

# Space Delay-Tolerant Networks Routing Using Artificial Neural Networks

A Thesis

Presented to

The Faculty of Department of Engineering Technology

University of Houston

In Partial Fulfillment

of The Requirements of The Degree of

Master of Science

in Engineering Technology

Saddam M Al Asadi

Nov. 2018

ProQuest Number:28180717

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent on the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 28180717

Published by ProQuest LLC (2020). Copyright of the Dissertation is held by the Author.

All Rights Reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 - 1346

# Space Delay-Tolerant Networks Routing

## Using Artificial Neural Networks

---

Saddam M Al Asadi

APPROVED:

---

Dr. Ricardo Lent, PhD  
Department of Engineering Technology

---

Dr. Junko Sugawara, PhD  
Department of Construction Management

---

Dr. Xiaojing Yuan, PhD  
Department of Engineering Technology

---

Dr. George Zouridakis, Associate Dean  
College of Technology

---

Dr. Wajiha Shireen, Professor and Chair  
Department of Engineering Technology

# Dedication

*To the shelter who kept away fright  
"Dad,"  
to the candle in dark nights  
"Mom,"  
to my sister "Huda" you made this possible,  
and to my brothers and sisters.*

*Saddam M. Al Asadi  
Nov. 2018*

# Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Dr. Ricardo Lent for the continuous support of my master's research and work, for believing that I can make this achievement, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me through the whole time of research and writing of this thesis. Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Xiaojin Yuan and Dr. Junko Sugawara, for their help sharpening this work, for giving me their time, effort, encouragement, insightful comments, and helpful questions. Finally, I would like to thank my friend Jose A Cruz Batista for his time and help in both thesis defense day and proofreading.

# Abstract

Most of the present communication networks count on having an end-to-end association between the sender and receiver to be able to establish a connection; unfortunately, that is not always possible due to obstacles or environmental difficulties. Delay Tolerant Networks (DTN) is the type of networks that is specially designed to overcome those difficulties. Instead of having an end-to-end connection between nodes, In general, DTN routing protocols use a "save when no link" and "forward when possible" policy.

Machine learning is an engineering field that uses statistical techniques and use it to make a system learn. This research will use machine learning technology, Artificial Neural Networks (ANN) specifically and deploy it to perform DTN routing. The neural network will be taught a database of previous patterns of network node Contact Graph (CG), in other words, the neural network will learn on when nodes will have direct sight of each other.

After training the network, a network node should be able to make the best decisions about the best next hop to reach the destination with minimum acceptable error rate.

The main advantage is that each node will be already trained on the best possible routes before the actual routing decisions need to be made, so by the time that any node is handed a bundle, it already knows where to forward the bundle instead of just flooding the network almost randomly trying to reach the destination like traditional routing protocols. A well-trained DTN node will be knowing and waiting for the next hop to appear in sight, and it will deliver its data as soon as the node-to-node connection is established. To the best of our knowledge, this is the first work that introduces and utilizes artificial neural networks to implement delay-tolerant networks routing.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background and Motivation . . . . .	4
	Delay Tolerant Networks Difficulties and Constrains . . . . .	4
1.1.1	DTN Contacts . . . . .	5
1.1.2	DTN Routes and Links Characteristics . . . . .	6
1.1.3	DTN Architecture . . . . .	7
1.1.4	DTN Nodes Characteristics . . . . .	8
1.2	Thesis Objective and Approach . . . . .	9
1.3	Contribution . . . . .	10
1.4	Thesis Organization . . . . .	11
<b>2</b>	<b>Literature Review</b>	<b>12</b>
2.1	Major DTN Routing Protocols . . . . .	12
2.1.1	Direct Transmission . . . . .	12
2.1.2	First Contact . . . . .	13
2.1.3	EPIDEMIC Routing . . . . .	13
2.1.4	ProPHET . . . . .	15



2.1.5	Spray And Wait . . . . .	15
2.1.6	Binary Spray and Wait . . . . .	16
2.1.7	Spray and Focus . . . . .	16
2.1.8	PRoPHET+ . . . . .	17
2.1.9	MaxProb . . . . .	17
2.1.10	Contact Graph Routing . . . . .	18
2.2	Additional Literature . . . . .	19
<b>3</b>	<b>Proposed Approach</b>	<b>23</b>
3.1	Proposed Approach Background . . . . .	23
3.1.1	Artificial Intelligence . . . . .	23
3.1.2	Machine Learning . . . . .	24
3.1.3	Deep Learning . . . . .	24
3.1.4	Artificial Neural Networks . . . . .	24
3.2	Proposed Solution . . . . .	37
<b>4</b>	<b>Experimental Evaluation and Tuning</b>	<b>43</b>
4.1	Thesis Work Experiments . . . . .	45
4.1.1	Neural Networks Depth Experiment . . . . .	45
4.1.2	Gradual Neural Network Widening Experiment . . . . .	45
4.1.3	Activation Functions Experiments . . . . .	46
4.1.4	Exponential Increase of Neural Network Width Experiment . . . . .	47
4.1.5	The Cone Effect Experiment . . . . .	48
4.1.6	Final Experiment: Space DTN ANN Routing Solution Testing . . . . .	48
4.2	Testbed . . . . .	49

4.2.1	Hardware . . . . .	49
4.2.2	Software . . . . .	50
4.2.3	Datasets . . . . .	50
<b>5</b>	<b>Experiments and Results</b>	<b>52</b>
5.1	Neural Networks Depth Experiment . . . . .	53
5.2	Gradual Neural Network Widening Experiment . . . . .	59
5.3	Activation Functions Experiments . . . . .	63
5.3.1	Relu Activation Function Experiment . . . . .	65
5.3.2	Softmax Activation Function Experiment . . . . .	66
5.3.3	Tanh Activation Function Experiment . . . . .	67
5.3.4	Sigmoid Activation Function Experiment . . . . .	69
5.3.5	Final Activation Functions Performance Comparison . . . . .	69
5.4	Exponential Increase of Neural Network Width Experiment . . . . .	72
5.5	The Cone Effect Experiment . . . . .	86
5.5.1	All-Layers-Wide Network Experiment . . . . .	87
5.5.2	Cone Shape Network Experiment . . . . .	89
5.6	Final Space DTN ANN Routing Solution . . . . .	94
<b>6</b>	<b>Conclusions and Future Work</b>	<b>101</b>
6.1	Conclusions . . . . .	101
6.2	Future Work . . . . .	104

# List of Figures

1.1	NASA Example of Space DTN Network . . . . .	8
3.1	Handwritten Numbers Image . . . . .	27
3.2	Perceptron Architecture . . . . .	28
3.3	Feed-Forward Neural Network . . . . .	29
3.4	Dense Deep Feed-Forward Neural Network . . . . .	30
3.5	Recurrent Neural Network . . . . .	30
3.6	Fully-Connected Neural Network . . . . .	31
3.7	Relu Activation Function . . . . .	32
3.8	Tanh Activation Function . . . . .	32
3.9	Sigmoid Activation Function . . . . .	33
3.10	SoftMax Activation Function Example . . . . .	34
3.11	Different Uses of Learning Per Data Type . . . . .	35
3.12	Binary Accuracy Calculation . . . . .	36
3.13	Proposed Solution First Step Design . . . . .	38
3.14	Proposed Solution Third Step . . . . .	41
3.15	Testing The Final ANN model . . . . .	41
3.16	Proposed Solution: DTN Artificial Neural Network Routing. . . . .	42

5.1	Different Number of layers training Accuracy comparison . . . . .	54
5.2	Training Accuracy vs No. of Layers . . . . .	55
5.3	Different Number of layers training Loss comparison . . . . .	56
5.4	Training Loss vs No. of Layers . . . . .	57
5.5	Different Number of layers Validation Accuracy comparison . . . . .	58
5.6	Validation Accuracy vs No. of Layers . . . . .	59
5.7	Different Number of layers Validation Loss comparison . . . . .	60
5.8	Validation Loss vs No. of Layers . . . . .	60
5.9	Effect of Widening Layers on Final Training Accuracy . . . . .	62
5.10	Effect of Widening Layers on Final Training Loss . . . . .	63
5.11	Effect of Widening Layers on Training Time Elapsed . . . . .	64
5.12	Activation Functions Experiment Neural Network Structure . . . . .	65
5.13	ReLU Activation Function Performance Per Epoch . . . . .	66
5.14	Softmax Activation Function Performance Per Epoch . . . . .	67
5.15	Tanh Activation Function Performance Per Epoch . . . . .	68
5.16	Sigmoid Activation Function Performance Per Epoch . . . . .	70
5.17	Activation Functions Training Accuracy Comparison . . . . .	71
5.18	Activation Functions Training Loss Comparison . . . . .	71
5.19	$10^0$ Neuron Width Neural Network Architecture . . . . .	73
5.20	Different Width Neural Networks Training Accuracy Per Epoch Number	74
5.21	Different Width Neural Networks Training Loss Per Epoch Number . .	75
5.22	Different Width Neural Networks Validation Accuracy Per Epoch Num- ber . . . . .	76
5.23	Different Widths Neural Networks Validation Loss Per Epoch Number .	77

5.24	$10^1$ Neuron Width Neural Network Architecture . . . . .	77
5.25	$10^2$ Neuron Width Neural Network Architecture . . . . .	79
5.26	$10^3$ Neuron Width Neural Network Architecture . . . . .	82
5.27	Training Accuracy Per Layer Width . . . . .	84
5.28	Training Loss Per Layer Width . . . . .	84
5.29	Validation Accuracy Per Layer Width . . . . .	85
5.30	Validation Loss Per Layer Width . . . . .	86
5.31	Wide Layers Neural Network Structure . . . . .	88
5.32	Training Accuracy of Cone-Shaped vs Wide ANNs Per Epoch . . . . .	89
5.33	Training Loss of Cone-Shaped vs Wide ANNs Per Epoch . . . . .	90
5.34	Validation Accuracy of Cone-Shaped vs Wide ANNs Per Epoch . . . . .	91
5.35	Validation Loss of Cone-Shaped vs Wide ANNs Per Epoch . . . . .	92
5.36	Cone-Shaped Neural Network Structure . . . . .	92
5.37	Wide Vs Cone-Shaped Training Accuracy and Loss . . . . .	93
5.38	Wide vs Cone-Shaped Validation Accuracy and Loss . . . . .	94
5.39	Proposed Neural Network Structure . . . . .	95
5.40	Proposed Solution Training Accuracy and Training Loss Per Epoch . . . . .	96
5.41	Proposed Solution Training Accuracy and Training Loss Trend . . . . .	96
5.42	Proposed Solution Validation Accuracy and Loss Per Epoch . . . . .	97
5.43	Proposed Solution Validation Accuracy and Validation Loss Trends Per Epoch . . . . .	98
5.44	Validation Accuracy of Proposed Solution vs Ideal World Scenario . . . . .	100

# Chapter 1

## Introduction

Communication networks are not only spreading rapidly all around us, but they are also getting more advanced in the method they are handling their connectivity. Most of the current networks use protocols like Transfer Control Protocol (TCP)/Internet Protocol (IP) and User Datagram Protocol (UDP), and those require a virtual end-to-end connection between the two endpoints. The infrastructure for such types of networks is usually well organized, maintained, and they are also located in secure and environmentally controlled spaces. That makes the traditional communications networks eligible to have powerful devices and reliable power sources. Furthermore, and due to all those factors, redundancy is one of the main characteristics of these networks. Powerful devices, reliable power sources, environmentally controlled secured spaces, and a wide range of shareable routes enable these networks to have high communication reliability, fast end-to-end data rates, and fewer error rates. Not only traditional communication networks have smaller error rates, but also when an error happens, they are quicker to detect errors and send back data to recover the lost data as the case in TCP/IP or to

continue transmitting data as in UDP. Unlike TCP/IP, UDP keeps sending data to the destination in real time mostly without user-felt service interruption, as in this protocol continuity of streaming is the goal rather than an utterly error-free service experience as in TCP/IP.

What if the communication faces challenges to keep that virtual connection, like being in the deep space and communicating back and forth into earth? [1] that is when DTN [2] [3] comes in place. DTN are networks that lack continuous end-to-end connections among their nodes due to node mobility, constrained power sources, or limited data storage [4]. The main aspect of DTN networks is that they overcome the intermittent node-to-node connections by storing data when next hop is not available and forwarding data when next hop is available. The methodology of forwarding data depends on the DTN routing protocol used and on the stage of the process of the protocol itself. There are many DTN routing protocols some are simple, like Direct Transmission, First Contact, and Epidemic; others are more complex like Probabilistic Routing Protocol using History of Encounters and Transitivity (ProPHET), Spray and Wait, and Binary Spray and Wait.

Most research and digital solutions use human-written codes to solve present problems and future time predictions. A code is just a series of commands written by a human being to be executed on a computer. Having the code written by programmers means that they reflect and follow their chain of thoughts and state them as specific fixed steps to solve the problem as best of their knowledge can provide. As human being's intention for achieving more than ever before, having traditional tools that are already

handmade to reach goals that are behind their imagination is becoming an obstacle that slows down the whole scientific progress, in other words, hand-made codes are reaching their limit to achieve goals as big as human beings' dreams. Not only are we witnessing a stage of a need for more powerful solutions than mostly fixed hand-made steps of commands, but also executing these codes is getting slower as the problems to be solved are getting more complex and the codes are executed sequentially.

For most problem-solving cases, code solutions represent a bottle-neck to achieve bigger scientific goals as those solutions count on relatively-slow human interactions to keep working. Not only faster processing solutions are needed, but also autonomous ones. That is when Machine Learning (ML) comes in place. Machine learning is a technology that uses machine speed to solve problems autonomously. One important branch of machine learning is ANN. ANN is the technology that not only deploys the high machine speed, but also it mimics the human brain to solve problems and performs vast amounts of calculations in parallel rather than traditional sequential codes method.

This research will use machine learning technology, (ANN), specifically and deploy it to perform DTN routing. The ANN will be taught a database of previous patterns of network node CG, in other words, the ANN will learn on when nodes will have a direct line-of-sight of each other. After training the network, a network node should be able to make the best decisions about the best next hop to reach the destination with minimum acceptable error rate.



## 1.1 Background and Motivation

This thesis work is motivated by the goal of overcoming the challenges and difficulties that DTNs face due to their structure and the environmental hostility surrounds them, that is by creating a better routing solution, a solution that will avoid the shortcomings of the traditional routing protocols controlling DTN operations. For the sake of explaining the motivation of this thesis work, a detailed review of all these difficulties and constraints is discussed in depth in the sections below; furthermore, a comprehensive review of DTN routing protocols and their disadvantages that this thesis work aims to overcome will be reviewed in section 2.1.

### DTN Difficulties and Constraints

Networks that lack continuous end-to-end connections among their nodes due to node mobility, constrained power sources, or limited data storage space are called DTN [4]. DTN could also be defined as the occasionally-connected networks that may undergo many partitioning or disconnection [5]. This could be related to networks that are having constraints such as lack of infrastructure, disconnection, disruption, and lack of resources. DTN networks are used when power consumption and direct connection are not always available, for example, DTN is used in deep space and animal monitoring [1]. In general, DTN nodes perform two tasks, store the bundles and forward a copy to one or more nodes. DTN protocols differ from each other in the methodology of them spreading the bundles and the criteria they use to decide that.

1. Exotic Media Networks: that includes near earth and Inter-planet satellite networks, these are the types of networks that are spread among earth, satellites,

planets, and aircraft. This type of network usually produces big amounts of data that need to be sent back and forth to the space bases on earth. The internet service on this network is called Interplanetary (IPN) Internet [6] [7].

2. Sensors/actuator networks: where sensors are having limited resources and they work periodically to sense and send data, this network is widely used in animal monitoring aforementioned [7] [3].
3. Military networks: these are the networks in which nodes move with the military wherever it goes, these networks are designed to keep working even if some nodes are lost under action [7].

### **1.1.1 DTN Contacts**

Contacts are the nodes that the DTN network consists of. Depending on whether the network nodes are having previous knowledge of other nodes or not, we can classify DTN contacts into two types:

1. Scheduled or predictable Contacts (Deterministic): these are the contacts that get encountered according to a certain pattern, for example, moon, earth, and other planets.
2. Intermittent Opportunistic Contacts: these are the contacts that are non-predictable. This case applies to disasters where users are completely disconnected from the rest of the world, and they don't know when they will have a connection using,

for example, an emergency network node.

To be able to provide end-to-end connectivity across heterogeneous networks, DTN relies on creating a “Bundle Layer” between two layers of the Open System Interconnection (OSI) model, those are transport and application layers [8].

### **1.1.2 DTN Routes and Links Characteristics**

Unlike traditional networks; DTNs face many challenges like delays, higher error rates, and instability. It is good to mention some of DTNs characteristics that these challenges produce.

1. **High Latency and Low Data Rate:** In addition to processing and queueing delays, links in DTN suffer from propagation delays which are caused by the transmission medium. One example of that is the underwater communication, despite that some links can only carry speed of 10 kbps, the accompanied link delays could reach one second. Another consideration of link delays is that DTNs usually face asymmetries between the up and down links, that is due to the continuous change in the relative nodes locations. If we take for example deep space, planets, and satellites are on continuous movement; thus a path that consists of certain sequence and link properties to send a data signal will not be the same path to receive the response. To be more accurate, most of the time if the sequence is the same for sending and receiving the data, links in that path could never match between up-link and down-link only in the case of having all nodes stationary,

which is likely not the case.

2. **Disconnection:** Disconnection happens in DTNs due to many reasons. It could be because of a faulty node. Other reasons could be due to the movement of nodes, which could be a random walk to single or both nodes of each link or could be periodic due to satellites and planets movements on orbits. It is good to mention also that some cut offs occur due to the nodes periodic power saving, like low power nodes in sensors networks.
3. **Unexpected lengthened queuing:** in conventional networks, queuing delays could be parts of seconds to seconds in extreme cases. But due to the abnormal circumstances of DTNs queuing delays could reach days if not hours. What contributes to those longer queuing delays is the nature of DTNs where many copies are traversing among the nodes, not to mention the retransmission that takes place when data cannot reach their destinations.
4. **High Error Rate:** Due to nodes limitations and hostile environment, the error rate is much bigger than of the traditional networks. However, services and applications should keep working even with a Bit Error Rate (BER) of  $10^{-3}$  [9].

### **1.1.3 DTN Architecture**

The special thing about DTNs is that they don't have a systemized architecture because the focus of building them is still on producing more reliable connectivity rather than standardizing many small networks under one umbrella. Figure 1.1 shows an example

of a space DTN.

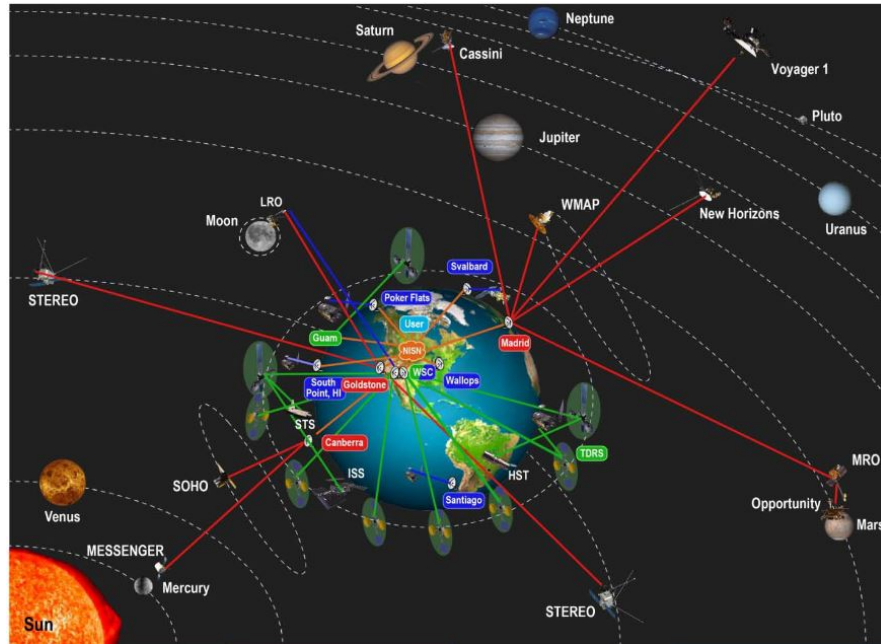


Figure 1.1: An example of Space DTN by NASA

courtesy of [www.nasa.gov](http://www.nasa.gov)

#### 1.1.4 DTN Nodes Characteristics

1. **Limited Endurance:** since DTNs are made to cover spaces where traditional networks cannot work; they usually spread around hostile environments. Some of the environments could be a desert where an army is moving or the wilderness where tracking devices are attached to some animals. It is important to achieve connectivity in such harsh environments, and the limitation of DTN nodes delegates the delivery assurance to nodes that rely just outside DTNs where nodes are provided with better power supply and maintenance.

2. **Low Duty Cycle Operation:** Due to power limitation, most of DTN nodes are turned on periodically to preserve power. DTN nodes will turn on and off on a timely manner just in enough frequency and time length to keep the network end-to-end connectivity while saving power as much as possible for the longest possible timespans.
3. **Limited Resources:** To keep nodes working the longest, not only low duty cycle type of operation is vital, limiting the resources of the node itself can make a lot of difference when it comes to saving power. Two nodes under same circumstances and duty cycle will not last for the same time if one of them uses less power to function, that is by having, for example, less memory, transmission range, and power of processing unit [7].

## 1.2 Thesis Objective and Approach

The goal of this thesis work is to use machine learning technology and experiment with, design, and then build an ANN to deploy it as an effective alternate DTN routing solution. The reason for turning into machine learning as a routing solution is to overcome the constraints of flooding, leftover bundle copies, and power and time-consuming routing-calculations that DTNs face under the present conventional DTN routing protocols.

The approach to reach the goal of this thesis is to build ANNs and experiment with the possible modifications and variations that could be done to them and examine the

effect of that on the ANN output and the measured performance metrics. Performing a set of experiments on the artificial neural networks will enable us to have a broad knowledge about the specific effects of each alteration and modification. The experience and lessons learned from the experiments will be used as an insight to build the final space DTN artificial neural network routing system.

### 1.3 Contribution

This research will find a method to train ANN on a DTN CG and use the trained model to give routing knowledge to its nodes. Knowing the future events yet to come is a huge advantage that represents the power of making decisions the earliest possible. Future knowledge means each node will already be trained on the best possible routes before the actual routing decisions need to occur, so by the time that any node is handed a bundle, it already knows where to forward the bundle instead of just flooding the network almost randomly trying to reach the destination like traditional routing protocols, or to start a usually overwhelming overhead process to find the best routes. A well-trained DTN node will be knowing and waiting for the next hop to appear in sight, and it will deliver its data as soon as the node-to-node connection is established. Creating a machine learning solution in general ( an artificial neural network solution in specific) will contribute in many ways:

1. It will introduce machine learning and specifically artificial neural networks to routing world in general and DTN routing field in specific.
2. It will contribute by demonstrating that DTN routing without using algorithms or

traditional coding is possible.

3. It will spotlight current DTN routing protocols by introducing a unified survey.
4. This work will also contribute by introducing many possible future achievements that will just become possible by proposing this work. This thesis work could be the basement which many new possibilities could be built on top of.

## **1.4 Thesis Organization**

This thesis is organized into six chapters, the first chapter is Introduction which includes the background of DTN and its difficulties and constraints as a motivation for this thesis work, it also includes objective and approach, contribution, and thesis organization; the second chapter is a literature review that is organized according to DTN routing protocols; the third chapter is exploring the proposed approach; the fourth is presenting the experimental evaluation and tuning as well as listing the details of the testbed; the fifth is discussing the experiments and results in details; and finally the sixth chapter is listing the conclusions and future work suggestions.



# Chapter 2

## Literature Review

This chapter surveys some of the relevant literature in the active area of DTN routing to explore the findings and achievements of the researchers over the years until the present time. Here are some of these intellectuals contributions organized according to their specific point of interest.

### 2.1 Major DTN Routing Protocols

#### 2.1.1 Direct Transmission

This routing protocol could be considered the simplest. A node forwards a message to another node it encounters, only if this last node is the message's destination. That happens when the sender waits all the time until it has a direct sight with the destination. The whole process of this protocol occurs in one step transmission. Advantages of this protocol are that it needs minimum resources, while the disadvantages are that it

is extremely slow, having one copy of the data risking the whole data to be lost in case we lose this single copy data, and the delivery delay cannot be predicted [10] [11].

### **2.1.2 First Contact**

In this routing protocol, each node sends the message (not a copy of it) to any node as soon as it starts having contact with it. And from a list of possible current contacts, an edge is chosen randomly. If the nodes know nothing about the evolving network topology, then all the nodes can do is to forward the bundles to their neighbors. A possible decision is to transmit a message to a randomly selected next-hop. This is the approach of the First Contact because of the random next hop choice, in some cases using this protocol could cause the message to move away from the destination, and thus the bundle faces significant delays. The advantage of this protocol is the shorter delay compared to other protocols. Disadvantages are that there will be many messages wandering around the network due to random next hop divergence effect [12].

### **2.1.3 EPIDEMIC Routing**

EPIDEMIC [13] [14] routing is flooding-based, as nodes continuously replicate and transmit messages to newly discovered contacts that do not already possess a copy of the bundle, that is by using a summary vector in which the node keeps a record of each node it encounters and handshake bundles with [15]. The advantage of EPIDEMIC routing protocol is that it is good for entirely random nodes encounters. Disadvantages are consuming resources' buffers as we pass the data to all nodes. Even when the mes-

sage reaches the destination, some nodes continue passing unneeded messages which exhaust resources, since epidemic routing will flood the network with many copies of the sent bundle, and these bundles will remain even after the bundle reaches its destination. This will reduce the free space left in the buffers and may help to skater new incoming bundles. Thus, two solutions are used to solve this problem:

1. **Timer:** where each bundle is combined with a timer, when the time is up the bundle will be dropped. This method should be used carefully as if the timer is too long; it will keep the buffers busy for an unwanted longer time. On the other hand, if the timer is too short, then the bundles may be dropped even before they encounter a potential node that may make them able to reach the destination.
2. **Vaccine:** As the EPIDEMIC spreads like a disease; this method terminology comes from "Infection and Vaccination" so it is called "Vaccination." When the bundle reaches the destination, the destination will create anti-packet. Furthermore, the last node that delivers that bundle to its target will create an anti-packet and circulates it back through the network using the conventional epidemic routing. When anti-packet hits every node that is still carrying a copy of the original bundle sent in its buffer, the node will drop this copy. The process continues until all nodes in the network are reached, hence it will make sure that all the left-over copies are deleted [15].

#### 2.1.4 ProPHET

ProPHET [16] Is a DTN routing protocol for systems that are having nodes expected communications encounters. It calculates the probability of each Mule to reach the destination, and the routing is done according to the predictability using the probability of reaching the destination. The known pattern information will help to have better routing. Each node calculates a probabilistic metric (Delivery Predictability) for each destination possible. The advantages of ProPHET are the less message exchange, less communication delay, and higher delivery rates compared to EPIDEMIC; however, more memory size is needed to be able to store the Delivery Predictability Parameter. In this protocol some bundles might be dropped, that is due to the First-in-First-Out (FIFO) policy, that is when some bundles are directed to a concentrated node.

#### 2.1.5 Spray And Wait

Spray and Wait (SnW) [17] is an extension of Epidemic protocol. The reason for creating it is to reduce the overhead of EPIDEMIC flooding and thus reduce network congestion. That is done by forwarding copies of bundles combined with some probabilities. The probability is nothing but a timer that indicates the time elapsed since the two nodes last encountered the node containing the record. Spray and Wait protocol has two phases:

1. **Spray phase:** Message is spread randomly to L relay nodes. (L copies created)
2. **Wait phase:** destination not found in phase 1? If not, then nodes wait for direct transmission. The disadvantage of this protocol is processing overhead.

### 2.1.6 Binary Spray and Wait

This protocol is called like that because each node here has  $n$  copies of bundles, and each node will give half of the copies to the next node in sight and keep the other half, that is  $n/2$  being sent to the next node and  $n/2$  stays in the same current node. The main advantage of Binary Spray and wait protocol is that it reduces the flooding of the EPIDEMIC protocol. Disadvantages of it are that it still suffers from delays and resources consumption [18].

### 2.1.7 Spray and Focus

Another spray DTN protocol is Spray and Focus (SNF) [19], this protocol has two phases:

1. **Spray Phase:** In this phase, the sending node creates a message and accompany it with  $n$  forwarding tokens. When any current node meets another node, it copies the bundle with  $n/2$  tokens, and so on. This process keeps going until the last node has only one forwarding token, that's when phase 2 starts (Focus Phase).
2. **Focus Phase:** This is the phase that counts on another criterion for deciding transmission, for example, when the phase "focuses" on Timer. The timer is a record that records the time since the last two nodes encounter each other. Each node sets a timer to reach all other possible nodes.

Advantages of this protocol are refined selection criterion where satisfied criterion can be chosen for making transmission decisions, that leads to higher delivery rates and fewer copies to be spread around the network. The disadvantage is the large resource consumption overhead.

### **2.1.8 P<sub>RO</sub>PHET+**

This DTN routing protocol differs from the traditional ProPHET routing protocol in that it does not only count on the probability which is the case in the conventional ProPHET, but it also relies on many other parameters like buffer, power, bandwidth, location, and popularity to reduce the loss. Advantages of this DTN protocol are that it has less bundle dropping than conventional ProPHET and it has a better delivery rate. Disadvantages are that it has big calculations overhead at each node and if at any time any node did not cooperate, then that will cause significant problems for the sender.

### **2.1.9 MaxProb**

This DTN protocol assumes no prior knowledge about the network; rather it uses its own local information to select the next best hop. MaxProb [20] has three main components:

1. Estimating Delivery Likelihood, and it does the following:
  - (a) Draws a direct graph of the network of all nodes which are connected by edges.

- (b) Uses Dijkstra algorithm to find the shortest paths at any given time.
- 2. Complementary Mechanism: it describes the priority of which messages to be exchanged first when nodes encounter each other.

The advantages of MaxProb DTN routing protocol are: first, using Dijkstra ensures the shortest paths and least delivery latency, and second buffer management ensures lower dropping rates. While disadvantages are: first, the table exchange overhead causes a decrease in the effective time for message exchange, and second, this protocol is not suited for sparse networks, as there will be no proper graph to draw.

### **2.1.10 Contact Graph Routing**

Contact Graph Routing (CGR) is a time-varying algorithm that counts on deterministic timed contact periods in DTNs to find best routes. Despite great previous knowledge of nodes communication plan using historical facts, CGR needs heavy calculations to be able to use this knowledge in finding the best routes. CGR has many advantages that no other routing protocol have, some of these advantages are.

1. Previous network knowledge: CGR has prior knowledge about future topologies, as technically most of the nodes will exist, yet the change in the topology is due to lose the sight of contact or periodic power cycle.
2. Ahead of time readiness: Another significant advantage that other routing solutions lack is that the routing data are propagated way ahead of time before the need to use them rises. In DTNs topology changes are slower than the speed

of data spreading around them, that helps to propagate a considerably accurate time-based routing data [21].

## 2.2 Additional Literature

Abdelkader et al. [4] have contributed with a comparison survey among the significant heuristic routing protocols: EPIDEMIC, Spray-and-Wait, ProPHET, and MAXPROP and compared them to optimal routing. This work has done five experiments and the methodology used was a simulator that is built in MATLAB to simulate DTN. Inputs are starting times and duration of node contacts. To compare them to optimal routing, the optimal routing has been performed using open source mixed integer linear programming package, EPSOLVE. For optimal routing, two objective functions: Minimize The Total Number of Hops (MINH) and Minimize The Total End-to-End Delay (MIND). Metrics used: Delivery Ration (DR), Packet Delivery Cost (DVC), and Average Packet Delay (Del). Parameters used: buffer capacity, Time to Live (TTL), node density (changing number of nodes in a network), and traffic load (changing the number of generation rate) data taken from real life cars and pedestrians in Helsinki downtown. Results of the work show that first metric Delivery Ration (best to worst) are MIND, MINH, MaxProp, ProPHET, SnW, and then comes EPIDEMIC. Second metric Delivery Cost (lowest to highest): MINH, MIND, ProPHET, SnW, MaxProp, and highest is EPIDEMIC. Third, average delay (lowest to highest): MIND, MaxProp, SnW, EPIDEMIC, ProPHET, and MINH. The results came out logical except for average delay metric, MINH gave unpredicted delay (it is one of the optimal protocols). Although



this paper did a good job discussing DTN routing protocols, it did not consider varying the number of nodes on the two optimal protocols (MINH ad MIND).

Araniti et al. [21] addresses solving DTN routing problems by presenting enhancements to CGR. The importance of this work is related to its addressing of planned travel of spaceships which covers the lost communication and delay in many places of the trip. This research proposes a CGR for such time-varying topology; it uses it to make a list of routing plans for each stage of the space trip. The contact plan contains routing plans related to each period of connection change (T1 and T2), and also contains nominal transmission rates (R). Routing table plans built for each stage of the trip using Dijkstra [22] Algorithm. When it comes to accuracy, CGR could be compared to IP.

This work provides two modifications for CGR; first one is (Short Term Evolution) when contact happens, a problem occurs if the neighboring node does not already have a load in the buffer, thus delay for the transmission takes place. Due to the priority of having communications done within (T2-T1), CGR-Earliest Transmission Opportunity (ETO) is proposed [23]. (T2 and T1) to be replaced with ETO contact parameter during contact graph traversals. The problem of estimates by other nodes solved by using Contact Plan Update Protocol (CPUP). The second modification is Long Term Evolution, that is by using Path Encoding CGR Extension. It takes the calculated path and attaches it to the message; thus, nodes do not need to do complex path calculations. Paper also suggest the using of CGR extensions not only for deterministic scenarios but also for opportunistic ones. Paper describes four experiments over space networks Deep Impact

Network Experiment (DINET) using Interplanetary Overlay Network (ION) software on spacecraft EPOXI) took four weeks of flight testing DTN, another experiment of JAXA DRTS Testing in co-operation with National Aeronautics and Space Administration (NASA) to evaluate DTN and CGR (performance of Bundle Protocol (BP), Licklider Transmission Protocol (LTP), and CGR was tested) through several network topologies. The third Experiment was (Space Data Routers) to improve the dissemination of space mission data concerning volume, time-lines, and continuity. The fourth Experiment was the Pilot Operation of the ION implementation on the international space station.

The work conducted by Dudukovich et al. in [24] focus on developing a ML algorithm for DTN routing. This work tried to merge two cognitive learning methods, the reinforcement learning and Bayesian learning to come up with a new solution that still holds the advantages of CGR and have more adaptability to DTN changing topologies. This effort is different from others in its methodology, as it used information taken from lower level protocol layers to make routing decisions depending on connections reliability criteria. Simulation using Objective Modular Network Testbed in C++ (OMNET++) have been implemented using a nine node mesh and each simulation last 2.7 hours. The Q-routing reacted to the network load as expected, and as the network load increased Q-Routing outperformed the shortest path algorithm.

Kurg et al. [25] discussed Mobile Ad hoc Networks (MANET) and they explain that MANETs are the best options to support first responders after disasters in disastrous areas. They also explained that MANETs would work well within each group of first

responders, yet they will suffer from intermittent connectivity among separated groups. The authors used Opportunistic Network Environment (ONE) mentioned in [26] and fed it realistic disaster scenarios data on both first responders movement mentioned in [27] and traffic characteristics as cited in [5]. Kurg et al. [25] considered Movement Characteristics, Traffic Characteristics as messages will spread among groups each with different source-destination, Overhearing Capabilities as messages will flood among groups trying to reach their destinations, Relay Types as it is important to avoid delays as much as possible, and Reliability as in disasters it is crucial to maximizing the delivery ratio. The work suggested adding Unmanned Ariel Vehicles (UAV) that are moving according to a moving plan over the on-ground disaster recovery nodes (devices connected to first responders vehicles). Work chose ProPHET, MaxProp, and Rapid protocols in the simulation as they are the only protocols that meet most of the needed characteristics which are movement, overhearing, and reliability characteristics. Results showed enhancement of delivery delay by reducing it by 40% of the same situation without the UAVs.

# Chapter 3

## Proposed Approach

### 3.1 Proposed Approach Background

Recently we hear more than ever about Artificial intelligence (AI), ML, artificial cognition, and deep learning. These terms are similar in how they sound, but they are different in other ways. This chapter will discuss smart systems in general, and that includes, AI, ML, and ANN which will be used later in the thesis work.

#### 3.1.1 Artificial Intelligence

Artificial Intelligence is the technological science that enables machines to do intelligent tasks similar to the tasks done by human beings.

### 3.1.2 Machine Learning

Machine learning is a sub-type of artificial intelligence which counts on algorithms to learn and then predict, machine learning uses database statistics. The main reasons for using machine learning are:

1. **Solve a problem:** problems that machine learning can solve are many. Maybe a business problem, marketing, profit, reduce time or produce more.
2. **Technical achievement:** many companies compete through achievements they make by using machine learning. Like the recent breakthrough that Google achieved by beating the world's champion in Go game.

This work has both, the motivations, to solve the problems that DTN routing faces, as well as to create a technical achievement in that field. Coding is reaching its limit to implement bigger scientific goals; autonomous solutions are needed.

### 3.1.3 Deep Learning

Deep learning is a sub-type of artificial intelligence that counts on algorithms to learn and then predict and uses a lot of data.

### 3.1.4 Artificial Neural Networks

A human brain is an advanced machine, for example, let's take figure 3.1, many people can distinguish effortlessly that the image is showing the number (504192). Brain's

powerful capability to comprehend the images is due to the 140 billion neurons that it consists of. How about when we want to find a handmade machine that can do such tasks? That is when ANNs come in.

Generally speaking, an ANN is a machine that is designed to model the way in which the brain performs a particular task or function of interest [28]. Neural networks are the artificial copy of the human nervous system where artificial neurons are used to mimic human brain operations. Artificial neural networks use mathematical formulas to perform operations on inputs with the goal of giving the desired output. In general what ANNs do is that they take the input data, apply the designed calculations on them and provide an output. These networks keep repeating the calculations process to learn more and get closer to the desired result, by reducing the error between the produced output and the desired one, this process is called "Training."

While science is trying to mimic the human brain into software or hardware neural network, the knowledge and learning experience could go backward too: "Complete understanding of the brain could be achieved only when one successfully models neural networks and systems so as to reproduce entire aspects of brain functions" [29].

### **Artificial Neural Networks advantages**

ANNs have a great problem-solving capability due to two reasons, first their parallel distributed structure, and second their ability to learn and therefore generalize. Generalization is the ability of a neural network to give a reasonable output for an input data

that has similar features to what the neural network had been trained on before. It is important to realize that neural networks have many advantages such as:

1. **Nonlinearity:** due to the structural shape of ANNs, they are nonlinear themselves. Furthermore, this nonlinearity gives neural networks the ability to adapt for more complex problem-solving.
2. **Input-Output mapping:** Neural networks have a vital feature, that is their ability to learn a specific pattern of inputs as well as a related output pattern or (samples) which represent the desired output for each information set fed to the input layer, in other words mapping each input set to an output set, this is called supervised training.
3. **Adaptivity:** this advantage is the soul of ANNs. Neural networks can adapt their synaptic weights with the change of the environment they are put in, that is by changing their weights according to the data fed to them.
4. **Fault Tolerance:** When it comes to implementing neural networks using hardware, they are fault tolerant. Neural networks can keep giving relatively correct output to some extent not only when neurons themselves get damages but also when links get damaged too. In like manner, a network will result in abnormal output only when the damage reaches substantial magnitude.
5. **Uniformity of Analysis and Design:** ANNs have a universal method of processing the information with small adaption here and there. This feature makes it easier to understand, design, and implement neural networks to solve problems and find solutions.

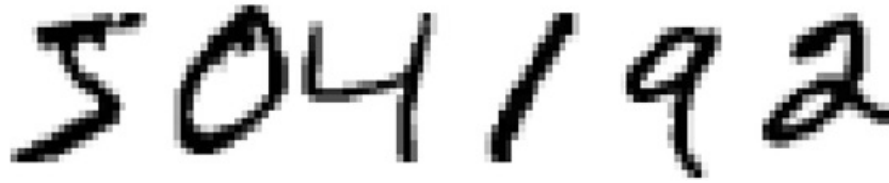
A handwritten number '504192' in black ink on a white background. The digits are slightly irregular and slanted, typical of a person's handwriting.

Figure 3.1: Example of a handwritten numbers to be recognized by artificial neural networks

courtesy of <http://neuralnetworksanddeeplearning.com>

6. **Neurobiological Analogy:** Neural networks principle is the human brain, and its goal is to achieve whatever a human brain can. Moreover, the human brain is a living proof that learning and parallel processing is an excellent strength to possess and use to find answers for unsolved problems [30].

### Perceptron

A Perceptron is simply a single neuron. The Perceptron is the smallest unit in any ANN; it calculates the output by multiplying the inputs with the weights and adding the biases to the result. Equation 3.1 shows how a perceptron calculates its output and figure 3.2 shows its structure.

$$f = \sum_{i=1}^n x_i w_i + b \quad (3.1)$$

where:

$x_i$  is the input

$w_i$  is the weight



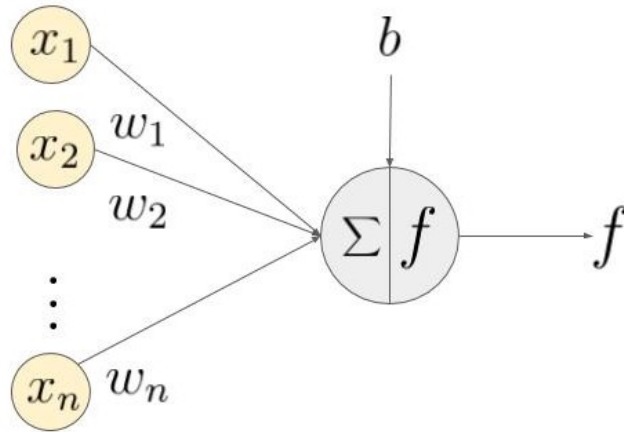


Figure 3.2: Perceptron Components

courtesy of <https://www.learnopencv.com>

$n$  is the number of inputs

$b$  is the bias

## Neural Networks Architectures

Despite the similarity of the basic principle of how artificial neural networks work, they differ in many ways. Artificial neural networks could be categorized according to the method they learn, depth or number of layers, connections number, connections endpoints, and many other distinctive factors. Major types of ANNs will be discussed here. Neural networks could be categorized depending on the way neurons are connected within.

### 1. Feed-Forward Networks

This type of networks consists of layers of neurons having connections to the

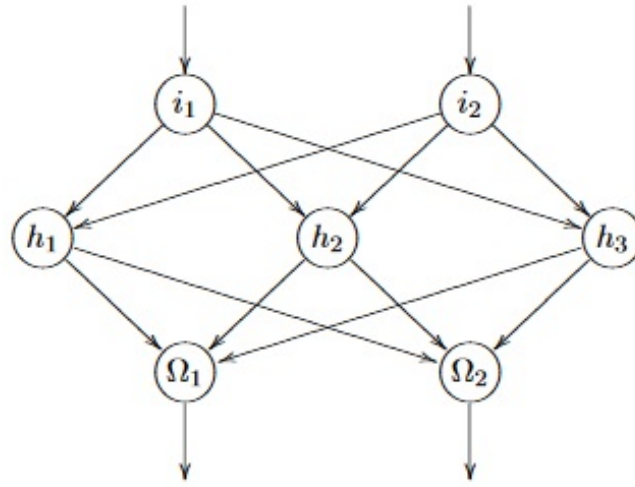


Figure 3.3: Simple feed-forward neural network example  
courtesy of [30]

following layer. Besides, the first layer that receives the inputs to the network is called "Input Layer," the final layer where the output is received is called "Output layer," and in the case of having other layers that reside between the input and output layers these layers are called "Hidden Layers." Similarly, a feed-forward neural network that has one input layer, one hidden layer, and an output layer is called "Shallow Feed-Forward ANN." Additionally, a network that has more than one hidden layer is qualified to be called "Deep Feed-Forward Neural Network."

Figure 3.3 shows the layout of a feed-forward neural network [30] and figure 3.4 shows a dense deep feed-forward neural network.

## 2. Recurrent Neural Networks

Recurrent Neural Network (RNN) is the type of ANNs that nodes have a con-

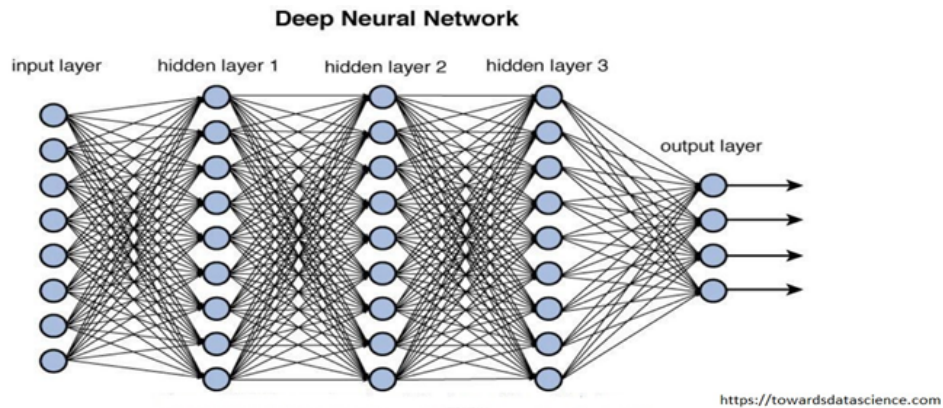


Figure 3.4: Example of a deep feed-forward neural network

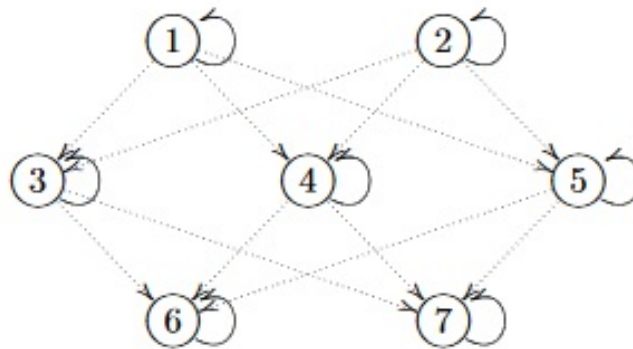


Figure 3.5: Simple example of recurrent neural network  
courtesy of [30]

nection to themselves. As we notice in figure 3.5 the recurrence is denoted by looped links going out and coming back into the network nodes [30].

### 3. Fully-Connected Neural Networks

They are the ANNs that have each node connected to every other node in the entire network, in other words, there is a direct link between every two nodes in the network, noting that there is no recurrent link in this type of networks as can be seen in figure 3.6 [30].

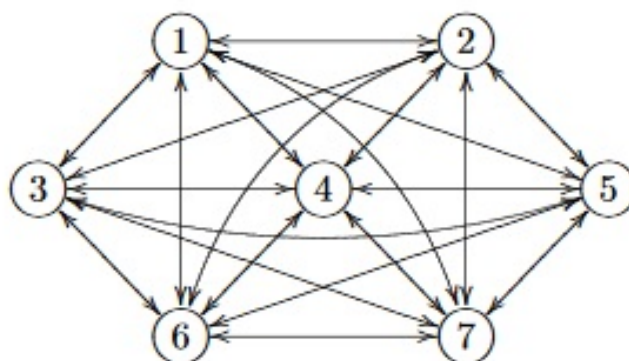


Figure 3.6: Simple recurrent neural network example  
courtesy of [30]

## Activation Functions

### 1. ReLU Activation Function

A Rectified Linear Unit (ReLU) is an activation function that gives an output of zero for all input values less than or equal to zero and gives an output equal to the input itself everywhere else. Figure 3.7 shows ReLU function behavior.

### 2. hyperbolic Tangent (Tanh) Activation Function

A Tanh is an activation function that gives an S shape output around the x-axis, passes through the origin, and has y-axis values range of (1.0,-1.0). Figure 3.8 shows Tanh activation function behavior.

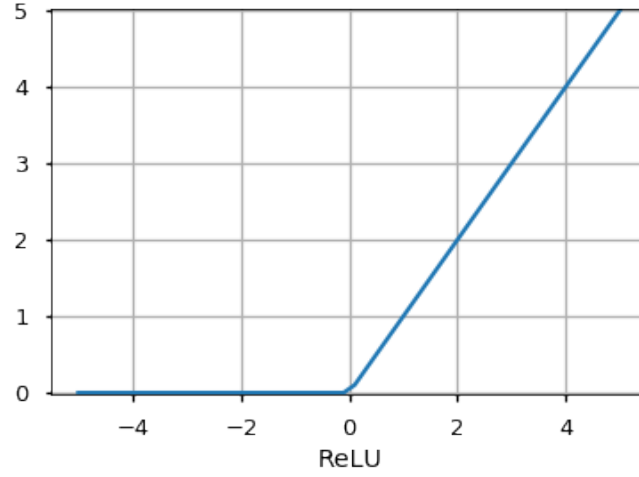


Figure 3.7: ReLU activation function graph  
courtesy of <https://www.learnopencv.com>

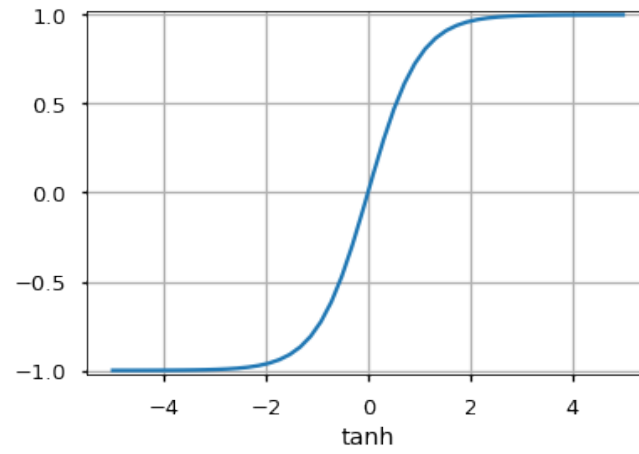


Figure 3.8: Tanh activation function graph  
courtesy of <https://www.learnopencv.com>

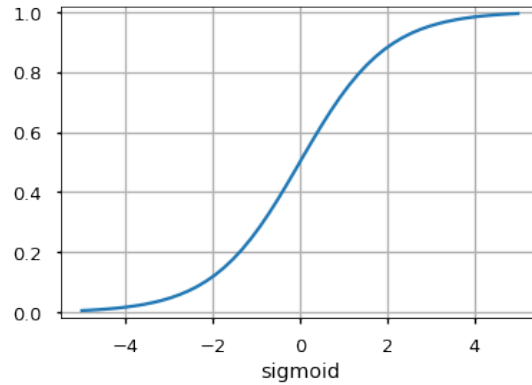


Figure 3.9: Sigmoid activation function graph  
courtesy of <https://www.learnopencv.com>

### 3. Sigmoid Activation Function

A sigmoid is an activation function that gives an S shape output around the x-axis and has y-axis values range of (0,1.0).. Figure 3.9 shows sigmoid function behavior.

### 4. SoftMax Activation Function

A SoftMax is an activation function that regulates the inputs into percentages or fractions of 1, in other words, SoftMax will weigh the inputs and transform their values into fractions, all these fractions add up to 1. SoftMax is useful for classification problems. Figure 3.10 shows a simple explanation of SoftMax activation function.



Figure 3.10: SoftMax activation function graph

courtesy of <https://github.com/>

## Artificial Neural Networks Training Types

Human beings learn by observing their surroundings or by receiving a feedback from them. Likewise, ANNs could learn under supervision or by themselves [31]; thus there are two training methods.

### 1. Supervised Training:

It is the training method in which two sets of data are presented to the ANN, one of them represents the input and the other represents the mapping output or as called "labels". Learning through supervised training is also called "error-back propagation." The correction-learning algorithm trains the network based on the input-output samples and calculates the error, then it uses the error to adjust back the weights and so on. The most common metric to measure the error while training ANNs is the Mean Square Error (MSE), MSE equation could be seen in equation 3.2 [32].

### 2. Unsupervised Training:

Self-Organizing ANNs learn using unsupervised learning algorithms to identify hidden patterns in unlabelled input data. This unsupervised refers to the ability to learn and organize information without providing an error metric to evaluate the potential solution [33].

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

Figure 3.11: Types of training and their possible applications

Courtesy of <https://towardsdatascience.com>

Generally speaking, supervised learning with discrete data could be used for classification or categorization, while with continuous data it gives regression. On the other hand, unsupervised learning with discrete data sets does clustering, and with continuous data, it makes dimensionality reduction. It is good to mention that dimensionality reduction is a feature selection that can be a profitable tool to improve the performance of unsupervised learning [34]. Figure 3.11 briefs the different uses of learning per data type.

$$MSE = 1/N \sum_{i=1}^n x_i (d_i - y_i)^2 = 1/N \sum_{i=1}^N e_i^2 \quad (3.2)$$



```
K.mean(K.equal(y_true, K.round(y_pred)))
```

Figure 3.12: Code line for calculating binary accuracy

courtesy of [www.stackexchange.com](http://www.stackexchange.com)

$N$	= number of total training samples
The input of the $i$ th sample	= $n$ dimensions vector $x_i$
Its expected output	= $m$ dimensions vector $d_i$
The corresponding output of the neural network	= $y_i$
$e_i$	= the sample error

### Artificial Neural Networks Accuracy and Loss

Accuracy and loss are methods of measuring ANNs performance. In this research, Binary Accuracy will be used. Binary accuracy calculates the mean accuracy rate across all the predictions for binary classification problems. The binary accuracy is calculated merely by comparing the rounded output of the trained ANN with the expected output, and then take the mean of that.

The code in figure 3.12 shows how binary accuracy is calculated. Another important metric in the ANN training of the proposed solution is the loss function, the loss is simply a measurement of the training performance, in other words, it is a reading of how well the ANN is reaching the desired outputs. Equation 3.3 shows how loss is calculated [31].

$$L(d,y) = |d - y| \quad (3.3)$$

where:

$L$  is the Loss

$d$  is the desired response

$y$  is the neural network output

## 3.2 Proposed Solution

This section is discussing the solution proposed in this thesis work. The proposed solution here is just a layout of an early design with as much as possible added details at this stage of thesis work. Later in chapter 5, the particulars of this design will be described in further information after running all the tuning and modification experiments.

### Roadmap to Achieve This Thesis Work

The roadmap of the proposed solution describes the steps of thinking about finding the best solution and then designing and building the final solution of this thesis work.

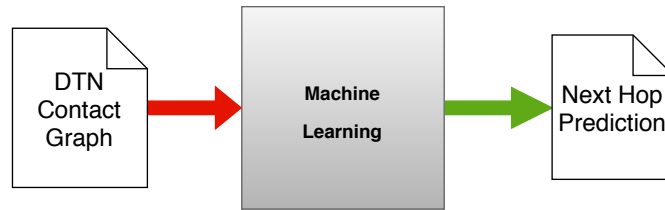


Figure 3.13: First step in creating the proposed solution by using Machine Learning Model, the model can predict next hops when fed a new contact graph dataset.

### First Step

The first step of the proposed solution roadmap is, in general, to build a machine learning model that can predict the right next hop when the related contact graph data entry is plugged into it. This thesis work is aiming to use ML due to its great benefits discussed before in this chapter, as it will help to avoid the shortcomings of the traditional space DTN routing protocols. Figure 3.13 explains the general goal of this thesis work (Step One of the Proposed Solution).

### Second Step

The second step of designing the solution is creating a Space DTN Contact Graph dataset that represents three space DTN nodes. The dataset created consists of six-inputs which are mapped into two outputs. The six inputs are divided into three pairs; each pair is representing the start and end time of the availability of space DTN link. Depending on the three pairs of inputs, the two mapped outputs serve as an indication of which next hop to take. The dataset is generated by creating an algorithm code using (Python). The dataset consists of 600,000 data entries, and each has six inputs that are mapped to two outputs. Another data set that consists of 10,000 data entries is also

created to be used in the solution tuning experiments in chapter 5, and a third data set that consists of 60,000 data entries will be used to test the final proposed solution to measure its prediction accuracy and performance.

### **Third Step**

Earlier in this chapter, we have highlighted the significant advantages of artificial neural networks, including their adaptability and customization ability to target problems and design and tailor a solution for them. This thesis work will choose ANN, to take advantages of the beneficial characteristics they are offering. The main factor that made this work choosing ANNs is the "Fitting" ability to map specific input data-entries to match them with certain output data-entries, that could be achieved using Supervised Training (as discussed in 3.1.4).

After creating an artificial neural network, it will pass through a set of experiments to tune and customize it. The customization will target building a neural network that uses the least possible resources yet gives an acceptable value of performance. The set of experiments will include:

1. Neural network depth experiment, to be able to choose good depth for the proposed neural network.
2. Gradual neural network widening experiment, to be able to test the effect of having different widths layers.
3. Activation functions experiments, to examine the activation functions and be able

to choose a suitable activation function that can give the best performance when working on our space contact graph dataset.

4. Exponential Increase of neural network width experiment, this experiment is to test the impact of widening the whole neural network drastically and check the effect of that on the training time as well as performance.
5. The cone effect experiment, this experiment will test how much the network performance will be affected by having all layers of an equal width versus having them gradually decreasing (like a cone shape).
6. Final space DTN neural network routing solution, this is the last experiment in which all the knowledge gathered from all the previous experiments will be utilized to build a robust neural network using the least possible resources.

Figure 3.14 demonstrates the third step of the proposed solution roadmap.

#### **Fourth Step**

This is the testing step where the ANN trained model will be tested. The test will be implemented using the new contact graph dataset that the neural network has never seen before. Using that new dataset will be a fair evaluation for the proposed space DTN ANN routing solution. Figure 3.15 demonstrates the last step of the proposed solution roadmap which is testing the final ANN model. Figure 3.16 demonstrates the proposed solution of this thesis work, which is a space delay-tolerant network artificial neural network routing model.

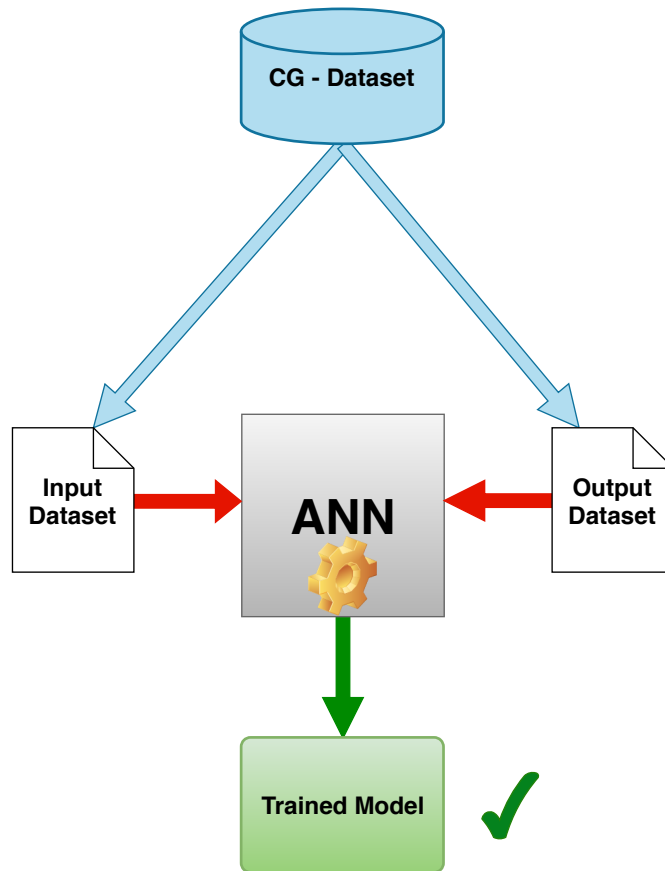


Figure 3.14: Third step in creating the proposed solution by using an ANN , it includes creating a general ANN, perform experiments on it, and ultimately create the final trained model.

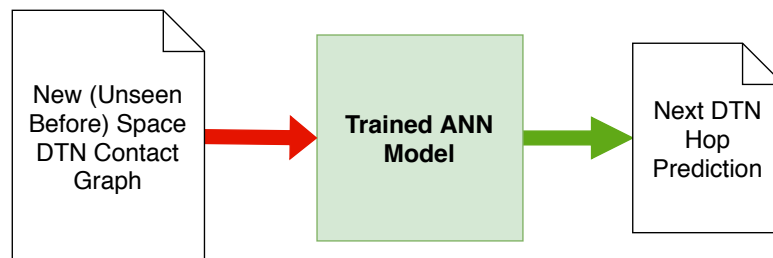


Figure 3.15: Last step in creating the proposed solution, it includes testing the ANN trained model with a new dataset

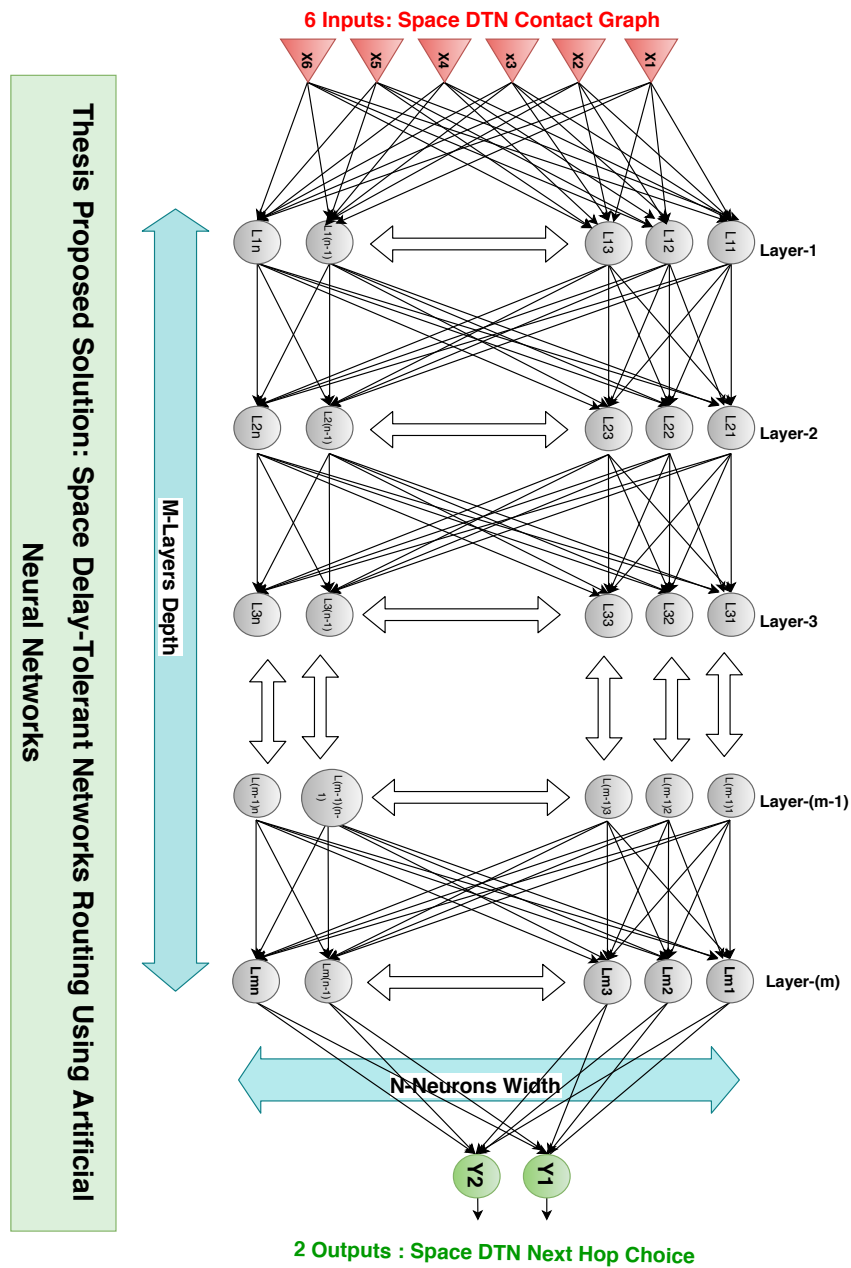


Figure 3.16: Proposed Solution: DTN Artificial Neural Network Routing. Feed-forward neural network with  $(m+1)$  layers and  $(n)$  neurons width, 6 inputs represent the space contact graph for 3 DTN nodes and 2 outputs represent the output indicating the chosen next hop

## Chapter 4

# Experimental Evaluation and Tuning

This chapter will outline of the experiments that will investigate all the variations that could be made to the initial feed-forward neural network that will be developed and enhanced later to become the final solution. Furthermore, the impact of these alterations on the neural network performance will be recorded and evaluated. These experiments are the roadmap to create a feeling and sense of changing both networks topologies and parameters. The knowledge bank that will be gained after performing all of these trials will be used as a foundation to build a neural network that is capable of handling space DTN routing CG dataset, train on it, learn from it, and gain acceptable accuracy and reduce the loss to a minimal negligible value. Later in the results chapter, a discussion of the results will take place to find the reasoning behind the specifics of the results and explain the trend of the plots that will come out of them. The following sections are the road plan for the path of experiments taken in this research searching for the most efficient space delay-tolerant networks ANN routing solution.



Throughout this whole research and in all its experiments, feed-forward neural network topologies have been used combined with the popular widely used back-propagation training algorithm [35]. Furthermore, on top of the back-propagation, Adaptive Momentum Estimation (Adam) [36] has been utilized to maximize learning efficiency and convergence. Adam is an optimizer that does not count on a fixed learning rate; it keeps changing the learning rate for maximum movement toward the optimum solution. Also, different ANN topologies have been used throughout the experiments in this research, with the intention of making the most notable results for obtaining the clearest plots and hence analysis. Besides, the training parameters have been highly tuned to magnify the output results for the best comparison support. In contrast to choosing different neural networks structures and parameters to perform the experiments, when it comes to each experiment, the settings are fixed in the beginning, and only one factor is changed at a time.

By sticking to all of these rules, the research is aiming to give the possible most accurate comparison for evaluating factors that can enhance ANNs performance. The goal is to achieve a powerful space delay-tolerant networks routing using minimal hardware. Focusing on maximizing performance while minimizing hardware is the basic requirement to comply with DTN limitations. Finally, for each experiment, the same data set will be plugged in, that is to be able to have fair and accurate comparisons.

## **4.1 Thesis Work Experiments**

The following are the experiments that will be performed in this thesis work in order to be able to evaluate and tune the final artificial neural network that will perform delay-tolerant networks routing.

### **4.1.1 Neural Networks Depth Experiment**

In this experiment, a feed-forward network will be used to show the impact of changing the number of network layers on its performance. Before performing this experiment, network width will be fixed as well as its training parameters. The first trial will begin by training the network with one layer of depth only, then two, and so on up to five layers. Accuracy and loss of training will be calculated. Then the network will be exposed to a new set of data in order to perform a validation test, the validation accuracy and loss will be recorded. Finally, the results will be discussed per number of layers as well as training and validation results per epoch .

### **4.1.2 Gradual Neural Network Widening Experiment**

In this experiment, a feed-forward network will be used to show the impact of gradually changing layers width on the neural network performance from both training and validation perspectives. Before performing this experiment, network depth will be fixed as well as its training parameters. The first trial will begin by training the network with  $n$  number of neurons width only, then gradually will widen more layers to be of a  $(10*n)$  neurons width. Accuracy and loss of training will be calculated. Then the network

will be exposed to a new set of data in order to perform a validation test, the validation accuracy and loss will be calculated. Subsequently, the results will be discussed per number of wide layers. Finally, the time of training each of the produced networks will be measured to examine if there is any time difference among the different trials.

### **4.1.3 Activation Functions Experiments**

After experimenting with the depth and width of the ANN experimental topology, it is essential to spend some time and effort exploring and testing neural networks activation functions. This group of experiments will apply training tests on the primary activation functions to demonstrate their strength which will enable us to choose the most suitable activation function for our DTN neural network routing solution. The following experiments will test Relu, Softmax, Tanh, and Sigmoid activation functions. To magnify the effect of activation functions, a neural network of five layers will be used. In every experiment, one activation function is chosen, and it is the only function that is applied throughout the whole network. This scheme will put the whole focus on one activation function at a time, and thus the result will be counting totally on it. Using this methodology will end up giving as much difference as possible among the training results.

In this group of experiments the same topology of the network will be used for all activation functions, the only difference here is to change the activation functions into ReLU, SoftMax, Tanh, and then sigmoid. The number of epochs that will be used to train all the networks has been fixed to a constant value. Furthermore, for a fair comparison among the activation functions, training parameters of learning rate and the batch

number will be kept constant throughout all the experiments. Several training processes will be performed to evaluate the results of the training accuracy, training loss, evaluation accuracy, and evaluation loss. For any specific activation function, the training and evaluation results stay almost the same only with a negligible change in values. Because of that, the next step will be to stop any randomness in the training process by fixing the seed value and stop the shuffling in the training epochs, that is to make an accurate comparison. Eliminating the randomness in all of the experiments will not change the results of both training and evaluation accuracy and loss, on the contrary, it will make all of the training trials start from the same training point. Thus, the process will be more effective. Using this approach will help a lot to do more experiments in a shorter time.

#### **4.1.4 Exponential Increase of Neural Network Width Experiment**

It is a fact that changing any dimension of any neural network will change how the network will perform, that change may not only alter the final results of training accuracy, training loss, evaluation accuracy, and evaluation loss but it may also have other effects on the training process itself. This experiment will examine the impact of changing the width of the neural network exponentially, that is by changing all the layers width at the same time. In the beginning, the experiment will start the training and evaluation steps using  $n$  neurons in each layer; the second step will be changing each layer's neurons to be  $10n$ , the third step will be  $100n$  and so on. This series of experiments will continue to see how much accuracy we are gaining while taking into consideration how complex and costly the whole topology will be. Throughout all the experimental trials, training

times will be measured and recorded to be able to analyze the effect of the exponential width increase on training times.

#### **4.1.5 The Cone Effect Experiment**

This experiment will study the difference between having an artificial neural network with similar width layers versus cone-shaped architecture. Cone shape is nothing but the input layer being the widest and the output layer the thinnest, while the medium layers gradually decrease in width as we move toward the output layer. To be able to investigate this effect before start generalizing it, an experiment will be implemented to compare a fully wide neural network with a cone-shaped network. In both training trials, the number of epochs and training parameters are fixed. Both training and validation accuracy and loss will be measured and plotted for analysis purposes.

#### **4.1.6 Final Experiment: Space DTN ANN Routing Solution Testing**

This experiment is the last experiment to be implemented; it is the result of pouring all the knowledge and lessons learned from all the previous experiments and trials. Here, the implementation will try to achieve best training and validation results with as least resources as possible. Furthermore, time will also be taken into consideration in pursuing an optimum solution. The Experiment will start by focusing on using the least number of resources, and if the results are not satisfying, then resources will be in-

creased just enough to achieve acceptable accuracy and minimal loss.

## 4.2 Testbed

The following three sections are representing the testbed that will be used to perform this thesis work. The testbed consists of three parts: Hardware, Software, and Dataset.

### 4.2.1 Hardware

For this research work, the computer that will be to do all the neural network processing has the following features:

1. Model: HP Spectre x360 - 15-bl152nr
2. Operating System: Windows 10 Home 64-bit
3. Processor: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz  
Central Processing Unit (CPU)
4. Random Access Memory (RAM): 16GB:
  - (a) Memory Slot 1: 8GB Micron 2667MHz
  - (b) Memory Slot 2: 8GB Micron 2667MHz
5. Solid State Drive (SSD): sk hynix pc300 hfs512gd9mnd-5510a
6. System Board: 827F 95.10

## 7. System Basic Input/Output System (BIOS): F.37

### 4.2.2 Software

The artificial neural network training processes performed in this thesis work will be built and developed using the following coding languages and software:

1. Python: version 3.5.4
2. Google TensorFlow: version 1.10.0
3. Keras: version 2.2.2

additional: Matlab: 9.3.0.713579 (R2017b)

### 4.2.3 Datasets

For this research, space DTN contact graph datasets will be generated. The datasets reflect how network nodes act in space; nodes have sight of contacts with each other from time to time for certain periods of time intervals. Those Datasets reflect nodes behavior inside their DTN, whether they are active (the node gets connected or disconnected due to its movement), or passive (the node is stationary, but it gets connected or disconnected due to other nodes movement), or a combination of both. We should not forget here that connections and disconnections could be caused by third-party nodes (other than the sender or the receiver) by being in the middle of the sight of contact and thus blocking nodes from seeing each other.

Neural networks learn more efficiently when they are exposed to a broader range of data in which more possible behavioral scenarios are represented, to achieve that goal 600,000 space DTN CG data entries will be produced using Python. The aforementioned vast dataset will be used to train the final solution ANN. On the other hand, for the tuning experiments, a smaller dataset of 10,000 space CG entries will be used. The reason behind using smaller dataset is that there is no need for big dataset if the goal is just experimenting to compare performances rather than training to reach high accuracy. Furthermore, smaller dataset means less time to perform experiments and hence more time available to do even more experiments.



## Chapter 5

# Experiments and Results

In order to explore how do neural networks behave in different situations, many experiments will be implemented in this thesis. The experiments will involve investigating the effect of changing the parameters that are required to build ANNs. Not only the parameters will be changed, but additional components will also be varied to check the difference in the network behavior. Activation functions are one of the elements that will not modify the topology of an artificial neural network, but they may impact its performance. On the other hand, changes like network depth and width change the topology itself, and we will investigate whether or not they will change the performance as well.

Working with artificial neural networks could not only be described as being a mathematical or experimental effort, at his point of this thesis, we can also say fine-tuning the neural network and customizing it is becoming a type of art. The following experiments will try to find a solution that is potent enough, that is by using the least possible resources, giving the best accuracy, least loss, and least training time.

## 5.1 Neural Networks Depth Experiment

This experiment has two phases:

### 1. Training Phase

To demonstrate how the number of layers impacts ANNs performance, in this experiment many training processes have been implemented. For these experiments, a data set of 10,000 DTN CG scenarios have been used. The experiment started by using only one layer and then continued by adding one more layer at a time, that resulted in networks with one, two, three, four, and five layers. Figure 5.1 shows the training accuracy change of the various depth networks per epoch number.

The figure shows that in general the more layers the network has, the higher training accuracy values it gets for the same epoch number. It is good to mention that for the one-layer neural network, the training accuracy is barely increased after five epochs, it starts at approximately 0.53283 and ends at 0.54594. Another important output trend to watch is the training accuracy of the four and five layered neural networks; it is noticeable that they both tend to gain smaller training accuracy knowledge as they move through the epochs. These smaller training accuracy steps are taking place due to the rapid training accuracy those networks get by the end of the second epoch, that gives them fewer features to learn as

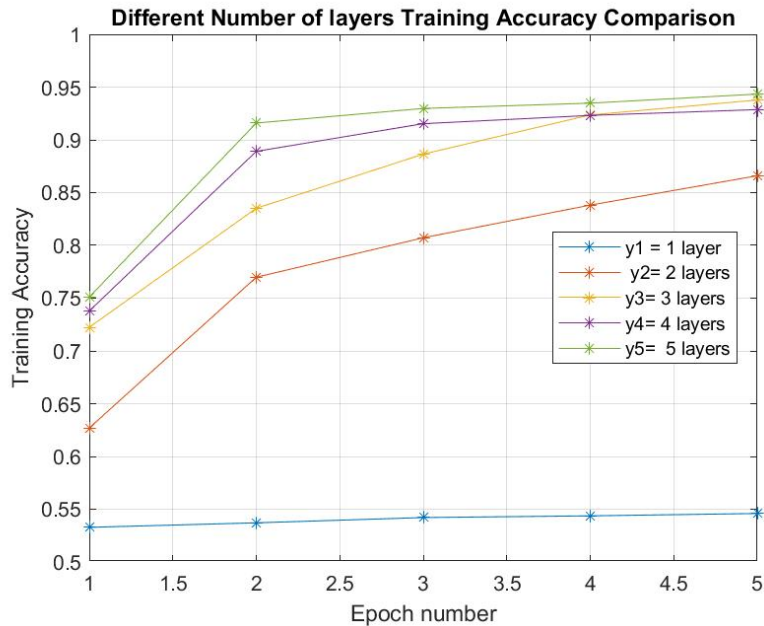


Figure 5.1: The reaction of training accuracy values to the change of number of layers in artificial neural networks per training epoch

training process keeps going. Lastly, the three-layered neural network shows the most efficient training accuracy trend, as the network keeps getting higher values of training accuracy to reach 0.93777 by the end of the fifth epoch and that is so close to the four and five-layered ANNs which reach 0.92866 and 0.94338 respectively. The optimum case could be a three-layered network as using more than three layers may cause an early over-training problem.

Figure 5.2 shows that training accuracy increases as the number of layers increases, it also shows how efficient is the three-layered neural network by giving a close result to even that one of the five-layered neural network.

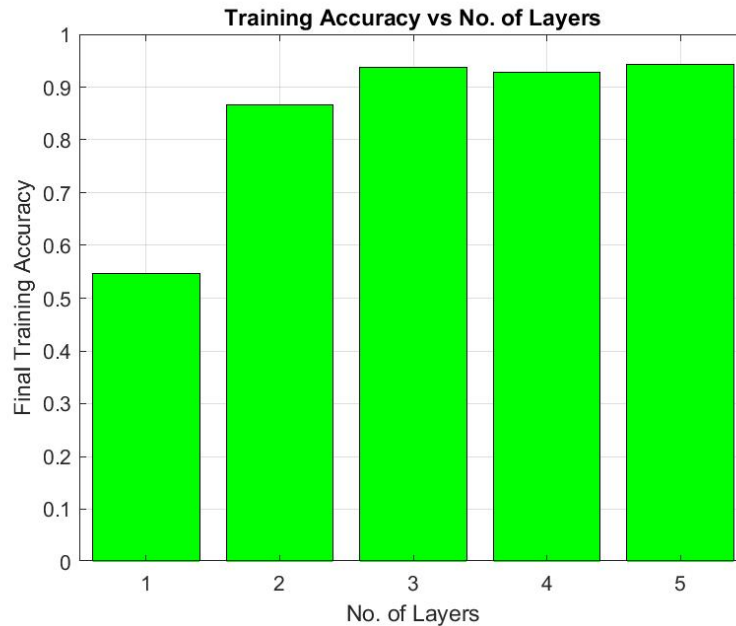


Figure 5.2: The change of the final training accuracy values for artificial neural networks with different depths

Fig 5.3 demonstrates what happens concerning training accuracy from another point of view; it shows the training loss over training epochs. For the network with one layer, the training loss shows a much smaller range of change than other networks after finishing the fifth epoch (0.466028125277 to 0.451575071025). Unlike one-layered ANN, the shift in training loss changes more rapidly for neural networks with a higher number of layers. As the five-layered neural network reveals the lowest loss value among all networks after the end of the fifth epoch. In conclusion, the three-layered neural network reflects the best source utilization, as it reaches 0.047678555284100844 training accuracy, while the five-layered reaches 0.03764490149987295 which is not so much less than the fewer neurons three-layered network.

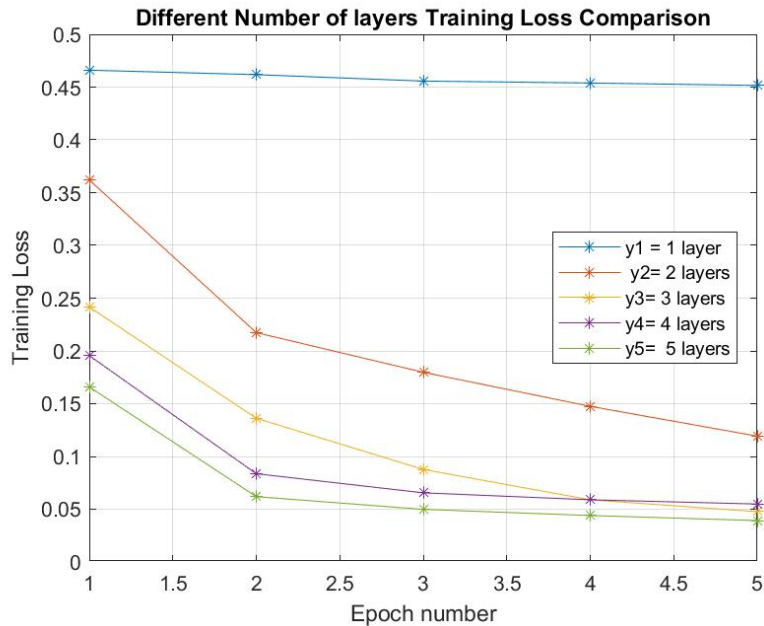


Figure 5.3: The reaction of neural networks with different depths per training epoch

Figure 5.4 displays final training loss versus the number of layers each neural network has. From the figure, it is easy to notice that probably the three-layered ANN is sufficient to achieve the goal of training without wasting more resources like the four and five-layered neural networks.

## 2. Validation Phase

When it comes to validation accuracy, it is more apparent that neural networks with a higher number of layers can recognize routing paths with higher accuracy as figure 5.5 explains.

Noticing that the behavior of one and two-layered neural networks consolidate

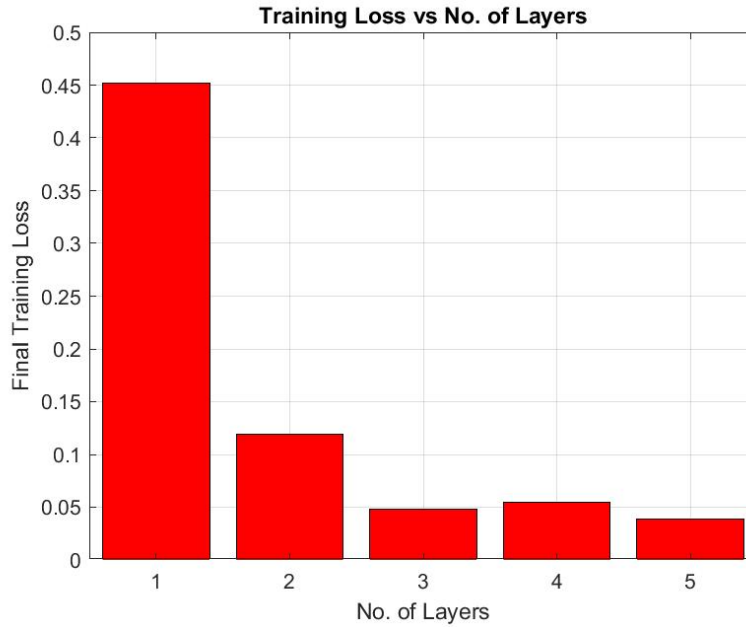


Figure 5.4: The change of the final training loss values for artificial neural networks with different depths

the previous analysis of training accuracy, that is by showing that after completing all training epochs the one-layered neural network reaches only a validation accuracy of (0.5480000089108) while it starts with (0.54150000602006), the difference is negligible. On another hand when it comes to comparing resources allocation efficiency, three-layered neural network shines, but five-layered ANN gives an amazing validation accuracy of 0.94999999493360 after finishing the fifth epoch. Four-layered network shows some hesitation with recognizing new patterns of routing CG which goes down from 0.91549999505281 in the third epoch into 0.91399999380111 in the fourth, that is 0.0015 decrease; so let's keep focusing on its behavior while discussing validation loss that comes in the next section to have a better explanation for such behavior.

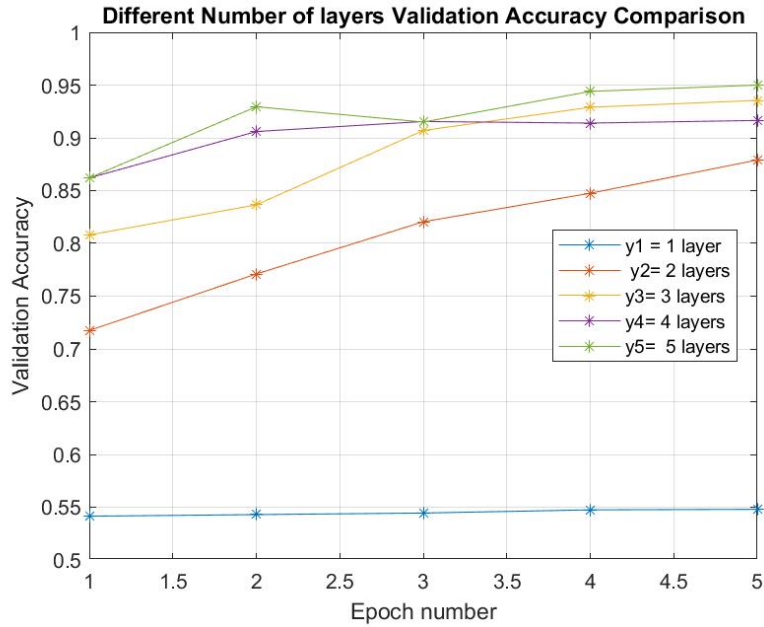


Figure 5.5: The change of validation accuracy for artificial neural networks with different depths per epoch

Figure 5.6 could be considered the most important plot among all plots in this experiment. This figure reflects easily that the more layers you have in a neural network, the better it will recognize the right DTN routing path. However, three-layered neural network and up will usually perform so good recognizing unseen routing scenarios and give the correct next-hop routing output.

As per figure 5.7, all validation losses for the neural networks are showing a good trend, that is by showing a continuous decrease in the values, except for the five-layered network, as losses get these values (0.057056315523, 0.05991196240766, and 0.04181236806894) for the second, third, and fourth epochs respectively.

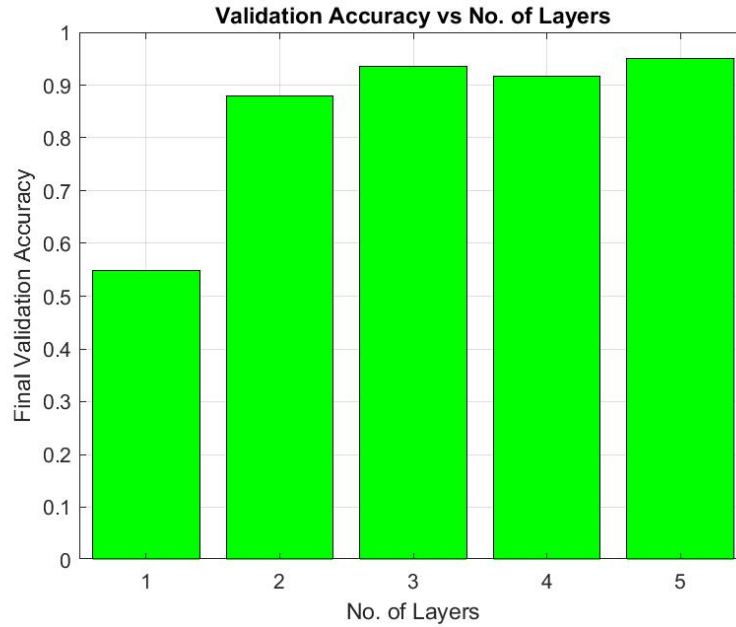


Figure 5.6: Final values of validation accuracies for neural networks with different depths

This oscillation indicates that there is a potential over-fitting issue; thus, a network of five layers under these circumstances is not necessarily useful, and the training process could do just fine with four or fewer layers. Lastly, we can say that the most efficient case is to have three layers ANN, as the four-layered neural network validation loss values do not tend to change as much as the training goes on. Figure 5.8 emphasizes showing the difference of final validation losses among networks with different depths.

## 5.2 Gradual Neural Network Widening Experiment

This experiment explores the effect of gradual widening neural network layers on its performance. The network will be trained on space contact graph data, and then the



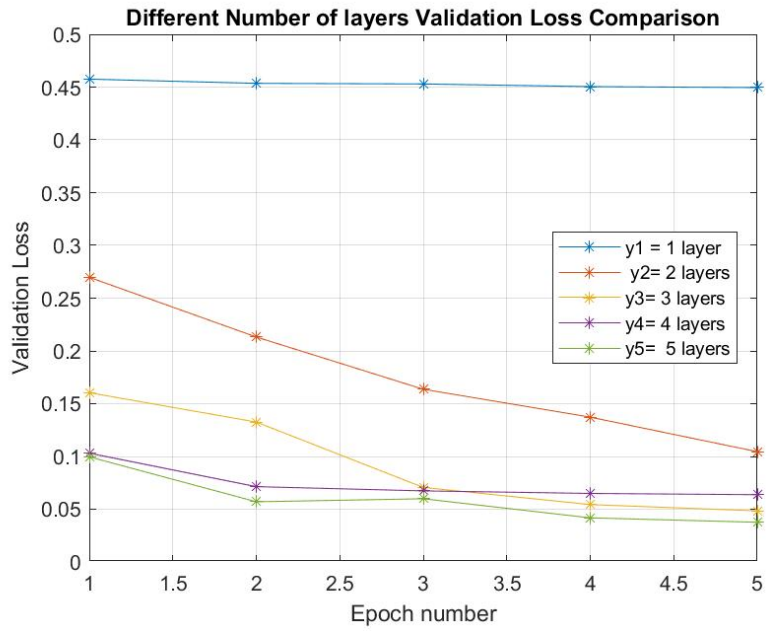


Figure 5.7: Validation loss behavior for neural networks with different depths

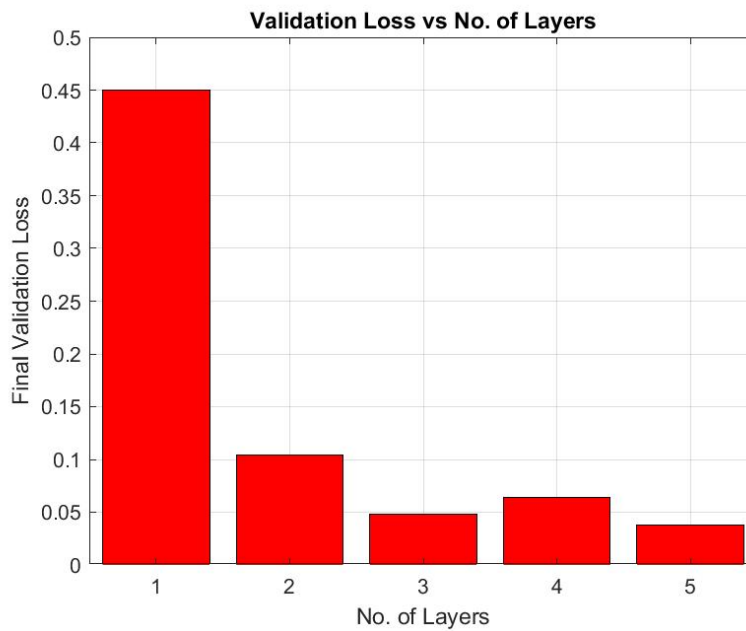


Figure 5.8: Final values of validation loss for networks with different depths

trained model will be used to perform predictions for an input dataset that it never witnessed before.

The dataset will consist of 10,000 data entries; each entry has six inputs and two mapping outputs. The inputs represent the nodes, and the outputs represent the right path choice. For this experiment, a neural network has been built with a total of five layers. The first stage of investigation starts by having all layers of two neurons width, then the network will be trained, and its performance will be measured. The following steps of the experiment include examining widening the input layer ten times the previous width and see how that affects the overall performance. Secondly, is to examine expanding the first hidden layer, and so on. This process continues by increasing the number of widened layers ten times their initial width moving all the way from the input layer down to the last hidden layer. Finally, this experimental procedure will end up by investigating the effect of widening ANN layers on total training time. The first step includes expanding the input layer of the neural network ten times its original width (ten times neurons). The second step is to widen the first hidden layer in addition to the input layer which is already widespread in the first step. The third step is to have three widespread layers and so on. For each of these steps, a complete separate training process will be done. Training accuracy, training loss, and time in which training took place for each case will be measured.

Figure 5.9 demonstrates the effect of widening neural network layers on its final training accuracy; it compares the closing training accuracy for all combinations of ANNs, that is from a fully thin network to an entirely widespread one. The figure exhibits a clear vision of an increasing trend of the final training accuracy as the number

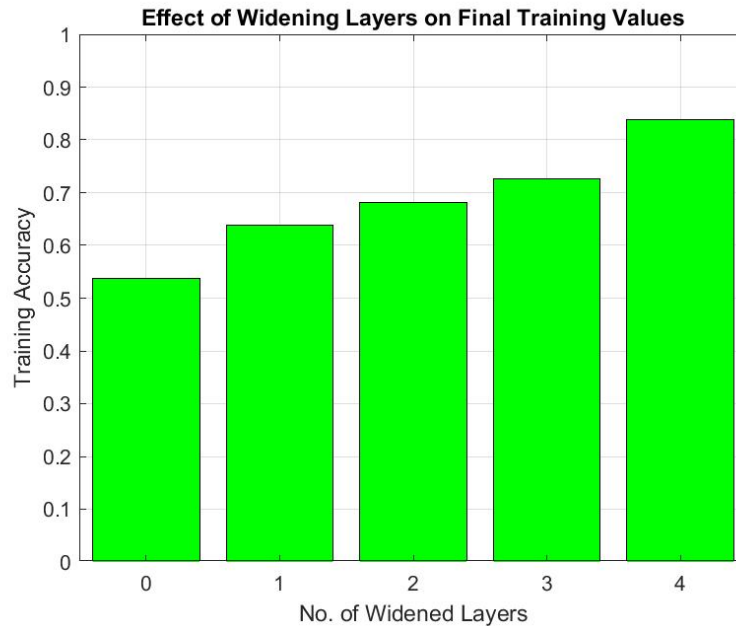


Figure 5.9: The effect of widening the layers on final training accuracy values

of thin-to-wide layers increases in the neural network topology. In other words, we can say that the more wide layers we have in a neural network, the more powerful this network will be.

Figure 5.10 consolidates the earlier understanding of the training accuracy, as it shows a descending trend in the training loss. A zero-width layer neural network ends up with high training loss value compared to an entirely widespread artificial neural network.

When it comes to measuring training elapsed time, the results are logical. Figure 5.11 displays the time consumed for training each neural network combination; an observer can easily see that the wider layers the neural network has, the more time it takes

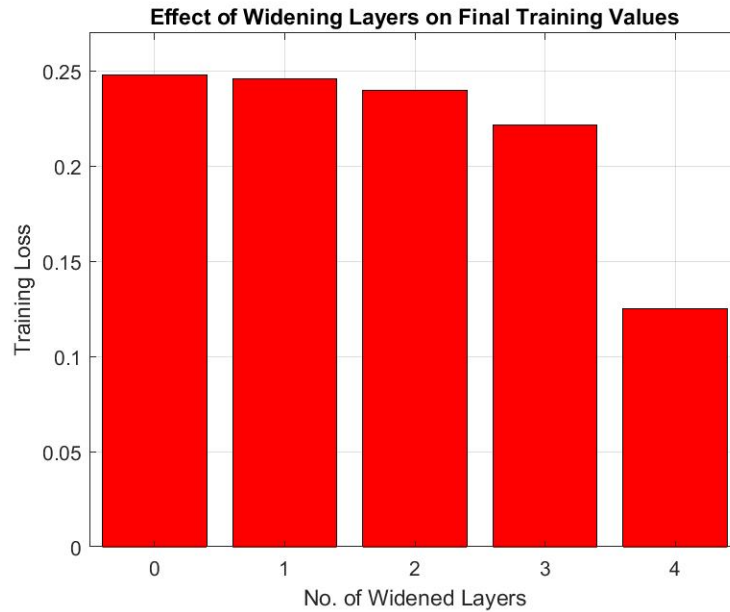


Figure 5.10: The effect of widening the neural networks layers on the final values of training loss

to train it. To explain the reason behind that, let's take any wide layer, being wide means that there are more neurons within it, and more neurons mean more processing, and in general, more processing ends up taking longer time to complete. In this experiment, the time difference between having a complete thin ANN and having a complete wide one is 1.8 seconds approximately.

### 5.3 Activation Functions Experiments

This experiment will check how the performance of the neural network will differ just by changing the activation function in all of its layers. The activation functions that will be used are Relu, SoftMax, Tanh, and Sigmoid (for more details about these functions see 3.1.4). The neural network that will be used in this experiment is a five-layered,

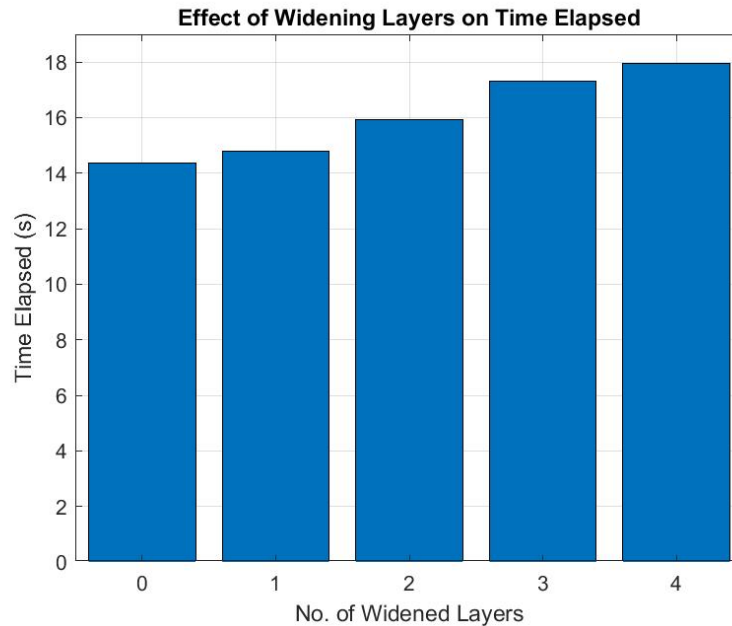


Figure 5.11: The effect of widening the neural network layers on training time elapsed dense feed-forward topology, in which each layer has two neurons with a total number of trainable parameters of 38. Input is of six values, grouped into three pairs; each pair represents a time interval beginning and end; thus input data represents three space DTN nodes CG. The output layer includes two neurons to indicate choosing one of the two next hops over the other. The dataset that will be used in this experiment consists of 10,000 samples, the sequential model will be built using TensorFlow Keras. The following sections will discuss the results of each activation function separately. The structure of the network that will be used for these experiments could be seen in figure 5.12. Learning rate that will be used for this experiments is ( $lr=0.001$ ), one-tenth of the data will be taken for the validation test, the batch size will be (10), and finally, the number of epochs will be (10) too.

```
model.summary()
```

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 2)	14
dense_2 (Dense)	(None, 2)	6
dense_3 (Dense)	(None, 2)	6
dense_4 (Dense)	(None, 2)	6
dense_5 (Dense)	(None, 2)	6

Total params: 38  
Trainable params: 38  
Non-trainable params: 0

Figure 5.12: The structure of the neural network used for activation functions experiment

### 5.3.1 Relu Activation Function Experiment

The experiment covers ReLU (discussed in 3.1.4) effect on ANNs overall performance. All the five layers of this ANN use ReLU, the reason for that is to maximize the difference among possible contradictory findings in the results. First training accuracy value started with (0.53680000007152) while training loss started with (0.46045094758272), however, after a whole ten epochs the training finished with only (0.536850000619888) and (0.45897954583168032) training accuracy and loss respectively. Comparing the values of where did the training started and where it did finish, we are looking at only Training Accuracy Gain= 5.000055e-05 and Training Loss Gain= -1.471402e-03. Which reflects an inferior performance, therefore ReLU activation function will be discarded from the solution design. Next step is to explore other activation functions. Figure 5.13 presents training accuracy and loss per epoch; it is evident that the values are barely changing throughout the training process.

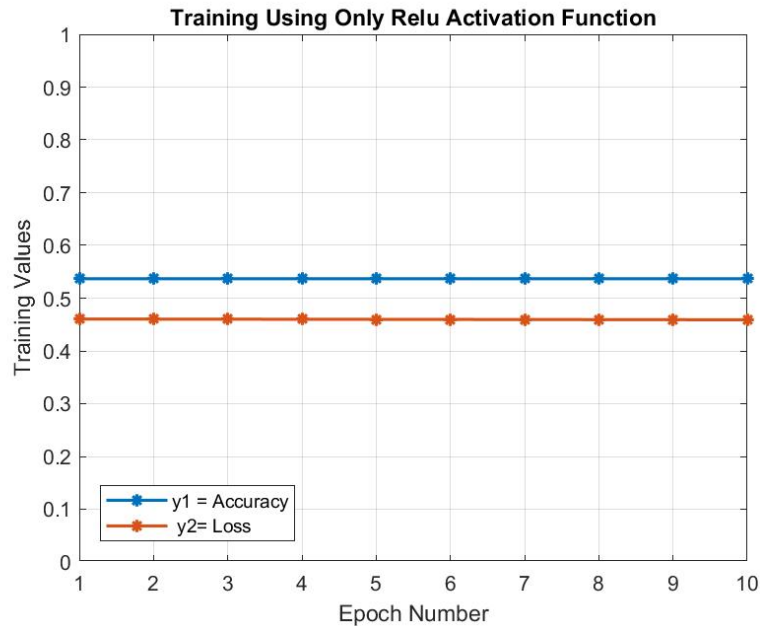


Figure 5.13: Training accuracy and loss of neural network using only ReLU activation functions

### 5.3.2 Softmax Activation Function Experiment

Here Softmax activation function (discussed in 3.1.4) will be the only activation function plugged into all the neurons of the five-layered artificial neural network. The training starts by giving a training accuracy value of 0.64805000192672013 in the first epoch to end with 0.79644999966025354 after the tenth epoch, on the other hand, training loss begins with 0.24067155095934867 and ends with 0.16154721200466157. In other words, we are looking at Training Accuracy Gain= 1.484000e-01 and Training Loss Gain= -7.912434e-02. As these final performance figures are not so good, yet they are much better than the results of using ReLU activation function, because of that, the current best option is Softmax until the subsequent experiments prove that wrong. Training accuracy and loss values per epoch could be seen in figure 5.14.

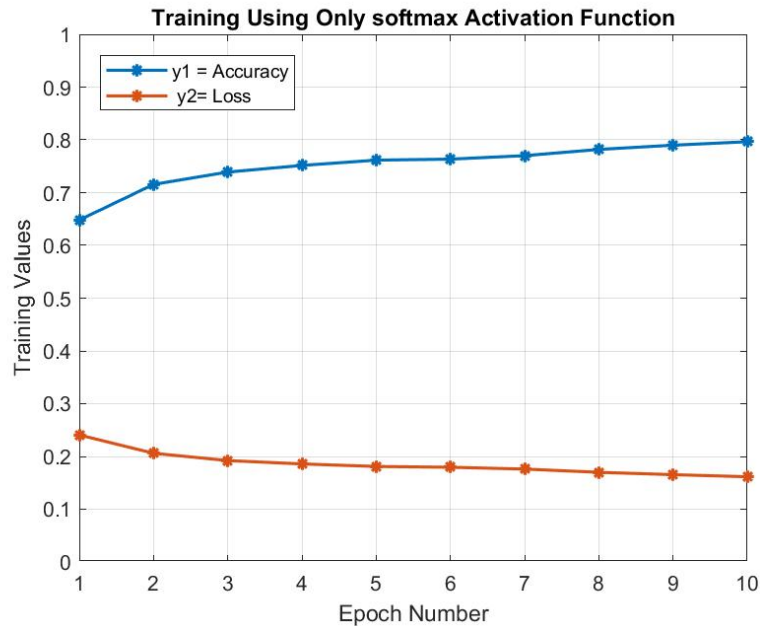


Figure 5.14: Training accuracy and loss of neural network using only SoftMax activation functions

### 5.3.3 Tanh Activation Function Experiment

Tanh (explained in 3.1.4) activation function is the focus of this experiment, the topology of the artificial neural network that will be used in this experiment is the same one that will be used for all other activation functions, with one difference, that is all the activation functions in the network here are Tanh. The training process starts by giving a training accuracy value of 0.548750006020069 which is surprisingly smaller than what Softmax starts with, yet Tanh ends the tenth epoch by giving 0.803350000113248 training accuracy which is more than final softmax training accuracy. Furthermore, the better training performance carries on into calculating the training loss. Tanh activation



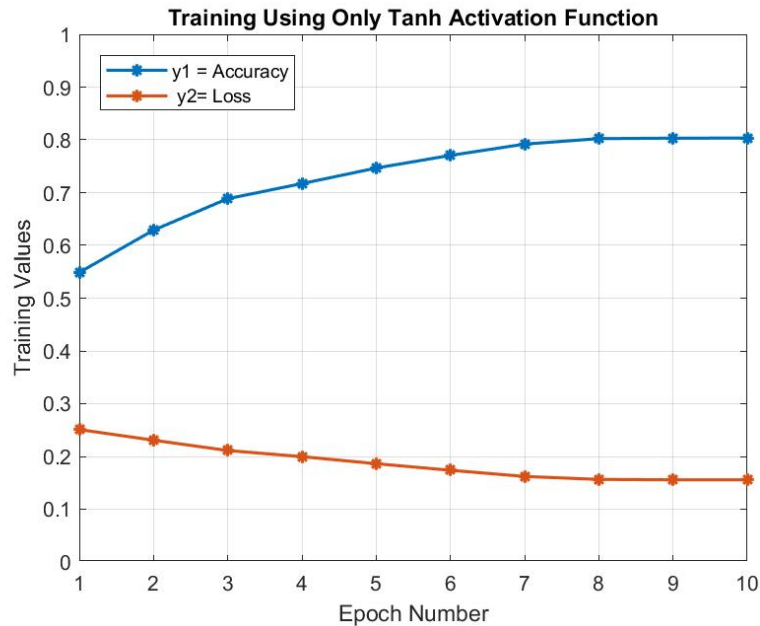


Figure 5.15: Training accuracy and loss of neural network using only Tanh activation functions

function starts with 0.25099441988766191 training loss which looks worse than softmax, yet during the ten epochs, Tanh catches up and exceeds softmax by ending the training with 0.15564228301495314 training loss. In brief, Tanh starts in a worse spot than Softmax, yet it outperforms it as epochs go by. As a result of Tanh final training readings we have Training Accuracy Gain= 2.546000e-01 and Training Loss Gain= -9.535214e-02. Figure 5.15 explains Tanh training accuracy and loss versus the epoch number.

### 5.3.4 Sigmoid Activation Function Experiment

This is final experiment exploring ANN activation functions. Here, Sigmoid (discussed in 3.1.4) activation function will be used. Training will be using the same topology used for the other activation functions experiments as well as training parameters combined with ten epochs; sigmoid started its training accuracy with a value that is even lower than Tanh 0.50480000229924915. From loss side, it also starts slightly worse than Tanh with first epoch training loss of 0.25186538897454741. Sigmoid activation function treats Tanh the same way that Tanh treated Softmax, Sigmoid starts in worse training accuracy and loss situation than Tanh, yet it outperforms it by the end of the tenth epoch 0.83929999968409541 training accuracy and 0.12802263548970222 training loss. Final gains after ten epochs are as follows: Training Accuracy Gain= 3.345000e-01 and Training Loss Gain= -1.238428e-01. Figure 5.16 illustrates the behavior of the neural network using the Sigmoid activation function per epoch. So far, Sigmoid seems to be the best activation function that could be used in the ANN that will be used to implement space DTN routing.

### 5.3.5 Final Activation Functions Performance Comparison

Figure 5.17 Compares ReLU, Softmax, Tanh, and Sigmoid activation functions training accuracy values after completing ten epochs of training. From that figure it is noticeable that ReLU has the least accuracy of 0.5368500006198883, next higher training accuracy goes to Softmax 0.79644999966025354, followed by Tanh with a slightly higher accuracy of 0.80335000011324886, and maximum tenth epoch training accu-

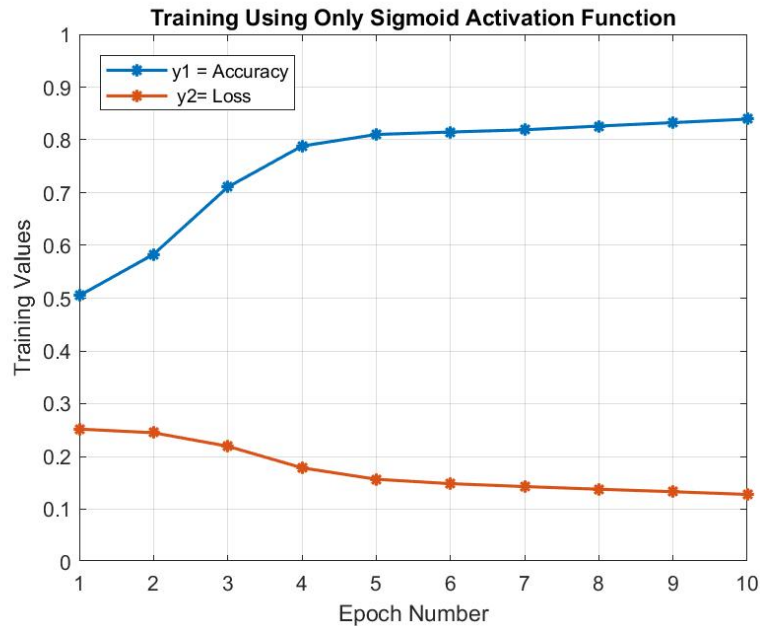


Figure 5.16: Training accuracy and loss of neural network using only Sigmoid activation functions

accuracy goes to Sigmoid 0.83929999968409541.

From loss side of view 5.18 Compares the same activation functions, by using training loss values this time. Values are after completing ten epochs of training. From that figure, it is noticeable that ReLU has greatest training loss of 0.45897954583168032, next comes Softmax with less training loss of 0.16154721200466157, followed by Tanh with a slightly less loss of 0.15564228301495314, and the least tenth epoch training loss goes to Sigmoid 0.12802263548970222.

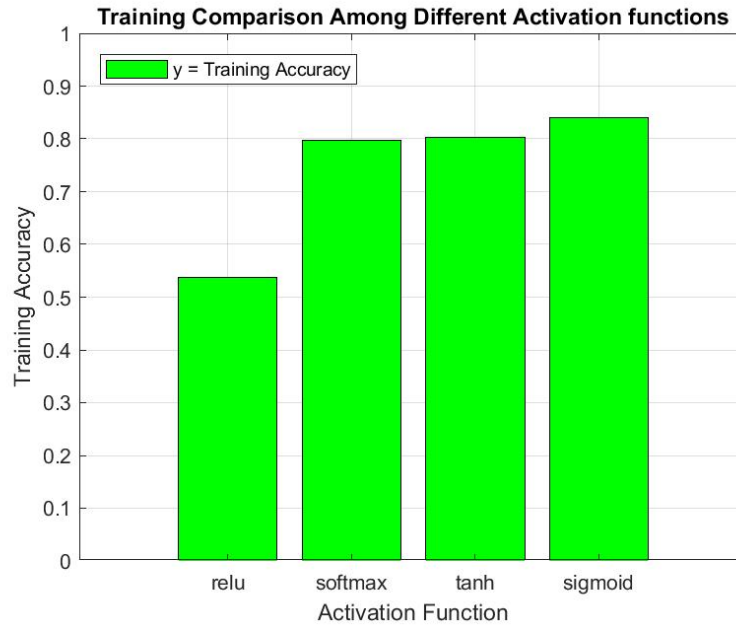


Figure 5.17: Final training accuracy comparison among neural networks with totally different activation functions

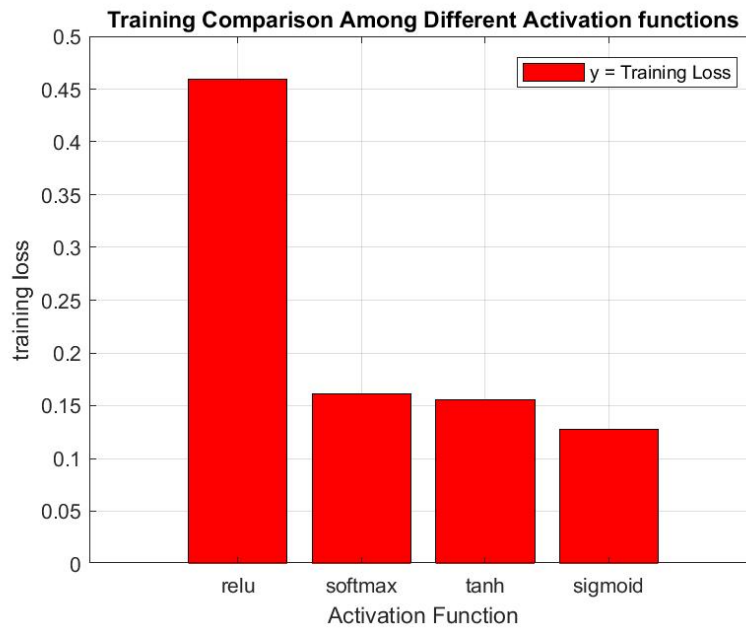


Figure 5.18: Final training loss comparison among neural networks with totally different activation functions

## 5.4 Exponential Increase of Neural Network Width Experiment

This experiment uses five layers feed-forward neural network, with the output layer having two nodes. Many topologies will be built, each with a different width, the widths will be of  $10^n$  neurons, where  $n=0,1,2,3\dots$ , the reason of that is to run many experiments while varying the width and investigate how the performance of the ANN will be. The activation function is set to ReLU for the input layer and Sigmoid for the rest.

The dataset used to train the network consists of 10,000 data entries of which 10% will be used for validation purposes; each entry has six inputs and two mapping outputs. The inputs represent the CG nodes, and the output represents the right path choice. The first part of the experiment will be the training part, that is when the ANN starts adjusting its weights and re-correct them as it gets exposed to more data entries. Increasing the neural network width and training it will continue until reaching a relatively good training accuracy and acceptable training loss. Training will consist of 5 epochs. To see this network in action, the trained neural network will be passed to the second part of the experiment. The second part of this experiment is to present a new unseen data set to the trained neural network and observe how the network will perform. The second part is used as a validation assurance for this experiment. First and second parts of this experiment will be performed on all possible width scenarios until continuing the experiment becomes not practical anymore due to either long training time or reaching an over-training point.

```
model.summary()
```

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1)	7
dense_2 (Dense)	(None, 1)	2
dense_3 (Dense)	(None, 1)	2
dense_4 (Dense)	(None, 1)	2
dense_5 (Dense)	(None, 2)	4

Total params: 17  
Trainable params: 17  
Non-trainable params: 0

Figure 5.19: The structural design of the neural network used for 10<sup>0</sup> Neurons Width experiment

### 1. 10<sup>0</sup> Neurons Width Experiment

In this experiment, the width of the neural network layers will be set to 10<sup>0</sup> except for the output layer (the fifth layer), where it will be kept to two in order to stay matching the intended CG output for DTN routing. Figure 5.19 shows the different layers of the neural network. As the figure explains, this network has 17 total trainable parameters.

- (a) **Training Phase** As expected from thin layers network, training accuracy values are not promising, at the end of the first epoch, training accuracy equals (0.46394444655213) and changes to only (0.536055561701456) after finishing the 5th epoch. That gives (Training Accuracy Gain for 1 Neuron Width= 7.211112e-02) only, that means the network is barely learning any DTN best routes. Figure 5.20 shows the values of training accuracy per epoch number.

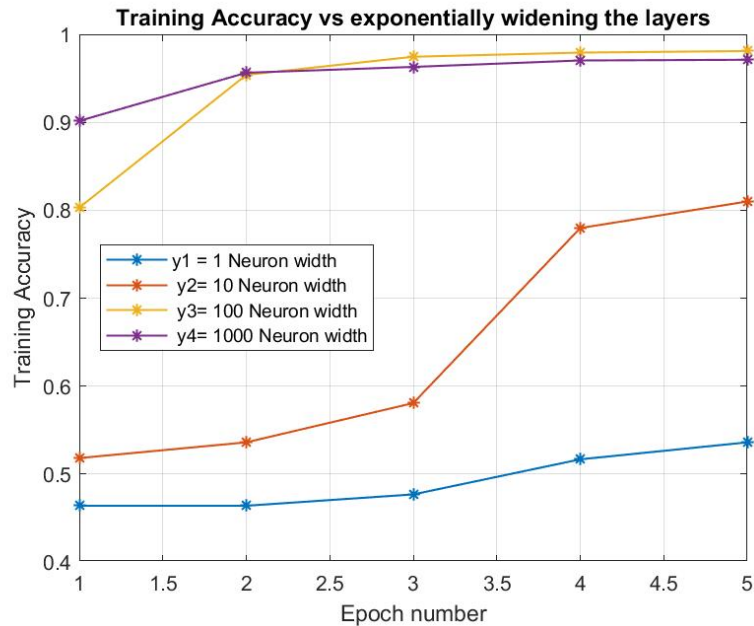


Figure 5.20: Training accuracy of neural networks with different exponentially widened widths vs number of epochs

Likewise, when it comes to the training loss, the network is struggling to reduce the loss, as it starts with (0.25753519722984897) and ends up with (0.24914302771290142), that means (Training Loss Gain for 1 Neuron Width=  $-8.392170e-03$ ) only. Figure 5.21 shows the values of training loss per epoch number.

- (b) **Validation Phase** It is evident that the impact of poor training reflects on the validation accuracy. That translates into a first epoch value of training accuracy of (0.45600000254809) and makes small changes through the five epochs to reach only (0.54400000572204). That gives (Validation Accuracy Gain for 1 Neuron Width=  $8.800000e-02$ ) only, that could be expressed as

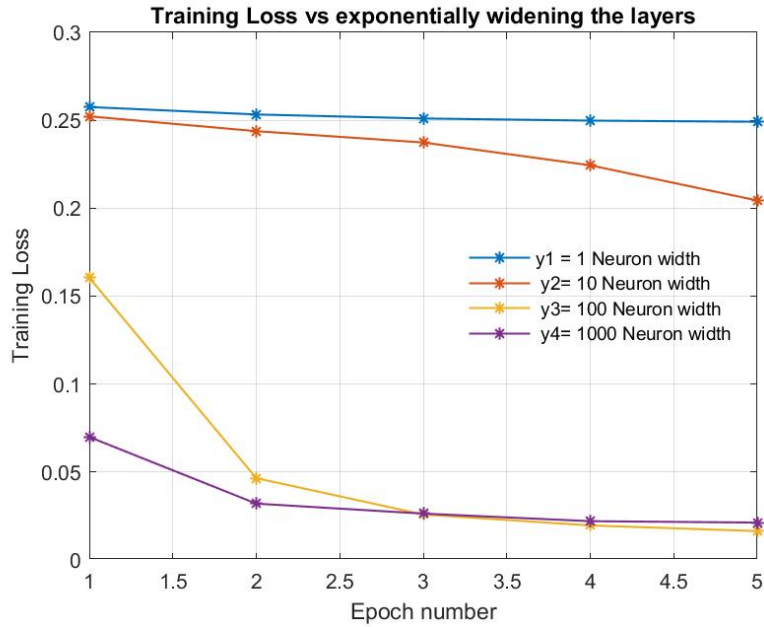


Figure 5.21: Training loss of neural networks with different exponentially widened layers vs number of epoch

that the neural network learned so little information regarding the space DTN routing contact graph, as illustrated in figure 5.22. Validation-loss speaking, the network is showing lack of knowledge to reduce it, as it starts with (0.256148095279932) and ends up with (0.24864349104464), that means (Validation Loss Gain for 1 Neuron Width=  $-7.504604e-03$ ) only. Figure 5.23 shows the trend of validation loss versus epoch number.

## 2. $10^1$ Neurons Width Experiment

In this experiment, the width of the neural network layers will be set to  $10^1$  except for the output layer (the fifth layer), where it will be kept to two in order to stay matching the intended output for DTN routing purpose. Figure 5.24



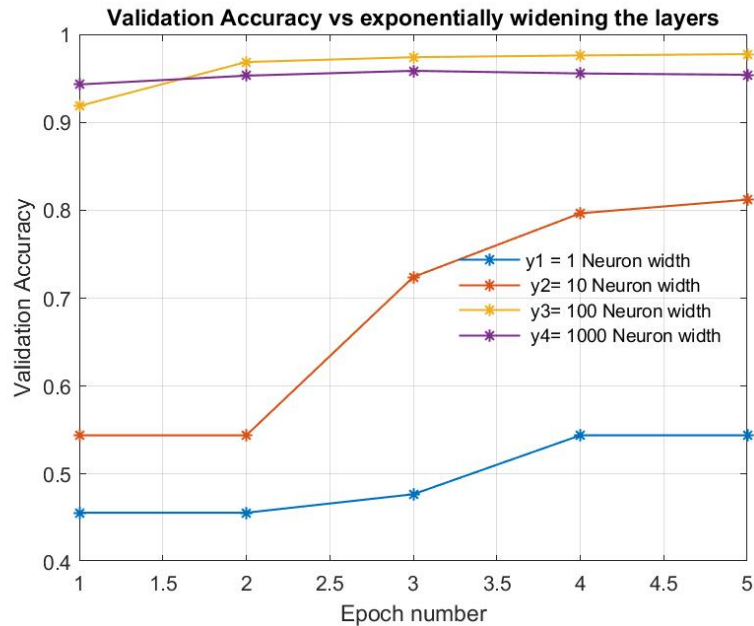


Figure 5.22: Validation accuracy of neural networks with different exponentially widened widths vs number of epochs

demonstrates the different layers of the neural network. As the figure explains, this network has 422 total trainable parameters.

- (a) **Training Phase** Training accuracy in this part of the experiment is showing enhanced results, the trend of accuracy starts higher than the one of one neuron width neural network and the amount of increase is having a higher elevation. Training accuracy begins with the value of (0.51816667368842495) after finishing the first epoch and ends with (0.80988888670172954) after completing the fifth. Thus, having a better Training Accuracy Gain for 10 Neuron Width= 2.917222e-01. Figure 5.20 shows the values of training accuracy per epoch number. When it comes to the training loss, the neural network here starts with a value that is a little bit less than the neural

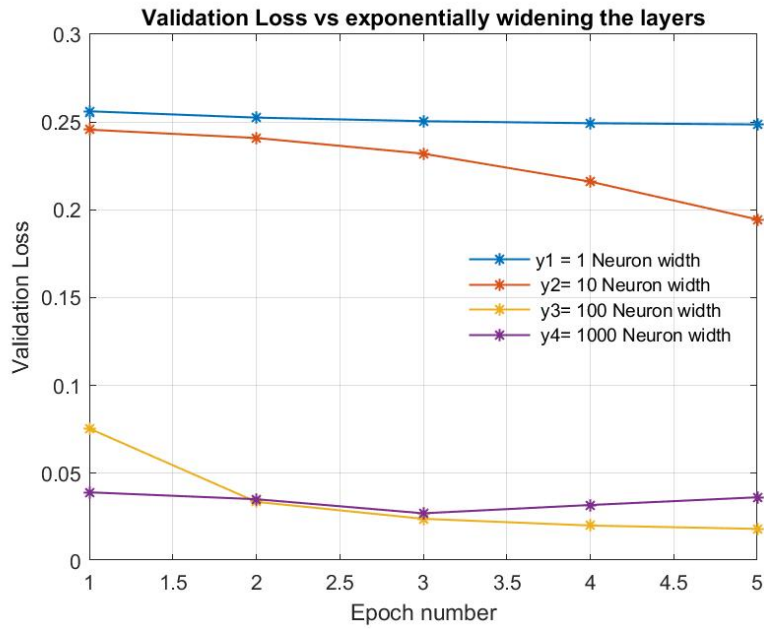


Figure 5.23: Validation loss of neural networks with different exponentially widened layers vs number of epochs

```
model.summary()
```

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 10)	70
dense_2 (Dense)	(None, 10)	110
dense_3 (Dense)	(None, 10)	110
dense_4 (Dense)	(None, 10)	110
dense_5 (Dense)	(None, 2)	22

Total params: 422  
Trainable params: 422  
Non-trainable params: 0

Figure 5.24: Structural design of the neural network used for  $10^1$  Neurons Width experiment

network with one neuron width (0.25223253986901706) and then it shows a higher momentum to decrease it to (0.20440251971284548) after finishing the fifth epoch. That means Training Loss Gain for 10 Neuron Width= $-4.783002e-02$ . Figure 5.21 shows the values of training loss per epoch number. From the perspective of training performance, the neural network with  $10^1$  width is outperforming the network of  $10^0$  width.

- (b) **Validation Phase** Validation accuracy in this part of the experiment is more promising than the experimental indications found in the previous analysis, the trend of accuracy starts higher than the one of one neuron width neural network and with a higher increase momentum. Validation accuracy begins with the value of (0.544000005) after finishing the first epoch and ends with (0.8119999994) after completing the fifth. Thus, having a better Validation Accuracy Gain for 10 Neuron Width= $2.680000e-01$ . Figure 5.22 shows the values of training accuracy per epoch number. Validation loss of this neural network starts with a value that is slightly less than the neural network with one neuron width with a value of (0.24566377587616445) and then shows higher momentum to decrease its loss to reach (0.19455052629113198) after finishing the fifth epoch. That means Validation Loss Gain for 10 Neuron Width= $-5.111325e-02$ . Figure 5.23 shows the values of training loss per epoch number. The validation accuracy and loss show that by having a wider network than the previous experiment, the performance values of validation are more promising. Here the network can give better prediction when given the dataset of space DTN routing contact graph.

```
model.summary()
```

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 100)	700
dense_7 (Dense)	(None, 100)	10100
dense_8 (Dense)	(None, 100)	10100
dense_9 (Dense)	(None, 100)	10100
dense_10 (Dense)	(None, 2)	202

Total params: 31,202  
Trainable params: 31,202  
Non-trainable params: 0

Figure 5.25: Structural design of the neural network used for  $10^2$  Neurons Width experiment

### 3. $10^2$ Neurons Width Experiment

In this experiment, the width of the neural network layers will be set to  $10^2$  except for the output layer (the fifth layer), where it will be kept to two in order to stay matching the intended output for DTN routing purpose. Figure 5.25 demonstrates the different layers of the neural network. As the figure explains, this network has 31,202 total trainable parameters.

- (a) **Training Phase** Having the network with a width of 100 neurons makes the training accuracy in this part of the experiment with a much better shape than both previous ones. Here it does not only start with good value, but also it moves to the boundary of (above 0.9) before even reaching the first epoch.

Training accuracy begins with the value of (0.80294444331692327) after finishing the first epoch and ends with (0.9809999986820751) after completing the fifth. The gain, in this case, is less than the case of 10 neurons width network (Training Accuracy Gain for 100 Neuron Width= 1.780556e-01), that is because the training accuracy here already starts with a much higher value than before; thus it has a smaller range of progress. Figure 5.20 shows the values of training accuracy per epoch number.

The training loss of this neural network starts in a nice spot on the figure with a value that is much more less than the neural network with 10 neurons width with a value of (0.16049624023441639) and then shows huge reduction to reach the value of (0.016421659979724709) after finishing the fifth epoch. That means (Training Loss Gain for 100 Neuron Width= - 1.440746e-01). Figure 5.21 shows the values of training loss per epoch number.

The training accuracy and loss show that by having a wider network than the previous experiment, the performance values of training show great improvement, that is a significant increase in training accuracy and fast reduction in training loss. So far, this network is the best network to learn our space DTN routing data entries efficiently.

- (b) **Validation Phase** Validation accuracy in the case of 100 neurons width starts better than the previous two instances at (0.91849999517202374) and

the biggest gain is by the end of the second epoch, then it keeps increasing slowly until reaching its final value of (0.97749999880790706) which is a very good value. The Validation Accuracy Gain for 100 Neuron Width= 5.900000e-02 that is because there is no much space for the validation accuracy to increase under current circumstances. Figure 5.22 shows the values of validation accuracy per epoch number.

Validation loss of this neural network starts in a better position than the previous two experiments, as here it starts at (0.075479853274300693) and keeps decreasing on the second and third epochs until it reaches a value of (0.018346351644731841) with a Validation Loss Gain for 100 Neuron Width= -5.713350e-02. Figure 5.23 shows the values of validation loss per epoch number. This case is an improvement over the previous two experiments. Next step is to increase the width even more and do the same process of training and calculations.

#### 4. $10^3$ Neurons Width Experiment

In this experiment, the width of the neural network layers will be set to  $10^3$  except for the output layer (the fifth layer), where it will be kept to two in order to stay matching the intended output for DTN routing purpose. Figure 5.26 demonstrates the different layers of the neural network. As the figure explains, this network has 3,012,002 total trainable parameters.

```

: model.summary()

```

Layer (type)	Output Shape	Param #
dense_11 (Dense)	(None, 1000)	7000
dense_12 (Dense)	(None, 1000)	1001000
dense_13 (Dense)	(None, 1000)	1001000
dense_14 (Dense)	(None, 1000)	1001000
dense_15 (Dense)	(None, 2)	2002

```

Total params: 3,012,002
Trainable params: 3,012,002
Non-trainable params: 0

```

Figure 5.26: Structural design of the neural network used for  $10^3$  Neurons Width experiment

- (a) **Training Phase** Testing the network of 1000 neurons width makes the training accuracy starts really good compared to all of the previous cases, yet after finishing the second epoch it intersects with the 100 neurons width network and then the training accuracy gain slows down making it having values inferior to the previous network, that is an indication of potential over-training issue. Training accuracy begins with the value of (0.90155555350) after finishing the first epoch and ends with (0.97105555444955827) after completing the fifth. In this case Training Accuracy Gain for 1000 Neuron Width=  $6.950000e-02$ . Figure 5.20 shows the values of training accuracy per epoch number.

Again the training loss of this neural network starts in a nice spot on the chart with a value that is inferior to the neural network with 100 neurons width with a value of (0.069936941449609297) and then shows small re-

duction to reach the value of (0.021268892775564471) after finishing the fifth epoch. That means (Training Loss Gain for 1000 Neuron Width= - 4.866805e-02). This case does support the previous finding of a possible over-training problem. Figure 5.21 shows the values of training loss per epoch number.

- (b) **Validation Phase** This is the last network tested for exponential width increase, here a width of 1000 neurons makes the validation accuracy in the best shape among all other networks tested. Starts at the highest value of validation accuracy among all, it begins with the value of (0.9430000007) after finishing the first epoch and ends with (0.954000000) after completing the fifth. The gain, in this case, is less than the case of 100 neurons width network (Validation Accuracy Gain for 1000 Neuron Width= 1.100000e-02), that is because the validation accuracy here already starts with a much higher value than before; thus it has a smaller range of progress. Not only the validation gain is smaller than the previous case, but it also ends up with a value that is less than it, that is an indication of start having an over-training case, thus having 1000 neurons width is not recommended, and therefore no more width increase experiments will be performed. Figure 5.22 shows the values of validation accuracy per epoch number.

The validation loss of this neural network starts in a better position than the previous experiment, which is (0.039163163619308536) and it keeps decreasing on the second and third epochs. A strange behavior start after





Figure 5.27: Training accuracy final values of each of the exponentially widened neural networks

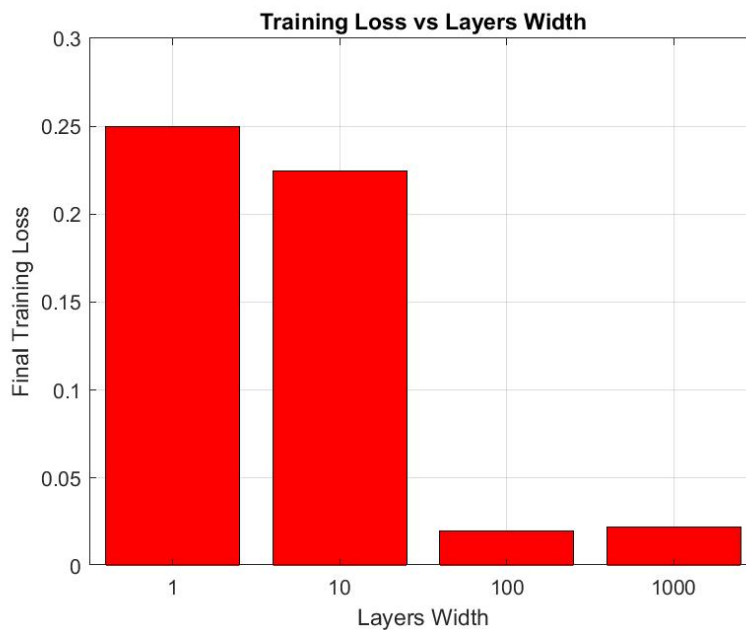


Figure 5.28: Training loss final values vs each of the exponentially widened neural networks

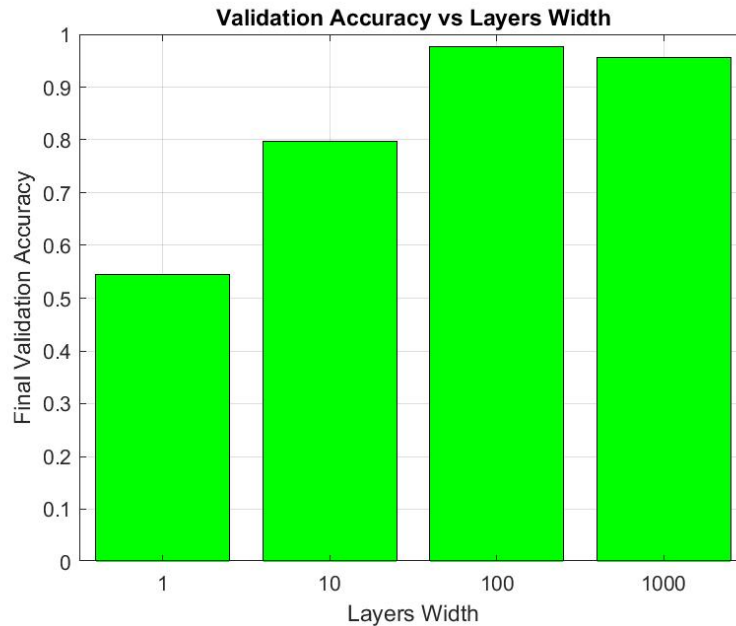


Figure 5.29: Validation accuracy final values vs each of the exponentially widened neural networks

the third epoch, that is an increase in the validation loss to reach a value of (0.03633980268224711) at the end of the training. This confirms the analysis of having an over-fitting or over-training case. Here we have (Validation Loss Gain for 1000 Neuron Width= -2.823361e-03).

Figure 5.23 shows the values of validation loss per epoch number. From this experiment, we can conclude that the validation accuracy and loss show that having a wider network from the previous experiment is not a good choice in this scenario. Not always having a wider network is an indication for a better performance, on the contrary, that may lead to an early over-training. The solution for that is to have less number of epochs if the width

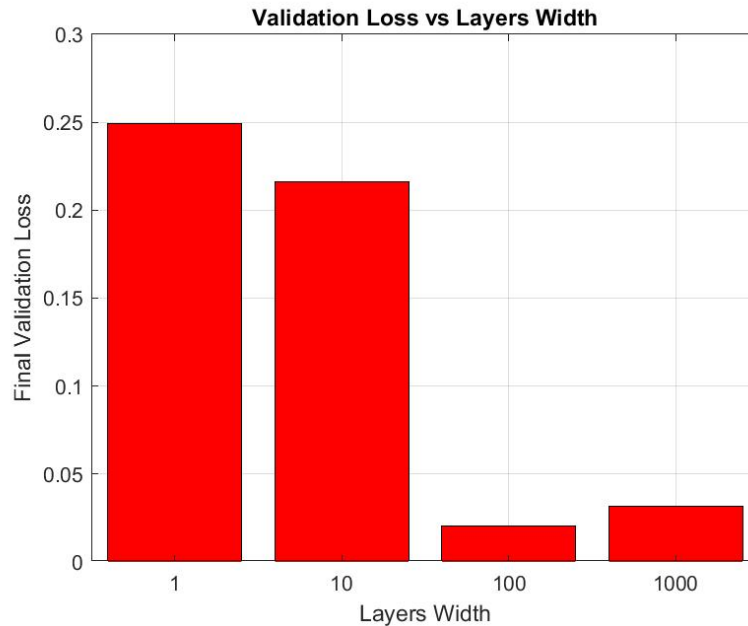


Figure 5.30: Validation loss final values vs each of the exponentially widened neural networks

is kept high, reduce the learning rate, or to decrease the number of batches.

Figure 5.27 shows the training accuracy per network width, Figure 5.28 explains the training losses per network width, Figure 5.29 compares the validation accuracies per network width, and finally 5.30 summarizes validation losses per network width.

## 5.5 The Cone Effect Experiment

During the experimental design procedures, it has happened that in some cases ANNs gave better results when the width of layers gets reduced as we move toward the output

layer, that is like a cone shape where its broadside is the input layer, and its tip is the output layer. This is an experiment to highlight this behavior to make sure if this is a commonly known fact or not. This experiment will test the same network that has shown such behavior before; therefore this network consists of five layers. Activation functions are ReLU, sigmoid, ReLU, sigmoid, and sigmoid from the input layer to the output layer respectively. This network has an output layer of 2 neurons to match the space DTN routing contact graph dataset. The data set consists of (10,000) entries, divided into two parts; six columns are representing the input contact graph, and two columns represent the best-chosen space DTN route. 90% of this data set will be used for training the ANN while the rest will be used for testing it as a validation process. The training will be through five epochs, with a learning rate of (0.00001), and batch size of (5). Metrics used to measure the performance will be the mean square error for loss and binary accuracy.

### **5.5.1 All-Layers-Wide Network Experiment**

The structure of this experiment consists of (31,202) trainable parameters that include all the weights. The width tested in this structure is of (100) neurons per layer. Figure 5.31 explains the structure of the wide-layers neural network used for this part of this experiment.

#### **1. Training Phase**

Training starts with first epoch training accuracy value having (0.66105555839)

Layer (type)	Output Shape	Param #
dense_109 (Dense)	(None, 100)	700
dense_110 (Dense)	(None, 100)	10100
dense_111 (Dense)	(None, 100)	10100
dense_112 (Dense)	(None, 100)	10100
dense_113 (Dense)	(None, 2)	202
Total params: 31,202		
Trainable params: 31,202		
Non-trainable params: 0		

Figure 5.31: Structural design of the neural network used for testing the (all-layers-wide network) part of the cone effect experiment

and keeps increasing per epoch until it reaches the final value of (0.9026666637) which reflects good progress. While the training loss starts with (0.235629853) by the first epoch and ends with (0.0952481381) which confirms the good overall training process. Figure 5.32 reflects the graph of training accuracy per epoch, while figure 5.33 shows the chart of training loss per epoch.

## 2. Validation Phase

The validation accuracy trend for All-layers-wide network confirms the success of the network, as it starts with a value (0.764499998688) after the first epoch and keeps increasing in a proper slope until it reaches the value of (0.90299999). On the other hand, validation loss starts with a value of (0.217754091918468) and decreases rapidly to reach a value of (0.090335655016) after finishing the fifth epoch. This reflects typically good performance. Figure 5.34 shows the graph of validation accuracy, while figure 5.35 shows the validation loss.

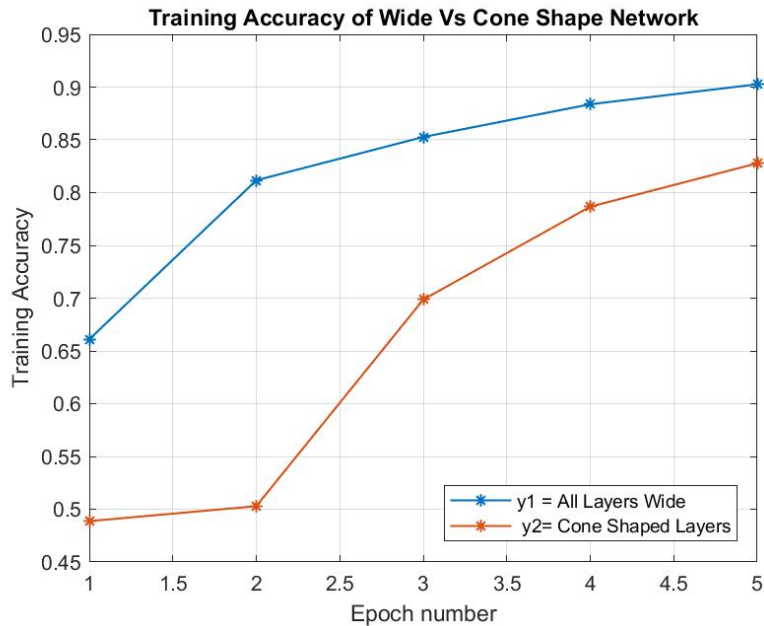


Figure 5.32: Training accuracy comparison between all-wide-layers and cone-shaped networks per epoch number

## 5.5.2 Cone Shape Network Experiment

The structure of this experiment consists of (9,632) trainable parameters that include all the weights. Layer widths used in this structure are of (100, 60, 40, 10, and 2) neurons for the first, second, third, fourth, and fifth layers respectively. Figure 5.36 explains the structure of the neural network used for this part of this experiment.

### 1. Training Phase

For Cone-Shaped ANN, training starts with the first epoch training accuracy value having (0.4887222296) and then increases slightly by the second epoch to

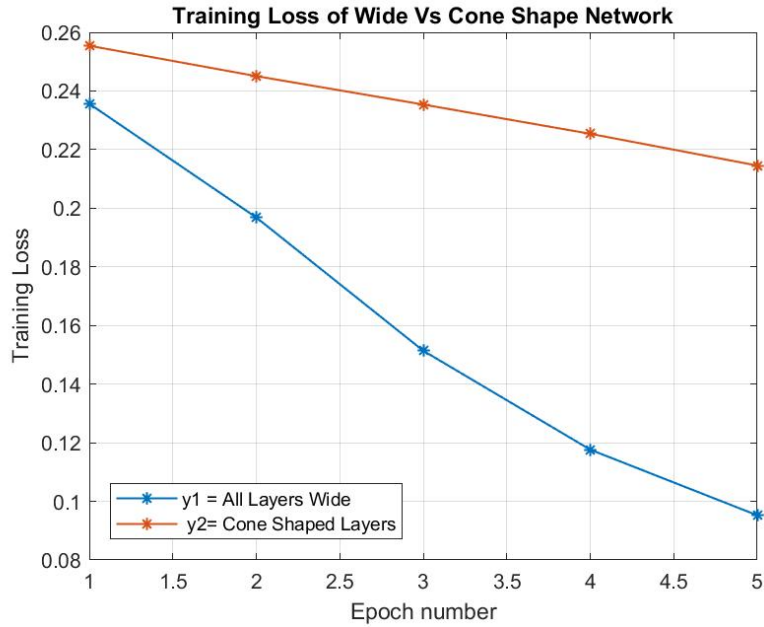


Figure 5.33: Training loss comparison between all-wide-layers and cone-shaped networks per epoch number

reach (0.5028888965). After the second epoch, a rapid increase takes place until it reaches the final value of (0.82777777374618) which reflects good progress; however, that is indicating less learning than the wide neural network in the section before. From the perspective of training loss, it starts with (0.25542107214) by the first epoch and ends with (0.21455759516192) which again indicates a possible worse learning process than the previous case. Figure 5.32 reflects the graph of training accuracy per epoch, while figure 5.33 shows the graph of training loss per epoch.

## 2. Validation Phase

Discussing the results of validation accuracy and loss will give us a more ex-

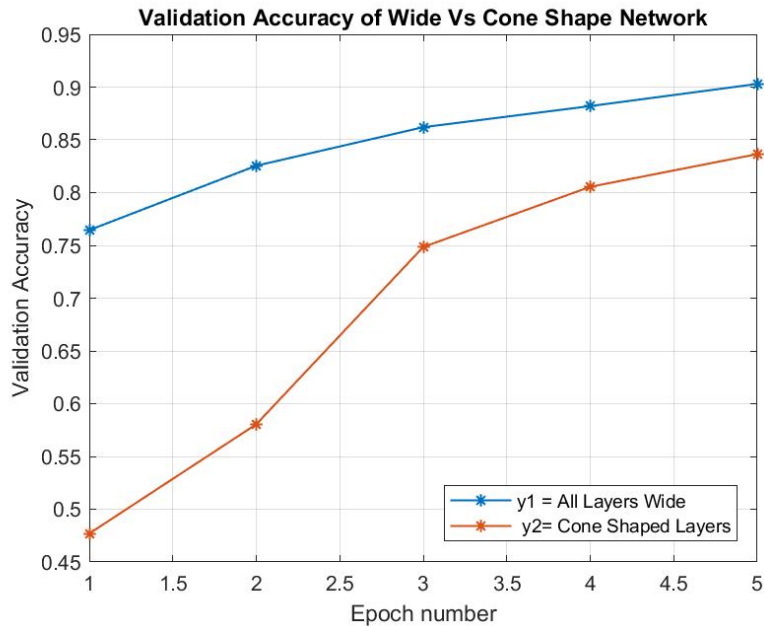


Figure 5.34: Validation accuracy comparison between all-wide-layers and cone-shaped networks per epoch number

explicit indication of the results of the training phase. Validation accuracy of cone-shaped network plot starts from a much lower value than that of the wide neural network in the previous section, as it starts with a value (0.76449999868869) but surprisingly it starts rapidly increasing to reach close value to that of the wide network, that value is (0.83649999484419). The validation loss starts with a value of (0.25052284553647) and then it slowly decreases to reach a value of (0.2099493139237) after finishing the fifth epoch. The validation loss part proves that there is no much decrease in the validation loss as to compete with the all-layers-wide neural network. We conclude that a fully wide network is performing better than a cone-shaped network. Figure 5.34 shows the graph of validation accuracy, while figure 5.35 shows the validation loss.



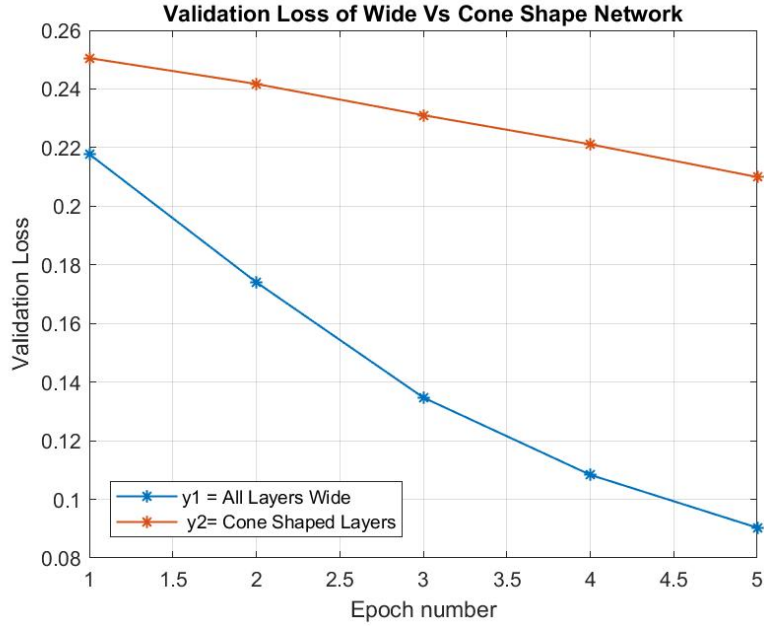


Figure 5.35: Validation loss comparison between all-wide-layers and cone-shaped networks per epoch number

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 100)	700
dense_9 (Dense)	(None, 60)	6060
dense_10 (Dense)	(None, 40)	2440
dense_11 (Dense)	(None, 10)	410
dense_12 (Dense)	(None, 2)	22
Total params: 9,632		
Trainable params: 9,632		
Non-trainable params: 0		

Figure 5.36: Structural design of the neural network used for testing the cone-shaped-network (a part of the cone effect experiment)

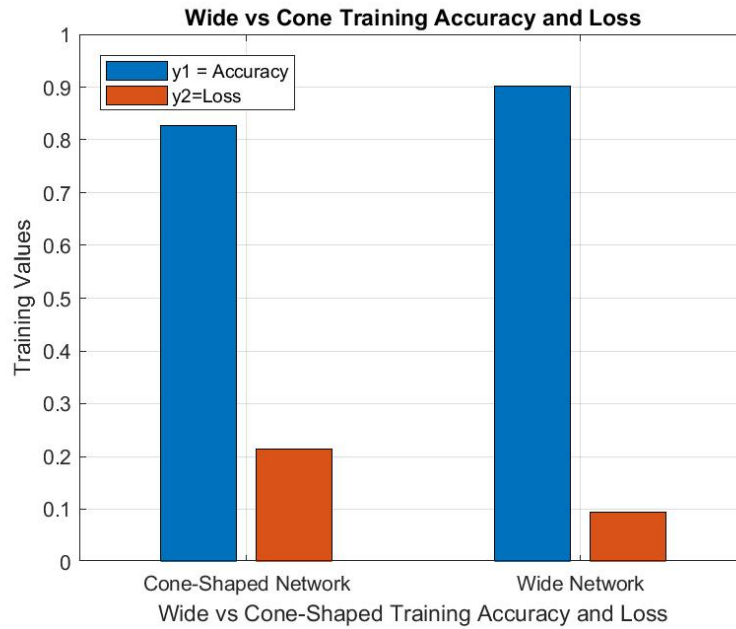


Figure 5.37: Final training performance comparison between cone-shaped and all-layers-wide networks

Figure 5.37 concludes the comparison of training accuracy and loss of both all-wide and cone-shaped neural networks. It is evident in this figure the superiority of the all-wide-network through having higher final training accuracy and less final training loss. On the other hand, figure 5.38 confirms the exceedance of all-wide neural network by having a higher validation accuracy as well as less validation loss.

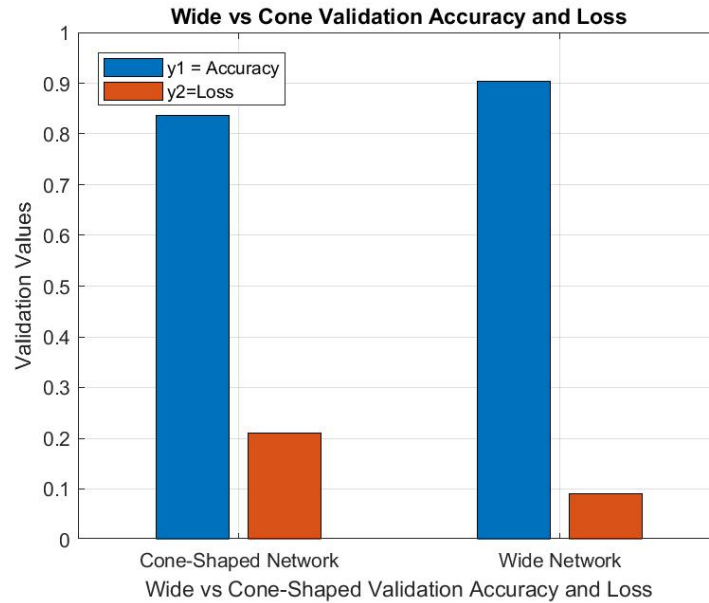


Figure 5.38: Final Validation performance comparison between cone-shaped and all-layers-wide networks

## 5.6 Final Space DTN ANN Routing Solution

This experiment is going to utilize all the knowledge gathered throughout this research as well as the experiments implemented in this chapter. Not only the goal is to have an accurate and powerful space DTN routing using an ANN, but also is to achieve that using minimal resources. The purpose behind that is not only to make this solution robust enough to be implemented on earth stations, but also to build it with least possible resources to make it easier to install it on space DTN nodes themselves.

Using machine learning is time critical in many applications, but when it comes to DTN routing, ML time is not as a vital factor as much. Time is insignificant when it comes to the proposed machine learning DTN routing solution, that is because all

Layer (type)	Output Shape	Param #
dense_10 (Dense)	(None, 250)	1750
dense_11 (Dense)	(None, 100)	25100
dense_12 (Dense)	(None, 2)	202

Total params: 27,052  
Trainable params: 27,052  
Non-trainable params: 0

Figure 5.39: Structural design of the proposed solution (Space DTN ANN routing)

the routing training could be done anytime and as much early as desired, that could be way ahead of the time to receive any data bundle. Not only the proposed solution processes could be performed on early stages, but it could also be synchronized with nodes' periodic power cycling. Furthermore, the proposed solution processing could be done when nodes are idling.

Regardless of time insignificance on the proposed solution as discussed above, this research is already taking into consideration time as one of its essential metrics to minimize. The time needed to train the proposed neural network is around 120 Seconds only. The proposed neural network has 27,052 trainable parameters, figure 5.39 layouts the structure of the proposed artificial neural network. The dataset used for this experiment consists of (600,000) data entries, having the same structure of datasets used in the previous experiments, that is six inputs and two outputs. The inputs represent time intervals of the aforementioned CG, and the output consists of two values representing the best chosen route for the data bundle.



Figure 5.40: Training accuracy and loss of the proposed solution per epoch



Figure 5.41: Training accuracy and loss trends of the proposed solution per epoch

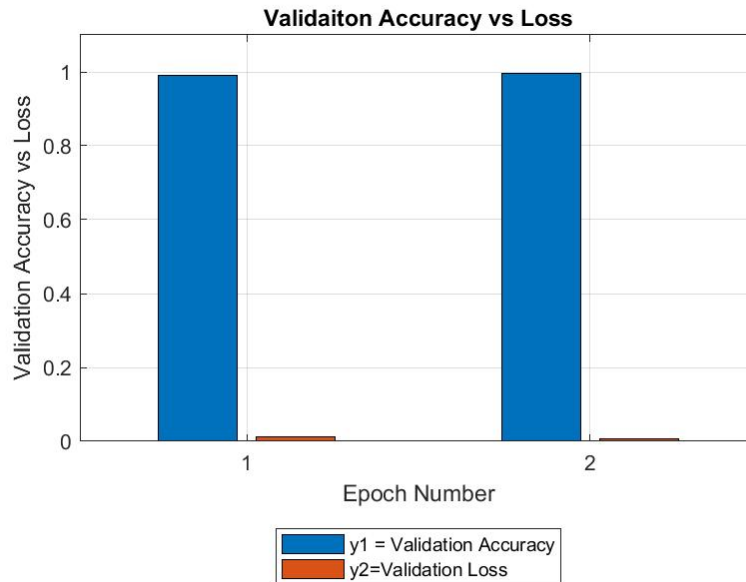


Figure 5.42: Validation accuracy and loss of the proposed solution per epoch

### 1. Training Phase

The training of this network starts very strong that is by achieving (0.975259256) accuracy by the end of the first epoch and (0.992169442819) by the end of the second, which is excellent. Training loss starts with (0.027837005235) by the first epoch and drops down to (0.00903504066500) by the second which also is so promising. To be able to confirm such great results, validation tests will be performed in the next section. Figures 5.40 shows the relation between training accuracy and loss per epoch and figure 5.41 shows the trend of training accuracy and loss.

### 2. Validation Phase

The validation part of this experiment represents the moment of truth, now in this

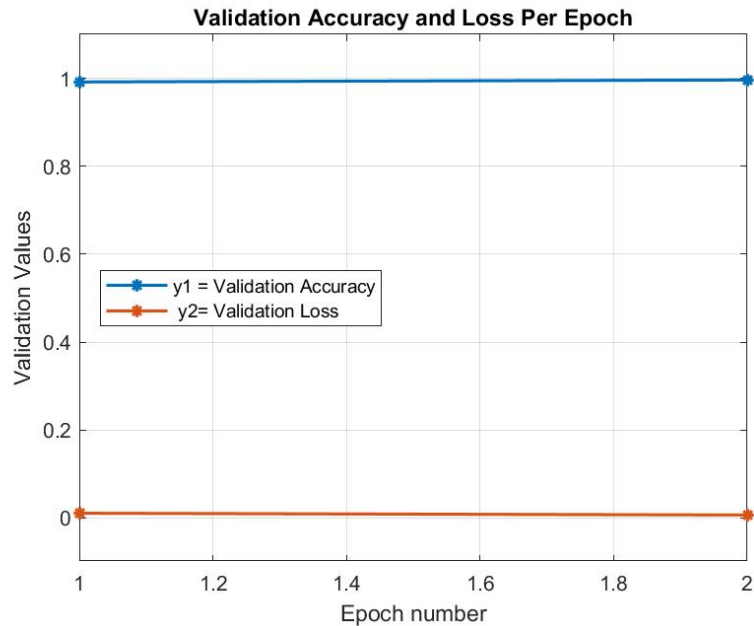


Figure 5.43: Validation accuracy and loss trends of the proposed solution per epoch

part the ANN will be tested with new data set that it has never witnessed before. The dataset used for the testing consists of (60,000) data entries (that is 10% of the size of the original training data set). After the first epoch of training the network, testing the network with the new data set gives (0.99169999829) validation accuracy which is phenomenal as it even exceeds the accuracy of the training itself. Moving to the second epoch, the validation accuracy increases even more to have a value of (0.9961249991) which also exceeds what is expected from the network, because that value exceeds the final training accuracy again.

While validation accuracy values are getting higher, the opposite is happening with the validation loss with values of (0.0108792559) in the first epoch and (0.0067781) in the second. If the decrease in validation loss values means some-

thing, it would be that the training process is successful and it is also far away from reaching the over-training stage. Figure 5.42 illustrates validation accuracy and loss per epoch, while figure 5.43 shows the trend of the proposed solution validation accuracy and loss.

### **Comparison Between The Proposed Solution And An Ideal Solution**

Although the achieved results are so promising, it is helpful to compare them to a perfect world scenario results. In an ideal world, the results would be (100%) validation accuracy, that is a complete 1.0 and a (0%) validation loss, that is (0.0). Now, The difference between the ideal accuracy and the accuracy of the proposed solution is just (0.0039). Furthermore, when the proposed neural network has an accuracy of (99.6125%) that does not mean that it will give (0.39%) wrong outputs, instead this accuracy is just showing the difference of predictions before applying the thresholds. For example, let's consider a case when the ANN will give an output of (0.9,0) instead of an ideal (1.0,0), even though the accuracy will indicate (0.9) after applying the threshold the regulated output would be (1.0,0) which is the same output of a 100% accurate system.

From a routing perspective, if the proposed solution gave a route a value of 0.99 and 0.01 to another, by using a threshold of 0.9 as a deciding edge, Then after applying this threshold, the ANN will still give an answer of 1 to the first route and 0 to the second. As a conclusion, the proposed ANN can provide similar results to the ideal



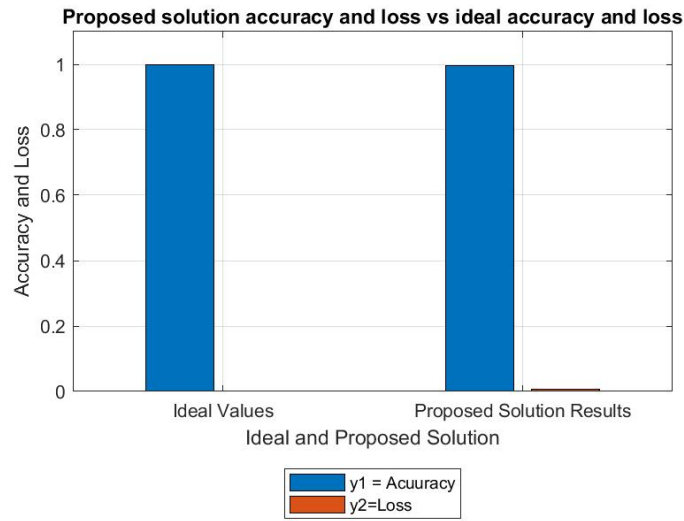


Figure 5.44: Proposed solution validation accuracy and loss vs ideal accuracy and loss world scenarios.

Figure 5.44 shows the difference between both the proposed solution and an ideal world solution before applying thresholds, while when it comes to the actual final decisions made by the proposed neural network, the small difference will be even reduced significantly.

# Chapter 6

## Conclusions and Future Work

### 6.1 Conclusions

Knowing what will happen in the future is a huge advantage. It gives us the power to make early decisions. Likewise, in the proposed solution, the artificial neural network will be trained on future events and will be waiting and ready. By the time any node receives a bundle, no processing is need, as all the routing processing has been completed already. The only task that the node is required to do is to forward the bundle to the right next-hop when the time comes and it appears in sight of contact. Applying the proposed artificial neural network will prevent unnecessary processing as well as flooding found in common DTN routing protocols. To the best of our knowledge, this is the first work that introduces and utilizes artificial neural networks to implement delay-tolerant networks routing.

Conclusions of this thesis work could be categorized into two types, proposed solu-

tion implementation conclusions, and solution concluded advantages.

### **Proposed Solution Implementation Conclusions**

These are the conclusions obtained by going through the experimental steps to build this solution, and they are:

1. Disregarding their width, adding more layers into an artificial neural network causes its training and validation accuracy trends to escalate (considering the accuracy values are still not reaching the over-training zone). In other words, adding more layers to an artificial neural network will make it gain more significant increments of accuracy per epoch. The opposite could be said regarding training and validation losses as they start having a noticeable decrease in their values by adding more layers to the network.
2. Adding more layers into artificial neural networks empower them to reach higher accuracy values for the same number of epochs.
3. Adding more layers to artificial neural networks may cause to over-train them, to avoid that no more layers should be added when a neural network starts showing a negligible amount of accuracy improvements.
4. It is much harder for an artificial neural network with one layer to solve problems, in fact, some problems can never be solved with one layer, as the case of one layer perceptron trying to solve an XOR logic function.
5. Adding more layers to artificial neural networks causes longer training times for the same number of epochs.

6. For every problem to be solved using artificial neural networks, there is a point when adding more layers becomes inefficient, either by spending longer training times, wasting resources, or both.

The achieved goal of this thesis holds many advantages over the traditional routing protocols, some of these advantages are:

1. The time of data to move from source to destination would be just the time of actual transmission, as routing would be done before data arrive to any node and thus routing processing time equals zero.
2. Many popular DTN routing protocols are to some extent replication-based such that several copies of a message are kept in the network to improve the message delivery. This method leads to an unnecessary waste of nodes resources [37]. In this proposed solution we can have zero replicated-based data.
3. Many replication-based approaches lack the capability of a garbage-data collection, that is, although a bundle has been delivered, the buffer resources associated with replicas of that bundle in other nodes cannot be freed automatically even though these copies are not useful anymore. Moreover, such bundles are forwarded continually, wasting even more resources [37]. The proposed solution will produce zero-garbage-data.
4. The solution proposed in the research does not only find the route to the destination, but it also exposes the route with Earliest Transmission Opportunity (ETO), if that means anything it would be fastest possible delivery to the target.

5. The deeper the DTN nodes go into space, the more efficient they will work if using the proposed solution. As DTN routing using ANN will provide the nodes with future knowledge about when to transmit/receive the bundles, that could be used to turn off the nodes for more extended periods and turn them on just before data handshaking is about to take place. Thus, this solution minimizes DTN nodes power consumption.
6. The proposed solution can increase network security, as nodes will be out of the grid or wholly powered off for the time that they are not needed. That prevents nodes from being open for security attacks. Furthermore, the nodes will be already knowing which their next encounter is and will not make contact with any other node.
7. It is easy to add or remove new nodes to the network, all that it takes is to feed them the latest learned model like all the other nodes in the network.

## 6.2 Future Work

There are several areas of possible future work that could be built on top of the solution proposed by this thesis, few of them are listed below:

1. Virtual Private Network (VPN) becomes possible if the presented solution is used, that is by feeding chosen group of DTN nodes a trained neural network model that includes only those nodes with no other. That's is simply how a DTN VPN could be created.

2. With the proposed solution nodes could have different levels of priority. Express routes could be built by synchronizing priority nodes according to specific patterns or according to priority nodes power-cycle.
3. Emergency hot-lines are possible by using the proposed implementation, a trained model can contain zero-possibility to some standby nodes, and when an emergency message want to be sent from space crew back to earth, they will just have to send "Emergency Model" into neighboring nodes, and that will turn-on and deploy the emergency route for that purpose only.

# Bibliography

- [1] Z. Jiao, R. Tian, B. Zhang, and C. Li, "Dtn routing with back-pressure based replica distribution," *Journal of Communications and Networks*, vol. 16, no. 4, pp. 378–384, 2014.
- [2] S. Burleigh, "Interplanetary overlay network: An implementation of the dtn bundle protocol," *Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2006, 2007*.
- [3] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 27–34.
- [4] T. Abdelkader, K. Naik, A. Nayak, N. Goel, and V. Srivastava, "A performance comparison of delay-tolerant network routing protocols," *IEEE Network*, vol. 30, no. 2, pp. 46–53, 2016.
- [5] P. Puri and M. P. Singh, "A survey paper on routing in delay-tolerant networks," in *Information Systems and Computer Networks (ISCON), 2013 International Conference on*. IEEE, 2013, pp. 215–220.

- [6] F. De Rango, M. Tropea, G. B. Laratta, and S. Marano, "Hop-by-hop local flow control over interplanetary networks based on dtn architecture," in *Communications, 2008. ICC'08. IEEE International Conference on*. IEEE, 2008, pp. 1920–1924.
- [7] A. Socievole, F. De Rango, and C. Coscarella, "Routing approaches and performance evaluation in delay tolerant networks," in *Wireless Telecommunications Symposium (WTS), 2011*. IEEE, 2011, pp. 1–6.
- [8] C. Caini, R. Firrincieli, and M. Livini, "Dtn bundle layer over tcp: Retransmission algorithms in the presence of channel disruptions." *JCM*, vol. 5, no. 2, pp. 106–116, 2010.
- [9] Z. Huakai, D. Guangliang, and L. Haitao, "A self-adaptive deep space dtn routing model," in *Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference on*. IEEE, 2016, pp. 333–336.
- [10] R. H. Frenkiel, B. Badrinath, J. Borras, and R. D. Yates, "The infostations challenge: Balancing cost and ubiquity in delivering wireless data," *IEEE Personal Communications*, vol. 7, no. 2, pp. 66–71, 2000.
- [11] J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM, 2005, pp. 284–291.
- [12] S. Jain, K. Fall, and R. Patra, *Routing in a delay tolerant network*. ACM, 2004, vol. 34, no. 4.



- [13] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," *ACM SIGOPS Operating Systems Review*, vol. 22, no. 1, pp. 8–32, 1988.
- [14] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," *Technical Report CS-200006, Duke University*, 2000.
- [15] Z. Feng and K.-W. Chin, "A unified study of epidemic routing protocols and their enhancements," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 1484–1493.
- [16] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *Service assurance with partial and intermittent resources*. Springer, 2004, pp. 239–254.
- [17] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM, 2005, pp. 252–259.
- [18] P. Maitreyi and M. S. Rao, "Design of binary spray and wait protocol for intermittently connected mobile networks," in *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*. IEEE, 2017, pp. 1–3.
- [19] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility," in *null*. IEEE, 2007, pp. 79–85.

- [20] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, “Maxprop: Routing for vehicle-based disruption-tolerant networks,” in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, April 2006, pp. 1–11.
- [21] G. Araniti, N. Bezirgiannidis, E. Birrane, I. Bisio, S. Burleigh, C. Caini, M. Feldmann, M. Marchese, J. Segui, and K. Suzuki, “Contact graph routing in dtn space networks: overview, enhancements and performance,” *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38–46, March 2015.
- [22] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [23] N. Bezirgiannidis, F. Tsapeli, S. Diamantopoulos, and V. Tsaoussidis, “Towards flexibility and accuracy in space dtn communications,” in *Proceedings of the 8th ACM MobiCom workshop on Challenged networks*. ACM, 2013, pp. 43–48.
- [24] R. Dudukovich, A. Hylton, and C. Papachristou, “A machine learning concept for dtn routing,” in *Wireless for Space and Extreme Environments (WiSEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 110–115.
- [25] S. Krug and J. Seitz, “Challenges of applying dtn routing protocols in realistic disaster scenarios,” in *Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on*. IEEE, 2016, pp. 784–789.
- [26] A. Keränen, J. Ott, and T. Kärkkäinen, “The one simulator for dtn protocol evaluation,” in *Proceedings of the 2nd international conference on simulation tools*

- and techniques.* ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 55.
- [27] S. Krug, M. F. Siracusa, S. Schellenberg, P. Begerow, J. Seitz, T. Finke, and J. Schroeder, “Movement patterns for mobile networks in disaster scenarios,” in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, June 2014, pp. 1–6.
- [28] S. Haykin, *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.
- [29] M. Ito, “What can we expect from neural network models?” in *Neural Networks, 1993. IJCNN’93-Nagoya. Proceedings of 1993 International Joint Conference on*, vol. 1. IEEE, 1993, p. 5.
- [30] D. Kriesel, “A brief introduction on neural networks,” *Citeseer*, 2007.
- [31] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Neural networks and learning machines.* Pearson Upper Saddle River, 2009, vol. 3.
- [32] Y. Sai, R. Jinxia, and L. Zhongxia, “Learning of neural networks based on weighted mean squares error function,” in *2009 Second International Symposium on Computational Intelligence and Design*, vol. 1, Dec 2009, pp. 241–244.
- [33] R. Sathya and A. Abraham, “Comparison of supervised and unsupervised learning algorithms for pattern classification,” *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34–38, 2013.

- [34] J. M. Peña, J. A. Lozano, P. Larrañaga, and I. Inza, “Dimensionality reduction in unsupervised learning of conditional gaussian networks,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 590–603, 2001.
- [35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [36] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [37] Y. Li, Y. Li, L. Wolf, A. Lingren, and J. Wang, “A named data approach for dtn routing,” in *Wireless Days, 2017*. IEEE, 2017, pp. 163–166.

# Abbreviations

**Adam** Adaptive Momentum Estimation. 44

**AI** Artificial intelligence. 23

**ANN** Artificial Neural Networks. iii, vi, 3, 9, 10, 23, 25–30, 34, 36, 39, 43, 44, 46, 48, 51–57, 59, 61, 63, 65, 69, 72, 86, 87, 89, 94, 98, 99, 104

**BER** Bit Error Rate. 7

**BIOS** Basic Input/Output System. 50

**BP** Bundle Protocol. 21

**CG** Contact Graph. iii, 3, 10, 43, 51, 53, 57, 64, 72, 73, 95

**CGR** Contact Graph Routing. 18, 20, 21

**CPU** Central Processing Unit. 49

**CPUP** Contact Plan Update Protocol. 20

**Del** Average Packet Delay. 19

**DINET** Deep Impact Network Experiment. 20

**DR** Delivery Ration. 19

**DTN** Delay Tolerant Networks. iii–vii, 2–12, 15–21, 24, 38, 43, 44, 46, 48, 50, 51, 53, 58, 64, 87, 94, 103, 104

**DVC** Packet Delivery Cost. 19

**ETO** Earliest Transmission Opportunity. 20

**FIFO** First-in-First-Out. 15

**ION** Interplanetary Overlay Network. 21

**IP** Internet Protocol. 1, 2, 20

**IPN** Interplanetary. 5

**LTP** Licklider Transmission Protocol. 21

**MANET** Mobile Ad hoc Networks. 21

**MIND** Minimize The Total End-to-End Delay. 19, 20

**MINH** Minimize The Total Number of Hops. 19, 20

**ML** Machine Learning. 3, 21, 23, 38, 94

**MSE** Mean Square Error. 34

**NASA** National Aeronautics and Space Administration. 21

**OMNET++** Objective Modular Network Testbed in C++. 21

**ONE** Opportunistic Network Environment. 22

**OSI** Open System Interconnection. 6

**ProPHET** Probabilistic Routing Protocol using History of Encounters and Transitivity. 2, 15, 17, 19, 22

**RAM** Random Access Memory. 49

**ReLU** Rectified Linear Unit. 31, 46, 65, 69, 70, 72, 87

**RNN** Recurrent Neural Network. 29

**SNF** Spray and Focus. 16

**SnW** Spray and Wait. 15, 19

**SSD** Solid State Drive. 49

**Tanh** hyperbolic Tangent. 31, 46, 63, 67–70

**TCP** Transfer Control Protocol. 1, 2

**TTL** Time to Live. 19

**UAV** Unmanned Ariel Vehicles. 22

**UDP** User Datagram Protocol. 1, 2

**VPN** Virtual Private Network. 104