ATHLONE INSTITUTE OF TECHNOLOGY

Multi-Resolution Pre-Processing for Pattern Recognition in Images and Audio Signals.

Author:
Noha MANSOR

Supervisors:
Dr. Mark DALY
Dr. Ronan FLYNN

A Master thesis submitted in partial fulfilment of the requirements for the degree of MSc. (Research) in Software Engineering

in the

Software Research Institute
Department of Electronics & Informatics

October 5, 2020

Declaration of Authorship

I, Noha MANSOR, declare that this research project titled, Multi-Resolution Pre-Processing for Pattern Recognition in Images and Audio Signals. and the work presented in it are my own. I confirm that:

- This work has being done wholly or mainly while in candidature for a research degree.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this progress report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the progress report is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

To find signals in data, we must learn to reduce the noise - not just the noise that resides in the data, but also the noise that resides in us. It is nearly impossible for noisy minds to perceive anything but noise in data.

Stephen Few

Abstract

With the rapid growth of technology and the proliferation of data in this "digital age", most current data (imaging and audio) applications require higher data resolution, higher data transmission rates, and better compression techniques to meet the ever increasing demands placed on them. In the case of data compression, an ideal system must produce good quality of data with high compression ratios while minimising the computational overhead. In other words, bandwidth cost money, therefore, the transmission and storage of information become costly. However, if we can use less data, both transmission and storage become cheaper.

The Haar Wavelet Transform (HWT) has been chosen, here, to fulfil the data compression requirement of this research. The HWT is a mathematical tool that is ideally suited for hierarchically decomposing image information. For example, it is the preferred method by the JPEG format to produce encoded images at higher compression ratios. In addition, it can also be used to remove irrelevant/redundant image data to facilitate the recording, transmission, and searching of data effectively. HWT based image compression is computationally efficient and is symmetric in nature (both the forward and the inverse transforms have the same complexity) allowing for the building of fast compression and decompression routines.

The aim of this research is to determine optimum compression rates for data using Machine Learning methods; typically, Artificial Neural Network (ANN). The MNIST dataset images and the TI46 database audio are subject to a HWT transformation and a signature of the information is constructed by removing irrelevant/redundant data by means of a cut-off function. Multiple signatures for images and audio are built for differing cut-offs are then input into a neural network. Their performances are then compared to find the optimum HWT based compression. Which, it has made the input vectors of the network much more compact and thus easier to establish relationships between them, leading to a rapid and effective classification of the signal.

Acknowledgements

First and foremost, I would like to thank God for giving me the strength, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily.

I would like to express my deep gratitude to Dr. Mark Daly and Dr. Ronan Flynn, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. Their guidance helped me in research and writing of this thesis. I could not have imagined having a better advisor and mentor for my research study.

I have great pleasure in acknowledging my gratitude to my colleagues at Athlone Institute of Technology, who have, in their own ways, kept me going on my path to success, assisting me as per their abilities, in whatever manner possible and for ensuring that good times keep flowing.

A special mention of thanks to my friends the in SRI; Ashima, Debora, Emer, Conor, Jonny, Thiago, Eoghan, Andrzej, and many more for their constant support and cooperation during my research. Their timely help and friendship will always be remembered.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family. To my parents, thank you for your unconditional love and support which has meant the world to me. To my brothers Amr and Mohamed, thanks for their love and affection. To my best Friend Heba Eid, thank you for friendship and moral support. To my children Mohamed, Mera and Omar, you are my inspiration to achieve greatness. To my husband Haitham Gaballa, thank you for being there for me at the end of the day. Your love is what has gotten me through when I wanted to give up.

I thank them for putting up with me at difficult moments when I felt stumped and for goading me on to follow my dream of getting this degree. This would not have been possible without their unwavering and unselfish love and support for me at all times.

Contents

List of Figures

List of Tables

List of Abbreviations

HWT	**H**aar **W**avelet **T**ransform
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
ANN	**A**rtificial **N**eural **N**etwork
MNIST	**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology
TI46	**T**exas **I**nstruments of **46** words
PNG	**P**ortable **N**etwork **G**raphics
TIFF	**T**agged **I**mage **F**ile **F**ormat
TGA	**T**ruevision **G**raphics **A**dapter
GIF	**G**raphics **I**nterchange **F**ormat
ALAC	**A**pple **L**ossless **A**udio **C**odec
MPEG	**M**oving **P**icture **E**xperts **G**roup
AI	**A**rtificial **I**ntelligent
SAINT	**S**ymbolic **A**utomatic **INT**egrator
SIR	**S**emantic **I**nformation **R**etrieval
(GPS	**G**eneral **b**roblem **S**olver
IQ	**I**ntelligence **Q**uotient
ML	**M**achine **L**earning
ALife	**A**rtificial **Life**
KDD	**K**nowledge **D**iscovery in **D**atabases
IBM	**I**nternational **B**usiness **M**achines Corporation
MIT	**M**assachusetts **I**nstitute of **T**echnology
CMU	**C**arnegie **M**ellon **U**niversity
LISP	**LIS**t **P**rocessor
SOM	**S**elf **O**rganizing **M**ap
SGM	**S**elf **G**enerating **M**ap
FP-Growth	**F**requent **P**attern- **G**rowth
BMU	**B**est **M**atched **U**nit
SVM	**S**upport **V**ector **M**achines
BNN	**B**iology **N**eural **N**etwork
LZW	**L**empel-**Z**iv-**W**elch
RLE	**R**un **L**ength **E**ncoding
BMP	**B**it**M**a**P**

PDF	**P**ortable **D**ocument **F**ormat
GUI	**G**raphical **U**ser **I**nterface
MP3	**MP**EG-1 Audio Layer-**3**
MP4	**MP**EG-**4** part 14
WMV	**W**indows **M**edia **V**ideo
PSNR	**P**eak **S**ignal-to-**N**oise **R**atio
DTW	**D**ynamic **T**ime **W**arping
HMM	**H**idden **M**arkov **M**odel
ASR	**A**utomatic **S**peech **R**ecognition
STFT	**S**hort-**T**ime **F**ourier **T**ransformation
MFCCs	**M**el **F**requency **C**epstral **C**oefficient**s**
DCT	**D**iscrete **C**osine **T**ransform
ReLU	**Re**ctified **L**inear **U**nit
SELU	**S**caled **E**xponential **L**inear **U**nits
GPU	**G**raphics **P**rocessing **U**nit
CPU	**C**entral **P**rocessing **U**nit
FWT	**F**ast **W**avelet **T**ransformation
DWT	**D**iscrete **W**avelet **T**ransformation
QMFs	**Q**uadrature **M**irror **F**ilter**s**
CQFs	**C**onjugate **Q**uadrature **F**ilter**s**
AWA	**A**dapted **W**aveforms **A**nalysis

Chapter 1

Introduction

1.1 Background

Typical data objects contain redundant data; i.e. data that is not necessary to convey the meaning of the object. In an image, redundant data could manifest itself as an image border. In a speech signal, redundant data can be the presence of temporal and frequency variability, such as intra speaker variability and inter speaker variability. Using data compression techniques, unnecessary data can be erased without adversely affecting data quality.

Data compression techniques can be either lossy or lossless. Lossless compression techniques, by allowing exact reconstruction of original parent data, don't add noise to the data. For images, lossless compression is principally employed in medical image processing [1] and satellite imaging, where accurate reproduction of the images is an essential requirement. All PNG, TIFF, TGA, GIF image formats normally employ lossless compression techniques. For audio signals, lossless compression is a necessity for high-quality audio reproduction [2]. ALAC (Apple Lossless Audio Codec), RealPlayer and Monkey's Audio formats use lossless audio compression techniques. One of the most widely used lossless compression techniques is entropy coding [3]. Entropy coding creates and assigns a unique prefix-free code to each unique symbol that occurs in the input. Huffman encoding and arithmetic coding are well known entropy encoding methods applied in JPEG and MPEG coding standards.

Lossy compression techniques [4] can produce high compression ratios but the decompressed data is only an approximation of the original. If, as in image and audio data, there is significant redundant data then lossy methods are suitable. There are two basic lossy compression schemes; 1) the lossy transform codecs, samples of picture or sound are taken, chopped into small segments, transformed into a new basis space, and quantised. The resulting quantised values are then entropy coded. 2) lossy predictive codecs, previous and/or subsequent decoded data is used to predict the current sound sample or image frame. The error between the predicted

data and the real data, together with any extra information needed to reproduce the prediction, is then quantised and coded. In some systems the two techniques are combined, with transform codecs being used to compress the error signals generated by the predictive stage.

Lossy compression techniques are mostly used in Internet transfer and media streaming, including multimedia audio, video and still images data.

The Haar Wavelet Transform (HWT) [5] is an effective tool used in lossy compression to remove redundant data because of its ability to decompose information hierarchically.

The Artificial Neural Network (ANN) [6] has developed into a very Adaptable Machine Learning tool with many areas of application from image processing to signal processing. There are different types of neural networks that have been developed over the years. It is a self-adapting algorithm, in other words, it learns a problem with examples and then recognises patterns, trends and hidden relationships in similar data-related problems.

1.2 Overview of Proposed Research

In this project, we implement a HWT-based data transform on data being fed into an ANN, specifically, to recognise images of handwritten digits from 0 to 9 from the MNIST data-set [7] and to recognise audio of spoken words from the TI46 [8] speech data-set. Recognition transformer results using different activation functions and different numbers of hidden layers in the ANN are compared and analysed for accuracy, robustness and efficiency.

1.3 Research Aim

The aim of this project is to develop an efficient signature or fingerprint for data by using lossy data compression, particularly, wavelet techniques. This algorithm will use to remove unnecessarily information from the data that will be passed into the Artificial Neural Network (ANN). The signature should enable the ANN's framework to make decisions more accurately and rapidly. While the project will seek to address the issue of ANN net complexity by simplifying the data patterns that will be used as inputs, the signature devised and used should lead to an efficient and intelligent.

1.4 Research Questions

The main questions in this research are identified as follows:

• Among with various rates of compression, what is the ideal Haar-based compression signature for data (images and audio signals) that can be input into an ANN?

• What is the increase in performance of an ANN when using such signatures?

• How does the performance of the ANN change when using different activation functions and different numbers of hidden layers in the ANN?

1.5 Thesis Outline

The rest of the thesis is organised as follows: In chapter 2, the concepts, definitions, histories, algorithms, and applications for artificial intelligent, machine learning, artificial life, knowledge discovery in databases and compressed data techniques are reviewed with reference to the current literature.

In chapter 3, examines various ANN typologies and architectures, with particular emphasis on back propagation and resilient ANNs using adaptive back propagation. In chapter 4, contains a brief introduction to wavelet transformations with an emphasis on wavelets applied to image and signal processing.

In the next two chapters were focus in simulation results and tests. Which in chapter 5, contains results from experiments with an ANN, using one hidden layer, applied to MINIST and TI46 data-sets. The ANN uses two types of activation functions, sigmoid and softplus. Measurements of the accuracy of the ANN as a function of the HWT compression rate are used to determine the optimal configuration of the HWT and the ANN. And in chapter 6, deals with deep neural networks using the Keras library. The addition of more hidden layers to the ANN as a means of optimising performance is considered. The results from experiments on both the MINIST and TI46 data-sets are presented here.

Finally, chapter 7 contains conclusions and discussion on the success of the research. Proposals for future work in this area are also included.

Chapter 2

Literature Review

The increasing use of digital data has resulted in the important problem of storing and transferring the massive amount of data. For example, multimedia data (graphics, audio and video) require considerable bandwidth and storage capacity. While mass storage density is progressing rapidly, the performance of any data system depends primarily on the speed of the processor.

In addition to this increasing use of digital data, the important issue of data storage and transfer of enormous volumes of data, requires considerable bandwidth and storage capacity. By using data compression techniques, some redundant information can be removed from the data and the size of a graphics file can be minimised without adversely affecting the quality of the data.

The aim of this project is to implement a HWT compression on data being fed into an Artificial Neural Network. In order to put this work in context this chapter will present a review of the research literature in such areas as:

- Artificial Intelligent (AI).
- Machine Learning (ML).
- Artificial Life (Alife).
- Knowledge discovery in Databases (KDD).
- Artificial Neural Network (ANN).
- The datasets used in this project; MINIST and T146.
- Compressed data techniques.
- Multi-resolution Analysis.
- Filter bank.
- Python and its Machine Learning libraries.

2.1 Artificial Intelligent (AI)

Nowadays Artificial Intelligence (AI) is widely used in several of the branches that normally require human intelligence; for example, pattern recognition, data searching, learning from experience, planning, and much more. AI [9], itself, is a young

field. However, as shown in figure 2.1, it has emerged from ideas, views, and techniques from many other fields:

• From ancient times, the emergence of philosophy and its associated theories of reasoning and learning, together with the view that the mind is a physical system.

• From millennia of mathematics, the formal theories of algorithms, logic, probability, decision making, and computation were formed.

• From psychology, the tools with which to study the human mind, and process cognitive information.

• From linguistics, the theories of the structure and meaning of language.

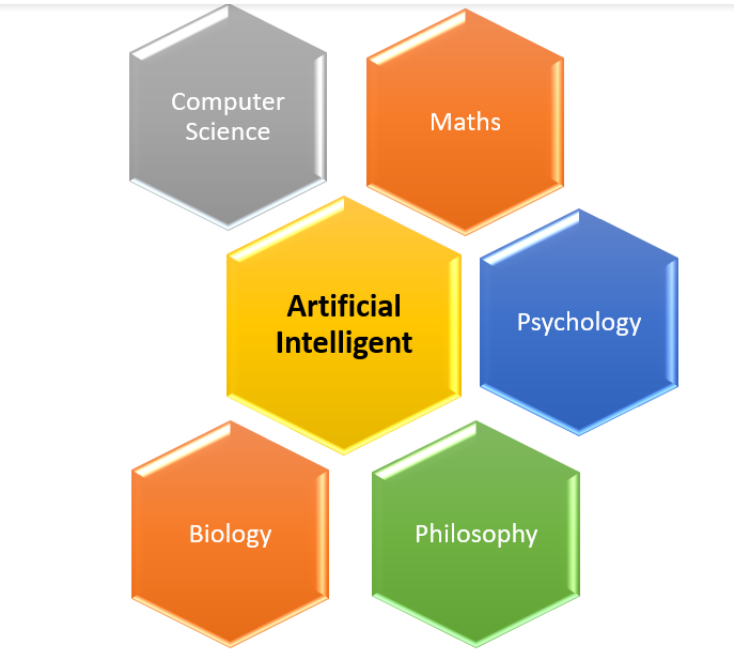


FIGURE 2.1: Areas which contribute in Artificial intelligence.

2.1.1 Philosophy in artificial intelligence history

The philosophy of artificial intelligence is a collection of issues, primarily concerned with the possibility of building an intelligent thinking machine. The philosopher Hubert Dreyfus [10] said that "The story of artificial intelligence might well begin around 450 B.C.". This started back with the big three in Greek philosophy (Plato, Socrates, and Aristotle). Much of Western philosophy finds its basis in thoughts on Ethics, Political Philosophy, Epistemology, Metaphysics, and Moral Psychology and their coalescence into an interconnected and systematic philosophy. From Plato's Socratic dialogues [11], we found that Socrates developed a casual system of proper reasoning, which in principle allowed one to mechanically generate conclusions, given initial premises. Aristotle did not believe that creative products come through

unique processes. He believed they come from logical steps of nature law, and he also had a notion of intuitive reason [12].

In the early seventeenth century [13], Rene Descartes, the "Father of Modern Philosophy" and eminent mathematician spoke of 'thinking machines' in his Discourse on Method. His opinion, on what today would be deemed AI theory, was to discount the very possibility of thinking machines going to far as to say that the very term 'thinking machine' is a misnomer. In another word, he held that there is a part of the mind (or soul or spirit) that is outside of nature and exempt from physical laws.

Several generations of philosophers failed to prove whether machines can think or not. In 1950, Alan Turing [14], the eminent English mathematician, logician, philosopher, biologist, founding father of theoretical computer science and artificial intelligence, published his paper "Computing Machinery and Intelligence"[15] which introduced his theoretical Turing Test. He replaced the question "Can machines think?" by another, "Are there imaginable digital computers which would do well in the imitation game?".

2.1.2 Mathematics in artificial intelligence history

Even though philosophy and philosophers were key in conceptually formulating the idea of AI, mathematics was needed to turn it into reality. The mathematical formalisation adopted four main areas: computation, logic, algorithms and probability.

An algorithm [16] is a procedure or formula for solving a problem, based on conducting a sequence of specified actions. It is extensively used in information technology and software development. Algorithms have a long history and the word was introduced in the 9^{th} century when the name of the Persian scientist, astronomer, and mathematician Abdullah Muhammad bin Musa al-Khwarizmi[17], often referred to as "The father of Algebra", was Latinised to create the term "Algorithm". Mathematical Logic owes its origins to George Boole (1815-1864) an English mathematician, professor and logician, In 1854, he published his work [18] that introduced algebraic logic (Boolean Algebra) which is fundamental to the operation of all computer software and hardware in use today. The theory of probability was formulated by the Italian Gerolamo Cardano (1501-1576) [19], a mathematician, physician, biologist, physicist, chemist, astrologer, astronomer, philosopher, writer, and gambler. He first framed the idea of probability, describing it in terms of the possible outcomes of gambling events.

2.1.3 Psychology in Artificial Intelligence history

Psychology is a foundation of artificial intelligence; in fact it can be considered a primary source for artificial intelligence[20]. In 1879, Wilhelm Wundt (1832-1920) [21], a German physician, physiologist, philosopher, and professor, and the first person ever to be called a psychologist, opened the first laboratory of experimental psychology at the University of Leipzig. Wundt's experimental went a long way to make psychology a science.

At the beginning of the 20th century psychology was dominated by behavioural actions. John Watson (1878-1958) and Edward Lee Thorndike (1874-1949) argued against this subjectivism, rejecting any theory involving mental processes on the grounds that introspection could not provide reliable evidence. Behaviourism was concerned primarily with the learning of associations, particularly in nonhuman species (animals), and it constrained theorizing to stimulus–response notions.

In the late 1950s [22], what is commonly referred to as the 'cognitive revolution' occurred. This "Birth of Cognitive Psychology" is often attributed to George Miller [23] and his paper "The Magical Number 7 Plus or Minus 2". Cognitive psychologists consider it essential to look at the mental processes of an organism and how these influence behaviours. In the comparison between the cognitive and the behavioral approach, the latter only studies the behavior that can be observed externally (stimulus and response), which can be measured objectively. This inner behavior can not be studied because it is impossible to know and measure objectively what happens in the mind of a person. Conversely, the cognitive approach advocates that experiments can scientifically study internal mental behavior. Cognitive psychology assumes that between stimulus / input and response / output a mental process takes place. Such mental processes can be memory, perception, attention, resolution of problems, etc. They are called mental processes because they mediate between stimulation and reaction. They follow the stimulus and the response.

2.1.4 The history of Artificial Intelligence

AI started to be recognized by Warren McCulloch and Walter Pitts [24] in (1943). They drew on three sources: knowledge of the basic physiology and function of neurons in the brain, the formal analysis of propositional logic due to Russell and Whitehead, and Turing's theory of computation. They showed that any computable function could be computed by some network of connected neurons, and that all the logical connectives could be implemented by simple net structures. They also suggested that suitably defined networks could learn.

Claude Shannon [25] (1950) and Alan Turing (1953) were writing chess programs for von Neumann-style conventional computers. The first neural network computer was built in 1951.

The official birthplace of AI is considered to a workshop held at Dartmouth college in Hanover, New Hampshire, in the summer of 1956 [26]. John McCarthy [27] induced Minsky [28]. Claude Shannon, and Nathaniel Rochester brought together U.S. researchers interested in automata theory, neural nets, and the study of intelligence to the workshop. Among the attendees were Arthur Samuel (who coined the term "machine learning" in 1959) [29] from IBM, Ray Solomonoff (founder of the theory of Universal Inductive Inference) [30], and Oliver Selfridge (the "Father of Machine Perception.") [31] from MIT. The workshop introduced all the major figures in this emerging field of study to each other. The lasting legacy of the workshop was an agreement to adopt McCarthy's new name for the field: Artificial Intelligence.

Newell [32] and Herbert A. Simon 's [33] had early successes in the application of General Problem Solver (GPS) [34]. This program was designed from the start to imitate human problem-solving protocols. Thus, GPS was probably the first program to embody the "thinking humanly" approach. This combination of AI and cognitive science is still an active area of research at Carnegie Mellon University (CMU).

In 1959 McCarthy defined the high-level language LISP, which was to become the dominant AI programming language. LISP is the second-oldest language in current use With LISP, McCarthy had the tool he needed, but access to scarce and expensive computing resources continued to be a serious problem.

In 1960s, Minsky supervised a group of students who chose limited problems that apparently required intelligence to solve. These limited domains became known as microworlds. James Slagle published a paper "A heuristic program that solves symbolic integration problems in freshman calculus: Symbolic Automatic Integrator (SAINT)" [35]. Tom Evans's ANALOGY [36] program solved geometric analogy problems that appear in IQ tests. Bertram Raphael's (1968) (authored The Thinking Computer: Mind Inside Matter (1976)) [37] SIR (Semantic Information Retrieval) was able to accept input statements in a very restricted subset of English and answer questions thereon. Daniel Bobrow's [38] student program (1967) solved algebra story problems such as "If the number of customers Tom gets is twice the square of 20 percent of the number of advertisements he runs, and the number of advertisements he runs is 45, what is the number of customers Tom gets?"

The history of AI has had cycles of success, and failure. Moreover, there have also been cycles of introducing new creative approaches and systematically refining the best ones. However, in 1980 the first successful commercial expert system began operating. The program helped to configure orders for new computer systems, and

by 1986 it was saving its parent company an estimated 40 million Dollar a year.

2.1.5 Definitions

As Figure 2.2 indicates, the definitions of artificial intelligence, are classified into two categories:

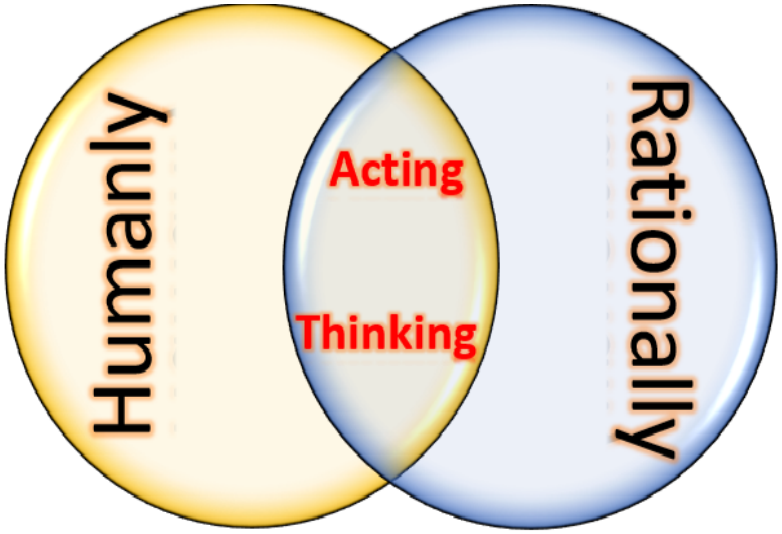


FIGURE 2.2: AI Definitions Categories

AI systems resulting from these two categories can be further subdivided into four groups: systems that think like humans, systems that act like humans, systems that think rationally, and systems that act rationally. These are considered in turn.

2.1.5.1 Systems that think like humans.

"The exciting new effort to make computers think . . . machines with minds, in the full and literal sense" (Haugeland [39], 1985)

"The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..."(Bellman [40], 1978)

If a system is to be designed to mimic human thinking, it is necessary, initially, to ascertain how human's think by studying the human mind itself. Therefore, experimental techniques from psychological science have been devised to precisely test theories of the actions of the human mind. If the program's input/output and timing behaviour matches human behaviour, then this is considered proof that many of the program's mechanisms may additionally be operational in humans. This appeared clearly in GPS, the "General Problem Solver" (Newell and Simon, 1961), as

they were more concerned with comparing the program's reasoning steps with human subject's resolution of specific issues. This approach differs from other contemporary researchers (e.g. Wang (I960)), who were involved with obtaining the correct answers regardless of how human subjects tackled it.

2.1.5.2 Systems that act like humans.

"The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil [41], 1990) "The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight [42], 1991) The Turing Test, developed by Alan Turing (1950), described intelligent behavior as the ability to realize human-level performance by performing instructed psychological tasks well enough to fool an interrogator. Roughly speaking, the test proposed for the computer to be interrogated by a human via a teletype, and to have deemed ti have passed the test if the interrogator cannot tell if there's a laptop or human at the opposite end. Currently, the world of computer programming uses this test in:

- natural language processing.
- knowledge representation.
- automated reasoning.
- machine learning.

The issue of mimicking a human arises once AI programs got to interact with individuals. These programs should behave according to the normal conventions of human interaction to make themselves understood. The underlying illustration and reasoning in such a system may or may not be compatible with the human's model.

2.1.5.3 Systems that think rationally.

"The study of mental faculties through the use of computational models" (Charniak and McDermott [43], 1985).

"The study of the computations that make it possible to perceive, reason, and act" (Winston [44], 1992).

By 1965, the logics tradition within artificial intelligence had appeared when the first program (Logic Theorist program [45]) started using logical notation to describe and solve problems that were proving to be otherwise unfeasible. The Logic Theorist written by Allen Newell, Herbert A. Simon and Cliff Shaw introduced the search tree [46]; a tree-like data structure which could be used to explore reasoning, heuristics and list processing.

2.1.5.4 Systems that act rationally.

"A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff [47], 1990).

"The branch of computer science that is concerned with the automation of intelligent behaviour" (Luger and Stubblefield [48], 1993).

The study of AI as a rational system was influenced by two approaches.

• The "laws of thought" approach: correct inference is only a useful mechanism for achieving rationality, and not a necessary one.

• AI is more amenable to scientific development than approaches based on human behavior or human thought, because the standard of rationality is clearly defined and completely general. Human behavior, on the other hand, is well-adapted for one specific environment and is the product, in part, of a complicated and largely unknown evolutionary process that still may be far from achieving perfection.

As figure 2.1 illustrates, AI's can behave rationally far more effectively than humans (the area of rational thought excluding human thought is far greater than the intersection) and so AI's that are properly constructed have a high potential for rational behaviour. If the AI is required to behave both rationally and humanely then it requires finer tuning to be in the intersection.

2.2 Applications of AI

Figure 2.3 below shows the current extent of the application of AI to problems in areas ranging from machine learning to computer vision.

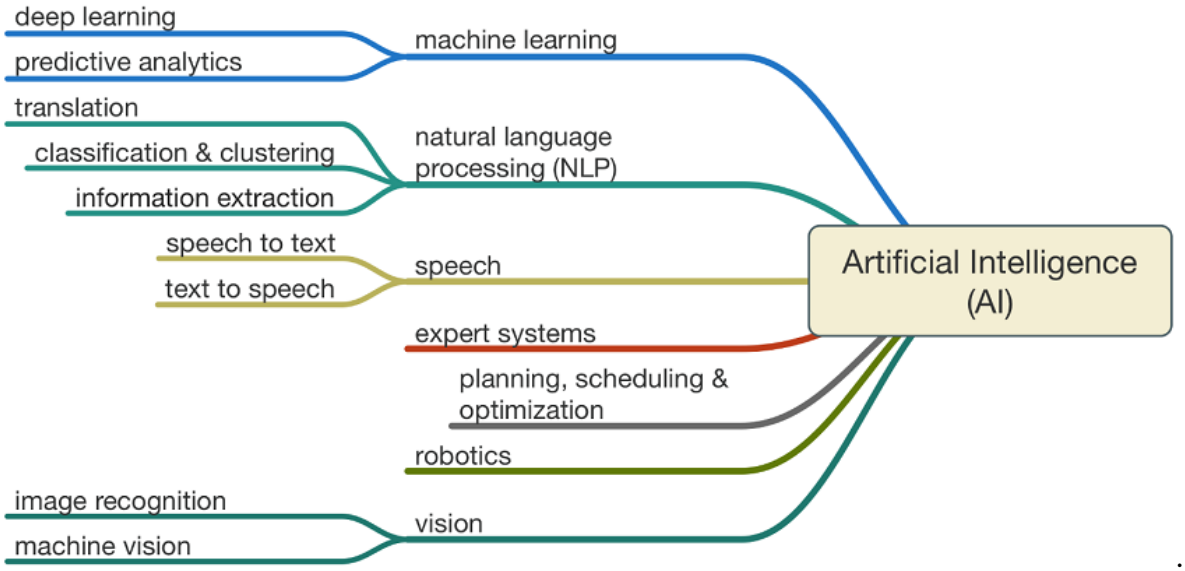


FIGURE 2.3: The different branches of AI [49]

2.3 Artificial life (AL or A-life)

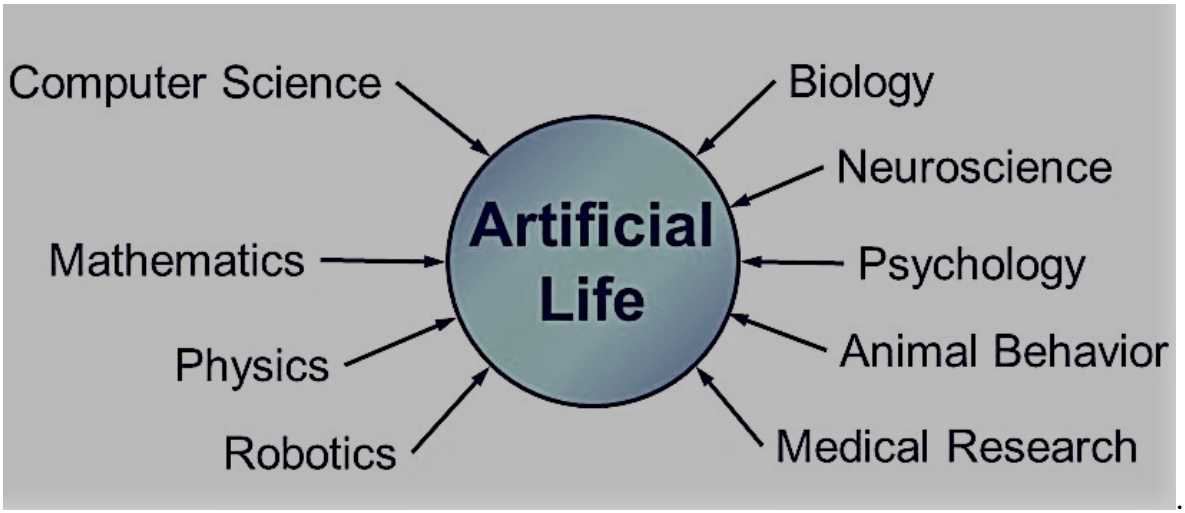


FIGURE 2.4: The foundations of AL [50].

Artificial life [51] (ALife) is a sub field in artificial intelligence in which living behaviours are simulated and analysed within computers. Thus, a simulation of living systems can be built to emulate, to some extent, real organisms. Virtually, ALife can be roughly used to study of living behaviours and systems. The foundations of AL are shown in Figure 2.4.

In recent years there has been growing interest in the concept of ALife that has brought together people interested in computer models of adaptive and self-organising systems. This interest is not restricted to just biology but to economics, social science, physical, chemistry and many more.

The creation of ALife would necessarily constitute an artificial living entity, henceforth an 'artificial organism'. Processes such as self-reproduction, growth, development, and learning require modelling in some form. Software based ALife systems, by necessity, work by creating models of real biological systems and by then evaluating the abilities of such systems through observation with comparable real-world systems.

ALife systems can be realised using groups of small robots called animats to study the design principles of autonomous system or agents that can act their environment through sensors and effectors [52] [53].

John von Neumann developed the idea of cellular automatons in the 1940s to create a theoretical model for a self-reproduction machine. He described a model of a self-reproduction machine in a paper written for the Hixon Symposium called "The General and Logical Theory of Automata. " [53]. Von Neumann described a cell car with twenty-nine possible cell states in which each cell is connected to

the cell above, below, to the left and to the right (neighborhood). Von Neumann describes this system in his book Von Neumann describes this system in his book Theory of Automatic Von Neumann describes this system in his book "Theory of Self-Reproducing Automata " [54] , which was completed and published by Arthur Walter Burks in 1966 after his death.

2.3.1 Wain

Wain [55] is an artificial agent in an advanced computational ecosystem. This system was created in Athlone Institute of technology to evolve the artificial agent into a "smarter" one both during its lifetime and it progeny in later generations by studying the brain structure, appearance, and metabolism. It also describes how they interact with their environment, how they make decisions and learn from their mistakes, and how they eat, mate, rear children, and play.

The design of the classifier component of the brain allows it to effortlessly reveal the patterns identified. When poisonous data was introduced to their habitat, wains adapted, primarily by modifying their learning rates through evolution. Evolution also made the brains of wains more efficient, by reducing the number of patterns that their brains, through the use of Kohonen Maps (SOM's), stored without affecting the Wains ability to identify nutritious data. It was found that Wain populations exhibited complex, interesting behaviour after as little as 12 generations. A more detailed at SOM's is taken in section 2.4.3.

2.4 Machine learning

ML [56] can play an important role in a wide range of critical application such as data mining, natural language processing, image recognition and more. This field is expanding rapidly.

Machine Learning is defined as an application of Artificial intelligence that provides systems with the ability to learn, improve, and adapt by accessing data and using it to learn autonomously. The primary aim is to enable the system to learn without human intervention by learning from instruction or direct experience in order to make a better decision. In Figure 2.5 the relation between all AI, ML, NN and DL are shown .

2.4.1 Machine learning vs Statistics.

ML and statistical modelling [57] have their origins in different disciplines. Statistical modelling is a sub-field of mathematics which deals with finding relationships

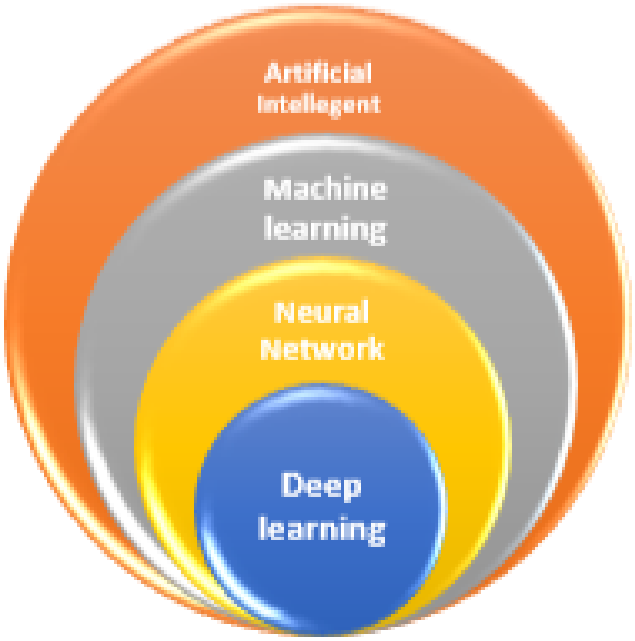


FIGURE 2.5: The relation between all AI, ML, NN and DL.

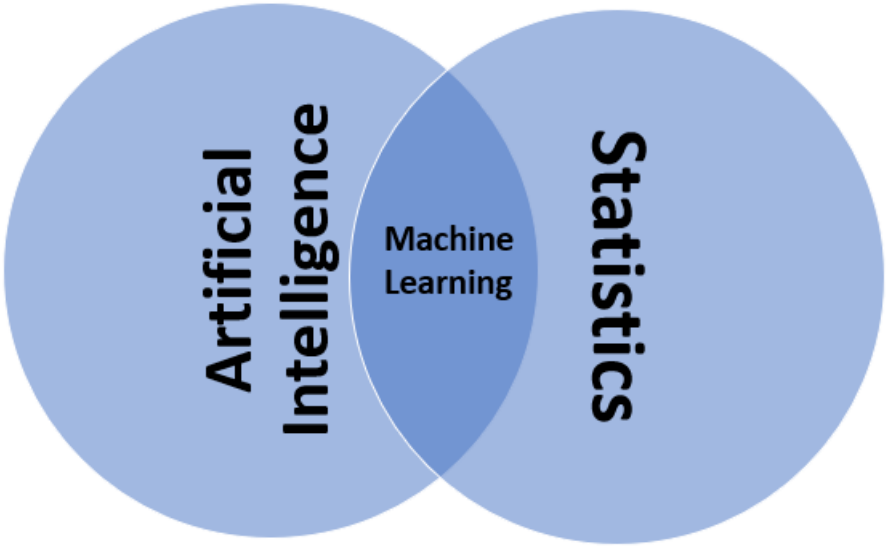


FIGURE 2.6: The relation between Artificial Intelligence, Machine learning, and Statistics.

between variables to predict an outcome and has been around since the work by Cardano in the16th century. On the other hand, ML was only formally defined more recently by computer scientists such as Arthur Samuel and Tom Mitchell at the end of 1950's. ML flourished in the 1990's, emerging from being a purely computer science based discipline and evolving to the study of pattern recognition and computational learning theory in AI.

TABLE 2.1: The different between machine learning and statistics [58].

	MACHINE LEARNERS	STATISTICIANS
Network/Graphs vs. Models	Network/Graphs to train and test data	Models to create predictive power
Weights vs. Parameters	Weights used to maximize accuracy scoring and hand tuning	Parameters used to interpret real-world phenomena - stress on magnitude
Confidence Interval	There is no notion of uncertainty	Capturing the variability and uncertainty of parameters
Assumptions	No prior assumption (we learn from the data)	Explicit a-priori assumptions
Distribution	Unknown a priori	A-priori well-defined distribution
Fit	Best fit to learning models (generalization)	Fit to the distribution

There are differences between statistics and ML; statisticians need to understand the underlying population distribution under study and develop parameters that will provide predictive power. The goal for a statistician is to predict a certain degree of certainty between variables. In other hand, ML advocates want to create algorithms that most accurately predict, classify and cluster.

In spite of the above differences, ML and statistics are very similar. In fact, they are so similar that they talk about almost all the same subjects, methods and concepts, but for many of them they have different names.

Professor Rob Tibshirani [59] has created a glossary that compares the main terms in machine learning vs statistics. Elements of this glossary are shown in Table 2.2. Professor Tibshirani is one of the authors of the book [60] "An Introduction to Statistical Learning".

2.4.2 The common algorithms used in machine learning Signals

In Figure 2.7 presents the various classifications of algorithms used in ML systems as a mind map. The three major sub-categories will be considered in turn.

TABLE 2.2: The similarity between the machine learning and statistics.

Machine learning	Statistics
network, graphs model	weights parameters
learning fitting	generalization test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant= $50,000

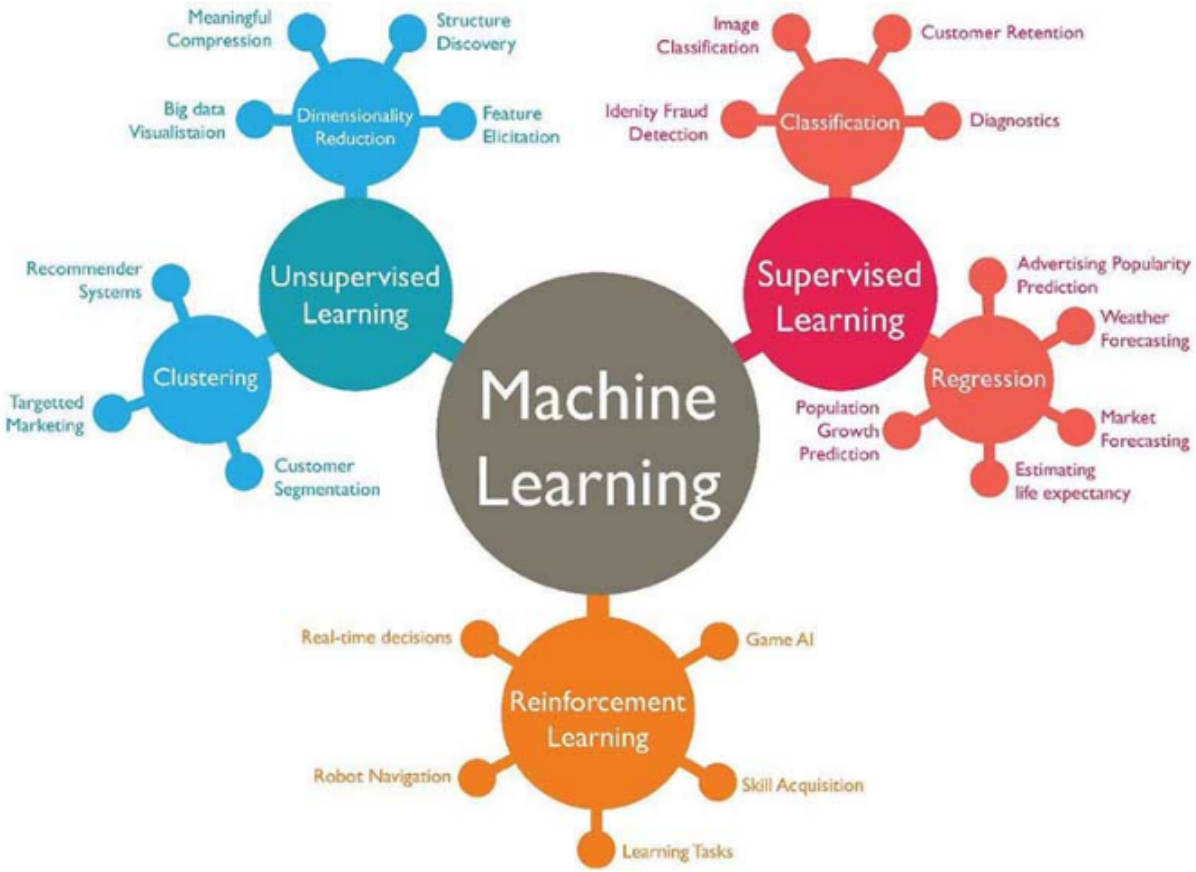


FIGURE 2.7: Typical for the 3 types of machine learning application [61].

Supervised learning

Where the system is required to learn a function which maps a vector into one of several classes by looking at several input-output examples of the function [62]. In Figure 2.8 presents the different between main types of supervised learning algorithms include:

• Classification algorithms: These algorithms build predictive models by means of the training data (the input) which have features and class labels. These predictive models in-turn use the features learnt from training data on new, previously unseen data to predict their class labels. The output classes are separate. Types of classification algorithms include decision trees, random forests, support vector machines, and much more.

• Regression algorithms: These algorithms are used to predict output values based on some input features obtained from the input. In another words, the algorithm builds a model based on features and output values of the training data and this model is used to predict values for new data. The output values in this case are continuous and not separate. Types of regression algorithms include linear regression, multivariate regression, regression trees, lasso regression, and more.

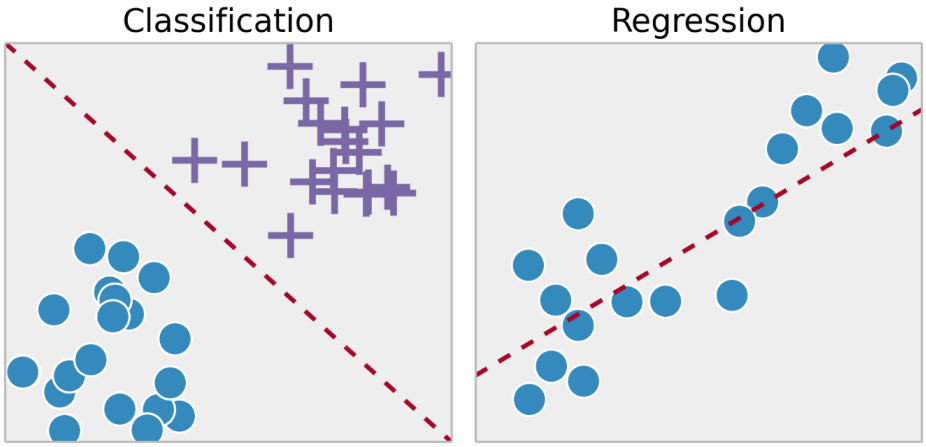


FIGURE 2.8: An illustration of the different between classification and regression techniques[63].

Unsupervised learning

Where the system used to draw inferences from datasets consisting of input without identifying the result before the termination of the operation [64]. The main types of unsupervised learning algorithms include:

• Association rule learning algorithms: These algorithms are used to mine and extract rules and patterns from data sets. These rules explain relationships between different variables and attributes, and depict frequent item sets and patterns which occur in the data. These rules in turn can aid in the discovery of previously unknown properties of the data sets. Popular algorithms include Apriori and FP Growth.

• Clustering algorithms: The main goal in these algorithms is to cluster or group input data points into different classes or categories using just the features derived from the input data alone and no other external information, as shown in Figure 2.9. Unlike classification, the output labels are unknown in advance of clustering. There are different approaches to build clustering models, for example by using means, medoids, hierarchies, and many more. Some popular clustering algorithms include k-means, k-medoids, hierarchical clustering and the self-organizing or Kohonen map.

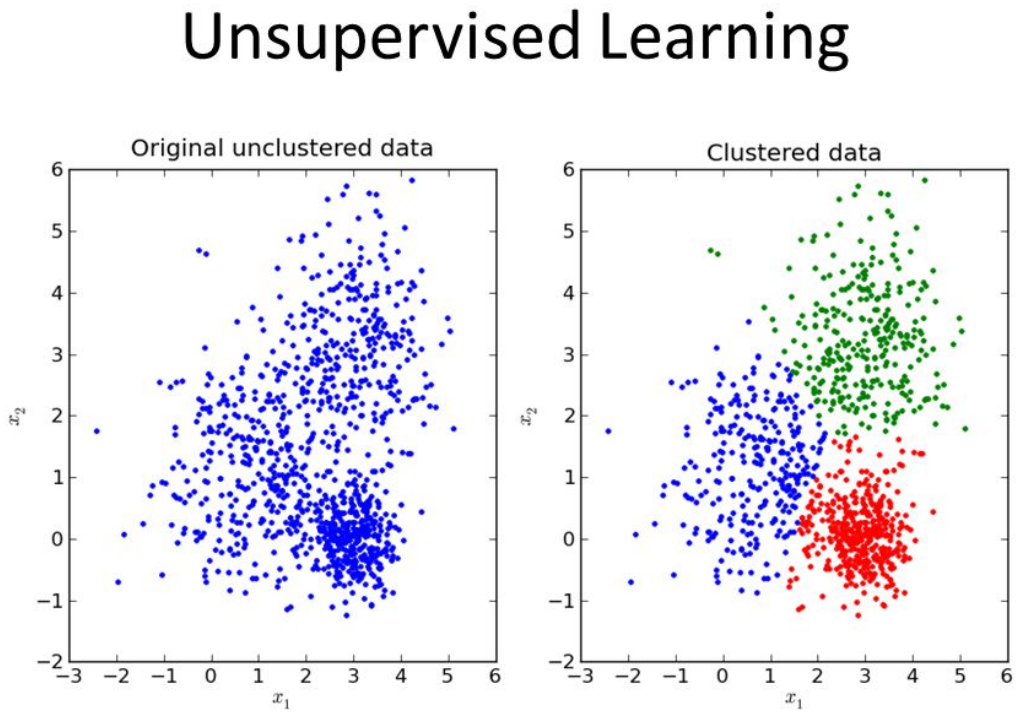


FIGURE 2.9: An example of how a clustering algorithm can be used to separate data into classes; 3 in this example [65].

mm

Reinforcement learning

It is a method that aims at using observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk. The reinforcement learning algorithm (called the agent) continuously learns from the environment in an iterative fashion. In the process, the agent learns from

its experiences of the environment until it explores the full range of possible states [66].

Semi-Supervised learning

This is where the system combines supervised learning methods with unsupervised learning methods. This has the goal of improving generalization on supervised tasks using unlabelled data [67].

2.4.3 Self-organizing Map (SOM) and self-Generating Model (SGM)

The Kohonen or Self-Organizing Map (SOM) algorithm uses a pre-determine matrix of nodes each of which, initially, contains a random model of the type under investigation [68]. The input data model is compared with these nodes with the closest match being declared the winning node or Best Matched Unit (BMU). The BMU's model is changed to align itself more closely with the input data pattern with the amount of change being determined by a number of parameters including a learning rate. All other nodes within a pre-determined radius of the BMU are also subject to changes governed by a 2D Gaussian envelope. The SOM will eventually converge to a state where the matrix of nodes contains models that preserve the topology of the input set. This method is ideal for reducing high dimensional data to, typically, 2D models.

The self-Generating Model (SGM) algorithm is like the SOM. In fact, it was derived from it. It does not serve to preserve the topology of the input set and so a major computing overhead of SOM's is eliminated. When the BMU is chosen the difference between it and the input pattern is measured and, if it is found to be greater than a difference threshold, a new model node is created. As a result, SGM's have proven themselves to be more accurate and faster than the SOM's on the same data sets.

2.5 Knowledge Discovery from databases (KDD)

The Knowledge discovery in databases (KDD) [69] pipeline is the process of discovering useful knowledge from a collection of data. Data, in its raw form, is just a collection of objects (numbers, words, entities, etc.), where little information might be obtained without some organisation. The development of information discovery methods enabled useful information to be found in large data sets efficiently and robustly. KDD processing includes data preparation and selection, data cleansing, incorporating prior knowledge on data sets, and interpreting accurate solutions

from the observed results. Major KDD application areas include marketing, economics, and telecommunications. The goal here is to extract high-level knowledge from low-level data.

2.5.1 Knowledge Discovery in Databases stages

As shown in Figure 2.10, Data Mining [70] is the pattern extraction phase within the KDD pipeline and makes use of differing techniques of visualization, reduction of dimensionality, and classification to construction of data models to efficiently find data of interest. Data mining takes much of its inspiration and methodologies from disciplines as diverse as statistical analysis, AI, management science, and information systems in order for its standard activities of pattern recognition, mathematical modelling, and database searching to be effectively implemented.

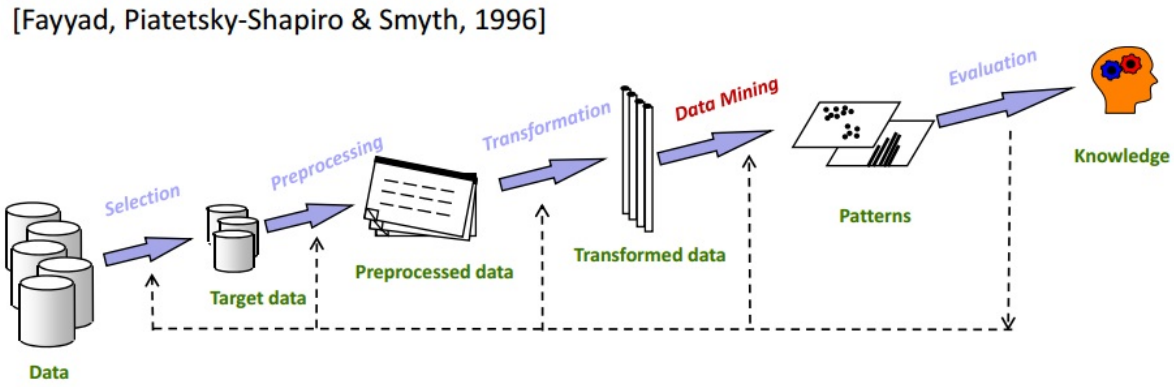


FIGURE 2.10: Knowledge Discovery stages [71].

2.5.1.1 Data selection

Selection and integration are fundamental to KDD processing. This reflects the nature of data gathering; that target data may come from many different and heterogeneous sources [72]. Lines along which data can be integrated and selected include relational databases, document collections, e-mails, photographs, video, and audio.

2.5.1.2 Data Pre-processing

The data pre-processing stage [73] is crucial since it deals with issues such as incomplete data (missing value), noisy data, and inconsistencies in the data selected which will affect the quality of the results obtained in the KDD processing. The pre-processing step attempts to reduce the complexity of the data and offers better

conditions to subsequent analysis. Through this the nature of the data is better understood and the data analysis is performed more accurately and efficiently. Data pre-processing is challenging as it involves extensive manual effort and time in developing the data operation scripts.

2.5.1.3 Transformation

This is the stage where data [74] are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations. The number of effective variables is reduced and only useful features are selected to represent the data more efficiently and effectively. In short, data is transformed into appropriate forms making it ready for the data mining step.

2.5.1.4 Data Mining

Data mining [75] is the process by which search and querying performance on massive data sets is optimised against the imminent requirement for turning such data into useful information in addition to knowledge. Data mining can take on several types, the option chosen will depend on the desired outcomes.

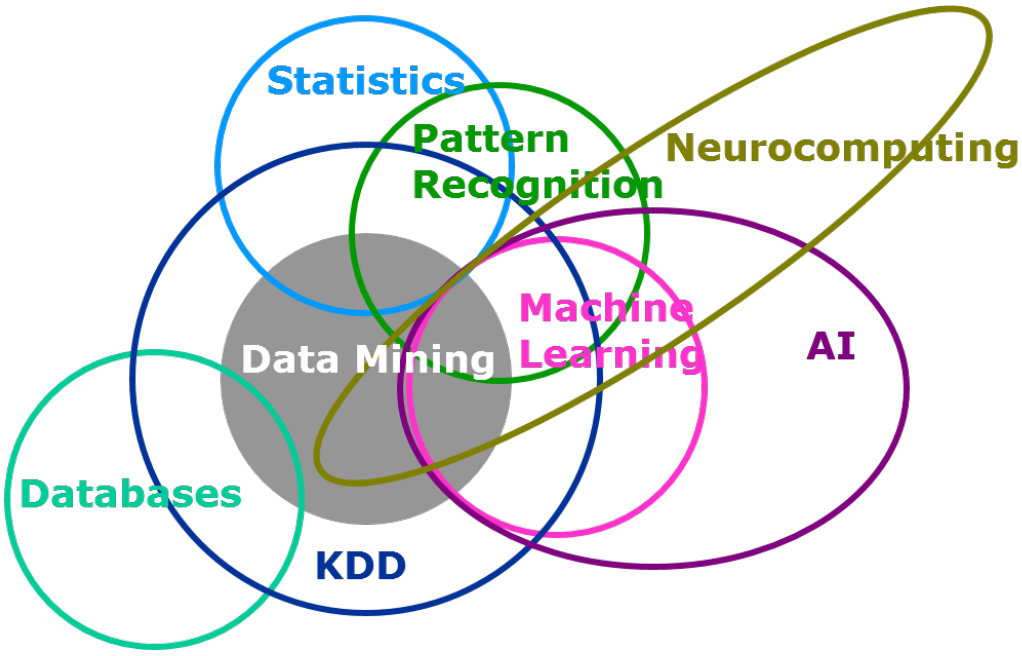


FIGURE 2.11: Data Mining Is Multidisciplinary [70]

As stated above, data mining [70] is the pattern extraction phase within the KDD pipeline and makes use of differing techniques of visualization, reduction of dimensionality, and classification to construction of data models to efficiently find data of interest. Accordance with Figure 2.11, Data mining takes much of its inspiration

and methodologies from disciplines as diverse as statistical analysis, AI, management science, and information systems in order for its standard activities of pattern recognition, mathematical modelling, and database searching to be effectively implemented.

2.5.1.5 Evaluation

In this step [76] results to be interpreted and evaluated to discover knowledge from the patterns are analysed. This step evaluates the properties and usefulness of extracted patterns. In this step we also visualize the extracted patterns and models and also visualize the data given to the extracted models.

2.5.2 Taxonomy of Data Mining Methods

Data mining is a very interdisciplinary field with many different classes of data mining algorithms. Taxonomy is helpful for understanding the different kind of methods as shown in Figure 2.12 below:

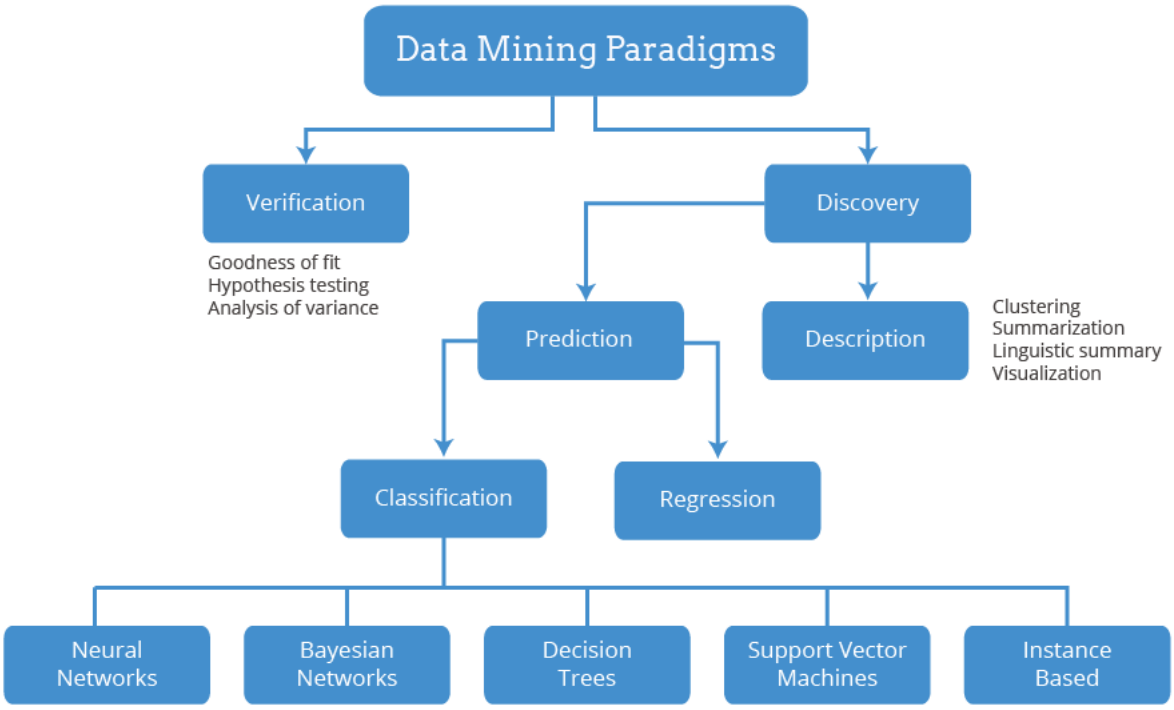


FIGURE 2.12: Data Mining Paradigms [77]

There are two type of data mining: [78] verification-oriented (the system verifies the user's hypothesis) and discovery-oriented (the system finds new rules and patterns autonomously). Verification-oriented methods deal with the evaluation of

a hypothesis proposed by external sources (like an expert). These methods include the most common methods of traditional statistics, like goodness of fit test, tests of hypotheses (e.g., t-test of means), and analysis of variance (ANOVA).

On the other hand, Discovery methods [79] are those that automatically identify patterns in the data. The discovery method consists of descriptive methods that are comprised of unsupervised learning functions or prediction-oriented methods. The unsupervised learning functions do not predict a target value, but focus more on the intrinsic structure, relations, interconnectedness, etc. of the data. The most common models for descriptive methods are clustering and visualization.

2.5.2.1 Clustering analysis

Identifies clusters embedded in the data. A cluster is a collection of data objects that are similar in some sense to one another. A good clustering method produces high-quality clusters to ensure that the inter-cluster similarity is low and the intra-cluster similarity is high; in other words, members of a cluster are more like each other than they are like members of a different cluster.

2.5.2.2 Visualization analysis

Visualization analysis Analysis of visualization is the process of converting textual or numerical data to meaningful images. This process, in other words, recognises patterns like man's way. Brain can make the understanding of patterns simpler. The visualization can complement the data mining techniques referred to above. The combination of data mining and data visualization, together with the enormous storage space available in today's data warehouse, can provide business decision makers with valuable information.

2.5.2.3 Prediction-oriented methods

The other branch in discovery methods is Prediction-oriented methods That include functions for supervised learning. These functions are designed to create a behavioral model that collects new and unseen samples and can predict the values of one or more sample variables. It also develops patterns that form the discovered knowledge in a manner that is easy to understand and operate. Some Prediction-oriented methods can also help understand the data. Regression and classification are the most common models for prediction-oriented methods.

The regression functions [80] are used to determine the relationship between the dependent variable (target field) and one or more independent variables. The dependent variable is the one whose values you want to predict, whereas the independent variables are the variables that you base your prediction on. A Regression Model defines three types of regression models: linear, polynomial, and logistic regression. The model type attribute indicates the type of regression used.

The classification functions [81] try to discover relationships between the attributes that would make it possible to predict the outcome. They are given a data set not seen before, called the prediction set, which contains the same set of attributes, except for the prediction attributes – not yet known. The algorithm analyses the input and produces a prediction. The most common model in classification is support vector machine, decision tree and neural network.

The Support Vector Machine (SVM) [82] is powerful, a state-of-the-art algorithm with strong theoretical foundations.A SVM has strong regularization properties. Regularization refers to the application of the model to new data. The ability to apply this technique to most problems and the ease of training the SVM is far beyond the capacities of more traditional methods such as ANNs and radial basis functions.

The decision tree [83] algorithm is a mathematical graph theory based algorithm that seeks to construct a tree (useful in the creation of data mining models) to construct induction learning algorithm based on examples. This method makes it easy to extract display rules, has smaller computation overheads, could display important decision property, and own higher classification precision.

2.6 The Data sets

2.6.1 The MNIST database

The MNIST database [85] of handwritten digits, with each image labelled by the integer, has a training set of 60,000 examples, in addition to a test set of 10,000 examples. It was collected by the US based National Institute of Standards and Technology (NIST). The digits have been modified to have the same size-normalized and centred in a fixed-size image such that the width or height of the bounding box equals 28x28 pixels.

The MNIST database is a standard database for people researching learning techniques and pattern recognition methods as it consists of real-world data that has been pre-processed thus eliminating the need to expend effort on data formatting.

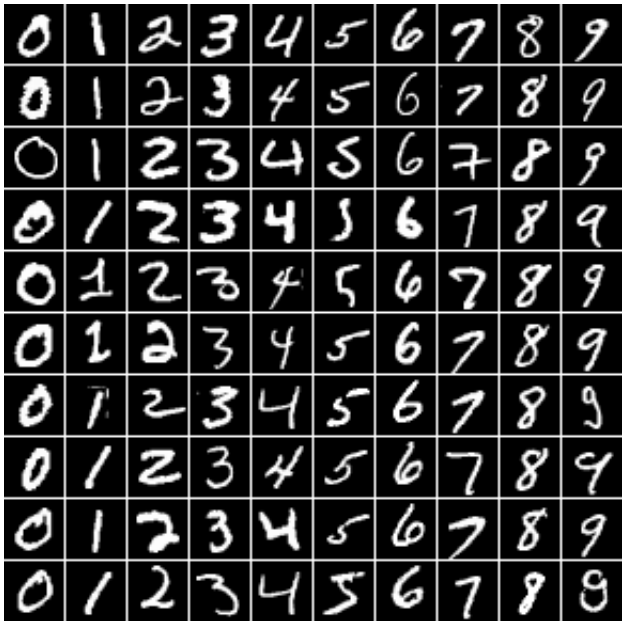


FIGURE 2.13: MNIST dataset sample. [84]

The training set is used in the project to teach the algorithm to predict the correct label, the integer, while the test is employed to check how accurate the artificial neural network can become.

2.6.2 The TI46 speech dataset

The TI46 [86]: database is a corpus of 46 isolated of spoken words, digits, with each word referred to the numbers from zero to nine, has a training set of 1594 samples, in addition to a test set of 2594 samples. the corpus is intended for the evaluation of Automated Speech Recognition (ASR) [87].

The HCopy tool provided as part of the Hidden Markov Model Toolkit (HTK) was extracted from the samples in the TI46 corpus. Each frame (feature vector) has 13 static coefficients, 13 velocity coefficients (first derivatives), and 13 acceleration coefficients (second derivatives) for a total of 39 coefficients. Pre-emphasis of the first order was applied with a coefficient of 0.97. There were 23 channels of filter banks (which show the amount of energy in each frequency region) and 22 coefficients of cepstral liftering (inverse filtering, This means that a signal is filtered on an inverse Fourier transformation). The frame rate was 10 ms with a Hamming window of 25ms (a math function). It was selected these configuration parameters to maximise the accuracy with which the end-pointed samples were identified by the Hidden Markov Model (HMM) [88].

Using the HTK Speech Recognition Toolkit, the HMM classifier is implemented. There are ten whole words of HMMs, one for each number, each with three states,

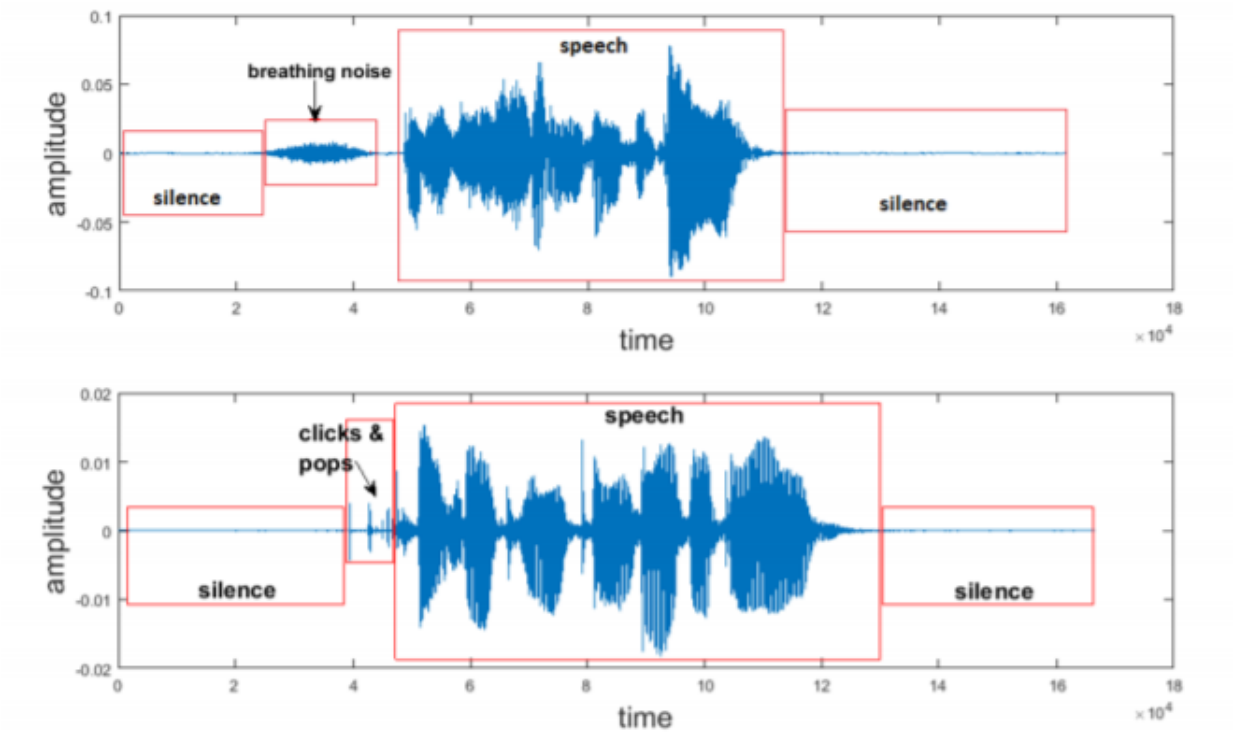


FIGURE 2.14: A speech signals containing breathing noise and mouth clicks and pops along with leading and trailing silence section. [89]

with three Gaussian mixtures in each state. Two additional models are defined for working with non-endpointed samples to represent pauses in speech, sil and sp. The sil model has three states and six mixtures in each state. The sp model has only one state.

To simplify the classification task, end-pointing is the process of removing silence from the beginning and end of an audio sample as shown in Figure 2.14. The short-term energy for each frame is the sum of the absolute values of the amplitudes of the sample in the frame. The final pointing is carried out by determining whether the short-term energy of successive frames exceeds a defined threshold (to determine the beginning of the utterance) or below a defined threshold (to determine the end of the utterance). For example, look for three consecutive frames with energy above the threshold to get the starting point; the first frame of the three is assumed to be the beginning of the utterance.

2.7 The Neural Network

A neuroscientist [90] Called Santiago Ramón Cajal conducted studies on the human nervous system in the late 19th century. Cajal found that the human nervous system actually consisted of discrete neurons that communicated with signals passing through axons, dendrites and synapses. This discovery led to further research identifying various types of neurons and the types of signals passed between them. However, it was difficult for neurobiologists to understand how neurons worked together to achieve such a high level of functionality.

It was not until the advancement of modern computing allowed researchers to develop neural systems working models. These systems gave a better understanding of how the brain's neural system operates. In 1943, Warren McCulloch and Walter Pitts [91] created an early neuron model. This model has become known as a linear binary gate. With a series of inputs, a sum would be calculated against normalized weight values in the range of either (0, 1) or (-1, 1) and would then be associated with a particular input. Given a certain threshold, the output is one of two binary classes based on whether or not the sum exceeds the threshold.

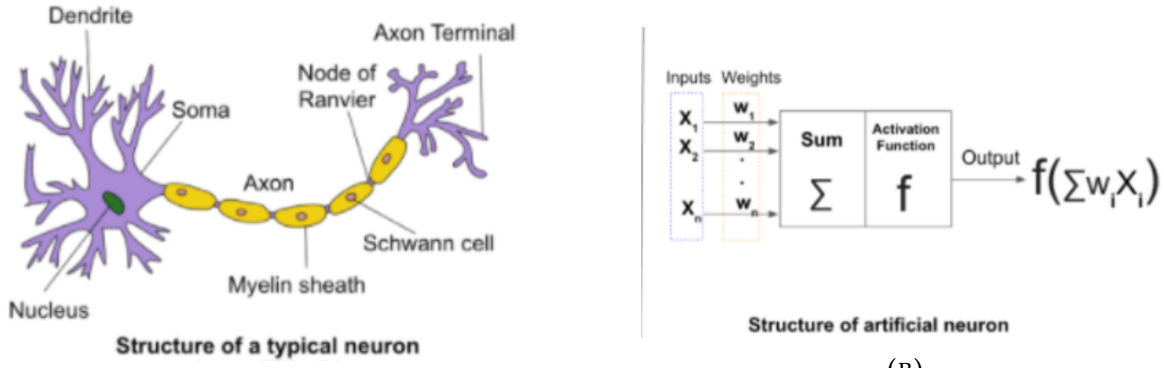
With this model of an artificial neuron, it has now been demonstrated that neuron systems assembled into a finite state automaton can calculate any arbitrary function if appropriate values for the weights between the neurons are provided. Researchers continued to implement learning procedures that were able to find appropriate weight values automatically, allowing a network to calculate any specific function. As the years have gone by, more research was carried out. Basic neural network attributes have been defined, different training methods have been created and new network architectures have been implemented.

2.7.1 A BIO Neural Network (BNN) and An Artificial Neural Network (ANN)

In Figure 2.15 (A) below, a visualization of a biological neuron is shown. The axon is responsible for output connections from the nucleus to other neurons. The dendritic tree receives input to the nucleus from other neurons. Electrochemical signals from neurons (synapses) are aggregated in the nucleus. If the aggregation surpasses a synaptic threshold, an electrochemical spike (synapse) propagates down the axon to dendrites of other neurons.

In Figure 2.15 (B) An ANN, consisting of layers made up of interconnected neurons that receive a set of inputs and a set of weights, is illustrated. The ANN performs mathematical computations on the inputs and outputs the results as a set of

"activations" that are analogous to synapses in biological neurons. While ANNs typically consist of hundreds to maybe thousands of neurons, the biological neural network of the human brain consists of billions [92].



(A) (B)

FIGURE 2.15: A typical BNN and a simple ANN [93].

A neural network consists of four components: neurons, topology (the connectivity path between neurons), weights and a learning algorithm.

Each of these components differ substantially between the biological neural networks of the human brain and the artificial neural networks expressed in software.

The table 2.3 compare the ANN and BNN in more details [92].

TABLE 2.3: Compare between ANN and BNN

	ANN	B NN
Structure	• Node present • Input present • Weight present • Output present The connections of the neurons made up the network and have three layers. The first layer (input layer) transmits signals to the neurons in the next layer (hidden layer). This layer extracts relevant features or patterns from the received signals. Those features or patterns that are considered important are then directed to the output layer	• a central cell body (soma) • dendrites • Synapse present • Axons. Information from other neurons, in the form of electrical impulses, enters the dendrites at connection points called synapses. The information flows from the dendrites to the soma where it is processed. The output signal, a train of impulses, is then sent down the axon to the synapse of other neurons.
Size	$10^2 – 10^4$ neurons	10^{11} neurons
Processing	• Processing abilities captured highly parallel computation based on distributed representations. • The model emulates a biological neural network. • 10^8 Transistors	• Processing abilities follow highly parallel processes operating on representations that are distributed over many neurons. • The human brain is a massively parallel processing system. • 10^{14} Synapses
Processing speed	• Fast, because this can achieve switching times of a few nanoseconds. • Silicon gate times are on the order of one nanosecond, that is, a million times faster. (109 Hz).	• Slow, because neurons need several milliseconds to react to stimulus. • The elementary 'cycle time' of neurons is of the order of one millisecond. (100 Hz).
Connection	• Less edges but often fully connected topology. • Connectivity is precisely specified.	• Much higher number of connections between neurons. • Neural tissue involves random connectivity with no master blueprint.
Strength	Synapse strength determined by weight only.	Other factors like neuro- transmitter influencing synapse strength.
Information storage	It is stored at the weights matrix.	It is stored at the synapses.
Information transmission	the transmit information using electrical signals	Neurons transmit information using electrical signals

2.7.2 The most common neural network models

The four most common Neural Network topologies are:

• *Perceptron* [94]: is a simple neuron node which has two inputs and one output without a hidden layer.

• *Radial Basis Function Network*[95]: offers a viable alternative to the two-layer neural network in many applications of signal processing. A common learning algorithm for radial basis function networks is based on first choosing, randomly, some data points as radial basis function centres and then using singular value decomposition to solve for the weights of the network.

• *Multilayer Perceptron networks* [96]:trained with backpropagation. This is an appropriate ANN for the task of parameter estimation, as the input can be an integral number of values over a wide range and the output is also several values over a range. In Chapter 3, we will explore this again in more details.

• *Recurrent Neural Network* [97]: it is called recurrent because it performs the same task for every element of a sequence, with the output being dependent on the previous computations.

2.8 Data Compression

Database have become ubiquitous in today's data drives society. Data compression as a means of efficient storage and retrieval, have become increasingly important in current database deployments [98].

Data compression methods [99] are used to convert an input stream (original raw data) into a smaller output stream (compressed stream) . A stream may be a file or a memory buffer. Data compression can also be defined as a scientific discipline for compact information representation. These compact representations can be created by identifying and using structures that exist in the data. Data can be in the form of text file characters, numbers that are samples of speech or image waveforms, or numerical sequences generated by other processes. Data compression techniques fall into two categories:

- Lossless Data Compression
- Lossy Data Compression

2.8.1 Lossless Data Compression

Lossless compression [100] means that when the data is decompressed no data is lost. By Decompressing the compressed data an exact replica of the original data is obtained. The compressed file is generally used for storage and/or transmission. It is necessary to decompress the file for general purposes.

Lossless Compression is typically used in [101] text files, database tables and medical imaging. Some of the main techniques employed are Run Length Encoding, Arthimatic Encoding, Shannon Fano, Lempel-Ziv-Welch, Huffman coding.

Compression without loss for audio signals is a requirement for high-quality audio reproduction [101]. All audio formats ALAC (Apple Lossless Audio Codec), RealPlayer and Monkey prefer lossless audio compression. Entropy coding is one of the most widely used lossless compression techniques.

While the advantage in the Lossless Compression is that it maintains quality, the main disadvantage is that the file size does not reduce as much as the loss compression [102].

2.8.1.1 Entropy Encoding

Entropy encoding is a lossless data compression scheme that doesn't depend on the medium's particular characteristics [103].

One of the principle foarms of entropy coding creates and assigns a special and unique prefix-free code to each special input symbol. These entropy encoders then

compress data using the corresponding variable-length prefix-free output codeword to replace each fixed-length input symbol. The length of each codeword is roughly proportional to the probability's negative logarithm. The most common symbols therefore use the shortest codes. Huffman coding and arithmetic coding are two of the most common entropy encoding techniques.

2.8.1.1.1 The Huffman coding

To encode a source symbol (such as a file character) Huffman coding [104] uses variable length code based on the approximate probability of occurrence for each possible source symbol value. In this compression technique a table is created bared on the frequency of the occurrence of a symbol. Then a tree is generated from this table with high-frequency symbols (with fewer bits) and less frequent symbols (with many bits) are assigned to this tree. This generates the table of codes. Zip, ARG, JPEG and MPEG are the most common applications using this code.

The disadvantage of the Huffman code [104] is that, due to different coding lengths, decoding is difficult. the computational overhead for Huffman can be considerable.

2.8.1.1.2 Arithmetic coding

Arithmetic coding [105] differs from other forms of entropy encoding, such as Huffman coding, in that rather than separating the input into component symbols and replacing each with a code, arithmetic coding encodes the entire message into a single number, a fraction n where [0.0 n < 1.0).

2.8.1.2 Run Length Encoding (RLE)

RLE [106] is a simple algorithm for data compression that supports bitmap file formats such as BMP. RLE essentially compresses data by minimizing the physical size of repeated character strings. This repeating string is called a run that is typically encoded in two bytes, the total number of characters is the run and the run count is called and replaces runs of two or more characters of the same character with a number that represents the length of the run followed by the real character and the single character is encoded as runs of 1. When data redundancy is high, RLE is useful and can also be used in combination with other compression techniques. The most common applications use this code are TIFF, BMP, PDF.

RLE 's [106] drawback can not achieve high compression ratios compared to other advances in compression methods.

2.8.2 Lossy Data Compression

Lossy compression algorithm is [107] ignore less important data in the compression process with the result that an exact replica of the original file can not be obtained from the compressed file. Lossy Compression is based on the assumption [107] that more information is saved in recent data files than can be seen by humans. Digital cameras can use lossy image compression to increase storage capacity with minimal degradation of the image quality. Similarly, psychoacoustic digital compression is used to remove audible signal components. Compression of human speech is often carried out using even more efficient techniques, so that "video compression "is sometimes distinguished as a separate discipline from "audio compression".

Typically, for images JPEG uses lossy compression, while in audio MP3,and MP4 make use of such techniques.

The main advantage of lossy compression [108] is the compressed file can be much smaller than its original size. The main disadvantage is the irreparable loss of information. It can also not be used in all file types because it works by deleting data. It is not advisable to use this technique to compress text because it has no redundant information.

2.8.2.1 Discrete Cosine Transform (DCT)

A DCT [109] transforms a sequence of data points into a sum of cosine functions of various frequencies. DCT is a lossy compression technique that is widely used for the compression of images and audio. DCTs are used in the summation of cosine waves oscillating at different frequencies to convert data. The DCT uses of cosine functions, which makes it much more efficient than FFTs because fewer functions are required to approximate a signal.

2.8.2.2 Discrete HAAR Wavelet Transform (DHWT)

Discrete Haar wavelet Transform [110] is an efficient way of compressing both lossless and lossy data. DWHT is one of the simplest and most basic space domain and local frequency domain transformations. It depends on averaging and differentiating values in an image matrix to create a sparse or nearly sparse matrix . A sparse matrix is a matrix that contains a large proportion of 0. A sparse matrix can be efficiently stored, which results in smaller file sizes. This technique is the primary form of compression used in the research undertaken in this report.

2.9 Filter Bank

Filter banks are arrangements for the spectral decomposition and structure of signals of low pass, band-pass and high-pass filters. In many modern apps such as audio and image code processing, they play an significant role. The reason for their popularity is that the spectral parts of a signal can be rapidly extracted. Since most filter banks have different sampling rates, they are also called multi-rate systems [111].

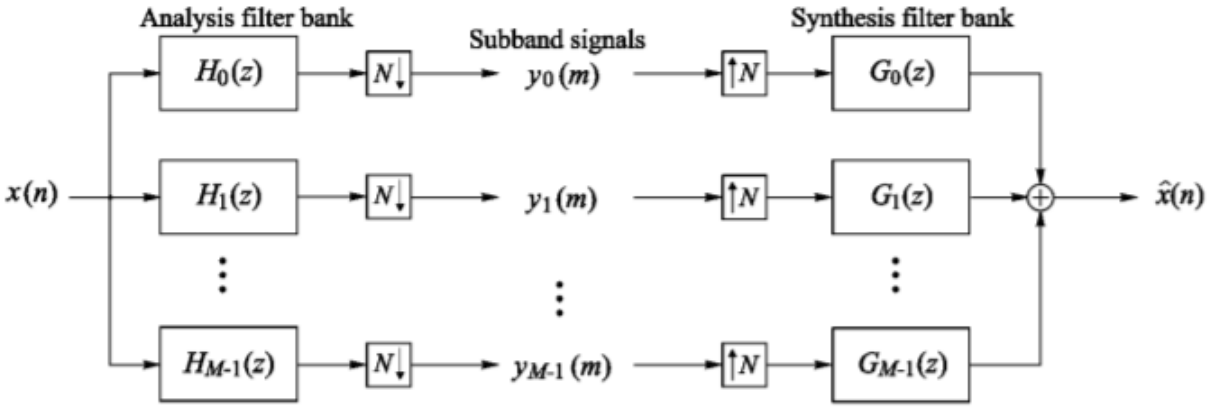


FIGURE 2.16: M-channel filter bank [112].

In Figure 2.16, the input signal is decomposed into M so-called sub-band signals by applying M different filters band-pass. Thus, each of the sub-band signals carries information on the input signal in a particular frequency band. The blocks with arrows pointing downwards indicate down sampling (sub-sampling) by factor N, and the blocks with arrows pointing upwards indicate up sampling by N [112].

Sub-sampling by N means that only every N-th sample is taken. This operation serves to reduce or eliminate redundancies in the M sub-band factor for which perfect reconstruction can be achieved. Perfect reconstruction means that the output signal is a copy of the input signal with no further distortion than a time shift and amplitude scaling. The DHWT is an example of wavelet based filter bank.

2.10 Multiresolution Wavelet Analysis

Generic wavelet transforms [113] perform multiresolution signal analysis on an input signal. Multiresolution signal analysis is decomposes the signal into multiple frequency bands, in order to process the signal in multiple frequency bands independently. Therefore, in both time and frequency domains, the wavelet should have compact support. Historically, it has been a difficult to find a kernel function that is compact in both time and frequency domains.

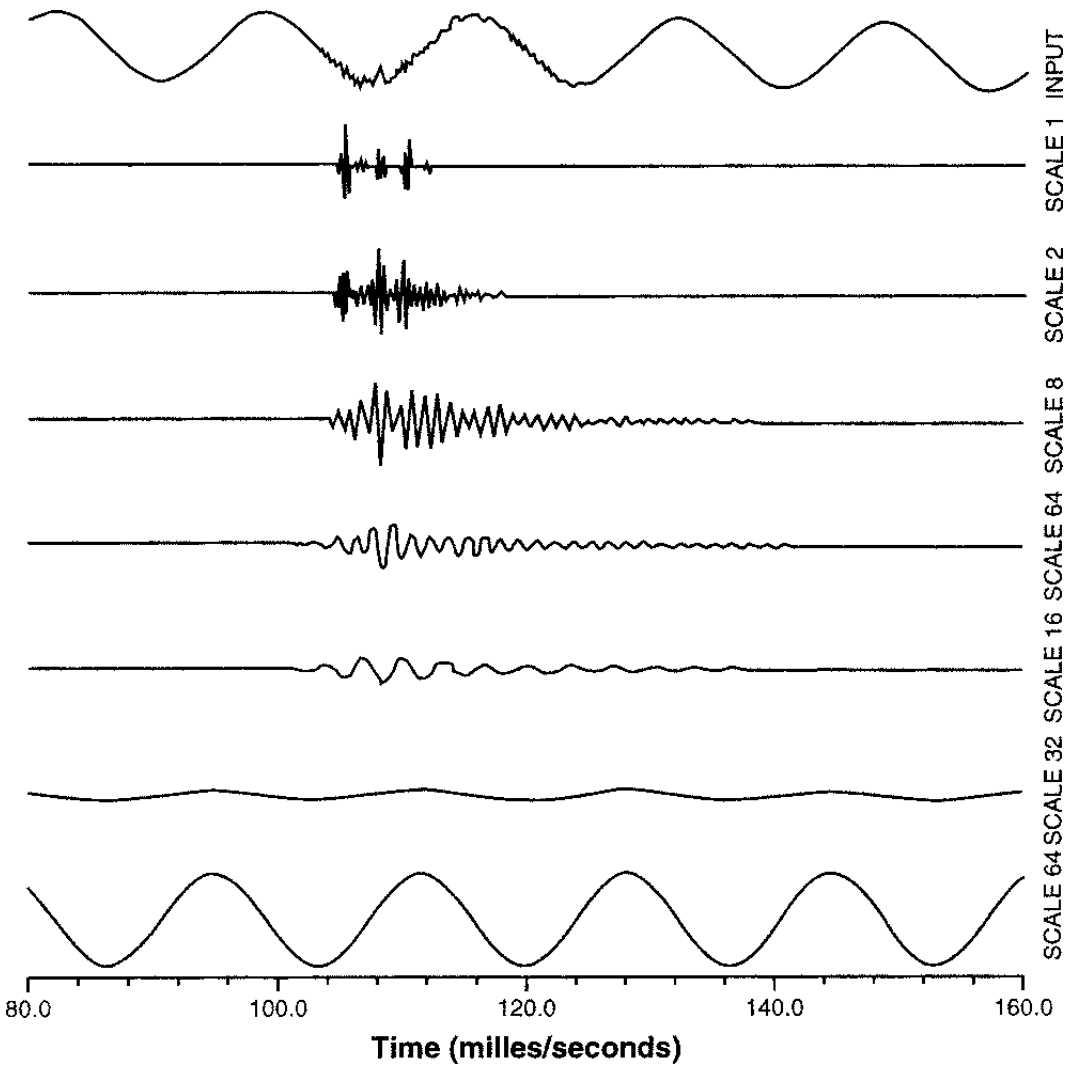


FIGURE 2.17: Multiresolution wavelet analysis of a transient signal in the electrical power system [113].

Figure 2.17 shows a typical multi-resolution wavelet analysis for a transient signal of an electrical power system. The signal is decomposed with different resolutions that match different wavelet scale factors. The signal components are well presented in the figure in multiple frequency bands and the times of occurrence of these components. This figure is a joint time-scale representation, with the vertical axis representing the amplitude of wavelet components in each discrete scale.

2.11 Python

Python is becoming popular every day, replacing numerous common industry languages [114]. The primary reason why Python is popular is that:

- Python is popularly known for its simplicity as the beginners language.

• Python assists developers from creation to implementation and maintenance in being much more productive.

• In comparison to Java, C and C++, the syntax of Python is very easy and high. Applications with less lines can therefore be created.

• Python has an extensive library collection.

for that The simplicity of python has resulted in a large number of libraries for Machine learning and Data Science [114].

2.11.1 Python library for Machine Learning

For the work carried out in this report,the most appropriate Machine Learning libraries for Python are :

2.11.1.1 Tensorflow

The Tensorflow's finest Open Source library. It was created by the Brain Team at Google [115]. Nearly all Google apps use Machine Learning Tensorflow. It is only a computing framework for the expression of algorithms that involve a large number of tensor operations, since the use of Tensorflow as a series of operations on tensors can be expressed as computational graphs. N-dimensional matrices representing our data are tensors.

2.11.1.2 Numpy

Numpy is one of Python's largest science and mathematics software libraries [116]. For performing several tensor activities, tensorflow and other platforms utilize Numpy internally. The Array interface is one of Numpy's key features.

This interface can be used as an array of real numbers with n dimensions to convey image, sound waves and raw binary streams. For machine learning and data science, knowledge of Numpy is very important.

2.11.1.3 Keras

The Keras Machine Learning Library is one of the most significant. It makes it simpler for Neural networks to express themselves. It also offers several utility for data set processing, models compilation, performance assessment, graph visualization, and much more.

2.11.1.4 Theano

Theano is a computational framework for multidimensional arrays computation [117]. Theano is comparable to Tensorflow, but Theano is not as effective as Tensorflow due to its failure to fit into manufacturing settings. Theano can be used in prallel or distributed settings just like Tensorflow.

2.12 Related Work

As it was mentioned before, Compression methods are rapidly being developed to compress large data files such as images and audio, where data compression has recently become more important in multimedia applications. With increasing technological growth and the entry into the digital age, a large amount of data must be properly stored using efficient methods to generally compress data while maintaining high quality and a marginal reduction in size [99]. For solving this conflict, many compression algorithms are suggested by the researchers like lossy compression and lossless compression. However, still the researchers think of developing a more suitable and more effective algorithm for achieving high compression and for decompressing the original information back.

2.12.1 Related work in images

For this purpose, the researchers did sufficient work and are still working on this scenario. Some of them are presented in this section; some research papers present the use of one dimensional (1D) DCT technique to achieve a compressed model for data compression. P.Kumar [118] presents the use of wavelet transformation to compress image using wavelet technique. Gupta presented the use of 2D DCT for compression of images [119]. Porwik [120] presents the use of different wavelet techniques to achieve image compression. His work shows the efficiency and results of different wavelet transformations special HWT. A similar work is presented Both HWT and wavelet transform are investigated and implemented for image compression purposes. The use of these methods depends on the high decompression ratio can provided 60-75%[121].

Moreover, wavelet transforms are commonly used for images with high PSNR to obtain a high compression rate in the compressed data and is used in lossy compression methods for some of the data compression standards [122]. In contrast to the discrete cosine transformation, the wavelet transformation is not based on trigonometric function but piece-wise functions and so wavelets handle data discontinuities better.

For instance, previous works using Haar image compression include an application for adaptive data hiding for images by dividing the original image into 8x8 sub - blocks and then reconstructing the images after compression with decent quality [123]. In addition, wavelet transformation has been used to digitally compress fingerprints and reconstruct original images via approximation components, horizontal detail, vertical detail and diagonal detail from of the input image [124].

In recent years, the implementation of ANNs in image processing applications has increased significantly. Image compression using wavelet transform and a neural network has been proposed [125]. In addition, for different applications, different image compression techniques were combined with a neural network classifier. A neural network model called direct classification was also proposed; this is a hybrid between a subset of the self-organising Kohonen model and the model of adaptive resonance theory to compress the image data [126]. An algorithm based on vector quantization image compression has been proposed based on a competitive neural networks quantizer and neural networks predictor [127] [128].

More work has recently emerged on the application of ANNs to imaging. Neural multi-resolution filter bank (MRNN) and its potential as a coding transformation for a sub-band coding framework has been proposed [129].

A method of direct solution for compression of images using neural networks [130] has been suggested using image compression based on the principle component analysis. In addition, It was suggested in 2005 that a neural network quantizer could be used compress an image at a high compression ratio with loss and then compressing the error image without loss. This results in an image that is not only strictly lossless, but is also expected to produce a high compression ratio, especially if the loss compression technique is decent [131].

2.12.2 Related work in Audio

For speech compression, various techniques is implemented . Transform coding is based on compression of signal by removing redundancies is presented in [132] . It is a process of transforming signal into compressed or compact form so that the signal could be stored with less bandwidth. In this paper DCT and DWT based speech compression techniques are implemented with Run Length Encoding, Huffmann Encoding. Reconstructed signals are compared using factors like Signal to Noise Ratio (SNR), Peak Signal to Noise Ratio (PSNR). Jaber and Tanboura [133] gave a lossy algorithm to compress speech signal using DWT techniques. Due to growth of multimedia technology over the past decade, demand for digital information has increased. The only way to overcome this situation is to compress the information signal by removing the redundancies present in them. The compression ratio can be easily varied by using wavelets while other methods have fixed compression ratios. they presented Increasing the scale value in wavelet-based speech coder gives higher compression ratios, but this cost decreasing in quality of the speech signal. Habib Harding and Aisha-Hassan [134] described the importance and need for audio compression. Audio compression has become one of the most basic technologies

of this period.

Besides, a lot of work has been done in speech recognition [135], [136]. A computer intelligence solution usually uses a classifier to recognise speech. Three most commonly used classifiers are: Dynamic time warping (DTW), hidden Markov model (HMM) and artificial neural networks (ANN). Researchers have used these techniques with various constraints to obtain robust results. A combination of ANN and HMM with spatio-temporal ANN used on a small number of speech samples [137]. A vector based on the wavelet features is tested with DTW . Another approach to speech recognition uses wavelet transformation and ANN as the final classifier in an ASR speaker [135]. The study proposes a fuzzy inference system (WPANFIS) based on a wavelet packet adaptive network. The sample speech signals are reported to be 92% accurate. Detailed methodologies exist for the recognition of speech by using Short-Time Fourier Transformation (STFT) with ANN as the final classification [138]. STFT is the analysis technique based on the constant size of the window when the speech sample is analysed.The suggested technique is based on the wavelet analysis that can analyze a speech sample using a differential window size based on the speech segment frequency [138].

A speech recognition support system called BizVoice is proposed to access speech during a session and convert voice data to text data using speech recognition technology [139]. During a classroom session at Aoyama Gakuin University, the efficiency of the system was evaluated. In [140], a system based on a combination of the deep bidirectional LSTM recurrent neural network (RNN) architecture and the Temporary Contortionist Classification Objective function is presented. The approach transcribes audio data directly with text. The error rate is reported at 6.7 %. The work shows the transcription of speech at the level of character using a RNN. An interesting way to recognise speech using convolutionary neural networks (CNNs) is presented in[141]. Results show a 6-10 % reduction in error due to CNNs. The speech in this paper [141] is analysed using a Hamming window of 25 ms with a fixed frame rate of 10 ms. Data samples are standardised to a zero mean and unit variance.

An approximate method is proposed to convert a feed-forward NNLM to a back-off ngram language model [142]. The model is used to recognise speech. The approach can be used for efficient decoding of the non-back-off language model. The work in[143] uses deep RNN for speech recognition. The approach is evaluated on the second challenge of CHiME (track 2) and Aurora-4 tasks, which show an improvement of 7 percent and 4 percent respectively. The accelerated implementation of the Lithuanian isolated word recognition system is presented [144]. Two databases are used for testing the approach . The first database contains 100 words,

each ten times pronounced. The second database consists of ten words each spoken ten times by ten speakers. The work in [145] presents a DTW-based approach to speech recognition in an intelligent electric wheelchair (Dynamic Time Warping). Mel Frequency Cepstral Coefficients (MFCCs) are used to obtain key features. The voice signals are then transformed into different commands. To recognise isolated Urdu words, an approach based on linear discriminant analysis is presented in[146]. 52 Mel Frequency Cepstral Coefficients are extracted for each item in the database for the purpose of classification. In the system, audio samples of seven speakers were used. For most samples, the percentage error of less than 33 percent is reported.

A comparative analysis of features based on DWT and MFCC is presented that [147] the word database is selected from the most commonly used Urdu words. For speech recognition, a system using an open source language recognition framework called Sphinx4 was presented in [148]. The language recognition targets the Urdu language with a 52 isolated words vocabulary.

2.13 Summary

In this chapter a review of the literature to find the current state of play in relevant ML space was undertaken.The definitions, algorithms, and applications for AI, ML, KDD, ANN, The MINIST data set, and the T146 speech set were presented and their contentions discussed. concepts pertaining to compressed data, Multi-resolution Analysis, and Filter banks were discussed along with some of their more common application .

The next chapter provides a detailed explanation of ANNs and how the network architecture was selected for the work presented in this research.

Chapter 3

Artificial Neural Networks

3.1 Overview

The mathematical model of a neural network is considerably simple than biological neural networks [92]. Furthermore such networks are based on more than a few assumptions. One of them is that all neurons are synchronised; the signal passing from one neuron to another takes the same time for all connections.

Another significant assumption is that every neuron is a transfer function, which determines a neuron's output signal based on the input signal's strength, and is time-independent.

Once the signal passes the synapse, it is assumed change to linearly, i.e. the signal value is multiplied by a number known as synaptic weight. These assumptions greatly simplify the initial biological neural network. Despite these simplifications, artificial neural networks still preserve the most significant characteristics of biological networks - adaptability and ability to learn [92].

In general, an ANN is a set of artificial neurons arranged in layers. It is a powerful tool for solving problems such as classification or prediction [149]. Feed-forward neural networks are the most commonly encountered and are used in many diverse applications. Back-propagation algorithm is the mostly used method in the training of feed-forward neural networks, but it is also used, along with modified versions of the algorithm, in the training of other types of neural networks.

This chapter describes the multi-layer feed-forward back-propagation algorithm used for supervised learning. The choice of the network architecture is discussed, including the numbers of hidden neurons and layers, this number of input and output nodes. Concepts such as the bias, learning rate, the momentum, synaptic weight, epochs and activation functions are also introduced.

3.2 The fundamentals of ANN

Figure 3.1 shows a typical multi-layer ANN are divided into layers. Input and output neurons form visible input and output layers. Hidden neuron layers provide the inter-connectivity from the input layer to the output layer. Every neuron, except for input neurons, is connected by synapses with all neurons of the previous layer [149].

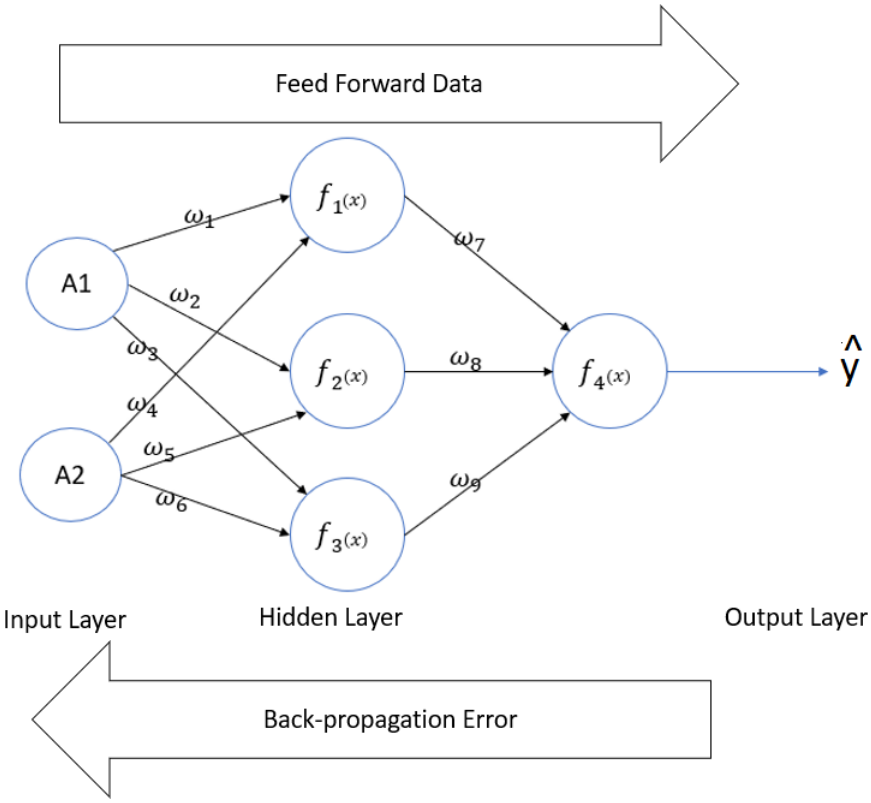


FIGURE 3.1: A schematic diagram of artificial neural network and architecture of the feed forward network with one hidden layer.

3.2.1 The Layers of ANNs

Input Layer

This layer has neurons take input vectors that encode external environment actions or information. Input neurons do not perform any type of computation, but only pass the impact to neurons in the next layer (the first hidden layer) [150].

Output Layer

It receives and transforms signals from the preceding layer's neurons. These values represent the entire neural network's output [150].

Hidden Layer

These neurons receive the signal from the input neurons, or preceding hidden layer neurons in multi-layer ANNs, process it and then transmit the resulting signals to the next layer's neurons(hidden or output) [150].

In most implementations, there is no clear way to determine the best number of hidden neurons without training several networks to estimate the generalisation error for each network.

Too few hidden units can cause a high training error due to under-fitting. Too many hidden units can result in a low training error, but still have a high generalisation error due to over-fitting.

Furthermore, if the number of neurons is lower than the complexity of the problem data, it leads to under-fitting [151].

A common Rule of thumb method for determining the correct number of neurons in the hidden layers states that [151]:

- The number of hidden neurons should be in the range between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 of the input layer size, plus the size of the output layer.
- The number of hidden neurons should be less than twice the input layer size.

3.2.2 Synaptic Weight

The idea of synaptic weight is a foundation concept in artificial neural networks. A set of weighted inputs allows each artificial neuron or node in the system to produce related outputs. Professionals dealing with machine learning and artificial intelligence projects where artificial neural networks for similar systems are used often talk about weight as a function of both biological and technological systems.

The most important feature of synaptic weight is that it changes over time. Time dependence makes it possible for the brain to react differently at different times on the same input. The most common synaptic weight initialisation method selects small random numbers with zero mean and uniform distribution value [152].

3.2.3 The Bias

Bias is just like an intercept added in a linear equation. It is an additional neuron added to each pre-input layer that contains the value of one. It is connected to any previous layer and consequently doesn't represent a true activity [150]. which is

used to adjust the output along with the weighted sum of the inputs to the neuron. Moreover, bias value allows you to shift the activation function to either right or left.

$$output = sum(inputs * weights) + bias \tag{3.1}$$

The output is calculated by multiplying the inputs with their weights and then passing it through an activation function like the sigmoid function, etc. Here, bias acts like a constant which helps the model to fit the given data. The steepness of the sigmoid depends on the weight of the inputs.

It allows you to move the line down and up fitting the prediction with the data better. If the bias is absent then the line will pass through the origin (0, 0) and you will get a poorer fit.

3.2.4 Gradient Descent

Gradient descent is an efficient optimisation algorithm that works by trying to find a local or global function minimum. It allows a model to use the gradient or direction to reduce errors (differences between the existing y and the predicted y). As the model iterates, it converges gradually to a minimum where further adjustments to the parameters lead to little or no loss change. Thus, the learning process enables corrective updates to the learned estimates that move the model towards an optimal parameter combination convergence [153].

The alternative to the process of gradient descent would be a brute force a approach with a potentially infinite combination of parameters to identify the set that minimises the cost.

Learning rate [152] is a gradient descent parameter. It is a number between 0 and 1 that determines how quickly the ANN adjusts to the training data patterns. It isn't necessarily constant and can decrease or increase over time. This parameter must be carefully selected; too small a parameter makes the learning process slow and too large it can cause a undesirable divergent behaviour in loss function.

3.2.5 The ANN algorithm using Gradient Descent

A short overview of feed-forward neural networks is presented here. Feed-forward neural networks with gradient descent are the most widely used in a variety of applications . Therefore, this type of ANN is chosen to illustrate the artificial Neural

network. The back-propagation algorithm is also described in detail. The back-propagation algorithm is the most commonly used algorithm in the training of feed-forward neural networks, but it is also used in the training of other types of neural networks together with the modified versions of the algorithm.

Feed-forward Neural networks

The neuron can be considered to be a simple machine of black box transforming input signals into output. Operation of the model can be summarised as follows:

- Neuron synapse inputs receive N signals$[A_1, ..., A_N]$.
- Each synapse changes the signal linearly using its synaptic weight$A_i w_i$.
- The set of weighted $[A_1 w_1, ..., A_N w_N]$ are summed.
- An additional input value with a weight value of one to represent a neuron's threshold or bias [154] can be added to the sum of input signal.
- The sum, x_N, applied to activation function.
- Output is given.

The summation function is executed as:

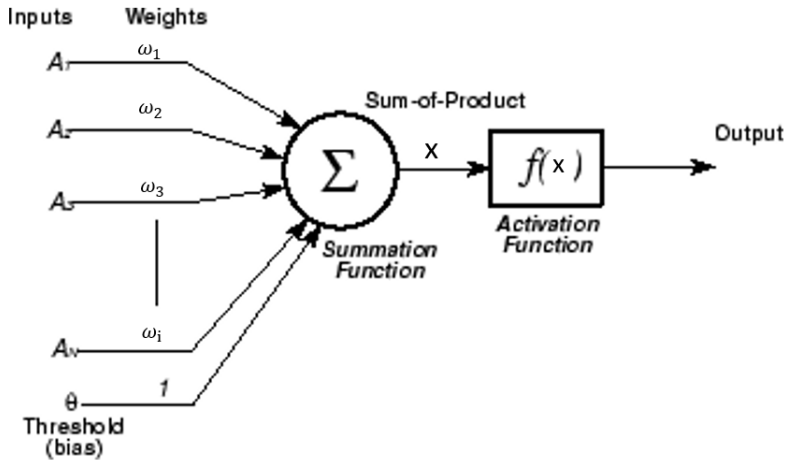


FIGURE 3.2: Node from ANN training procedure.

$$x_N = \sum_{i=1}^{N} A_i w_i + \theta \tag{3.2}$$

In this case:

- N is the number of synaptic weights for all layer neurons. This number is determined by the number of neurons in the previous layer and is the same for all the neurons in the same layer.

•x_N is the calculate the weighted sum of the inputs f a non-linear transfer function (activation function, will be discussed in section 3.3) to produce the current output for the neuron [152].
• the final signal is sent to the next layer $(\hat{y})$.

$$\hat{y} = f(x_N) \tag{3.3}$$

The network transmits the input vector from layer to layer and calculates the outputs sequentially for all the neurons of that layer. For the next layer these outputs form an input vector. The output signals are calculated from the previous layer output signals for each neuron in the layer (except for input neurons).

Initially weights for all layers are randomly assigned process varies the threshold and weight coefficients to minimise the sum of the square of the differences between calculated $\hat{y}$ and required output value y . This is achieved by reducing the loss function, L:

$$L = \sum \frac{1}{2}(y - \hat{y})^2 \tag{3.4}$$

The back-propagation Algorithm

The change in the weight resulting from this error reduction is fed back into earlier layers using back-propagation[155]. During back-propagation, synaptic weights are adjusted to produce a smaller error value δ. These adjustments are derived from the error obtained in equation 3.5. The idea is to propagate the error signal (computed in one forward teaching step) back to all neurons connected to the output neuron being considered.

If the sigmoid function is used as activation function and square error as loss function we can rewrite it as

$$\delta = \frac{\partial L}{\partial \hat{y}} = \frac{\partial(\frac{1}{2}\sum(y - \hat{y})^2)}{\partial \hat{y}} = (y - \hat{y}) \tag{3.5}$$

During this propagation phase the nodal weights in each layer are left unchanged initially. Errors for these nodes are calculated and the the weight adjusted to reduce these errors. These are then fed back into the previous layer (with initially unchanged weights) with errors for this layer now calculated. Weights for nodes in this layer are changed to reduce the error and the process continues to the next layer back until all layers are processed [154].

The equation in 3.6 show how the δ error values at Each Node are calculated for the ANN in Figure 3.3.

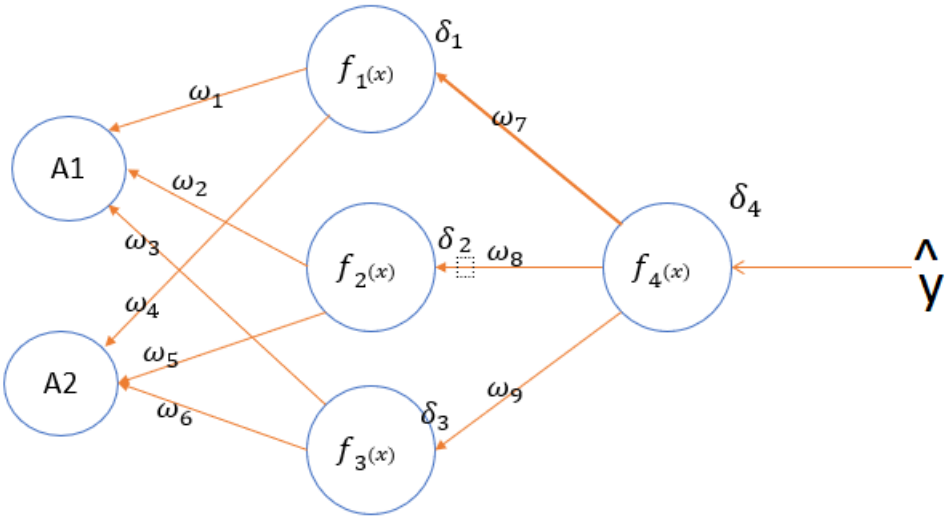


FIGURE 3.3: Calculation of the Error at Each Node and weight updating in propagation process.

To minimise the error, Gradient Descent is often used. The weights for the layers in the ANN are adjusted and updating as per the equation in 3.7.

Here $f'(x)$ refers to the derivative with respect to x of the activation function (sigmoid function) in the hidden layer and η is the learning rate parameter.

Output_error(δ_4) = (target-output) x derivative of the output

$$\begin{aligned} \delta_4 &= (y-\hat{y}) \times f_4'(x) = (y-\hat{y}) \times (1-\hat{y}) \times \hat{y} \\ \delta_1 &= w_7 \times \delta_4 \times f_1'(x) = w_7 \times \delta_4 \times (1-f_1(x)) \times f_1(x) \\ \delta_2 &= w_8 \times \delta_4 \times f_2'(x) = w_8 \times \delta_4 \times (1-f_2(x)) \times f_2(x) \\ \delta_3 &= w_9 \times \delta_4 \times f_3'(x) = w_9 \times \delta_4 \times (1-f_3(x)) \times f_3(x) \end{aligned} \tag{3.6}$$

$$new\_weight = old\_weight + (learning\_rate \times error\_value \times input)$$

$$
\begin{aligned}
w_1 &= w_1 + \eta\delta_1 A_1 \\
w_2 &= w_2 + \eta\delta_2 A_1 \\
w_3 &= w_3 + \eta\delta_3 A_1 \\
w_4 &= w_4 + \eta\delta_1 A_2 \\
w_5 &= w_5 + \eta\delta_2 A_2 \\
w_6 &= w_6 + \eta\delta_3 A_2 \\
w_7 &= w_7 + \eta\delta_4 f_1(x) \\
w_8 &= w_8 + \eta\delta_4 f_2(x) \\
w_9 &= w_9 + \eta\delta_4 f_3(x)
\end{aligned}
\tag{3.7}
$$

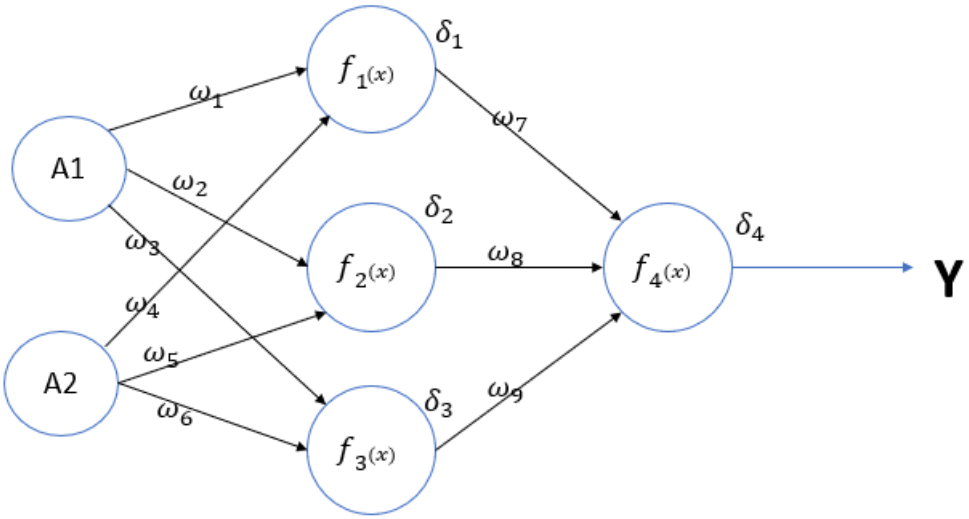


FIGURE 3.4: ANN after the weights are now adjusted to minimize the error in the output.

In Figure 3.4, if the ANN runs again with these new values, as the weights are now adjusted to minimize the error in the output. The output node errors have now been reduced compared to earlier. In other words, our network has learned to classify our first training example better; i.e it is learning.

This process is repeated with all other examples in the input dataset. It is then said that the ANN has learned from these examples.

3.2.6 Other fundamentals effect in ANN

Momentum

Momentum [156] influences the way in which previous weights affect the current one. It helps prevent the algorithm from being stuck in a local minimum. This value must also be carefully selected and experimentally determined. Momentum can be omited in an ANN. However, it can greatly improve the performance of the ANN and is therefore commonly used.

If the momentum is high, the rate of learning should be set lower. A large momentum value can result in the convergence happening quickly. However, if both the momentum and the learning rate are maintained at a high level, the minimum could be missed with a big step.

Epochs

One epoch has elapsed when the whole dataset [156] is passed forward and backward once through the ANN.

As the number of epochs increases, the weights in the ANN change and the the model of ANN goes from underfitting to optimal to overfitting.

Batches

The batch is a group of samples. The gradient descent of mini-batches is the recommended gradient descent variant for most applications, especially in deep learning [152].

Mini-batch sizes, commonly referred to as batch sizes for briefness, are often tuned to a computer architecture on which the implementation is carried out; e.g. powers of two that fit the GPU or CPU hardware's memory requirements.

For batches:

- Small sizes provide a learning process that converges rapidly at the cost of noise.
- Large values give a learning process that converges slowly with accurate estimates of the error gradient.

Regularisation

It's a technique used the problem of overfitting of the ANN [153]. Two types of regularisations are:

- *L1 Regularisation or Lasso Regularisation*

Introduces an error penalty function that is the sum of the absolute weight values; i.e.

$$L(x,y) = \sum_{i=1}^{n}(y_i - w_i x_i)^2 + \underbrace{\rho \sum_{i=1}^{\pi} |w_i|}_{\text{Error Function(L1).}} \tag{3.8}$$

where ρ is the penalty term or regularisation parameter that determines how much to the weights and i is input variables.

When ρ is zero then the regularisation term becomes zero. We are back to the original loss function.

In L1 regularisation we penalise the absolute value of the weights.

• *L2 Regularisation or Ridge Regularisation*

In L2 regularisation, the sum of square of all feature weights is used (as shown equation 3.9). It forces the weights to be small but does not make them zero.

L2 is not robust to outliers as the squared terms amplifies the error differences of the outliers and the regularisation term tries to fix it by penalising these weights.

$$L(x,y) = \sum_{i=1}^{n}(y_i - w_i x_i)^2 + \underbrace{\rho \sum_{i=1}^{\pi} (w)^2}_{\text{Error Function (L2)}} \tag{3.9}$$

3.3 Activation Functions

The main purpose of the activation function is to convert an input signal of a node in an ANN to an output signal. This signal becomes an input to the nodes at the next layer [157].

In an ANN layer the sum of input products (x) and their corresponding weights (w) use the activation function $f(x)$ to create that layer's output feed to the next layer [157]. The output signal would be a simple linear function if the activation function was not applied. A linear equation is easy to solve, but its complexity is limited and it has less power to learn complex functional mappings from input data. An ANN without an activation function would simply be a linear regression model with limited appeal. Therefore, an ANN would not be able to learn effectively without an activation function [158].

Some of the popular Types Of activation functions described in the following sections.

TABLE 3.1: Different Activation Functions for ANN.

Activation Function	Equation	Derivate	Range
Sigmoid	$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$	$(0,1)$
Rectified Linear Units (RELU)	$f(x) = max(0,x)$	$f'(x) = \begin{cases} 1 & x > c \\ 0 & x \leq c \end{cases}$	$[0,\infty)$
Leaky (RELU)	$f(x) = \begin{cases} ax & x < 1 \\ x & x >= 1 \end{cases}$		$(-\infty,\infty)$
Softplus	$f(x) = \ln(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$	$(0,\infty)$
Tanh	$f(x) = \tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$	$f'(x) = 1 - f(x)^2$	$(-1,1)$

3.3.1 Sigmoid

The Sigmoid function has the mathematical form:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.10}$$

Its derivative is

$$f'(x) = f(x)(1-f(x)) \tag{3.11}$$

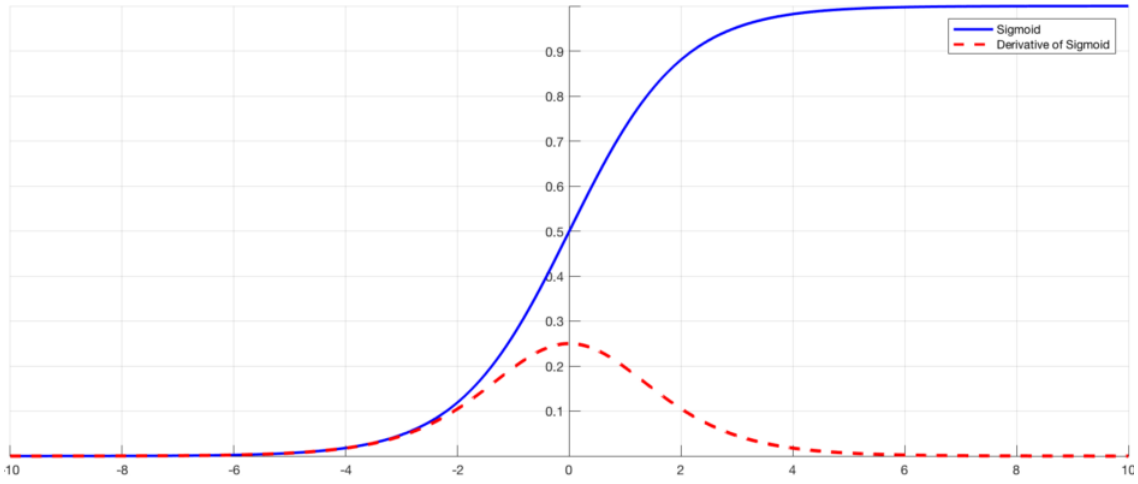


FIGURE 3.5: Sigmoid Function and derivative of it [159].

The sigmoid function domain is R and its range is [0,1]. Large negative numbers are effectively 0 and large positive numbers effectively 1. The sigmoid function [160] has historically been used because it mimics well the firing rate of a neuron; from not firing at all (0) to fully saturated firing at a maximum frequency (1).

It has two main disadvantages: sigmoid saturates and kills gradients. A very able characteristic of the sigmoid function is that when the activation of the neuron saturates at either tail of 0 or 1 the gradient is almost zero in these regions. Therefore, if the local gradient is very small, it can effectively kill the gradient and almost no signal flows through the neuron to its weights and its data. In addition, extra caution must be paid when initialising the weights of sigmoid neurons in order to prevent saturation. If the initial weights are too large, for example, most neurons would be saturated and the network doesn't effectively learn.

3.3.2 Rectified Linear Units (ReLU)

In the last few years, ReLU has become popular. The threshold of this activation function is zero [161]. The function is defined:

$$f(x) = max(0, x) \tag{3.12}$$

The derivative of ReLU is

$$f'(x) = \begin{cases} 1 & : x > \epsilon \\ 0 & : x \leq \epsilon \end{cases} \tag{3.13}$$

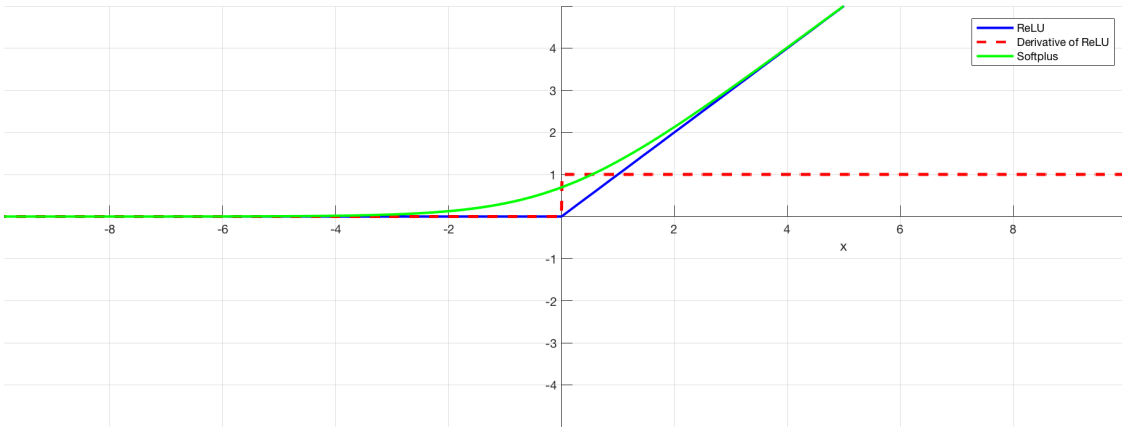


FIGURE 3.6: ReLU Function and its derivative [159]

There are several advantages and disadvantages with the use of ReLUs [161]. The main advantages is the convergence of stochastic gradient descent, compared to the sigmoid and tanh functions, is greatly accelerated. It is argued that this is because of its linear, unsaturated shape. In addition, tanh and sigmoid neurons involve costly operations (exponentials, etc.), whereas the ReLU can be implemented simply by setting a zero activation matrix.

However, ReLU units can be fragile and die during training. For example, a large gradient that flows through a ReLU neuron can update the weights so that the neuron never again activates at any data point. If this happens, the gradient that flows through the unit from that point is zero forever. The ReLU units can therefore die irreversibly during training.

Leaky ReLU

Leaky ReLU is an attempt to solve the problem of a dying ReLU. Instead of having zero for x < 0, a leaky ReLU has a small negative slope [161]. The function is defined:

$$f(x) = \begin{cases} \alpha x & x < 1 \\ x & x >= 1 \end{cases} \tag{3.14}$$

where α is a small constant.

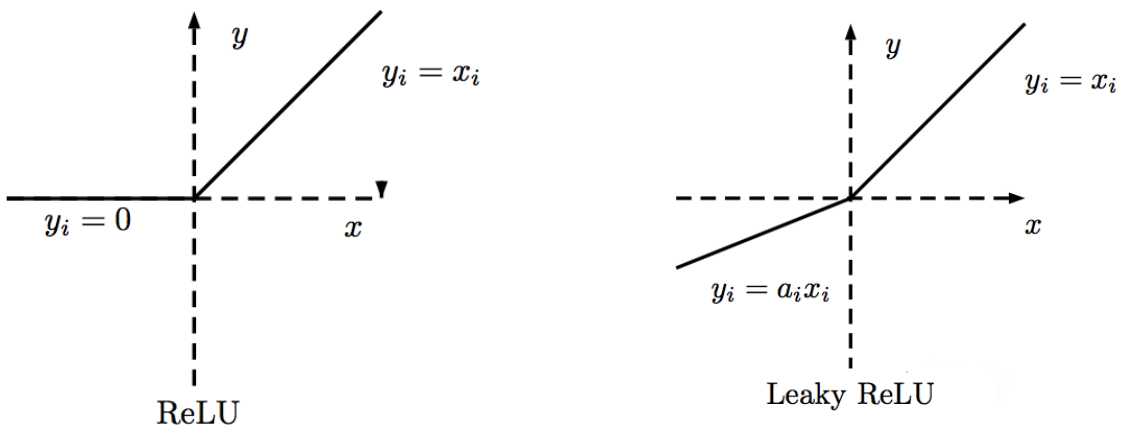


FIGURE 3.7: (A) ReLU Function (B) Leak ReLU Function [159]

3.2.10.2 Softplus

Softplus is a smooth rectifier approximation. Its mathematical formula is

$$f(x) = \ln(1 + e^x) \tag{3.15}$$

The derivative is

$$f'(x) = \frac{1}{1 + e^{-x}} \tag{3.16}$$

The softplus derivative is the logistical function. The ReLU and softplus are largely similar, except near zero, where the Softplus is smooth and distinguishable [162]. However, it is much easier and more efficient to calculate ReLU and its derivative than the softplus function in its form. However, the non-linear nature of the softplus activation function is necessary for deep ANNs. Furthermore, it doesn't suffer from saturation at large values as happens to the sigmoid.

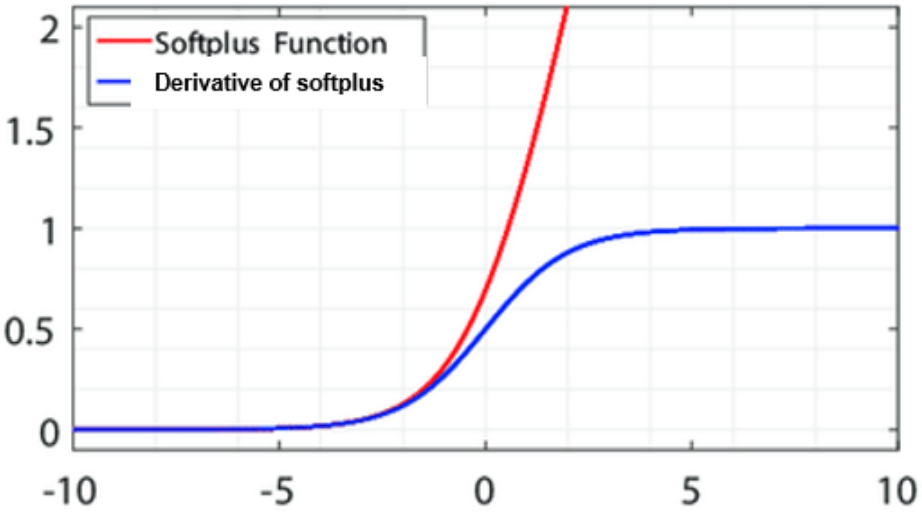


FIGURE 3.8: Softplus Function and derivative of it [162]

3.2.10.3 Hyperbolic Tangent Function- Tanh

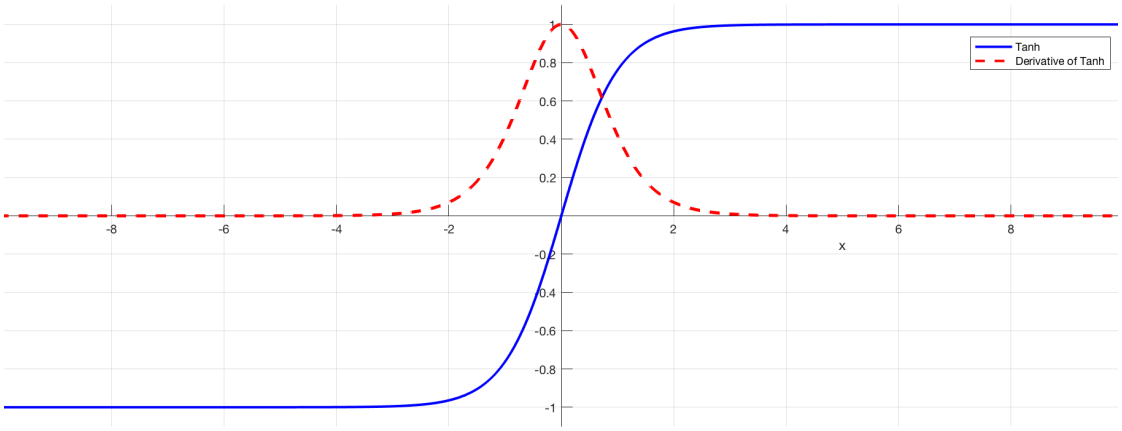


FIGURE 3.9: Tanh Function and its derivative [159]

Like the sigmoid neuron, tanh activations saturate, but its output is zero-centered, unlike the sigmoid neuron. Note also that the tanh neuron is just a scaled sigmoid neuron. Its mathematical formula is

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.17}$$

The derivative is

$$f'(x) = 1 - (f(x))^2 \tag{3.18}$$

3.4 Example of simple ANN.

To illustrate the backpropagation training algorithm, let's consider the ANN below:

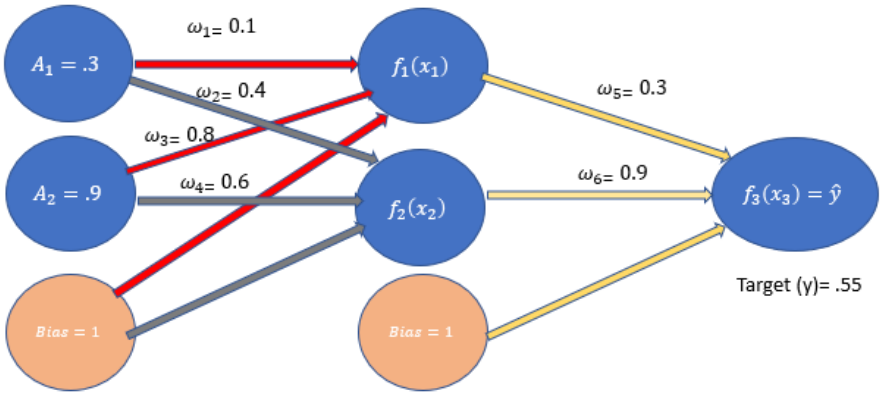


FIGURE 3.10: Example ANN illustrating backpropagation training algorithm.

This network, shown in Figure 3.10, has two nodes in the input layer, two nodes in the hidden layer, and a single node in the output layer. Assume that the nodes have a Sigmoid activation and the learning rate is 0.01.So let's:

(1) Perform a forward pass on the network.

(2) Perform a reverse pass (training) once (target = 0.5).

(3) Perform a further forward pass and comment on the result.

from equation 3.2 to calculate the sum of production. a Sigmoid activation function from equation 3.10.

$$x_1 = (0.3 \times 0.1) + (0.9 \times 0.8) + 1 = 1.75$$

$$f_1(x_1) = 1/(1 + e^{-1.75}) = 0.852$$

$$x_2 = (0.3 \times 0.4) + (0.9 \times 0.6) + 1 = 1.66$$

$$f_2(x_2) = 1/(1 + e^{-1.66}) = 0.840$$

$$x_3 = (0.852 \times .3) + (0.84 \times 0.9) + 1 = 2.012$$

$$f_3(x_3) = \hat{y} = 1/(1 + e^{-2.012}) = 0.882$$

The equation in 3.6 show how the delta error values are calculated for the ANN in Figure 3.11.

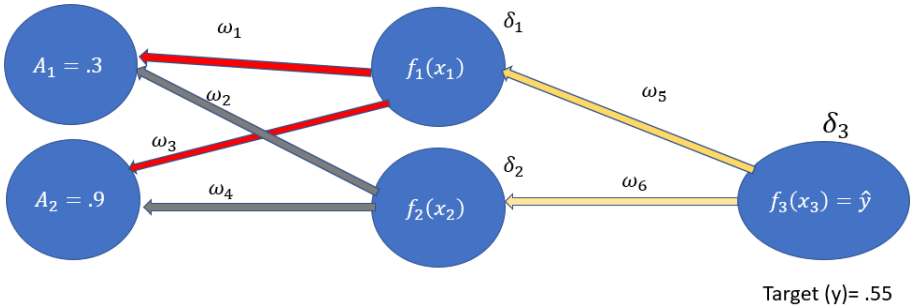


FIGURE 3.11: Example backpropagation error calculation algorithm.

$$\delta_3 = (y - \hat{y}) \times (1 - \hat{y}) \times \hat{y} = (0.55 - 0.882) \times (1 - 0.882) \times 0.882 = -0.0322$$

$$\delta_1 = w_5 \times \delta_3 \times (1 - f_1(x)) \times f_1(x) = 0.3 \times -0.332 \times (1 - 0.852) \times 0.852 = -1.22x10^{-3}$$

$$\delta_2 = w_6 \times \delta_3 \times (1 - f_2(x)) \times f_2(x) = 0.9 \times -0.332 \times (1 - 0.84) \times 0.84 = -3.90x10^{-3}$$

The weights for the layers in the ANN are adjusted and updating as per the equation in 3.7

$$w_1 = w_1 + \eta\delta_1 A_1 = 0.1 + (0.01 \times -1.22x10^{-3} \times 0.3) = 0.099$$

$$w_2 = w_2 + \eta\delta_2 A_1 = 0.4 + (0.01 \times -3.90 \times 10^{-3} \times 0.3) = 0.399$$

$$w_3 = w_3 + \eta\delta_1 A_2 = 0.8 + (0.01 \times -1.22x10^{-3} \times 0.9) = 0.799$$

$$w_4 = w_4 + \eta\delta_2 A_2 = 0.6 + (0.01 \times -3.90x10^{-3} \times 0.9) = 0.599$$

$$w_5 = w_5 + \eta\delta_3 f_1(x_1) = 0.3 + (0.01 \times -0.0322 \times 0.852) = 0.299$$

$$w_6 = w_6 + \eta\delta_3 f_2(x_2) = 0.9 + (0.01 \times -0.0322 \times 0.840) = 0.899$$

repeat the process after updating the weight

$$x_1 = (0.3 \times 0.099) + (0.9 \times 0.799) + 1 = 1.748$$

$$f_1(x_1) = 1/(1 + e^{-1.75}) = 0.851$$

$$x_2 = (0.3 \times 0.399) + (0.9 \times 0.599) + 1 = 1.656$$

$$f_2(x_2) = 1/(1 + e^{-1.66}) = 0.839$$

$$x_3 = (0.851 \times .299) + (0.839 \times 0.899) + 1 = 2.009$$

$$f_3(x_3) = \hat{y} = 1/(1 + e^{-2.012}) = 0.869$$

The old error was = (0.55-0.882) = -0.332 and the new error is = (0.55-0.869) = -0.319.

Therefore, the error has reduced. The training continues until the network converges to a solution with acceptable tolerance of error, or reaches the maximum defined number of iterations or epochs.

3.5 Summary

This chapter has examined, in details, the multi-layer feed-forward back-propagation algorithm that belongs to the group of ML algorithms called supervised learning. The mechanisms of how an ANN can learn were also considered along with the issues pertaining to ANN architecture selection.

In the next chapter the Haar Wavelet Transform will be examined in detail along with its use in image compression.

Chapter 4

Haar Wavelet Transform

4.1 Overview

As mentioned previously, The Haar wavelet transform (HWT) is widely used in several specialised data processing applications for signal and image processing. In this chapter the following are covered:

- Fundamental definitions and the different kinds of wavelets.
- The fundamentals of the Haar wavelet transform.

4.2 Wavelet

A wavelet [163] is a mathematical function that decomposes data into different frequency components allowing each component to be studied while matching a resolution to its scale. In fact, wavelets are functions that transform data using scalable, orthogonal, and, in some cases, complete mathematical functions. In many fields, such as image analysis, communication systems, bio-medical imaging, radar and other signal processing domains, the wavelet is an excellent tool. The use of wavelets, for these purposes, is relatively recent although the theory is not new. A wavelet transform is the representation of a function using wavelets.

4.2.1 Examples of Wavelet Family

Each family of wavelets [163] has a different shape, smoothness and compactness and is useful for a different purpose. As it is shown in Figure 4.1. The two mathematical conditions a wavelet has to satisfy are the normalisation and orthogonalisation constraints. A wavelet must have finite energy and zero mean.

Finite energy means that it is localised in time and frequency; it is integrable and the inner product between the wavelet and the signal always exists. A zero mean in the time-domain therefore a zero at zero frequency, necessary to ensure that it is integrable and the inverse of the wavelet transform can also be calculated.

4.2.1.1 Haar wavelet

The Haar wavelet [164] is the simplest orthonormal wavelet basis. The Haar wavelet is conceptually simple, memory efficient, exactly reversible (without the edge effects characteristic of other wavelets) and computationally cheap. The Haar transform does not have overlapping windows and measures only changes between adjacent pixel pairs. it uses just two scaling and wavelet function coefficients, thus calculates pair wise averages and differences.

4.2.1.2 Daubechies

The Daubechies wavelet family [165] is the most popular wavelet family used for texture feature analysis, due to orthogonal and compact support abilities. The Daubechies wavelet uses overlapping windows, so the results reflect all changes between pixel intensities. The Daubechies D4 transform has four wavelet and scaling coefficients. The sum of the scaling function coefficients are one, thus the calculation is averaging over four adjacent pixels.

4.2.1.3 Symlets

Symlets [165] are Daubechies least symmetric wavelets and are very compactly supported. The construction of symlets is very similar to that of Daubechies but their symmetry is stronger than that of Daubechies. Because, the associated scaling filters are near linear-phase filters.

4.2.1.4 Gabor

Gabor wavelets were invented by Dennis Gabor [166] using complex functions constructed to serve as a basis for Fourier transforms in information theory applications. They are also closely related to Morlet wavelets and Gabor filters. The important property of the wavelet is that it minimised the product of its standard deviations in the time and frequency domain; the uncertainty in information carried by this wavelet is minimised. However they have the downside of being non-orthogonal, so efficient decomposition into the basis is difficult. Since their inception, various applications have appeared, from image processing to analysing neurons in the human visual system.

4.2.1.5 Coiflets

Coiflets [164] were originally derived from the Daubechies wavelet. They have an even higher computational overhead and use windows that overlap more. Coiflets use six scaling and wavelet function coefficients. This increase in pixel averaging and differencing leads to a smoother wavelet and increased capabilities in several image-processing techniques (de-noising images, etc.). The filter follows the same structure as both Haar and Daubechies. It calculates both averages and differences using the same format, only with six adjacent pixels.

Daubechies wavelet families are more complex and generally have a higher computational overhead than the Haar Wavelet. In some applications, the Haar wavelet performs better than Daubechies wavelets, especially in image compression [165].

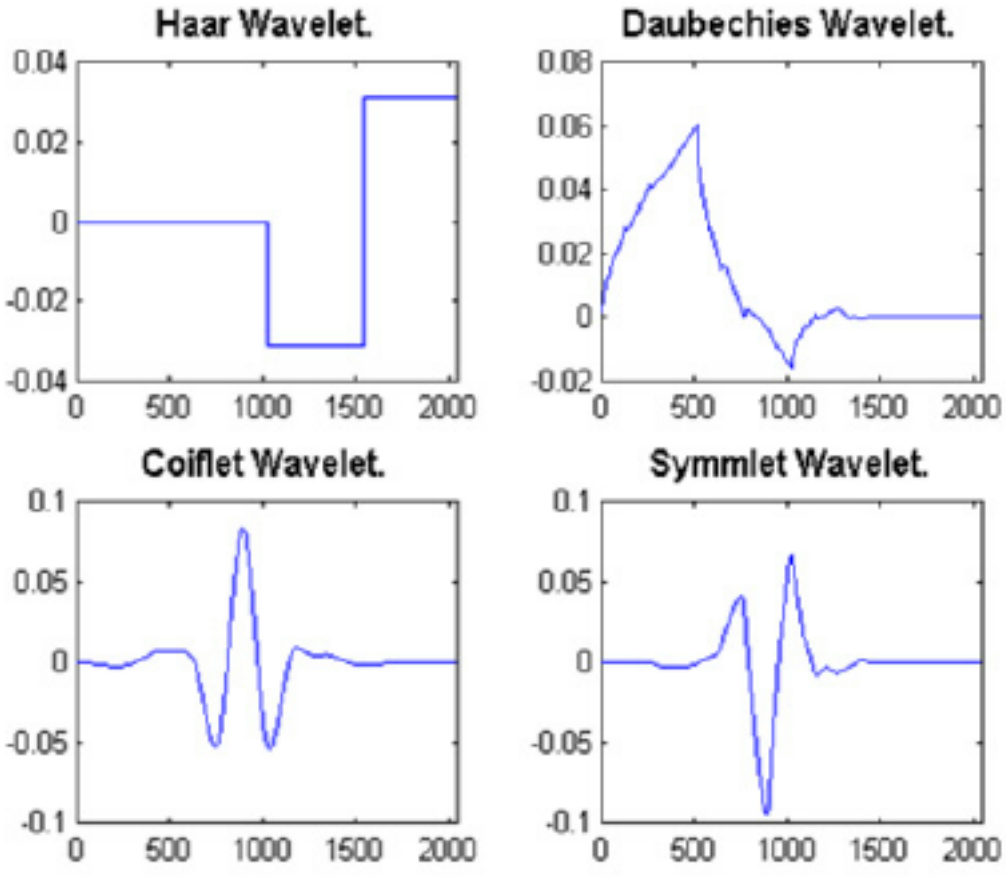


FIGURE 4.1: Several Wavelet families: haar, daubechies, coiflet and symmlet. [167].

4.2.2 Wavelet Transform

The Fourier transform uses a series of sine-waves with different frequencies to analyse a signal; i.e. signal is represented through a linear combination of sine-waves. The Wavelet Transform uses a series of functions called wavelets essentially a small wave each with a different scale.

As shown in Figure 4.2, the main difference between a sine wave and a wavelet is that the sine-wave is not localise in time or space (it stretches out from -infinity to +infinity) while a wavelet is localized in time or space. This allows the wavelet transform to obtain time-information in addition to frequency information.

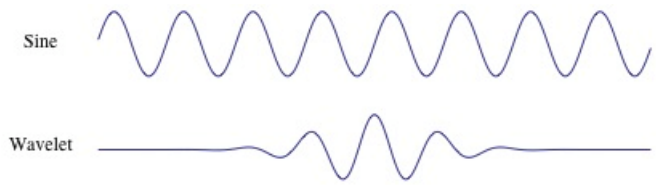


FIGURE 4.2: The difference between a sine-wave and a wavelet. The sine-wave is infinitely long and the wavelet is localised in time [168].

Since the Wavelet is localized in time, we can multiply our signal with the wavelet at different locations in time; This procedure is known as a convolution. This convolution process is repeated for all scaled and translated wavelets in the filter bank [168].

4.3 The Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) [169] is an execution of the wavelet transformation using a discrete set of wavelet scales and translations that comply with certain rules. In other words, this transformation breaks down the signal into a mutually orthogonal set of wavelets. The wavelet is obtained by scaling as

$$\psi(x) = \sum_{k=-\infty}^{\infty} (-1)^k a_{(n-1-k)} \psi(2x-k) \tag{4.1}$$

Where n is an integer, even. The wavelet set forms an orthonormal base for decomposing signals. Note that usually only a few of the coefficients are nonzero, which simplifies the calculations significantly [169]. The discrete Haar wavelet transformation (HWT) is the easiest of all discrete wavelet transformations.

4.4 The 1-D Haar wavelet transform

To demonstrate this process, consider a 1-D image (Figure 4.3) with a resolution of eight pixels, having values [32 0 16 64 128 64 32 16].

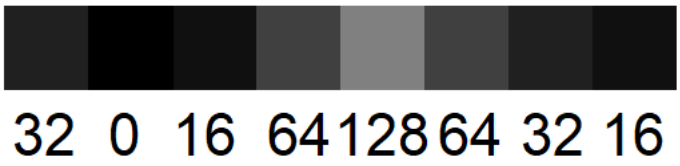


FIGURE 4.3: a 1-D image with a resolution of eight pixels

This image can be represented by calculating a wavelet transform via the Haar Wavelet basis. Start by pairing the pixels to get the lower resolution image. Start by averaging the pixels together, pairwise, to get the lower resolution image.

[16 40 96 24]

In this example, the information is lost by averaging process, in this case the termed detail coefficients which contain information about the original image are needed. The first coefficient is 16 because 32-16=16 and 0+16=16. The second coefficient is -24 because 16-(-24)=40 and 64+(-24)=40. Similarly the third and fourth coefficients can be found to be 32 and 8 respectively. Then the original image has been decomposed into a lower resolution image with detail coefficients as follows in Table 4.1:

TABLE 4.1: 1D Haar wavelet transform

Resolution	**Averages**	**Detail Coefficients**
8	[32 0 16 64 128 64 32 16]	
4	[16 40 96 24]	[16 -24 32 8]
2	[28 60]	[-12 36]
1	[44]	[-16]

Then the wavelet transform (or wavelet decomposition) of the original image is the single coefficient (essentially the pairwise average over all the eight pixels) followed by the detail coefficients:

[44 -16 -12 36 16 -24 32 8]

The process of averaging and differencing recursively is called a filter bank. In this process, no information was obtained or lost: the original image had 8 pixel values and eight coefficients in the transformed image. An image can be rebuilt to any resolution given the details and transformation coefficients.

4.4.1 The 1-D Haar basis functions

The images will be considered as piece by piece at a half-open interval in this section[0,1). It will also use a vector space concept (mainly a vector collection defined

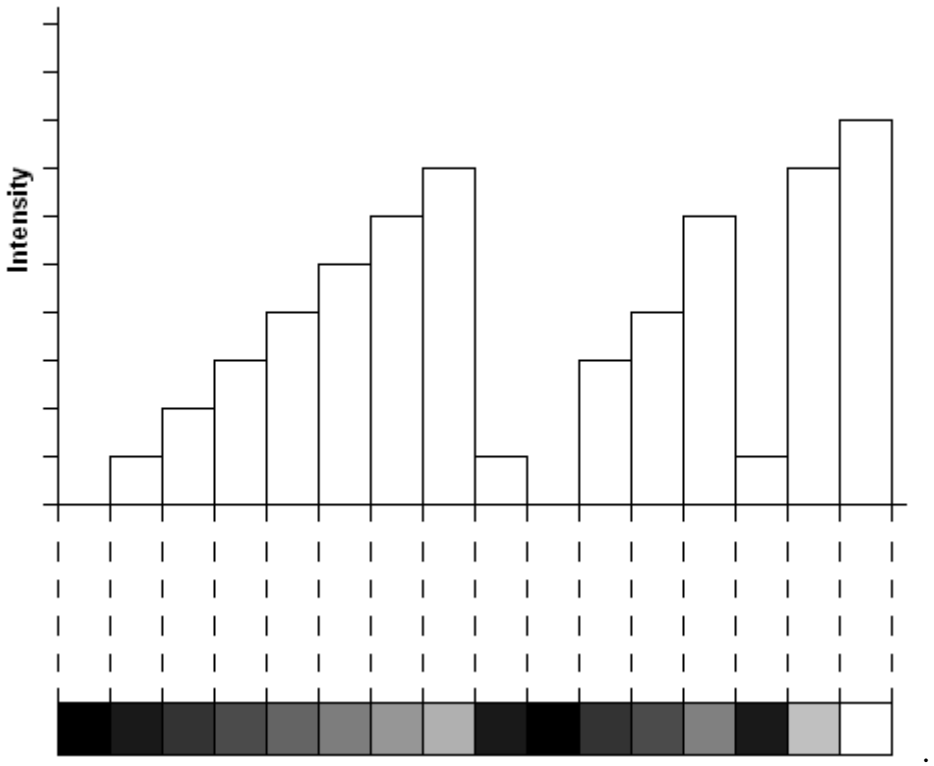


FIGURE 4.4: The intensity profile of a line image.

by addition and scalar multiplication). Thus addition of vectors, scaling and subtraction (addition after scaling by -1) are allowed in this space.

Consider in Figure 4.4 "line-image" (mainly a line of various intensities pixels). Through statistics, the intensities of the pixels can be represented in a graph similar to a histogram. That pixel is represented essentially by a rectangle whose width is determined by the number of pixels in the "line-image," and whose height is determined by a vertical intensity scale. It is this representation that will promote the change from thinking of images to functions as graphical objects.

For example, the one-pixel image can be likened to a function which is constant over the entire interval [0,1). it can be liken each constant function over the interval [0,1) to a vector and It is known as the space of all such vectorsV^0. In a similar way, a two-pixel image is a function having two constant pieces over the intervals [0, .5) and [.5, 1). It is called the space of all vectors of this type V^1.

Continuing in this manner, the vector space V^n includes all piecewise-constant functions defined on the half-open unit interval [0,1) with constant pieces over each of the 2^n equally sized half-open sub-intervals. Since each vector in V^n is a function defined on the unit interval then every vector in V^n is contained in V^{n+1}. The vector spaces are nested subspaces: i.e.

$$V^0 \subset V^1 \subset V^2 \subset ..... \tag{4.2}$$

To define a basis set for the vector space V^n the functions which make up the basis set for V^n scaling functions. The symbol of these functions are ϕ A simple basis for V^n is given by the set of scaled and translated box functions:

$$\phi_m^n(x) := \phi(2^n x - m) m = 0, \ldots, 2^n - 1 \tag{4.3}$$

where

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & otherwise. \end{cases} \tag{4.4}$$

For more Explanation, this the eight box functions forming a basis are shown in Figure 4.5:

4.5.1.1 Support of a function

The support of a function is the region of the domain for which the function is non-zero. Therefore the support of $\phi_1^2(x)$ above would be $[1/4, 1/2)$. Functions which have support over a finite interval are said to have compact support. There for all the box functions forming a basis for V^n have compact support.

4.5.1.2 Function Innerproduct

To define an innerproduct on the vector space to facilitate the construction of an orthogonal (and orthonormal) basis set of functions. For example, the standard integral innerproduct.

$$\langle f|g\rangle := \int_0^1 f(x)g(x)dx \tag{4.5}$$

will suffice. Any two vectors $\vec{u}$ and $\vec{v}$ and are mutually orthogonal if $\langle \vec{u}|\vec{v}\rangle = 0$. Similarly any two box functions in the same space, V^n, will be mutually orthogonal since, the support for each box function is located in a region which no other box function in that space has its support.

$$\langle \phi_m^n | \phi_p^n \rangle = 0 \Leftrightarrow m \neq p \tag{4.6}$$

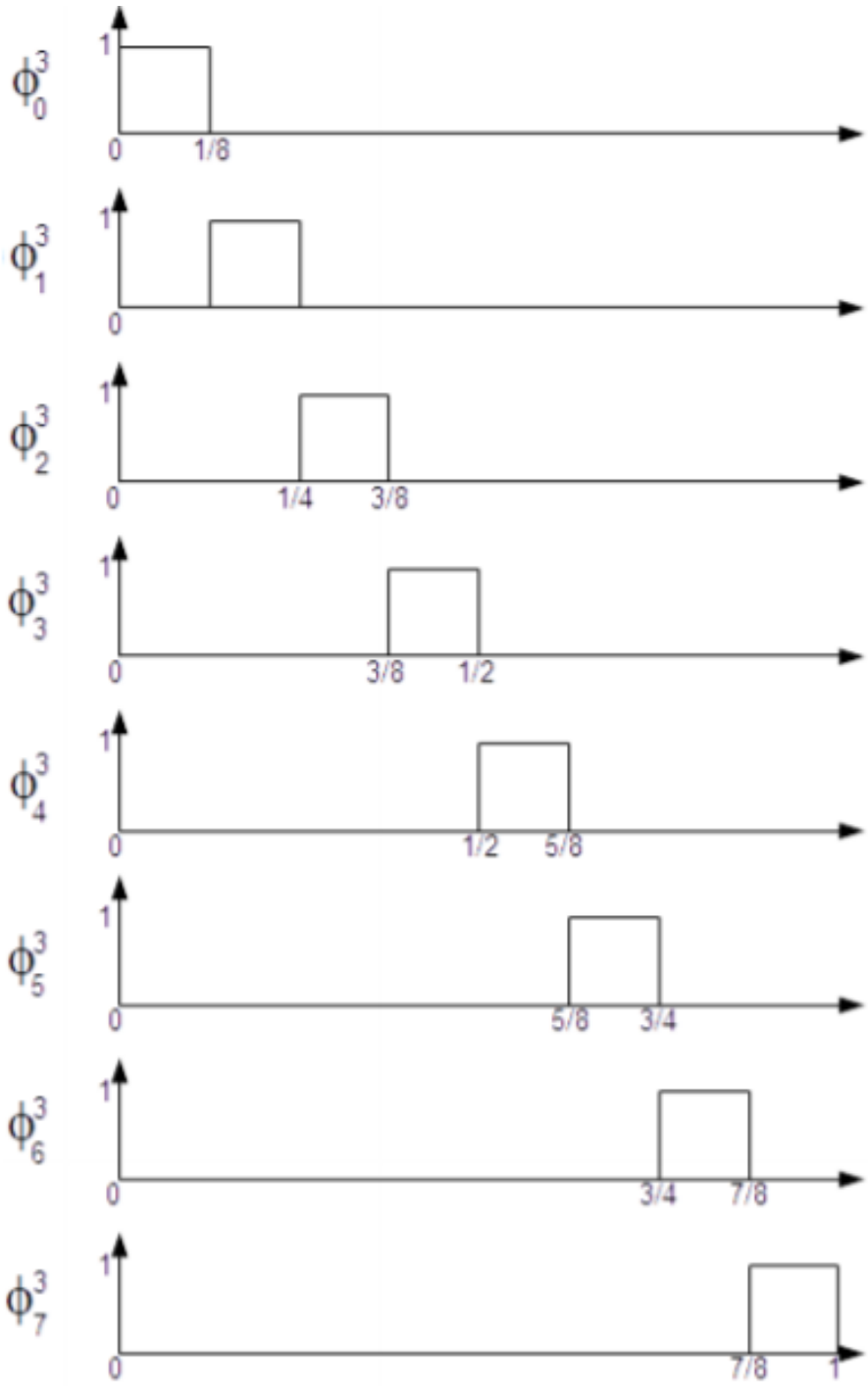


FIGURE 4.5: The box function basis for an eight pixel line image

So, the definition of orthogonal complement W^n to V^n in V^{n+1} where W^n is the space of all function in V^{n+1} which are orthogonal to all functions in V^n under the defined innerproduct on V^{n+1}.

A collection of linearly independent functions spanning Wn are called wavelets and are denoted $\phi_m^n(x)$.

It can be liken the wavelets in W^n as being functions which can represent parts

of a function in V^{n+1} which cannot be represented in V^n. Thus the detail coefficients are essentially the coefficients of the wavelet basis functions.

The wavelets corresponding to the box functions are known as Haar wavelets defined:

$$\psi_m^n(x) := \psi(2^n x - m) m = 0, \ldots, 2^n - 1 \tag{4.7}$$

where

$$\psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} \leq x < 1 \\ 0 & otherwise. \end{cases} \tag{4.8}$$

In Figure 4.6 the four Haar wavelets spanning W^2.

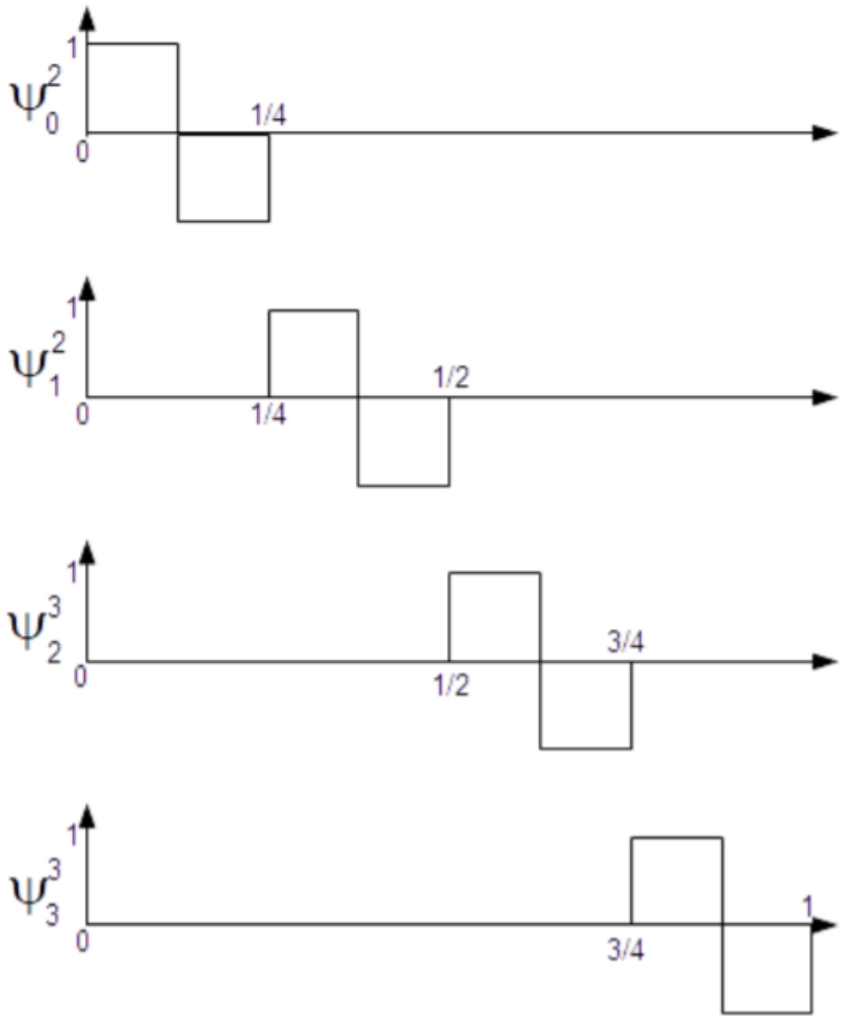


FIGURE 4.6: The four Haar wavelets spanning W^2.

This time the more sophisticated ideas from this section are applied to the problem. Start by expressing the image I(x) as a linear combination of the box basis functions in V^3

$$I(x)=a_0^3\phi_0^3(x)+a_1^3\phi_1^3(x)+a_2^3\phi_2^3(x)+a_3^3\phi_3^3(x)+a_4^3\phi_4^3(x)+a_5^3\phi_5^3(x)+a_6^3\phi_6^3(x)+a_7^3\phi_7^3(x) \tag{4.9}$$

or more graphically in Figure 4.7:

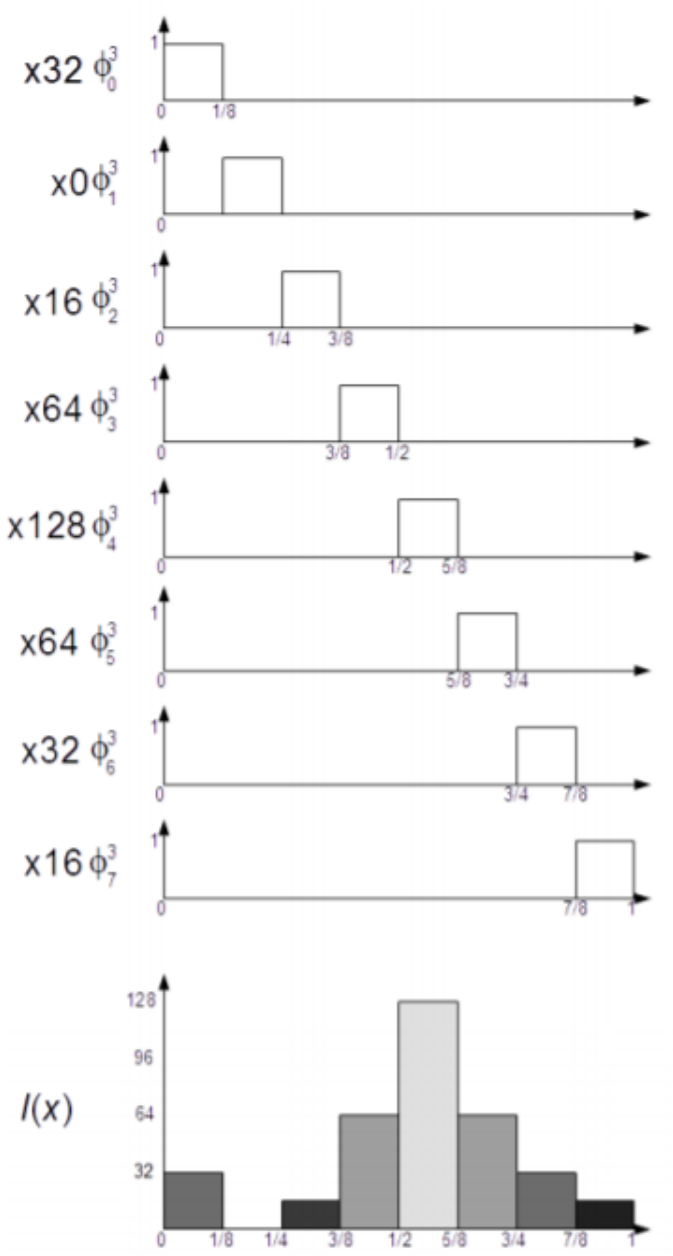


FIGURE 4.7: The 8 pixel image as a linear combination of box functions in V^3 Here the coefficients are just the eight original pixel values [32 0 16 64 128 64 32 16]

By rewriting that in terms of the V^2 and W^2 base functions using the first stage of pairwise averaging and differencing

$$I(x)=a_0^2\phi_0^2(x)+a_1^2\phi_1^2(x)+a_2^2\phi_2^2(x)+a_3^2\phi_3^2(x)+b_0^2\psi_0^2(x)+b_1^2\psi_1^2(x)+b_2^2\psi_2^2(x)+b_3^2\psi_3^2(x) \quad (4.10)$$

and then in terms of the basis functions ofV^1 , W^1 andW^2 ; i.e. the second stage of averaging and differencing

$$I(x)=a_0^1\phi_0^1(x)+a_1^1\phi_1^1(x)+b_0^1\psi_0^1(x)+b_1^1\psi_1^1(x)+b_0^2\psi_0^2(x)+b_1^2\psi_1^2(x)+b_2^2\psi_2^2(x)+b_3^2\psi_3^2(x) \quad (4.11)$$

These are shown for the image in Figure 4.8.

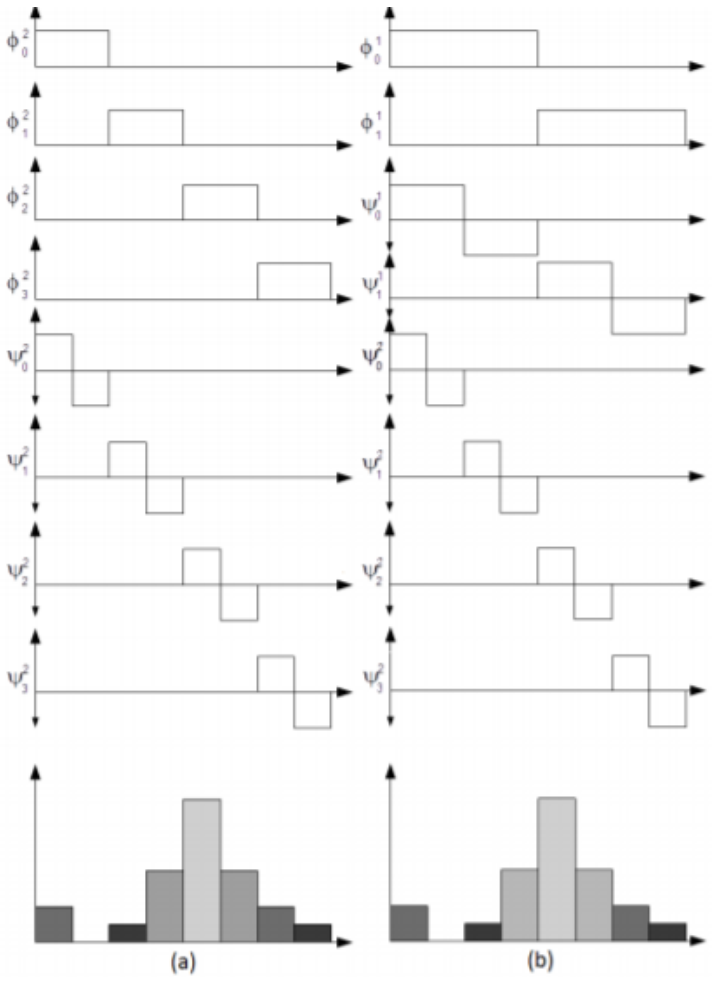


FIGURE 4.8: (a) The first stage averaging and differencing for the 8 pixel image. (b) The second stage averaging and differencing for the 8 pixel image.

Finally re-write the expression in terms of the basis functions for V^0, W^0 and W^1.

$$I(x)=a_0^0\phi_0^0(x)+b_0^0\psi_0^0(x)+b_0^1\psi_0^1(x)+b_1^1\psi_1^1(x)+b_0^2\psi_0^2(x)+b_1^2\psi_1^2(x)+b_2^2\psi_2^2(x)+b_3^2\psi_3^2(x) \tag{4.12}$$

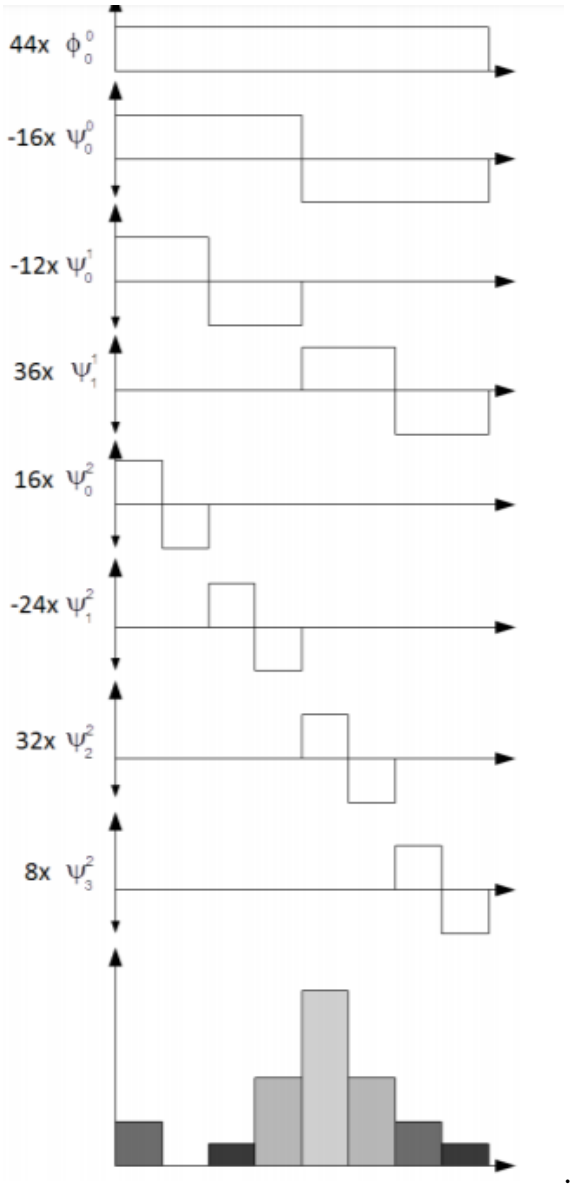


FIGURE 4.9: The final stage decomposition of the 8 pixel image.

The coefficients of the Figure 4.9 for the Haar wavelet transform of the original image are the detail coefficients mentioned previously. The eight functions above constitute the Haar basis for V^3. So, instead of using the usual four box functions, we can use $\psi_0^0, \psi_0^0, \psi_1^0, \psi_1^1, \psi_0^2, \psi_1^2, \psi_2^2, \psi_3^2$ to represent the overall average, broad detail and the two stages of finer detail which a function can have in V^3. For V^n (where

n>3) the Haar basis includes these eight functions plus higher resolution functions which are even narrower versions of the wavelet $\psi(x)$.

Orthogonality and Normalisation

The Haar basis has different property than wavelet bases orthogonality. This means that bases orthogonality have elemental functions stronger condition to each other than wavelets only being orthogonal to all the scaling functions at the same hierarchy n. Normalisation is another important property of some wavelet bases. This property is satisfied by a basis function if

$$\langle u|u\rangle = 1 \tag{4.13}$$

The Haar basis can be normalised by multiplying the original basis by $\sqrt{2^n}$:

$$\begin{aligned} \varphi_m^n(x) &:= \sqrt{2^n}\varphi(2^n x - m) \\ \psi_m^n(x) &:= \sqrt{2^n}\psi(2^n x - m) \end{aligned} \tag{4.14}$$

The $\sqrt{2^n}$ factor is chosen to satisfy the condition uu=1. With this new definition, the new normalised coefficients are obtained by dividing the old coefficients by $\sqrt{2^n}$. Thus, for the running example, the unnormalised coefficients [44 -16 -12 36 16 -24 32 8] become [44-16-12/2 36/2 8-12164] It will become apparent that using a basis that is both orthogonal and normalised, termed orthonormal, is advantageous when compressing an image or function.

Wavelet Compression

The objective is to express th information originally contained in a set of data using a smaller set with or without information loss. The function $f(x)$ which is made up of a linear combination of basis function$u(x)$:

$$f(x) = \sum_{i=1}^{m} a_i u_i(x) \tag{4.15}$$

The unique data for this function consists of the coefficients $a1, ..., am$. To represent $f(x)$ using fewer coefficients by using a different basis (canonical transformations). So given a specified error, ε, we want to find

$$\hat{f}(x) = \sum_{i=1}^{\hat{m}} a_i u_i(x) \tag{4.16}$$

such that

$$\hat{m} < m \quad \text{and} \parallel f(x) - \hat{f}(x) \parallel < \varepsilon \tag{4.17}$$

for some norm defined on the space of $f(x)$. When rebuild a basis, $u_1, \ldots, u_{\hat{m}}$ a few coefficients is provided a good approximation to $f(x)$. This search for a new basis (as done for canonical transformations of any basis: recti-linear to spherical, etc.) is very involved and so approximating $f(x)$ for the basis $f(x)$ was originally expressed with.

Compression means representing a function with as few coefficients as possible as opposed to encoding that refers to storing the information in as few bits as possible. Subtle but important difference.

One method of compression is to order the coefficients, $a_1, ..., a_m$, so that for every $m < \hat{m}$, the first $\hat{m}$ elements of the sequence give the best approximation for $f(x)$ as measured in the L^2 norm. If the basis is orthonormal then the problem is relatively straightforward.

Since the Haar basis is orthonormal the problem admits a solution readily. By letting $\pi(i)$be a permutation of 1, ..., m and $\hat{f}(x)$be a function that uses the coefficients corresponding to the first $\hat{m}$ numbers of the permutation $\pi(i)$:

$$\hat{f}(x) = \sum_{i=1}^{\hat{m}} c_{\pi(i)} u_{\pi(i)} \tag{4.18}$$

The square of the L^2 error in this approximation is given by

$$\begin{aligned} \|f(x) - \hat{f}(x)\|_2^2 &= \langle f(x) - \hat{f}(x) | f(x) - \hat{f}(x) \rangle \\ &= \langle \sum_{i=\hat{m}+1}^{m} c_{\pi(i)} u_{\pi(i)} | \sum_{j=\hat{m}+1}^{m} c_{\pi(j)} u_{\pi(j)} \rangle \\ &= \sum_{i=\hat{m}+1}^{m} \sum_{j=\hat{m}+1}^{m} c_{\pi(i)}\, c_{\pi(j)} \langle u_{\pi(i)} | u_{\pi(j)} \rangle \\ &= \sum_{i=\hat{m}+1}^{m} (c_{\pi(i)})^2 \end{aligned} \tag{4.19}$$

Therefore, the square of the L^2 error is just the sum of the squares of the coefficients. In order to minimise the error for any given the best choice of $\pi(i)$ is the permutation which sorts the coefficients in decreasing magnitude: i.e. $\pi(i)$ satisfies

$$|a_{\pi(1)}| \geq ... \geq |a_{\pi(m)}| \tag{4.20}$$

To apply L^2 compression to the coefficients by ignoring or omitting those which have the smallest magnitude. this is shown in FIG 4.10.

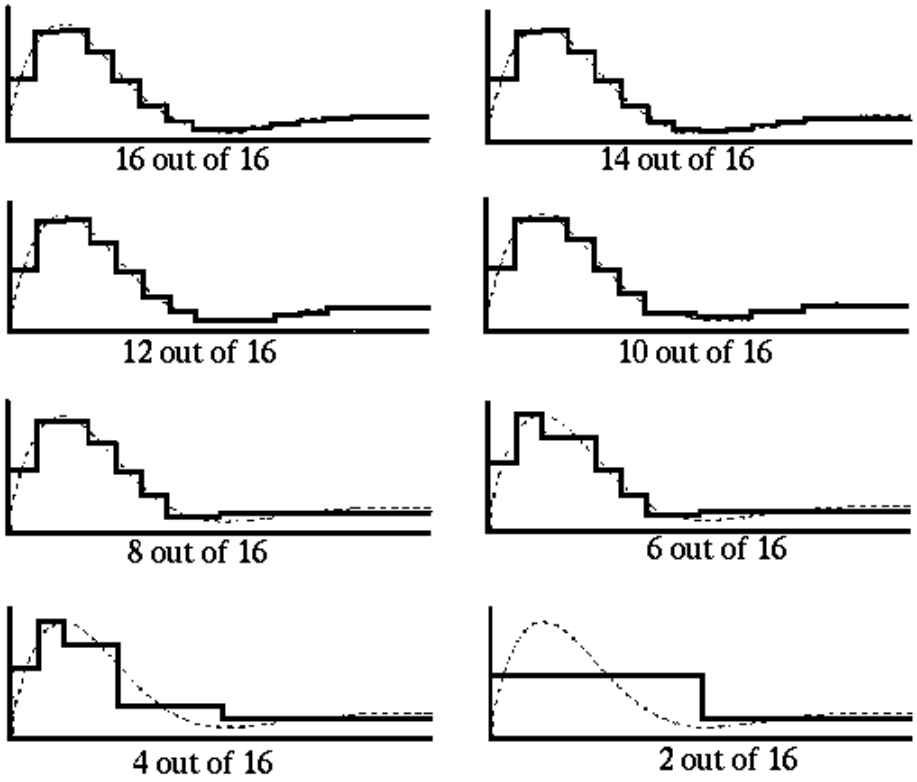


FIGURE 4.10: Various "compressions" of an analog signal (dashed) using haar wavelets.

By varying the amount of compression, a series of approximations which become more coarse as more coefficients are left out. The original 16 coefficients approximates the dashed line curve well whereas the final 2 coefficients image is a crude approximation to the original.

To visualise the situation Figure 4.11 the general Shape of the Haar Wavelet, with respect to the averaging and differencing calculation done. The two pixel's are located in a line image of 2^n pixels. Naturally, each pixel is mapped to an interval of width $(\frac{1}{2})^n$ and the two pixels correspond to the (k-1)-th and k-th pixels.

The two pixels average is located exactly half way between the two levels of intensity (shown in red in Figure 4.11). The region outlined with a blue dashed line is the mathematical function that needs to be added to the average to restore the average of the two original pixel values. Therefore, the blue region corresponds to the detail coefficient and is a Haar wavelet. The average corresponds to a box function of a lower resolution and hence lower V space. In this example the average box function would belong to $V^{(n-1)}$ and the Haar wavelets to $W^{(n-1)}$ with the union of these space being the space V^n.

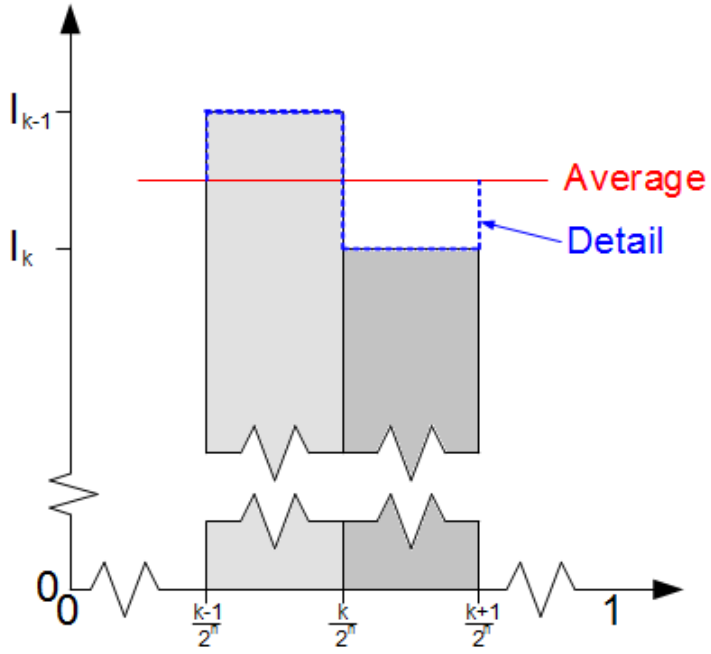


FIGURE 4.11: How the Haar wavelet (in blue) restores pixel values after averaging and differencing is performed.

4.5 Compression of 2D image with Haar Wavelet Technique

Standard decomposition and non-standard decomposition are the two common methods used to transform image pixels using wavelets. The benefits, complexity and overhead computational of each will be considered in turn.

4.5.1 Non-Standard Decomposition

This wavelet transform performs operations alternatively between rows and columns. First, on each pixel value on each row the pairwise averaging and differencing is performed . Then, followed by pairwise averaging and differencing on each column. The process is repeated recursively only on the quadrant containing the averages of both directions.

4.5.2 Standard Decomposition

In standard decomposition of an image, the 1-D wavelet transform is applied to each row of pixel values to obtain the which gives us an average value and difference values for each row of pixels. These transformed rows are then treated as a 2-D image,

and then the columns are transformed using a 1-D wavelet transformation. The result is an image with the same dimensions as the original image which consists of coefficients of difference and a single overall average value for the whole image. In Figures 4.12, 4.13, 4.14, 4,15 and 4.16 shown example of Standard Decomposition

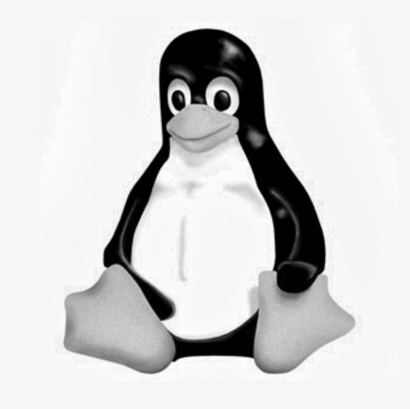


FIGURE 4.12: Original Image

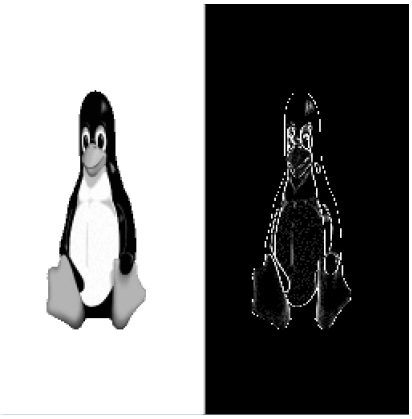


FIGURE 4.13: The first decomposition of an image along the rows

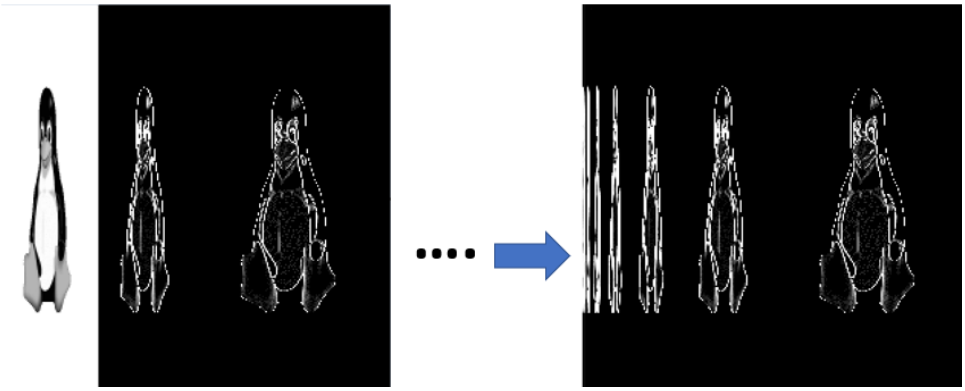


FIGURE 4.14: The decomposition of an image along all rows

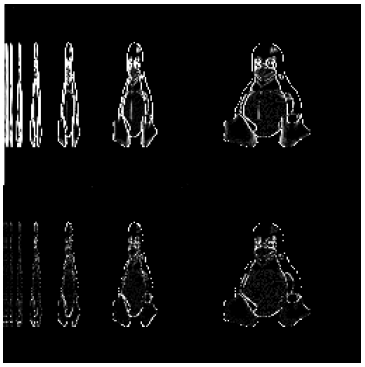


FIGURE 4.15: The first decomposition of an image down the column

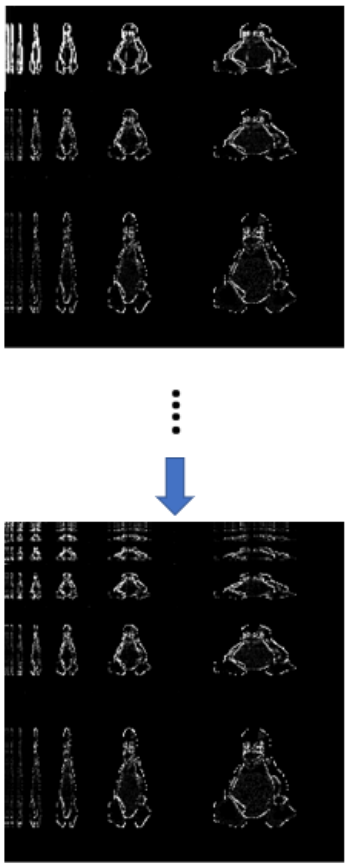


FIGURE 4.16: The decomposition of an image in all columns

The original image can be reconstructed by reverse averaging and differentiation from the transformed image Without Information Loss. Therefore, the picture results in a lossless compression. However, Lossy compression must be considered in order to achieve a greater degree of compression.

4.5.3 Lossy Compression

As mentioned in the Standard Decomposition example, loss of information must be considered when any degree of compression is required.

First a non-negative threshold value ε is chosen. Any detail coefficient in the transformed data with a magnitude below or equal to ε is set to zero. This increases the number of zeros in the transformed matrix and reduces the amount of non-zero data stored. When this ε is used, approximations at the original image created. The setting of the threshold value is very important, as the quality of the compressed image is highly dependent on it. There are various threshold methods, such as hard thresholds, soft thresholds and universal thresholds.Hard thresholds are used here and defined as:

$$T(\varepsilon, x) = \begin{cases} 0 & If|x| < \varepsilon \\ x & otherwise. \end{cases} \tag{4.21}$$

The image compression ratio is calculated by the number of non-zero elements in the original matrix; The threshold is applied to the detail coefficients only to obtain a desired level of compression in the image. This means that all magnitude coefficients below the thresholds are removed. In Figure 4.18 different compression ratios are shown for the image.

FIGURE 4.17: original image

(A) 10%

(B) 70%

(C) 80%

(D) 90%

(E) 95%

(F) 99%

FIGURE 4.18: Different compression ratios for the image in figure 4.17.

4.6 Summary

This chapter recommends a simple but efficient calculation scheme for the compression of images and signals using the Haar wavelet transformation. The proposed work aims to develop computationally efficient and effective compression algorithms using wavelet techniques for loss image compression.

In next chapter 5, the HWT is applied to the input data (MINIST and TI46 datasets) and a supervised neural network based on the back propagation learning algorithm is used to determine optimum compression ratios.

Chapter 5

Simulation Results And Analysis Using simple one hidden Layer ANN

5.1 Overview

In this chapter, the HWT is applied to the MNIST and TI46 Data-sets as a means of reducing the input data to the ANN. The accuracy of the ANN as a function of the HWT compression and the activation function is measured and the optimal configuration of the ANN determined.

In these experiments, a HP Z640 with a Xeon CPU at 2.4 GHz, 16 GB RAM, Ubuntu 15.10 with anaconda was used. The ANN used had one hidden layer and utilised back-propagation (BP).

5.1.1 ANN overview

ANN models have been shown to be suitable for data processing will be considered by comparing the efficiency and accuracy of systems using the sigmoid and softplus functions applied to the two databases; MNIST and T146.

The simple one hidden Layer ANN configuration used for each set of features had the following in both activation function sigmoid and softplus applied to the two data-sets in Table 5.1.

5.2 Experiment using the MNIST Dataset

This experiment is split into two parts. The first part is to apply the HWT to the input images. The second part is to feed the image to the ANN using two types of different activation functions sigmoid and softplus, measure the accuracy of the ANN as a function of compression and determine the optimal configuration of the ANN.

TABLE 5.1: The simple one hidden Layer ANN configuration used for each set of features.

	MNIST		**T146**	
	sigmoid	softplus	sigmoid	softplus
Hidden nodes	1500	1500	4500	4500
learning rate	0.001	0.001	0.01	0.01
momentum	0.001	0.001	0.001	0.001
batch size	40	40	50	50
initial weights	-1.0 : 1.0	-0.1 : 0.1	-1.0 : 1.0	-0.1 : 0.1
Regulisation	0.0	0.01	0.0	0.01
epochs	250	450	400	600

When the HWT is applied, optimal computational requirements are determined by applying different compression thresholds to the HWT detail coefficients while monitoring the image quality. The resulting reconstructed images showed that image quality and significant image details can be maintained while achieving high compression rates.

The Proposed algorithm used in these experiments is illustrated in Figure 5.1.

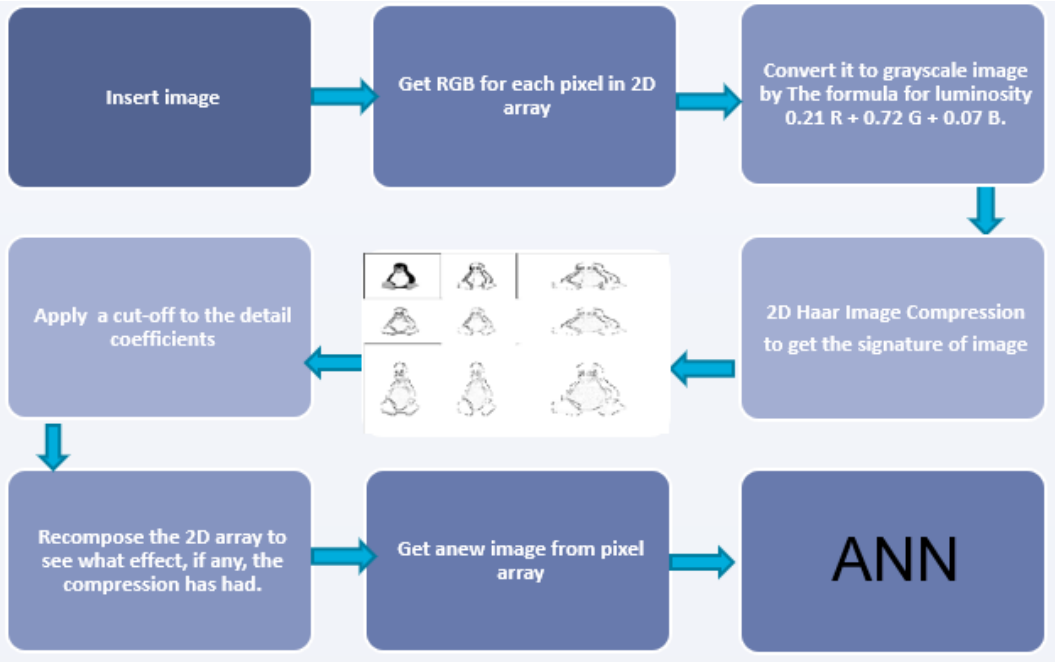


FIGURE 5.1: The algorithm for HWT for MNIST data-set before feed to the ANN .

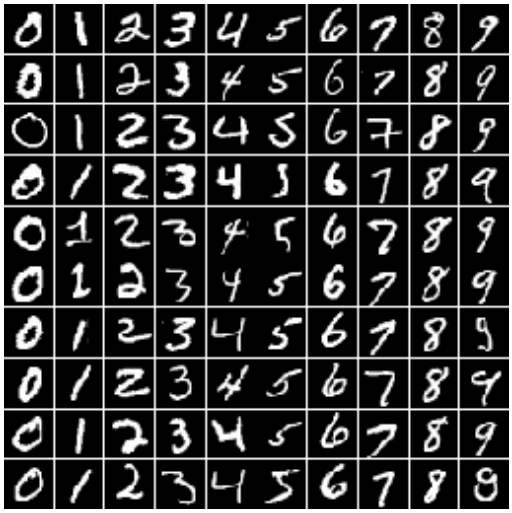
These experiments were conducted on the MNIST data-set. All digits are normalised, centred and re-sized from a 28x28 to a 32x32 image. The training set was divided in 10000 patterns for training (1000 images per digit). The testing set was divided into 1000 patterns for training (100 images per digit). The Haar compression was applied to the data-set, with seven different compression ratios (50%, 60%, 70%, 80%, 85%, 90% and 95%) as shown in Figure 5.2.

5.2.1 ANN configuration

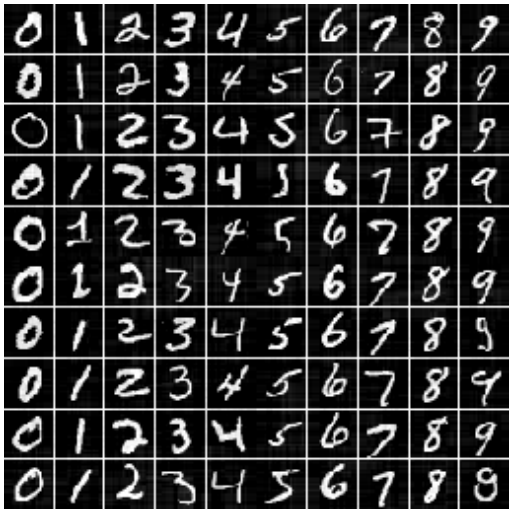
To select the configuration for each feature such as hidden layer nodes, learning rate and epochs was used to ensure achieving the best results where a test of variety of ANN configurations to generate the one that gives the best predictions for the data. By running the same model on the same training set MNIST data-sets and the best configuration for the other features but with different configurations for the feature each time and selecting the network with the best performance.

As mention in table 5.1, the ANN configuration used for each set of features had the following values after testing to get the high accuracy for the ANN in both activation function sigmoid and softplus:

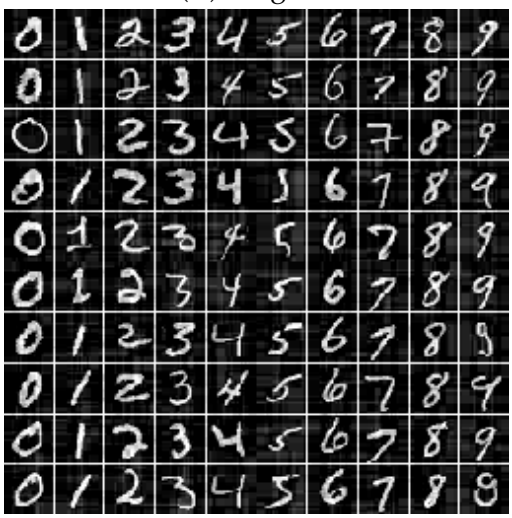
- Momentum rate: It was fixed at 0.001. The accuracy didn't change appreciably when this rate was remain unchanged by multiples of 0.00001.
- The number of hidden nodes was set to 1500 after testing at 100, 1000, 1500, 1700 and 2000. As shown in Figure 5.3
- The batch size was set to 40 after testing it 10, 15, 20, 35, 40 and 100 . This shown in Figure 5.4.
- The learning rate was selected to be 0.001 after testing it at 0.00001, 0.0001, 0.001, 0.01 and 0.1). As shown in Figure 5.5.



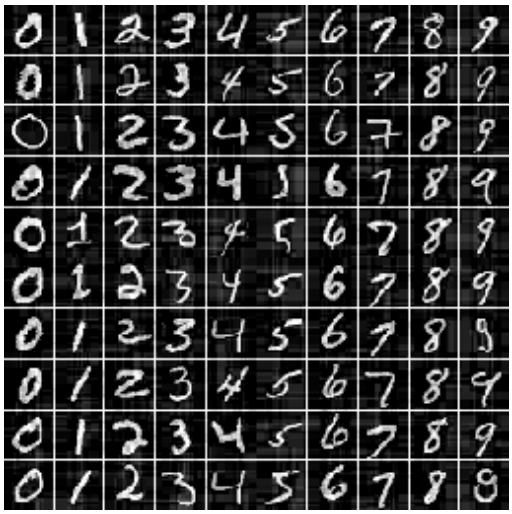
(A) Original



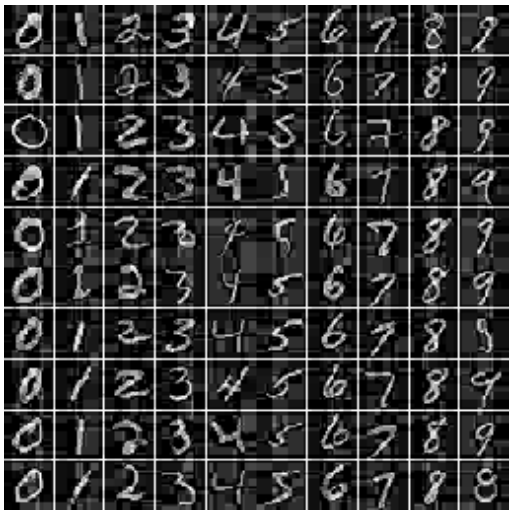
(B) 70%



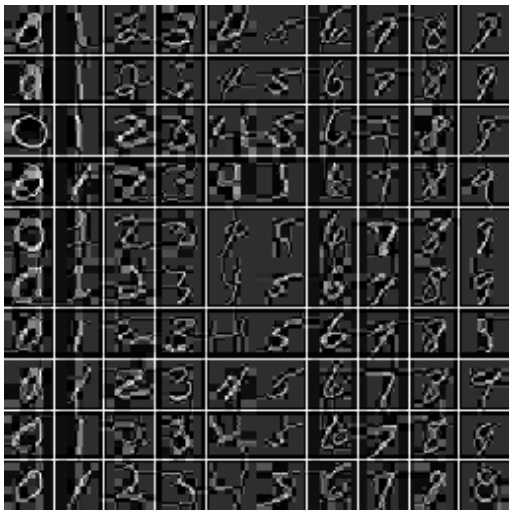
(C) 80%



(D) 85%



(E) 90%



(F) 95%

FIGURE 5.2: Examples of decomposes image of MNIST database after compressed using HWT showing how the image be blurry in high compression ratios.

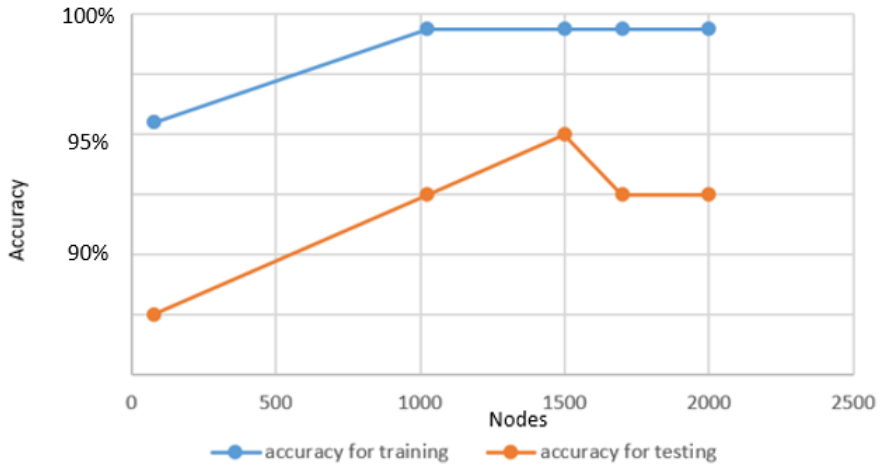


FIGURE 5.3: Testing for different numbers of nodes in the hidden layer.

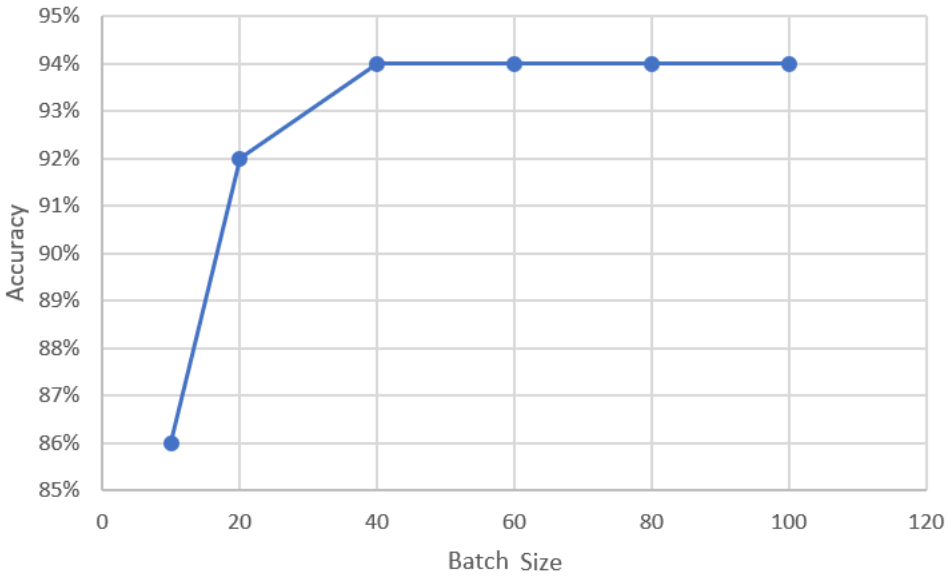


FIGURE 5.4: Accuracy vs batch size for chosen ANN.

The following experiments were conducted without HWT compression on the input. The ANN, using the sigmoid activation function, the initialised weights on nodes were normalised between -1.0 and 1.0. The regularisation was set at 0. The accuracy was plotted against epochs to ascertain the optimum accuracy in Figure 5.6, It can be seen to reach a max of 99.5% for 250 epochs.

For the ANN, using softplus activation function, the initialised weights on nodes were normalised between -0.1 and 0.1. The regularisation was set at 0.01; any-other value caused under-fitting or over-fitting in the network. For this softplus ANN, the accuracy was plotted vs epochs in Figure 5.7. The optimum accuracy of 99% was achieved at 450 epochs.

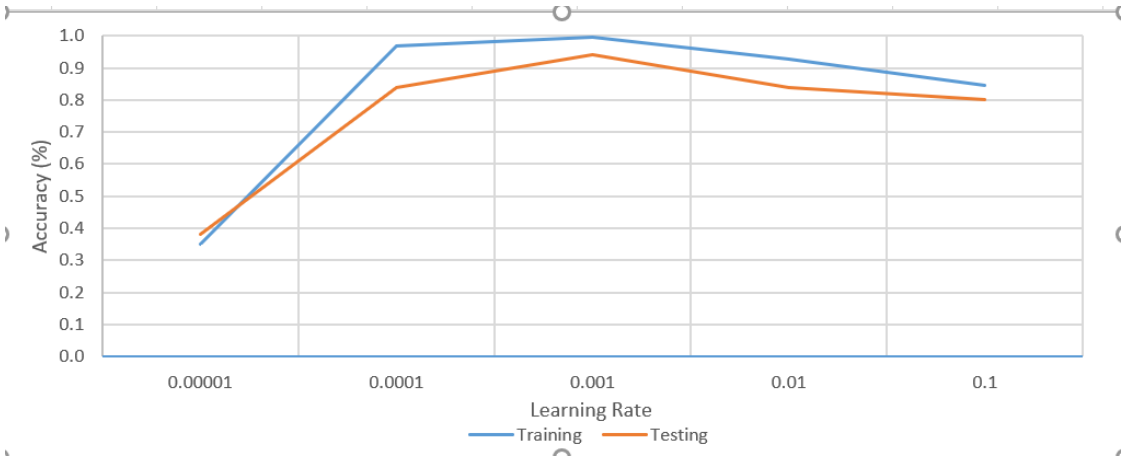


FIGURE 5.5: Accuracy vs learning rate for chosen ANN.

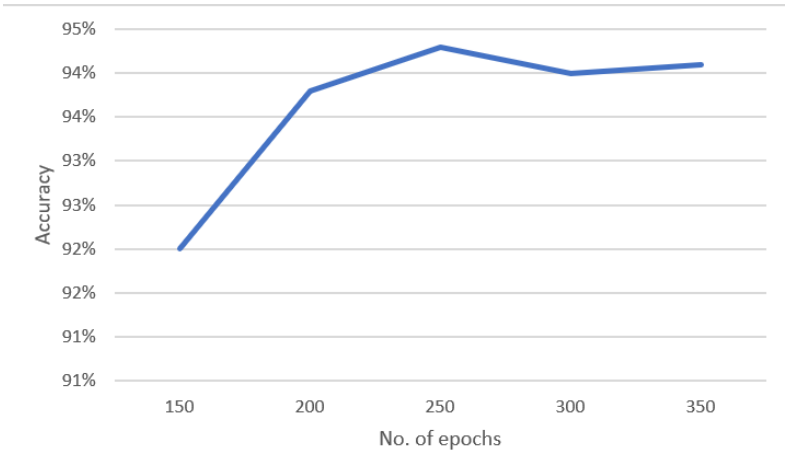


FIGURE 5.6: Accuracy vs no. of epochs for the ANN using the sigmoid activation function.

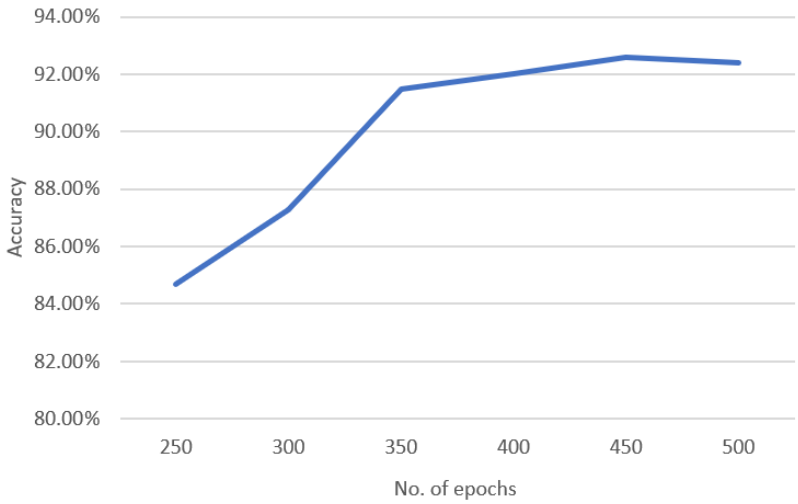


FIGURE 5.7: Accuracy vs no. of epochs for the ANN using the softplus activation function.

Results

The ANN performance on the test data with HWT is shown in Table 5.2. For the compression ratio was set at 50%, 60%, 70% and 80% the recognition rate accuracy for tests using sigmoid was 94.3% and softplus tests yielded 92.6%. However, the recognition rate decreased significantly for compression ratios greater than 85% This was due to the relatively small size (32x32) of the MNIST images.

Figure 5.8 gives a confusion matrix about the highest result details about MNIST data-set with softplus and sigmoid activation function.

TABLE 5.2: The Accuracy for MNIST test set

The image signature	Accuracy using sigmoid	Accuracy using softplus
Compression ratio 0%	94.3%	92.6%
Compression ratio 50%	94.3%	92.6%
Compression ratio 60%	94.3%	92.6%
Compression ratio 70%	94.3%	92.6%
Compression ratio 80%	94.3%	92.6%
Compression ratio 85%	91%	88.9%
Compression ratio 90%	79.9%	76%
Compression ratio 95%	68.2%	64.2%

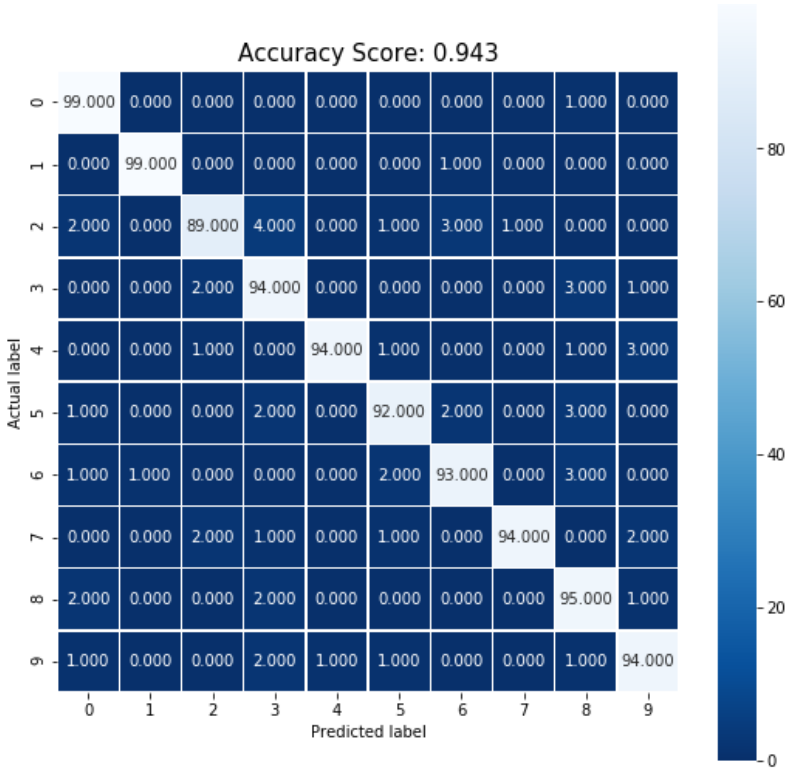
5.2.2 Experiment using the TI46 Data-sets

As with MNIST, this experiment is split into two parts. The first part is the application of HWT to the audio signal. The second part is to feed the signal into an ANN using either sigmoid or softplus. The accuracy of the ANN as a function of HWT compression is measured and the optimal configuration of the ANN determined.

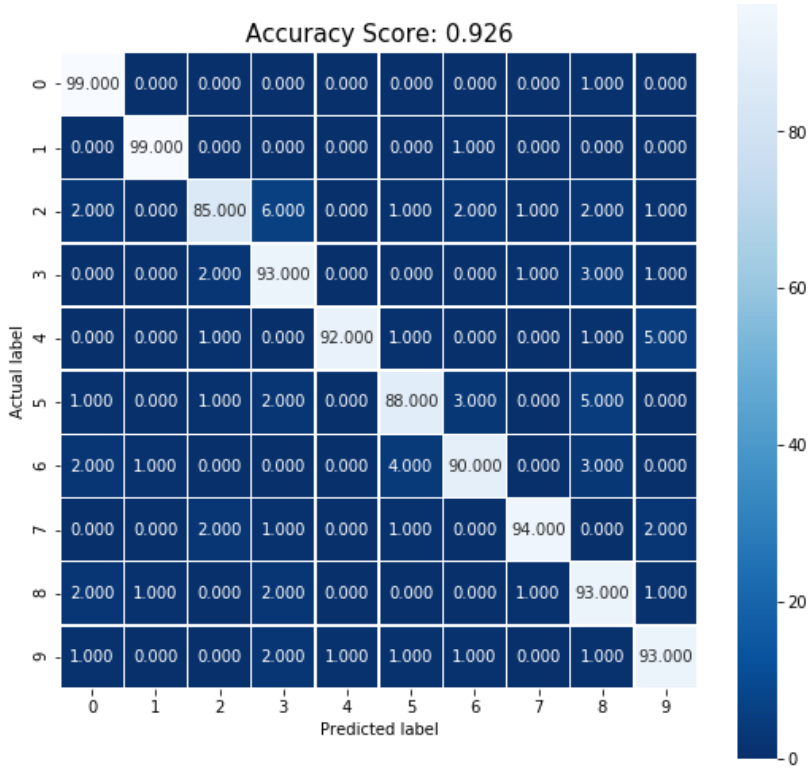
The optimum computational requirements are determined by applying different compression thresholds to the HWT coefficients while checking the audio quality. The resulting reconstructed audio showed that audio quality can be maintained while achieving high compression rates with reduced computational overheads.

The algorithm for TI46 (non end-pointed set and end-pointed set) is

1. Read the Audio from the user.
2. Convert the .mfc file to excel file using Matlab.
3. Reshape the 2-D Matrix to 1-D.



(A) The testing result using sigmoid activation function.



(B) The testing result using softplus activation function.

FIGURE 5.8: Confusion matrix for the highest testing result using MNIST dataset.

4. get the size for each audio, and resize all of them by if the number is less than 4096 add 0 until become 4096. on other hand if the number is more than 4096 cut the rest of them.
5. Apply 1-D DWT using Haar wavelets over the audio.
6. For the computation of Haar wavelet transform, set the threshold Compression Ratios value i.e., the percentage threshold for serving detail coefficients.
7. Reconstruct an estimate of the original audio by applying the corresponding inverse transform.
8. Display the resulting audio and comment on the quality of the audio.
9. Compare different Compression Ratios for corresponding reconstructed audio.
10. The same process is repeated for different signal and the performance determined.

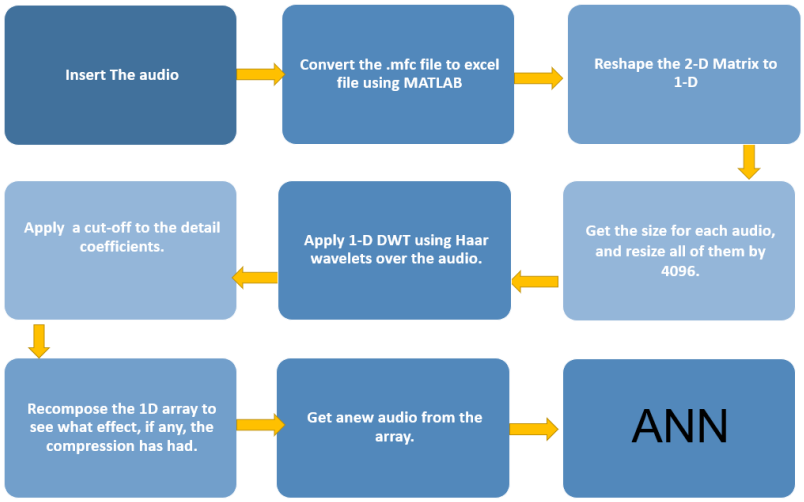


FIGURE 5.9: The algorithm for HWT for TI46 Data-sets before feed to the ANN.

5.2.3 The ANN configuration

To select the configuration for each feature such as hidden layer nodes, learning rate and epochs was used to ensure achieving the best results where a test of variety of ANN configurations to generate the one that gives the best predictions for the data. By running the same model on the same training set TI46 data-sets and the best configuration for the other features but with different configurations for the feature each time and selecting the network with the best performance.

The ANN configuration used for each feature for TI46 data-sets for the ANN in both activation function sigmoid and softplus, as mention before in Table 5.1, was:

- momentum rate 0.001.
- decrease constant 0.00001.
- Hidden layer nodes 4500. The Hidden layer nodes value of 4500 was selected after testing at value of 1500, 2000, 3000, 4000, 4500, 4700 and 5000. The plot in Figure 5.10 shown accuracy vs hidden layer nodes for this ANN.

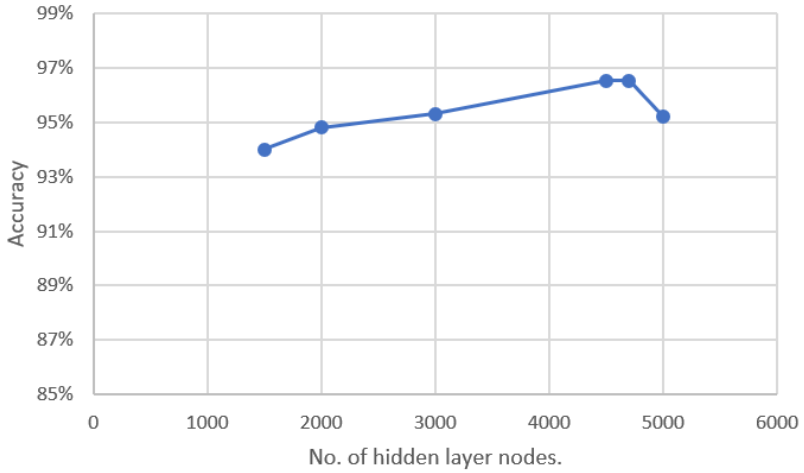


FIGURE 5.10: Accuracy vs hidden layer nodes for this ANN using TI46 non end-pointed data-set.

- The optimum learning rate was found to be 0.01, as shown Figure 5.11.

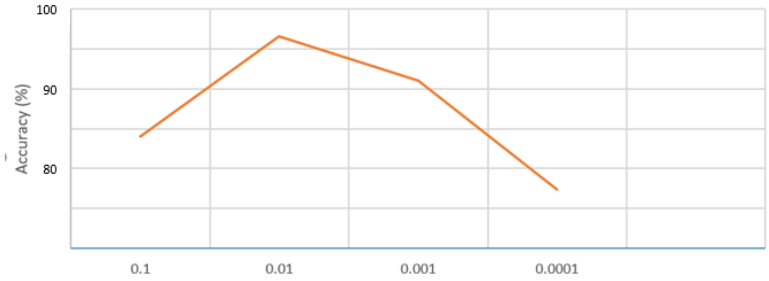


FIGURE 5.11: Accuracy vs learning rate for this ANN using TI46 non end-pointed data-set.

For the ANN using the sigmoid activation function, the initialised weights on nodes were normalised between -1.0 and 1.0 with the regularisation set to 0. As shown in Figure 5.12, the accuracy reached a maximum of 99.8% for 400 epochs.

The ANN using the softplus activation function had initialised weights on nodes were normalised between -0.1 and 0.1 and the regularisation was set to 0.01. A

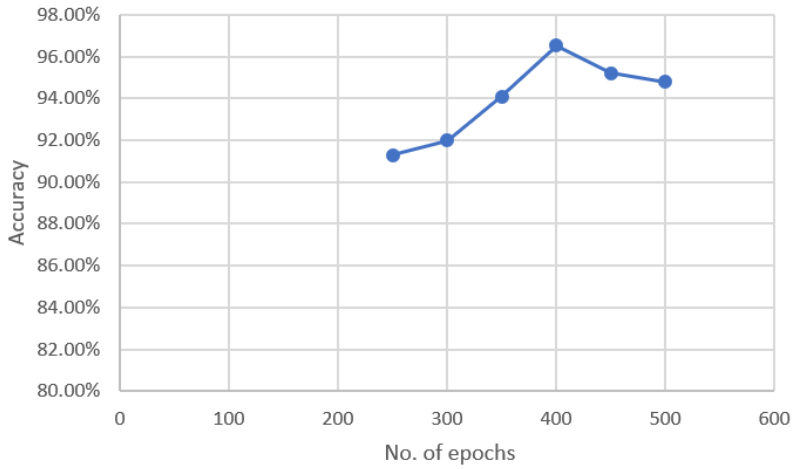


FIGURE 5.12: The accuracy vs epochs for the ANN using sigmoid activation function on the using TI46 non end-pointed data-set.

maximum of 600 iterations (epochs) was imposed because the accuracy reached a maximum of 99.5% for this number of epochs as shown in Figure 5.13.

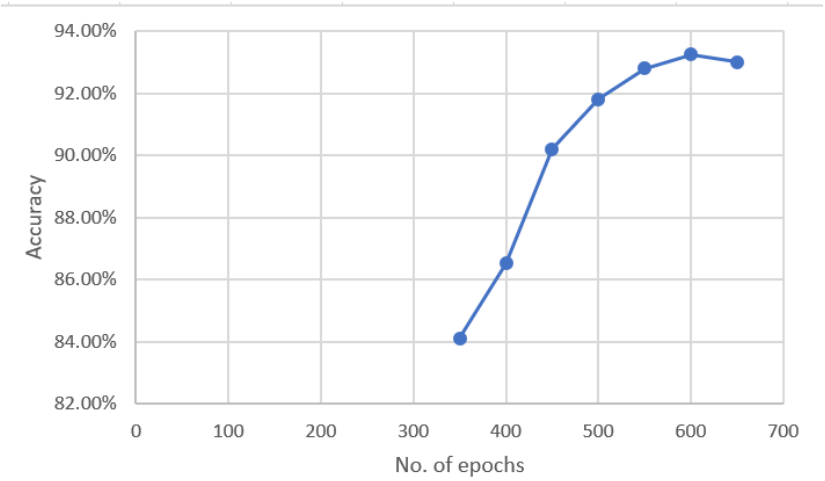


FIGURE 5.13: The accuracy vs epochs for the ANN using softplus activation function on the using TI46 non end-pointed data-set.

All signals are normalised and resized to 4096. The testing set was divided into 1500 patterns for training (150 audio per digit). As for the MNIST experiment, seven compression ratios (50%, 60%, 70%, 80%,82%, 85%, 90% and 95%) were applied.

5.2.4 Results

The performance of the ANN using sigmoid activation function on the TI46 non-end point data is shown in table 5.3. A 96.52% recognition rate was achieved when

TABLE 5.3: The Accuracy for TI46 non end-point test set.

The signature	**Accuracy using sigmoid**	**Accuracy using softplus**
Compression ratio 0%	96.52%	93.24%
Compression ratio 50%	96.52%	93.24%
Compression ratio 60%	96.52%	93.24%
Compression ratio 70%	96.52%	93.24%
Compression ratio 80%	96.52%	93.24%
Compression ratio 82%	96.52%	93.24%
Compression ratio 85%	92.08%	86.32%
Compression ratio 90%	87.56%	71.46%
Compression ratio 95%	79.32%	65.23%

the compression ratio was 82% or less. However, for compression ratios greater than 85% the recognition rate fell significantly. The tests using softplus yielded 93.24% recognition rate when the compression ratio was set 82% or less. However, for compression ratios greater than 85% the recognition rate fell significantly.

The performance of the ANN function on the TI46 end point data is shown in table 5.4. For using sigmoid activation function 98.8% recognition rate was achieved when the compression ratio was 82% or less. For compression ratios greater than 85%, the recognition rate fell significantly. On the other hand, tests using softplus yielded 97.32% recognition rate when the compression ratio was set 82% or less. Again, however, for compression ratios greater than 85%, the recognition rate fell significantly.

In

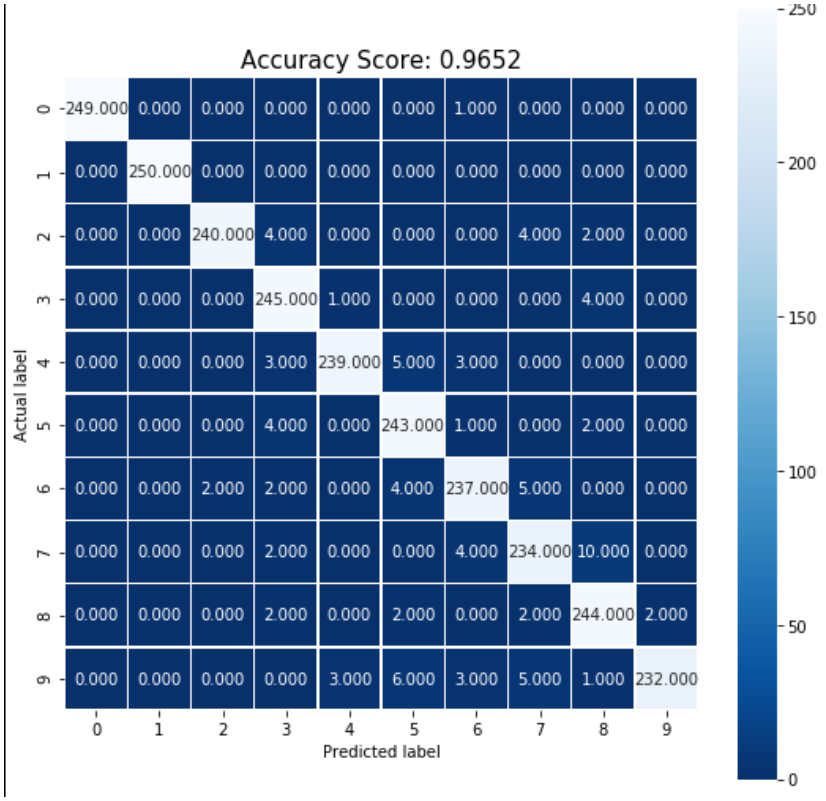
TABLE 5.4: The Accuracy for TI46 end-point test set.

The TI46 signature	Accuracy using sigmoid	Accuracy using softplus
Compression ratio 0%	98.80%	97.32%
Compression ratio 50%	98.80%	97.32%
Compression ratio 60%	98.80%	97.32%
Compression ratio 70%	98.80%	97.32%
Compression ratio 80%	98.80%	97.32%
Compression ratio 82%	98.80%	97.32%
Compression ratio 85%	98.80%	96.24%
Compression ratio 90%	94.76%	90.57%
Compression ratio 95%	92.62%	88.34%

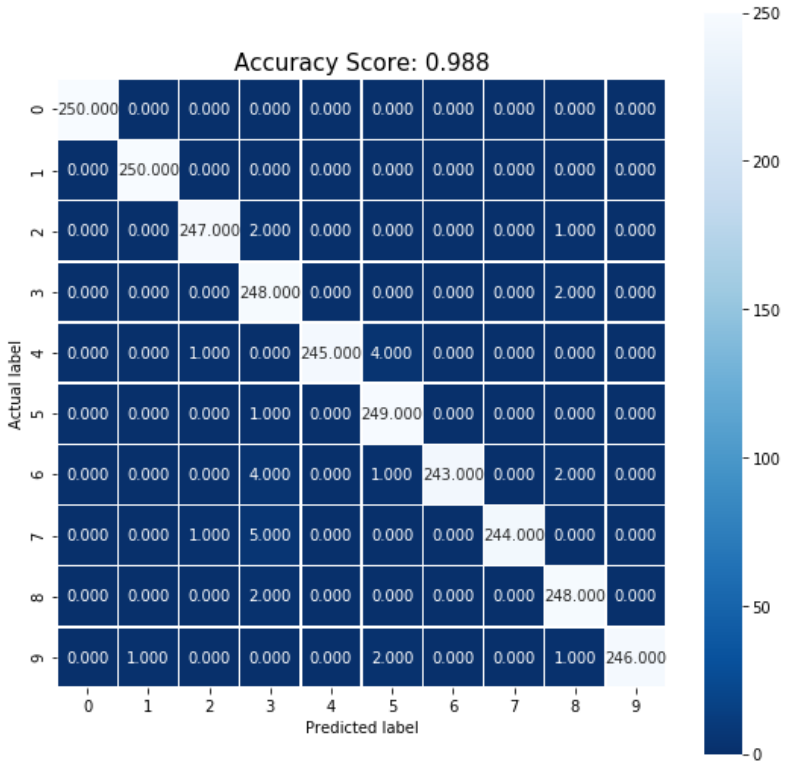
5.3 Summary

In this chapter, The HWT was applied to the input data (MNIST Data-set and TI46 Data-set) and an ANN used to ascertain optimum compression rates. By, keeping the quality of the data considerably same even after cut off details coefficients from the data. It can be conducted from the observed values of the experimental study, that the HWT approach when applied 80% to MNIST Data-set compression and 82% to TI46 Data-set compression can provide better results than any other high percentage of data compression. In addition to this ANN using sigmoid activation function is better than ANN using softplus activation function with higher accuracy and less of epochs.

In the next chapter, both data-sets will be used in a supervised ANN from the keras library to determine optimum compression. Comparison between the ANNs in this chapter and the Keras library are made and conduction drawn.



(A) The testing result using sigmoid activation function TI46 non end-point test set.



(B) The testing result using sigmoid activation function TI46 end-point test set.

FIGURE 5.14: Confusion matrix for the TI46 data-sets with the best result.

Chapter 6

Simulation Results And Analysis Using Keras

In this section, the Keras python library is used in order to add additional hidden layers to the ANN and to analyse the impact these addition layers have on the results with the two activation functions, sigmoid and softplus. The experiments with keras use the same number of hidden nodes and epochs that were used in Chapter 5 to allow a comparative analysis.

6.1 Overview of Keras

Keras [172] is a modular Python-based neural network library that can run on either TensorFlow or Theano. Keras is one of the leading high-level neural network APIs. It supports multiple back-end neural network computation engines. The library was designed to allow users to implement deep learning models easily ; enabling quick progress from concept to deployment. Furthermore, Keras [173] can run on both a CPU and a GPU, making it very efficient computationally.

In short, Keras provides the benefits of large acceptance, support for a wide spectrum of manufacturing deployment, integration with at least five backend motors (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and powerful support for various GPUs and distributed training.

A nice design feature of Keras is the availability of neural layers, cost functions, optimizers, initialisation schemes, activation functions and regularisation schemes as standalone modules that can be readily combined to create new models. Thus, new modules are simple to add either as new classes and/or functions.

6.1.1 The parameters of Keras

The user first defines a model with Keras [172], which can be selected from a sequential model or a graph model. In a sequential model the layers are stacked and a

layer output feeds the next layer input until the output layer is reached. The graph model allows users to take the output from a desired layer and feed the output to a desired layer, allowing multiple output networks to be generated or receiving the output in the model's intermediate layer.

The following parameters are used in the implementation of keras [173]:

• **The optimizer**
This parameter determines the model's learning and convergence. In Keras there are many predefined optimizers, such as stochastic gradient descend (SGD), Adam, RMSprop, and Adagrad. Each optimizer has its own parameters, but the learning rate is common to all of them. This parameter defines how much the nodal weights are changed after each epoch; i.e, the weight change will be higher for a high learning rate than for a small learning rate. Another important parameter is weight decline; an additional term in the weight update rule that causes weights to decline to zero exponentially if no other update is planned.

• **Loss**
The loss parameter, defines the objective of the training that the model has to optimize. There are many different objectives defined, for example: mean square error (MSE), mean squared logarithmic error (MSLE), categorical crossentropy that computes the logarithmic difference of the output with all the classes, and many more

•**Metrics**
After each training step this parameter allows the user to see the model's accuracy.

• **Dropout**
Dropout is a technique of regularisation used to reduce deep neural network overfitting. In addition, as the resulting network becomes smaller it speeds up the learning process. Dropout means that during the learning process neurons are literally dropped (temporarily). Input and hidden neurons, along with their incoming and outgoing connections, are removed from the network architecture. These neurons are removed randomly.

With the model compiled, the training can be started. The Keras function to train a model is called "fit" and has a lot of parameters that can be modified as well, which are listed in Table 6.1

TABLE 6.1: The Definition/ Purpose for the parameters for our model.

Parameter	Definition/ Purpose
X	A Numpy training data array.
Y	A target data Numpy array.
Batch size	number of samples per update gradient.
No. of epochs	number of times over training data to iterate.
Callbacks	This parameter allows the user to save the weights of the network after each epoch if the loss is lower than any previous value.
Validation Data	is a sub-set validating the performance of the model.
Shuffle	enables the user to shuffle the training data after each epoch automatically.

6.1.2 Keras parameters used

The selected parameters for this experiment are as follows:

• The Model: The sequential model with dense layers, using the add method. It is a linear stack of layers, and the layers can be described very simply.

• Optimizer: Adam (ADAptive Moment estimation) was the optimizer used because of the relatively low memory requirements.

• Loss: The objective decided for this task has to be the "categorical crossentropy" since it is the most accurate parameter for performing a classification with multiple classes. Furthermore, in the MNIST dataset problem (where you have images of the numbers 0,1, 2, 3, 4, 5, 6, 7, 8, and 9) categorical crossentropy gives the probability that an image of a number is, for example, a 4 or a 9. Mathematically, this function calculates in Equation 6.1.

$$H(p,q) = xp(x)log(q(x)) \tag{6.1}$$

where $H(p,q)$ is the cross-entropy between p and q with p, and q define two distributions of probability.

• Metrics: The accuracy metric computes the accuracy rate across all predictions.

•The ANN configuration using keras library setup as Table 6.2

TABLE 6.2: The ANN using keras library configuration used for each set of features.

	MNIST		**T146**	
	sigmoid	softplus	sigmoid	softplus
Hidden nodes	1500	1500	4500	4500
learning rate	0.001	0.001	0.01	0.01
batch size	40	40	50	50
Drop out(uniform)	0.0 : 1.0	0.0 : 1.0	0.0 : 1.0	0.0 : 1.0
epochs	250	250	450	450

6.2 Experiment using the MNIST Data-set

These experiments were conducted on the MNIST database. The training set was divided was 10,000 patterns for training (1,000 images per digit). The testing set was divided into 1000 patterns for training (100 images per digit). Haar compression was applied to the data-set, with seven different compression ratios (50%, 60%,70%, 80%, 85%, 90% and 95%).

The training parameters for ANN using keras were chosen to be the same with ANN Using simple one hidden Layer to compare the performance for each one :

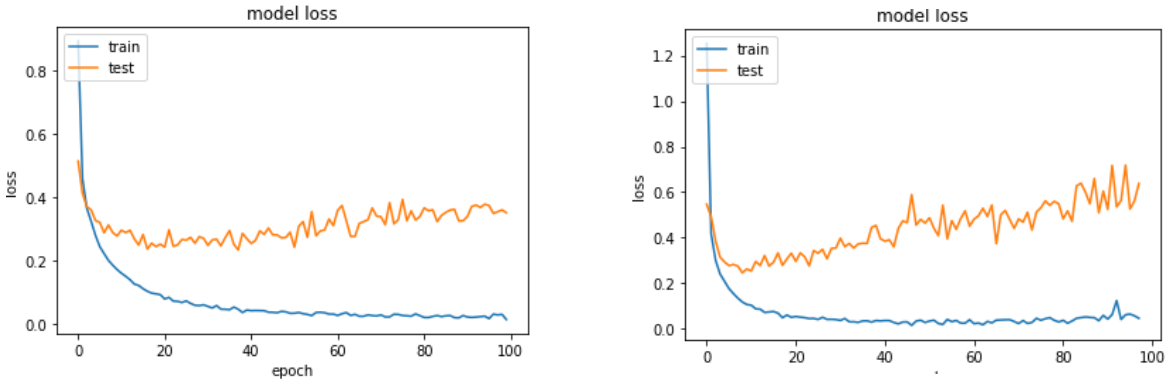
- •The batch size was set to 40.
- •The number of epochs selected was 250.
- •The ANN consisting of 1, 2, 3 , 4 and 5 hidden layers are considered in these experiments with 1500 nodes in each layer.
- • For tuning the dropout, it was selected uniform between (0 and 1) .

6.2.1 Result using the MNIST Data-set

The performance scores as a function of the number of hidden layers can be seen in Table 6.2 and Table 6.3, There is an improvement after the addition of the second hidden layer. The addition of more hidden layers, however, doesn't lead to further improvements. In some cases, the results are somewhat worse. As mentioned [174], increasing the number of hidden layers much more than the sufficient number of layers will cause accuracy in the test set to decrease . It will cause ANN to overfit to the training set. Therefore, it will learn the training data, but it won't be able to generalise to new unseen data.

For example in Figure 6.1, in (A) this model showing that the loss in test is decreasing witch mean more accuracy result. on other hand, (B) the overfitting starts

to occur earlier for the model having four hidden layers (having more capacity). This overfitting point can be seen as when the validation cost stops decreasing and starts to increase.



(A) The two hidden layers ANN.

(B) The four hidden layers ANN.

FIGURE 6.1: the two models of ANN using softplus showing the different loss in both training and testing .

The recognition rate for tests using sigmoid was 95% when the compression ratio was set at 50%, 60%, 70% and 80% using two hidden layers. However, the recognition rate decreased significantly for compression ratios greater than 85% or by adding more hidden layers. On the other hand, the softplus tests yielded a recognition rate of 95.2%, the confusion matrix for this result shown in Figure 6.2, when the compression ratio was set at 50%, 60%, 70% and 80% while using two hidden layers. The recognition rate decreased significantly for compression ratios greater than 85% or by adding more hidden layers. As mentioned [175], a problem with sigmoid functions is that they relatively quickly saturate in its limit values. on other hand, the function softplus does not suffer from vanishing gradient problem, and this is what may happened in the ANN. Because, sigmoid maps the actual number line to a "small" [0, 1] range, particularly with a very flat function on most number line. As a consequence, wide areas of the input field are mapped to a very small range.

TABLE 6.3: Accuracy as function of the number of hidden layers for various applied compression ratios using the sigmoid activation function by using the MNIST Data-set.

The image signature	1 Layer	2 Layers	3 Layers	4 Layers	5 Layers
Compression ratio 0%	94.8%	95%	94.5%	93.6%	91%
Compression ratio 50%	94.8%	95%	94.5%	93.6%	91%
Compression ratio 60%	94.8%	95%	94.5%	93.6%	91%
Compression ratio 70%	94.8%	95%	94.5%	93.6%	91%
Compression ratio 80%	94.8%	95%	94.5%	93.6%	91%
Compression ratio 85%	92.7%	93.3%	92.6%	89%	83.9%
Compression ratio 90%	84.1%	84.4%	81.5%	76.8%	71.2%
Compression ratio 95%	64.4%	67.9%	61.7%	53.2%	50%

TABLE 6.4: Accuracy as function of the number of hidden layers for various applied compression ratios using the softplus activation function by using the MNIST Data-set.

The image signature	1 Layer	2 Layers	3 Layers	4 Layers	5 Layers
Compression ratio 0%	95%	95.2%	93.8%	93%	Nan
Compression ratio 50%	95%	95.2%	93.8%	93%	Nan
Compression ratio 60%	95%	95.2%	93.8%	93%	Nan
Compression ratio 70%	95%	95.2%	93.8%	93%	Nan
Compression ratio 80%	95%	95.2%	93.8%	93%	Nan
Compression ratio 85%	93.1%	93.4%	91.6%	89%	Nan
Compression ratio 90%	85.7%	86.6%	82.1%	78.9%	Nan
Compression ratio 95%	67.9%	69.9%	63.9%	56%	Nan

6.2.2 Experiment using the TI46 Data-set

These experiments were conducted on the TI46 Data-set. There are two endpointed and non-endpointed data-sets. The training set was divided in 1500 patterns for training (150 audios per digit). The testing set was divided into 1570 patterns for

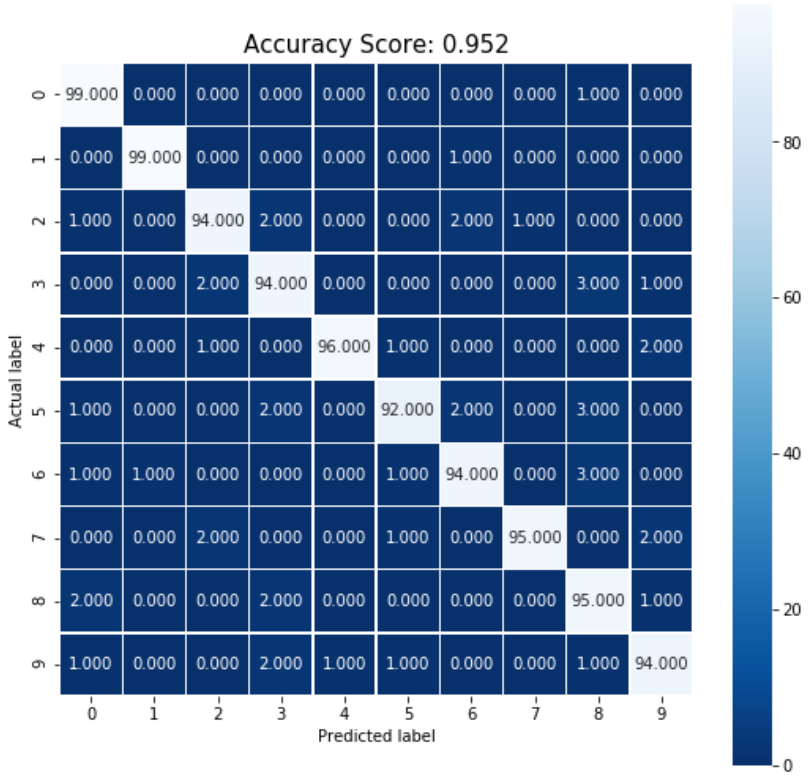


FIGURE 6.2: Confusion matrix for the highest testing result for MNIST data-set using softplus activation function with keras library.

training (157 audios per digit). The Haar compression has been applied to the data-set, with seven different compression ratios (50%, 60%,70%, 80%, 85%, 90% and 95%).

The training parameters for ANN using keras were chosen to be the same with ANN Using simple one hidden Layer to compare the performance for each one:

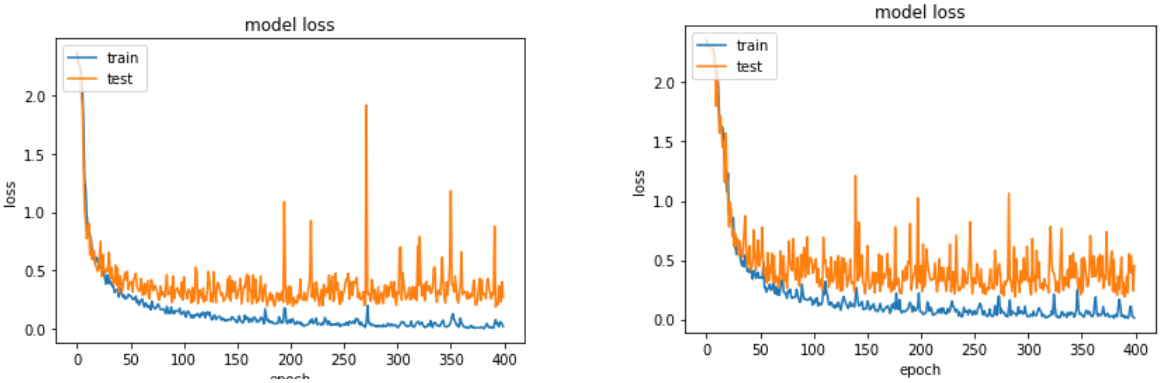
- The batch size was set to 50 as in some tests the training accelerated and even improved the network accuracy by this batch size.
- The number of epochs selected was 450.
- The ANN consisting of 1, 2 and 3 hidden layers are considered in these experiments with 4500 nodes in each layer.
- To select dropout tuning it was selected uniform to the between (0 and 1) .

The result for testing TI46 data-sets

The performance scores as a function of the number of hidden layers are presented in Table 6.5 Table 6.6, Table 6.7 and Table 6.8. There is no discernible improvement after adding the second hidden layer, whereas, the addition of hidden layers, led to decrease accuracy. As mentioned [174], increasing the number of hidden layers much more than the number of layers would result in decreasing accuracy in the

test set. ANN was overly fitting for the training set. This means that it learns the training data, but can not generalise to new invisible data.

For example in Figure 6.3, this model shows in (A) that the loss in the test is that the outcome the average is less than 0.5 means more accuracy. On the other hand, (B) for a model with two hidden layers (having more capacity) starts earlier. point can be seen as when the cost of validation the average between 0.5 and 1.0.



(A) The one hidden layer ANN.

(B) The two hidden layers ANN.

FIGURE 6.3: the two models of ANN using sigmoid showing the different loss in both training and testing .

The recognition rate for tests using sigmoid was 98.68% with TI46 endpoint and 94.44% with TI46 non endpoint when the compression ratio was set at 50%, 60%, 70%, 80% and 82% while using one hidden layer. However, the recognition rate decreased significantly for compression ratios greater than 85% or adding more hidden layers.

The softplus tests yielded a recognition rate of 98.56%, the confusion matrix for this result shown in Figure 6.4, with TI46 endpoint and 95% with TI46 non endpoint when the compression ratio was set at 50%, 60%, 70%, 80% and 82% one hidden layer. However, the recognition rate decreased significantly for compression ratios greater than 85% or adding more hidden layers.

Because, as mentioned [175] Problem with sigmoid functions is that relatively quickly saturate in its limit values. In other hand the function softplus does not suffer from vanishing gradient problem.

TABLE 6.5: Accuracy as function of the number of hidden layer for various applied compression ratios using the sigmoid activation function for TI46 end-point data-set.

The TI46 end-point signature	1 Layer	2 Layers	3 Layers
Compression ratio 0%	98.68%	98.22%	Nan
Compression ratio 50%	98.68%	98.22%	Nan
Compression ratio 60%	98.68%	98.22%	Nan
Compression ratio 70%	98.68%	98.22%	Nan
Compression ratio 80%	98.68%	98.22%	Nan
Compression ratio 82%	98.68%	98.22%	Nan
Compression ratio 85%	98.48%	97.90%	Nan
Compression ratio 90%	94.64%	92.92%	Nan
Compression ratio 95%	93.16%	91.52%	Nan

TABLE 6.6: Accuracy as function of the number of hidden layer for various applied compression ratios using the softplus activation function for TI46 end-point data-set.

The TI46 end-point signature	1 Layer	2 Layers	3 Layers
Compression ratio 0%	98.56%	98.09%	Nan
Compression ratio 50%	98.56%	98.09%	Nan
Compression ratio 60%	98.56%	98.09%	Nan
Compression ratio 70%	98.56%	98.09%	Nan
Compression ratio 80%	98.56%	98.09%	Nan
Compression ratio 82%	98.56%	98.09%	Nan
Compression ratio 85%	98.48%	97.64%	Nan
Compression ratio 90%	95.08%	93.44%	Nan
Compression ratio 95%	94.52%	92.53%	Nan

TABLE 6.7: Accuracy as function of the number of hidden layer for various applied Compression ratios using the sigmoid activation function for TI46 non endpoint data-set.

The TI46 signature	1 Layer	2 Layers	3 Layers
Compression ratio 0%	94.32%	94.44%	Nan
Compression ratio 50%	94.32%	94.44%	Nan
Compression ratio 60%	94.32%	94.44%	Nan
Compression ratio 70%	94.32%	94.44%	Nan
Compression ratio 80%	94.32%	94.44%	Nan
Compression ratio 82%	94.32%	94.44%	Nan
Compression ratio 85%	93.11%	93.3%	Nan
Compression ratio 90%	88.24%	85.84%	Nan
Compression ratio 95%	73%	72.4%	Nan

TABLE 6.8: Accuracy as function of the number of hidden layer for various applied Compression ratios using the softplus activation function for TI46 non endpoint data-set.

The TI46 signature	1 Layer	2 Layers	3 Layers
Compression ratio 0%	95%	94.48%	Nan
Compression ratio 50%	95%	94.48%	Nan
Compression ratio 60%	95%	94.48%	Nan
Compression ratio 70%	95%	94.48%	Nan
Compression ratio 80%	95%	94.48%	Nan
Compression ratio 82%	95%	94.48%	Nan
Compression ratio 85%	93.2%	93.6%	Nan
Compression ratio 90%	90.4%	87.67%	Nan
Compression ratio 95%	77.12%	72.4%	Nan

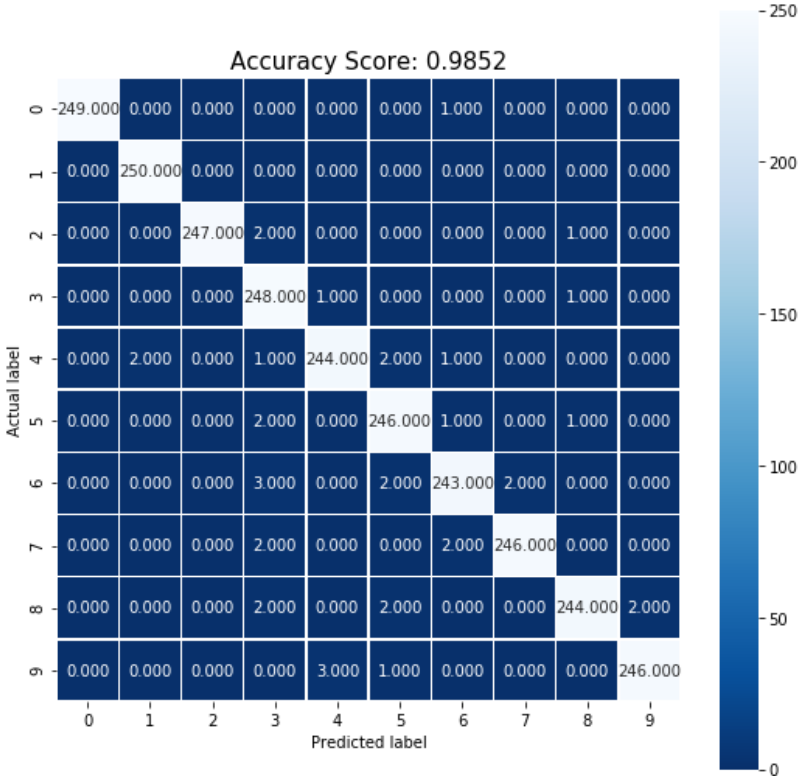


FIGURE 6.4: Confusion matrix for the highest testing result for TI46 non endpoint data-set using softplus activation function with keras library.

6.3 Discussion

In this section presents a discussion of the experimental results obtained from the testing of the proposed ANN systems the simple one layer model and keras model.

In our experiments the first part, the compression is obtained in MNIST and TI46 data-sets by wavelet coefficient threshold using different threshold percentages. All coefficients below defined some threshold are neglected and the compression ratio is computed. Compression algorithm operation is compression ratio is fixed to the required level and threshold value has been changed to achieve required compression ratio; It is noted that a larger number of compression ratio causes the loss efficiency. Since, lossy compression causes loss of information, that may no longer be visible on the archived the data, for example image, after decomposition, making it impossible to view the same image that originally saw. Therefore, decomposition is required to achieve balance between the quality and computational complexity.

The second part, the decomposition MNIST and TI46 data-sets were tested using ANN systems the simple one layer model and keras model with two different activation functions sigmoid and softplus.

6.3.1 MNIST Data-set

This section compere two methods for image compression using an ANN. Implementation of the proposed method uses Haar image compression where, due to the nature of loss wavelet compression, the quality of the compressed images degrades at higher compression ratios. The objective of an optimal ratio is to combine a high compression ratio with compressed image of good quality.

In this work, a maximum accuracy level of 95.2% for a cut-off of (50%, 60%, 70% and 80%) using keras library with a two hidden layer ANN using the softplus activation function was considered an acceptable trade-off between accuracy and cut-off ratio. However, for compression ratios greater than 85% the recognition rate fell significantly. The successful implementation of the proposed method using ANNs with the HWT is evident from the high recognition rates shown in the Figure 6.5 and Figure 6.6.

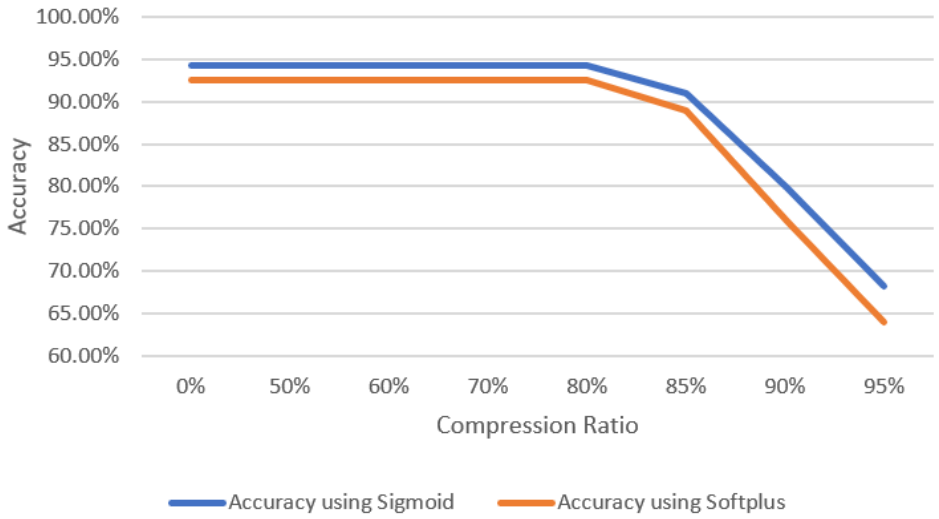


FIGURE 6.5: MNIST Data results for different compression rate using simple one hidden Layer Model with softplus and sigmoid activation function.

6.3.2 TI46 Data-set endpoint and non-endpoint Data-sets

This section proposes an audio compression method that uses ANN. For these simulations,The TI46 Data-set endpoint and non-endpoint Data-sets were used. All is converted for all data to a 1D array and resized to 4096. The training set was split into 1500 training patterns (150 audios per digit). The test set was split into 1570 training patterns (157 audios per digit).

FIGURE 6.6: MNIST Data result for different compression rate using Keras Model with softplus and sigmoid activation function.

Execution of the proposed method uses Haar audio compression where the quality of the compressed audio degrades at higher compression ratios due to the nature of loss wavelet compression. The main objective of an optimal ratio is to combine a high compression ratio with decent quality compressed audio.

TI46 end-point Data-set

On other hand by using TI46 end-point Data-set, an appropriate trade-off between accuracy and cut-off ratio was considered to be a maximum accuracy level of 98.8% for a cut-off of (50%, 60%, 70%, 80% and 82%) using a single hidden layer ANN with sigmoid activation function was considered an acceptable trade-off between accuracy and cut-off ratio. However, the recognition rate fell significantly for compression ratios greater than 85%. The successful implementation of the proposed method using ANN's with the HWT is evident from the high recognition rates shown in the Figure 6.7 and Figure 6.8.

TI46 non-endpoint Dataset

In this work, a maximum accuracy level of 96.52% for a cut-off of (50%, 60%, 70% and 82%) using a simple single hidden Layer Model with sigmoid activation function was considered an acceptable trade-off between accuracy and cut-off ratio. However, for compression ratios greater than 85% the recognition rate fell significantly. The successful implementation of the proposed method using ANN's with the HWT is evident from the high recognition rates shown in the Figure 6.9 and Figure 6.10.

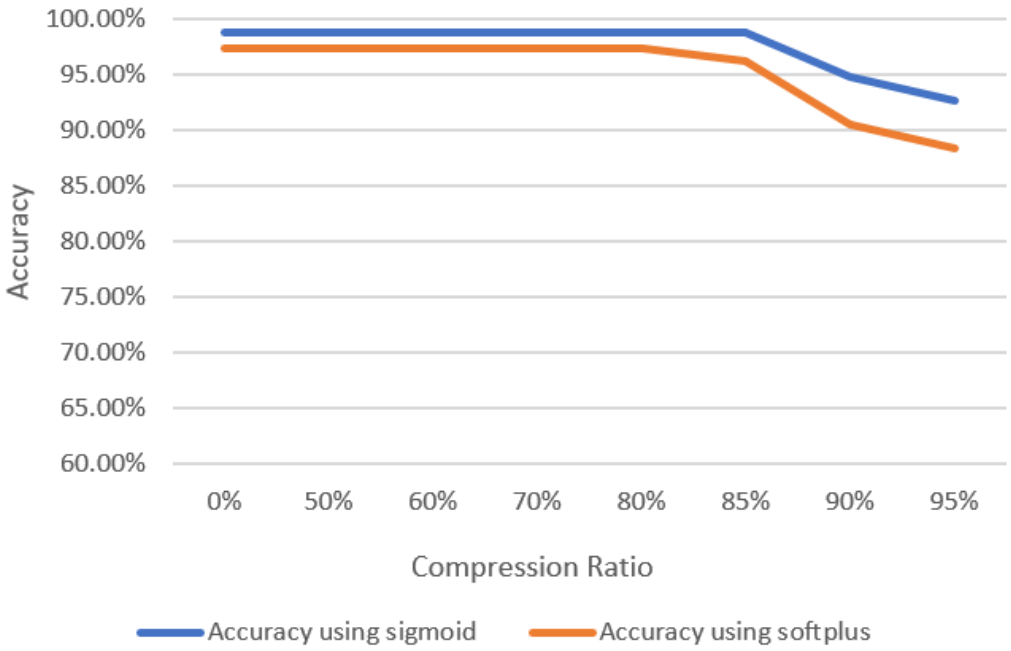


FIGURE 6.7: TI46 Data-set endpoint Data results for different compression rate using simple one hidden Layer Model with softplus and sigmoid activation function.

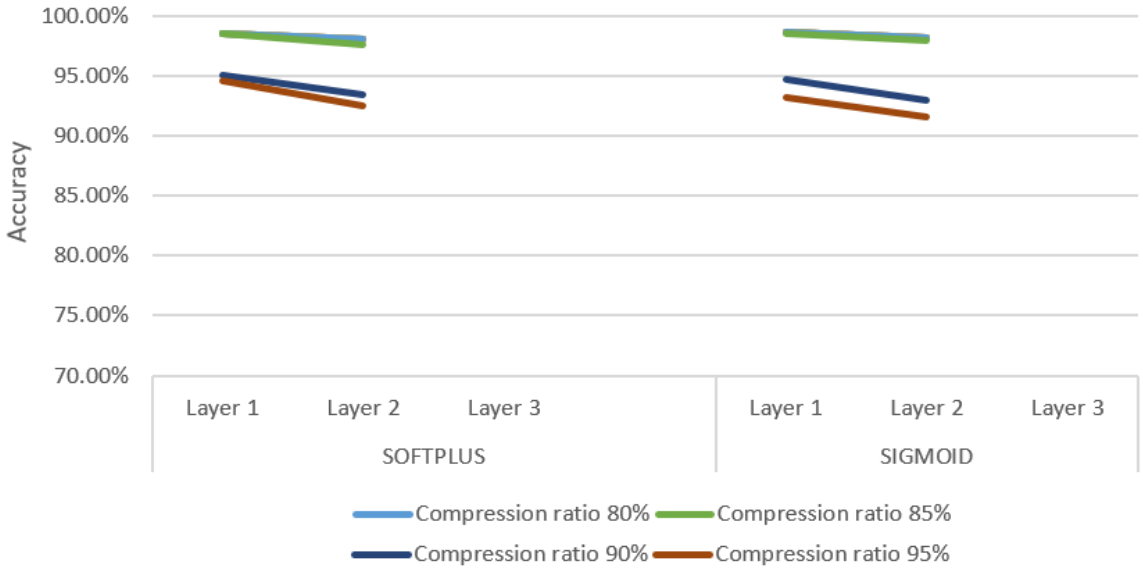


FIGURE 6.8: TI46 Data-set endpoint Data result for different compression rate using Keras Model with softplus and sigmoid activation function.

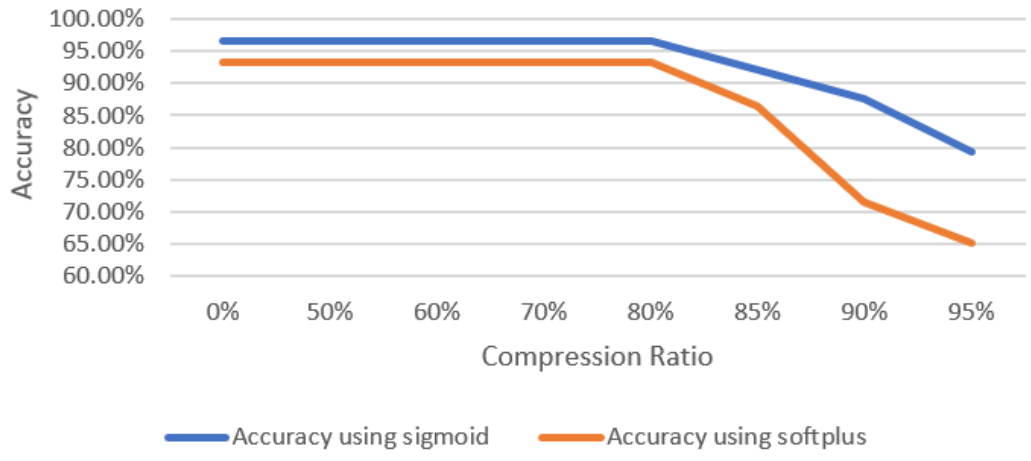


FIGURE 6.9: TI46 Data-set non-endpoint Data results for different compression rate using simple one hidden Layer Model with softplus and sigmoid activation function.

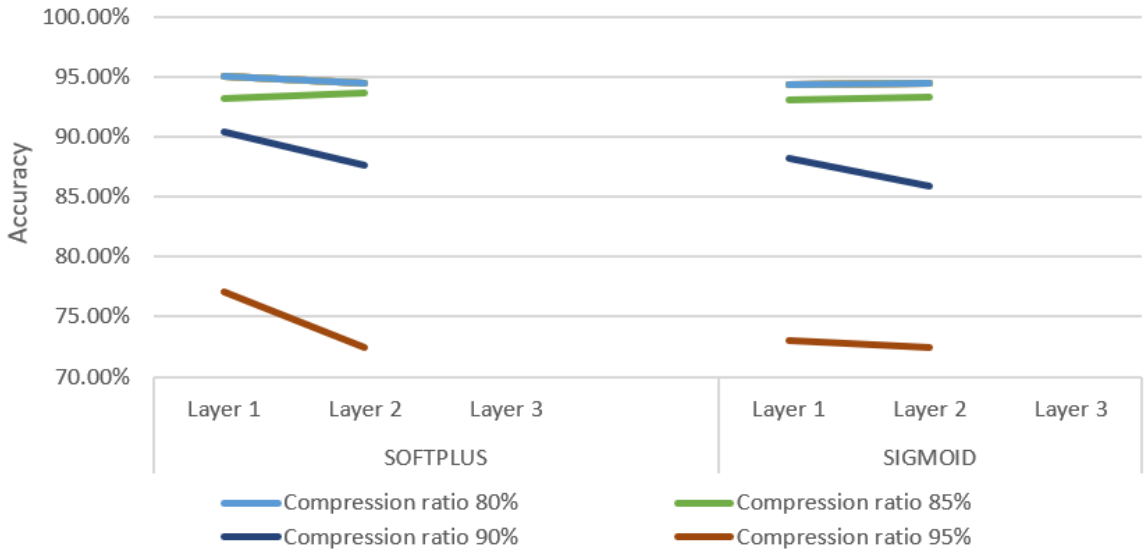


FIGURE 6.10: TI46 Data-set non-endpoint Data result for different compression rate using Keras Model with softplus and sigmoid activation function.

6.4 Summary

In this chapter, The data-set (MINIST Data-set and TI46 Data-set) which HWT was applied to it. The ANN using keras library were determined the optimum compression rates. By, keeping the data quality considerably the same even after the data coefficients have been cut off from the data. the testing accuracy were obtained after adding more hidden layers to the model and show the advantage and disadvantage

from adding them to the ANN.

Moreover, in section 6 presents a discussion of the experimental results obtained from the testing of the proposed system the simple one layer model and keras model in chapter 5 and earlier in this chapter.

In next chapter it will be the Conclusion of the thesis.

Chapter 7

Conclusion

In this final chapter, the conclusion of the thesis will be given by answering the research question stated in the introduction, and the thesis will conclude with potentials future work and improvements to our approaches.

7.1 Summary Research

As mention in chapter 2, in parallel with the increase in aggregate bandwidth for audio and image transmission ,storage and feature extraction by using fewer bits to represent the data information. Therefore, the researchers did sufficient work and are still working on this scenario. As, the demand for compression technology increases each year.

That's the reason, the project is aimed at developing a computationally efficient and effective algorithm for lossy data compression using wavelet techniques. The general idea behind it is to remove the redundancy in the data to find a more compact representation. This proposed algorithm was developed to compress the data such as MNIST Data-set, TI46 Data-set endpoint and non-endpoint to a high compression ratio. The results from the simulations showed that the quality of the reconstructed data preserved details while achieving significant compression rates.

Testing this reconstructed data using the ANN models. As, the ANN has emerged as a very beneficial in many areas of application in pattern recognition. The various types of ANNs developed over the years. Among them multi-layer feed forward network trained with back-propagation algorithm has been found as the most commonly used one. It is a supervised algorithm, which mean, it learns a problem with examples then recognises patterns, trends and hidden relationships within the similar problems represented by data with huge amount and complexity.

For the experiments, the ANN with two different activation function sigmoid and softplus, in addition to simulation the result using simple one hidden layer and using keras library is used to get the optimum compression rates . Confusion matrix was also used to judge the performance of the trained neural network. Experimental

results have shown a good performance for the ANN and HWT in both MNIST and TI46 data-sets.

The researcher has confidence in the thesis and problem description can be useful to researchers working in different disciplines where the HWT is used with these Data-sets using different kind ANN. The proposed algorithm presented the efficient, simple, and easy to implement. In addition, this algorithm helps in achieving high compression rate value that promises significant details and high quality during reconstruction of data. Sometimes, some compression algorithms are hard to implement or lose significant information during reconstruction, but the proposed method promises the reconstruction process with 95.2% accuracy for a cut-off of 80% in MNIST Data-set and 98.8% accuracy for a cut-off of 82% in TI46 Data-set with a minor loss of redundant information.

7.2 Conclusion

It is now possible to answer the research questions in chapter 1 (Introduction):

- **Question 1**

Among with various rates of compression, what is the ideal Haar-based compression signature for data (images and audio signals) that can be input into an ANN?

As seen in section 6.3, the compression rates to achieve balance between the quality and computational complexity it were 80% for MNIST Data-set and 82% for TI46 Data-set endpoint and non-endpoint. The results were considered an acceptable trade-off between accuracy and cut-off ratio.

- **Question 2**

What is the increase in performance of an ANN when using such signatures?

The HWT algorithm did not directly improve the result of pre-processing the data, but it opens the opportunities of training the ANN models with losing more than 80% of the data with the same accuracy without loss, which in respond improve the performance of an ANN.

- **Question 3**

How does the performance of the ANN change when using different activation functions and different numbers of hidden layers in the ANN?

As seen in section 6.3.1, the beneficial for MNIST Data-set of adding hidden layer as the highest accuracy level is 95.2% for using keras library with a two hidden layer ANN with the softplus activation function compared the results using one hidden layer or sigmoid activation function.

On other hand in section 6.3.2, the TI46 Data-sets experiments are shown that a maximum accuracy level of 98.8% using a single hidden layer ANN with sigmoid activation function. Witch mean, no beneficial from adding more hidden layers in the ANN.

7.3 Future work

In this section will introduce the future direction for research continuing on form:

7.3.1 Using different wavelet functions.

The benefit of using the algorithms proposed in this research Haar wavelet transform has been mention in the Introduction chapter. The HWT decomposition approach was not able to achieve a better result than this. However, it indicates that different other wavelet family, as mentioned in section 4.2.1 (Examples of Wavelet Family) Only the active functions should be changed from Haar functions to the other wavelet functions, such as Gabor and Daubechies wavelet may affect. And compare the result between of them.

7.3.2 Using different data-sets.

The MNIST and TI46 Data-sets were used in this thesis, but other data-sets can be used to perform similar experiments. The result of the experiments is encouraging to use in another data-sets. We believe our method can be extended to handle other data types, especially to process image querying, medical images, emotional (face and audio) data-sets where the concept of HWT is widely used.

7.3.3 Improve the configuration of ANN

It can be improve the configuration by doing more testing, by changing the feature numbers in ANN models, and use validation test for it.

As mentioned in 3.3 (Activation Functions), there is alot of different kind of activation function such as Hyperbolic Tangent Function- Tanh or Rectified Linear Units (ReLU). it can tested to see if there will be effect in the ANN models.

Moreover, Increasing the size of the data-set will help training the ANN and increasing the accuracy.

7.3.4 Using different ANN.

As mentioned in section 2.3 (Artificial life) , the creatur-wains A data mining unsupervised learning agent called the "Wain" is designed to run in the Créatúr framework by Amy de Buitléir. It is a new framework which we believe the HWT will improve the performance in it. Or increase the accuracy for pre-processing data.

Bibliography

[1] C. Preston, Z. Arnavut, and B. Koc, 'Lossless compression of medical images using Burrows-Wheeler Transformation with Inversion Coder', in 2015 37th Annual Interna-tional Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015, pp. 2956–2959.

[2] A. Avramović, G. Banjac, and J. Galić, 'Lossless audio compression using modular arithmetic and performance-based adaptation', in 2012 20th Telecommunications Forum (FOR), 2012, pp. 1256–1259.

[3] M. Ezhilarasan, P. Thambidurai, K. Praveena, S. Srinivasan, and N. Sumathi, 'A New Entropy Encoding Technique for Multimedia Data Compression', in International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), 2007, vol. 4, pp. 157–161.

[4] 5. K. K. Shukla and M. V. Prasad, Lossy Image Compression: Domain Decomposition-Based Algorithms. Springer Science and Business Media, 2011

[5] Grochenig K., Madych W. R.: Multiresolution Analysis, Haar bases and self–similar tilings of Rn. IEEE Transactions on Information Theory, 38(2), 556–568. 1994

[6] M. Ghiassi and H. Saidane, 'A dynamic architecture for artificial neural networks', Neurocomputing, vol. 63, pp. 397–413, Jan. 2005.

[7] 'MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges'. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[8] Mark Liberman et al. TI46wordLDC93s9.PhiladelphiawordLDC93s9.Philadelphia,1991. URL: https://catalog.ldc.upenn.edu/docs/LDC93S9/ti46.readme.html.

[9] N.G. Bourbakis, "A heuristic real-time path planning for collision free navigation in a dynamic multiple robots unknown environment", Tools for Artificial Intelligence 1989. Architectures Languages and Algorithms IEEE International Workshop on, pp. 658-667, 1989.

[10] H. L. Dreyfus, What Computers Cant Do. Bibliolife DBA of Bibilio Bazaar II LLC, 2016.

[11] F. E. Baird, Philosophic Classics: From Plato to Derrida. Routledge, 2016.

[12] A. J. Starko, Creativity in the Classroom: Schools of Curious Delight. Taylor Francis, 2010.

[13] J. R. Visintainer, 'Descartes' theory against artificial intelligence and the micro -world', Ph.D., Marquette University, United States – Wisconsin, 2002.

[14] 'Turing, Alan Mathison, (23 June 1912–7 June 1954), Reader in Mathematics, Manchester University, since 1948 | WHO'S WHO WHO WAS WHO'. [Online]. Available: http://www.ukwhoswho.com/view/10.1093/ww/9780199540891.001.0001/ww-9780199540884-e-243891. [Accessed: 23-Jan-2018].

[15] F. P. Miller, A. F. Vandome, and M. John, Computing Machinery and Intelligence. VDM Publishing, 2010.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction To Algorithms. MIT Press, 2001.

[17] A. Ǧafar M. b M. wārizmī, The algebra of Mohammed ben Musa. Murray, 1831.

[18] G. Boole, An Investigation of the Laws of Thought: On which are Founded the Mathematical Theories of Logic and Probabilities. Walton and Maberly, 1854.

[19] E. T. Jaynes, Probability Theory: The Logic of Science. Cambridge University Press, 2003.

[20] R. Pratik and R. Abhishek, The relationship between artificial intelligence and psychological theories. 2012.

[21] N. R. Carlson, H. L. Miller, D. S. Heth, G. N. Martin, and J. W. Donahoe, Psychology: The Science of Behavior. Pearson College Division, 2009.

[22] D. F. Barone, J. E. Maddux, and C. R. Snyder, Social Cognitive Psychology: History and Current Domains. Springer Science Business Media, 2012.

[23] 'George A Miller: The Magical Number Seven Plus or Minus Two', Psychology. [Online]. Available: https://www.all-about-psychology.com/george-a-miller.html. [Accessed: 05-Feb-2018].

[24] Pitts W, McCulloch WS. How we know universals the perception of auditory and visual forms. The Bulletin of mathematical biophysics. 1947 Sep 1;9(3):127-47.

[25] G. Johnson, 'Claude Shannon, Mathematician, Dies at 84', The New York Times, 27-Feb-2001.

[26] S. Dasgupta, It Began with Babbage: The Genesis of Computer Science. Oxford University Press, 2014.

[27] D. Shasha and C. Lazere, Out of their Minds: The Lives and Discoveries of 15 Great Computer Scientists. Springer Science Business Media, 1998.

[28] L. Uchitelle, 'H. P. Minsky, 77, Economist Who Decoded Lending Trends', The New York Times, 26-Oct-1996.

[29] J. McCarthy and E. A. Feigenbaum, 'In Memoriam: Arthur Samuel: Pioneer in Machine Learning', AI Mag., vol. 11, no. 3, p. 10, Sep. 1990.

[30] 'Ray Solomonoff, 83, Pioneer in Artificial Intelligence - The New York Times'. [Online]. Available: http://www.nytimes.com/2010/01/10/science/10solomonoff.html. [Accessed: 06-Feb-2018].

[31] J. Markoff, 'Oliver Selfridge, an Early Innovator in Artificial Intelligence, Dies at 82', The New York Times, 03-Dec-2008.

[32] 'Allen Newell'. [Online]. Available: http://www.umsl.edu/ piccininig/Newell%205.htm. [Accessed: 05-Feb-2018].

[33] Leahey TH. Herbert A. Simon: Nobel Prize in Economic Sciences, 1978. American Psychologist. 2003 Sep;58(9):753.

[34] Elouafiq A. Machine Cognition Models: EPAM and GPS. arXiv preprint arXiv:1204.1653. 2012 Apr 7.

[35] T. G. Evans, 'A Heuristic Program to Solve Geometric-analogy Problems', in Proceedings of the April 21-23, 1964, Spring Joint Computer Conference, New York, NY, USA, 1964, pp. 327–338.

[36] J. R. (James R. Slagle, 'A heuristic program that solves symbolic integration problems in freshman calculus: symbolic automatic integrator (SAINT)', Thesis, Massachusetts Institute of Technology, 1961.

[37] Nilsson NJ. turis. em. web. id Layanan Informasi 17 Jam.

[38] Kurzweil R, Richter R, Kurzweil R, Schneider ML. The age of intelligent machines. Cambridge, MA: MIT press; 1990 Sep.

[39] J. Haugeland, Artificial Intelligence: The Very Idea. Cambridge, MA, USA: Massachusetts Institute of Technology, 1985.

[40] R. E. Bellman, Artificial intelligence: can computers think? Boyd Fraser Pub. Co., 1978.

[41] R. Kurzweil, The age of intelligent machines. MIT Press, 1990.

[42] K. Knight, Artificial Intelligence. McGraw-Hill, 1991.

[43] E. A. CHARNIAK and D. V. McDermott, Introduction to Artificial Intelligence. Addison-Wesley Pub.Company, 1985.

[44] A. S. Winston, The Winstons of Hanover County, Virginia and Related Families, 1666-1992. Gateway Press, 1992.

[45] Schönberger D. Deep Copyright: Up-and Downstream Questions Related to Artificial Intelligence (AI) and Machine Learning (ML). Zeitschrift fuer Geistiges Eigentum/Intellectual Property Journal. 2018 Mar 1;10(1):35-58.

[46] Guttman A. R-trees: A dynamic index structure for spatial searching. ACM; 1984 Jun 18.

[47] R. J. Schalkoff, Artificial Intelligence: An Engineering Approach. McGraw-Hill, 1990.

[48] G. F. Luger and W. A. Stubblefield, Artificial intelligence: structures and strategies for complex problem solving. Benjamin/Cummings Pub. Co., 1993.

[49] 'Artificial intelligence (AI) in Architecture. What are the practical applications?', The community of innovative architects. [Online]. Available: http://groups.openbricks.io/t/artificial-intelligence-ai-in-architecture-what-are-the-practical-applications/364.

[50] 'Evolutionary Computation and beyond Elhanan Borenstein May ppt download'. [Online]. Available: http://slideplayer.com/slide/5139566/. [Accessed: 23-Feb-2018].

[51] Wang BY, Zhao XZ, Kang XY. Artificial life through GA in simulation of modern anti-surface warfare of warship fleet. InIntelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09. International Conference on 2009 Aug 26 (Vol. 1, pp. 379-382). IEEE.

[52] Teahan WJ. Artificial Intelligence–Agents and Environments. BookBoon; 2010.

[53] Von Neumann J. The general and logical theory of automata. Cerebral mechanisms in behavior. 1951;1(41):1-2.

[54] J. Von Neumann and A. W. (Arthur W. Burks, Theory of self-reproducing automata. Urbana, University of Illinois Press, 1966.

[55] A. de Buitléir, M. Russell, and M. Daly, 'Wains: A Pattern-Seeking Artificial Life Species', Artif. Life, vol. 18, no. 4, pp. 399–423, Oct. 2012.

[56] Martínez AM, Kak AC. Pca versus lda. IEEE Transactions on Pattern Analysis Machine Intelligence. 2001 Feb 1(2):228-33.

[57] P. Dangeti, Statistics for Machine Learning. Packt Publishing Ltd, 2017.

[58] 'The Difference Between Machine Learning and Statistics', Galvanize Blog, 26-Aug-2015. .

[59] 'Robert Tibshirani | Department of Statistics'. [Online]. Available: https://statistics.stanford.edu/people/robert-tibshirani. [Accessed: 05-Mar-2018].

[60] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning: with Applications in R. Springer Science Business Media, 2013.

[61] 'Machine Learning Algorithm - Backbone of emerging technologies', TechLeer. [Online]. Available: https://www.techleer.com/articles/203-machine-learning-algorithm-backbone-of-emerging-technologies/. [Accessed: 27-Feb-2018].

[62] Mohri M, Talwalkar A, Rostamizadeh A. Foundations of machine learning (adaptive computation and machine learning series). Cambridge, MA: Mit Press; 2012.

[63] M. E. Celebi and K. Aydin, Unsupervised Learning Algorithms. Springer, 2016.

[64] Andrey P, Tarroux P. Unsupervised image segmentation using a distributed genetic algorithm. Pattern Recognition. 1994 May 1;27(5):659-73.

[65] 'Unsupervised Learning Projects and Research Topics – MTech Projects'. [Online]. Available: http://www.mtechprojects.org/unsupervised-learning-projects.html.

[66] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. MIT Press, 1998.

[67] X. Zhu and A. B. Goldberg, Introduction to Semi-supervised Learning. Morgan Claypool Publishers, 2009.

[68] de Buitléir, A.; Russell, M. Daly, M. (2015), 'The self-generating Model and its use in Handwritten Numeral Recognition.', Artificial Life ,self-organizing map , self-generating Model .

[69] KDD: Proceedings. AAAI Press, 2000.

[70] S. Agarwal, 'Data Mining: Data Mining Concepts and Techniques', in 2013 International Conference on Machine Intelligence and Research Advancement, 2013, pp. 203–207.

[71] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, 'Advances in Knowledge Discovery and Data Mining', U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996, pp. 1–34.

[72] S. Zhang, C. Zhang, and Q. Yang, 'Data preparation for data mining', Appl. Artif. Intell., vol. 17, no. 5–6, pp. 375–381, May 2003.

[73] A. Duhamel, M. C. Nuttens, P. Devos, M. Picavet, and R. Beuscart, 'A preprocessing method for improving data mining techniques. Application to a large medical diabetes database', Stud. Health Technol. Inform., vol. 95, pp. 269–274, 2003.

[74] A. A. Freitas, 'The principle of transformation between efficiency and effectiveness: Towards a fair evaluation of the cost-effectiveness of KDD techniques', in Principles of Data Mining and Knowledge Discovery, 1997, pp. 299–306.

[75] U. Fayyad and P. Stolorz, 'Data mining and KDD: Promise and challenges', Future Gener. Comput. Syst., vol. 13, no. 2, pp. 99–115, Nov. 1997.

[76] S. Bagga and D. G. N. Singh, 'CONCEPTUAL THREE PHASE ITERATIVE MODEL OF KDD', Int. J. Comput. Technol., vol. 2, no. 1, pp. 6–8, Jan. 2003.

[77] J. Mahoney, 'The Logic of Process Tracing Tests in the Social Sciences', Sociol. Methods Res., vol. 41, pp. 570–597, Mar. 2012.

[78] O. Z. Maimon and L. Rokach, Data Mining and Knowledge Discovery Handbook. Springer Science Business Media, 2005.

[79] H. J. Miller and J. Han, Geographic Data Mining and Knowledge Discovery, Second Edition. CRC Press, 2009.

[80] M. Rathi, 'Regression Modeling Technique on Data Mining for Prediction of CRM', in Information and Communication Technologies, 2010, pp. 195–200.

[81] A. Giusti, G. Ritter, and M. Vichi, Classification and Data Mining. Springer Science Business Media, 2012.

[82] L. Wang, Support Vector Machines: Theory and Applications. Springer Science Business Media, 2005.

[83] M. O. Z and R. Lior, Data Mining With Decision Trees: Theory And Applications (2nd Edition). World Scientific, 2014.

[84] 'Datasets for Machine Learning', webkid blog, 15-Aug-2016. [Online]. Available: http://blog.webkid.io/datasets-for-machine-learning/. [Accessed: 05-Mar-2018].

[85] 'MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges'. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 15-Jan-2018].

[86] Mark Liberman et al. TI46wordLDC93s9.PhiladelphiawordLDC93s9.Philadelphia,1991. URL: https://catalog.ldc.upenn.edu/docs/LDC93S9/ti46.readme.html.

[87] de Buitléir A, Flynn R, Russell M, Daly M. An architecture for pattern recognition and decision-making. InInternational Conference on Simulation of Adaptive Behavior 2016 Aug 23 (pp. 22-33). Springer, Cham.

[88] Varga A, Moore RK. Hidden Markov model decomposition of speech and noise. InInternational Conference on Acoustics, Speech, and Signal Processing 1990 Apr 3 (pp. 845-848). IEEE.

[89] Roy T, Marwala T, Chakraverty S. Precise detection of speech endpoints dynamically: A wavelet convolution based approach. Communications in Nonlinear Science and Numerical Simulation. 2019 Feb 1;67:162-75.

[90] Cajal SR. Texture of the Nervous System of Man and the Vertebrates. Springer Science Business Media; 1999 Mar 2.

[91] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics. 1943 Dec 1;5(4):115-33.

[92] S. K. Rogers and M. Kabrisky, An Introduction to Biological and Artificial Neural Networks for Pattern Recognition. SPIE Press, 1991.

[93] madelineschiappa. Man vs machine: comparing artificial and biological neural networks [Internet]. Sophos News. 2017 [cited 2018 Dec 23]. Available from: https://news.sophos.com/en-us/2017/09/21/man-vs-machine-comparing-artificial-and-biological-neural-networks/

[94] Beale R, Jackson T. Neural Computing-an introduction. CRC Press; 1990 Jan 1.

[95] Billings SA, Zheng GL. Radial basis function network configuration using genetic algorithms. Neural Networks. 1995 Jan 1;8(6):877-90.

[96] A. G. Parlos, B. Fernandez, A. F. Atiya, J. Muthusami, and W. K. Tsai, 'An accelerated learning algorithm for multilayer perceptron networks', IEEE Trans. Neural Netw., vol. 5, no. 3, pp. 493–497, May 1994.

[97] W. Zaremba, I. Sutskever, and O. Vinyals, 'Recurrent Neural Network Regularization', ArXiv14092329 Cs, Sep. 2014.

[98] Salomon D. Data compression: the complete reference. Springer Science Business Media; 2004 Feb 26.

[99] Fallon JJ, inventor; Realtime Data LLC, assignee. Content independent data compression method and system. United States patent US 6,309,424. 2001 Oct 30

[100] Fallon JJ, Bo SL, inventors; Realtime Data LLC, assignee. System and method for lossless data compression and decompression. United States patent US 6,597,812. 2003 Jul 22.

[101] Kodituwakku SR, Amarasinghe US. Comparison of lossless data compression algorithms for text data. Indian journal of computer science and engineering. 2010;1(4):416-25.

[102] Arnold R, Bell T. A corpus for the evaluation of lossless compression algorithms. Indcc 1997 Mar 25 (p. 201). IEEE.

[103] Wang Z, Houle PS, inventors; AOL Inc, assignee. Data compression using adaptive bit allocation and hybrid lossless entropy encoding. United States patent US 6,049,630. 2000 Apr 11.

[104] Lu WW, Gough MP. A fast-adaptive Huffman coding algorithm. IEEE Transactions on communications. 1993 Apr;41(4):535-8.

[105] Witten IH, Neal RM, Cleary JG. Arithmetic coding for data compression. Communications of the ACM. 1987 Jun 1;30(6):520-40.

[106] Hauck EL, inventor; INTELLIGENT STORAGE Inc, assignee. Data compression using run length encoding and statistical encoding. United States patent US 4,626,829. 1986 Dec 2.

[107] Kontoyiannis I. Pointwise redundancy in lossy data compression and universal lossy data compression. IEEE Transactions on Information Theory. 2000 Jan;46(1):136-52.

[108] Al-Shaykh OK, Mersereau RM. Lossy compression of noisy images. IEEE Transactions on Image Processing. 1998 Dec 1;7(12):1641-52.

[109] Rao KR, Yip P. Discrete cosine transform: algorithms, advantages, applications. Academic press; 2014 Jun 28.

[110] Stanković RS, Falkowski BJ. The Haar wavelet transform: its status and achievements. Computers Electrical Engineering. 2003 Jan 1;29(1):25-44.

[111] Princen J, Johnson A, Bradley A. Subband/transform coding using filter bank designs based on time domain aliasing cancellation. InAcoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87. 1987 Apr (Vol. 12, pp. 2161-2164). IEEE.

[112] Mertins A, Mertins DA. Signal analysis: wavelets, filter banks, time-frequency transforms and applications. John Wiley Sons, Inc.; 1999 Jun 1.

[113] Robertson DC, Camps OI, Mayer J. Wavelets and power system transients: feature detection and classification. InWavelet Applications 1994 Mar 15 (Vol. 2242, pp. 474-488). International Society for Optics and Photonics.

[114] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825-30.

[115] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M. Tensorflow: A system for large-scale machine learning. In12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) 2016 (pp. 265-283).

[116] Van Der Walt S, Colbert SC, Varoquaux G. The NumPy array: a structure for efficient numerical computation. Computing in Science Engineering. 2011 Mar 14;13(2):22.

[117] Bastien F, Lamblin P, Pascanu R, Bergstra J, Goodfellow I, Bergeron A, Bouchard N, Warde-Farley D, Bengio Y. Theano: new features and speed improvements. arXiv preprint arXiv:1211.5590. 2012 Nov 23.

[118] Kumar PA. Implementation of image compression algorithm using Verilog with area, power and timing constraints. Master's Thesis, National Institute of Technology, Rourkela, India, 2009.

[119] Gupta, M, Garg, AK. Analysis of image compression algorithm using DCT. Int J Eng Res Appl 2012; 2: 515–521.

[120] Porwik P and Lisowska A. The Haar-wavelet transform in digital image processing: its status and achievements. Mach Graph Vision 2004; 13: 79–98.

[121] Kaur EK, Chawla ER and Kaur EI. Image compression using wavelet transform. Int J Inf Technol 2012; 5: 421–423.

[122] Talukder KH, Harada K. Haar wavelet based approach for image compression and quality assessment of compressed image. arXiv preprint arXiv:1010.4084. 2010 Oct 20.

[123] Lai BL, Chang LW. Adaptive data hiding for images based on harr discrete wavelet transform. InPacific-Rim Symposium on Image and Video Technology 2006 Dec 10 (pp. 1085-1093). Springer, Berlin, Heidelberg.

[124] Ye Z, Mohamadian H, Ye Y. Information measures for biometric identification via 2d discrete wavelet transform. In2007 IEEE International Conference on Automation Science and Engineering 2007 Sep 22 (pp. 835-840). IEEE.

[125] Osowski S, Waszczuk R, Bojarczak P. Image compression using feedforward neural networks—Hierarchical approach. InInternational Workshop on Artificial Neural Networks 1995 Jun 7 (pp. 1009-1015). Springer, Berlin, Heidelberg.

[126] Ma L, Khorasani K. Adaptive constructive neural networks using hermite polynomials for image compression. InInternational Symposium on Neural Networks 2005 May 30 (pp. 713-722). Springer, Berlin, Heidelberg.

[127] Karlık B. Medical image compression by using vector quantization neural network (VQNN). Neural Network World. 2006;4:06.

[128] Soliman HS, Omari M. A neural networks approach to image data compression. Applied Soft Computing. 2006 Mar 1;6(3):258-71.

[129] Northan B, Dony RD. Image compression with a multiresolution neural network. Canadian Journal of Electrical and Computer Engineering. 2006;31(1):49-58.

[130] Vilovic I. An experience in image compression using neural networks. InProceedings ELMAR 2006 2006 Jun (pp. 95-98). IEEE.

[131] Ashraf R, Akbar M. Absolutely lossless compression of medical images. In2005 IEEE Engineering in Medicine and Biology 27th Annual Conference 2006 Jan 17 (pp. 4006-4009). IEEE.

[132] O. Rioul and M. Vetterli, "Wavelets and Signal Processing", IEEE Signal Process. Mag. Vol 8, pp. 14-38, Oct. 1991..

[133] HatemElaydi and Mustafi I. Jaber and Mohammed B. Tanboura, "Speech compression using Wavelets", International Journal for Applied Sciences, Vol 2, 1-4,Sep 2011

[134] Othman O. Khalifa, SeringHabib Harding Aisha-Hassan A. Hashim "Compression using Wavelet Transform" in Signal Processing: An International Journal, Volume (2) : Issue (5).

[135] Graves A, Mohamed AR, Hinton G. Speech recognition with deep recurrent neural networks. In2013 IEEE international conference on acoustics, speech and signal processing 2013 May 26 (pp. 6645-6649). IEEE.

[136] Dahl GE, Yu D, Deng L, Acero A. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on audio, speech, and language processing. 2012 Jan;20(1):30-42.

[137] Yeow WL, Mahmud R, Raj RG. An application of case-based reasoning with machine learning for forensic autopsy. Expert Systems with Applications. 2014 Jun 1;41(7):3497-505.

[138] Wu JD, Liu CH. An expert system for fault diagnosis in internal combustion engines using wavelet packet transform and neural network. Expert systems with applications. 2009 Apr 1;36(3):4278-86.

[139] Sakata D, Akiyama Y, Kaneko M, Kumagai S. Education System to Learn the Skills of Management Decision-Making by Using Business Simulator with Speech Recognition Technology. Industrial Engineering Management Systems. 2014 Sep;13(3):267-77.

[140] Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks. InInternational Conference on Machine Learning 2014 Jan 27 (pp. 1764-1772).

[141] Abdel-Hamid O, Mohamed AR, Jiang H, Deng L, Penn G, Yu D. Convolutional neural networks for speech recognition. IEEE/ACM Transactions on audio, speech, and language processing. 2014 Oct;22(10):1533-45.

[142] Arısoy E, Chen SF, Ramabhadran B, Sethy A. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2014 Jan;22(1):184-92.

[143] Raj RG, Abdul-Kareem S. Information Dissemination And Storage For Tele-Text Based Conversational Systems' Learning. Malaysian Journal of Computer Science. 2009 Dec 1;22(2):138-60.

[144] Tamulevičius G, Arminas V, Ivanovas E, Navakauskas D. Hardware accelerated FPGA implementation of Lithuanian isolated word recognition system. Elektronika ir elektrotechnika. 2010 Jan 1;99(3):57-62.

[145] Hai NT, Van Thuyen N, Mai TT, Van Toi V. MFCC-DTW algorithm for speech recognition in an intelligent wheelchair. In5th international conference on biomedical engineering in Vietnam 2015 (pp. 417-421). Springer, Cham.

[146] Ali H, Ahmad N, Zhou X, Ali M, Manjotho AA. Linear discriminant analysis based approach for automatic speech recognition of Urdu isolated words. InInternational Multi Topic Conference 2013 Dec 18 (pp. 24-34). Springer, Cham.

[147] Ali H, Ahmad N, Zhou X, Iqbal K, Ali SM. DWT features performance analysis for automatic speech recognition of Urdu. SpringerPlus. 2014 Dec;3(1):204.

[148] Ashraf J, Iqbal N, Khattak NS, Zaidi AM. Speaker independent Urdu speech recognition using HMM. In2010 The 7th International Conference on Informatics and Systems (INFOS) 2010 Mar 28 (pp. 1-5). IEEE.

[149] Jain AK, Mao J, Mohiuddin KM. Artificial neural networks: A tutorial. Computer. 1996 Mar;29(3):31-44.

[150] Schalkoff RJ. Artificial neural networks. New York: McGraw-Hill; 1997 Jan.

[151] Panchal G, Panchal M. Review on methods of selecting number of hidden nodes in artificial neural network. International Journal of Computer Science and Mobile Computing. 2014;3(11):455-64.

[152] Basheer IA, Hajmeer M. Artificial neural networks: fundamentals, computing, design, and application. Journal of microbiological methods. 2000 Dec 1;43(1):3-1.

[153] Huang GB, Zhu QY, Siew CK. Extreme learning machine: a new learning scheme of feedforward neural networks. InNeural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on 2004 Jul 25 (Vol. 2, pp. 985-990). IEEE.

[154] Hagan MT, Menhaj MB. Training feedforward networks with the Marquardt algorithm. IEEE transactions on Neural Networks. 1994 Nov;5(6):989-93.

[155] Mavrovouniotis M, Yang S. Training neural networks with ant colony optimization algorithms for pattern classification. Soft Computing. 2015 Jun 1;19(6):1511-22.

[156] Gardner MW, Dorling SR. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment. 1998 Aug 1;32(14-15):2627-36.

[157] Jain AK, Mao J, Mohiuddin KM. Artificial neural networks: A tutorial. Computer. 1996 Mar;29(3):31-44.

[158] Specht DF. A general regression neural network. IEEE transactions on neural networks. 1991 Nov;2(6):568-76.

[159] 1. Hao Z. Activation Functions in Neural Networks [Internet]. Isaac Changhau. 2017.

[160] Park DC, El-Sharkawi MA, Marks RJ, Atlas LE, Damborg MJ. Electric load forecasting using an artificial neural network. IEEE transactions on Power Systems. 1991 May;6(2):442-9.

[161] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. InProceedings of the 27th international conference on machine learning (ICML-10) 2010 (pp. 807-814).

[162] Roffo G. Ranking to Learn and Learning to Rank: On the Role of Ranking in Pattern Recognition Applications. arXiv preprint arXiv:1706.05933. 2017 Jun 1.

[163] C. K. Chui, An Introduction to Wavelets. Elsevier, 2016.

[164] Domingues MO, Mendes Jr O, Da Costa AM. On wavelet techniques in atmospheric sciences. Advances in Space Research. 2005 Jan 1;35(5):831-42.

[165] Singh R, Vasquez RE, Singh R. Comparison of Daubechies, Coiflet, and Symlet for edge detection. InVisual Information Processing VI 1997 Jul 22 (Vol. 3074, pp. 151-159). International Society for Optics and Photonics.

[166] Allibone TE. Dennis Gabor, 5 June 1900-9 February 1979.

[167] Cebrián JJ, González CM, Ornedo VN. 2-D Discrete Wavelet Transform for Hand Palm Texture Biometric Identification and Verification. InWavelet Transforms and Their Recent Applications in Biology and Geoscience 2012 Mar 2. IntechOpen.

[168] A guide for using the Wavelet Transform in Machine Learning [Internet]. Ahmet Taspinar. 2019 [cited 2019Nov12]. Available from: http://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

[169] Shensa MJ. The discrete wavelet transform: wedding the a trous and Mallat algorithms. IEEE Transactions on signal processing. 1992 Oct;40(10):2464-82.

[170] Stanković RS, Falkowski BJ. The Haar wavelet transform: its status and achievements. Computers Electrical Engineering. 2003 Jan 1;29(1):25-44.

[171] Agarwal N, Gupta S, Pradhan A, Vishwanathan K, Panigrahi PK. Wavelet transform of breast tissue fluorescence spectra: a technique for diagnosis of tumors. IEEE Journal of selected topics in quantum electronics. 2003 Mar;9(2):154-61.

[172] Chollet F. Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH Co. KG; 2018 May 24.

[173] Gulli A, Pal S. Deep Learning with Keras. Packt Publishing Ltd; 2017 Apr 26.

[174] Goodfellow I, Bengio Y, Courville A. Deep learning. Book in preparation for MIT Press. URL¡ http://www. deeplearningbook. org. 2016.

[175] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. InProceedings of the 27th international conference on machine learning (ICML-10) 2010 (pp. 807-814).

[176] W. Zaremba, I. Sutskever, and O. Vinyals, 'Recurrent Neural Network Regularization', ArXiv14092329 Cs, Sep. 2014.