GCU
Glasgow Caledonian
University

A Comparison of Artificial Neural Network Learning Algorithms to Predict Voice over IP Call Quality

Honours Project (MHG405293)
Honours Project Final Report
Module Leader: Dr Richard Foley

Student JJ 10-11
BSc (Hons) Networking & Systems Support
Matriculation Number: 2006xxxx

Project Supervisor: Kapilan Radhakrishnan
Second Marker: Dr Richard Foley

"Except where explicitly stated all work in this document is my own"
Signed:______________________ Date:_____________________

Abstract

Voice over IP (VoIP) technology has enjoyed increased implementation during the last decade but suffers from many Quality of Service issues. It has become necessary for service providers to be able to monitor and track the quality of calls. Many solutions are currently available in the industry such as the Perceptual Evaluation of Speech Quality (PESQ) and E-Model methods, but both of these common standards have drawbacks such as the intrusive method of the PESQ reference signals, which makes it unsuitable for use in real-time situations or the drawback of the heavy computational costs of the E-Model which also suffers from lower accuracy than PESQ. A lightweight, non-intrusive computational model taking into account network parameters while maintaining the accuracy of PESQ was required. Artificial neural networks (ANN) are a type of artificial intelligence based on the central nervous system. They are commonly applied to problems of pattern recognition between non-linear variables. Using artificial neural networks it was possible to establish a relationship between the network parameters of delay, packet loss and jitter to predict the Mean Opinion Score (perceived quality) of the call.

The aim of this report was to investigate, through simulation, the ability of ANN learning algorithms to accurately predict the quality of a call compared to the existing PESQ benchmark. In order to provide realistic results, data sets generated by the PESQ method were sourced. To find the best ANN paradigm comparisons were drawn between two learning algorithms; Gradient Descent and Levenberg-Marquardt and various network architectures.

The conclusions of the experiment clearly demonstrate that ANNs are comparable to previous methods used. The Levenberg-Marquardt algorithm performs to a high standard with little computational cost. This displays the efficiency of the Levenberg-Marquardt algorithm not only over Gradient-Descent, but over previous prediction methods. The speed of the algorithm means it could be applied in real-time situations, while constantly learning and evolving.

Table of Contents

1.0 Introduction **3**

1.1 Background 3

1.2 Project Outline & Research Question 8

1.3 Aim and Objectives 8

1.4 Hypotheses 10

1.5 Report Structure 11

2.0 Literature Review **13**

2.1 Investigation into the characteristics of Voice over IP 13

2.2 Investigation into call quality standards and current measurement methodologies 17

2.3 Investigation into artificial neural networks and associated learning algorithms 24

3.0 Methods **29**

3.1 Primary Research Methodology and Justification 29

3.2 Design of the Primary Research Method 29

3.3 Data Collection 30

3.4 Network Topology Design 31

3.5 Simulation Phases 33

3.6 Data Analysis Stage 33

3.7 Hypothesis Testing 34

4.0 Presentation of Results **36**

1.1 Results with SPEEX Codec 36

1.1 Results with G711a Codec 54

1.1 Misc. Results 70

5.0 Summary and Conclusions **79**

6.0 References **85**

7.0 Appendixes **91**

1.0 Introduction

The following section will introduce the background of the two research areas of Voice over IP technologies and artificial neural networks. Initially VoIP is introduced by outlining the history and the increased usage trends in businesses, educational facilities and personal use. The quality issues faced in VoIP communication is highlighted and traditional methods for speech quality prediction are introduced.

Artificial neural network history and concepts are illustrated briefly along with machine learning and the background of the learning algorithms. The supplemental use of neural networks to provide quality control in communication networks is also introduced.

The remainder of this section identifies the research question, aim, objectives and hypotheses that will be tested. This section concludes with an overview of the content of the document.

1.1 Background

1.1.1 Voice over Internet Protocol

Voice over Internet Protocol (VoIP) describes a family of transmission protocols to provide the ability to transfer voice communication digitally over existing Internet Protocol (IP) networks. When a network carries both voice and data it is referred to as network convergence. The bonuses of network convergence include lower costs, simpler management/maintenance and improved cooperation between services (Gorur P.Y. 2006).

Previously, voice communication was handled solely by the Public Switched Telephone Network (PSTN). PSTN handles calls by circuit switching which means on a call there is one continuous link from end-to-end. Whereas a VoIP call, by nature running over an IP packet switched network means call data is digitised and split into sequenced chunks and sent across the communications network (Wallace, 2009).

In order to have any success call quality of VoIP calls had to be of satisfactory level for users in comparison to the current standard enjoyed through the PSTN. Broß and Meinel (2008) suggest the PSTN to be "the ultimate benchmark" for VoIP communication.

VoIP has enjoyed increased implementation in the corporate world – VoIP service providers have had a substantial increase of subscribers in Europe from 6 million subscribers in 2005 to 34.6 million in 2008. (TeleGeography, 2009a). This is due to businesses are discovering large monetary savings by using the IP network that they already have in place, which many corporate environments often find are not used to their full potential and thus require no expansion in order to carry voice traffic (Wallace, 2009). The increased VoIP usage across corporate infrastructures is also related to the level of control available on privately managed networks allowing increased security options and management of the service quality (Cisco Systems, 2009). Cearley et al (2005) predict that by the end of 2010 over 95% of large corporations and establishments will have converged networks carrying digitised voice traffic.

Home users are also beginning to explore VoIP technologies more and more, such with the usage of Skype and Google Voice (Dudman, 2006). These technologies allow users to communicate with friends or colleagues all over the world without worrying about the cost of international calls due to reduced cost of using VoIP. Skype now accounts for almost one in ten of every international voice call made (TeleGeography, 2009b).

Dudman (2006) discusses the use of VoIP technologies in Further/Higher Education which is allowing students, researchers and teachers more accessible learning and even easing those with disabilities into the learning environment.

1.1.2 Voice over IP Quality Issues

Voice communication traditionally has enjoyed high quality call transmission but when IP networks were developed, they were constructed without the ability to carry voice communication in mind so the initial period of VoIP was troublesome and suffered from poor call quality (Keepence, 2004). This has been apparent throughout the development of VoIP.

Network parameters that directly affect the transmission of a call have been identified by Goode (2002) as delay, packet loss and variation of delay (jitter). This has been validated further by Sun (2004) & Radhakrishnan et al (2010) in which they demonstrate a method for predicting call quality based on these parameters.

When VoIP traffic is transmitted across an IP network, it is broken down into packets, like all IP traffic whereby each packet of data is given a sequence number. It is when these packets arrive at different times that delay and jitter come into play thus causing late arrival of speech or clipping of the voice. Packet loss then causes calls to miss out parts of speech completely or even drop the call (Goode, 2002) (Wallace, 2009).

Users of VoIP have indicated that the primary reason for dissatisfaction with VoIP technologies is due to the level of quality experienced in calls. Phillips et al (2009) analysed that over 90% of those who abandoned a VoIP implementation was due to dissatisfaction with call quality and 70% of those considering VoIP solution hold reservations on call quality.

1.1.3 Call Quality Measurement Methodologies

The International Telecommunications Union provides a scale for measuring the perceived speech quality of voice transmissions known as the Mean Opinion Score (MOS). The MOS scale is a numerical representation of transmission quality spanning from 1 to 5 or bad to excellent. (International Telecommunications Union, 1996)
Call quality measurement methods are categorised as either subjective or objective (Sun, Ifeachor 2006). Traditional subjective testing is performed by human test participants giving a transmission a rating such as MOS. These tests are considered to be resource heavy in terms of time and cost (Mahdi, 2007). Objective test methods are computational models built to replicate human analysis of perceived speech quality. (Radhakrishnan et al 2010).

Perceived call quality can be predicted using the ITU-T PESQ (Perceptual Evaluation of Speech Quality) model. This standard is used to measure perceived call quality worldwide by communication systems manufacturers. It is used to measure end-to-end perceived voice quality by measuring the difference in quality of two sound samples. (International Telecommunications Union, 2001) Due to the use of a reference signal it is unsuitable for dynamic and real-time measurement of perceived call quality in a VoIP environment. (Chi, Womack, 2009) This objective method is an intrusive method of measuring voice quality by comparing the end to end signal (Lepetgui et al 2010).

The ITU-T E-Model is a speech quality prediction computational model which outputs the value of ‘R’, this metric describes the speech quality rating. The E-Model takes into account various factors such as equipment and network parameters; for example probability of packet loss and equipment rating (International Telecommunications Union, 2009). Radhakrishnan et al (2010) acknowledge the ability of the E-Model as a model to method to predict VoIP quality but criticise the high computational costs of a practical implementation. A modified version of the E-Model was presented by Carvalho et al (2005); this practical implementation gave accurate results based on the E-Model’s methodology giving results that could be used to intelligently manage VoIP traffic. The E-Model is considered Non-Intrusive as it does not compare the voice signal. (Lepetegui et al, 2010)

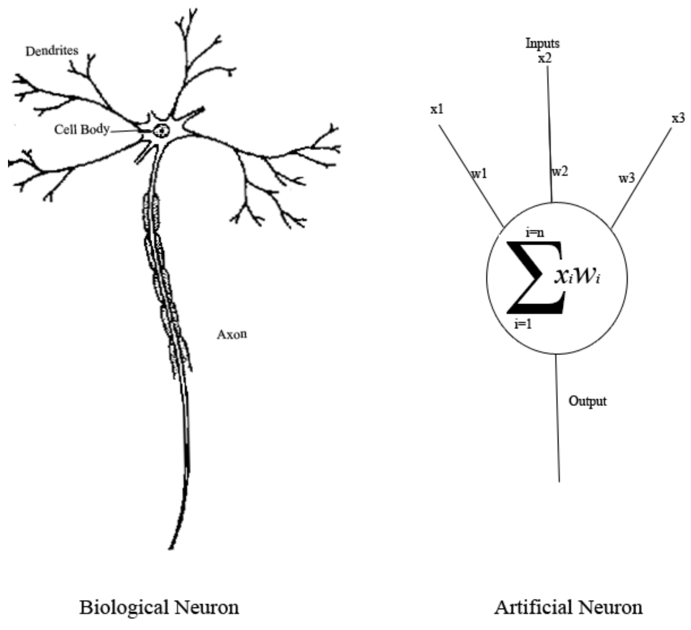
1.1.4 Artificial Neural Networks

Artificial neural networks are simply an attempt to emulate the central nervous system of biological entities such as animals and humans in order to intelligently process information. (Gupta et al 2003) The human brain is composed of a network of approximately 10^{11} neurons; artificial neural networks are a relatively simplified emulation of this. (Gelenbe, 2002)

As a concept, man-made simulations of the central nervous system have been hinted at for over a century with William James, an American psychologist. Eberhart & Dobbins (1990) believe that his conclusion that a neuron’s function is the sum of its inputs has thus paved the way for simulating the brain’s central nervous system. By 1940 artificial neural networks had conceptualised to be able to calculate any mathematical problem, theoretically (Hagan et al, 1996). Furthermore Hopfield (1982) describes a paradigm which researchers in the field of Neural Networks had been thus far unable to do – which was achieve practicality in the implementation and applications of his model. This, and his model of associative memory paved the way for further research into the field as research into artificial neural networks had somewhat halted previously (Eberhart, Dobbins, 1990).

Artificial neural networks are computational models where each neuron represents a mathematical function. (Dreyfus, 2005) Each Neuron can have multiple inputs and outputs leading to multiple neurons. The construction of an artificial neuron is much like that of a biological neuron. As shown in Figure 1.1, the biological neuron compares to the Artificial Neuron in the following ways:

- The dendrites of the biological neuron are like the inputs of the artificial neuron, there are many to one neuron.
- The cell body of the biological neuron is directly comparable to the function of the artificial neuron, as the signals carried by the dendrites or inputs are calculated within this element.
- The axon of the biological neuron is what would be considered the output of the artificial neuron. (Hagan et al 1996)



Biological Neuron from http://www.arthursclipart.org/medical/digestive/neuron.gif

Figure 1.1 - Biological Neuron and Artificial Neuron (based on research from Hagan et al 1996)

Neurons are interconnected to form neural networks or the central nervous system. Like their biological counterparts artificial neurons are also linked together and the most common type of artificial neural network is a feedforward network. Feedforward networks only receive input from the previous layers in the network. (Demuth, Beale 1997)

Artificial neural networks have varied applications but they are best at problems which involve pattern recognition (Bakircioglu, Tocak, 2000) not just in computational or engineering disciplines but in problems such as financial stocks and handwriting recognition (Hagan et al 1996).

1.1.5 Artificial Neural Network Training

Neural Networks are trained by methodologies known as Learning Algorithms or Training Algorithms. This technique is called supervised training which requires the teacher to have the ability to calculate the output and provide exemplar data for the algorithm to use. (Dreyfus 2005)

Gelenbe & Hussain (2002) note that learning is the aspect that makes neural networks useful and this technique identifies their key usage as pattern detection.

Gradient Descent is a training algorithm for feedforward neural networks. This methodology adjusts the weights of the neural network by a proportional amount based on the target and actual outputs. (Kartakapoulos, S.V, 1997)

The Levenberg-Marquardt method is an improved algorithm based on previous backpropagation learning rules. This algorithm enables the networks to learn at greater speed than previously capable using Gradient Descent. (Ranganathan, 2004) Demuth & Beale (1997) estimate the Levenberg-Marquardt algorithm to be 10-100 times faster at training a neural network than Gradient Descent.

Direct comparison has been made by Escalante & Malki (2006) in which they pitted the learning algorithms against each other at solving chess problems. This shows the direct comparability between the two learning algorithms for solving problems of pattern recognition.

1.1.6 Voice over IP & Neural Networks

Neural networks have been used to control systems as they are able to accurately detect patterns (Hagan, Demuth, 1999) and implementation of this technology has been researched in communication networks for some time. The limitations of IP networks have brought about the need for intelligent decisions to be made based on Quality of Service parameters as identified by Gelenbe, (2003). These intelligent decisions are one application of neural networks which have been shown by the current applications of neural networks in IP communications.

As quality issues in VoIP have been previously been identified as strictly through patterns of adverse network conditions thus the application of artificial neural networks to predict call quality is fundamental. Neural networks have been implemented in many forms to supplement VoIP.

An artificial neural network solution based on packet loss was developed by Sun & Ifeachor (2002) and the results proved conclusive, but no comparison was drawn on learning algorithms and including additional statistics.

Masugi (2002) developed a mapping tool which displayed ‘Quality of Service Level’ mapping on a display using neural networks. This implementation demonstrates a practical application of neural networks with Voice over IP.

Radhakrishnan et al (2010) developed a method to assess call quality over VoIP using random neural networks tested against the ITU-T PESQ the results from their experiments noted highly reliable outcomes from the neural networks. No comparison of learning algorithms was ever made in order to assess the effect on the results.

1.1.7 Voice over IP & Neural Networks

This experimental project will provide a method of assessing call quality using artificial neural networks specifically using two learning algorithms. The results of which can be assessed against the ITU-T PESQ model to measure the accuracy.

The key advantage of using a trained artificial neural network over the existing “static” methodologies is their ability to dynamically react and adapt to the unpredictable network parameters that affect Voice over IP traffic. (Sun, Ifeachor, 2002)

This method will allow further research into other artificial or random neural network learning algorithms in order to draw comparison. Further this could be applied in order to modify network parameters based on the output of the neural network developing truly intelligent network architecture.

1.2 Project Outline & Research Question

This section identifies the project outline and the research question that the primary research will address. The aim and objectives are then detailed along with the hypotheses along with their justification.

1.2.1 Project Overview

In order to maintain a high Quality of Experience (QoE) for the end user, Voice Traffic should be monitored and managed in real time (Yamato & Beerends, 1997). Initial research has revealed the limitations of current methods in both the inability for ITU-T PESQ to monitor real-time traffic, and the heavy computational requirements of the ITU-T E-Model (Radhakrishnan et al 2010). An experimental based project will be undertaken to investigate the ability for artificial neural network learning algorithms to train an artificial neural network to predict the perceived speech quality on a communications network carrying VoIP traffic.

The accuracy of the method will be detailed against the ITU-T PESQ model which previous research by Paglierani and Petri (2007) concludes to be the most accurate form of objective testing with figures from the ITU-T showing a high correlation to subjective tests furthering the reliability (International Telecommunications Union, 2001). This research will provide insight into the ability of a trained artificial neural network in comparison to existing methodologies and subsequently the ability of the individual learning algorithms identified.

From this, the research question of this project has been derived and is stated below.

> **"How do the learning algorithms of Gradient Descent & Levenberg-Marquardt affect the ability of an artificial neural network to predict speech quality on a communications network running Voice over IP traffic taking into account delay, packet-loss and jitter in comparison to the ITU-T PESQ model?"**

1.3 Aim and Objectives

The main aim of the project is to perform an experiment that will compare the ability and performance and identify the strengths and weaknesses of two learning algorithms to train an artificial neural network to predict the Mean Opinion Score of the voice quality on a communications network. This will be measured against the ITU-T PESQ results. This is to meet the demand for an efficient, low computational cost, real-time paradigm for monitoring speech quality. In order to complete this aim various objectives have been identified as both secondary and primary research.

The objectives that will be met through secondary research (literature review) are:

- Identify the characteristics of VoIP implementations and identify communication network parameters that are detrimental to a VoIP call.
    - This research will allow for an understanding of VoIP calls and the network parameters are detrimental to the quality of voice experienced.

The research will focus on delay, packet loss and jitter and from this the suitability of a data-set can be confirmed.

- Realize call quality standards and current quality measuring/prediction methodologies
    - Call standards will be important for analysing the results of the experiment along with understanding the current methodologies so it can be understood how the networks perform in relation.
- Research artificial neural networks and associated learning algorithms
    - Artificial neural network methodology and the suitability of the architecture and methodology will be researched. Gradient Descent and Levenberg-Marquardt will be researched in heavy detail in order to justify hypothesis. Comparative metrics will be discovered.
- Investigate the supplemental use of neural networks with Voice over IP
    - The current use of neural networks to supplement VoIP will be researched in order to outline the suitability, advantages and drawbacks of the implementations

The objectives that will be met through primary research are:

- Construct an artificial neural network model in MATLAB
    - This artificial neural network will be designed in MATLAB in order to be trained by both learning algorithms detailed. It will have three input nodes. In order to maintain credibility between results both learning algorithms will train the same network design.
- Train artificial neural network with suitable data-sets using the Gradient Descent Algorithm
    - Gradient Descent will be used to train the neural network in order to adjust synaptic weights on the neurons. To maintain credibility between the results both training algorithms will use identical data sets.
- Train artificial neural network with suitable data-set using the Levenberg-Marquardt algorithm
    - Levenberg-Marquardt will be used to train the neural network in order to adjust synaptic weights on the neurons. To maintain credibility between the results both training algorithms will use identical data sets.
- Test & Validate the network through use of suitable test-data
    - In order to ensure the network is operating as expected it must be tested with suitable test data. This must be done for both learning methods.
- Collect results from the network after training by each learning algorithm
    - In order to statistically analyze the results of the experiment many statistics must be gathered. The mean opinion scores generated by the network will be gathered, along with the mean squared errors, gradient and the impact of delay, packet loss and jitter on the MOS.
- Statistically analyse results
    - Plot graphs in MATLAB using appropriate analysis techniques in order to illustrate the results gathered and prove in relation to the hypotheses and research question. When relevant reference PESQ results also. Relationships will be identified based on plotted graphs. MATLAB will be used for further statistical analysis.
- Conclude results and write final report

- On completion of both primary and secondary research the project must be documented. The project will be detailed entirely and conclusions will be drawn from the hypotheses and research question.

1.4 Hypotheses

Detailed below are the hypotheses derived from the full literature review:

H1: When the Levenberg-Marquardt algorithm is used to train the artificial neural network the results produced will be closer to the ITU-T PESQ MOS than the Gradient Descent trained artificial neural network.

Justification:

Beale & Demuth (1997) have indicated that the Levenberg-Marquardt Algorithm is more efficient at training a neural network than the Gradient Descent method. Based on research by Escalante & Malki (2006) Levenberg-Marquardt produces more dependable results when compared with Gradient Descent. Overall, Levenberg-Marquardt is predicted to have more accurate results and achieve them in a much quicker timescale making practical application of a tool developed using the Levenberg-Marquardt algorithm more likely.

H2: The Levenberg-Marquardt algorithm will train the artificial neural network in a smaller period of time than the Gradient-Descent method.

Justification:

The results of the experiment are expected to be that the Levenberg-Marquardt algorithm is able to train the neural network to predict call quality between 10-100 times faster than the Gradient Descent. (Demuth & Beale, 1996)

H3: The Levenberg-Marquardt algorithm will train the artificial neural network more efficiently than the Gradient Descent method.

Justification:

The learning rate of the Levenberg-Marquardt training is expected to be higher than the Gradient-Descent method based on research by Chester (1993) and Demuth & Beale (1996) in the following section highlighting the enhancements and increased speed of the algorithm at adjusting weights closer to the minimum at a faster rate than Gradient-Descent.

H4: The results produced by the artificial neural network in both cases will be less accurate than the results of the ITU-T PESQ model.

Justification:

Based on research by Sun (2008), Sun & Ifeachor (2008), Radhakrishnan & Larijani (2010), Radhakrishnan et al (2010) it is expected that the artificial *neural network models will not be able to predict Mean Opinion Score to the same accuracy as PESQ.*

H5: The number of neurons in the hidden layer will affect the ability of the networks when training with Backpropagation.

Justification:

Research by Demuth & Beale(1996) show that too many neurons and too few neurons in the hidden layer will affect the ability of Backpropagation methods to train a network effectively. Radhakrishnan et al (2010) noted an increased ability of their neural network model when more hidden neurons were present, thus it is expected that the network will perform better with more hidden neurons. It is predicted that either five or six hidden neurons will be the optimum configuration.

1.5 Report Structure

This section details a brief overview of the remaining chapters of this report. Detailed below are the literature review and the methods section.

1.5.1 Literature Review

Chapter two of this report is the literature review which will investigate through secondary research the areas of voice over IP communications, current methodologies and artificial neural networks.

The research begins with an overview of general information in each area before narrowing the research to the relevant areas of the fields relating to the objectives.

Research regarding voice over IP will begin with an investigation into the protocol along with detrimental communication network parameters to voice over IP calls. Research will also be investigated in the areas of current prediction or measuring methods along with standards expected for IP calls and PSTN calls. Finally the research will conclude with an investigation into artificial neural networks and Backpropagation training. Each area will be discussed and evaluated in detail; the outcome will generate conclusions relevant to the primary research or research methodology.

1.5.2 Methods

Chapter three of this document highlights the primary method to be used in the project. The experimental approach that is being utilised will be highlighted and justified. The outline of each stage of the experiment will be introduced and discussed. This section will conclude with an overview of the remaining tasks to be completed in order to meet the primary research goals, conclude upon the research question and hypotheses and deliverable requirements.

1.5.3 Presentation of Results

This penultimate section displays the results found in the primary research method which were achieved through the use of a simulation environment. Through the results collected it can be seen how each learning algorithm performs and adapts based on the architecture of the network. A comparison between benchmark paradigm PESQ and the neural network is also shown and discussed in this section.

1.5.4 Summary & Conclusions

This final section details and discusses the results obtained within the primary research method and evaluates them against the hypotheses and research question. This section also looks at the strengths and weaknesses of the report and details possible future research.

2.0 Literature Review

This section aims to address the areas discussed in background in section 1.1 and the objectives in section 1.2 by providing technical detail and research based discussion. This literature review will provide the basis of knowledge by which to understand and undertake primary research.

As identified in section 1.2; the literature review will complete the following objectives and provide a framework of knowledge upon which to complete the primary research. The objectives were identified as:

- Identify the characteristics of Voice over IP
- Identify the characteristics of artificial neural networks and associated learning algorithms
- Identify call quality standards and current measurement methodologies including neural network methodologies

2.1 Investigation into the characteristics of Voice over IP

This section will discuss the characteristics of Voice over IP and the challenges faced by VoIP such as detrimental network parameters.

2.1.1 Overview of Voice over IP

Voice over IP is a suite of protocols that runs over UDP/IP in order to transmit voice traffic over existing communication networks that transport data traffic. This is referred to as network convergence. Due to the "real-time" nature of voice traffic it is unsuitable for use transmission over TCP/IP due to excessive overheads included in the packet headers and the reliability of packet retransmission redundant in real time traffic. VoIP runs on UDP/IP with RTP, this is more suitable for VoIP as identified by Durkin (2003), due to the lack of reliability (retransmission of lost packets) and also the time-stamping, reordering and multiplexing provided by the combination of UDP/IP with RTP which gives all the features beneficial to VoIP which TCP/IP also includes without the overhead.

Voice over IP differs from the traditional PSTN network in that VoIP is a Packet-Switched technology and the PSTN is Circuit-Switched transmission. In Circuit-Switching analog phones connect to the PSTN and the circuit generated is a constant, direct link to the end point of the conversation. In packet switching transmission the voice is digitised and sent across a shared network medium as packets (Durkin, 2003). Coviello (1979) illustrates that in order to transfer voice across a packet switched network it would be a battle of minimising delays and ensuring adequate bandwidth in order to compete with the quality of the PSTN.

H.323 is a signalling standard and protocol suite used in Voice over IP. It is the most commonly deployed standard for voice transmission over packet switched networks (Wallace, 2009). It provides generation and exchange of control information to establish, monitor and terminate connections between end points. H.323 also recommends registration, audio codecs and video codecs. (International Telecommunications Union, 2009b)

The real-time nature of Voice over IP traffic it is essential to monitor the perceived call quality for users as network parameters can cause breaks in speech, scrambled words and call drops. Voice over IP traffic faces many challenges in ensuring high call quality; this is detailed by Furuya et al (2003) as delay, packet loss and jitter. These network parameters have been identified as having a large, detrimental impact on perceived speech quality and that monitoring voice quality based on these parameters would be possible.

Coder/Decoders (codecs) have a large impact on the perceived quality of speech across the network (or indeed any sort of audio/video format). Goode (2002) describes this as the process of encoding a voice into digital format (bits) for placing on the wire and transmitting and decoding before the listener hears it (perhaps even on an analog telephone). Many codecs are in operation on VoIP networks. This study will focus on two commonly deployed codecs (SPEEX and G711a).

2.1.2 Investigation into the impact of delay in Voice over IP

Network delay or end-to-end delay is the total time it takes for packets to arrive at the intended destination. End-to-end delay takes into account various different types of delay. These various delays are as follows (Cisco Systems, 2006a):

- Processing Delay
    - The time it takes for a router to receive the packet on an input interface, process it and put it in the routers output queue.
- Queuing Delay
    - The time a packet spends in a routers queue waiting to be forwarded. This is directly affected by the queuing method in place, i.e FIFO, WFQ or CBWFQ and the priority of the traffic.
- Serialization Delay
    - The delay value from placing the bits onto the physical transport medium
- Propagation Delay
    - The time it takes to cross the transport medium.
- De-jitter Delay
    - The time held in the de-jitter buffer to ensure packets are not played with varying timescales

Delay in communication networks, specifically when affecting voice over IP traffic occurs due to voice processing, packetization and propagation problems (Mansousos 2005).

The optimum voice quality recommendation made by the ITU suggests that delay should not exceed 150ms in order to ensure optimum voice quality like experienced on PSTN calls (International Telecommunications Union, 2003). Delays up until 150ms will only cause minor impairments on call quality. (Yammato & Beerends, 1997)

Delays of 150-400ms are acceptable in network planning but calls will be impacted as delay increases (International Telecommunications Union, 2003). Delays of over 400ms are unacceptable for most applications (Cisco Systems, 2006a).

When delay values reach greater than 30ms echo cancellation must be used in order to maintain the quality of the call. Packet network delay would cause for users to perceive false gaps in the conversation where they believe the other user is no longer talking, this may cause both users speech to overlap (Kampichler & Gooschav, 2001).

2.1.3 Impact of Jitter on Voice over IP

Jitter or the variation of delay is the difference in time between subsequent packet arrival times as defined by the (Demichilis & Chimento, 2002). Due to the real time nature of voice traffic sequential packets arrival time should remain as constant as possible.

Figure 2.1 shows the sender generating voice traffic at a constant flow of evenly spaced traffic; due to adverse network conditions the flow of voice packets becomes uneven (Cisco Systems, 2006b).

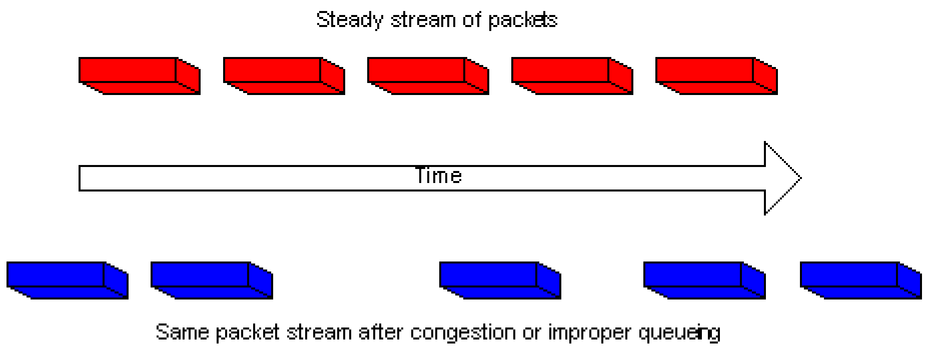


Figure 2.1 – Jitter Diagram (Cisco Systems, 2006b)

Jitter is caused by sequential packets taking different (longer) or more congested routes through the network to the destination than the previous packet. Zheng at el (2001) also note that jitter is also a result of buffering in packet switched networks, and is oftentimes highly increased during periods of high bursty traffic. The impact of jitter can be lessened by prioritizing voice packets using queuing mechanisms or by compensating for jitter using play out delay buffers which are buffers designed to delay the playback process enough to compensate for the eventual late arrival of packets (Jelassi et al, 2009).

Zheng et al (2001) show that jitter affects traffic in two ways; negative jitter will result in packet loss, where as sequential positive jitter will result in increased network delay.

Recommendations by Cisco Systems (2004) state that to achieve optimum call quality on a packet switched network the delay variation of sequential voice packets should not exceed 30 ms, when jitter values reach above 50 ms dejitter buffers will struggle to compensate for the variations of arrival time resulting in the speech flow becoming disrupted.

The impact of Jitter on voice traffic would cause the flow of speech to break or even data loss entirely as packets arriving too late will be dropped (Durkin, 2003).

Duysburgh et al (2001) show the devastating effect jitter has on call quality in Figure 2.2 below. This shows as the average variation time between packets increases call quality drops dramatically. Also illustrated in Figure 2.3 is the relationship between increased jitter and packet loss against call quality.

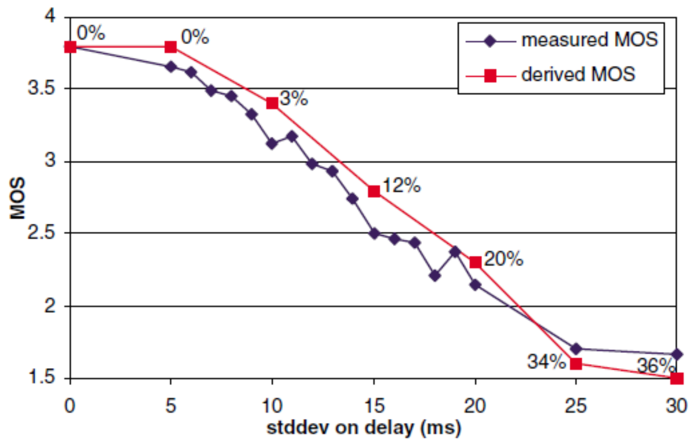


Figure 2.2 – Jitter Versus Mean Opinion Score (Duysburg et al, 2001)

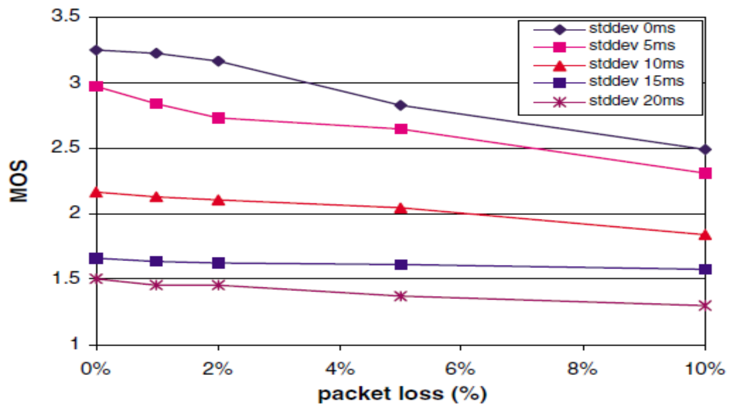


Figure 2.3 – Jitter & Packet Loss Versus Mean Opinion Score (Duysburgh et al 2001)

2.1.4 Analysis of Packet Loss on Voice over IP

Packet loss in communications networks generally results in packet retransmission by IP and for data transfer this is acceptable as they are not time-sensitive applications such as FTP. This is not possible for real-time applications such as voice and video. Data loss in a VoIP call cause gaps in audio wave playback (Wallace, 2009).

Network conditions that cause packet loss are:

- Link Congestion
    - When the bandwidth is contested for and not sufficient for the volume of traffic being carried
- Delay variation (Jitter)
    - When the delay between packets becomes too great for playback to be possible the packets are dropped.
- Unstable topologies

- Links and other networking equipment are prone to interference and downtime. This may cause call drops and missing packets.

Packet loss can be compensated for by packet loss concealment technologies which generate replacement packets but can only compensate for a small amount of data loss (Ding & Gourban, 2003b). Packet loss can be reduced by ensuring link bandwidth is sufficient and prioritizing time sensitive traffic (Durkin, 2003).

The relationship of packet loss relative to voice quality is shown in Figure 2.4 below. This shows direct correlation to perceived voice quality to a high degree (Ding & Gourban, 2003b).

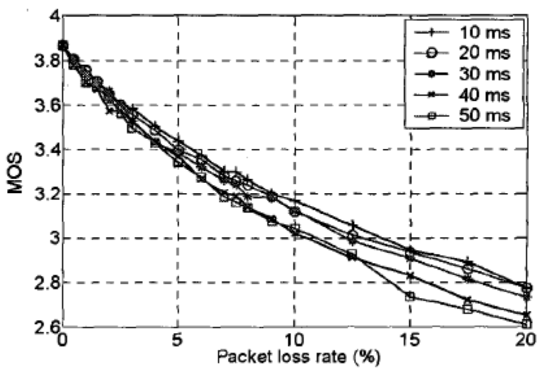


Figure 2.4 – Packet Loss Versus Percieved Voice Quality (Ding & Gourban, 2003b)

2.1.5 Conclusion

The nature of IP networks has made it hard for Voice over IP providers to meet quality expectations customers have based on PSTN experience. Service providers need the ability to monitor network parameters in real-time in order to assess the level of service being provided (Chi & Womack, 2009). Various sources outlined the importance of delay, jitter and packet loss to voice quality and the close relationship of these parameters will be used in the primary research method. The understanding of voice communications generated has shown the importance of maintaining a high level of service on par with the PSTN.

2.2 Investigation into call quality standards and current measurement methodologies

2.2.1 Overview of standards and prediction methods

The ITU-T is the governing body for telecommunications transmission. Many standards have been put in place for the measuring of voice quality by the ITU-T such as PESQ (Perceptual Evaluation of Speech Quality) and the E-Model. The ITU-T also makes recommendations on transmission quality factors that will have a direct impact on the perceived quality of the call.

Testing can be considered as either subjective or objective. Subjective testing uses human participants to measure the voice quality. Subjective testing while accurate is also the most expensive and time consuming way of determining voice quality. Objective methods are computational models to assess the quality of a voice transmission in an attempt to automate and mimic the accuracy of subjective testing. Objective methods can be further clarified as either intrusive or non-intrusive. Intrusive methodologies make use of a reference signal in order to determine voice quality where as non-intrusive methods do not. The use of a reference signal makes instrusive methods unsuitable for real time traffic monitoring applications using QoS parameters, but presents an optimum benchmark. (Abereghi et al, 2008)

2.2.2 Mean Opinion Score

Mean Opinion Score is an internationally recognised (Sun & Ifeachor, 2006) numerical scale for rating the quality of a voice signal. The ITU-T recommends an industry standard scale in ITU-T Recommendation P.800 for use in voice quality measurement in both subjective and objective methodologies. This is shown in Table 2.1 below (International Telecommunications Union, 1996)

Table 2.1 – Mean Opinion Score (International Telecommunications Union, 1996)

Call Quality	**Mean Opinion Score**
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

The P.800 Recommendation not only recommends how the MOS should be measured but also the condition of the recording and listening environments for the source signal such as room size, noise level and reverberation time. (International Telecommunications Union, 1996)

The Mean Opinion Score is tested by using appropriate industry standard sentences which encompass various sounds used in the language being tested. An example of an English language test sentence is “Nowadays, a chicken leg is a rare dish.” which encompasses long and short vowels along with hard and soft sounds. (Wallace, 2009)

In subjective testing Mean Opinion Scores are generally carried out by focus groups of sixteen or more people. Each participate rates the quality of the transmission between one & five. The results of these tests are averaged to formulate the Mean Opinion Score (Vlahdimitropolous & Kastros, 2007).

PSTN calls generally have a high Mean Opinion Score of 4.0. This shows that in order to achieve like-PSTN call quality it is necessary to monitor the detrimental communication network parameters to ensure the high standards that have been set by PSTN are met (Yamato & Beerends, 1997).

2.2.3 ITU-T Perceptual Evaluation of Speech Quality

PESQ or the Perceptual Evaluation of Speech Quality is a methodology defined by the ITU-T in recommendation P.862. PESQ is an objective method of determining voice quality specifically developed for testing end-to-end voice quality taking into account network conditions.

PESQ compares the original signal to a degraded signal after passing through the communications network. Two error parameters are computed and combined to give the objective listening quality PESQ score. These two values are the average asymmetrical disturbance value and average disturbance value (International Telecommunications Union, 2001).

Subjective testing methods were used in order to benchmark the PESQ model for accuracy in measuring human perceived speech quality. In order to do this, subjective tests were concurrently concluded alongside the PESQ testing. Using both the results, the coefficient correlation was shown to be 0.935 showing that PESQ accurately predicts perceived speech quality (International Telecommunications Union, 2001).

PESQs output metric is directly comparable to MOS with no further calculation required. PESQ's output ranges from -0.5 to 4.5 which is on the same scale as MOS (International Telecommunications Union, 2001).

PESQ can be used as a tool to highlight performance problems, codec development and network planning (Manjunath, 2009). PESQ is often used in development in order to obtain reference Mean Opinion Score values due to its effectiveness at mirroring subjective tests (Paglierani and Petri, 2007). Although PESQ is considered to be the most accurate means of objective testing it does has its limitations such as the lack of real-time monitoring and subjective testing still remains the most accurate way to assess transmission quality. (Manjunath, 2009)

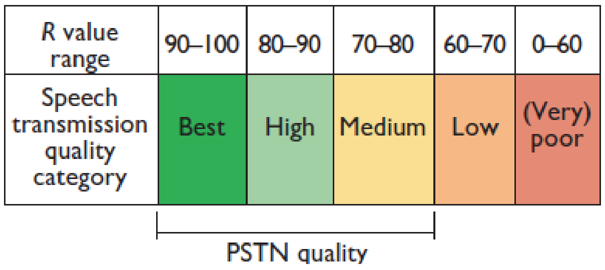
2.2.4 ITU-T E-Model

The ITU-T E-Model is a transmission planning tool for telecommunication networks such as PSTN and converged IP networks carrying Voice over IP traffic. It is a computational model which is effective in predicting the perceived speech quality on the medium. It is a non intrusive methodology as it does not inject any traffic on to the network or interrupt network flow in any way (Ding & Gourban, 2003). The primary output of the E-Model is the R-Value, which is the rating of the perceived speech quality. The R-Value is expressed between 50 and 100 as shown in Table 2.2 below in relation to Mean Opinion Score (International Telecommunications Union, 2009a).

Table 2.2 – R-Value Versus Mean Opinion Score (International Telecommunications Union, 2009)

***R*-value (lower limit)**	**MOS_{CQE} (lower limit)**	**User satisfaction**
90	4.34	Very satisfied
80	4.03	Satisfied
70	3.60	Some users dissatisfied
60	3.10	Many users dissatisfied
50	2.58	Nearly all users dissatisfied

Janssen et al (2002) show in Figure 2.5 the correlation between the R Value and PSTN quality in the figure below showing calls must fall between 70 & 100 in order to achieve PSTN-like quality.



R value range	90–100	80–90	70–80	60–70	0–60
Speech transmission quality category	Best	High	Medium	Low	(Very) poor

PSTN quality

Figure 2.5 – R-Value to Speech Quality (Janssen et al, 2002)

The R-Value is converted to a Mean Opinion Score using equations found in ITU recommendation x. R-Value is calculated using the follow equation in Figure 2.6:

Figure 2.6 – R-Value Calculation (International Telecommunications Union, 2009a)

= Signal To Noise Ratio
= Speech Impairments such as quantization noise, speech loudness
= Delayed Speech Impairments such as send/receive echo, absolute delay
= Equipment impairment such as packet loss, delay and jitter
= Advantage factor. This optional value is used to express the medium the voice is travelling across (International Telecommunications Union, 2009a).

A practical implementation of the E-Model on a VoIP network was investigated by Carvalho et al (2005). The results were shown to be accurate but the E-Model's practicality is slighted due to its high computational costs (Radhakrishnan et al, 2010).

Ding & Goubran (2003a) investigate the ability of the extended E-Model to predict call quality based on packet loss, delay and jitter. Their model successfully predicts the Mean Opinion Score (after conversion from R value) to ±0.10 of the target output value when only taking into account one of the network parameters but does not perform as well when combining all three parameters. This finding shows the need to establish the relationship between non-linear variables.

Sun & Ifeachor (2006) present a hybrid model based on PESQ & the E-Model taking into account packet loss, delay and jitter and results showed high correlations to target mean opinion score output.

2.2.5 Neural Network Model To Measure Voice Quality

Artificial neural networks have been applied to measuring the human perceived mean opinion score due to their advanced pattern recognition ability. Two key researchers in this area have been identified.

Sun (2004) illustrates the use of an artificial neural network to predict mean opinion score non-intrusively based on gender, codec and loss rates. The ability of neural networks to calculate the mean opinion score using the relationship between the four input non-linear variables and one output non-linear variable of VoIP traffic was demonstrated using a three layer feed-forward neural network design as shown in Figure 2.7 below. This research highlights the ability of a three layer design to calculate the mean opinion score.

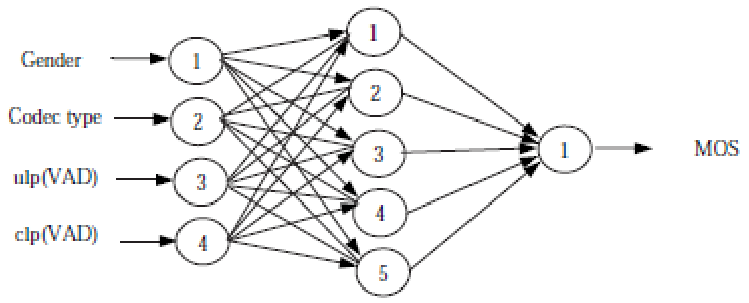


Figure 2.7 – Artificial Neural Network Model (Sun, 2004)

The model was trained using Gradient Descent to adjust the synaptic weights. This was shown to be almost as accurate as PESQ with a correlation coefficient of 0.967 and an error rate of 0.12 during training. The accuracy was decreased slightly when the test data was applied with a correlation coefficient of 0.952 and average error of 0.15. This model demonstrates the ability of artificial neural networks to compete with PESQ prediction quality but leaves room for optimization to bring the results closer to current standards. This research also highlights Gradient Descent as an effective way to train artificial neural networks to predict the mean opinion score.

Radhakrishnan & Larijani (2010) use a random neural network model to measure voice quality taking into account packet loss, delay and jitter. The random neural network model also shows the ability of a three layer design in order to calculate the mean opinion score for perceived voice quality. The network weights have been trained using the Gradient Descent learning algorithm. This research once again highlights the suitability of the Gradient Descent method to predict perceived speech quality.

As shown in Figure 2.8 below, Radhakrishnan & Larijani (2010) use the three layered model to compute the non-linear relationship between three input variables and one output variable, highlighting the suitability of this model to predict perceived voice quality.

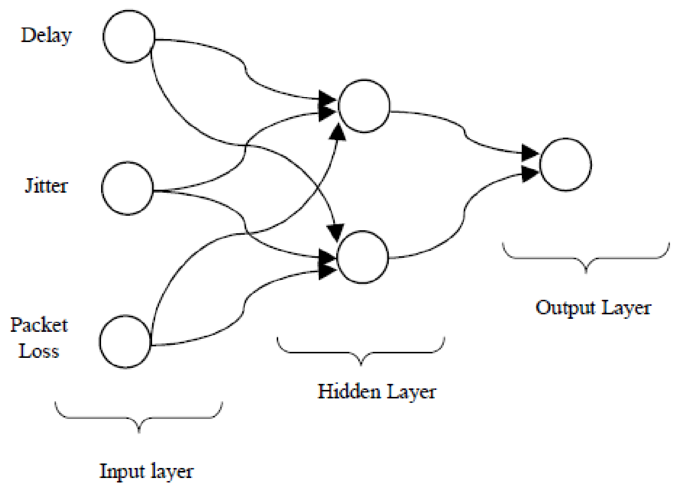


Figure 2.8 – Three layer feedforward network (Radhakrishan & Larijani, 2010)

The RNN Model results are presented in Figure 2.9 below. The figure shows the relationship of the PESQ MOS to the RNN MOS. It is shown that the random neural network does match to the results shown by PESQ to a degree, sometimes closer than others.

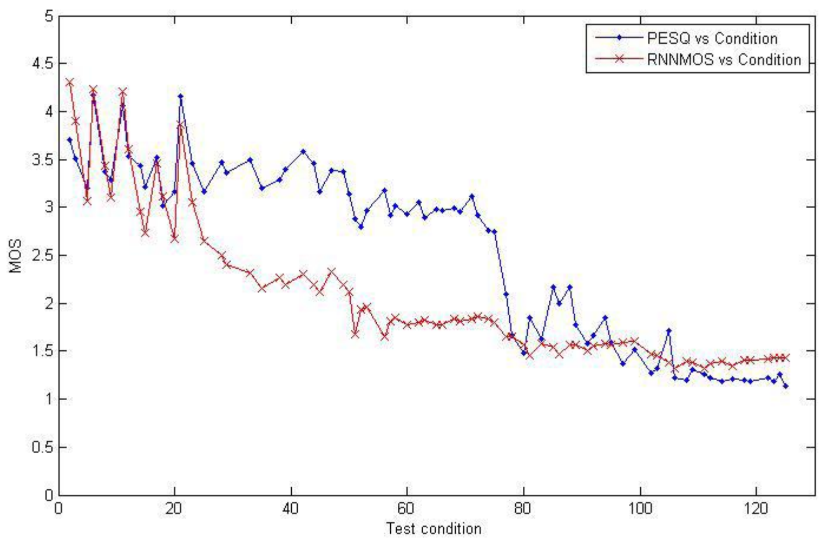


Figure 2.9 – Random Neural Network Versus PESQ (Radhakrishnan & Larijani, 2010)

Radhakrishan et al (2010) improve upon the random neural network model by increasing the input variables from three to four, to now include codec and also by testing models with a varying number of hidden layer neurons as show in Figure 2.10 below.

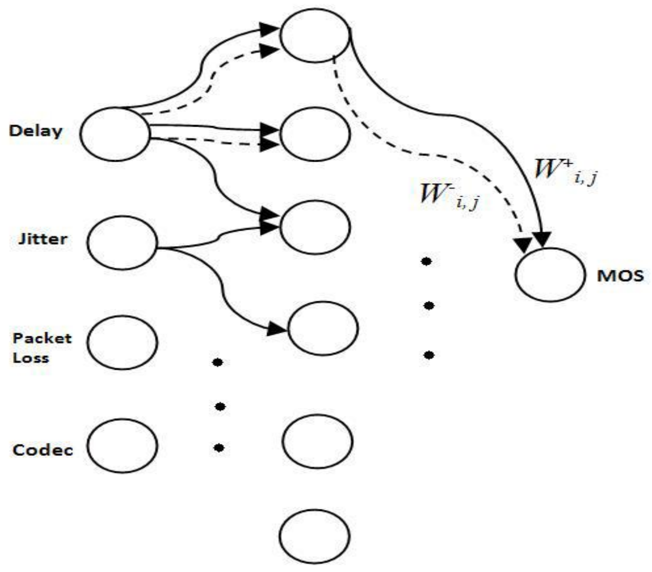


Figure 2.10 - Four input model with a variable hidden layer (Radhakrishnan et al, 2010)

The results in Figure 2.11 do show improved correlation to the PESQ Mean Opinion Score due to the increased input of codec and number of hidden neurons. It is shown that the Mean Opinion Score of PESQ and the RNN model correlate to the highest degree when using five or six neurons at the hidden layer as shown in Figure 2.12. This research shows the importance of the number of hidden neurons to accurately predict the MOS.

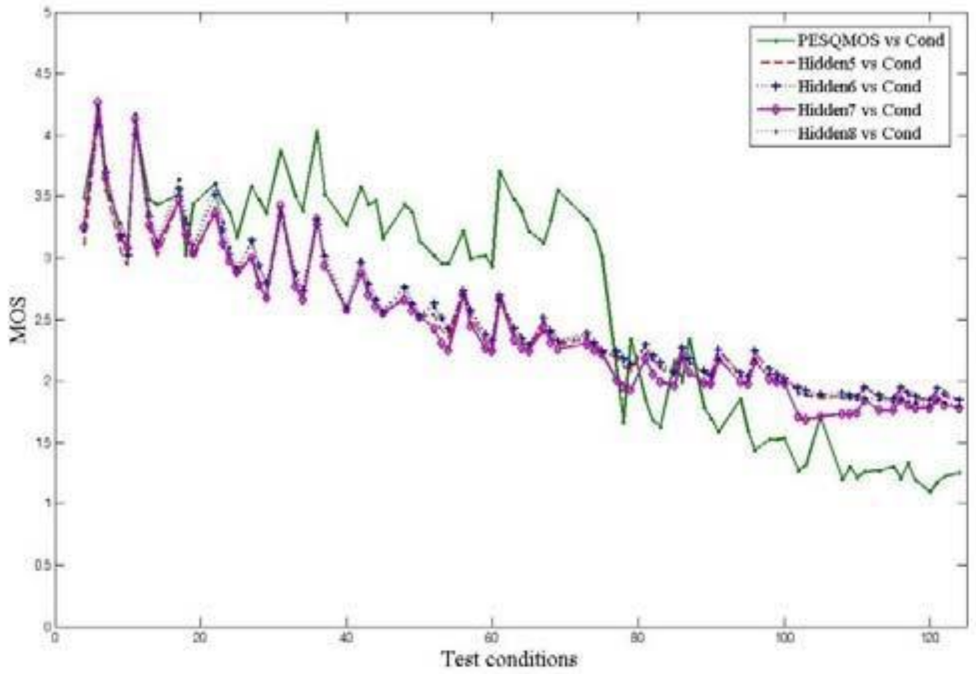


Figure 2.11 – All hidden layer models Versus PESQ (Radhakrishnan et al, 2010)

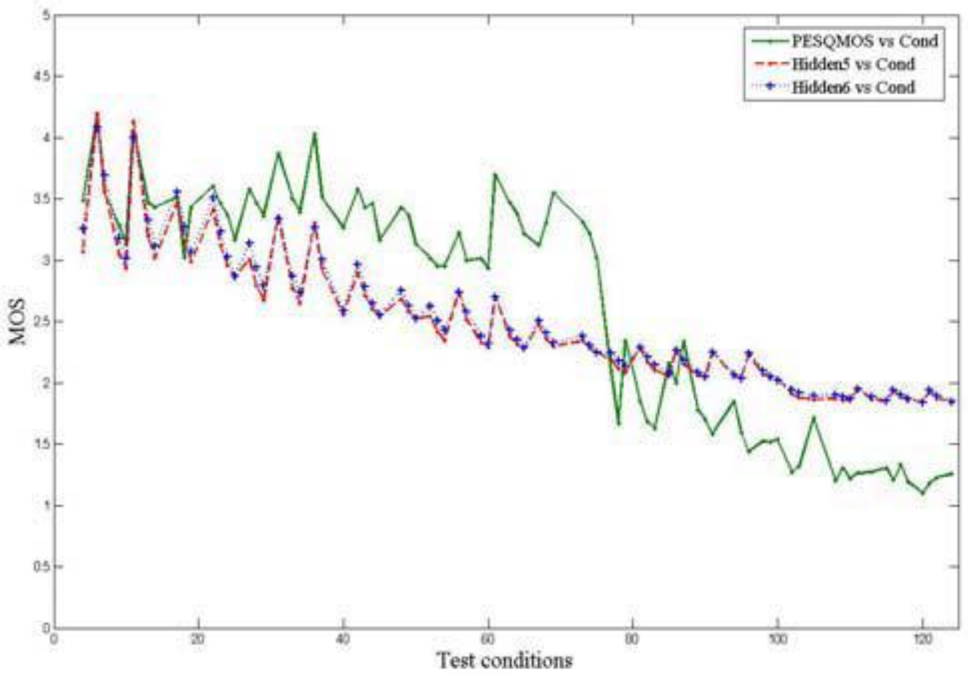


Figure 2.12 – Five and six hidden neurons Versus PESQ (Radhakrishnan et al, 2010)

2.2.6 Conclusion

The investigation into Mean Opinion Score has shown the effectiveness in using this metric to describe perceived voice quality. Using MOS as our output will allow for training of the artificial neural network model to be developed using accurate results from a methodology widely accepted (International Telecommunications Union, 1997) (Sun & Ifeachor, 2006). From analysis (Carvalo et al, 2005) it can be concluded that the E-Model's practical use is computationally heavy (Radhakrishnan et al, 2010) and could be improved upon using a trained artificial neural network to monitor MOS. The PESQ methodology although has been shown to be inappropriate for live traffic has been identified as a benchmark for the method used in the primary research (Sun & Ifeachor, 2006). Also identified was the importance of the hidden neurons to a neural network model to provide stability. Radhakrishnan et al (2010) identifies five or six to be the most stable for the model applied.

2.3 Investigation into artificial neural networks and associated learning algorithms

2.3.1 Overview of artificial neural networks

Artificial neural networks are typically applied to problems of pattern recognition Hagan et al (1996). This is due to their ability to compute non-linear variables and their relationships (Chester, 1993). A typical artificial neural network has a feedforward structure in which all layers are connected to the one above (Dreyfus, 2005). Other possible topologies of artificial neural network are totally connected and mixed connectivity networks (Chester, 1993).

In an artificial neural network each interconnected node has a weight on the connection (synaptic weight) that is adjusted during the training process in order to get closer to the target value (Chester, 1993).

Each neuron has an activation value and when that value is met the individual neuron generates an output signal. Each neuron is only aware of the signal it receives and knows not of other neuron's internal workings (Dreyfus, 2005).

Neural network design has six phases as shown by Hagan et al (2010) the phases are:

- Phase 1 – Data Collection
- Phase 2 – Create the neural network topology
- Phase 3 – Initialise network synaptic weights
- Phase 4 – Use an appropriate algorithm to train the network
- Phase 5 – Validate the network with suitable testing data
- Phase 6 – Use the network

This process as highlighted by Hagan et al (2010) gives a structured development cycle for the model in the primary research.

2.3.2 Overview of artificial neural network training

A supervised artificial neural network must be taught much like a human being in order to make informed and knowledgeable decisions. Using a suitable data set the neural network is taught which response it should get from the inputs given. This methodology requires the outside computation of results in order to train the network. Supervised learning is used when it is possible to accurately predict the output and is ideal for problems of benchmarking. During the learning process the weights are adjusted from comparing the computed output to the target output. Once the training is complete the network weights are frozen and the network can be shown new input data and make decisions based on the weights that were adjusted during the learning phase. (Chester, 1993).

An unsupervised artificial neural network is a model that has no target outputs available. The weights of the synapses are adjusted based on the input data and the algorithms in the learning rules which perform clustering operations (Hagan et al, 1996).

2.3.3 Backwards Propagation of Errors

Backwards propagation of errors (Backpropagation, backprop) is a paradigm used to adjust synaptic weights of neurons in multilayer feedforward networks during the learning phased in supervising training. It was originally developed by Bryson and Ho (1969) but not brought to light until the work of Rumelhart et al (1985). Neilsen (1989) considers Backpropagation to be the most used neural network architecture.

When using Backpropagation learning, networks are typically of a multilayer design including one or more hidden layers of neurons which are neither input nor output layers (Neilsen, 1989). Backpropagation training algorithms produce better or worse results based on the number of neurons in these hidden layers. When the number of hidden neurons is too little the network will not be able to solve the problem as displayed in the Figure 2.13 below. When the number of hidden neurons is excessive the network is unable to generate a smooth function as demonstrated within Figure 2.14 below (Demuth & Beale, 1992).

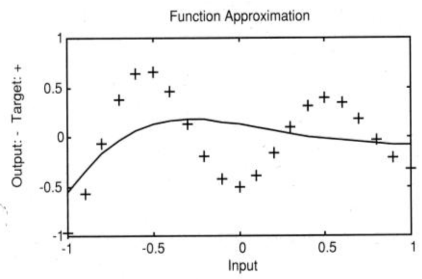


Figure 2.13 – Underfitting of gradient (Demuth & Beale, 1992)

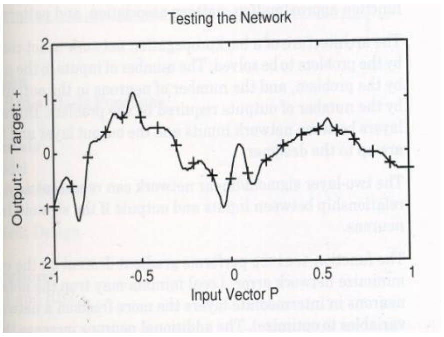


Figure 2.14 – All hidden layer models Versus PESQ (Demuth & Beale, 1992)

Bong et al (2010) point out that when an artificial neural network has too many hidden neurons in it's hidden layer it will just repeat the data set, and when it has too few it won't be able to make a generalized rule and it will plot inaccurately.

In Backpropagation synaptic weights are initialised as random weightings and are adjusted based on the error between the target output and the actual output (Chester, 2003). This is called the correction signal and this is propagated back through the layers of the network (Neilsen, 1989). Backpropagation uses a generalised delta rule from gradient descent as shown in Figure 2.15 below (Chester, 1993). The weight (*w*) for the *i*th input of neuron *j* is modified based on the following where is the learning rate and t_j and y_j are target and actual outputs respectably with x_i being the *i*th input:

Figure 2.15 – Gradient Descent Weight Adjustment (Chester, 1993)

The rule in Figure 2.16 used to calculate the correction signal (E) where t_j and y_j are target and actual outputs (Liu et al, 2008) is given as:

–

Figure 2.16 – Gradient Descent Error (Liu et al, 2008)

The error surface in Backpropagation is a representation of the inputs and all the possible states of the network and is displayed in peaks and craters. The Backpropagation methodology uses gradient descent to find the lowest possible minimum (crater) on the error surface (Neilsen, 1989)

The strength of Backpropagation learning rules lie in the ability for them to create complex multidimensional mapping (Chester, 1993).

Backpropagation is limited when the using a large training set. It will take many sweeps through the training data in order to full learn the patterns present and can be time and resource heavy (Chester, 1993).

Backpropagation algorithms all perform gradient descent in which weights are moved in the opposite direction of the error gradient, in order to reach the global minimum error using the delta rule. (Demuth & Beale, 1996) Gradient Descent and Levenberg-Marquardt are two Backpropagation algorithms that perform gradient descents to reach the global minimum.

Gradient descent has been named the simplest of all Backpropagation algorithms by Demuth & Beale (1996), as all it does to calculate the global minimum and adjust weights is use the Gradient Descent rules, but it is often times slow to converge (Khare et al, 2008). When the steps toward the minimum occur too quickly due to a high learning rate Gradient Descent oscillates around the global minimum, producing a ‘zigzag’ effect. A solution to this is a variable learning rate (Demuth & Beale, 1996). A danger when using gradient descent is that it may get trapped in the local-minimum rather than the global-minimum (Guo & Gelfand, 1991). When the minimum error gradient is reached the network has been trained and can be tested with fresh input data (Hagan et al, 2010).

Levenberg-Marquardt in comparison is named the fastest of all Backpropagation algorithms by Demuth. Although fast to converge, Wilamowski and Yu (2010) outline that Levenberg-Marquardt Backpropagation may be limited to small and medium sized neural networks as the Jacobian matrix (the way Levenberg-Marquardt generalises the gradient) would become too large. Levenberg-Marquardt is also limited to only feedforward networks and cannot be used for fully connected networks (Wilamowski & Yu, 2010). Levenberg-Marquardt requires more memory than typical Gradient Descent (Demuth & Beale, 1996). The weight update of Levenberg-Marquardt also has the goal of reaching the global minimum and is shown in Figure 2.17. The rule is stated by Hagan & Menhaj (1994) as:

Figure 2.17 – Levenberg-Marquardt Weight Adjustment (Hagan & Menhaj, 1994)

With J being the Jacobian matrix of derivatives of each error to each weight, is a scalar value for learning rate and e is the error vector. Using the large value the expression approximates gradient descent. Training is complete when the minimum error gradient occurs and the network can then be tested with fresh input data (Hagan & Menhaj, 1994)

2.3.4 Conclusion

Artificial neural networks by nature of pattern recognition have been identified as an ideal paradigm to predict the quality experienced by end users. An investigation into the Backpropagation technique demonstrated the ability of BP trained artificial neural networks to establish patterns with non-linear variables such as the relationship between delay, jitter, packet loss and mean opinion score. The analysis of the learning algorithms Gradient Descent and Levenberg-Marquardt allowed an understanding to be developed of the shortcomings of both techniques along with the suitability of both for the primary research.

3.0 Methods

The purpose of this section is to present further detail of the primary research used in this report. It will address why the primary method for this project was the most appropriate, specific details on how it will be carried out and the future stages involved in the completion of this report.

3.1 Primary Research Methodology and Justification

The primary research method of this report will be obtained through an experimental evaluation technique. The primary research will be conducted using the MATLAB software package with the Neural Network Toolbox. This package has been chosen due to an extensive list of features for neural networks including an extensive implementation of many different Backpropagation algorithms (Demuth & Beale date), evidence of previous successful use in related experiments (Radhakrishnan, Radhakrishnan et al 2010), powerful mathematical and statistical analysis abilities (Moler, 2004) and large amounts of reference texts available for guidance on operation.

The experimental nature of this method in a simulation package such as the MATLAB Neural Network Toolbox allows for repeatability and validation of results, and reduces the resources such as time and cost needed to implement a practical implementation such as hardware costs, focus group costs (subjective testing for voice quality) and time of developing a practical solution (Almeroth et al 2003). The experimental method is suitable for testing the two learning algorithms as they are already available within the MATLAB Neural Network toolbox, programming these in a high level coding environment would be impractical for addressing the primary of the project.

Alternative software was identified during the literature review by Sun (2004) with the Stuttgart Neural Network Simulation package. This package was used to train the artificial neural network in Sun’s solution but deemed unsuitable for the primary research of the project due to no support for Levenberg-Marquardt optimizations for Backpropagation and a lack of literature.

Due to hardware and time constraints a suitable data-set was identified during the literature review. This data set from Radhakrishnan et al (2010) encompasses delay, packet loss and jitter as inputs with Mean Opinion Scores generated by the ITU-T PESQ model as target outputs.

3.2 Design of the Primary Research Method

The design of this experiment will closely mirror the six principle stages of neural network design introduced by Hagan et al during the literature review. The six stages are defined as (note network means neural network):

- Stage 1 - Data Collection
- Stage 2 - Network Topology Design
- Stage 3 - Weight Initialisation
- Stage 4 - Training the network
- Stage 5 - Validation
- Stage 6 - Use the network

Stages 1-3 will be mirrored exactly as in the process; where as the final stages in the approach will differ. The training stage will have two iterations, as will the validation stage.

The original final stage will not be used in this process but will be replaced by a data analysis stage. The updated stages for the experiment are as follows:

- Stage 1 – Data Collection
- Stage 2 – Network Topology Design
- Stage 3 – Weight Initialisation
- Stage 4 – Train the network with Gradient Descent
- Stage 5 – Validation of the network based on Gradient Descent training
- Stage 6 – Train the network with Levenberg-Marquardt
- Stage 7 – Validation of the network based on Levenverg-Marquardt training
- Stage 8 – Statistical analysis and comparison of results

This outline clearly defines each stage and is an expansion on the original methodology proposed, specifically iterations of the training and validation stages to allow for a comparison of the training algorithms. The removal of the use the network field is due the experimental and comparative nature of the project.

3.3 Data Collection

As identified in the literature review from various sources, the network parameters that have a large impact on the perceived quality of a voice over IP call are:

- Delay
- Jitter
- Packet Loss

Generation of this type of data for voice calls would be time consuming and require large amounts of hardware to perform real world scenario tests.

The most efficient means for measuring the quality of a voice transmission was identified as Mean Opinion Score (International Telecommunications Union, 1996)(Sun & Ifeachor, 2006). ITU-T PESQ was identified in the literature review as the best means for benchmarking the results from the primary research method by various sources for reasons of accuracy of the PESQ models results when compared to other methods (Manjuath, 2009) and subjective benchmarking would be costly and inefficient on time (Aberghi et al, 2008) and the ITU-T requirements of samples and room sizes would are out with the available resource pool.

Due to the constraints on resources and time suitable data sets from Radhakrishnan et al (2010) have been identified for use in training the Neural Network. They consist of two input codecs (SPEEX & G711a) which will ensure that the trained network can perform for real world data with more than one codec type, the required input values of delay, jitter and packet loss with target output based on ITU-T PESQ model. The target output also doubles as a benchmark for the data analysis stage of the experiment where the ability of the trained network's can be tested against this metric. The data sets are included within the appendix.

Delay values are set as 0, 50, 100, 150 and 200 ms which is the maximum delay for acceptable call quality according to the ITU-T standard identified within the literature review.

Jitter has possible values of 0, 3, 5, 7 and 10ms. Packet loss has possible values between 0 and 10%.

3.4 Network Topology Design

In the network design phase various elements were taken into consideration, the first being the number of inputs and outputs. For each input variable a neuron is required on the input layer. This requires three input neurons (delay, jitter and packet loss). The target output for the network is to provide a mean opinion score, so one output neuron is required. The proposed network topology is shown below in Figure 3.1, consisting of three input neurons, a variable hidden layer and one output neuron.

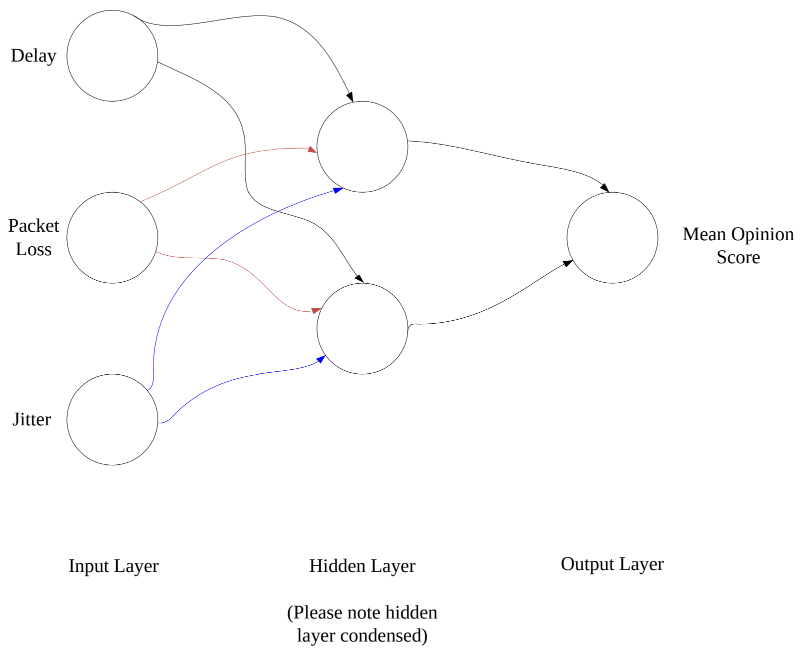


Figure 3.1 – Proposed feedforward design

Feedforward Backpropagation trained networks require at least one hidden layer of neurons for further stability of the outputs. From the literature review it was seen that too few hidden neurons would cause under-fitting of the gradient and too many hidden neurons would cause over-fitting (Demuth & Beale, 1997).

In a similar experiment by Radhakrishnan et al (2010) various configurations of hidden neurons were shown. The results from hidden layers with five or six neurons proved to be the most stable for a network with three inputs.

Thus to ensure that the most stable number of hidden neurons is identified, various network architectures will be explored, evaluated and compared for both learning algorithms and codecs. Training, testing and validation will occur within the MATLAB Neural Network Toolbox. The various network architectures are described within Table 3.1 – 3.4 below. The SPEEX experiments are the primary results with G711a being used to insure that the results from the SPEEX experiment are credible. Only one hidden layer will be tested with the G711a data.

Table 3.1 – Gradient Descent Experiments (SPEEX)

SPEEX Codec		
Algorithm	No. of Hidden Layers	No. of Hidden Neurons
Gradient Descent	1	1
		2
		3
		4
		5
		6
		7
		10
	2	5

Table 3.2 – Levenberg-Marquardt Experiments (SPEEX)

SPEEX Codec		
Algorithm	No. of Hidden Layers	No. of Hidden Neurons
Levenberg-Marquardt	1	1
		2
		3
		4
		5
		6
		7
		10
	2	5

Table 3.3 – Gradient Descent Experiments(G711a)

G711a Codec		
Algorithm	No. of Hidden Layers	No. of Hidden Neurons
Levenberg-Marquardt	1	1
		2
		3
		4
		5
		6
		7
		10

Table 3.4 – Levenberg-Marquardt Experiments(G711a)

G711a Codec		
Algorithm	No. of Hidden Layers	No. of Hidden Neurons
Levenberg-Marquardt	1	1
		2
		3
		4
		5
		6
		7
		10

3.5 Simulation Phases

Before simulation can begin the proposed topologies must be built in MATLAB Neural Network toolbox. This will be done using a series of scripts to construct the topologies and begin the various simulation stages. The proposed topologies will then be initialised with weights on the synapses. In Backpropagation these weights are random values initially (Hagan et al, 1996). The network will then be trained using the input data, some of which will be reserved for validation of the network (Hagan et al, 2010). Each learning algorithm and network will make use identical training data to preserve validity of the results.

To update the model introduced in section 3.2 Steps 4-7 will be expanded to include the two different codecs this is displayed below:

- Stage 4 – Train the network with Gradient Descent
    - Stage 4.1 Train the network using the SPEEX data set
    - Stage 4.2 Train the network using the G711a data set
- Stage 5 – Validation of the networks based on Gradient Descent training
- Stage 6 – Train the network with Levenberg-Marquardt
    - Stage 6.1 Train the network using the SPEEX data set
    - Stage 6. Train the network using the G711a data set
- Stage 7 – Validation of the network based on Levenberg-Marquardt training

When the network training has been complete in each stage the network must be validated using the reserved training data. This will be used to verify the network is operating as expected. Once training, testing and validation is complete the MOS values along with MSE values are output to files inside MATLAB. The number of epochs, time taken to train and the gradient are all found on the simulation tool.

3.6 Data Analysis Stage

The final stage of the design method adapted from the original methodology proposed in the literature review by Hagan et al (2010) is the new data analysis stage that replaces using the network. This stage will involve collecting, analysing and plotting various statistics using MATLAB. This will be used to prove or disprove the hypotheses outlined in section 1.3.

The statistics primarily being assessed are as follows:

- Mean Opinion Score – Gradient Descent, Levenberg-Marquardt & PESQ
    - The primary comparison of the learning algorithms will be the output mean opinion scores generated by the artificial neural networks when trained with the two different learning methodologies. This graph will be plotted for Gradient Descent and Levenberg-Marquardt trained networks and PESQ used to benchmark.
- Training time
    - The time measured in seconds that it takes for the algorithm to complete training, reach the global minimum or the maximum number of iterations.
- The Gradient

    - The correlation of coefficient between the different learning algorithms, hidden neurons against the PESQ values can be measured in order to compare results with one figure for each algorithm.

- Epochs
    - The number of epochs (iterations of the network) it takes for it to reach a useable state (global minimum) will be used to measure efficiency of the trained network.
- Mean squared errors
    - The mean squared errors (overall, training, testing and validation) give another metric to compare performance of both the learning algorithms which is the difference between the target output and the actual output at each stage.
- The perceived effect of packet loss, delay and jitter as measured by the proposed model
    - The differences in the way the implementations may be seen in plotting these graphs. The most accurate implementation will be a good measure on how these parameters affect the perceived voice quality of transmissions.

3.7 Hypothesis Testing

All experiments will be run in order to prove or disprove the fully justified hypotheses detailed in Section 1.4. Hypotheses will be tested in the following manner:

H1: When the Levenberg-Marquardt algorithm is used to train the artificial neural network the results produced will be closer to the ITU-T PESQ MOS than the Gradient Descent trained artificial neural network.
This can be tested directly by plotting the results (MOS Values) of the most efficient network configurations between the two learning algorithms, benchmarked against PESQ. Two more metric useful for the testing of this hypothesis are the MSE values which show the error difference during each stage of the simulation and the Gradient which shows the distribution of the results.

H2: The Levenberg-Marquardt algorithm will train the artificial neural network in a smaller period of time than the Gradient-Descent method.
This can be measured directly in the period of time it takes to train the network in seconds and also in the number of epochs required.

H3: The Levenberg-Marquardt algorithm will train the artificial neural network more efficiently than the Gradient Descent method.
This can be judged using the number of epochs required to train the network.

H4: The results produced by the artificial neural network in both cases will be less accurate than the results of the ITU-T PESQ model.
This will be proven or disproven alongside H1 using the plotted results for MOS and the perceived effects of packet loss, delay and jitter.

H5: The number of neurons in the hidden layer will affect the ability of the networks when training with Backpropagation.
This will be disproven using the plot graphs once again, taking into account the number of neurons and the number of hidden layers.

4.0 Presentation of Results

The purpose of this section is to display the results of the primary research method. These will presented with a short explanation of each experiment followed by discussion of the results found.

4.1 Results with SPEEX Codec

The following results are derived from training the artificial neural network with the data generated by the SPEEX codec.

4.1.1 Training ANN with a maximum of 1000 Epochs and One Hidden Neuron with the SPEEX codec

For the initial experiment the number of epochs will be limited to 1000 as reflected inside the scripts in appendix. The hidden layer of neurons will be restricted to one hidden neuron and the codec in use will be SPEEX. The data set can be found in the appendix.
Exemplar data will be used with 70% for training, 15% for testing and 15% for validation.

Gradient Descent

As displayed in Table 4.1 during the training process the algorithm reached the maximum number of epochs for the experiment which is set to 1000. This is the default number of epochs. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 16 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.1139 which significantly higher than 0.

Table 4.1 – Gradient Descent with One Hidden Neuron (SPEEX)

Epochs:	1000 (maximum)
Time:	16 seconds
Performance Training: (MSE)	0.1139
Performance Validation:	0.1057
Performance Testing:	0.1221
Overall performance:	0.1139
Gradient:	0.0633

Figure 4.1 below shows the correlation of the Mean Opinion Scores output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with only one hidden neuron is not able to successfully mimic the results obtained in PESQ and this becomes even more evident toward the final samples where there is a vast difference between the real PESQ scores and the predictions by the neural network. The neural network oscillates the plots between 2.8 and 3.2 consistently for the length of the simulation.

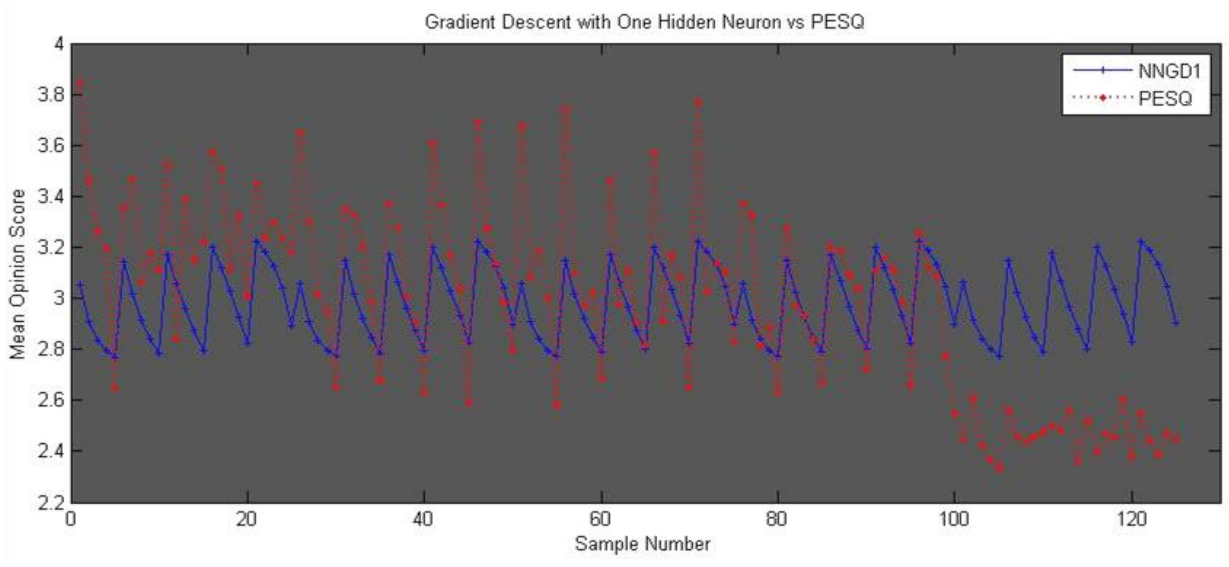


Figure 4.1 – Gradient Descent with One Hidden Neuron vs PESQ

Levenberg-Marquardt

Table 4.2 below shows the statistics from the initial experiment with the Levenberg-Marquardt learning algorithm when training an artificial neural network with one hidden neuron. Levenberg-Marquardt trains the artificial neural network within 8 epochs. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.1104 which is large number compared to 0.

Table 4.2 – Levenberg-Marquardt with One Hidden Neuron (SPEEX)

Epochs:	8
Time:	<1 second
Performance Training: (MSE)	0.1290
Performance Validation:	0.0463
Performance Testing:	0.0898
Overall performance:	0.1104
Gradient:	0.000210

Figure 4.2 that follows shows the correlation of the Mean Opinion Scores output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with only one hidden neuron is not able to identify a pattern between the input variables and the target output, and this becomes even more evident toward the final samples where there is a vast difference between the real PESQ scores and the predictions by the neural network. The neural network "zigzags" the plots between 2.7 and 3.3 consistently for the length of the simulation

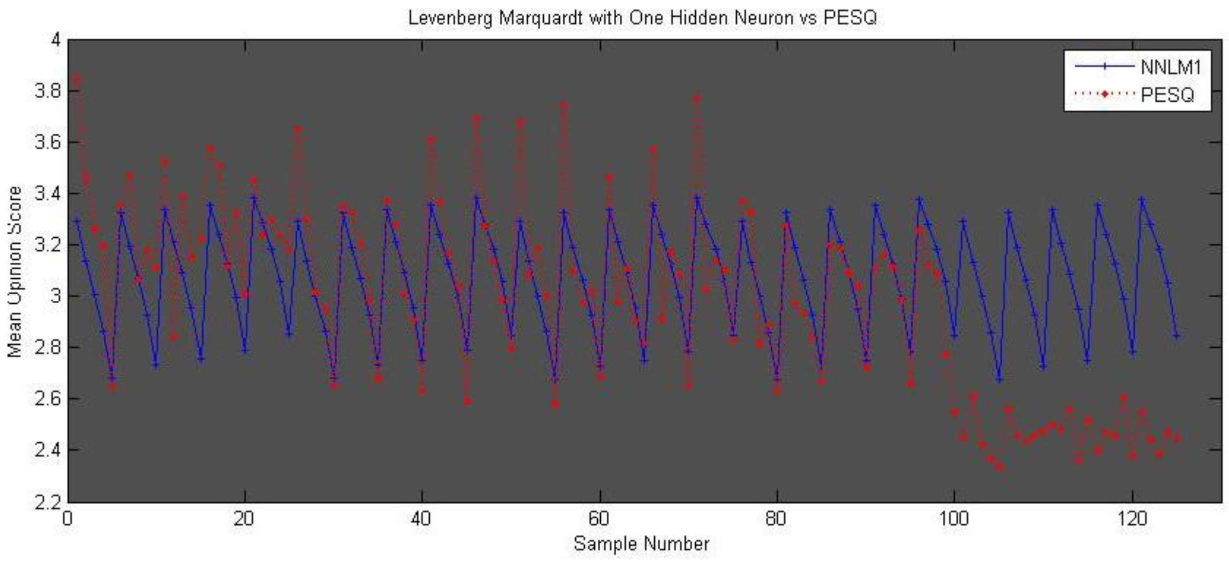


Figure 4.2 – Levenberg-Marquardt with One Hidden Neuron vs PESQ

Conclusion

As found during the literature review the performance of Gradient Descent was not only slower in this initial experiment but also more inefficient. The difference between the number of epochs required to train the network using the algorithms is vast, with Gradient Descent taking 992 more iterations than Levenberg-Marquardt. The network was expected to perform poorly with only one hidden neuron and these results match well with the hypotheses.

4.1.2 Training ANN with a maximum of 1000 Epochs and Two Hidden Neurons with the SPEEX codec

During this experiment the number of epochs will be limited to 1000 as reflected inside the scripts in appendix. The hidden layer of neurons will be restricted to two hidden neurons and the codec in use will be SPEEX.
Exemplar data will be used with 70% for training, 15% for testing and 15% for validation.

Gradient Descent

As displayed in Table 4.3 during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 15 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.1038 which significantly higher than 0.

Table 4.3 – Gradient Descent with Two Hidden Neurons (SPEEX)

Epochs:	1000 (maximum)
Time:	15 seconds
Performance Training: (MSE)	0.1078
Performance Validation:	0.1163
Performance Testing:	0.0729
Overall performance:	0.1038
Gradient:	0.0414

Figure 4.3 below shows the correlation of the Mean Opinion Scores output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with two hidden neurons is predict the MOS obtained in PESQ and this becomes even more evident toward the final samples where there is a vast difference between the real PESQ scores and the predictions by the neural network.

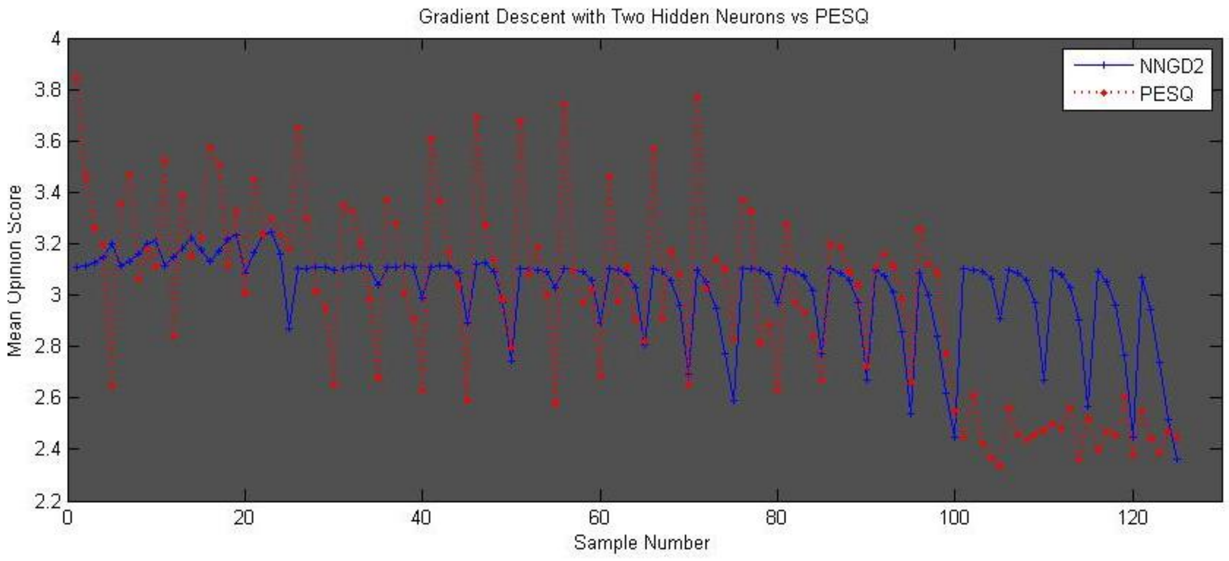


Figure 4.3 – Gradient Descent with Two Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.4 below shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with two hidden neurosn. Levenberg-Marquardt trains the artificial neural network within 15 epochs. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0392 which is large number compared to 0.

Table 4.4 – Levenberg-Marquardt with Two Hidden Neurons (SPEEX)

Epochs:	15
Time:	<1 second
Performance Training: (MSE)	0.0329
Performance Validation:	0.0610
Performance Testing:	0.0465
Overall performance:	0.0392
Gradient:	0.0414

Figure 4.4 that follows shows the correlation of the Mean Opinion Scores output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with two hidden neurons is able to vaguely identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line The neural network "zigzags" the plots between 2.7 and 3.3 consistently for the length of the simulation

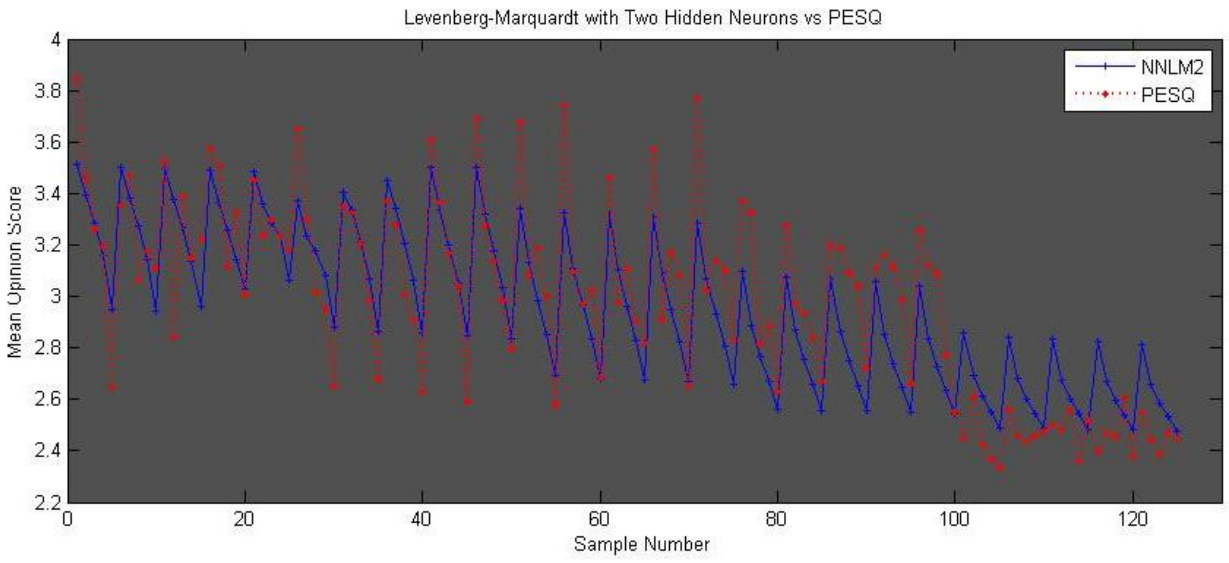


Figure 4.4 – Levenberg-Marquardt with Two Hidden Neurons vs PESQ

Conclusion

The results of this experiment are similar to those found in the first experiment. Gradient Descent once again showed poor results and efficiency, where as Levenberg-Marquardt showed a great increase with a small increase in the number of neurons with the overall MSE being reduced vastly and also forming a vague relationship to the PESQ MOS line plot.

4.1.3 Training ANN with a maximum of 1000 Epochs and Three Hidden Neurons with the SPEEX codec

During this experiment the number of epochs will be limited to 1000 as reflected inside the scripts in appendix. The hidden layer of neurons will be restricted to three hidden neurons and the codec in use will be SPEEX.

Exemplar data will be used with 70% for training, 15% for testing and 15% for validation

Gradient Descent

Table 4.5 shows that during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 14 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0715 which significantly higher than 0.

Table 4.5 – Gradient Descent with Three Hidden Neurons (SPEEX)

Epochs:	1000 (maximum)
Time:	14 seconds
Performance Training: (MSE)	0.0681
Performance Validation:	0.0652
Performance Testing:	0.0930
Overall performance:	0.0715
Gradient:	0.0333

Figure 4.5 below shows the relation between the Mean Opinion Scores output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with three hidden neurons is vaguely able to predict the MOS values and follows a slightly similar line to PESQ but is not accurate enough to produce reliable results.

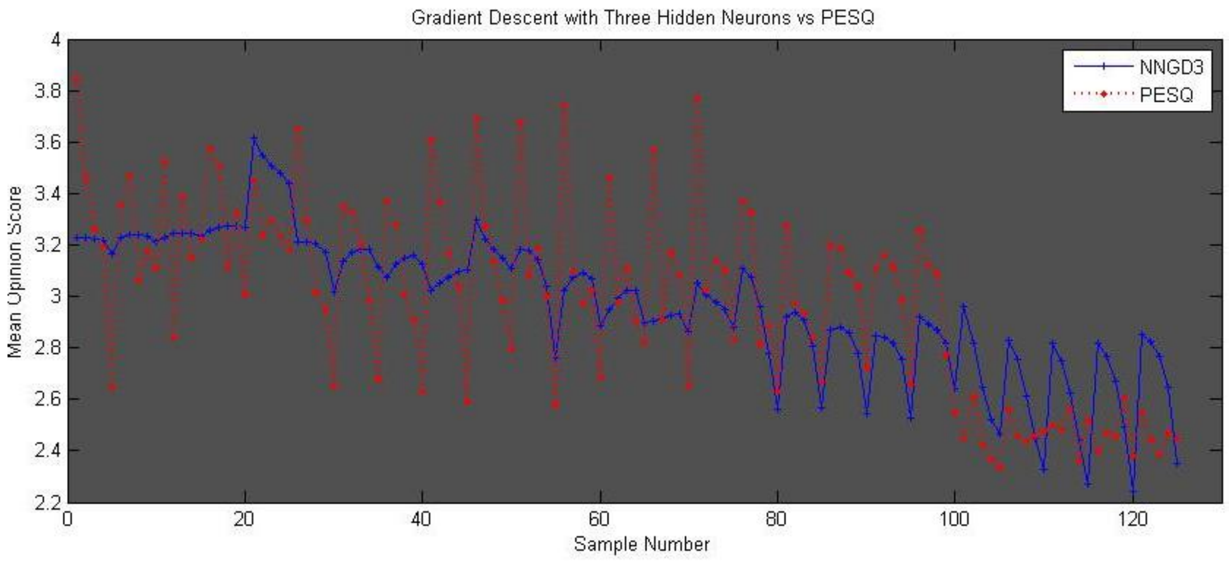


Figure 4.5 – Gradient Descent with Three Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.6 below shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with two hidden neurons. Levenberg-Marquardt trains the artificial neural network in 12 epochs. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0187 which is showing significant progress toward 0.

Table 4.6 – Levenberg-Marquardt with Three Hidden Neurons (SPEEX)

Epochs:	12
Time:	<1 second
Performance Training: (MSE)	0.0202
Performance Validation:	0.0138
Performance Testing:	0.0172
Overall performance:	0.0187
Gradient:	0.0122

Figure 4.6 that follows shows the correlation of the Mean Opinion Scores output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with three hidden neurons is able to closely identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. The accuracy of this line is high compared to previous experiments.

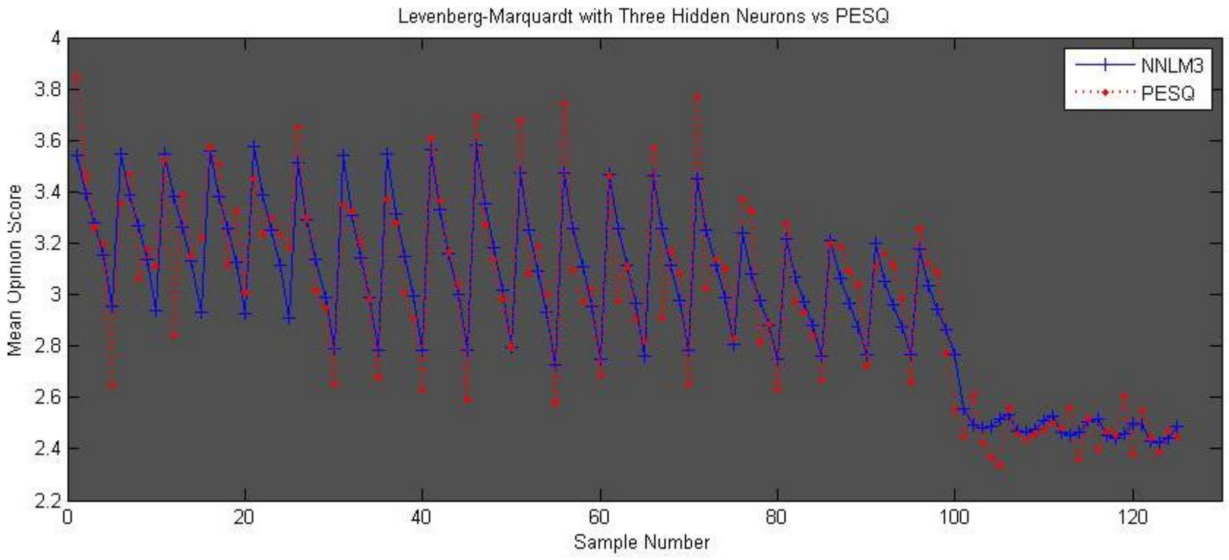


Figure 4.6 – Levenberg-Marquardt with Three Hidden Neurons vs PESQ

Conclusion

The results of this experiment are similar to those found in previous experiments. Gradient Descent once again showed poor results and efficiency, where as Levenberg-Marquardt showed a great increase with a small increase in the number of neurons with the overall MSE being reduced vastly and also forming a solid relationship to the PESQ MOS line plot.

4.1.4 Training ANN with a maximum of 1000 Epochs and Four Hidden Neurons with the SPEEX codec

Gradient Descent

As displayed in Table 4.7 during the training process the Gradient Descent algorithm reached 889 epochs. This means that the network reached the global minimum with Gradient Descent for the first time. The network took 12 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0717 which significantly higher than 0.

Table 4.7 – Gradient Descent with Four Hidden Neurons (SPEEX)

Epochs:	889
Time:	12 seconds
Performance Training: (MSE)	0.0669
Performance Validation:	0.0622
Performance Testing:	0.1033
Overall performance:	0.0717
Gradient:	0.0323

Figure 4.7 below shows the relation between the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with four hidden neurons is vaguely able to predict the MOS values and follows a slightly similar line to PESQ but is not accurate enough to produce reliable results. At approximately sample 100 the results become more distant from the targets.

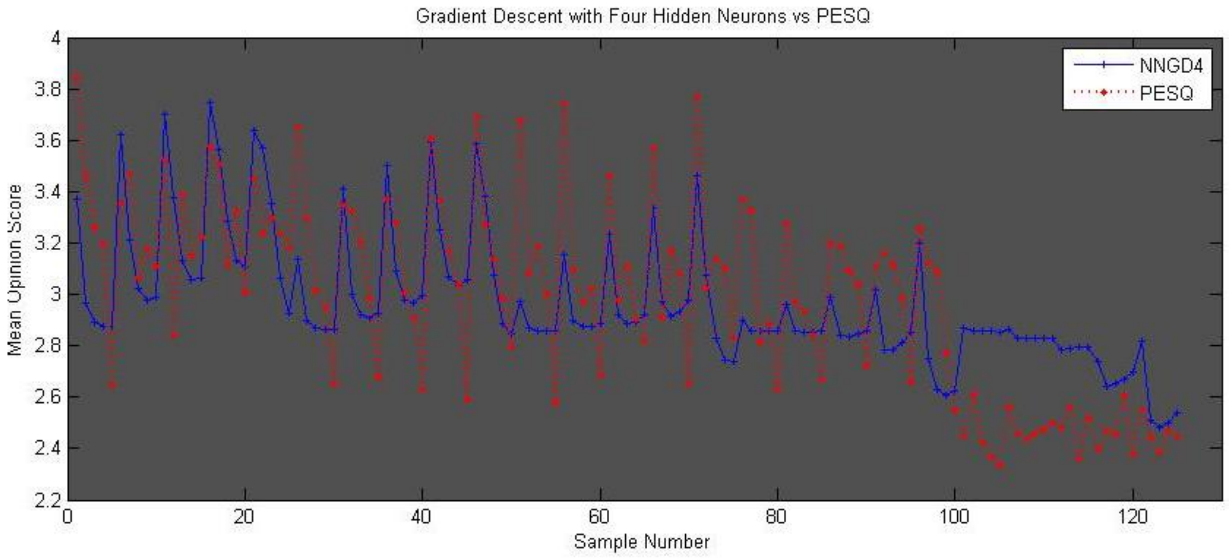


Figure 4.7 – Gradient Descent with Four Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.8 below shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with four hidden neurons. Levenberg-Marquardt trains the artificial neural network within 24 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0186 which is following the trend of approaching closer to 0.

Table 4.8 – Levenberg-Marquardt with Four Hidden Neurons (SPEEX)

Epochs:	24
Time:	<1 second
Performance Training: (MSE)	0.0162
Performance Validation:	0.0300
Performance Testing:	0.0184
Overall performance:	0.0186
Gradient:	0.000880

Figure 4.8 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with four hidden neurons is able to ac by the line descending along with the PESQ line. The artificial neural network predicted Mean Opinion Score values follow the peaks and lows of the PESQ scores to a high degree but fall short in some occasions as presented at approximately sample numbers 55 and 75.

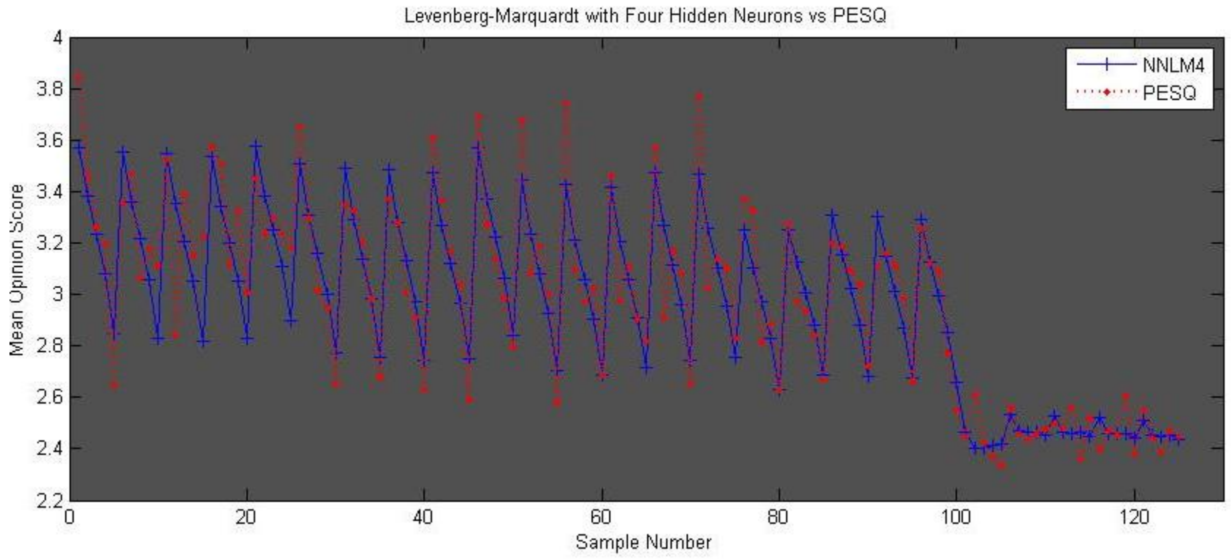


Figure 4.8 – Levenberg-Marquardt with Four Hidden Neurons vs PESQ

Conclusion
With the increment to four hidden neurons for both paradigms, significant improvement was found for both Levenberg-Marquardt and Gradient Descent. Levenberg-Marquardt's relationship to the PESQ benchmark has improved once again and is producing results that could be considered viable. Gradient Descent, although improving, is still falling short.

4.1.5 Training ANN with a maximum of 1000 Epochs and Five Hidden Neurons with the SPEEX codec

For this experiment the number of epochs will be limited to 1000 as reflected inside the scripts in appendix. The hidden layer of neurons will be restricted to five hidden neurons and the codec in use will be SPEEX. Exemplar data will be used with 70% for training, 15% for validation and 15% for testing.

Gradient Descent

Table 4.9 shows that during the training process the Gradient Descent algorithm once again reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum during training, the validation and testing phases begun. The network took 14 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0597.

Table 4.9 – Gradient Descent with Five Hidden Neurons (SPEEX)

Epochs:	1000 (maximum)
Time:	14 seconds
Performance Training: (MSE)	0.0541
Performance Validation:	0.0771
Performance Testing:	0.0481
Overall performance:	0.0597
Gradient:	0.0409

Figure 4.9 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with Five hidden neurons is able to identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. The artificial neural network predicted Mean Opinion Score values follow the peaks and lows of the PESQ scores to a certain extent but begin to deteriorate from approximately sample number 95.

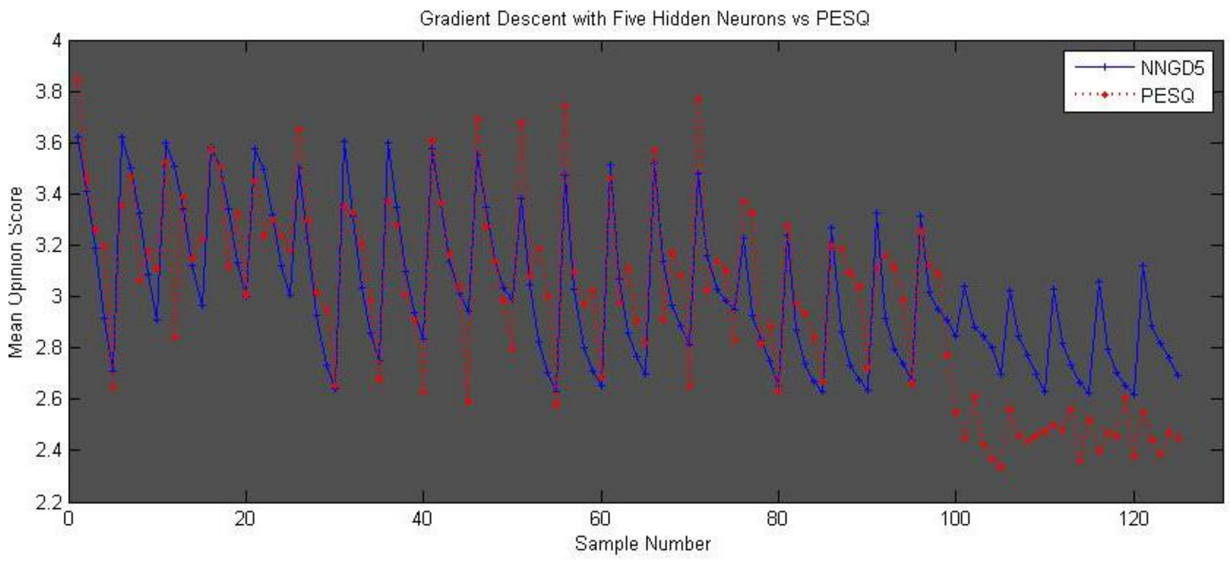


Figure 4.9 – Gradient Descent with Five Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.10 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with five hidden neurons. Levenberg-Marquardt trains the artificial neural network within 16 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0160 which is following the trend of approaching closer to 0 with each increment of the hidden layer.

Table 4.10 – Levenberg-Marquardt with Five Hidden Neurons (SPEEX)

Epochs:	16
Time:	<1 second
Performance Training: (MSE)	0.0171
Performance Validation:	0.0093
Performance Testing:	0.0217
Overall performance:	0.0160
Gradient:	0.0300

Figure 4.10 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with Five hidden neurons is able to accurately identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. The artificial neural network predicted Mean Opinion Score values follow the peaks and lows of the PESQ scores to a great extent.

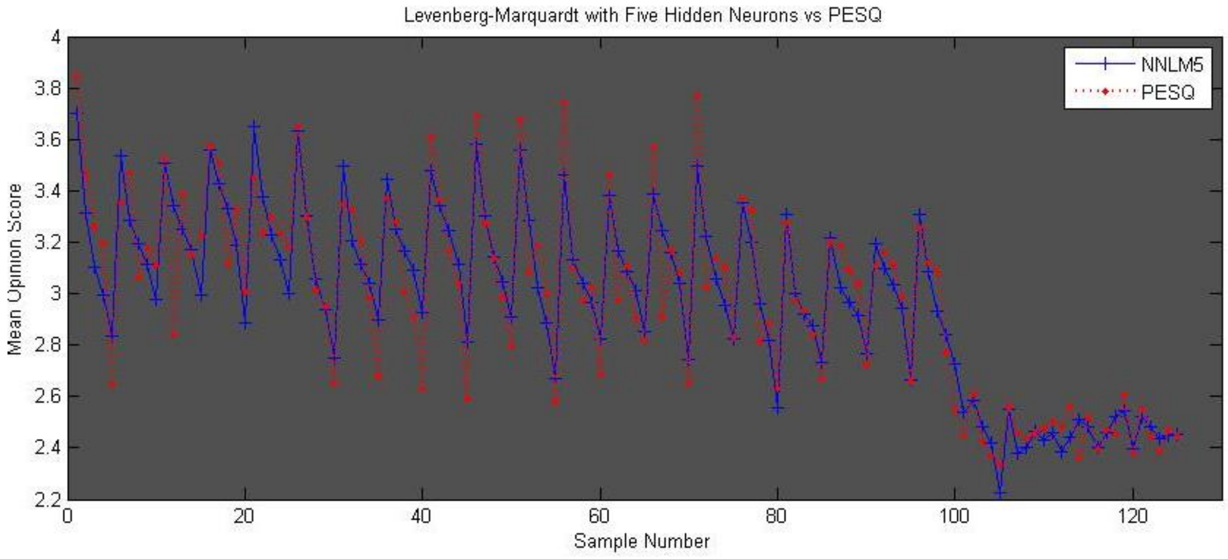


Figure 4.10 – Levenberg-Marquardt with Five Hidden Neurons vs PESQ

Conclusion

With the increment to five hidden neurons for both paradigms, significant improvement was found for both Levenberg-Marquardt and Gradient Descent. Levenberg-Marquardt's relationship to the PESQ benchmark has improved once again and is producing results that could be considered viable. Gradient Descent, although improving, is still falling short of being a viable paradigm with exaggerated peaks and lows and high error values.

4.1.6 Training ANN with a maximum of 1000 Epochs and Six Hidden Neurons

The number of epochs will be limited to 1000 as reflected inside the scripts in appendix. The hidden layer of neurons will be restricted to six hidden neurons.
Exemplar data will be used with 70% for training, 15% for validation and 15% for testing.

Gradient Descent

Table 4.11 shows that during the training process the Gradient Descent algorithm continued the trend of reaching the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum during training, the validation and testing phases begun. The network took 18 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0478.

Table 4.11 – Gradient Descent with Six Hidden Neurons (SPEEX)

Epochs:	1000 (maximum)
Time:	18 seconds
Performance Training: (MSE)	0.0457
Performance Validation:	0.523
Performance Testing:	0.0527
Overall performance:	0.0478
Gradient:	0.0424

Figure 4.11 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with six hidden neurons is unable to identify a pattern between the input variables and the target output.. The artificial neural network predicted Mean Opinion Score values vaguely follow the peaks and lows of the PESQ scores but many of these plots are exaggerated and become extremely inaccurate in the upper quadrant of samples.

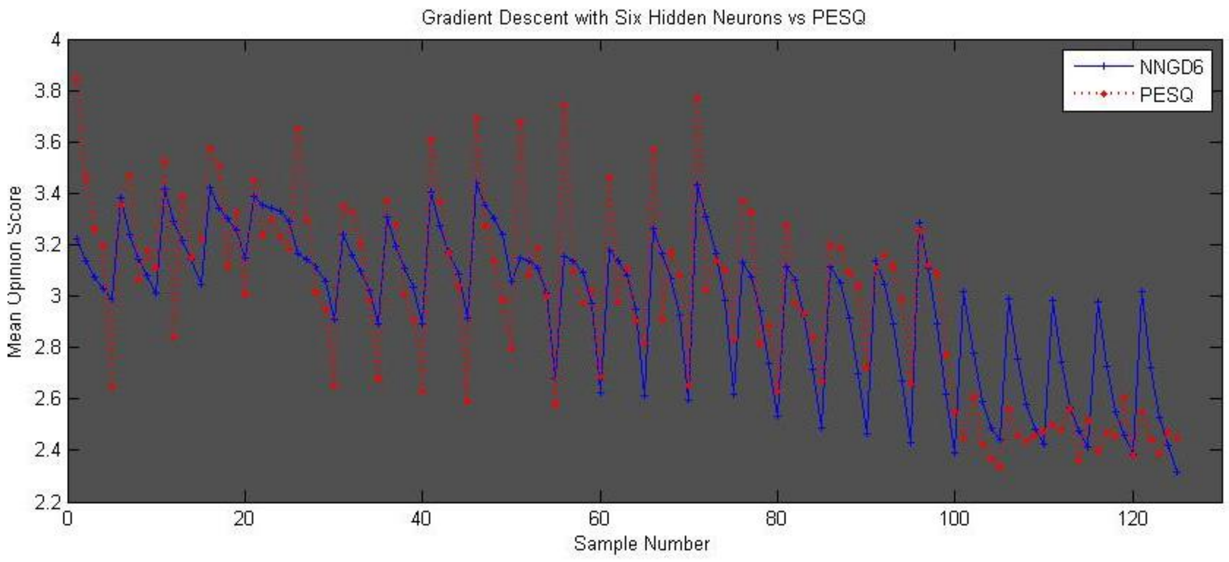


Figure 4.11 – Gradient Descent with Six Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.12 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with five hidden neurons. Levenberg-Marquardt trains the artificial neural network within 16 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0236 which is a larger number than the previous iteration of the Levenberg-Marquardt trained network.

Table 4.12 – Levenberg-Marquardt with Six Hidden Neurons (SPEEX)

Epochs:	16
Time:	<1 second
Performance Training: (MSE)	0.0122
Performance Validation:	0.0119
Performance Testing:	0.0314
Overall performance:	0.0236
Gradient:	0.0300

Figure 4.12 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquqardt with Five hidden neurons is able to accurately identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. The artificial neural network predicted Mean Opinion Score values follow the peaks and lows of the PESQ scores to a great extent.

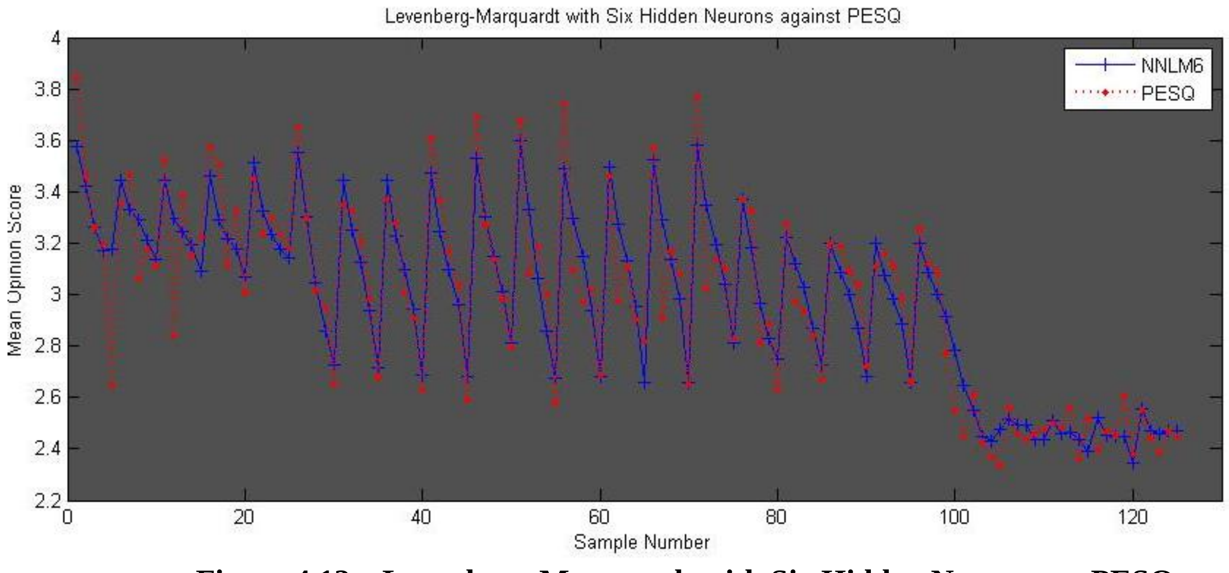


Figure 4.12 – Levenberg-Marquardt with Six Hidden Neurons vs PESQ

Conclusion

With the increment to six hidden neurons for both paradigms, significant improvement was found for both Levenberg-Marquardt and Gradient Descent. Levenberg-Marquardt's relationship to the PESQ benchmark has improved once again and is producing results that could be considered viable. Gradient Descent, although improving, is still falling short with greatly exaggerated peaks.

4.1.7 Training ANN with a maximum of 1000 Epochs and Seven Hidden Neurons

Gradient Descent

Table 4.13 shows that during the training process the Gradient Descent algorithm once again reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum during training, the validation and testing phases begun. The network took 15 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0486 which is higher than the previous iteration of the network trained by Gradient Descent.

Table 4.13 – Gradient Descent with Seven Hidden Neurons (SPEEX)

Epochs:	1000 (maximum)
Time:	15 seconds
Performance Training: (MSE)	0.0467
Performance Validation:	0.0654
Performance Testing:	0.0408
Overall performance:	0.0486
Gradient:	0.0266

Figure 4.13 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with seven hidden neurons is unable to identify a pattern between the input variables and the target output. The artificial neural network predicted Mean Opinion Score values vaguely follow the peaks and lows of the PESQ scores but many of these plots are exaggerated and become extremely inaccurate in the upper quadrant of samples.

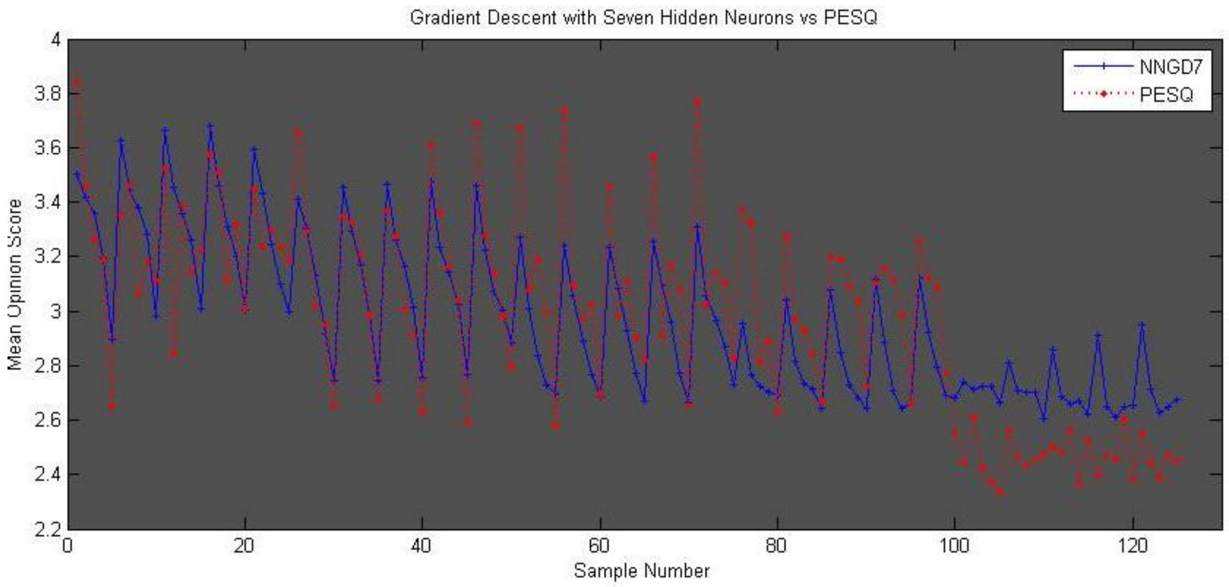


Figure 4.13 – Gradient Descent with Seven Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.14 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with five hidden neurons. Levenberg-Marquardt trains the artificial neural network within 8 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0458 which is a significantly larger number than the previous iteration of the Levenberg-Marquardt trained network.

Table 4.14 – Levenberg-Marquardt with Seven Hidden Neurons (SPEEX)

Epochs:	8
Time:	<1 second
Performance Training: (MSE)	0.0454
Performance Validation:	0.0205
Performance Testing:	0.0725
Overall performance:	0.0458
Gradient:	0.00236

Figure 4.14 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquqardt with seven hidden neurons is able to vaguely identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. The artificial neural network predicted Mean Opinion Score values follow the peaks and lows of the PESQ scores to some extent but this iteration shows greatly reduced performance from the previous Levenberg-Marquardt trained networks.

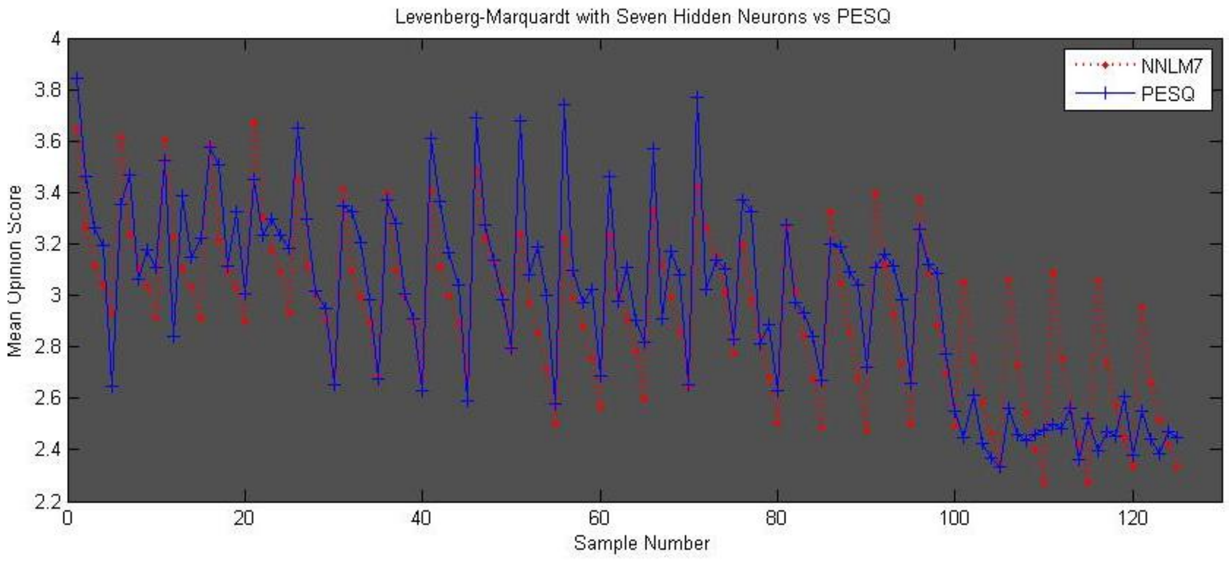


Figure 4.14 – Levenberg-Maquardt with Seven Hidden Neurons vs PESQ

Conclusion
With the increment to seven hidden neurons for both paradigms, a loss of accuracy for both Levenberg-Marquardt and Gradient Descent was discovered. Levenberg-Marquardt's relationship to the PESQ benchmark now has exaggerated peaks where as Gradient Descent has greatly exaggerated lows.

4.1.8 Training ANN with a maximum of 1000 Epochs and Ten Hidden Neurons

Gradient Descent

Table 4.15 shows that during the training process the Gradient Descent algorithm once again reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum during training, the validation and testing phases begun. The network took 14 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0662 which is significantly higher than the previous iteration of the network trained by Gradient Descent.

Table 4.15 – Gradient Descent with Ten Hidden Neurons (SPEEX)

Epochs:	1000 (maximum)
Time:	14 seconds
Performance Training: (MSE)	0.0586
Performance Validation:	0.1081
Performance Testing:	0.0590
Overall performance:	0.0662
Gradient:	0.0300

Figure 4.15 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with ten hidden neurons is vaguely able to map a pattern between the input variables and the target output. The artificial neural network predicted Mean Opinion Score values vaguely follow the peaks and lows of the PESQ scores but many of these plots are exaggerated and become extremely inaccurate in the upper quadrant of samples.

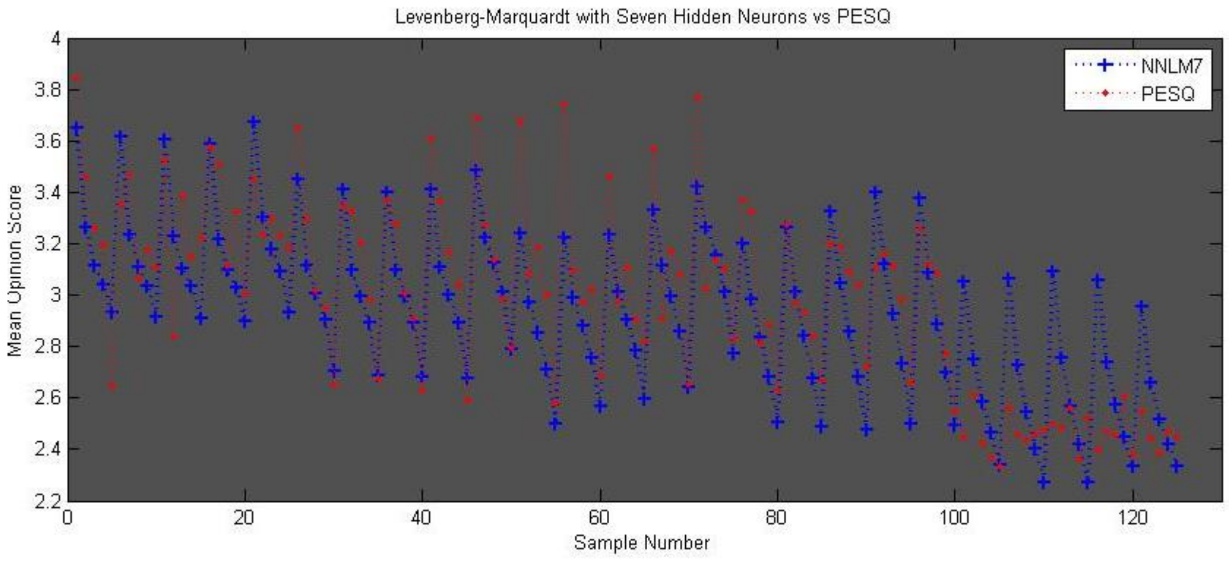


Figure 4.15 – Gradient Descent with Ten Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.16 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with ten hidden neurons. Levenberg-Marquardt trains the artificial neural network within 12 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0158 which is a significantly lower number than the previous iteration of the Levenberg-Marquardt trained network which is not following the trend set in previous iterations.

Table 4.16 – Levenberg-Marquardt with Ten Hidden Neurons (SPEEX)

Epochs:	12
Time:	<1 second
Performance Training: (MSE)	0.0130
Performance Validation:	0.0180
Performance Testing:	0.0267
Overall performance:	0.0192
Gradient:	0.0671

Figure 4.16 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Gradient Descent with ten hidden neurons is able to map a pattern between the input variables and the target output. The artificial neural network predicted Mean Opinion Score values follow the peaks and lows of the PESQ scores to a high degree but not as well as the results presented by both five and six hidden neurons.

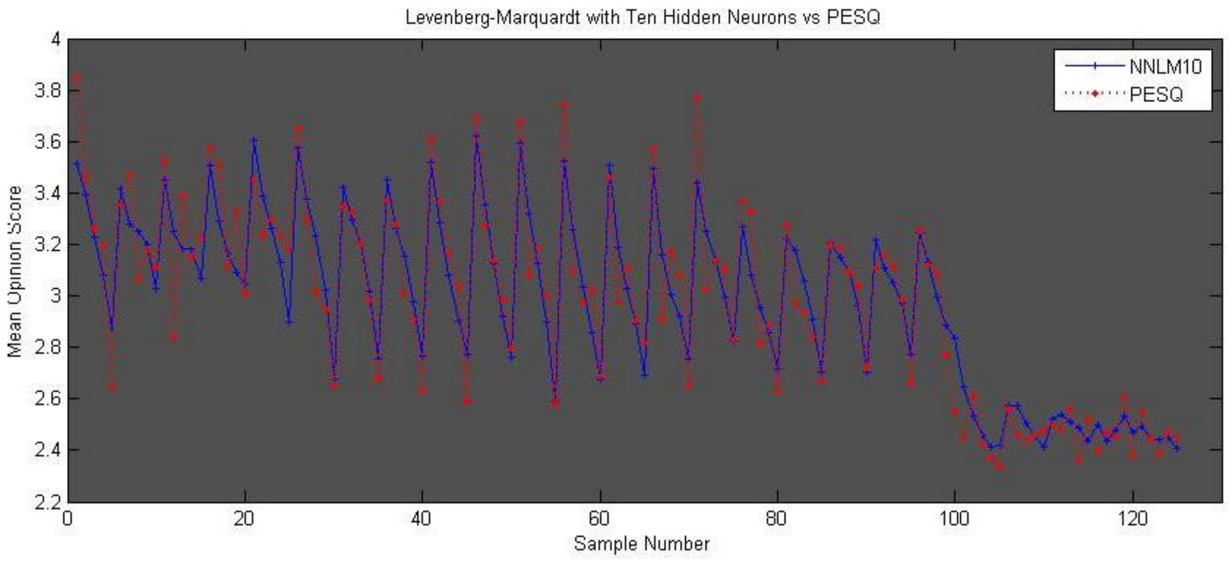


Figure 4.16 – Levenberg-Marquardt with Ten Hidden Neurons vs PESQ

Conclusion

With the increment to ten hidden neurons for both paradigms, significant improvement was found for both Levenberg-Marquardt and Gradient Descent when compared with the iteration of seven neurons. Levenberg-Marquardt's relationship to the PESQ benchmark has improved once again and is producing results that could be considered viable, but not to the degree of those found within the paradigms of five and six hidden neurons. Gradient Descent, although improving, is still falling short of the results from Levenberg-Marquardt.

4.1.9 Comparison of Results between Five & Six Hidden Neurons

A comparison of the results from sections 4.15 & 4.16 will be shown here to show the effectiveness of the learning algorithms compared against PESQ.

Gradient Descent against PESQ

Figure 4.17 displays the results of the Gradient Descent trained artificial neural networks with both five and six hidden neurons; which proved to be the most effective neural network architecture for this problem, against the original PESQ benchmark. As displayed below no architecture consistently mimics the line of the PESQ score, but in the first two quartiles NNGD5 performs slightly more accurately. In the latter two quadrants no architecture is able to accurately map the Mean Opinion Score to a useable level of accuracy.

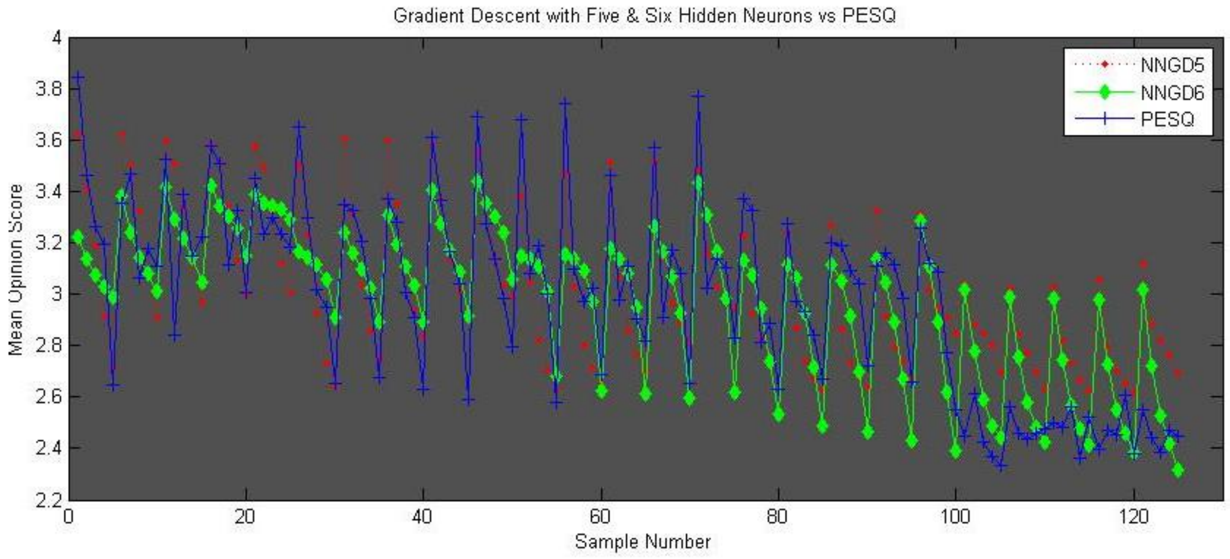


Figure 4.17 – Gradient Descent 5/6 Hidden Neurons vs PESQ

Levenberg-Marquardt against PESQ

Figure 4.18 displays the two most accurate result sets from the Levenberg-Marquardt trained network. NNLM5 (Levenberg-Marquardt with Five Hidden Neurons) and NNLM6 (Levenberg-Marquardt with Six Hidden Neurons) are plotted on the graph against the original PESQ benchmark.

From this graph it can be determined that while both architectures are accurate the NNLM5 architecture is slightly less accurate than the NNLM6 when plotting the Mean Opinion Score.

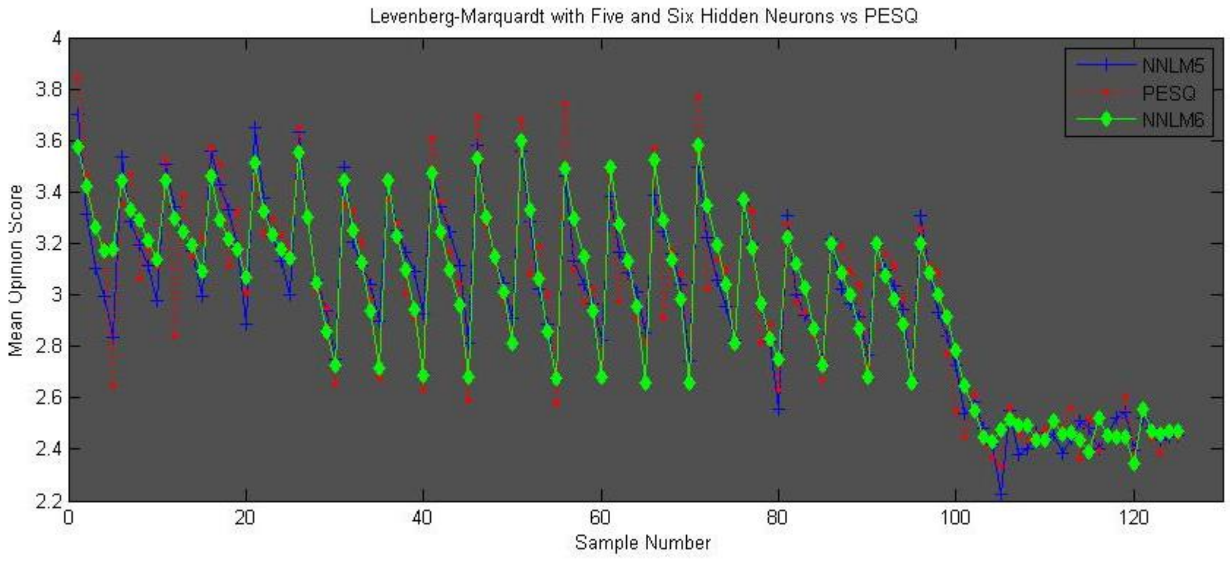


Figure 4.18 – Levenberg-Marquardt 5/6 Hidden Neurons vs PESQ

Gradient Descent and Levenberg-Marquardt with five and six hidden neurons against PESQ

Figure 4.19 shows both the results of Gradient Descent and Levenberg-Marquardt trained networks with five and six hidden neurons plotted against PESQ on a reduced sample size. By looking at this reduced sample size it is clear to see the large gap between the results produced by the Gradient Descent training and the Levenberg-Marquardt training.

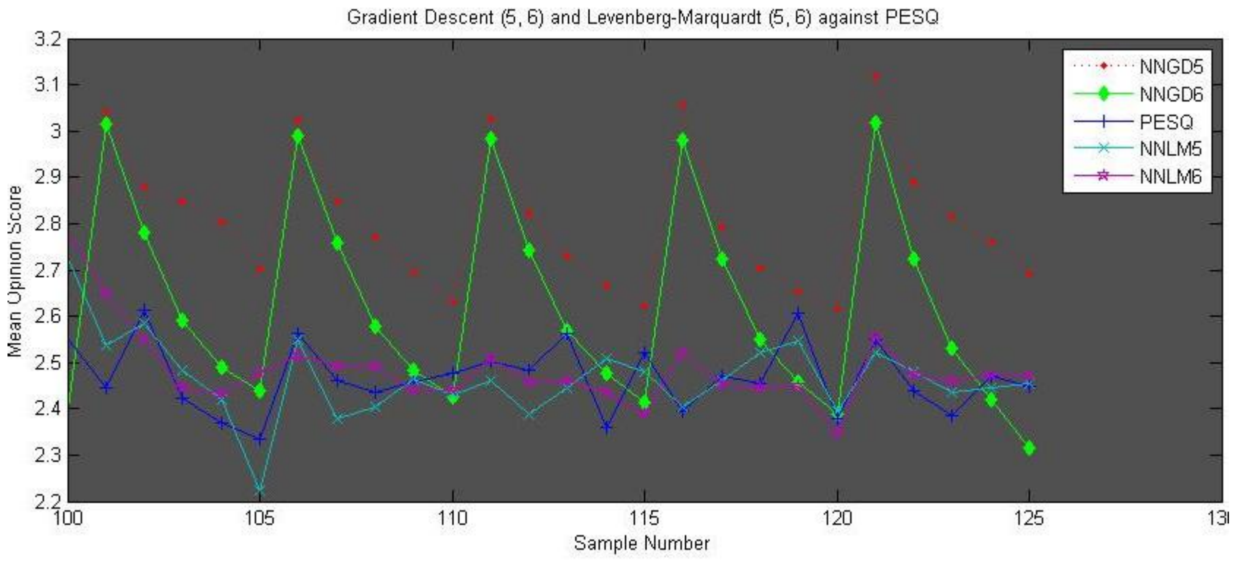


Figure 4.19 –GD 5/6 & LM 5/6 vs PESQ

Conclusion

The Levenberg-Marquardt paradigm with six hidden neurons is the most accurate overall when compared to the PESQ benchmark. Overall, the Gradient Descent algorithm is unable to perform to a satisfactory standard and achieve accurate Mean Opinion Scores.

4.2 Results with G711a Codec

4.2.1 Training ANN with a maximum of 1000 Epochs and One Hidden Neuron with the G711a codec

Gradient Descent

As displayed in Table 4.17 during the training process the algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 18 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0962 which significantly higher than 0.

Table 4.17 – Gradient Descent with One Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	18 seconds
Performance Training: (MSE)	0.0897
Performance Validation:	0.1089
Performance Testing:	0.1133
Overall performance:	0.0962
Gradient:	0.0351

Figure 4.20 shows the Mean Opinion Scores as predicted by the one hidden neuron artificial neural network trained by Gradient Descent against the original PESQ benchmark. The results show a vague correlation between the two results with the network able to predict the dip in the Mean Opinion Score at approximately sample 75.

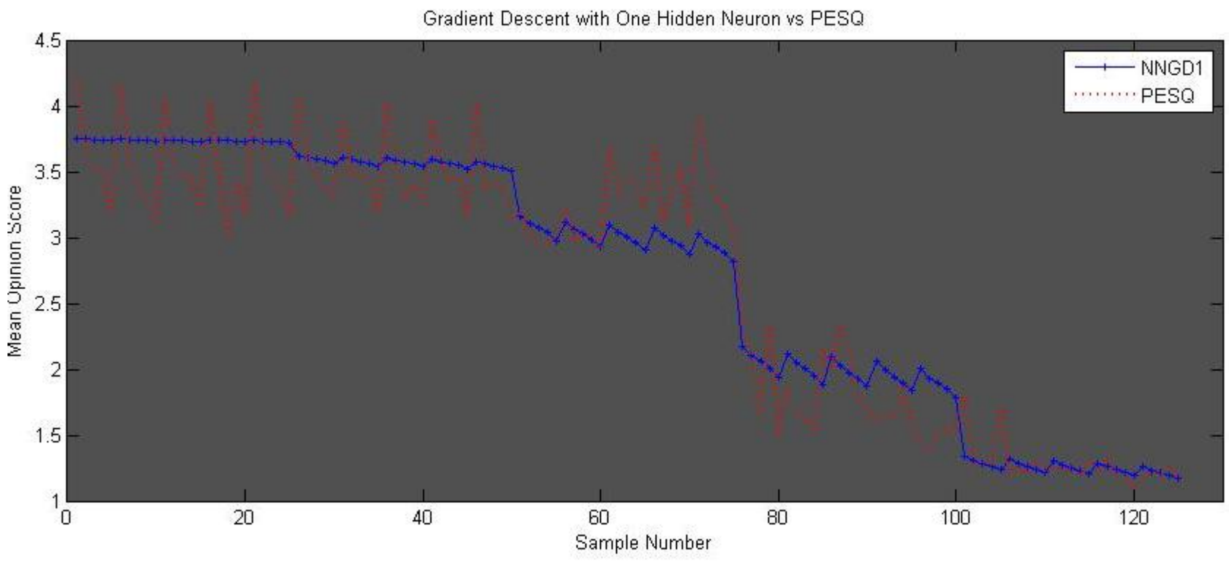


Figure 4.20 – Gradient Descent with One Hidden Neuron vs PESQ

Levenberg-Marquardt

Table 4.18 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with one hidden neuron. Levenberg-Marquardt trains the artificial neural network within 8 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.1380 which is a considerably large error value.

Table 4.18 – Levenberg-Marquardt with One Hidden Neuron (G711a)

Epochs:	8
Time:	<1 second
Performance Training: (MSE)	0.1380
Performance Validation:	0.0132
Performance Testing:	0.0655
Overall performance:	0.1080
Gradient:	0.000866

Figure 4.21 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the

network trained by Levenberg-Marquardt with one hidden neuron is able to vaguely identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. Sample numbers 1 through 50 prove to be extremely inaccurate in this iteration of the neural network. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network.

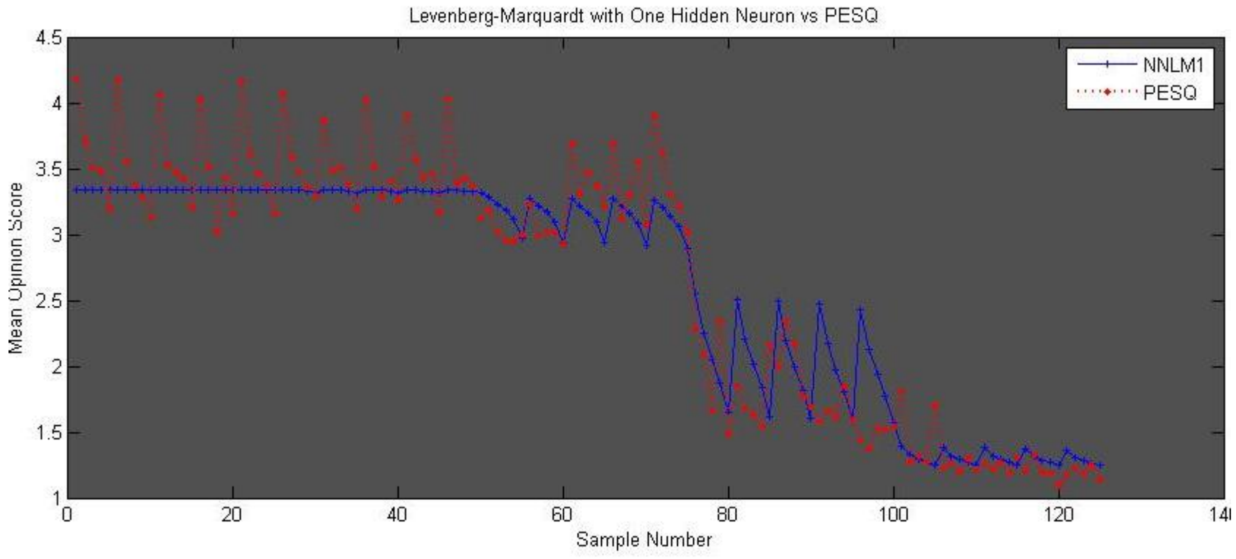


Figure 4.21 – Levenberg-Marquardt with One Hidden Neuron vs PESQ

Conclusion

Both paradigms perform slightly worse than their counterparts using the SPEEX training data. This unexpected result has shown both paradigms to perform extremely poorly with one hidden neuron.

4.2.2 Training ANN with a maximum of 1000 Epochs and Two Hidden Neuron with the G711a codec

Gradient Descent

As displayed in Table 4.19 during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 18 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.2765 which significantly higher than 0.

Table 4.19 – Gradient Descent with Two Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	18 seconds
Performance Training: (MSE)	0.2765
Performance Validation:	0.1618
Performance Testing:	0.2055
Overall performance:	0.2483
Gradient:	0.205

Figure 4.22 shows the Mean Opinion Scores as predicted by the artificial neural network with two hidden neurons trained by Gradient Descent against the original PESQ benchmark. The results show a vague correlation between the two results with the network showing an increased level of inaccuracy compared to the previous iteration with exaggerated peaks and lows.

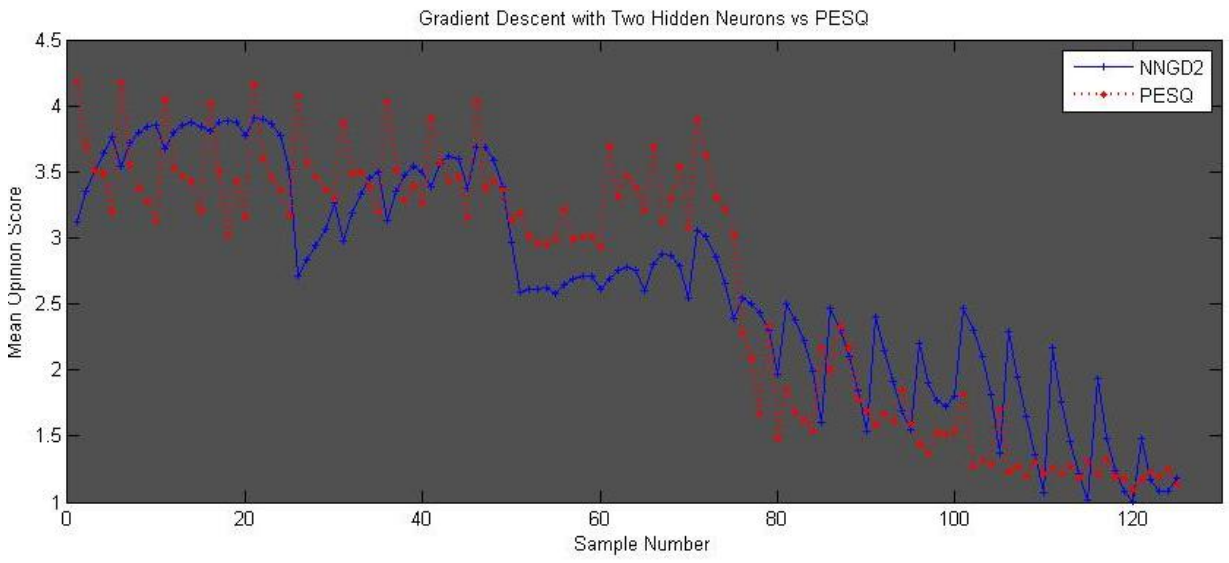


Figure 4.22 – Gradient Descent with Two Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.20 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with two hidden neurons. Levenberg-Marquardt trains the artificial neural network within 15 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0687 which is following the trend of approaching closer to 0 with each increment of the hidden layer.

Table 4.20 – Levenberg-Marquardt with Two Hidden Neurons (G711a)

Epochs:	15
Time:	<1 second
Performance Training: (MSE)	0.0567
Performance Validation:	0.1311
Performance Testing:	0.0610
Overall performance:	0.0687
Gradient:	0.000742

Figure 4.23 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with two hidden neurons once again is able to vaguely identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. As with the previous iteration, sample numbers 1

through 50 prove to be extremely inaccurate in this iteration of the neural network. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network.

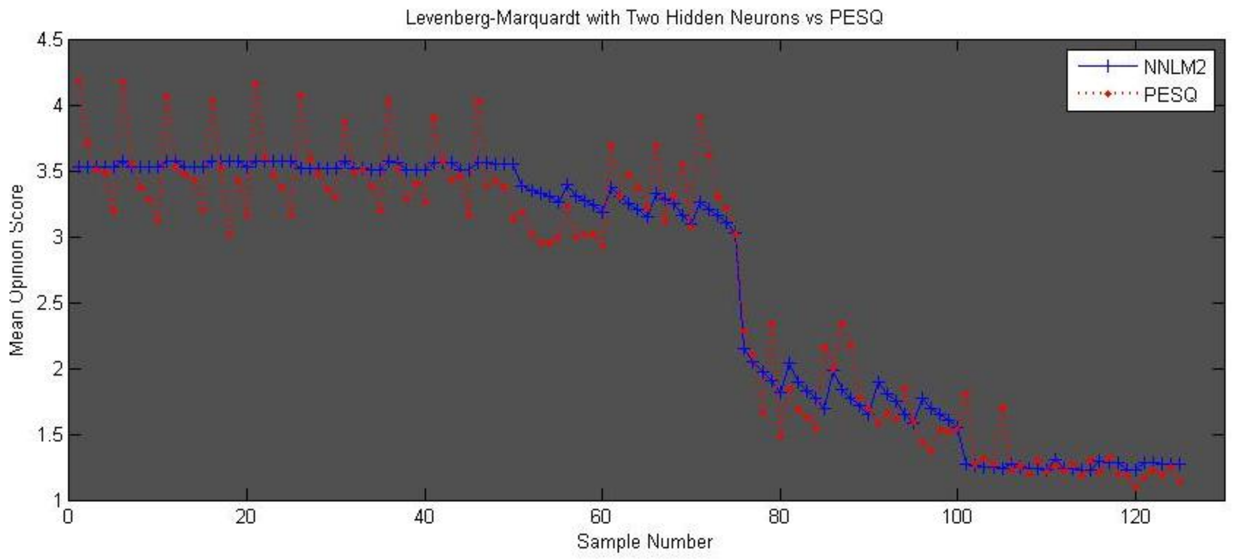


Figure 4.23 – Levenberg-Marquardt with Two Hidden Neurons vs PESQ

Conclusion
Results for both paradigms are slightly improved but are still lacking compared to the original SPEEX counterparts. Levenberg-Marquardt results remain fairly consistent with those found in the previous experiment. Gradient Descent shows some improvement. Both paradigms prove unviable.

4.2.3 Training ANN with a maximum of 1000 Epochs and Three Hidden Neuron with the G711a codec

Gradient Descent

As displayed in Table 4.21 during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 16 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.2014 which significantly higher than 0.

Table 4.21 – Gradient Descent with Three Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	16 seconds
Performance Training: (MSE)	0.1641
Performance Validation:	0.1489
Performance Testing:	0.4248
Overall performance:	0.2014
Gradient:	0.0928

Figure 4.24 shows the Mean Opinion Scores as predicted by the artificial neural network with three hidden neurons as trained by Gradient Descent against the original PESQ benchmark. The results show a vague correlation between the PESQ scoring and the predicted results

with the network showing a slightly increased accuracy than the previous iteration but it is still not able to mirror the results of the PESQ paradigm.

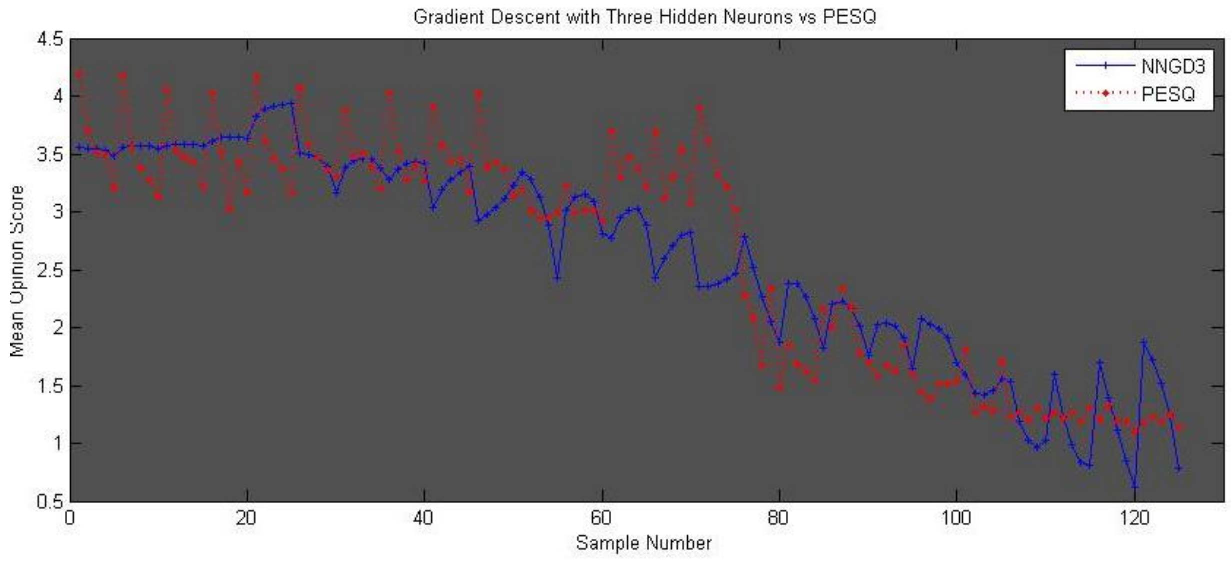


Figure 4.24 – Gradient Descent with Three Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.22 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with three hidden neurons. Levenberg-Marquardt trains the artificial neural network within 14 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0504 which is following the trend of approaching closer to 0 with each increment of the hidden layer.

Table 4.22 – Levenberg-Marquardt with Three Hidden Neurons (G711a)

Epochs:	14
Time:	<1 second
Performance Training: (MSE)	0.0407
Performance Validation:	0.0867
Performance Testing:	0.0585
Overall performance:	0.0504
Gradient:	0.00631

Figure 4.25 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with three hidden neurons once again is able to vaguely identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. In comparison with the previous iteration, sample numbers 1 through 50 show great improvement in this iteration of the neural network. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network.

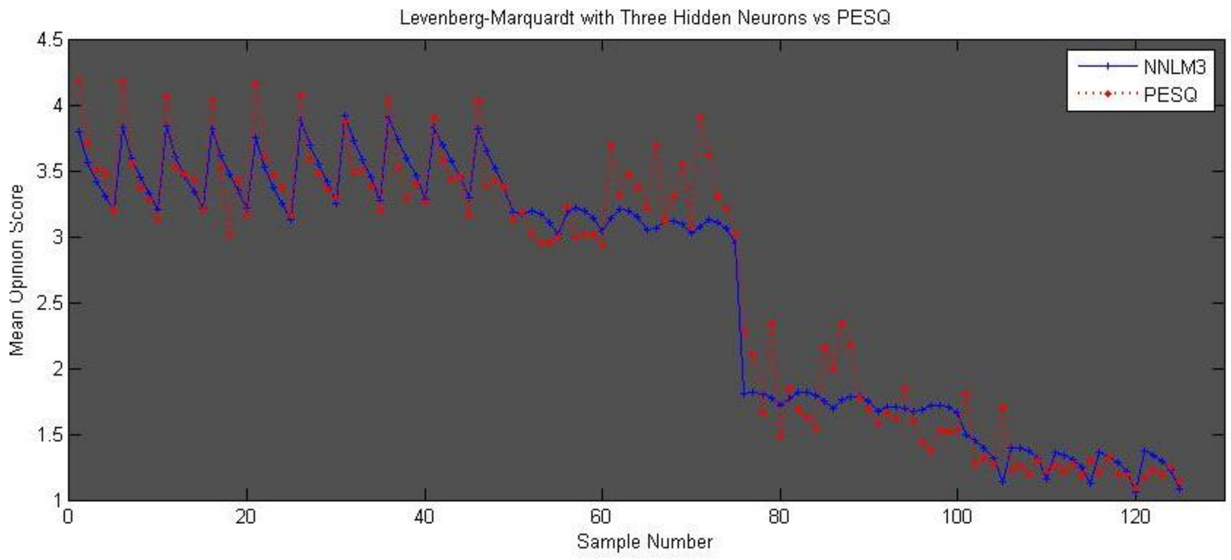


Figure 4.25 – Levenberg-Marquardt with Three Hidden Neurons vs PESQ

Conclusion

Both paradigms are beginning to show significant improvement with the addition of a third hidden neuron. Levenberg-Marquardt especially is producing more viable results and accurately mimics the dip found in the PESQ scoring.

4.2.4 Training ANN with a maximum of 1000 Epochs and Four Hidden Neuron with the G711a codec

As displayed in Table 4.23 during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 15 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.2742 which significantly higher than 0 and higher than the previous iteration of the artificial neural network.

Table 4.23 – Gradient Descent with Four Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	15 seconds
Performance Training: (MSE)	0.2638
Performance Validation:	0.3104
Performance Testing:	0.2857
Overall performance:	0.2742
Gradient:	0.106

Figure 4.26 shows the Mean Opinion Scores as predicted by the artificial neural network with four hidden neurons as trained by Gradient Descent against the original PESQ benchmark. The results show a vague correlation between the PESQ scoring and the predicted results

with the network showing a slightly increased accuracy than the previous iteration but it is still not able to mirror the results of the PESQ paradigm. This iteration once again suffers from exaggerated peaks and lows.

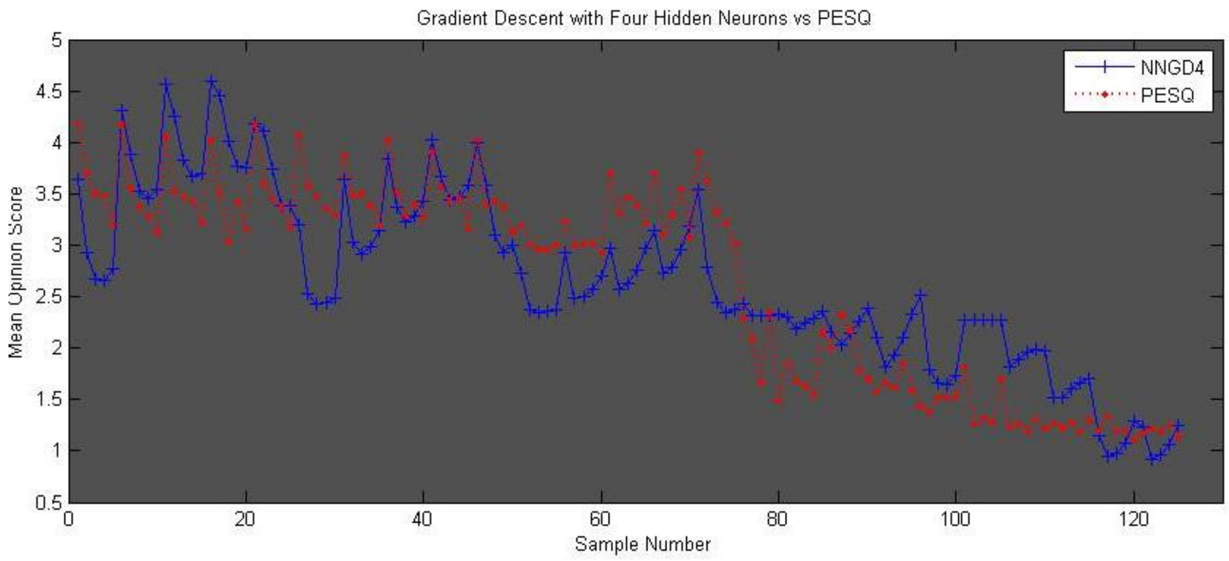


Figure 4.26 – Gradient Descent with Four Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.24 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with four hidden neurons. Levenberg-Marquardt trains the artificial neural network within 22 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0282 which is following the trend of approaching closer to 0 with each increment of the hidden layer.

Table 4.24 – Levenberg-Marquardt with Four Hidden Neurons (G711a)

Epochs:	22
Time:	<1 second
Performance Training: (MSE)	0.0196
Performance Validation:	0.0390
Performance Testing:	0.0569
Overall performance:	0.0282
Gradient:	0.00404

Figure 4.27 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with four hidden neurons once again is able to identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. As with the previous iteration, sample numbers 1 through 50 show great improvement in this iteration of the neural network. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network.

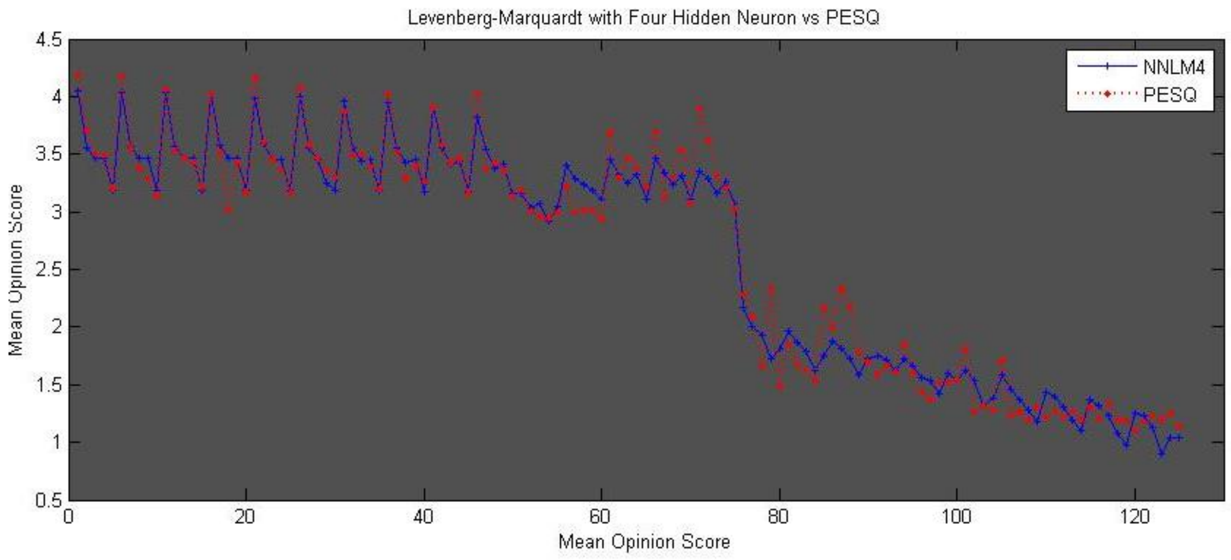


Figure 4.27 – Levenberg-Marquardt with Four Hidden Neurons vs PESQ

Conclusion

With four hidden neurons both paradigms have improved again but Levenberg-Marquardt is clearly identifying the relationship between the inputs and target outputs whereas Gradient Descent still has many exaggerated peaks and lows.

4.2.5 Training ANN with a maximum of 1000 Epochs and Five Hidden Neuron with the G711a codec

Gradient Descent

Table 4.25 displays the results from the artificial neural network with five hidden neurons. The Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 15 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.1600 which significantly higher than 0.

Table 4.25 – Gradient Descent with Five Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	15 seconds
Performance Training: (MSE)	0.1450
Performance Validation:	0.1894
Performance Testing:	0.1993
Overall performance:	0.1600
Gradient:	0.0920

Figure 4.28 shows the Mean Opinion Scores as predicted by the artificial neural network with five hidden neurons as trained by Gradient Descent against the original PESQ benchmark. The results show a vague correlation between the PESQ scoring and the predicted results with the network showing a slightly increased accuracy than the previous iteration but it is still not able to mirror the results of the PESQ paradigm. This iteration once again suffers from exaggerated peaks and lows. This iteration mirrors the dip shown in the PESQ scoring.

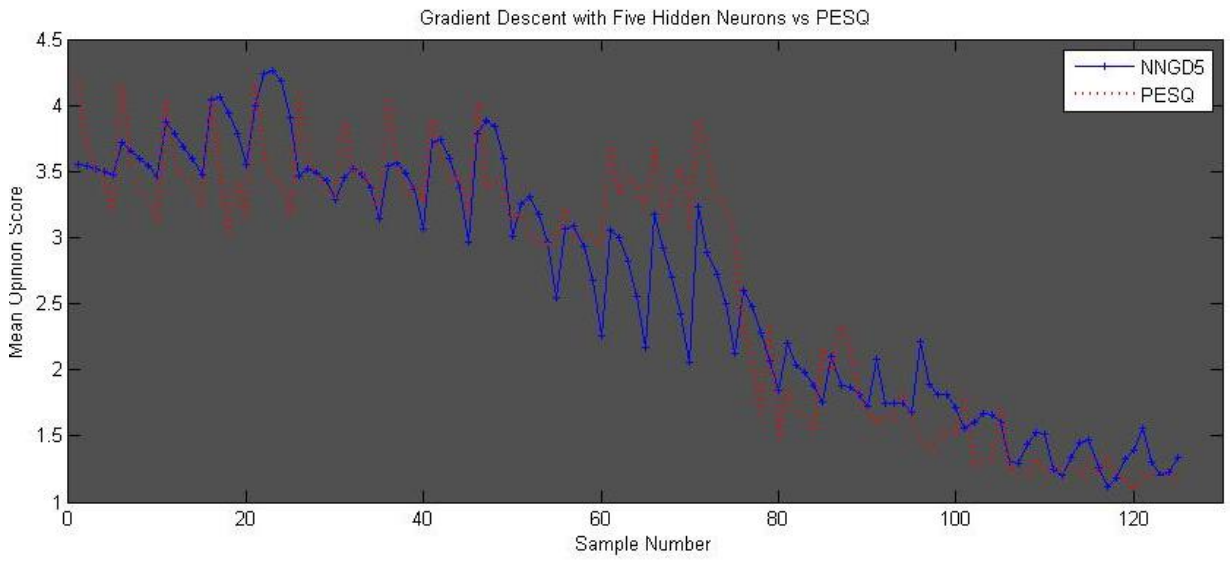


Figure 4.28 – Gradient Descent with Five Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.26 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with five hidden neurons. Levenberg-Marquardt trains the artificial neural network within 13 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0324 which is following the trend of approaching closer to 0 with each increment of the hidden layer.

Table 4.26 – Levenberg-Marquardt with Five Hidden Neurons (G711a)

Epochs:	13
Time:	<1 second
Performance Training: (MSE)	0.0280
Performance Validation:	0.0111
Performance Testing:	0.0582
Overall performance:	0.0324
Gradient:	0.114

Figure 4.29 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with five hidden neurons once again is able to identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. As with the previous iteration, sample numbers 1 through 50 show great improvement in this iteration of the neural network. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network. Samples 70 through 100 show an unexpected level of inaccuracy.

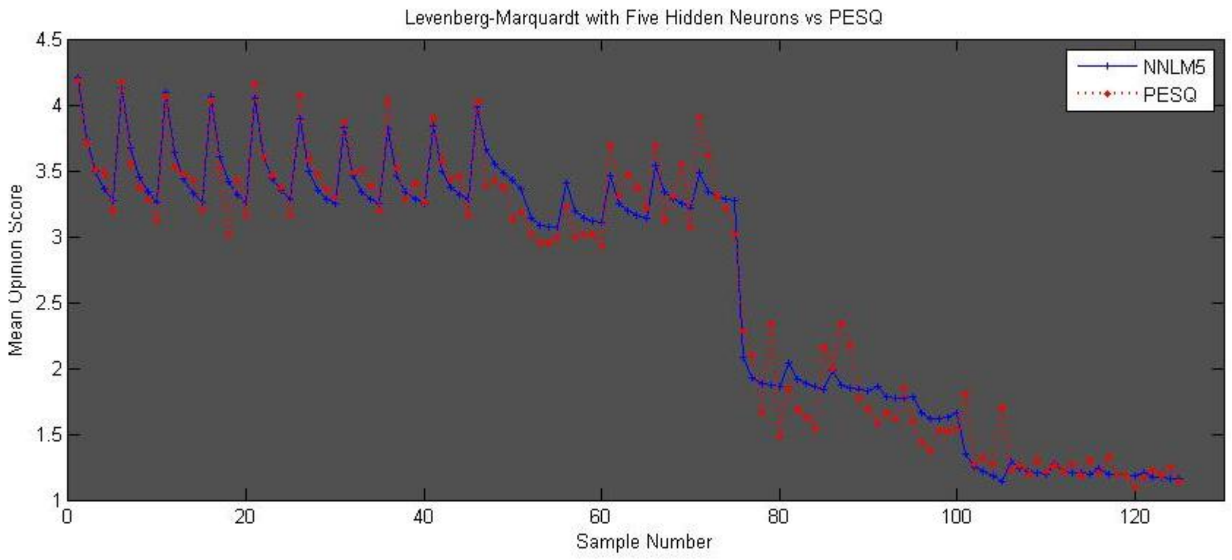


Figure 4.29 – Levenberg-Marquardt with Five Hidden Neurons vs PESQ

Conclusion

Levenberg-Marquardt performs consistently with the previous experiment and is able to mirror the dip and peaks found in the PESQ benchmark. Gradient Descent still suffers from exaggerated peaks and lows and does not mimic the dip found at sample 75 accurately.

4.2.6 Training ANN with a maximum of 1000 Epochs and Six Hidden Neuron with the G711a codec

Gradient Descent

As displayed in Table 4.27 during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 15 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.0899 which significantly higher than 0 but continues the trend of approaching closer to 0 with each increment of the number of neurons in the hidden layer.

Table 4.27 – Gradient Descent with Six Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	15 seconds
Performance Training: (MSE)	0.0804
Performance Validation:	0.1312
Performance Testing:	0.0922
Overall performance:	0.0899
Gradient:	0.116

Figure 4.30 shows the Mean Opinion Scores as predicted by the artificial neural network with four hidden neurons as trained by Gradient Descent against the original PESQ benchmark. The results show a good correlation between the PESQ scoring and the predicted results with the network showing a slightly increased accuracy than the previous iteration. This iteration once again suffers from exaggerated peaks but only in samples 80 through 100.

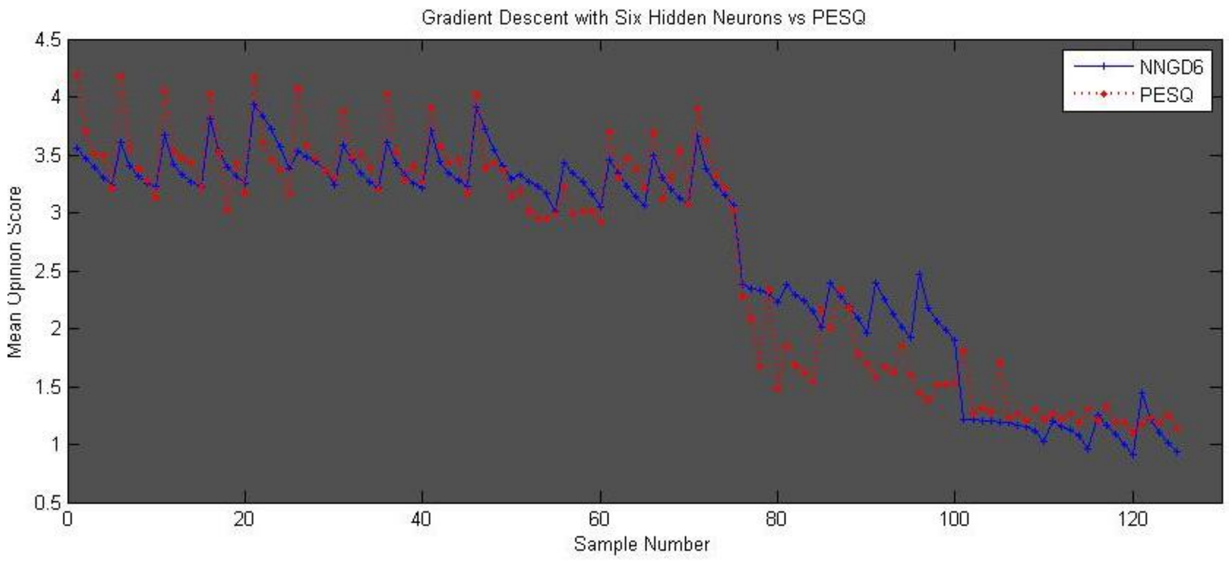


Figure 4.30 – Gradient Descent with Six Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.28 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with six hidden neurons. Levenberg-Marquardt trains the artificial neural network within 16 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0437 which is significantly higher than the previous iteration of the Levenberg-Marquardt trained network.

Table 4.28 – Levenberg-Marquardt with Six Hidden Neurons (G711a)

Epochs:	16
Time:	<1 second
Performance Training: (MSE)	0.0374
Performance Validation:	0.0343
Performance Testing:	0.0821
Overall performance:	0.0437
Gradient:	0.0214

Figure 4.31 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with six hidden neurons once again is able to identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. As with the previous iterations, sample numbers 1 through 50 show great improvement in this iteration of the neural network. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network. Samples 70 through 100 show an improved level of accuracy in this iteration of the network.

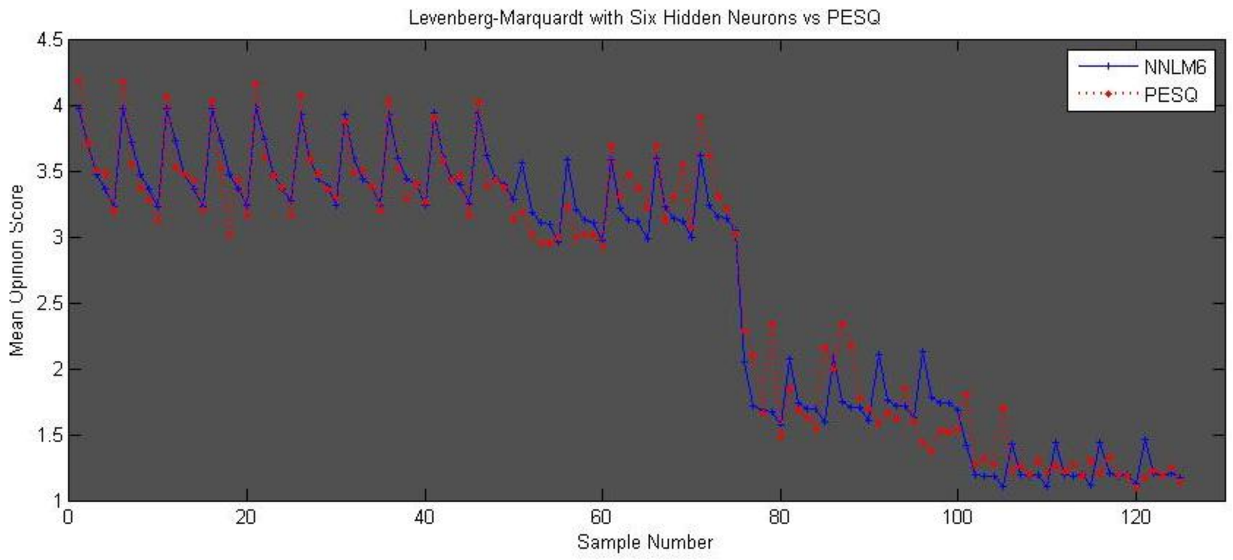


Figure 4.31 – Levenberg-Marquardt with Six Hidden Neurons vs PESQ

Conclusion

As with the SPEEX experiments, the Levenberg-Marquardt algorithm performs extremely accurately with six hidden neurons and truly can predict the Mean Opinion Score to a high degree such as with PESQ. Gradient Descent however is still suffering from poor accuracy

4.2.7 Training ANN with a maximum of 1000 Epochs and Seven Hidden Neuron with the G711a codec

Gradient Descent

As displayed in Table 4.29 during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 14 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.1504 which significantly higher than 0 and shows poorer performance than the previous iteration.

Table 4.29 – Gradient Descent with Seven Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	14 seconds
Performance Training: (MSE)	0.1216
Performance Validation:	0.2144
Performance Testing:	0.2185
Overall performance:	0.1504
Gradient:	0.0762

Figure 4.32 shows the Mean Opinion Scores as predicted by the artificial neural network with seven hidden neurons as trained by Gradient Descent against the original PESQ benchmark. The results show a vague correlation between the PESQ scoring and the predicted results with the network showing a slightly decreased accuracy than the previous iteration. This iteration as with previous paradigms once again suffers from exaggerated peaks and lows.

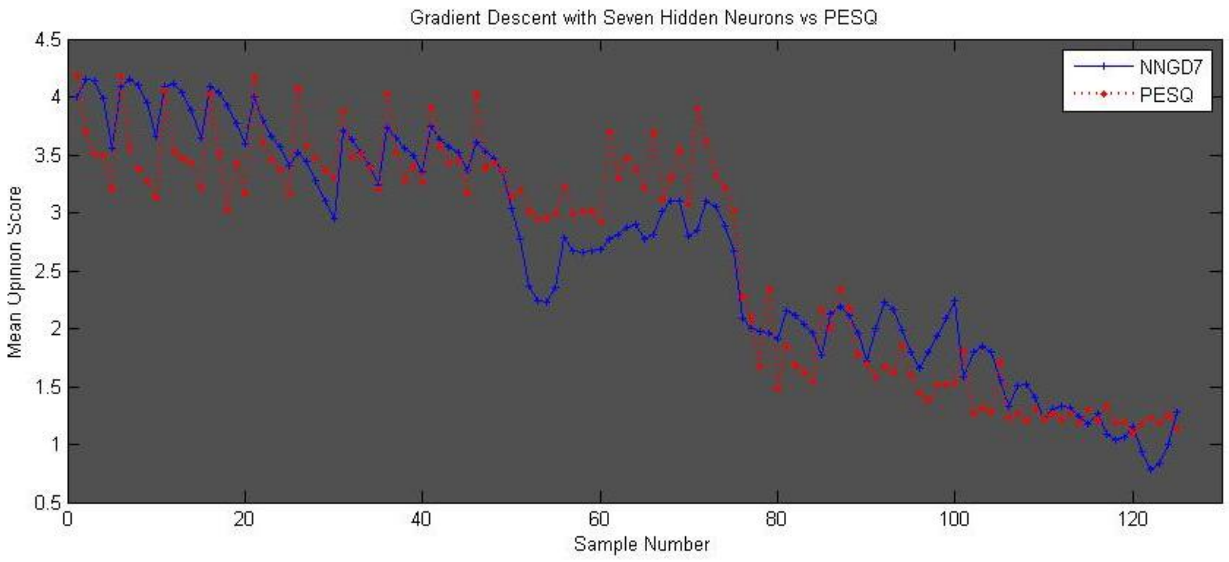


Figure 4.32 – Gradient Descent with Seven Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.30 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with seven hidden neurons. Levenberg-Marquardt trains the artificial neural network within 27 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0201 which is showing an improvement from the previous iteration of the network.

Table 4.30 – Levenberg-Marquardt with Seven Hidden Neurons (G711a)

Epochs:	27
Time:	<1 second
Performance Training: (MSE)	0.0184
Performance Validation:	0.0217
Performance Testing:	0.0260
Overall performance:	0.0201
Gradient:	0.0249

Figure 4.33 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with seven hidden neurons once again is able to identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. The accuracy of the results in sample numbers 1 through 75 is greatly improved in this iteration, accurately mirroring the peaks and lows of the PESQ output. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network. Samples 75 through 100 however show some inaccuracy in this iteration.

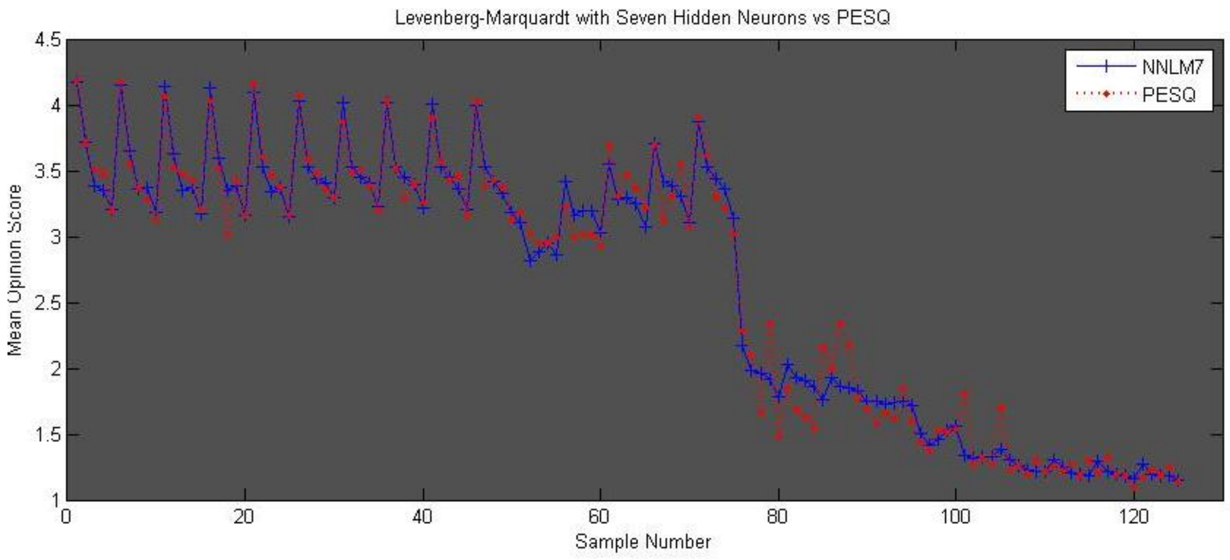


Figure 4.33 – Levenberg-Marquardt with Seven Hidden Neurons vs PESQ

Conclusion

The expected result from this experiment, based on the SPEEX experiment results was that the performance would suffer from the addition of another neuron compared to the six neuron paradigm and this was proven to be true. Both Levenberg-Marquardt and Gradient Descent lose some accuracy in this paradigm.

4.2.8 Training ANN with a maximum of 1000 Epochs and Ten Hidden Neurons with the G711a codec

Gradient Descent

As displayed in Table 4.31 during the training process the Gradient Descent algorithm reached the maximum number of epochs for the experiment which is set to 1000. This means that before the network achieved the global minimum the validation and testing phases begun. The network took 15 seconds from initialising the synaptic weights to completing the validation stage. The overall mean squared error performance achieved by the network was 0.1519 which significantly higher than 0.

Table 4.31– Gradient Descent with Ten Hidden Neurons (G711a)

Epochs:	1000 (maximum)
Time:	15 seconds
Performance Training: (MSE)	0.1264
Performance Validation:	0.1615
Performance Testing:	0.2592
Overall performance:	0.1519
Gradient:	0.0603

Figure 4.34 shows the Mean Opinion Scores as predicted by the artificial neural network with seven hidden neurons as trained by Gradient Descent against the original PESQ benchmark. The results show a vague correlation between the PESQ scoring and the predicted results

with the network showing a slightly decreased accuracy than the previous iteration. This iteration as with previous paradigms once again suffers from exaggerated peaks and lows.

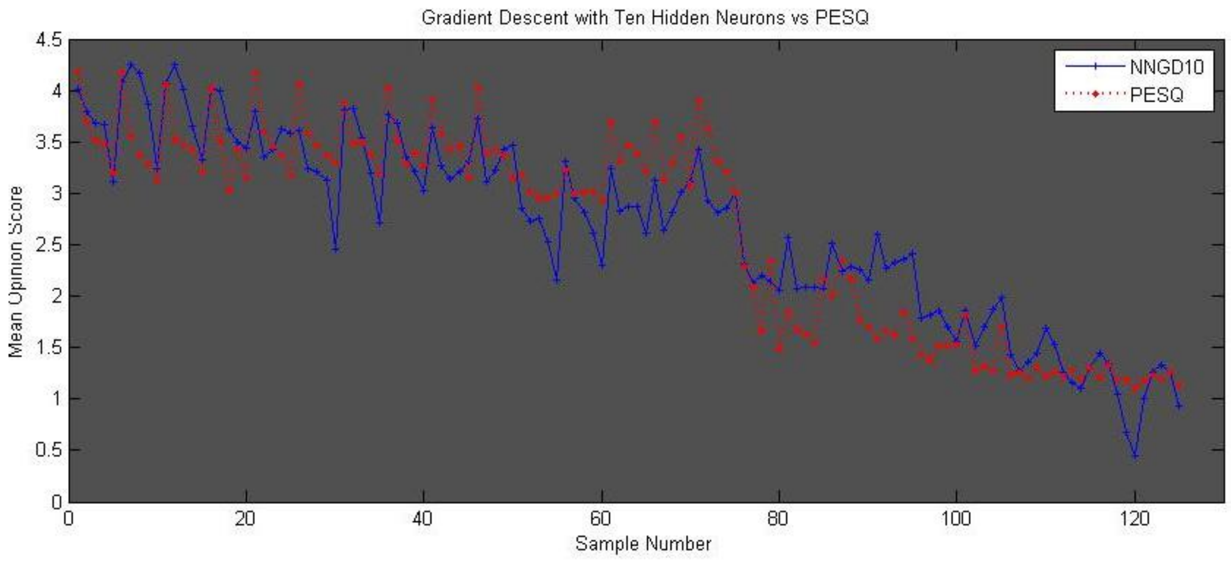


Figure 4.34 – Gradient Descent with Ten Hidden Neurons vs PESQ

Levenberg-Marquardt

Table 4.32 shows the statistics from the Levenberg-Marquardt learning algorithm when training an artificial neural network with ten hidden neurons. Levenberg-Marquardt trains the artificial neural network within 14 iterations. In under a second the process of initialising weights on the synaptic links, training, testing and validation took place. The overall MSE performance was 0.0364 which shows a slump in performance from the previous iteration.

Table 4.32 – Levenberg-Marquardt with Ten Hidden Neurons (G711a)

Epochs:	14
Time:	<1 second
Performance Training: (MSE)	0.0327
Performance Validation:	0.0620
Performance Testing:	0.0278
Overall performance:	0.0364
Gradient:	0.00905

Figure 4.35 shows the correlation of the Mean Opinion Score output by the trained artificial neural network compared to the benchmark results from PESQ. As shown in this graph the network trained by Levenberg-Marquardt with ten hidden neurons once again is able to identify a pattern between the input variables and the target output as shown by the line descending along with the PESQ line. The dip in the G711a Mean Opinion Score is mirrored between the PESQ results and that of the neural network.

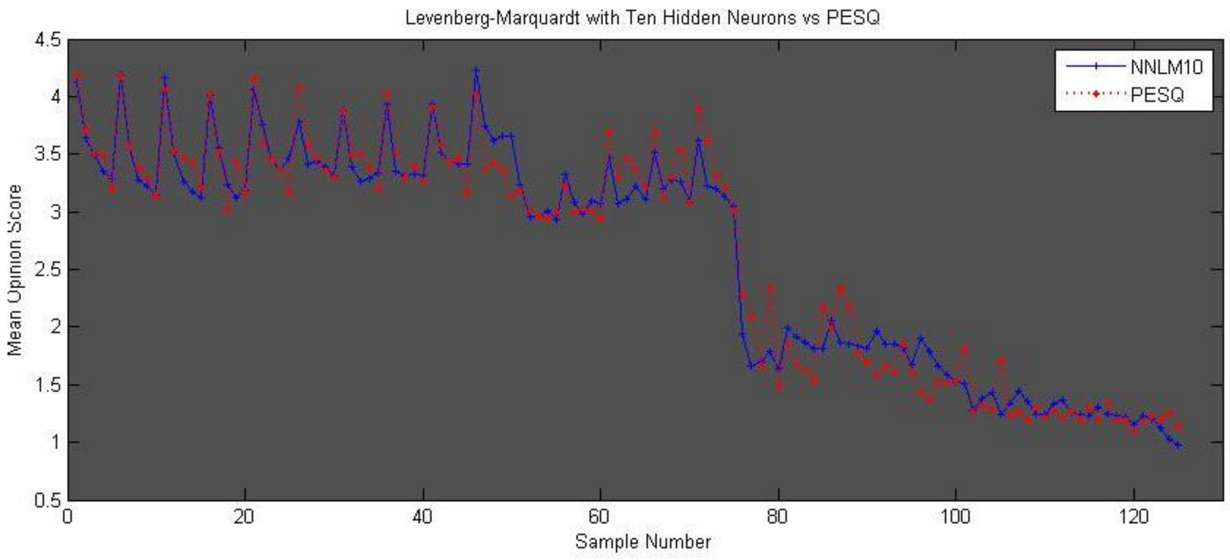


Figure 4.35 – Levenberg-Marquardt with Ten Hidden Neurons vs PESQ

Conclusion

As with the results of the SPEEX experiment, some accuracy is returned with an additional increment to 10 hidden neurons. This paradigm however is still inaccurate with Gradient Descent still unable to predict the dip in the Mean Opinion Score at approximately sample 75. Levenberg-Marquardt remains consistent and viable with previous results but not as accurate as the 5 & 6 hidden neuron paradigms.

4.3 Miscellaneous Results

4.3.1 Perceived effect of network parameters

The effect of the network parameters as perceived by the trained artificial neural networks was investigated for both the Levenberg-Marquardt and Gradient Descent Training, both with six hidden neurons. From Figure 4.36 below it is clear that with the same architecture, Levenberg-Marquardt has a stronger correlation to the original PESQ measurements, with Gradient Descent perceiving all the samples to have a Mean Opinion Score of approximately 3.2.

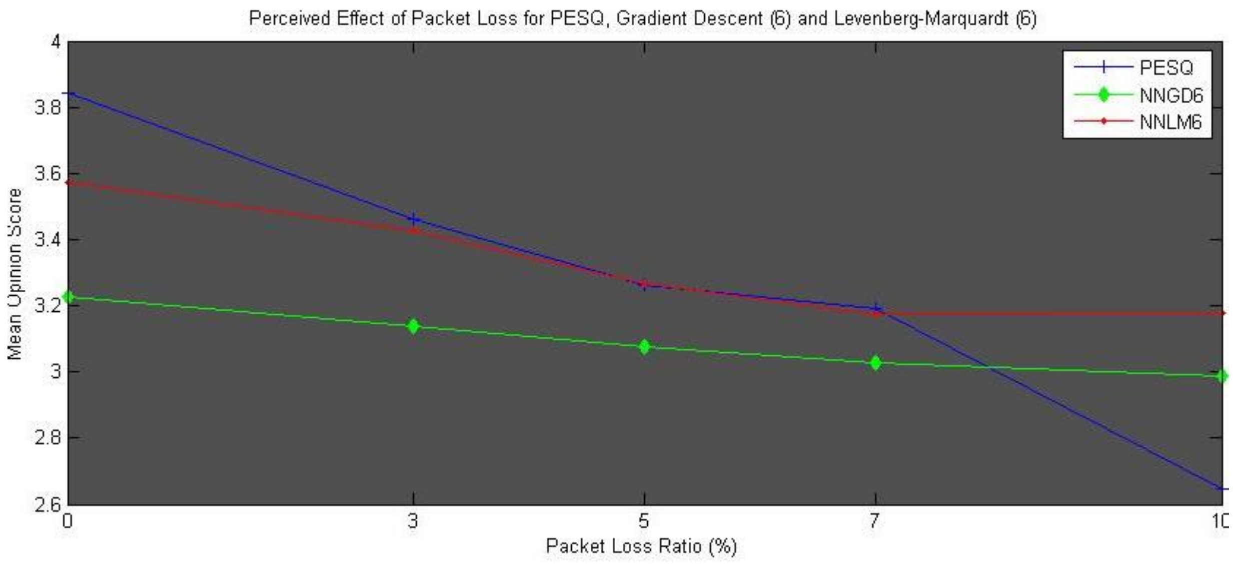


Figure 4.36 – Perceived Effect of Packet Loss

Once again, Figure 4.37 displays a similar result to that of Figure 4.36 with the perceived effect of network delay remaining constant for the artificial neural network trained by Gradient Descent, where as the Levenberg-Marquardt trained network tends to follow the line of the original PESQ measurements.

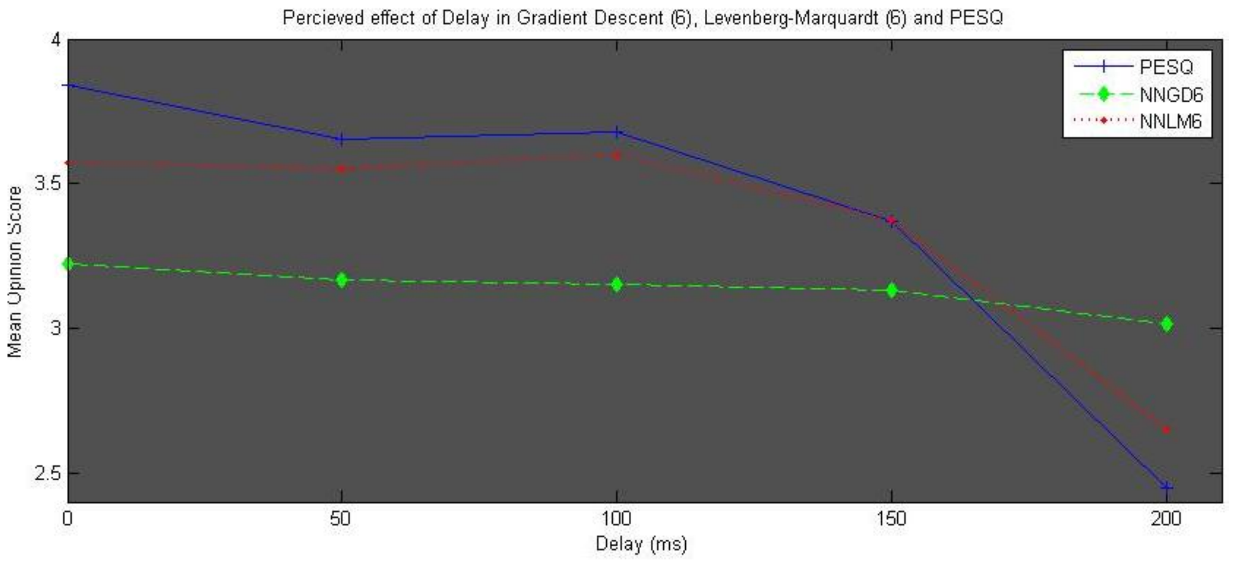


Figure 4.37 – Perceived Effect of Delay

Below in Figure 4.38 Gradient Descent and Levenberg-Marquardt both perform poorly when compared to the original PESQ result when measuring the perceived effect of delay jitter on the mean opinion score of a call. Gradient Descent performed considerably worse than the Levenberg-Marquardt trained network with the first sample being 0.5 away from the target Mean Opinion Score.

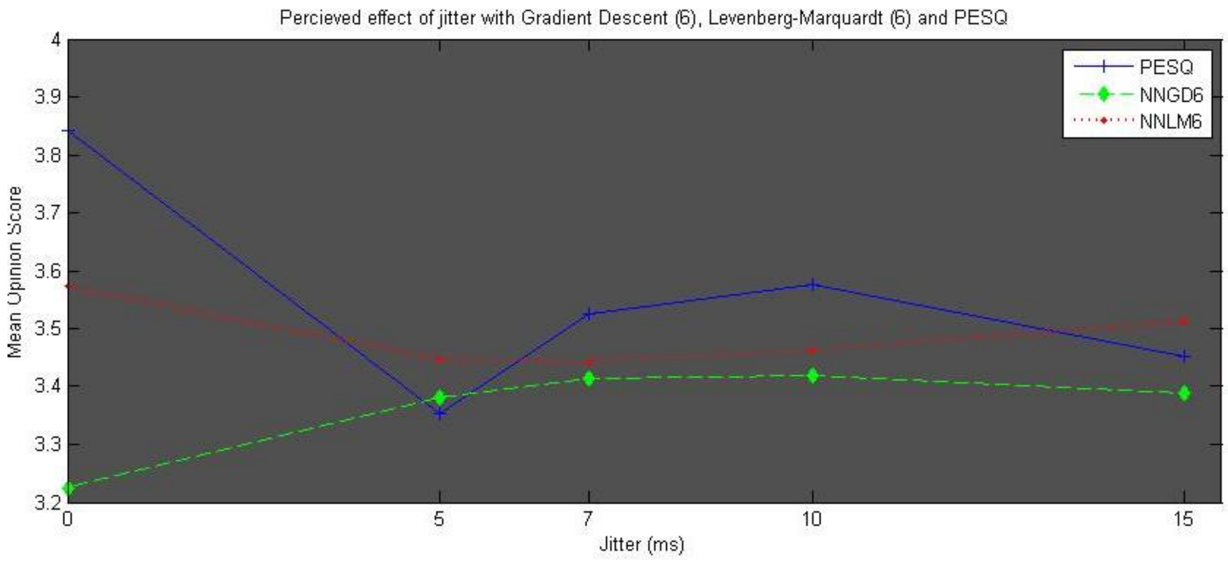


Figure 4.38 – Perceived Effect of Jitter

Conclusion

From the above experiments it can be noted that the perceived effect of delay and packet loss is predicted by Levenberg-Marquardt to a good degree. Gradient Descent proved to be unable to accurately predict the effect of these network parameters. Jitter proved to be more difficult than the other parameters for the neural networks to predict.

4.3.2 Effect of Increasing Maximum Epochs With Gradient Descent

As gradient descent reaches the maximum number of epochs before reaching the minima the maximum number of epochs was increased to see if the Gradient Descent trained network delivers better results when allowed to train longer.

Gradient Descent with Five Hidden neurons and SPEEX codec

For this experiment, the algorithm was allowed to train for double the previous number of iterations to see if that would improve the performance of the Gradient Descent trained network. Table 4.33 below shows the results from the artificial neural network trained with Gradient Descent. The network reached the maximum number of iterations in 29 seconds before it could reach the global minimum, thus, doubling the maximum iterations in this case has not allowed this architecture to complete the training phase. The overall performance measured in the MSE is 0.0538 which only shows a slight increase in performance when compared to the 1000 epoch result of 0.0597 displayed in Table 4.9.

Table 4.33 – Gradient Descent with Five Hidden Neurons (SPEEX)

Epochs:	2000 (maximum)
Time:	29 seconds
Performance Training: (MSE)	0.0451
Performance Validation:	0.0873
Performance Testing:	0.0598
Overall performance:	0.0538
Gradient:	0.0159

Figure 4.39 below shows the correlation of the Mean Opinion Scores of both the network allowed to train for 1000 epochs (NNGD5) and the network allowed to train for

2000 epochs (NNGD5(2000)). It is clear from the Figure that NNGD5 (2000) has a slight performance boost from the longer training period but it still produces inaccurate results.

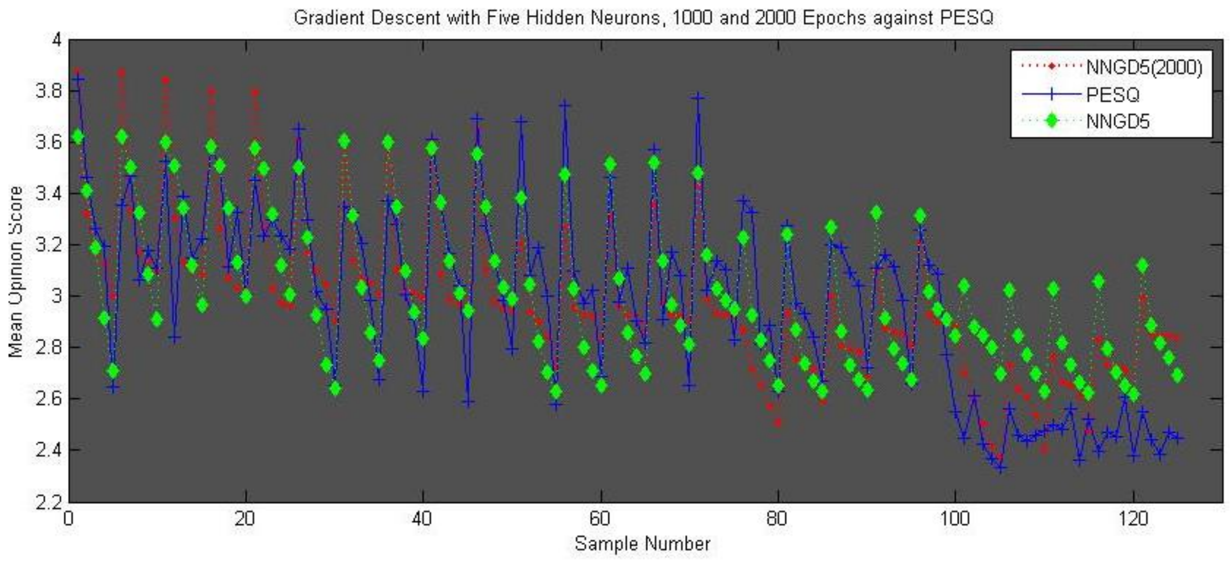


Figure 4.39 – Gradient Descent with Five Hidden Neurons (1000, 2000) vs PESQ

Conclusion

The added processing required by the increase of the maximum number of epochs is shown in Figure 4.39 to have little to no pay-off. While doubling the amount of time spent learning, the performance is not significantly increased.

Gradient Descent with Six Hidden neurons and SPEEX codec

The following results show the effect increasing the maximum number of epochs for the Gradient Descent training when using the six hidden neuron architecture. Table 4.33 shows that Gradient Descent is able to complete the training phase in 1104 iterations of the training process within 16 seconds. This shows great improvement from the previous results in Table 4.32. When compared with these results, Gradient Descent was able to complete training with 896 less iterations than the previous attempt, and took 13 seconds less time to do so.

When compared to the results in Table 4.11 there is great improvement in the performance in that the network trained 2 seconds faster than the previous network, and the MSE of the 2000 epoch trained network was 0.0334 compared to the performance in Table 4.11 of 0.0478.

Table 4.34 – Gradient Descent with Six Hidden Neurons (SPEEX)

Epochs:	1104
Time:	16 seconds
Performance Training: (MSE)	0.0384
Performance Validation:	0.0224
Performance Testing:	0.0212
Overall performance:	0.0334
Gradient:	0.0260

Figure 4.40 below shows the correlation of the Mean Opinion Scores of both the networks allowed to train for 1000 epochs (NNGD6) and the network allowed to train for 2000 epochs (NNGD6(2000)). It is clear from the Figure that NNGD6 (2000) has a great performance boost and is able to accurately follow the line depicted by the benchmark PESQ score.

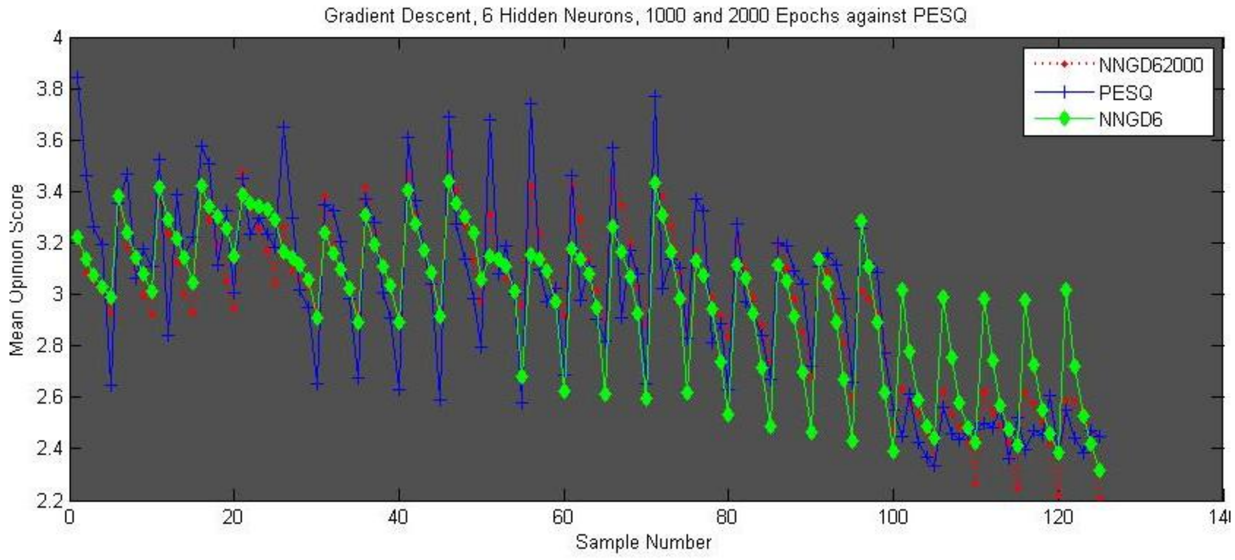


Figure 4.40 – Gradient Descent with Six Hidden Neurons (1000, 2000) vs PESQ

Conclusion
This paradigm shows a significant improvement in the ability of the Gradient Descent trained network but unfortunately still requires significantly more time and processing than the Levenberg-Marquardt trained networks.

Gradient Descent with Six Hidden Neurons and 2000 Epochs against Levenberg-Marquardt with Six Hidden Neurons & PESQ

Figure 4.41 below shows a comparison of both Levenberg-Marquardt with Six hidden neurons (NNLM6) along with Gradient Descent with six hidden neurons when allowed to train for 2000 epochs against PESQ. This table displays that Gradient Descent is able to produce results to compete with the accuracy of Levenberg-Marquardt but still falls slightly short of being on par with the Levenberg-Marquardt paradigm.

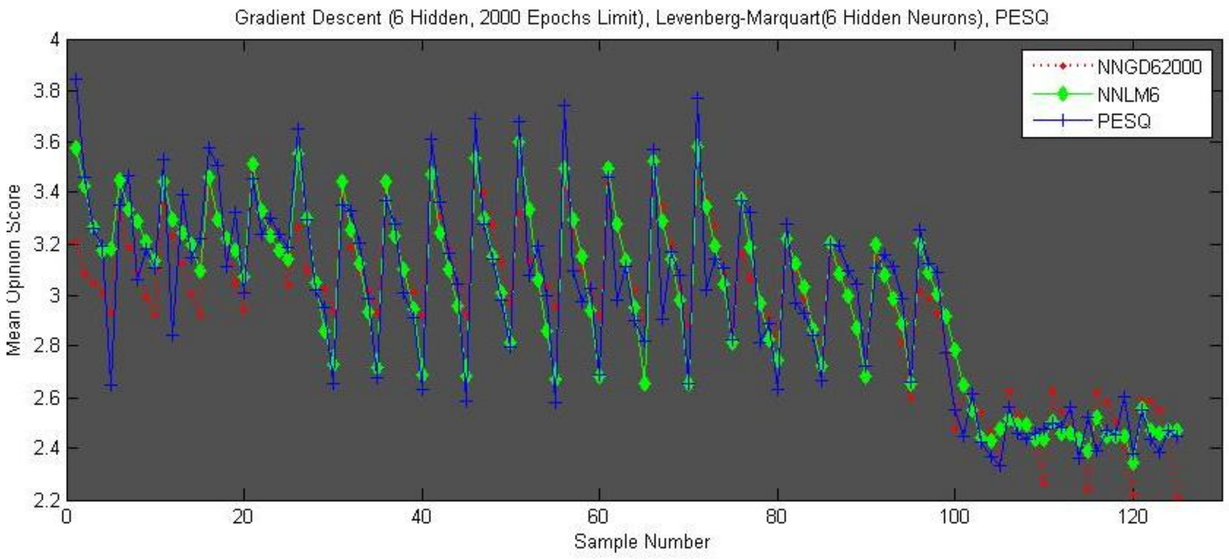


Figure 4.41 – Gradient Descent with Six Hidden Neurons (2000), Levenberg-Marquardt with Six Hidden Neurons vs PESQ

4.4 Training ANN with a maximum of 1000 Epochs with two hidden layers of neurons with data from SPEEX codec

For this experiment the number of epochs will be limited to 1000 as significant improvement was not concluded from the previous experiment. The architecture for the neural network will consist of one input layer of three neurons, two hidden layers of five neurons and one output layer with one neuron. This experiment will show the effect of the hidden layer on the ability of the neural network learning algorithms.

Gradient Descent with two layers of five hidden neurons and SPEEX codec

The following results show the effect increasing the number of hidden layers for the Gradient Descent training when using the five hidden neuron architecture. Table 4.35 shows that Gradient Descent is unable to complete the training phase in 1000 iterations of the training process which takes a total of 17 seconds. When comparing this to the results in Table 4.9 (14 seconds) for one layer of five hidden neurons it shows only a slight increase in training time. Previously, Table 4.15 showed the results for the Gradient Descent training with one layer of ten hidden neurons which took 14 seconds to train also. This shows that having two separate layers does increase the time taken to train the network.

The overall MSE presented below for the dual hidden layered architecture trained by Gradient Descent is 0.0547. When compared with the results from the single layer of five hidden neurons paradigm which has a MSE of 0.597 this does not show a considerable increase in performance. When compared with the results from the single layer of ten hidden neurons paradigm which has a MSE of 0.662 this again does not show a great leap in performance.

The gradients of all three paradigms can also be easily compared and the single layer of five neurons has the poorest performance in this metric, with a gradient of 0.0409. This is improved upon by the ten neuron paradigm which has a gradient of 0.0300. The dual-layered paradigm presented below only has a slight decrease in the Gradient with 0.0260.

Table 4.35 – Gradient Descent with Two Layers of Five Hidden Neurons (SPEEX)

Epochs:	1000 (maximum)
Time:	17 seconds
Performance Training: (MSE)	0.0513
Performance Validation:	0.0949
Performance Testing:	0.0300
Overall performance:	0.0547
Gradient:	0.0260

Figure 4.42 shows the correlation between the PESQ benchmark Mean Opinion Scores and the scores predicted by the Gradient Descent trained neural network with two hidden layers of five neurons. The results show a good correlation between the PESQ scoring and the predicted results with the network showing a drop in accuracy from approximately sample 100. This iteration as with previous paradigms once again suffers from exaggerated peaks in the latter quadrant.

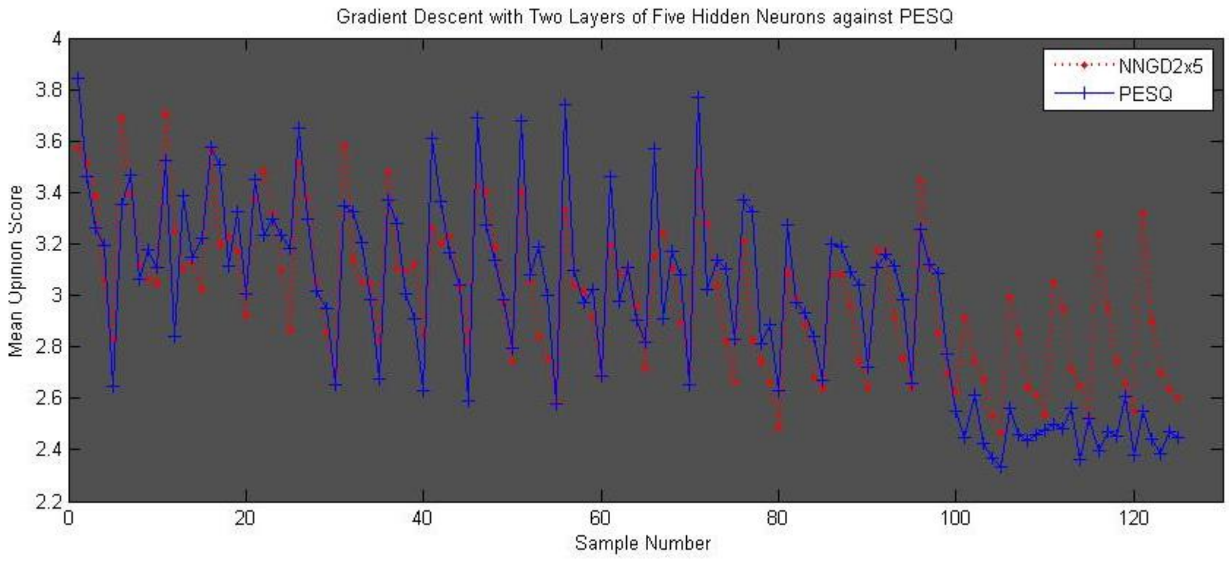


Figure 4.42 – Gradient Descent with two layers of Five Hidden Neurons vs PESQ

Figure 4.43 below shows the previous plots but alongside the original single layer of five neurons trained by Gradient Descent. From this figure it is clear that the paradigm does not improve the ability of Gradient Descent to train a neural network to predict the Mean Opinion Score of a VoIP call.

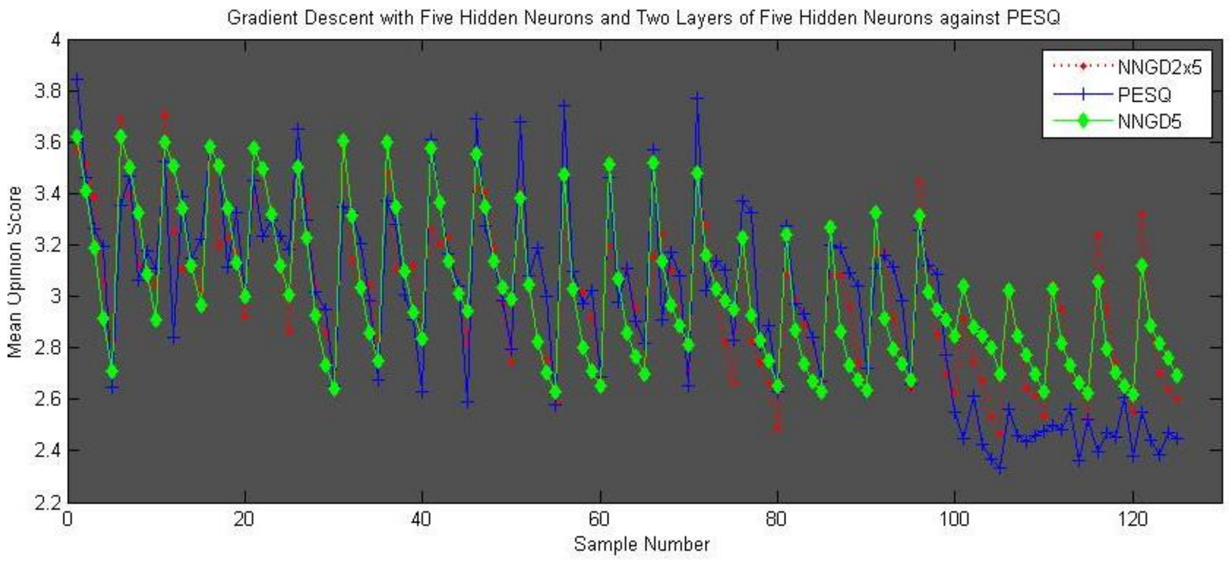


Figure 4.43 – Gradient Descent with Five Hidden Neurons and two layers of Five Hidden Neurons vs PESQ

Levenberg-Marquardt with two layers of five hidden neurons and SPEEX codec

The following results show the effect increasing the number of hidden layers for the Levenberg-Marquardt training when using the five hidden neuron architecture. Table 4.36 shows that Levenberg-Marquardt is able to complete the training phase in 24 iterations of the training process which takes a total of 1 second. When comparing this to the results in Table 4.10 (<1 second) for one layer of five hidden neurons it shows only a slight increase in training time. Previously, Table 4.15 showed the results for the Levenberg-Marquardt training with one layer of ten hidden neurons which took <1 second to train also. This shows that having two separate layers does increase the time taken to train the network slightly.

The overall MSE presented below for the dual hidden layered architecture trained by Levenberg-Marquardt is 0.0186. When compared with the results from the single layer of five hidden neurons paradigm which has a MSE of 0.0160 this shows a slight decrease in performance. When compared with the results from the single layer of ten hidden neurons paradigm which has a MSE of 0.158 this again shows a decrease in performance.

The gradients of all three paradigms can also be easily compared and the single layer of five neurons has the poorest performance in this metric, with a gradient of 0.0671. This is a much lower number than the ten neuron paradigm which has a gradient of 0.0300. The dual-layered paradigm presented below only has a slight decrease in the Gradient with 0.00501.

Table 4.36 – Levenberg-Marquardt with Two Layers of Five Hidden Neurons (SPEEX)

Epochs:	24
Time:	1 second
Performance Training: (MSE)	0.0078
Performance Validation:	0.0548
Performance Testing:	0.0316
Overall performance:	0.0186
Gradient:	0.00501

Figure 4.44 shows the correlation between the PESQ benchmark Mean Opinion Scores and the scores predicted by the Levenberg-Marquardt trained neural network with two hidden layers of five neurons. The results show a good correlation between the PESQ scoring and the predicted results with the network having few anomalies.

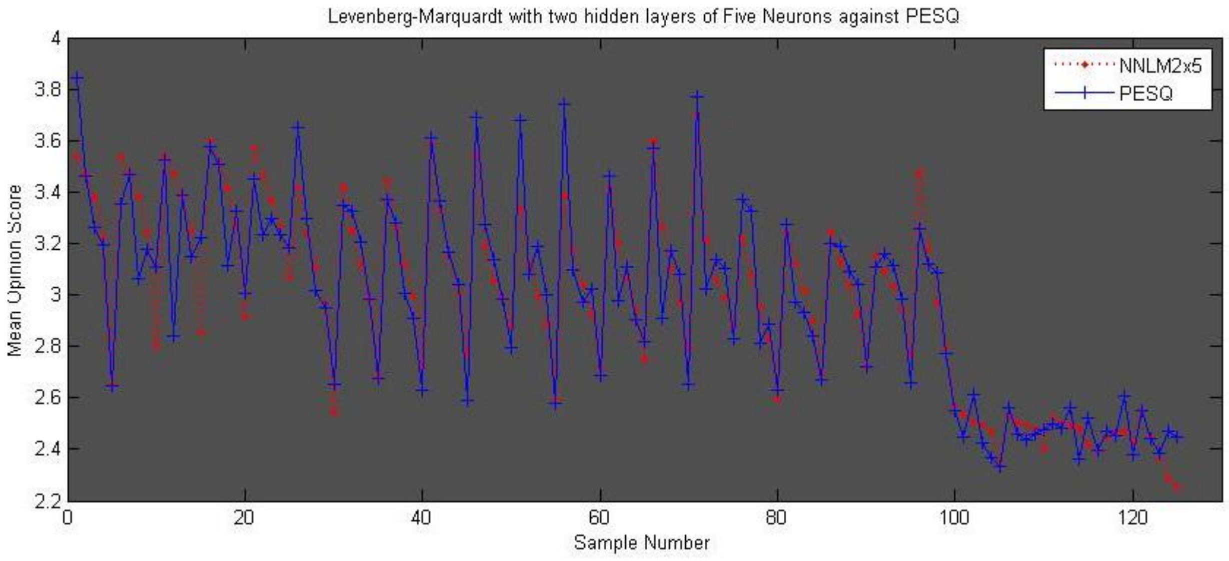


Figure 4.44 – Levenberg-Marquardt with two layers of Five Hidden Neurons vs PESQ

Figure 4.47 below shows the previous plots but alongside the original single layer of five neurons trained by Levenberg-Marquardt. From this figure it is clear that the paradigm does not improve the ability of Levenberg-Marquardt to train a neural network to predict the Mean Opinion Score of a VoIP call significantly.

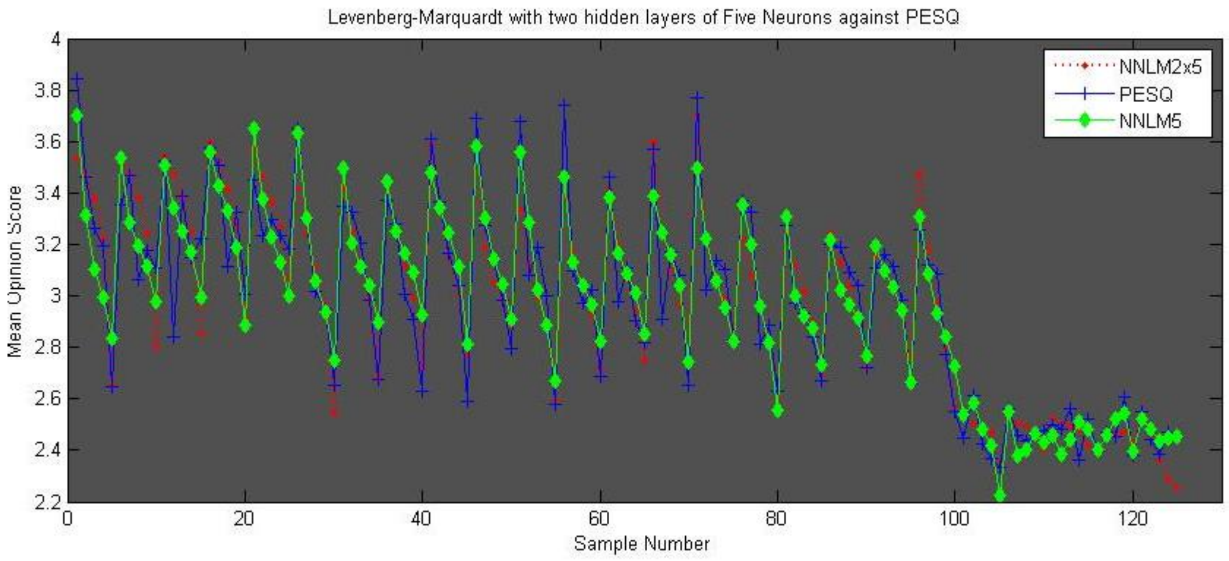


Figure 4.44 – Levenberg-Marquardt with Five Hidden Neurons and two layers of Five Hidden Neurons vs PESQ

5.0 Summary and Conclusions

This project and its results have identified the ability of two separate artificial neural network learning algorithms with various artificial neural network architectures and two commonly deployed voice codecs to predict voice quality over an IP network. The following section considers these results in terms of the research question, hypotheses and project aims which were discussed in section 1 of the report. This summary will also take into consideration the previous research sourced for the literature review found in section 2 of the report. Finally, this section will conclude by identifying the strengths and limitations of the study and identifying areas for future research.

5.1 Research Question

As deployment of Voice over IP services has increased, the demand for PSTN level quality has also grew. Research into the area of voice monitoring highlighted the need for a paradigm to monitor the mean opinion score of a Voice over IP call in real time without compromising accuracy. The use of neural networks to identify patterns in non-linear variables was also researched and found to be appropriate for this problem. From this, the following research question was derived:

> **"How do the learning algorithms of Gradient Descent & Levenberg-Marquardt affect the ability of an artificial neural network to predict speech quality on a communications network running Voice over IP traffic taking into account delay, packet-loss and jitter in comparison to the ITU-T PESQ model?"**

5.1.1 Research Question Results

This section concludes upon the results introduced in section 4 in relation to the research question.

Gradient Descent Results

The ability of the networks trained by the Gradient Descent algorithm performed to a low standard consistently. Many paradigms were tested in order to improve upon the predictions made by the algorithm. The most successful network model found for use with the Gradient Descent algorithm was with six hidden neurons when allowed to train for a maximum of 2000 epochs. This model had the closest correlation to the Mean Opinion Scores plotted by the PESQ algorithm. This still took 16 seconds and 1104 iterations of the network to achieve this level of accuracy with the Gradient Descent algorithm.

For the most part, Gradient Descent was an inefficient methodology for training an artificial neural network to predict the user perceived quality of a Voice over IP call. During simulation, Gradient Descent was used for training on 19 occasions and only twice reached the minimum before the maximum number of epochs was reached. This shows that the algorithm is computationally heavy. The time taken for Gradient Descent to train the artificial neural network ranged from 12 seconds to 29 seconds. This makes the mean time to train an artificial neural network using Gradient Descent 16.1 seconds.

The smallest Mean Squared Error presented by Gradient Descent was 0.0334 which was using the six hidden neuron paradigm when allowed to train for a maximum of 2000 epochs.

Gradient Descent was unable to accurately map the perceived effect of packet loss, delay or jitter on the Mean Opinion Score when compared to PESQ.

Levenberg-Marquardt Results
In parallel to the results produced by Gradient Descent, Levenberg-Marquardt performed to a consistently high standard when used to train an artificial neural network. Many paradigms were tested in order to improve upon the predictions made by the algorithm. The most successful network model found for use with the Levenberg-Marquardt algorithm was with six hidden neurons. This model had the closest correlation to the Mean Opinion Scores plotted by the PESQ algorithm. This model trained, tested and validated in less than a second with only 16 iterations of the network.

The Levenberg-Marquardt training had only one occurrence where simulation took 1 second. All other instances completed within less than a second. The strongest overall MSE came from the paradigm with 10 hidden neurons, even though it did not perform as well as the 6 hidden neuron paradigm on the plot graphs. This is due to slightly better performance during the testing stage as can be seen in Table 4.12 and 4.16.

Levenberg-Marquardt was able to predict the effect of packet loss and delay on the Mean Opinion Score but unable to provide satisfactory results for the effect of jitter when compared to PESQ.

Research Question Conclusion

Overall the Levenberg-Marquardt algorithm performed to a consistently accurate standard when compared to PESQ. Gradient Descent never achieved results that were comparable to the accuracy of PESQ. The Levenberg-Marquardt training generally completed in less than a second where as the mean time for the Gradient Descent trained networks was 16.1 seconds. This shows that the Levenberg-Marquardt algorithm is more suited to the real-time requirements of the problem. Levenberg-Marquardt epoch numbers range from 8 to 27 whereas the least number of epochs taken during Gradient Descent training was 889. This shows that the Gradient Descent algorithm requires a lot more computation for less accurate results than Levenberg-Marquardt.

In conclusion to the research question, the choice of learning algorithm greatly affects the ability of an artificial neural network to predict the quality of a voice call on an IP network but the learning algorithm is limited by the number of neurons in the hidden layer. Out of all paradigms tested; the Levenberg-Marquardt algorithm performed to the highest standard when the artificial neural network had a hidden layer of 6 neurons.

5.2 Project Objectives Results

The following section outlines the results from the primary research objectives identified in section 1.

- Construct an artificial neural network model in MATLAB
  - o **Achieved:** Various architectures of the feedforward artificial neural network were constructed in MATLAB for use in the experiment. These were used to test the ability of the learning algorithms.
- Train artificial neural network with suitable data-sets using the Gradient Descent Algorithm

    - **Achieved:** Two data-sets were used to train the artificial neural network using the Gradient Descent algorithm and the results proved to be consistent using both data-sets. The data-sets were used with 70% of the data for training. The data sets consisted of delay, packet loss and jitter along with PESQ scores. The data sets were based on two commonly deployed codecs: SPEEX & G711a.
- Train artificial neural network with suitable data-set using the Levenberg-Marquardt algorithm
    - **Achieved:** Two data-sets were used to train the artificial neural network using the Levenberg-Marquardt algorithm and the results proved to be consistent using both data-sets. . The data-sets were used with 70% of the data for training. The data sets consisted of delay, packet loss and jitter along with PESQ scores. The data sets were based on two commonly deployed codecs: SPEEX & G711a.
- Test & Validate the network through use of suitable test-data
    - **Achieved:** Using MATLAB all paradigms went through a testing and validation stage during simulation. This allowed the network to be confirmed as operational and fit for purpose. 30% of the data-sets were used for training & validation.
- Collect results from the network after training by each learning algorithm
    - **Achieved:** Mean Opinion Scores, MSE values, Training Time & Number of Epochs were output after each simulation.
- Statistically analyse results
    - **Achieved:** Using the Mean Opinion Scores graphs were plotted to show a comparison between all paradigms and PESQ. MSE values, training time and number of epochs were compared in order to show the efficiency of each model.

5.3 Project Hypotheses Results

The hypotheses detailed in section 1 are reviewed and discussed below.

H1: When the Levenberg-Marquardt algorithm is used to train the artificial neural network the results produced will be closer to the ITU-T PESQ MOS than the Gradient Descent trained artificial neural network.

Result: From the plots gathered in the results section this hypothesis was proven to be true. The results were consistent with existing literature available. The results from the project showed the Levenberg-Marquardt trained paradigm to perform extremely close to the existing PESQ benchmark. The Gradient Descent paradigm at best was inaccurate. The MSE value of the Levenberg-Marquardt trained artificial neural networks was consistently lower than that of the Gradient Descent method.

H2: The Levenberg-Marquardt algorithm will train the artificial neural network in a smaller period of time than the Gradient-Descent method.

Result: Consistent with the existing literature; the Levenberg-Marquardt constantly trained the artificial neural network in less than a second whereas the Gradient Descent method took anywhere up to 29 seconds, with a mean time of 16.1 seconds. The hypothesis was proven to be true.

H3: The Levenberg-Marquardt algorithm will train the artificial neural network more efficiently than the Gradient Descent method.

Result: Based on the results from the number of epochs per simulation for both Levenberg-Marquardt & Gradient Descent, it is clear that Levenberg-Marquardt is a much more efficient training method. Levenberg-Marquardt ranges from 8 to 27 epochs, whereas Gradient Descent only manages to complete training without reaching the maximum number of epochs (1000/2000) twice, thus the research has proven the hypothesis to be true.

H4: The results produced by the artificial neural network in both cases will be less accurate than the results of the ITU-T PESQ model.

Result: The results from the artificial neural network Mean Squared Error never reach exactly 0 so the hypothesis was proven to be true in that they can never truly match up with the PESQ scoring.

H5: The number of neurons in the hidden layer will affect the ability of the networks when training with Backpropagation.

Result: During the experiment many artificial neural network architectures were considered, all focussed on incrementing the hidden layer. These were 1, 2, 3, 4, 5, 6, 7 & 10 hidden neurons. Also tested was the ability of two hidden layers of five neurons with the SPEEX codec. The performance of each learning algorithm was different depending on the number of neurons in the hidden layer. The research from this project was consistent with the existing literature and thus this hypothesis was proven to be true.

5.4 Relation to Previous Studies

Many studies in the field of neural networks to compliment IP networks focus on the use of the artificial neural network but not the learning algorithm chosen. Both key studies identified (Radhakrishnan et al 2010, Sun 2007) focus on the use of the Gradient Descent algorithm. This study has shown the effectiveness of Levenberg-Marquardt algorithm over Gradient Descent to a large degree.

5.5 Irregular Results

During the experimentation and summary it became clear that there were irregularities in the output of the Gradient for the Levenberg-Marquardt algorithm. This will not affect the results of the study as it was simply a secondary metric for comparing the accuracy of the paradigm. Instead focus was maintained on the MSE of each result. For many outputs the Gradient was an irregularly small number which was inconsistent with previous Gradients this was run multiple times and irregularity was found in the numbers.

5.6 Strengths and Limitations

Due to constraints on time and scale of this project there are weaknesses which are identified along with the strengths of the project in the following section.

5.6.1 Project Strengths

The project gives a wide overview of the ability of neural networks as a whole as a problem solving paradigm. The Levenberg-Marquardt algorithm was proven to be much more effective than Gradient Descent which is a key highlight and strength of this project. A large number of paradigms were tested which show a broad range of experimentation was embarked upon. Two codecs were tested in order to prove widespread application of the tested paradigm. The research concluded within the paper was all well backed by previous research. All results were tested against the ITU-T industry standard PESQ algorithm which lends credibility to the findings of the report.

5.6.2 Project Limitations

The largest limitation to the project is that no practical implementation was developed. This is due largely to time constraints and programming skill. Many more architectures could have been investigated along with more codecs tested had there been less time constraints placed upon the project. This would have further proved the viability of the artificial neural network paradigm.

A further limitation of the study was again due to time constraints. The scenarios could have been run multiple times and averaged in order to get a mean score of each result, showing a more vigorous testing methodology.

A limitation of the study is also on the focus of two learning algorithms and the focus on the artificial neural network. Various forms of Gradient Descent exist and only the most basic version of the algorithm was tested. Random and recurrent neural networks were also not taken into account in this study.

Other network parameters or call quality factors could have been taken into account, such as gender and hardware limitations this was due to time and availability of the hardware required.

5.7 Future Research

As documented in the Project Limitations section a practical solution was not developed. This remains a highly interesting research area. This would carry on from the conclusions of this report and further experimentations would be carried on physical hardware with real traffic rather than simulations.

The learning algorithms used in this project were limited due to time but future research will be carried out into algorithms such as Gradient Descent with Momentum and Levenberg-Marquardt with Adapative Momentum (Ampazis and S. J. Perantonis, 2000).

5.8 Conclusion

This report has identified the ability of two artificial neural network learning algorithms to predict the quality of a call on an IP network. It has also identified the best model for the learning algorithm to train. It can be concluded that artificial neural networks are a viable real-time solution to monitoring the Mean Opinion Score as perceived by the listener when an efficient algorithm, such as Levenberg-Marquardt is used.

The research in this document would be of interest to academics interested in IP networks, Voice over IP communications, Artificial Intelligence and Neural Networks. In the industry the research in this paper may be useful to service providers, network technicians and developers.

It is therefore concluded that this experimental, simulation based investigation into artificial neural networks despite having limitations is a credible look into the application of artificial intelligence into IP networks. The use of the ITU-T PESQ model as a benchmark gives an industry standard credibility to the findings of this report.

6.0 References

A. E. Mahdi. (2007). *Voice Quality Measurement in Modern Telecommunication Networks.* 14th International Workshop on Systems, Signals and Image Processing, Maribor. pp25-33.

Almeroth, C., Belding, K., Elizabeth, M., Jardosh, A. & Suri, S. (2003). *Towards Realistic Mobility Models for Mobile Ad Hoc Networks*. Proceedings of the 9th annual international conference on Mobile computing and networking, 14-19, 2003. ACM, pp217-229.

Ananth Ranganathan. (2004). *The Levenberg-Marquardt Algorithm*.[online] Available: http://www.ananth.in/docs/lmtut.pdf. Last accessed 15th Oct 2010.

Bakircioglu, Tocak, (2000). *Survey of neural network applications.* European Journal of Operational Research. pp319-331.

Bart Duysburgh, Stefaan Vanhastel, Bart De Vreese, Cristinel Petrisor, Piet Demeester. (2001). *On the influence of best-effort network conditions on the perceived speech quality of VoIP connections*. Proceedings of the 10th International Conference on Computer Communications and Networks, Arizona. pp334-350.

Bogdan M. Wilamowksi, Hao Yu. (2010). *Neural Network Learning without Backpropagation.* IEEE TRANSACTIONS ON NEURAL NETWORKS, Vol 21 (11). pp1793-1804.

Bur Goode. (2002). *Voice Over Internet Protocol (VoIP).* Proceedings of the IEEE. Vol 90 (9), pp1495-1517.

Cearely D.W, Fenn J, Plummer D.C. (2005). *Gartner's Positions on the Five Hottest IT Topics and Trends in 2005* [online]. Gartner, Inc. Available: http://www.gartner.com/resources/125800/125868/gartners_positi.pdf. Last accessed 24th Oct 2010.

Cisco Systems. (2006a) *Understanding Delay in Packet Voice Networks.* [online]. Available: http://www.cisco.com/en/US/tech/tk652/tk698/technologies_white_paper09186a00800a8993.shtml. Last accessed: 12th December 2010.

Cisco Systems. (2006b) *Understanding Jitter in Packet Voice Networks.* [online]. Available: http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a00800945df.shtml. Last accessed: 12th December 2010.

Cisco Systems. (2009) *What is Voice over IP and What can it do for your business?* [online]. Available: http://www.cisco.com/en/US/prod/voicesw/networking_solutions_products_genericcontent0900aecd804f00ce.html. Last accessed: 10th Jan 2011.

Chester, Michael. (1993). *Neural Networks: A Tutorial*. Prentice Hall, New Jersey.

D.B.L. Bong, J.Y.B. Tan, A.R.H. Rigit. (2010). *Optimization of the Backpropagation Hidden Layer by Hybrid K-means-Greedy Algorithm for Time Series Prediction*. Symposium on Industrial Electronics and Applications, Malaysia. pp669-674.

Demichelis & Chimento. (2002). *IP Packet Delay Variation Metric for IP Performance Metrics*. RFC 3393. Internet Engineering Task Force.

Dreyfus, G (2005). *Neural Networks: Methodology and Applications* . Laboratoire d'Electronique, ESPCI: Springer Verlag. pp1-79.

Dudman, J. (2006). *Voice over IP: what it is, why people want it, and where it is going.* [online]. JISC Technology and Standards Watch. Available: http://www.jisc.ac.uk/whatwedo/services/techwatch/reports/horizonscanning/hs0604.aspx. Last accessed 20th Oct 2010.

Durkin, J. (2003*). Voice-Enabling the Data Network:H.323, MGCP,SIP,SLAs, and Security*. 1st ed. Indianapolis. IN: Cisco Press.

Eberhart, R.C., Dobbins, R.W. (1990). *Early Neural Network Development History: The Age of Camelot.* IEEE Engineering in Medicine and Biology Magazine. Vol. 9 (3), pp15-18.

Erol Gelenbe. (2003). *Sensible decisions based on QoS.* Computational Management Science. Vol 1 (1), pp1-14.

Erol Gelenbe, Khaled Hussain. (2002). *Learning in the Multiple Class Random Neural Network.* IEEE Transactions on Neural Networks. Vol 13 (6), pp1257-1267.

Erol Gelenbe, Ricardo Lent, Arturo Nunez. (2004). *Self-Aware Networks and QoS*. Proceedings of the IEEE. Vol 92 (9), pp1478-1489.

Gino J. Coviello. (1979). *Comparative Discussion of Circuit- vs. Packet-Switched Voice.* IEEE Transactions On Communications. pp1153-1161.

Gupta M.M, Liang J, Homma H (2003). *Static and Dynamic Neural Networks.* New Jersey: John Wiley & Sons.

Gorur P.Y. (2006). *Converged Network Management: Challenges and Solutions*. Optical Fiber Communications Conference. pp10-20.

Hakan Bakircioglu, Taskin Kocak. (2000). *Survey of random neural network* applications. European Journal of Operational Research. Vol 126 (2), pp319-330

Heng Guo & Saul B. Gelfand. (1991). *Analysis of Gradient Descent Learning Algorithms for Multilayer Feedforward Neural Networks*. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, Vol 38 (8). pp883-895.

Howard B. Demuth, Mark Beale. (1997). *Matlab Neural Network Toolbox*. Cochituate Place, Massachusetts, The Mathworks.

Hiroki Furuya, Shinichi Nomoto, Hideaki Yamada, Norihiro Fukumoto & Fumiaki Sugaya. (2003). *Experimental Investigation of the Relationship between IP Network Performances and Speech Quality of VoIP.* 10th International Conference On Telecommunications. pp543-553.

Iban Lopetegui, Rolando Antonio Carrasco and Said Boussakta. (2010). *Speech Quality Prediction in VoIP Concatenating Multiple Markov-Based Channels*. Sixth Advanced International Conference on Telecommunications, Barcelona. pp226-231.

International Telecommunication Union. (1996). *SERIES P: TELEPHONE TRANSMISSION QUALITY Methods for objective and subjective assessment of quality. ITU-T Recommendation P.800. ITU-T International Telecommunication Union. (2001). Perceptual evaluation of speech quality (PESQ).* ITU-T Recommendation P.862. ITU-T.

International Telecommunication Union. (2003). *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS International telephone connections and circuits – General Recommendations on the transmission quality for an entire international telephone connection.* ITU-T Recommendation G.114. ITU-T.

International Telecommunication Union. (2009a). *The E-model: a computational model for use intransmission planning*. ITU-T Recommendation G.107. ITU-T.

International Telecommunication Union. (2009b). *SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services – Systems and terminal equipment for audiovisual services.* ITU-T Recommendation H.323. ITU-T

J. Janssen, D. Vleeschauwer, M. Büchli, G.Petit. (2002). *Assessing voice quality in packet-based telephony.* IEEE Internet Computing. Vol. 3 (6). pp48-56.

J.J. Hopfield. (1982). *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the National Academy of Sciences of the United States of America. Vol. 79 (8), pp2254-2258.

Justus F.M. Broß, Christoph Meinel. 2008, *Can VoIP Live up to the QoS Standards of Traditional Wireline Telephony?*. The Fourth Advanced International Conference on Telecommunications. pp126-132.

Kartakapoulos S.V. (1997). *Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications.* Wiley-IEEE Press

K. Vlahodimitropoulos, E. Katsaros. (2007). *Monitoring the end user perceived speech quality using the derivative mean opinion score (MOS) key performance indicator.* 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Athens. pp1-5.

Keepence B. (1999*). Quality Of Service for Voice over IP.* IEE Colloquium on Services Over the Internet - What Does Quality Cost?. pp1-4.

L. Sun. (2004). *Speech Quality Prediction for Voice over Internet Protocol Networks.* Ph.D Thesis, Plymouth University

Leandro Carvalho, Edjair Mota, Regeane Aguiar, Ana F. Lima, José Neuman de Souza & Anderson Barreto. (2005). *An E-Model Implementation for Speech Quality Evaluation in VoIP Systems*. Proceedings of the 10th IEEE Symposium on Computers and Communications. pp933-938.

Li Zheng, Liren Zhang, Dong Xu. (2001). *Characteristics of Network Delay and Delay Jitter and its Effect on Voice over IP*. IEEE Conference on Communications, Helsinki. pp122-127.

Lijing Ding, Rafik A. Goubran. (2003a). *Speech Quality Prediction in VoIP Using the Extended E-Model.* IEEE Global Telecommunications Conference, 2003. pp3974-3979.

Lijing Ding, Rafik A. Goubran. (2003b). *Assessment of Effects of Packet Loss on Speech Qualityin VoIP*. The 2nd IEEE International Workshop on Audio & Visual Enviroments and Their Applications, Haptic. pp49-55.

Linhui Liu, Jie Chen, Lixin Xu. (2008). *Realization and application research of BP neural network based on MATLAB.* International Seminar on Future BioMedical Information Engineering, Hubei. pp130-134.

Lingfen Sun, Ifeachor E.C. (2002). *Perceived Speech Quality Prediction for Voice over IP-based Networks*. IEEE International Conference on Communications. p2573-2577.

Lingfen Sun, Ifeachor E.C. (2006). *Voice Quality Prediction Models and Their Application in VoIP Networks.* IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 8 (4). pp809-821.

L. Yamamoto, J.G. Beerends. (1997) *Impact of network performance parameters on the end-to-end perceived speech quality*. Proceedings. Expert ATM Traffic Symposium. pp180-191

Manjunath. T. (2009). *Limitations of Perceptual Evaluation of Speech Quality on VoIP Systems.* IEEE International Symposium on Broadband Multimedia Systems & Broadcasting, Bilbao. pp1-7.

Manousos, Apostolacos, Grammatikakis, Mexis, Kagklis & Sykas. (2005). *Voice-Quality Monitoring and Control for VoIP*. IEEE Internet Computing. pp35-43.

Martin T. Hagan, Howard B. Demuth, Mark Beale. (1996). *Neural Network Design*, PWS, Boston, Massachusetts

Martin T. Hagan, Howard B. Demuth. (1999). *Neural Networks for Control*. Proceedings of the American Control Conference. p1642-1656.

Martin T. Hagan, Howard B. Demuth, Mark Beale. (2010). *Neural Network Toolbox 7 User's Guide*. Cochituate Place, Massachusetts, The Mathworks

Martin T. Hagan, Mohammad B. Menhaj. (1994). *Training Feedforward Networks with the Marquardt Algorithm.* IEEE Transactions of Neural Networks. Vol. 5 (6), pp989-993.

Masao Masugi. (2002). *QoS Mapping of VoIP Communication using Self-Organizing Neural Network.* IEEE Workshop on IP Operations and Management. pp13-17.

Ming-Qing Ling, Wei-Wei Liu. (2008). *Research on Intrusion Detection Systems Based on Levenberg-Marquardt Algorithm.* Proceedings of the Seventh International Conference on Machine Learning and Cybernetics. pp3684-3688.

Mohammad Abareghi, M. Mehdi Homayounpour, Mehdi Dehghan, and Anahita Davoodi. (2008). *Improved ITU-P.563 Non-Intrusive Speech Quality Assessment Method For Covering VOIP Conditions*. 10th International Conference on Advanced Communication Technology, Gangwon-Do. pp354-359.

N. Ampazis and S. J. Perantonis. (2000). *Levenberg-Marquardt Algorithm with Adaptive Momentum for the Efficient Training of Feedforward Networks*. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Athens. pp126-132.

Nicola Baldo, Paolo Dini, Jaume Nin-Guerrero. (2010). *User-driven Call Admission Control for VoIP over WLAN with a Neural Network Based Cognitive Engine.* 2nd International Workshop on Cognitive Information Processing. pp52-56.

Nielsen. (1989). *Theory of the backpropagation neural network.* International Joint Conference on Neural Networks, Washington DC. pp593-606.

Pietro Paglierani, Dario Petri. (2007). *Uncertainty evaluation of speech quality measurement in VoIP systems.* International Workshop on Advanced Methods for Uncertainty Estimation in Measurement, Italy. pp104-109.

Phillips, T., Sephton, J. & Hone, J. (2009). *Deploying VoIP in real life.* Team Register.

Radhakrishnan, K, Larijani, H. (2010). *Voice Quality in VoIP Networks Based on Random Neural Networks.* Ninth International Conference on Networks, Menuires. pp89-93.

Radhakrishnan, K., Larijani, H., Buggy, T. (2010). *A Non-Intrusive Method to Assess Voice Quality Over Internet*. 2010 International Symposium on Performance Evaluation of Computer and Telecommunication Systems. pp380-386.

Robert G. Escalante, Heidar A. Malki. (2006). *Comparison Of Artificial Neural Network Architectures And Training Algorithms For Solving The Knight's Tours.* 2006 International Joint Conference on Neural Networks. pp1447-1450.

Sanghyun Chi, Baxter F. Womack. (2009). *Predicting the Quality of Voice over IP Networks.* IEEE International Workshop Technical Committee on Communications Quality & Reliability, Florida. pp1-4.

Sofiene Jelassi, Habib Youseff, Guy Pujolle. (2009). *Parametric Speech Quality Models for Measuring the Perceptual Effect of Network Delay Jitter.* IEEE 34th Conference on Local Computer Networks, Switzerland. pp193-201.

Telegeography, PriMetrica Inc. (2009a). *VoIP in Europe.* [online] Available: http://www.telegeography.com/cu/article.php?article_id=29486. Last accessed: 14th Jan 2011.

Telegeography, PriMetrica Inc. (2009b). *Skype's share of the international long-distance pie on the increase.* [online] Available: http://www.telegeography.com/cu/article.php?article_id=27800&email=html. Last accessed: 1st Nov 2010.

Wallace K. (2009). *Cisco Voice over IP*. 3rd ed. Indianapolis, IN: Cisco Press.

Wolfgang Kampichler, Karl Michael Goeschka. (2001). *Measuring Voice Readiness of Local Area Networks.* Global Telecommunications Conference, Texas. pp2501-2507.

7.0 Appendixes

Appendix A:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 1: Gradient Descent with One Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 1;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix **B:**

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 7: Levenberg-Marquardt with One Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 1;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm'; % Levenberg-Marquardt
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix C:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 2: Gradient Descent with Two Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 2;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix D:
```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 8: Levenberg-Marquardt with Two Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 2;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm'; % Levenberg-Marquardt
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix E:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 3: Gradient Descent with Three Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 3;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix F:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 9: Levenberg-Marquardt with Three Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 3;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm'; % Levenberg-Marquardt
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix G:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 4: Gradient Descent with Four Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 4;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix H:
```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 13: Levenberg-Marquardt with Ten Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 10;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm'; % Levenberg-Marquardt
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix I:

```matlab
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 4: Gradient Descent with Five Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 5;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix J:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 3: Levenberg-Marquardt with Five Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

hiddenLayerSize = 5;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 20/100;
net.divideParam.testRatio = 10/100;

net.trainFcn = 'trainlm';  % Levenberg-Marquardt algorithm selected
net.trainParam.epochs = 1000; %Let the maximum number of iterations be 2000
net.performFcn = 'mse';  % Mean squared error will be used to measure the
                         % performance of the network.

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix K:
```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 5: Gradient Descent with Six Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 6;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

```
Appendix L:
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 4: Levenberg-Marquardt with Six Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 6;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

net.trainFcn = 'trainlm';  % Levenberg-Marquardt algorithm selected
net.trainParam.epochs = 1000; %Let the maximum number of iterations be 2000
net.performFcn = 'mse';  % Mean squared error will be used to measure the
                         % performance of the network.

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix M:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 1a: Gradient Descent with Seven Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 7;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix N:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 1b: Levenberg-Marquardt with Seven Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 7;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm'; % Levenberg-Marquardt
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix O:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 6: Gradient Descent with Ten Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 10;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix P:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 13: Levenberg-Marquardt with Ten Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network
hiddenLayerSize = 10;
net = feedforwardnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm'; % Levenberg-Marquardt
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix Q:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 5: Gradient Descent with two layers of Five Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network

net = feedforwardnet([5 5]); %This function sets the net to have two layers
                             %of five hidden neurons.

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'traingd'; % Gradient-Descent
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```

Appendix R:

```
% Student JJ 10-11
% Glasgow Caledonian University
% Matriculation: 200604642
% Honours Project - 2010/11
%
% "A Comparison of Artificial Neural Network Learning Algorithms
%  To Predict Voice over IP Call Quality"
%
% Script 6: Levenberg-Marquardt with two layers of Five Hidden Neurons
%
% This script assumes these variables are defined:
%
%   Inputs - input data.
%   TargetOut - target data.

inputs = Inputs';
targets = TargetOut';

% Create a Fitting Network

net = feedforwardnet([5 5]); %This function sets the net to have two layers
                             %of five hidden neurons.

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm'; % Levenberg-Marquardt
net.trainParam.epochs = 1000;
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)
```