” *Futur. Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.

[59] S. Blum, Christian, “Training feed-forward neural networks with ant colony optimization: An application to pattern classification.,” *Fifth Int. Conf. Hybrid Intell. Syst.*, vol. IEEE, 2005.

[60] M. Samal, Pooja, “Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing.,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 3, pp. 416-419., 2013.

[61] Ž. Radojević, Branko, “Analysis of issues with load balancing algorithms in hosted (cloud) environments.,” *2011 Proc. 34th Int. Conv. MIPRO.*, vol. IEEE, pp. 416–420, 2011.

[62] K. LD, Dhinesh Babu, “Honey bee behavior inspired load balancing of tasks in cloud computing environments.,” *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2292–2303, 2013.

[63] V. B. and A. Kumar, “Cloud computing: Performance analysis of load balancing

algorithms in cloud heterogeneous environment," *2014 5th Int. Conf. - Conflu. Next Gener. Inf. Technol. Summit (Confluence),* pp. 200–205, 2014.

[64] U. Joshi, Shalini & Kumari, "A Comprehensive Analysis of Load Balancing Algorithms in Cloud Computing."

[65] Dasgupta, Kousik, "A genetic algorithm (ga) based load balancing strategy for cloud computing.," *Procedia Technol.*, vol. 10, pp. 340-347., 2013.

[66] J. W. Gao, Ren, "Dynamic load balancing strategy for cloud computing with ant colony optimization.," *Futur. Internet*, vol. 7, no. 4, pp. 465–483, 2015.

[67] Islam, Sadeka, "Empirical prediction models for adaptive resource provisioning in the cloud.," *Futur. Gener. Comput. Syst.*, vol. 28, no. 1, pp. 155–162, 2012.

[68] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013*, 2013, pp. 357–364, doi: 10.1109/SEAA.2013.23.

[69] F. Farahnakian, T. Pahikkala, P. Liljeberg, "Energy Aware Consolidation Algorithm Based on K-nearest Neighbour Regression for Cloud Centers," *IEEE 6th Int. Conf. Util. Cloud Comput.*, 2013.

[70] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Proceedings - 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2014*, 2014, pp. 500–507, doi: 10.1109/PDP.2014.109.

[71] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing," in

*Proceedings - 2015 IEEE 8th International Conference on Cloud Computing, CLOUD 2015*, Aug. 2015, pp. 381–388, doi: 10.1109/CLOUD.2015.58.

[72] M. Duggan, J. Duggan, E. Howley, and E. Barrett, "A reinforcement learning approach for the scheduling of live migration from under utilised hosts," *Memetic Comput.*, vol. 9, no. 4, pp. 283–293, Dec. 2017, doi: 10.1007/s12293-016-0218-x.

[73] M. Patel, S. Chaudhary, "Machine Learning Based Statistical Prediction Model for Improving Performance of Live Virtual Machine Migration," *Hindawi Publ. Corp. J. Eng.*, 2016.

[74] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," in *2017 12th International Conference for Internet Technology and Secured Transactions, ICITST 2017*, May 2018, pp. 67–72, doi: 10.23919/ICITST.2017.8356348.

[75] Q. Zia Ullah, S. Hassan, and G. M. Khan, "Adaptive Resource Utilization Prediction System for Infrastructure as a Service Cloud," *Comput. Intell. Neurosci.*, vol. 2017, 2017, doi: 10.1155/2017/4873459.

[76] A. Abdelsamea, A. A. El-Moursy, E. E. Hemayed, and H. Eldeeb, "Virtual machine consolidation enhancement using hybrid regression algorithms Virtual machine consolidation enhancement," *Egypt. Informatics J.*, vol. 18, no. 3, pp. 161–170, Nov. 2017, doi: 10.1016/j.eij.2016.12.002.

[77] M. A. Khoshkholghi, M. N. Derahman, A. Abdullah, S. Subramaniam, and M. Othman, "Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data Centers," *IEEE Access*, vol. 5, pp. 10709–10722, 2017, doi: 10.1109/ACCESS.2017.2711043.

[78] R. Shaw, E. Howley, and E. Barrett, “An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers,” in *2017 12th International Conference for Internet Technology and Secured Transactions, ICITST 2017*, May 2018, pp. 61–66, doi: 10.23919/ICITST.2017.8356347.

[79] S. Sotiriadis, N. Bessis, and R. Buyya, “Self managed virtual machine scheduling in Cloud systems,” *Inf. Sci. (Ny).*, vol. 433–434, pp. 381–400, Apr. 2018, doi: 10.1016/j.ins.2017.07.006.

[80] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, “Predicting host CPU utilization in the cloud using evolutionary neural networks,” *Futur. Gener. Comput. Syst.*, vol. 86, pp. 162–173, Sep. 2018, doi: 10.1016/j.future.2018.03.040.

[81] A. Rawat, R. Sushil, A. Agarwal, and A. Sikander, “A New Approach for VM Failure Prediction using Stochastic Model in Cloud,” *IETE J. Res.*, 2018, doi: 10.1080/03772063.2018.1537814.

[82] N. K. Gondhi and P. Kailu, “Prediction Based Energy Efficient Virtual Machine Consolidation in Cloud Computing,” in *Proceedings - 2015 2nd IEEE International Conference on Advances in Computing and Communication Engineering, ICACCE 2015*, Oct. 2015, pp. 437–441, doi: 10.1109/ICACCE.2015.148.

[83] Al Nuaimi, Klaithem, “A survey of load balancing in cloud computing: Challenges and algorithms.,” *2012 Second Symp. Netw. cloud Comput. Appl.*, vol. IEEE, pp. 137–142, 2012.

[84] P. Desai, Tushar, “A survey of various load balancing techniques and challenges in cloud computing,” *Int. J. Sci. Technol. Res.*, vol. 2, no. 11, pp. 158–161, 2013.

[85] T. Katoch, Swati, “Load balancing algoritms in cloud computing environment: a review,”

*Int. J. Recent Innov. Trends Comput. Commun.*, vol. 2, no. 8, pp. 2151–2156, 2014.

[86] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11. Elsevier Ltd, Nov. 01, 2018, doi: 10.1016/j.heliyon.2018.e00938.

[87] D. Wang, H. He, and D. Liu, "Intelligent Optimal Control with Critic Learning for a Nonlinear Overhead Crane System," *IEEE Trans. Ind. Informatics*, vol. 14, no. 7, pp. 2932–2940, Jul. 2018, doi: 10.1109/TII.2017.2771256.

[88] T. F. Awolusi, O. L. Oke, O. O. Akinkurolere, A. O. Sojobi, and O. G. Aluko, "Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete," *Heliyon*, vol. 5, no. 1, Jan. 2019, doi: 10.1016/j.heliyon.2018.e01115.

[89] S. Sremath Tirumala and G. Chen, "Load Balancing in a Distributed Network Environment -An ACO Inspired Approach," *IJCSN Int. J. Comput. Sci. Netw. ISSN*, vol. 4, no. 3, pp. 2277–5420, 2015, [Online]. Available: www.IJCSN.org.

[90] X. Zhou *et al.*, "Load balancing prediction method of cloud storage based on analytic hierarchy process and hybrid hierarchical genetic algorithm," *Springerplus*, vol. 5, no. 1, Dec. 2016, doi: 10.1186/s40064-016-3619-x.

[91] "https://commons.apache.org/proper/commons-math/." .

[92] "https://www.heatonresearch.com/2017/06/01/hidden-layers.html,"

[93] K. Socha, "Ant Colony Optimisation for Continuous and Mixed-variable Domains," Universite Libre de Bruxelles, 2009.