Expertise
and insight
for the future

Nguyen Mai Vinh

Artificial Intelligence application: Categorize route and non-route given predefined route

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

17 April 2020

Author	Nguyen Mai Vinh
Title	Artificial Intelligence application: Categorize route and non-route given predefined route
Number of Pages	58 pages + 3 appendices
Date	17 April 2020
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Smart Systems
Instructors	Patrick Ausderau, Principal Lecturer

The purpose of this final year project is to create an Artificial Intelligence application to study the possibility of using Neural networks to categorize route points based on predefined routes. The information can be used to determine how well the new route follows the predefined routes.

The method chosen was to construct a Neural network to learn from the data which is in the form of X and Y coordinates with 2 types of labels: route and non-route. The trained neural network will take the new route data points and determine which point is in the predefined route. As a result, the accuracy level is calculated for this route to examine how well the new route follows the predefined route.

The outcome of the project was a Python application which utilizes the TensorFlow framework to build the Neural Network for route points classification. Matplotlib library is used for data visualization.

In conclusion, the project proves that applying Artificial Intelligence for classification problems is possible. Based on the results, more complex data models and methods need to be explored for high accuracy in the more complex situations.

Keywords	Machine Learning, Artificial Intelligence, Artificial Neural Network, Python, TensorFlow

Contents

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
ML	Machine Learning
DL	Deep Learning
DS	Data Science
GPS	General Problem Solver
RL	Reinforcement Learning
NLP	Natural Language Processing
ReLu	Rectified Linear Units

Glossary

Quaternions	A mathematical notation for representing orientations and rotations of objects in 3-dimensional space
XY coordinates	Horizontal and vertical addresses of a point in a map or display screen

1 Introduction

Over the past few years, Machine Learning (ML) and Artificial Intelligence (AI) has become a big trend due to its contribution to various business sectors, manufacturing and military. [1 p.VIII] This trend is supported by two main factors: increasing computing power and the availability of big data. The more data is analyzed, the better the pattern recognition is performed and as a result, knowledge and insight view is deepened.

Wizense Oy[1] is a company that works on the development of wearable tracking devices using wireless technology. Based on the collected data, which includes timestamp and quaternions, useful applications could be created to assist users in many different ways; for example, the wearable device could advise on whether the user is following the correct suggested route.

The objective of this thesis is to propose a solution to the challenge by providing a Neural network model to learn from collected data and output useful advice for users. This study also aims to answer the following question: How could ML and AI could help companies to make better decisions. Various ML fields are also introduced and discussed.

The outcome of the project is an AI application system built on Neural Network which categorizes and labels the footstep positions of the wireless wearable devices users in terms of XY-coordinate to determine whether or not they are following a predefined optimized route. Python programming language, TensorFlow ML framework, Numpy and Pandas data framework are used for this project.

Firstly, data is obtained as XY-coordinates and saved in a csv file with the correct labels, then this data is fed into a Neural network model to learn. As a result, a trained model could be used consequently to categorize new route points data into two labels: route or non-route points. Finally, the graphical representation of the route data points and what is the percentage of them following the route is shown. Furthermore, this project also evaluates how well neural networks perform the above task in terms of accuracy and

[1] https://wizense.com/about-us

training time, different hyper-parameter tuning process, complexities of the predefined route, problem encountered during implementation and lesson learnt.

The study is divided into six chapters. The first chapter provides the introduction in which business requirements, objectives and outcomes of the study are introduced. The second chapter discusses theoretical background related to the study topics such as AI, ML, and Artificial Neural Network (ANN). The third chapter covers the project specifications. The fourth chapter describes the implementation in detail. The fifth chapter shows the result and discussion of the project. Finally, the sixth chapter provides the conclusion.

2 Theoretical background of Artificial Intelligence

This chapter explores key topics that relate to the Bachelor thesis. Therefore, it focuses on discussing Machine Learning, Artificial Intelligence and Neural Network.

2.1 History of Artificial Intelligence

The field of AI started with the birth of computers around the year 1950s. In the early development, the focus is on how to make computers imitate the intelligence of human beings [1 p.VII]. For example, there are attempts to make computers perform all aspects in human behaviour. This eventually leads to the philosophical discussion of how close to the human brain a computer is and whether or not the difference between them matters.

Even though evidence shows strong connection between the emergence of AI and the development of computers, the AI concept was found long before the existence of modern computers. [1 p3] For example, René Descartes, who is a XVII century French philosopher and mathematician, considered animals as a machine models in which the control mechanism is created to follow the predetermined sequence of instructions. Moreover, the concept of automaton is similar to the concept of humanoid robots today. A famous example of automaton is the Maillardet's automaton which was built in the 1800s by a Swiss mechanician, Henri Maillardet. It is currently displayed at the Franklin Institute and is shown in figure 1. However, the artificial beings could be discovered even further such as the Golem of Prague which was created by Rabbi Loew out of clay from the Vltava River and brought to life through rituals around the late 16th century.

Figure 1. Maillardet's automaton at the Franklin Institute [2]

The most relevant and strongest roots of modern AI could be linked to the work in 1943 of McCulloch and Pitts, who represented the mathematical models of neurons in the brain cells. These models are called perceptrons and are based on the detailed analysis of how biological brains work. They described how neurons operate in a switching binary fashion in which neurons either fire—"on" state—or do not fire the signal—"off" state. This model also explains how such neurons could learn from experience and change their actions according to external circumstances. [1 p2]

The first AI computer which was based on the neuron models were built by Marvin Minsky and Dean Edmonds. At the same time, Claude Shannon worked on the topic of computer playing chess and strategies on how to decide the next move. In 1956, these researchers came to the first workshop in Dartmouth College in the USA and laid the foundation for the new field of AI. [1 p3]

In the 1960s, there was a huge effort in making computers understand and respond to human natural language. This was not only driven by the Turing test which examines the machine's ability to show intelligent behavior indistinguishable from that of a human, but also by the desire of making computers ready to interface with the real world. [1] The remarkable example is Joseph Weisenbaum's ELIZA which is one of the best English-

speaking computers and laid the foundation for modern “Chatterbots”. [1 p4] The conversation produced by ELIZA was realistic enough that it is not known whether the communication is from machine or human.

During the 1960s, one of the most significant contributions to the AI field was the work of Newell and Simon on General Problem Solver (GPS). This is a multi-purpose computer designed to simulate problem solving methods of a human. The basic principle is to form real world problems into a set of well-formed formulas or Horn clauses. These formulas construct a graph with multiple sources and sinks in which sources refer to axioms and sinks refer to conclusions. [3]

To be more specific, the objects and operations on the objects are defined and GPS will solve the problem by finding heuristics using means-ends analysis. Unfortunately, the project was abandoned due to inefficiency of the technique, memory requirements and time taken to solve straightforward real problems. The basic algorithm of GPS and block diagram are described in figure 2.

```
begin:
- save initial conditions as "current state"
- try to achieve all goals

how to achieve all goals:
- for each goal:
  - try to achieve the goal
- if all goals were achieved, return success

how to achieve a goal:
- if goal is already met (in current state), return success (it's achieved)
- else, for each operator:
  - if operator's "add list" contains the goal,
    - try to apply the operator
    - if successful, return success
- if no operators were successful,
  - return failure

how to apply an operator:
- try to achieve all of the operator's preconditions (treat them as goals)
- if successful (all preconditions met),
  - mark the operator as "applied"
  - add conditions in the operator's "add list" to the current state
  - delete conditions in the operator's "delete list" from the current state
  - return success
- else,
  - return failure
```

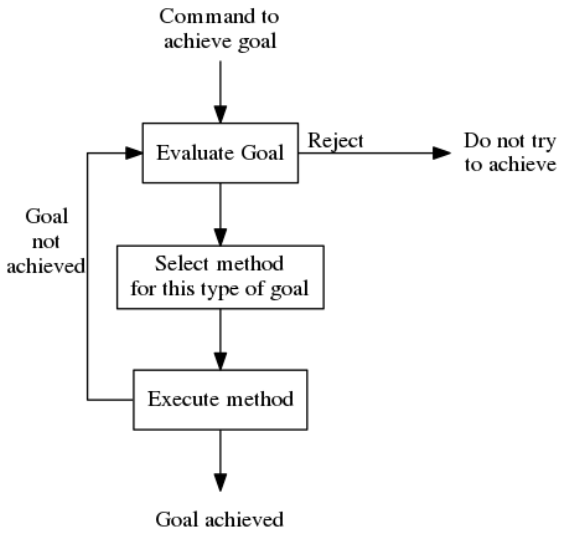

Figure 2. General Problem Solver search algorithm and block diagram [4]

In the 1980s, there was a new approach to AI in which AI is no longer restricted to copy human intelligence. As a result, AI could be intelligent in its own way. This implies that AI could be better than a human brain and outperform humans in certain tasks. Recent

development in the AI field has proven this point. AI applications have outperformed humans in the fields of military, finance and biology pattern detection. [1 pviii]

Moreover, AI also could be better than humans in playing games. The typical case is AlphaGo, which is the first computer that beat a professional Go player without handicap in October 2015. In March 2016, AlphaGo won Lee Sedol who is a 9-dan professional without handicap. Furthermore, in 2017, AlphaGo Master beat Ke Jie, the world top player at that time. Eventually, AlphaGo was awarded a 9-dan honorary professional by both Korea Badul Association and Chinese Weiqi Association. AlphaGo is a computer program, developed by DeepMind Technologies, that plays the board game Go. It combines Monte Carlo tree search and ANN to determine its moves based on learnt experience by extensive training on historical games which were played by both humans and computers. The combination enhances the quality of move selection and strengthens self-evolvement in the next iteration. [5]

2.2 What is intelligence

It is important to understand the concept of intelligence before the discussion of AI. What is considered as intelligence in different entities such as in humans, animals and machines. Obviously, the concept of intelligence is based on each person's views, experiences and what is important to him or her. The subjective nature of intelligence is significant since it implies that what is considered as intelligent could be easily changed dependent on time and place. For example, the New English Dictionary of 1932 clearly defines intelligence as follows: “The exercise of understanding; intellect power; acquire power or quickness of intellect”. [1 p13] This definition emphasizes human intelligence based on knowledge and speed. Recently, Macmillan Encyclopedia of 1995 defined intelligence as ability to reason and profit from experience. [1 p14] In this case, the focus is on the complicated interaction between an individual's qualities and the environment.

2.2.1 Animal Intelligence

In order to allow for more possibilities, it is also important to discuss the intelligence of animals. Many aspects of intelligence such as planning and communication as well as reasoning, learning and using tools will be considered.

It is observed that bees communicate with one another in the form of complex dance routines. Returning from pollen collection activity, a bee performs a dance in which it wiggles its bottom and moves forward in a straight line at the hive entrance. The distance moved by the bee in its dance is proportional to the distance from the hive to the pollen source. In addition, the angle which the bee moves also indicates the angle between the source and the sun. This is how bees could learn from each other how to determine good directions to fly. [1]

Water spiders exhibit excellent planning skills to catch their prey. A water spider builds an air-filled diving bell out of silk and waits patiently underwater for a shrimp to pass by. At the right moment, it pounces to give a deadly bite and pulls the shrimp into its lair. [1]

Furthermore, octopuses show interesting learning capability. If an octopus is trained to perform a task such as choosing objects of different colour, the second octopus also could perform the same task by simply watching the first one. [1 p14]

On the other hand, the ability to use tools is remarkable in the case of green herons. They drop small pieces of food into water to lure fish into that area and when a fish swims nearby to take the bait, they catch it. [1 p15]

However, the most studied non-human animal is chimpanzees and monkeys due to their genetic similarity to humans. They exhibit a variety of complex skills which are very close to humans such as planning hunting trips, using different tools and climbing for food search and collection. They also communicate and convey emotions with each other using complex vocalizations, facial expressions and body gestures. Studies have shown that the intelligence of animals is easier to evaluate if its expressions are close to that of humans and harder to measure if the expressions are meaningless to humans. [1 p15]

2.2.2 Sensing and interaction

Intelligence arguably depends not only on the brain—processing data—but also how it could sense—the amount of data—and interact or react with the surrounding environment—the output. These are the three basic pillars that support the intelligence concept. [1 p17] In terms of sense, a human has five basic senses which includes touch, smell, taste, hearing and vision. [6] In terms of vision, humans are not able to sense frequencies which are out of visible light such as ultraviolet or infrared frequency range. In terms of hearing, most humans can sense sound in the range of 20 Hz to 20000 Hz and we could not hear ultrasonic sound. Therefore, humans' perception of the world is limited. [1]

On the other hand, machines or animals with different senses are able to experience the reality that humans have no sense of that. For example, bats could detect frequencies in ultrasonic range which is from 100 kHz to 200 kHz. [1 p163] Another example is a thermal camera which is a device that creates images using infrared radiation. Compared to common cameras that capture visible light in the range of 400 - 700 nanometer wavelengths, thermal cameras are sensitive to infrared wavelengths ranging from approximately 0.75 to 1000 microns. [7] The difference between visible light and thermal views is illustrated in figure 3.

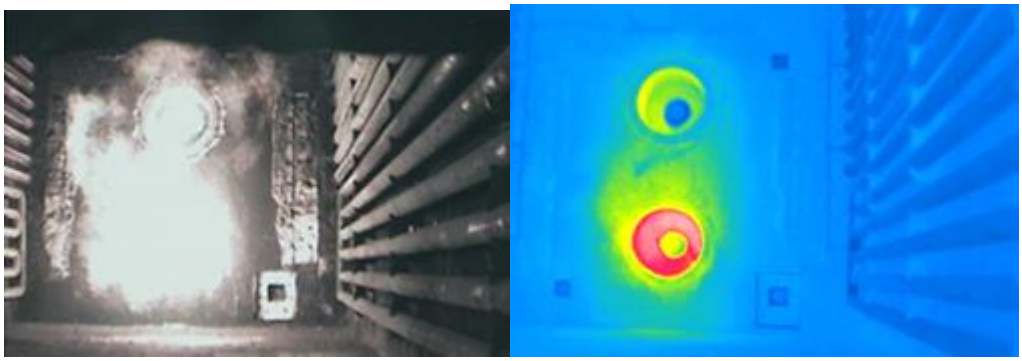

Figure 3. Left: The burner behind the flame is invisible. Right: Temperature distribution of burner is visible through flame. [8]

The success of a being could be evaluated based on how well it performs in its own environment and intelligence plays an important role in this success. Humans are not the only intelligent beings on this planet and the concept of intelligence should be open to include both human and non-human forms. Animals and machines could be intelligent

in their own way and should not be compared to humans' standards. This applies to the three pillars of intelligence. It is not appropriate to make a statement that a being is not or less intelligent because it could not perform a specific human task such as making a cup of tea. As a result, a more general definition of intelligence could be: "The variety of information-processing processes that collectively enable a being to autonomously pursue its survival". [1 p17]

Some brainless and single-cell organisms such as the paramecium and the sponge, do not have neurons. However, they can perform tasks like eating, moving away from aversive situation, reacting to environmental stimuli and even changing their own behavior after repeated simulation - a sign of intelligence. However, a being with its brain has two advantages compared to those brainless organisms. The first one is selective receiving and transmission of signal from and to distant parts of the body. The second one is adaptation using synaptic plasticity mechanisms. As a result, more complex bodies and better reaction and adaptation to a fast-changing environment are enabled. [1 p164]

2.3 Classical AI versus Modern AI

In classical AI, the top-down method in which knowledge-based or expert systems are the fundamental building blocks. There is an attempt to replicate the workings of a brain from outside using a rule-based approach such as logical `IF … THEN …` statement. [1 p31] This is based on the ability of the human brain to reason. If some facts are given, a human could decide a conclusion after making a reasoned assumption about those facts. [1 p24] For example, if it is December and the temperature is below 0 degree Celsius, then turn on the heater. This is how the first AI systems are built.

The classical approach follows this trend because of the desire to compare AI with human intelligence directly. This implies that the best one could only reach to the level of human intelligence and could not surpass it. [1 p31] As a result, the possibility of AI could have their own intelligence and outperform human intelligence is rejected. In addition, the classical technique is especially successful at solving well-defined tasks in which the whole set of clear and hard rules are created and computers could process them in a short time period. [1 p88] The CPU processing speed and large memory plays an

important role to support this. However, the classical AI approach becomes disadvantageous when slightly different situations happen and it needs to make decisions in these situations based on predefined rules. Being aware of unusual situations, comparing them with previous learned experiences and dealing with them is an important aspect of intelligence. This is a daily feature of intelligence in many animals. Therefore, modern AI has a better way of solving this problem by examining how the brain works in a fundamental way. [1 p89]

Recently modern AI has been focusing on a bottom-up approach which is putting together basic building blocks of intelligence and observing how the system could learn and develop over time. This is inspired by how a biological brain works such as how it learns, evolves and adapts over a period of time. [1 p88] The next step is to construct a simple model of fundamental elements of the brain. Lastly, these elements are built and simulated using a computer program or electronic circuits and as a result, an artificial brain or AI system could function in a brain-like fashion. In this way, modern AI is able to perform generalization which could be very difficult to achieve in classical AI.

2.4 Biological brain

Emotions, thoughts and behaviors are controlled by a complex network of cells called neurons which are the basic cells in a biological brain. [9 p163] There are about 100 billion of them in the typical human brain. They have diameters typically in the sizes of 2 to 30 micrometers. Complex network is formed with connected neurons and there are up to 10,000 connections between them. [1 p89] They are different from each other not only in terms of size, but also in which types of neurons they connect to and how strong the connections are. For example, sensory neurons receive and process signals from light, sound, etc. Meanwhile, motor neurons are specialized to send output signals to control muscles. There are also neurons that are associated with planning and reasoning. [9 p170] Even though they have considerably simple structures, the biological brain becomes a powerful tool when many of them form a complex network.

In general, every neuron has a cell body with a nucleus located at its center. A cell body receives input signals from other neurons through dendrites and sends output signals to other neurons through an axon. An axon consists of many branches and they connect to

the dendrites of other neurons, at a point called synapse. At the synapse, incoming electrical pulses cause the release of neurotransmitters. [9 p167] The structure of a biological neuron is illustrated in the figure 4.

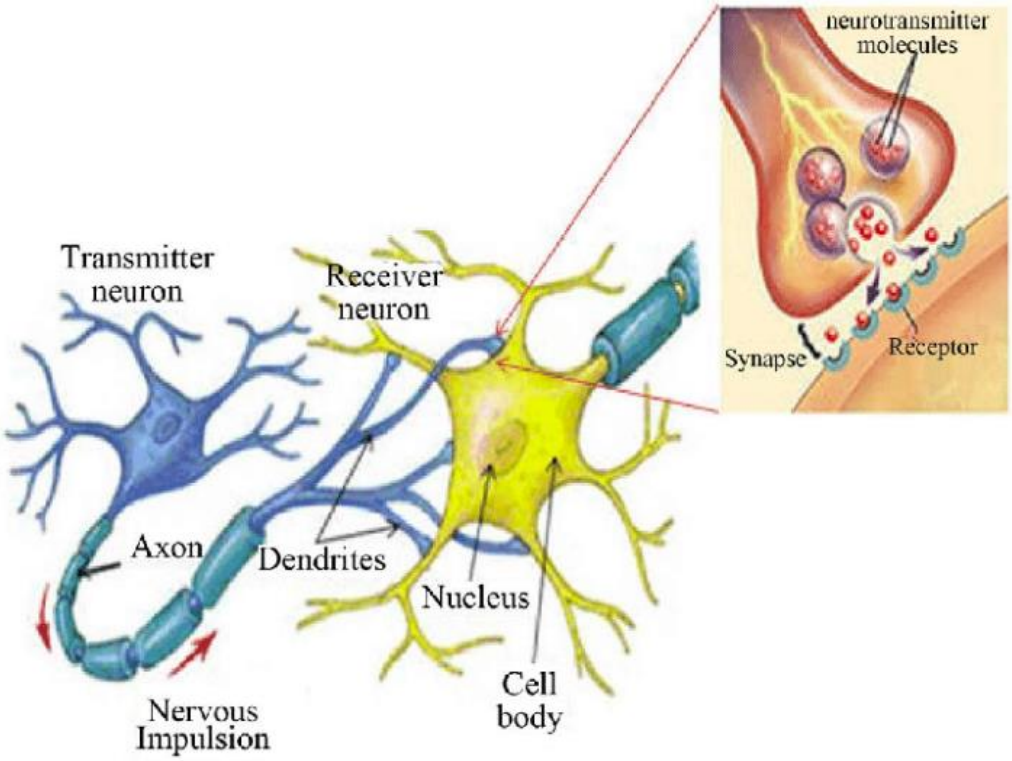

Figure 4. The structure of a biological neuron model [10]

Initially, a neuron is in an inactive state or resting state in which it does not receive or transmit any signal. In the active state, it receives input pulses which are electrical and chemical in nature called electro-chemical pulses along the dendrites from some of the connected neurons. The received pulses could change the potential voltage of the cell body. If the dendrites add to the cell potential voltage, they are called excitatory dendrites. On the other hand, the cell potential voltage is lower, these are called inhibitory dendrites. [1 p92]

At any time, if total potential electrical voltage received from the dendrites is above a certain threshold value, that particular cell body will in turn fire an electro-chemical pulse through its axon to those connected neurons. This is the mechanism how signals are

transmitted between neurons in a neuron network. After firing the pulse, the neuron will return to its inactive state and wait for the new incoming pulse again. [1 p90] However, the neuron does not fire its pulse in the case that threshold value is not reached. It is actually a binary process - the neuron either fires or it does not and therefore a Boolean logical function could be formed using this network of neurons. [9 p168]. The process is illustrated in figure 5.

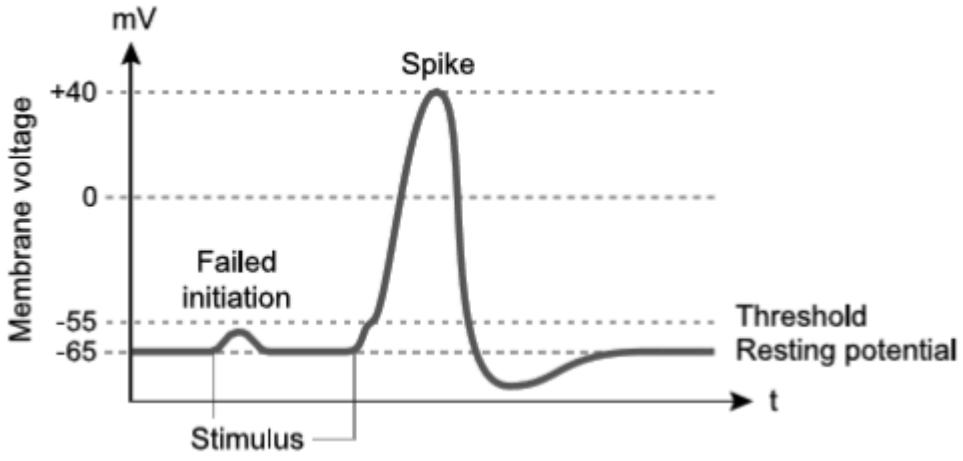

Figure 5. Resting state voltage, threshold voltage, failed stimulus and successful stimulus [9 p168]

The biological neural network is in fact much more complex than what is described in figure 5. It is observed that neurons could have different sizes; axons length also could vary from very long to very short; a neuron can connect to its neighboring neurons and they in turn could connect back to its dendrites; the connections can also be in different sizes and strengths. [1 p91]

This structure could be explained partly for genetic reasons and partly for development due to life experience. When an individual learns from his or her experience, the axon-dendrite connections or synaptic strength could either strengthen or weaken and in turn change the tendency of his or her reaction or behavior [9 p175]. As a result, a brain could learn, adapt and function differently depending on the patterns of signals it receives.

The idea of ANN is based on some of the characteristics of the biological brain. The aim is not to copy exactly the original structure of a biological brain, but to utilize some methods of operation in building the structure of the ANN. A typical ANN has approximately

below hundred neurons compared to 100 billion cells of a human brain. [1 p92] However, ANNs are powerful AI tools that are able to perform excellently in difficult tasks such as recognizing images, understanding speech, and identifying fraudulent activities in credit card usage. [1 p147] A basic artificial neuron model is introduced in section 2.5.

2.5 Artificial neuron model

The first task of building an ANN is to construct a simple model of an individual neuron that could be modeled into a computer program or simulated using an electronic circuit. The next step is to connect those individual neurons into larger neural networks to form an ANN. The typical and famous neuron model was invented by Warren McCulloch and Walter Pitts in 1943. [1 p93] The basic model of an artificial neuron is shown in figure 6.

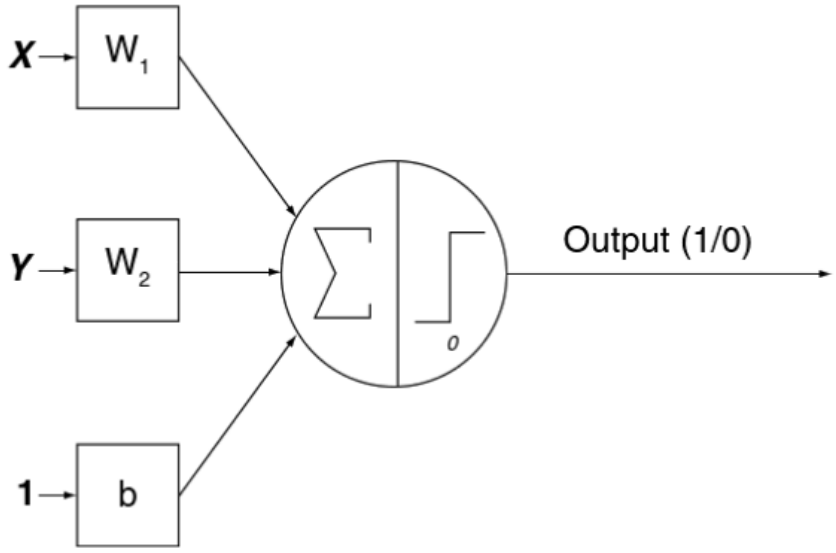

Figure 6. Basic model of an artificial neuron [1 p93]

The model stems from a branch of engineering named Neural Engineering which studies and reproduces the functionalities of the human brain in a bottom-up approach to engineer intelligent machines. Neural Engineering addresses problems such as learning algorithms, building high-level architectures that could create cognitive abilities, and implementing neural models in hardware. [9 p163] For comparison purposes, the similarity between an artificial neuron and a biological neuron is shown in figure 7.

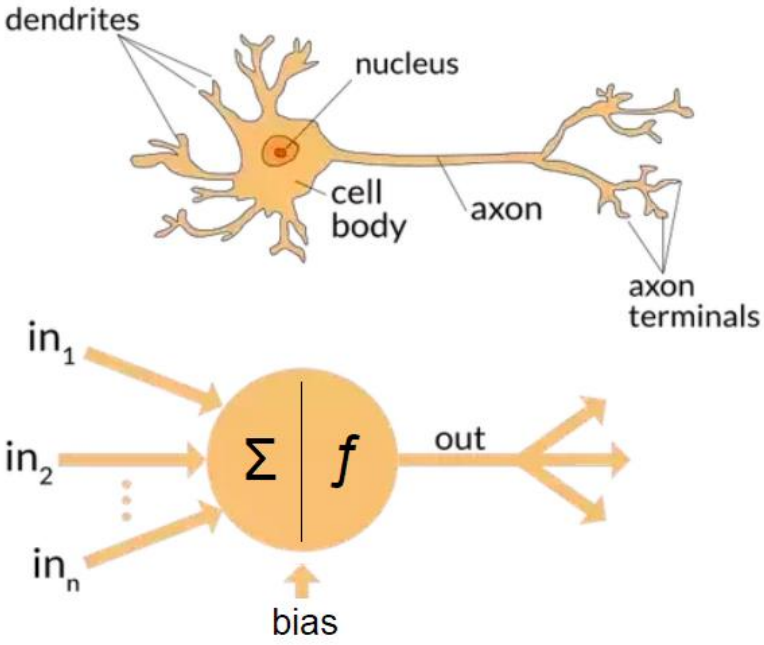

Figure 7. The similarity between an artificial neuron and a biological neuron [11]

In this model, the products of the inputs of x and y and their weights W_1 and W_2 are summed together and are expressed in formula 1. This sum is compared with the bias value b. If the value of the sum is equal or larger than the value of b, the neuron fires and gives the output of value 1. On the opposite, if the value of the sum is smaller than the value of b, the neuron does not fire and the output value is 0. The output then in turn becomes the input of the next connected neuron as mentioned in the biological brain model.

$$W_1 * x + W_2 * y \quad (1)$$

In this case, two inputs are used for illustration purposes and in reality, the number of inputs are not limited. [1 p93] This is the foundation for building a larger ANN using the basic building block of a simple artificial neuron model. The number of inputs is called the dimension or breadth of inputs.

2.6 Artificial Neural Networks

ANNs are computational models that simulate the adaptive and behavioral features of a biological network of neurons. An ANN is constructed using interconnected artificial neurons which is discussed in section 2.5. Input neurons receive information directly from the environment while output neurons interact with the environment. [9 p175] Other neurons, which communicate internally within the network, are called hidden or internal neurons as illustrated in figure 8, where each small circle represents an artificial neuron.

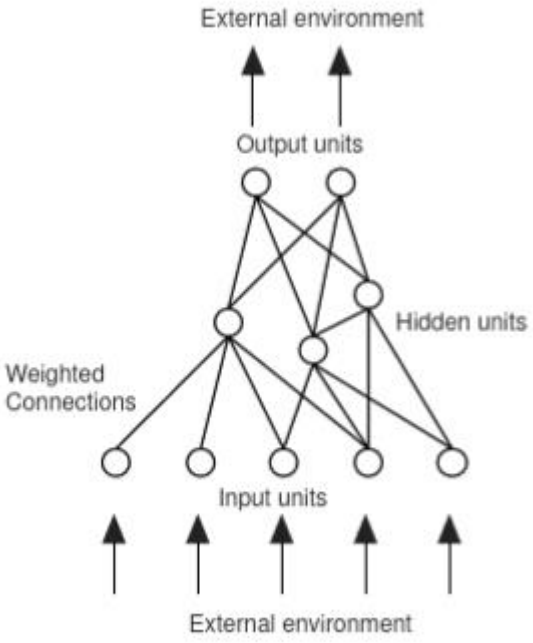

Figure 8. Generic neural network topology [9 p176]

When the neuron becomes active, it fires a signal to all neurons to which it connects. The weighted connections act like filters which either strengthen or weaken the input signals. This is also called synaptic strength which is similar to the biological model. Whereas biological neurons have either excitatory or inhibitory dendrites, artificial neurons simulate this feature by emitting positive and negative values to its connected neurons.

The output of an ANN to a stimulant input from the environment depends on its architecture and pattern of weighted connections or synaptic strength. The behavior and knowledge of the whole network is distributed across its connections; therefore, each

connection or neuron contributes its role accordingly. ANNs learn by modification of synaptic strengths when receiving input from the environment. [9 p176]

Technically, this is done through by introducing a loss function and performing back propagation to minimize this loss function. Normally, a set of inputs called a batch or smaller subsets of inputs called a mini batch is presented repeatedly to the ANN for a learning process. Each of the presentations is called one iteration. In each iteration, the loss function which is the difference in mathematical value between the correct answer—expected behavior—and the predicted answer—current behavior—is reduced by changing the weight values of the weight connections. As a result, learning is achieved by producing an answer which is closest to the expected answer.

2.6.1 Robustness, flexibility and content-based retrieval

ANNs are quite robust against many categories of signal degradation such as unit operation malfunction of the hardware implementations, quality of connection or signal noise. When the noise level increases, ANNs can distribute the errors uniformly across the input domain and maintain the correct response. Moreover, ANNs can be trained on mini-batch or single data incrementally to reduce the effect of noise or components' damage. [9 p176]

ANNs are not domain specific and could be applied to various types of problems. Due to this feature, ANNs are excellent in solving problems for which there is not any analytical solution. However, the tradeoff is even though a solution for such problems could be found, but effort of understanding them is given up. As a result, solving problems using ANNs might not strengthen our knowledge in a fundamental way. [9 p177]

Even in the case of an incomplete input data set or data corrupted by noise, ANNs can also retrieve memories by matching contents. It means that more familiar patterns are recognized easier and faster than those that are different or occur less frequently. This is different from conventional computer systems where data is obtained using the address of the memory cells. If the address number is corrupted, the entire memory is lost and therefore, data could not be retrieved. [9 p177]

2.6.2 Generalization

ANNs can provide the correct output to a data set which is not the training set. This means that it could act intelligently in a situation that it has never encountered before. This ability is due to the fact that ANNs could extract and store the invariant features of the training set in their weighted connections. How well an ANN could respond to a new data set depends on how similar the new data pattern can be described by the learnt invariant features. [9 p177]

Learning invariant features is also observed in the biological network of neurons. This allows them to deal with continuously changing environments. The capability to generalize is very important to problems in which it is costly or impossible to obtain all possible situations that the system is exposed to. [9 p177]

2.7 Difference between Artificial Intelligence, Machine Learning and Data Science

What is the difference between ML and AI? How does Data Science (DS) relate to AI? These are common questions when one is first introduced to those terminologies. Even though these three fields have similarities, they are not interchangeable. Those concepts could be used as fashionable words for marketing purposes, but most professionals working in these fields have intuitive understanding on how to categorize whether a certain matter belongs to AI, ML or DS. The simplified explanation about the difference between these three fields could be described as DS provides tools and insights, ML uses tools to make predictions, and AI produces actions. [12]

2.7.1 Data Science produces insights and Machine Learning produces predictions

The goal of DS is to gain insight and understanding, especially for humans. This can be used to distinguish from other two fields. The main point is that there is always a human in the process which performs raw data processing, puts structured data into a graph for visualization, and understands the insight. The classic Data science utilizes the combination of statistics, computer science and domain expertise and focuses on these tasks: domain knowledge, experimental design, statistical inference, data visualization and communication. [12]

If a certain problem could be summarized as the statement: “Given X with particular features, predict Y about it.” [12] To be more specific, given some training data, how ML can produce predictions or make conclusions about a new set of data. The prediction could be about a time series such as weather and stock market forecasts; or they could discover data patterns which are not easily detected.

On the other hand, there is also overlap between DS and ML. For example, logistic regression algorithms could be used to predict in ML or gain insights in DS. To be more specific, the following statement is the product of logistic regression in DS to understand customers better: *It is more likely that a customer with high income will purchase our product, should we change our marketing strategy?* Meanwhile, a statement like “This customer has approximately 56% chance of buying our new product, should it be recommended to him?” could be a prediction statement for a ML work. [12]

2.7.2 Artificial Intelligence produces actions

Historically, ML is considered as a subfield of AI. For example, computer vision was classified as an AI problem rather than ML problem. The relationship between AI, ML and Deep Learning (DL) is illustrated in figure 9.

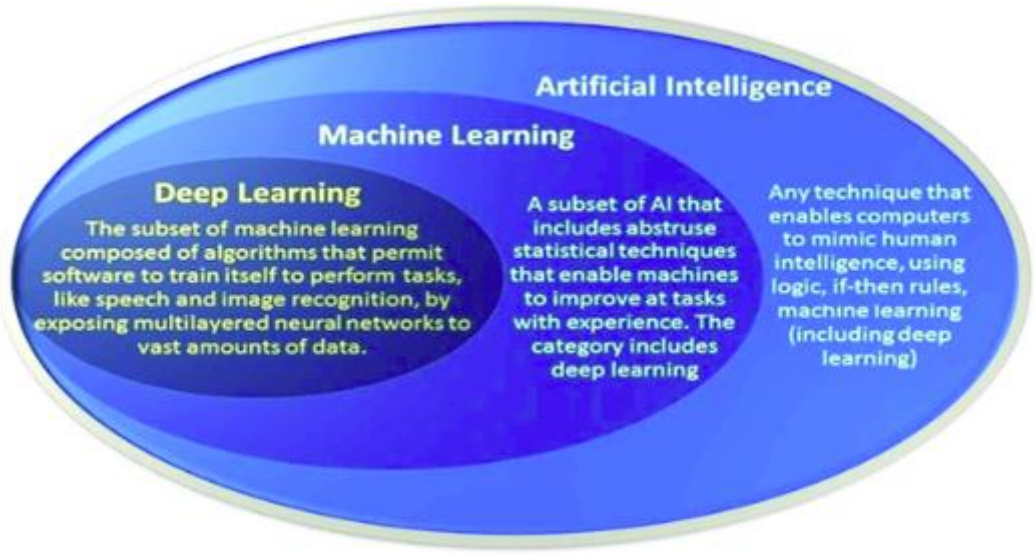

Figure 9. Generic neural network architecture [13]

AI is the oldest and the most widely recognized of these three fields. However, it is the most challenging to define due to extravagant publicity or promotion to get the attention, funding and investment by journalists, researchers and startups. AI is normally misinterpreted as general AI which is the capability to perform tasks from different domains. AI is also wrongly perceived as super intelligent AI, which outperforms human intelligence. This creates unrealistic expectations for an AI system. [12]

The definition of AI that is commonly accepted, for example by Poole, Mackworth and Goebel 1998 and Russell and Norvig 2003, is an agent that executes or recommends actions. [12] The typical AI system includes Reinforcement Learning (RL), Natural Language Processing (NLP), game-playing algorithms such as Deep Blue and AlphaGo, optimization solutions, for example Google Maps recommend an optimized route, Robotics and control theory.

2.8 Data normalization

Data normalization is a common and important technique for raw data processing before applying to ML or ANN models. The outcome of data normalization is to make values of different features in a data set into a common range without changing the internal relationships of data points in each feature. It is especially important when features have ranges which are much further away from each other. [14]

For example, a data set contains two features which are age and income. Age can be in the range from 0 to 100, while income values could be in the range from 0 to 100,000. If data normalization is not performed, the income feature could influence the output of a model due to its larger value. Therefore, the model could be a wrong predictor in a case where age is the true influential factor. [14] An experiment is conducted using normalized data and non-normalized data on the same deep neural network model.

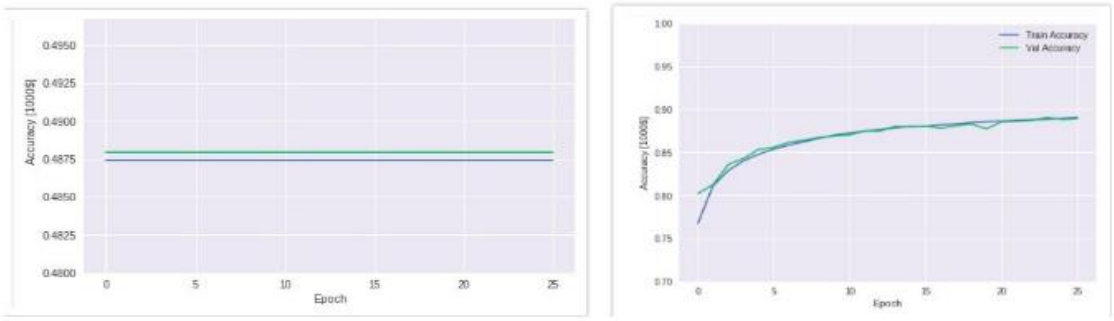

Figure 10. Left: Accuracy graph without normalized data, Right: Accuracy graph with normalized data [14]

As seen in figure 10, the accuracy without normalized data is only around 48.8% while accuracy of the model with normalized data could reach 88.93% on the validation data set. The straight line of the graph on the left can be explained that the accuracy remains unchanged and the model could not learn within 26 iteration steps. This implies that using non-normalized data could lead to long learning time because gradient descent algorithms can oscillate back and forth and are not able to find local or global minimums and therefore do not converge quickly. [14]

3 Project specifications

This chapter describes in detail the project specifications. Wizense Oy demands a study of the possibility of AI in categorizing and labeling route activities. The actual raw obtained from wearable devices is in the form of quaternions with the time stamps. After that, the quaternions are converted into 2 dimensional XY coordinates. Consequently, these XY coordinates are fed into an ANN model in order to train this model. After the model is trained, it can categorize whether a certain point is in or out of the predefined route.

The simplified version of the Metropolia Myyrmäki building layout and L-shaped predefined route is shown in figure 11. Black rectangles represent rooms in the building and yellow color rectangles represent predefined routes. Red rectangles are doors of the building. In this study, 3 predefined routes with increasing complexity will be examined to explore the ability to learn of the ANN.

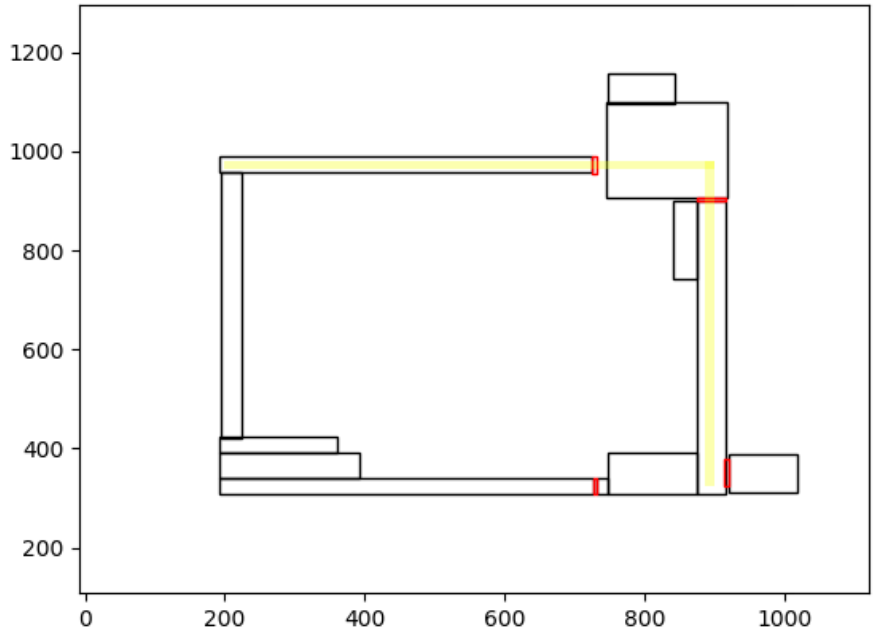

Figure 11. The simplified version of Metropolia Myyrmäki building layout with predefined route

In real life situations, there will be an experienced cleaner or worker who will wear a wearable device which generates step points and data is collected and converted to XY-coordinates. This experienced cleaner or worker performs his work following the optimal

route which will be the predefined route. As a result, all the step points collected by this person will be labeled as “in route” which is also Boolean value True or 1. Other parts of the room which are not covered by his route will be labeled as “out route” which also has Boolean value False or 0.

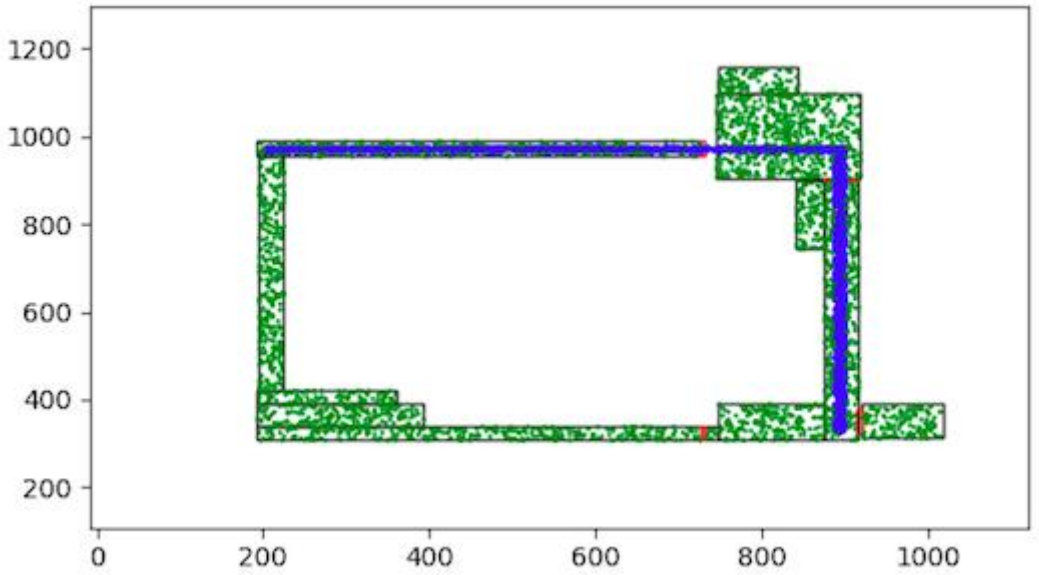

Figure 12. The simplified Metropolia Myyrmäki building layout with predefined route

Due to the inconvenience of walking on the real map and collecting large enough data for the neural network to learn, a random generated function will be used to generate steps data in XY coordinates and label them. The route data points are in blue color—Boolean logic 1— and non-route data points are in green color—Boolean logic 0—as illustrated in figure 12.

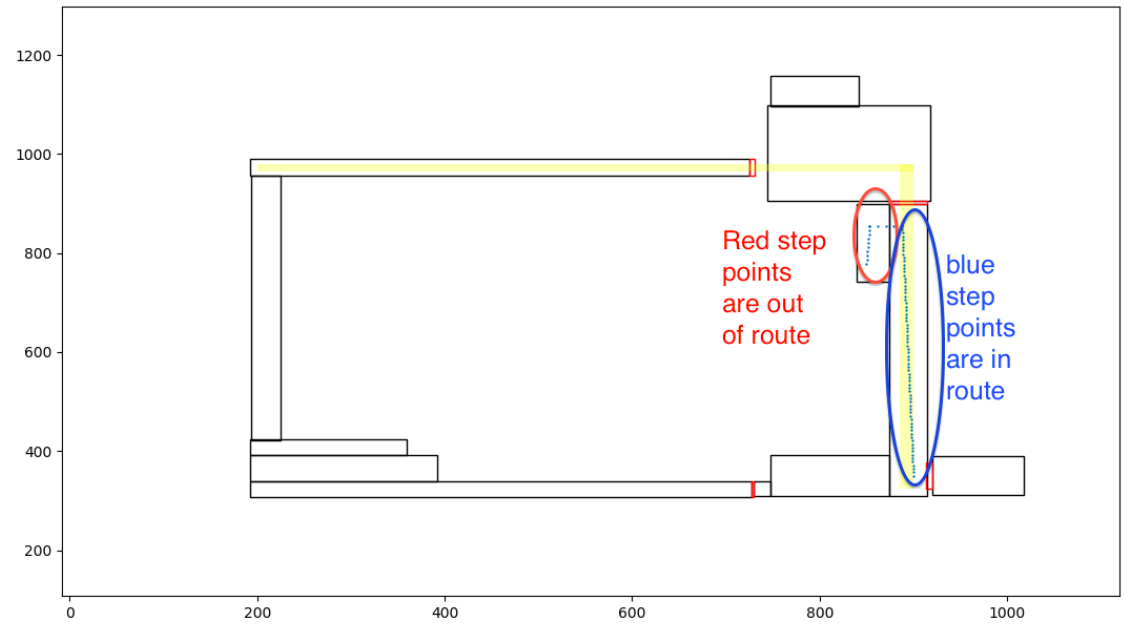

Figure 13. Expected outcomes from ANN model

A neural network will learn from these data points and with given input as step points of the new route, it will label which part of the new route is in or out of the predefined route as follows. The expected outcome of the ANN application is demonstrated in figure 13.

4 Implementation

4.1 Development process

The Waterfall model was adopted for the project development process. This is the traditional and predictive software development process in which the requirements are fully understood and not changing since any change in the requirements would make the completed tasks useless and extend the project deadline further. This is especially true for an individual short-term project such as this project which lasts only approximately 3 months. [15]

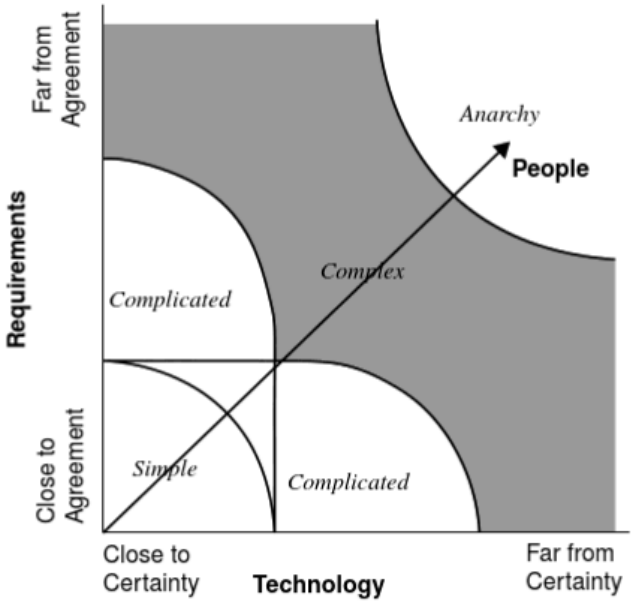

Figure 14 The Stacey Graph [15 p11]

The Waterfall model also requires the technology to be reliable and stable. Since the project is short in timespan in nature, technology changes are not expected to occur. Furthermore, the libraries and modules used in this project have a long historical record to be stable and also adopted by large companies such as Google. In brief, if the development process is certain and predictive, the Waterfall model is a suitable choice. Otherwise, the Agile model is a better choice. The Stacey Graph, which is a useful tool to assist in deciding which model to use for the development process, is described in figure 14.

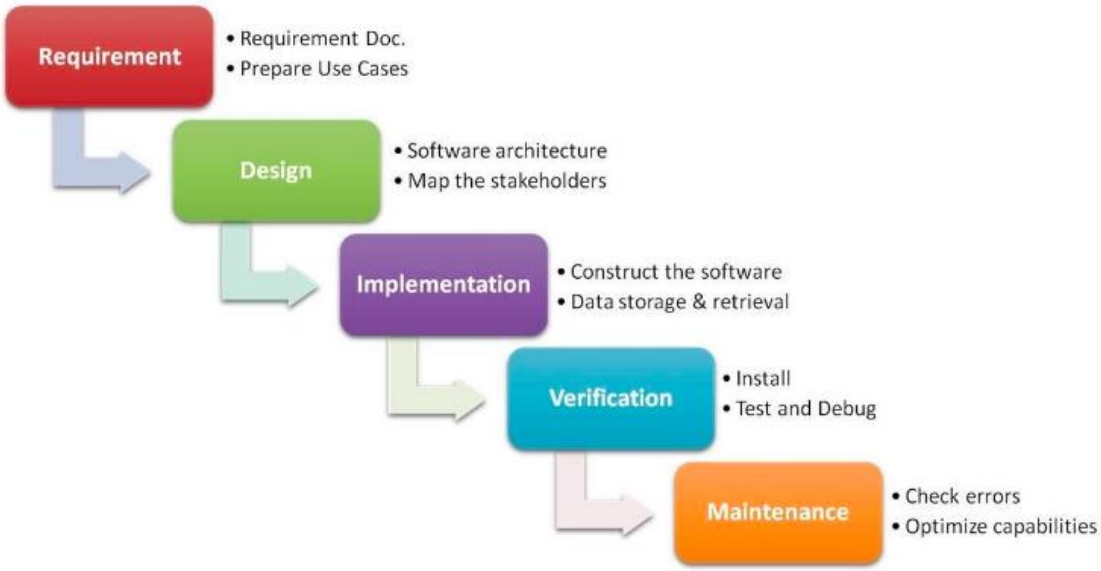

Figure 15 The Waterfall model [16]

The traditional Waterfall model contains 5 stages which includes requirement, design, implementation, verification and maintenance. [16] The model is described in figure 15. The requirement stage takes the first two weeks of the development process where project requirements and specifications were discussed and completely understood. The design phase takes two weeks in which different ANN models are studied and the most suitable one is selected. Dataflow and software architecture are also decided in this phase. The implementation phase takes approximately 2.5 months. The verification and maintenance take another two weeks.

4.2 Structure, tools and system used

The overall flowchart of the project is described in figure 16. Firstly, the CSV files which contain information about the dimensions such as XY coordinates, widths and heights of the rooms, doors and predefined routes is fed into a python program called dataGenerator.py. This Python program uses two Python modules called Numpy and Pandas for

data frames processing and matrix operation. Source code of the application is available on github[2] under GPL license.

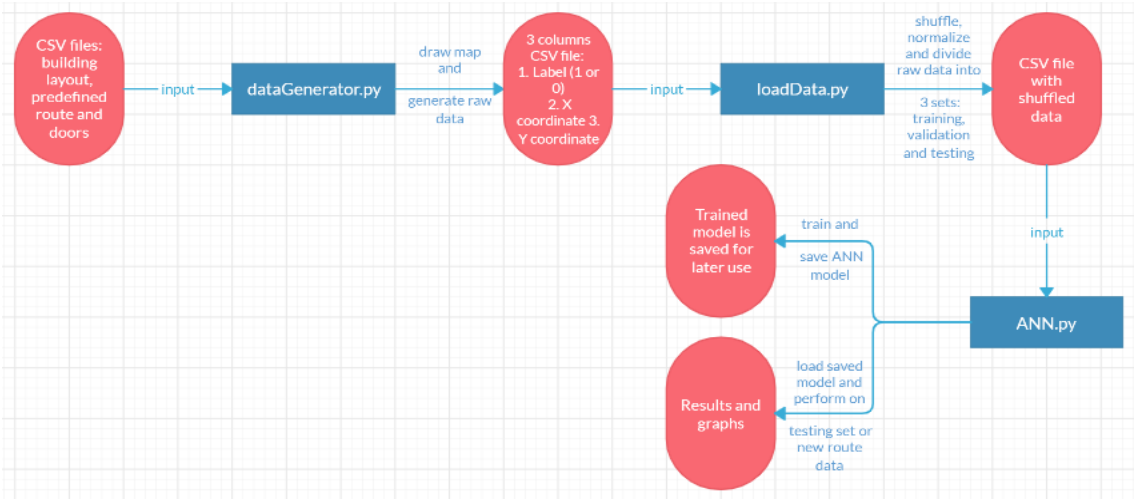

Figure 16. Application flow chart

The output of dataGenertor.py is the map of the building layout, rooms and predefined routes as shown in Figure 11. A few examples of rooms' dimensions are described in listing 1.

```
x,y,w,h
920,312,98,77
874,309,41,591
747,309,127,82
```

Listing 1. A few examples of rooms' dimensions in room.csv file

The dataGenerator.py also contains an algorithm to randomly generate raw data points in a CSV file. Each of the data points has three dimensions. The first dimension is that if a point is within the predefined route—yellow color area in figure 11—it has a value of 1 and has value of 0 otherwise. The second is the coordinate along the horizontal axis or X-axis. The third dimension is the coordinate along the vertical axis or Y-axis. The first few data samples are shown in listing 2. For example, the first data point has a label value of 1, X-coordinate is 896.93, and Y-coordinate is 561.47.

```
label,x,y
1,896.9340830129474,561.4713114655046
1,895.0969434356209,832.1046141747404
1,888.6445021660173,826.0015403966145
```

[2] https://github.com/vinhxu/BachelorThesis

Listing 2. First few data samples with label, X-coordinate, and Y-coordinate

Consequently, the raw data file acts as an input into the loadData.py program. The loadData.py has 3 main functions. The first function is to shuffle the raw data into random order. Since the training process for the ANN model is done in a mini batch data set, the shuffling prevents the mini batch data set from containing only samples from only one label. Intuitively, one aspect of a human mind is a tool to separate and differentiate. A human could not learn about the concept of darkness without the existence of the opposite concept which is light. Therefore, distinguishing and comparing is one of the ways to learn. In the same manner, if there are only data points which belong to the predefined route, an ANN model could not learn how to categorize them from non-route data points. Therefore, data shuffling is necessary in this case. After shuffling, the shuffled data is saved into a CSV file and is used as an input for ANN in ANN.py file.

The second function is to normalize the data into smaller ranges. As seen from listing 2, values of X and Y coordinates are quite large compared to the values of labels. To be more specific, the range of XY-coordinates are in the range of hundreds to thousands while the label is in the range from 0 to 1. There can be a large difference between the ranges of X and Y coordinates in the data set. To make the application useful in general cases, data normalization is necessary. As discussed in section 2.8, data normalization helps to improve the accuracy of an ANN model, decrease learning time and enhance data visualization as well. The MinMaxScaler function imported from sklearn.preprocessing library will be used for data normalization. This function uses an algorithm to transform data into a specific range which is similar to zero mean and unit variance scaling. The first few data samples are described in listing 3.

```
[[0.         0.74413307 0.9921823 ]
 [0.         0.85845287 0.75811799]
 [1.         0.39525113 0.78352972]
 ...
```

Listing 3. Normalized data in range from 0 to 1

The third function is to divide the whole data set into 3 smaller subsets which is training, validation and testing set. The main purpose is to detect the overfitting from the learning process and retrain the ANN model if this occurs. Finally, an ANN is built to learn from the processed data in the program file ANN.py. Numpy will be used again for matrix and

array operations and Pandas is used for data processing. In addition, Tensorflow open source library provides an excellent tool for ML or Deep Learning applications such as ANNs. Tensorflow library was developed by Google for both research and production. Lastly, matplotlib library is used for data visualization and drawing graphs. Time library is used to keep track of the learning time of ANN.

In ANN.py, hyper parameters which influence the performance of the ANN model are declared and initialized. Consequently, ANN topology is constructed and its parameters are initialized. After that, data from the training set is divided into smaller batches and fed into this ANN model. After a predefined number of iterations which are called epochs, ANN model is trained with high accuracy and small loss value. This also indicates convergence in which these values fluctuate in a very small range when the number of iterations increases. Lastly, trained models are saved into the 03_savedModel folder.These models could be restored to perform new route data points in the later stage. The results such as accuracy, loss on training, validation and testing data set are presented to the user for visualization purposes. During the training, accuracy and loss from both validation and training data is also printed to monitor the training progress and its convergence.

4.3 Implementation in detail

4.3.1 Generating raw data

Firstly, the information regarding dimensions of the rooms, doors and predefined routes are required so that the data generating function could generate the points and correctly label these points to their categories: route which has a value `1` or non-route which has value of 0. This is done by using Pandas `pd.read_csv(path_to_csv_file)` function. Furthermore, the vertices, which guide the path of the new route, are also defined in listing 4.

```
7   vertices = np.array([(900, 350), (888, 855), (853, 855), (850, 779)])
8
9   roomsDimension = pd.read_csv('./01_rawData/room.csv')
10  doorsDimension = pd.read_csv('./01_rawData/door.csv')
11  routesDimension = pd.read_csv('./01_rawData/route3.csv')
```

Listing 4. Defining vertices of new route and reading dimensions of rooms, routes and doors from CSV file.

The call of the plot function to draw all the graphs is shown in listing 5. The X-axis and Y-axis margins for the graph are set at the minimum and maximum values of the rooms' dimensions plus the offset value of 200.

```
14  plt.figure()
15  ax = plt.gca()
16
17  xlim_offset = 200
18  ylim_offset = 200
19  offset = 200
20
21  ax.axes.set_xlim([roomsDimension.min().values[0]-offset,roomsDimension.max().values[0]+offset])
22  ax.axes.set_ylim([roomsDimension.min().values[1]-offset,roomsDimension.max().values[1]+offset])
```

Listing 5. Set the X-axis and Y-axis limits

The function `draw_map` on lines 31 to 34 of listing 6 is to draw the simplified version of Metropolia Myyrmäki building layout with predefined route as shown in figure 11. Likewise, function `draw_walking_path` in listing 6 will generate the data points based on the defined vertices as a guide path and draw those data points as a walking path of the new route. The `stepDistance` parameter adjusts how big the step size is. The variable `margin` adds some noise to the data points so that they appear more natural compared to natural walking steps.

```
27  def draw_rectangle(rectangleDimensions, alpha=1, facecolor='none', edgecolor='black'):
28      for i in rectangleDimensions.values:
29          ax.add_patch(Rectangle((i[0], i[1]), i[2], i[3], alpha=alpha, facecolor=facecolor, edgecolor=edgecolor))
30
31  def draw_map():
32      draw_rectangle(roomsDimension)
33      draw_rectangle(doorsDimension,edgecolor='red')
34      draw_rectangle(routesDimension,alpha=0.3, edgecolor='none',facecolor='yellow')
35
36  def draw_walking_path(vertices, stepDistance, plot=plt):
37      angleCoefficiences = np.array([])
38      distances  = np.array([])
39      dividers   = np.array([])
40      xRange = np.array([])
41      yRange = np.array([])
42
43      for i in range(len(vertices)-1):
44          k = (float(vertices[i+1][1]) - vertices[i][1]) / (vertices[i+1][0] - vertices[i][0])
45          angleCoefficiences = np.append(angleCoefficiences, k)
46
47          d = abs(math.sqrt(1+k**2) * (vertices[i+1][0] - vertices[i][0]))
48          distances = np.append(distances, d)
49
50          dd = math.floor(float(d)/stepDistance)
51          dividers = np.append(dividers, dd)
52
53          xr = np.linspace(vertices[i][0],vertices[i+1][0],dd)
54          xRange = np.append(xRange, xr)
55
56          yr = np.linspace(vertices[i][1],vertices[i+1][1],dd)
57          yRange = np.append(yRange, yr)
58
59      # margin controls how "noisy" you want your fit to be.
60      margin = 0.11
61      noise  = margin*(np.random.random(len(yRange))-0.5)
62      yRange = yRange + noise
63      xRange = xRange + noise
64      # ax = plot.gca()
65      ax.plot(xRange, yRange, marker="o", ls="", ms=.7)
66      generate_walkingPath_xy(xRange,yRange)
67
68      return xRange, yRange
```

Listing 6. Functions that draw rooms, predefined routes and walking path of the new route

The function `isOnRoute` in listing 7 compares the X and Y coordinates of a data point with the coordinates of the predefined route and outputs the Boolean value of True or False accordingly. The function `calculate_total_area` calculates an area of a rectangle given its dimensions. The function `get_random_xy` from listing 7 takes dimensions of a rectangle and generates the desired number of data points in that rectangle. This function is to facilitate the control of how many data points in total as well as for each category in the data set. How the total number of samples and the ratio of data points in each label affect the training time, the accuracy of the ANN model will be discussed in the chapter 5.

```
129  def isOnRoute(routesDimension, xCoordinate, yCoordinate):
130      isOnRoute = np.array([], dtype=bool)
131      for i in routesDimension.values:
132          if(xCoordinate>=i[0] and xCoordinate<=(i[0]+i[2]) and yCoordinate>=i[1] and yCoordinate<=(i[1]+i[3])):
133              isOnRoute = np.append(isOnRoute, True)
134          else:
135              isOnRoute = np.append(isOnRoute, False)
136          finalCheck = False
137          for i in isOnRoute:
138              finalCheck = (finalCheck or i)
139      return finalCheck
140
141  def calculate_total_area(dimension):
142      totalArea = 0
143      for i in dimension.values:
144          totalArea +=  i[2] * i[3]
145      return totalArea
146
147
148  def get_random_xy(dimension, totalNumberOfSamples):
149      randomX = np.array([])
150      randomY = np.array([])
151      for i in dimension.values:
152          area = float(i[2] * i[3])
153          # print ('area: ' + str(area))
154          ratio = area/calculate_total_area(dimension)
155          # print ('ratio: ' + str(ratio))
156          numberOfSamples = int(ratio*totalNumberOfSamples)
157          # print(' ,numberOfSample: '+str(numberOfSamples))
158          randomX = np.append(randomX, np.random.uniform(i[0], i[0] + i[2], numberOfSamples))
159          randomY = np.append(randomY, np.random.uniform(i[1], i[1] + i[3], numberOfSamples))
160          # draw_points_xy(randomX,randomY)
161      return randomX, randomY
```

Listing 7. Functions which support the data generation process

Function `generate_route_xy` and `generate_nonRoute_xy` in listing 8 generate a desired number of route and non-route data points accordingly by calling the function `get_random_xy` from listing 7. Function `generate_map_xy` in listing 8 has an algorithm to balance the data from both categories. Firstly, the area of both route and non-route is calculated and the ratio between them. Based on this information combined with the parameterized total number of samples, the actual number of samples for both the labels is calculated using the formulas shown in line 186 and 187 of the listing 8.

```python
def generate_route_xy(numberOfSamples):
    routeX, routeY = get_random_xy(routesDimension, numberOfSamples)
    return routeX, routeY


def generate_nonRoute_xy(numberOfSamples):
    nonRouteX, nonRouteY = get_random_xy(roomsDimension, numberOfSamples)
    indexArray = np.array([])

    for i, x in enumerate(nonRouteX):
        if(isOnRoute(routesDimension, nonRouteX[i], nonRouteY[i])):
            indexArray = np.append(indexArray, i)
    nonRouteX = np.delete(nonRouteX, indexArray)
    nonRouteX = np.append(nonRouteX,[roomsDimension.min().values[0], roomsDimension.max().values[0]])
    nonRouteY = np.delete(nonRouteY, indexArray)
    nonRouteY = np.append(nonRouteY,[roomsDimension.min().values[1], roomsDimension.max().values[1]])

    return nonRouteX, nonRouteY

def generate_map_xy(totalNumberOfSamples):
    roomArea  = calculate_total_area(roomsDimension)
    routeArea = calculate_total_area(routesDimension)
    ratio     = float(routeArea)/roomArea
    routeSamples      = int(totalNumberOfSamples/2.0)
    nonRouteSamples   = int((1/(2.0*(1-ratio)))*totalNumberOfSamples)

    rx, ry  = generate_route_xy(routeSamples)
    nrx,nry = generate_nonRoute_xy(nonRouteSamples)

    draw_points_xy(rx,ry,color='blue')
    draw_points_xy(nrx,nry,color='green')

    route    = pd.DataFrame({'x': rx,    'y': ry,    'label':  1})
    nonRoute = pd.DataFrame({'x': nrx,   'y': nry,   'label':  0})
    data = pd.concat([route,nonRoute])

    data.to_csv('./01_rawData/rbRoute3Data_30k.csv', index=False)
```

Listing 8. Functions to generate data points

As a result, we could get the balanced data for the ANN model. The reason for balancing the data is because if the total number of data points which belongs to one category is much more than the others, it will affect the result of the ANN model. This point will be discussed further in chapter 5. The function `generate_map_xy` also produces raw data in a CSV file and will be processed in later stages.

4.3.2 Processing raw data

The processing raw data task resides in the file loadData.py. The function `shuffle_csv_data` which creates randomness in the data set for training, validation and testing purposes is shown in listing 9. The shuffling task is done by call methods `sample(frac=1)` and `reset_index(drop=True)` of the pandas library.

```
7   def shuffle_csv_data(path = './01_rawData/rbRoute2Data_10k.csv', after_shuffled_file_name = './02_processedData/pbRoute2Data_10k.csv'):
8
9       data = pd.read_csv(path)
10      #  Shuffle data using sample and reset to new index
11      data = data.sample(frac=1).reset_index(drop=True)
12
13      data.to_csv(after_shuffled_file_name, index=False)
14      return after_shuffled_file_name
```

Listing 9. Function to shuffle data into random order

Data normalization is performed using a wrapper function `scaler_min_max` that calls `MinMaxScaler` of `sklearn.preprocessing` library. The function returns processed data as well as the scaler. This scaler will be used to scale the new route data points for verification purposes. The implementation of the function `scaler_min_max` is described in listing 10.

```
19  def scaler_min_max(data,feature_range=(0,1)):
20      scaler = MinMaxScaler(feature_range= feature_range)
21      scaler.fit(data)
22      data = scaler.transform(data)
23      return data, scaler
```

Listing 10. Function to normalize data into range from 0 to 1

The `convert_to_oneHot` function is implemented in listing 11. In short, one hot encoding is a data transformation of categorical variables as binary vectors. This is done in two steps. Firstly, values of the categorical variables are represented in integer values. The second step is each integer value is converted into a binary vector that is all zeros except at the index of the integer which is 1. For example, given a data set “route, route, non-route”, after the first step, the following result is obtained: “1,1,0”. After the second step, we achieve the one hot representation of the original data set: “[0,1], [0,1], [1,0]”. The purpose of one hot encoding is to give probability prediction to each of the categories. For example, a point could be represented as [0.3, 0.7] which means that this point has a 30% chance of being out of the predefined route and 70% chance of being in a predefined route.

```
26  def convert_to_oneHot(original_array):
27      # return original_array
28      nb_classes = int(original_array.max()+1)
29      original_array.reshape(-1)
30      converted_array = np.eye(nb_classes)[original_array.astype(int)]
31      return converted_array
```

Listing 11. Function to convert to one Hot

As seen from listing 12, the `load_data` function which returns the correct set of data depending on the choice of mode. There are two modes that are important for an ANN model. The first mode is train mode in which both the training and validation data subsets are put into an ANN model. The reason for putting both these subsets is to prevent over-fitting issues and also help to monitor the convergence of the training process. The second mode is test mode in which the testing data subset is put into a trained ANN model. If the accuracy is high and fairly comparable to those obtained by training and validation data subsets, it is quite confident to conclude that the model is well trained and does not have both underfitting or overfitting problems.

```
33  def load_data(mode='train', oneHot=True):
34      """
35      Function to load correct data set
36      :param mode: train or test
37      :return: data set with XY-coordinates and correct labels
38      """
39
40
41      if mode == 'train':
42          x_train, y_train, x_valid, y_valid = points_train_coordinates, points_train_labels, \
43                                               points_valid_coordinates, points_valid_labels
44          if (oneHot==True):
45              y_train = convert_to_oneHot(y_train)
46              y_valid = convert_to_oneHot(y_valid)
47
48          return x_train, y_train, x_valid, y_valid
49
50      elif mode == 'test':
51          x_test, y_test = points_test_coordinates, points_test_labels
52          if (oneHot==True):
53              y_test = convert_to_oneHot(y_test)
54
55          return x_test, y_test
```

Listing 12. Function to load data to ANN model

As depicted in listing 13, the actual implementation of dividing original data set into smaller subsets of training, validation and testing set with the ratio of 79%, 7% and 14% accordingly. This is done by calling numpy function `np.floor` to determine the correct index giving the ration and then calling numpy method np.arange to put the values into correct order. There is no golden rule on how to divide a data set into these 3 subsets. Typically, a ratio of 80/10/10 or 70/15/15 is used for data set division. In this application, the training set is 79%, validation set is 7% and testing set is 14%.

The final step is to separate the XY-coordinates data and labels because only the XY-coordinates are put into the ANN model. Then outputs of the ANN model are compared

to the correct labels to make necessary adjustments to the weight connections. This process helps the ANN to produce better predictions. This separation is done using the code from line 75 to 80 in listing 13.

```
56  data = pd.read_csv('./02_processedData/pbRoute2Data_3k.csv').values
57  data, scaler = scaler_min_max(data, feature_range = (0,1))
58
59  # Train, Validation and Test data, Train: 79%, Validation 7%, Test: 14%
60  # Train: 79%
61  train_start = 0
62  train_end   = int(np.floor(0.79*len(data)))
63  data_train  = data[np.arange(train_start, train_end), :]
64  # Validation 7%
65  valid_start = train_end + 1
66  valid_end   = int(np.floor((0.79+0.07)*len(data)))
67  data_valid  = data[np.arange(valid_start, valid_end), :]
68  # Test: 14%
69  test_start  = valid_end + 1
70  test_end    = len(data)
71  data_test   = data[np.arange(test_start, test_end), :]
72
73
74  # # Build input (x,y) and output (label)
75  points_train_coordinates  = data_train[:, 1:]
76  points_train_labels       = data_train[:, 0]
77  points_valid_coordinates  = data_valid[:, 1:]
78  points_valid_labels       = data_valid[:, 0]
79  points_test_coordinates   = data_test[:, 1:]
80  points_test_labels        = data_test[:, 0]
```

Listing 13. Source code to divide data into 3 subsets: training, validation and testing

After the ANN model is trained, it is helpful to verify how well the trained model works on new data. Therefore, as mentioned in the requirement, new route data points are created using vertices as a guideline. Since the ANN model is trained with the use of the scaler, the same scaler needs to be applied to new data as well.

```
84  walkingPathData = pd.read_csv('./01_rawData/walkingPathData.csv').values
85
86  walkingPathData = np.append(walkingPathData, [[0, roomsDimension.min().values[0], roomsDimension.min().values[1]]], axis=0)
87  walkingPathData = np.append(walkingPathData, [[0, roomsDimension.max().values[0], roomsDimension.max().values[1]]], axis=0)
88
89  walkingPathData = scaler.transform(walkingPathData)
90
91
```

Listing 14. Source code to read new route data points and normalize them

The transformation from raw data into normalized data is implemented in line 89 of listing 14 by using the `scaler_min_max` function described in listing

10. Furthermore, 2 extreme points at the corner of the rooms so that the graph will have an auto scale feature as seen from line 86 and 87 of listing 14.

4.3.3 Building an ANN

Firstly, all data subsets are loaded into the ANN.py file as train mode and test mode as mentioned in subsection 5.2.2 . This is done using line 12 and 13 from listing 15. Variable `n_inputs` is the dimension of the inputs into the ANN model. In this case, the ANN model is trained with X coordinate and Y coordinate, hence variable `n_inputs` is equal to 2. Variable `n_classes` is the dimension of the outputs of the trained ANN model. Since there are two categories, which are route and non-route encoded into one hot encoding as shown in listing 11, `n_classes` variable takes the value of 2. The initialization of the two variables are shown in line 15 and 16 from listing 15.

Hyper-parameters are parameters that are important to the training process. These parameters can be adjusted to improve the accuracy and training time which are the most important aspects of an ANN model. These parameters include epochs, batch size, display frequency and learning rate from listing 15. Variable `epochs` is the total number of iterations. Variable `batch_size` is the random selected subset of the training data to speed up the training process. Display frequency allows monitoring the accuracy over the training process and also helps detect the convergence. Lastly, learning rate controls the magnitude of weights adjustments in each iteration step to give better predictions.

```
11  # Load data
12  x_train, y_train, x_valid, y_valid = load_data(mode="train")
13  x_test, y_test = load_data(mode='test')
14
15  n_inputs = 2
16  n_classes = 2
17
18  # Hyper-parameters
19  epochs = 50000          # Total number of training epochs
20  batch_size = 1000       # Training batch size
21  display_freq = 100      # Frequency of displaying the training results
22  learning_rate = 0.001   # initial learning rate
```

Listing 15. Load data set, declare input and output dimensions and set hyper-parameters

Function `fc_layer` in listing 16 creates a fully connected layer of ANN neurons given these parameters: the matrix from the previous layer denoted by x, number of neurons on this layer denoted by `num_units`, name of this layer denoted by `name`, and the variable `use_relu` which take Boolean value of 1 as a default. If the `use_relu` has value True, the layer of ANN neurons will use the Rectified Linear activation function. Otherwise, when it has value False, sigmoid activation function is used. Sigmoid and Rectified Linear Units (ReLu) are the most common activation functions that have been used regularly in building an ANN. [17] The Sigmoid function is in the form of formula 2 and produces the S-shaped curve. The ReLu function is a combination of two straight lines and expressed as formula 3.

$$S(x) = (1 - \exp(-2x)) / (1 + \exp(-2x)) \quad (2)$$

$$R(x) = \max(0,x) \text{ and if } x < 0, R(x) = 0, \text{ otherwise if } x >= 0, R(x) = x \quad (3)$$

`Weight_variable` function in listing 16 takes the dimension or shape of the matrix and returns the initialized weight matrix. Initialization process uses the `tf.random_normal_initializer` function of the tensorflow library to randomize the weights to small numbers which is close to zero. In the same manner, `bias_variable` function takes the shape of the required matrix and returns the bias matrix. Normally, bias is initialized with a value of 0. Besides hyper-parameters, ANN topology is a very important factor for an ANN model to successfully extract data patterns from a given data set. A generic neural network architecture or topology is illustrated in figure 8. ANN topology shows how each ANN neuron connects to other ANN neurons. Typically, a fully connected ANN topology is used so each neuron can receive all information from the others. The implementation of functions `weight_variable`, `bias_variable` and `fc_layer` is described in listing 16.

```
39  def weight_variable(name, shape):
40
41      initer = tf.random_normal_initializer()
42      return tf.get_variable('W_' + name,
43                             dtype=tf.float32,
44                             shape=shape,
45                             initializer=initer)
46
47  def bias_variable(name, shape):
48
49      initial = tf.constant(0.0, shape=shape, dtype=tf.float32)
50      return tf.get_variable('b_' + name,
51                             dtype=tf.float32,
52                             initializer=initial)
53
54  def fc_layer(x, num_units, name, use_relu=True):
55
56      in_dim = x.get_shape()[1]
57      W = weight_variable(name, shape=[in_dim, num_units])
58      b = bias_variable(name, [num_units])
59      layer = tf.matmul(x, W)
60      layer += b
61      if use_relu:
62          layer = tf.nn.relu(layer)
63      else:
64          layer = tf.nn.sigmoid(layer)
65      return layer
```

Listing 16. Functions to create a fully connected layer, initialize bias variable, and initialize weight variable

The implementation of an 8x3 ANN model in which there are 3 layers and each layer has 8 neurons as depicted in listing 17. In other words, the width of this ANN is 8 and the depth is 3. Variable `x` from listing 17 is the placeholder for the input matrix, variable `y` is the placeholder for the output matrix. Three hidden layers are constructed by calling the `fc_layer` function in listing 15 and the results are stored in variables `fc1`, `fc2` and `fc3` accordingly. The variable `output_logits` in listing 17 is one hot encoding of the output matrix.

To summarize, the ANN has the following topology: one input layer, 3 hidden layers and one output layer. The input layer is represented by the matrix with dimension which equals to the number of training samples times 2. Each of the hidden layers is a matrix of shape of 2 times 8. The output layer matrix is also in the shape of 2 times 8. Following the matrix multiplication rule, the ouput_logits has the dimension which is equivalent to the number of training samples times 2. This is also the shape of the variable `y`, the

placeholder for output matrix. The calculation is called dimension verification which prevents errors from occurring in matrix multiplication operation.

```
88    h1 = 8   # Number of units in the first hidden layer
89    h2 = 8   # Number of units in the second hidden layer
90    h3 = 8   # Number of units in the  hidden layer
91
92  # Create the graph for the linear model
93  # Placeholders for inputs (x) and outputs(y)
94    x = tf.placeholder(tf.float32, shape=[None, n_inputs], name='X')
95    y = tf.placeholder(tf.float32, shape=[None, n_classes], name='Y')
96
97
98    fc1 = fc_layer(x, h1, 'FC1', use_relu=True)
99    fc2 = fc_layer(fc1, h2, 'FC2', use_relu=True)
100   fc3 = fc_layer(fc2, h3, 'FC3', use_relu=True)
101   output_logits = fc_layer(fc3, n_classes, 'OUT', use_relu=True)
102 # print ('output_logits: ' + str(output_logits))
```

Listing 17. Fully connected ANN

Variable `cls_prediction` from listing 18 stores the value of the class or category which is predicted by the ANN model. It is the category in which the index has the highest probability among indexes of the output_logits one hot encoding matrix. Variable `cls_prediction` has either 0 or 1 in its value. Variable `y_true_test` from listing 18 stores the true classification or category of the training data set. Similar to variable `cls_prediction`, it also contains a value of either 0 or 1.

```
104 # Network predictions
105   cls_prediction = tf.argmax(output_logits, axis=1, name='predictions')
106   y_true_test  = tf.argmax(y_test, axis=1, name='y_true_test')
107   correct_prediction = tf.equal(tf.argmax(output_logits, 1), tf.argmax(y, 1), name='correct_pred')
108   accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32), name='accuracy')
```

Listing 18. Variables contains network predictions

Variable correct_prediction from listing 18 stores the result of the comparison between the predicted class from the ANN model and the correct class. If the predicted class is the same with the correct class, boolean value True is assigned to this variable. Otherwise, the value of correction_prediction is False.

```
111 # Define the loss function, optimizer, and accuracy
112   loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y, logits=output_logits), name='loss')
113   optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate, name='Adam-op').minimize(loss)
114 # optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate).minimize(loss)
```

Listing 19. Define loss function, optimizers for training process

The implementation of the loss function and optimizer for the ANN model is described in listing 19. Tensorflow funcion `tf.nn.softmax_cross_entropy_with_logits` is called for constructing the variable loss which contains the loss value for loss function at line 112. Loss function represents the difference in value of all the predictions. The goal of the optimizer is to reduce the loss value to its minimum. The variable optimizer from listing 19 contains the result of calling the function `tf.train.AdamOptimizer` to minimize the loss variable with the specified learning rate of 0.001 in listing 15.

```
118  # Create the op for initializing all variables
119  init = tf.global_variables_initializer()
120
121  sess = tf.InteractiveSession()
122  sess.run(init)
123
124  saver = tf.train.Saver()
```

Listing 20. Global variables, session and model saver initialization

The process of initialization of all global variables of the Tensorflow execution graph is depicted in line 119 of listing 20. After that, defining and running the session is implemented in lines 121 and 122 in listing 20. Finally, the saver function to save the trained ANN model for future use is declared in line 124.

5 Results and discussion

5.1 Results

5.1.1 Result on training, validation and testing subsets

The visualization of the trained ANN model with accuracy of 99.3% on the training set is shown in figure 17. Accuracy on validation data set and testing set are 98.9% and 99.6% accordingly. The small difference between the accuracies obtained on 3 subsets of data proves that the training process is successful and the trained ANN model could be able to generalize and provide prediction on the new route data set.

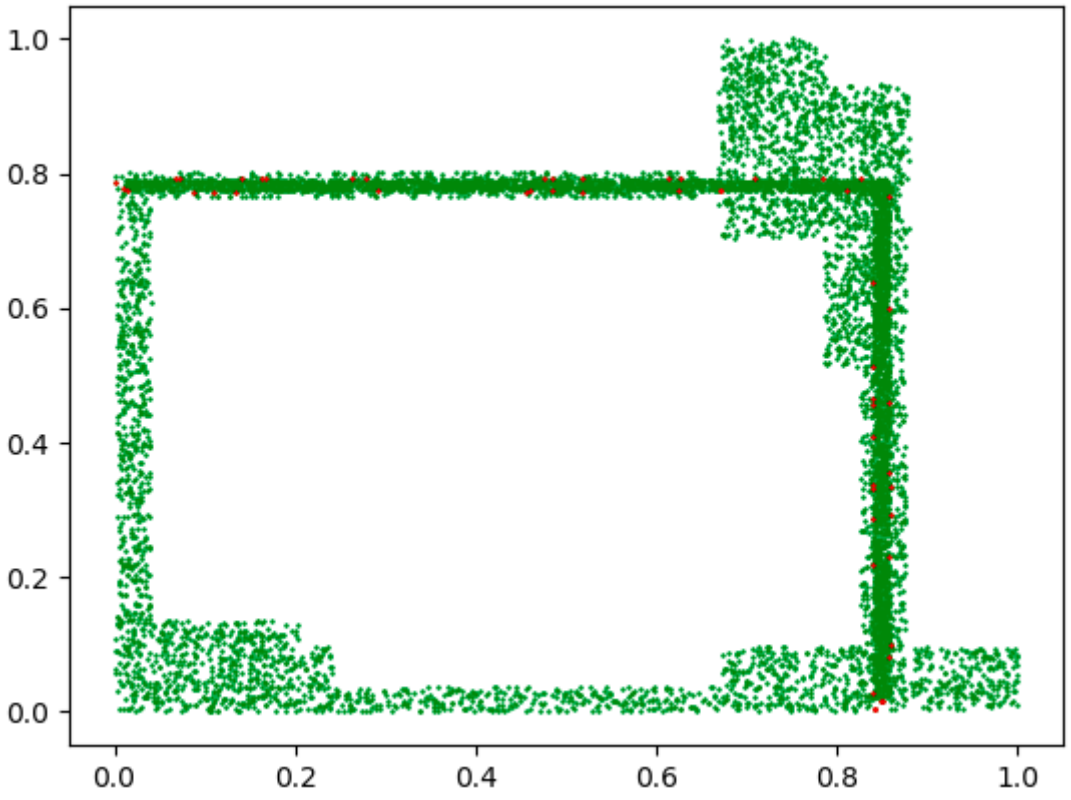

Figure 17. Trained ANN model with accuracy of 99.3% on the training set

The ANN model is trained on a data set of approximately 10,000 data points. Hyper-parameters include epochs of 20000 iterations, batch size is 3000 data points, learning

rate is 0.001. The topology of the ANN model is 1 input layer, 3 hidden layers and 1 output layer. The average training time for this ANN model is approximately 272 seconds which is around 4.5 minutes.

Points with green color are correctly categorized while points with red color are wrongly labeled as depicted in figure 17. It is noticed that most of the points , which are classified wrongly, are located near the border line between the route and the non-route area.

5.1.2 Result on a new route data

The visualization of a trained ANN model with accuracy of 100% on the new route data was shown in figure 18. Since all the points are in green color which means that all the points are correctly categorized. This proves that the model works quite well in classification problems.

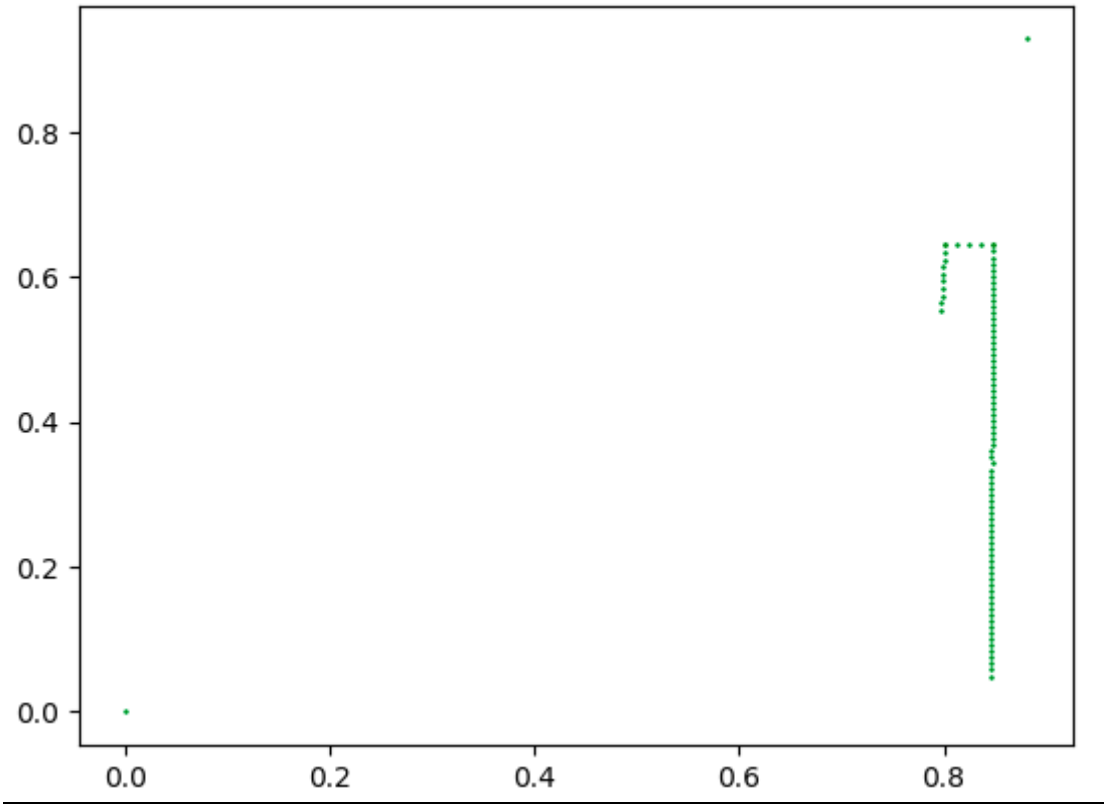

Figure 18. Trained ANN model with accuracy of 100% on the new route data

In the same manner, any new route data could be put into the trained ANN and the user could receive the classification result even though the data has not been introduced to the model before. This is an advantage of the ANN model of modern AI bottom-up approach compared to classical AI top-down approach with a hard set of rules of logical `IF … THEN …` statement.

5.2 Discussion

5.2.1 Unbalanced data problem

The achieved results as depicted in figure 17 is with the assumption of balanced data. This means that the total number of data points in both route and non-route are approximately at the same level. This condition also applies when there are more categories to be classified by an ANN model.

In this case, the area of the predefined route is approximately 10% of the total area, hence the ratio of non-route over route is around 9 times. This leads to the issue in which an ANN model could learn to predict all points as non-route data and still obtain the accuracy of 90%. Conducting an experiment on such unbalanced data proves the statement. As seen from figure 19, the trained ANN model predicts all data points in the predefined route as non-route and 90% accuracy is still obtained.

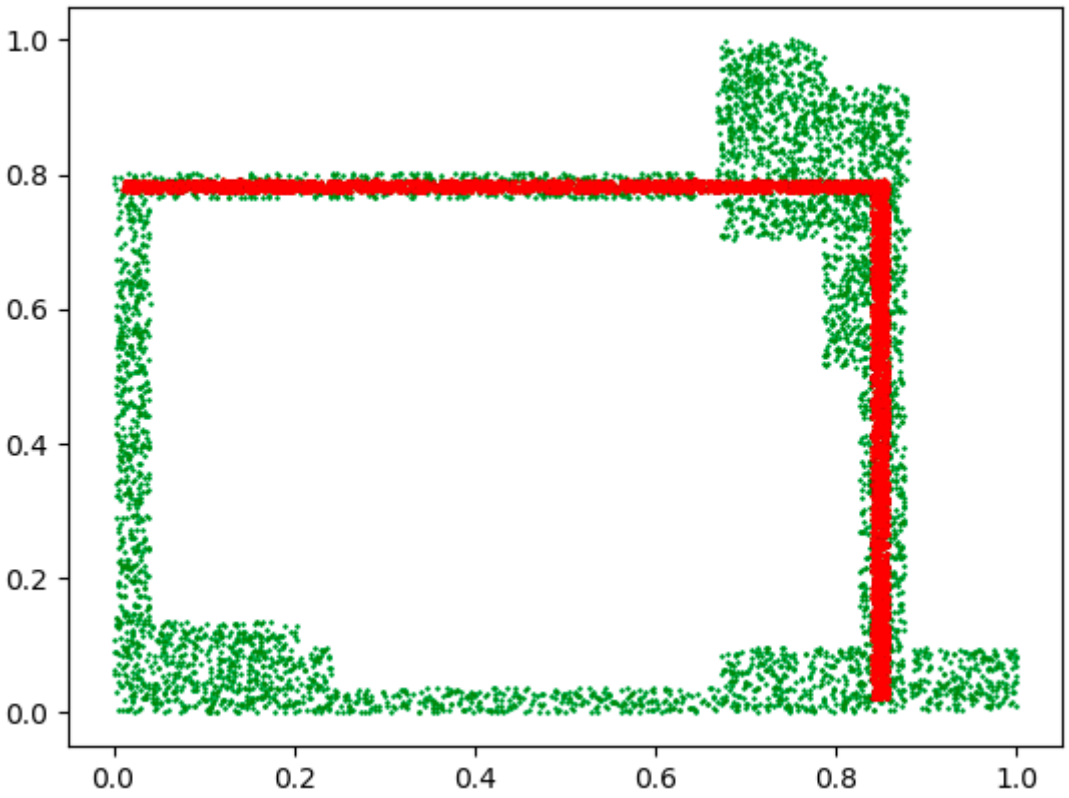

Figure 19 The result obtained given the unbalanced data

Therefore, it is very important to make data balanced before training an ANN model. There are two commonly used methods to achieve balanced data. The first method is to put more weights on the data set which is smaller so their influence on the result is bigger. The second method is to increase the number of data points of the smaller data set by adding randomized data points from itself so that the total number of data is approximately equal to other classes. In this project, the second method is used as shown in Listing 8.

5.2.2 Influence of epochs on accuracy and training time

This subsection explains how the number of epochs affects the performance of the ANN in terms of accuracy and training time. The total number of data points is 10,000 points with the batch size of 3,000 and the learning rate of 0.001 will be used to train the ANN. The ANN topology is 1 input layer, 3 hidden layers with 8 neurons on each of them, and

1 output layer. The number of epochs is set in increasing order starting from 0 to 30,000. The following result is obtained.

Table 1. Results of how epochs number affects accuracy and training time

Epochs	Accuracy	Training time
300	72.0%	4
1000	77.6%	14
3000	92.9%	39
10000	98.7%	135
20000	99.0%	272
30000	99.2%	395

As seen in table 1, when the number of epochs increases, the accuracy and training time also increases. To achieve the accuracy of 99% and above, the required training time is 272 ± 10 seconds which is approximately 5 minutes. Since this is a stochastic algorithm which uses randomness to solve optimization problems, it is observed that both accuracy and training time obtained is slightly different each time and this result represents the average plus/minus the error to conform with scientific notation.

5.2.3 Influence of batch size on accuracy and training time

The influence of the batch size on the performance of the ANN in terms of accuracy and training time will be conducted in this subsection. The total number of data points is 10,000 points with an epoch number of 20,000 and a learning rate of 0.001 will be used to train the ANN. The ANN topology is 1 input layer, 3 hidden layers with 8 neurons on each of them, and 1 output layer. The batch size is set in increasing order starting from 100 to 10,000. The result is described in table 2.

Table 2. Results of how batch size affects accuracy and training time

Batch size	Accuracy	Training time
100	99.1%	2193
300	99.0%	858
1000	99.3%	375
3000	99.4%	267
5000	53.1%	239
10000	52.9%	42

As seen from table 2, if batch size is equal or smaller than 3,000, the accuracy above 99% is achieved. If batch size is equal or above 5,000, the accuracy reduces dramatically to around 50% level which is the same result as without training. The reason for low accuracy at batch size of 5000 is due to the gradient descent algorithm only can only find the local minimum on that running session. Conducting more experiments with batch size above 5000 proves that the bigger the batch size, the higher the chance the algorithm could only find local minima. However, when batch size increases, the training time decreases significantly. From the observation, there is a tradeoff between higher chance of finding global minimum and higher training time versus lower chance of finding global minimum and lower training time.

5.2.4 Influence of learning rate on accuracy and training time

This subsection examines the effect of learning rate on the performance of the ANN in terms of accuracy and training time. The total number of data points is 10,000 points with batch size of 3,000 and epochs number of 20,000 will be used to train the ANN. The ANN topology is 1 input layer, 3 hidden layers with 8 neurons on each of them, and 1 output layer. The learning rate is set in ascending order starting from 0.001 to 0.1. The result is shown in table 3.

Table 3. Results of how learning rate affects accuracy and training time

Learning rate	Accuracy	Training time
0.001	99.4%	267
0.003	99.0%	268
0.01	98.9%	258
0.03	53.1%	257
0.1	53.1%	257

In general, when learning rate increases, the accuracy slightly reduces, however the training time remains almost unchanged as shown in table 3. Until the learning rate of 0.01, the training accuracy is kept close to 99% level. When the learning rate is at 0.03, there is a remarkable phenomenon in which the accuracy slowly increases from 53.1% to maximum of 95.3% at epoch 1098 and decreases back to 53.1% level. This phenomenon is due to too large a learning rate where instead of going downhill to find local or global minimum, the gradient descent algorithm goes uphill and as a result, accuracy reduces. The observation at learning rate of 0.1 supports the explanation when accuracy of 53.1% is obtained which means that if the learning rate is above a certain threshold, gradient descent algorithm could not go downhill to find its minimum. Nevertheless, the learning rate does not affect the training time since it remains at around 260 seconds.

5.2.5 Influence of ANN topology on accuracy and training time

This subsection studies how ANN topology affects the performance of the ANN in terms of accuracy and training time will be conducted. The total number of data points is 10,000 points with batch size of 3,000, epochs number of 20,000 and learning rate of 0.001 will be used to train the ANN. The number of hidden layers and number of neurons in each layer will be modified to generate a variety of ANN topologies for the study.

Table 4. Results of how an ANN topology affects accuracy and training time

Topology	Accuracy	Training time
in-888-out	99.4%	267
in-88-out	97.6%	235
in-8-out	85.7%	202
in-666-out	98.6%	269
in-444-out	72.1%	233
in-222-out	53.1%	249
in-864-out	95.9%	245

There are two ways to modify the ANN topology. The first method is to change the number of hidden layers which is the depth of the ANN and the second method is to change the number of neurons in each layer which is the breadth of the ANN.

When changing the number of hidden layers from 3 to 1 and keeping the number of neurons in each layer at 8, it is observed that the accuracy decreases from 99.4% to 85.7%. This is due to the fact that with less features, the ANN model could not draw necessary classification lines in the graph to make correct predictions and therefore accuracy is reduced. The training time also decreases if the number of hidden layers decreases as shown in table 4.

In the same manner, when changing the number of neurons in each layer from 8 to 2 and keeping the number of hidden layers at 3 layers, the accuracy also decreases. With the topology of in-444-out, the accuracy reduces to 72.1% and with topology of in-222-out the accuracy is only 53.1%. Lack of necessary features is also the root cause for this observation. However, when the number of neurons gradually decreases from 8 in layer 1 to 4 in layers 3, the accuracy remains at a high level of 95.9%.

The conclusion is that the ANN topology should accommodate the necessary data features, otherwise this will affect the accuracy of the model significantly. How wide and deep of an ANN depends on the nature of the data set. In the case that the width and

depth is too small, there will be a lack of materials to build the necessary patterns which are required by the learning process and leads to the underfitting situation—high bias. On the opposite, if the width and depth of the ANN is too large, it could result in the overfitting problem—high variance. Therefore, the tuning process is extremely important to discover the optimized hyper-parameters and ANN topology which enhances the performance of the model. Some of the results of this are illustrated in the Appendices 2 and 3.

5.2.6 Influence of total number of data points on accuracy and training time

This subsection examines the effect of total number of data points on the performance of the ANN in terms of accuracy and training time. The epoch number is 20,000 with batch size of ⅓ of the total number of data points, and a learning rate of 0.001 will be used to train the ANN. The ANN topology is 1 input layer, 3 hidden layers with 8 neurons on each of them, and 1 output layer. The number of data points is in increasing order starting from 1,000 to 30,000. The result is illustrated in table 5.

Table 5. Results of how a total number of data points affects accuracy and training time

Number of data points	**Accuracy**	**Time**
1000	91.7%	112
3000	98.6%	145
10000	99.1%	272
30000	99.5%	540

The batch size of ⅓ of the total number of data points is concluded from subsection 5.2.3 to be in the range of optimized hyper-parameters to use for an ANN model. As seen from table 5, the accuracy improves significantly with an increasing number of data points. This explains one of the facts that big data matters and hence the trend for big data in recent technology development. On the other hand, big data also comes with the cost of training time and energy consumption.

5.2.7 Influence of route complexity on accuracy and training time

The effect of route complexity on the performance of the ANN in terms of accuracy and training time will be studied in this subsection. The total number of data points is 10,000 points with batch size of 3,000, epoch number of 20,000, and a learning rate of 0.001 will be used to train the ANN. The ANN topology is 1 input layer, 3 hidden layers with 8 neurons on each of them, and 1 output layer. Three different predefined routes with increasing complexity will be used to train the ANN model. The predefined routes with level 2 and 3 complexities are shown in the Appendix 1.

Table 6. Results of how complexity of the route affects accuracy and training time

Route number	**Accuracy**	**Training time**
Route 1	99.1%	272
Route 2	97.1%	264
Route 3	95.7%	275

As observed from table 6, when the complexity of the route increases, the accuracy of the trained ANN model decreases. This can be explained that with current ANN topology it might not be enough features to accommodate a more complex data set. As a result, increasing both the depth and breadth of the ANN could improve the accuracy. Another method to improve the accuracy performance is to increase the total number of data points to 30,000 as explained in subsection 5.2.6.

5.2.8 Possible improvement

There are 3 basic models, which includes linear regression, logistic regression and ANN, are studied for the classification task. Linear regression is useful for predicting continuous values such as house price, temperature. However, classifying route and non-route belongs to the discrete value domain so linear regression is not a good model. Logistic regression is a good candidate to solve discrete value classification problems because most of the outcome values of logistic function are either 1 or 0 in value. Nevertheless, the disadvantage of using this model is that it requires explicit modeling of the

interactions which is quite similar to hard rules of classical AI approach. Therefore, ANN is the best suitable model for the route classification task. However, the logistic regression can be interesting to explore so that performance could be compared between these two models.

Typical ratio of 80/10/10 or 70/15/15 is used for data subsets division. The largest proportion is included in the training data set so that the ANN model could obtain sufficient data points to learn from since less data points could lead to lower accuracy as shown in subsection 5.2.6. Normally, the proportion of validation data set and testing set is equal since it serves the monitoring convergence and testing purposes quite well. In this application, the training set is 79%, validation set is 7% and testing set is 14%. The reason for setting the testing set proportion two times of the validation set is to achieve a larger number of wrongly labeled points (red points) for model limitation tests. The lesser the testing data set is, the lesser the data points lie near to the border line between route and non-route and therefore increasing the chance of correct predictions.

One of the advantages of the ReLu activation function compared with the Sigmoid activation function is it allows faster learning which means it costs less in training time. Another advantage is that it reduces the vanishing gradient descent issue due to constant gradient of the function. The vanishing gradient descent problem seriously affects the performance of the Sigmoid activation function because the slope of this function is near zero value as the function approaches 0 and 1 values on the vertical axis. The scope of this project does not include the influence of activation function on the performance of the ANN. However, it would be an interesting topic to explore even further in subsequent studies. As explained in subsection 5.2.3, there is a tradeoff between different batch sizes regarding the chance of finding global minimum and training time. More experiments could be conducted to find out what is the optimal expectation value considering the probability of finding global minimum.

Even though the basic requirements of the project are fulfilled, there are rooms for further improvement and development as well. The first area for future development is to take the direction of a predefined route into consideration. This requires the introduction to one more input feature which is time, therefore there will be a total of 3 input features which are X-coordinate, Y-coordinate and time. The output remains at 2 dimensions

which are either route or non-route as one hot encoding vector. However, there is one aspect that needs to be noticed which is the size of the data. An ANN model could not learn from positive examples only but also from negative examples. Without the time dimension, the number of data points, which are used to train the ANN model, is 10,000. These data points are divided equally for both positive and negative examples as a requirement of balanced data. When a time dimension is added, for each time step, the model needs approximately 10,000 data points to train. For a data set of the size of 10,000, it takes around 5 minutes to train. The more the number of time steps increase, the longer time the training process requires. In the same manner, it would take 50 minutes to train a data set with only 10-time steps, and hence this approach becomes not realistic. The unsupervised learning algorithm such as K-means clustering could be a good candidate to solve this problem.

Since the scope of this project is to explore the possibility of using ANN to solve real-life problems, using complex data flow and software architecture to enhance performance is avoided. Therefore, the second area for future development is how to propose a software architect to integrate this solution into the existing software architecture, especially making ANN models running in embedded systems like Linux real time operating system which is the same system used by the wearable devices.

The final area for improvement is the User Interface (UI) and User Experience (UX). In this application, the building layout is simplified in order to focus into exploration of the possibility of the ANN model to solve the classification challenge. More realistic building layout and user-oriented application can be a target for next delivery.

6 Conclusion

This is the era of big data and those who own more data could have huge advantages compared to those who do not. Instead of the gold rush, the data rush has been seen and is going to be seen in the near future. The next question is what to do with the big data, what information could benefit us as a whole. AI and ML provide an answer to those questions by extracting useful data patterns which used to be very hard or inefficient using conventional methods. This project is an attempt to use modern AI tools to solve real-life problems.

In this project, the relationship and similarity between the biological network of neurons and ANN has been explained in detail. The application using Python programming language and Tensorflow framework demonstrates how to apply the model to perform classification tasks such as determining whether data points belong to a predefined route. The advantage of modern AI models is also emphasized compared to classical models. For example, the trained ANN model could categorize the data points from new routes which it does not learn from. This could be a difficult task for models such as GPS.

This project also attempts to test different hyperparameters to see how they affect the performance of the ANN model in terms of accuracy and training time. These parameters include epochs, batch size, learning rate, ANN topology, total number of data points and route complexity. Furthermore, the study also provides a method on how to find the optimal values for these hyperparameters for the dataset from predefined route 1. This method could also be applied to different datasets for performance optimization.

Since the scope of the project is limited to exploring the possibility of applying ANN model to solve real-life problems; software architecture, data flow, UI and UX implementation is kept at minimal level and hence these areas could be targets for improvement for future development. Besides, alternative models such as logistic regression could be implemented for performance benchmarking with the current ANN model. Based on calculation of theoretical training time, it is also interesting to notice that there is a limitation for the practical use of ANN model if a time dimension is added as another input feature. Therefore, unsupervised learning models such as K-means clustering could be an alternative solution to solve classification problems with 3 input features.

For the route complexity, further study could be done to verify that routes with higher complexities such as curved routes or curved and lined combination routes could be classified effectively using the ANN model. This project has introduced the method to find the optimal hyperparameters to obtain the best performance of the mode in terms of accuracy and training time. However, because of the time constraint of the project, the current error range of the obtained parameters is quite large. In the subsequent study, a fine-tuning process could be performed to reduce the errors even further.

References

1. Warwick K. Artificial Intelligence: the Basics. Florence: Taylor & Francis Group; 2011.

2. Paul P. Maillardet's Automaton [Internet]. Atlas Obscura. 2020 [cited 16 April 2020]. Available from: https://www.atlasobscura.com/places/maillardets-automaton

3. Allen N. A guide to the general problem-solver program GPS-2-2. California: Rand Corporation; 1963.

4. Joshua E. General Problem Solver (GPS) [Internet]. Ai-su13.artifice.cc. 2020 [cited 16 April 2020]. Available from: https://ai-su13.artifice.cc/gps.html

5. Coppey L. What does AlphaGo vs Lee Sedol tell us about the interaction between humans and intelligent systems? [Internet]. Medium. 2020 [cited 16 April 2020]. Available from: https://medium.com/point-nine-news/what-does-alphago-vs-8dadec65aaf

6. Alina B. The Five (and More) Senses [Internet]. livescience.com. 2020 [cited 16 April 2020]. Available from: https://www.livescience.com/60752-human-senses.html

7. Meola C, editor. Infrared Thermography : Recent Advances And Future Trends. SAIF Zone: Bentham Science Publishers; 2012.

8. Nippon Avionics Co.,Ltd. InfReC R300BP-TF | Infrared Thermography | NIPPON AVIONICS CO.,LTD. [Internet]. Infrared.avio.co.jp. 2020 [cited 16 April 2020]. Available from: http://www.infrared.avio.co.jp/en/products/ir-thermo/lineup/r300bp-tf/index.html

9. Floreano D, Mattiussi C. Bio-Inspired Artificial Intelligence : Theories, Methods, and Technologies. Cambridge: MIT Press; 2008.

10. Hafsi B, Elmissaoui R, Kalboussi A. Neural Network Based on SET Inverter Structures: Neuro-Inspired Memory. World Journal of Nano Science and Engineering. 2014;04(04):134-142.

11. Richard N. The differences between Artificial and Biological Neural Networks [Internet]. Medium. 2020 [cited 16 April 2020]. Available from: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

12. David R. What's the Difference Between Data Science, Machine Learning, and Artificial Intelligence? [Internet]. Pubs.spe.org. 2020 [cited 16 April 2020]. Available from: http://pubs.spe.org/en/twa/twa-article-detail/?art=3781&gclid=Cj0KCQiAv8PyBRDMARIsAFo4wK3GSgMifi-PDEF7oU872Dn6DSoH8Dojq-W-_IT8XF4IaLuPkrz5jVwaAg0uEALw_wcB.

13. Moore J, Raghavachari N. Artificial Intelligence Based Approaches to Identify Molecular Determinants of Exceptional Health and Life Span-An Interdisciplinary Workshop at the National Institute on Aging. Frontiers in Artificial Intelligence. 2019;2.

14. Urvashi J. Why Data Normalization is necessary for Machine Learning models [Internet]. Medium. 2020 [cited 16 April 2020]. Available from: https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

15. Schwaber K, Sutherland J, Sutherland J. Software in 30 Days : How Agile Managers Beat the Odds, Delight Their Customers, and Leave Competitors in the Dust. Hoboken: John Wiley & Sons, Incorporated; 2012.

16. UKEssays. Waterfall Methodology in Software Development [Internet]. UKEssays.com. 2020 [cited 16 April 2020]. Available from: https://www.ukessays.com/essays/computer-science/waterfall-methodology-in-software-development.php

17. Anish S. Activation functions and it's types-Which is better? [Internet]. Medium. 2020 [cited 16 April 2020]. Available from: https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Routes with different levels of complexity

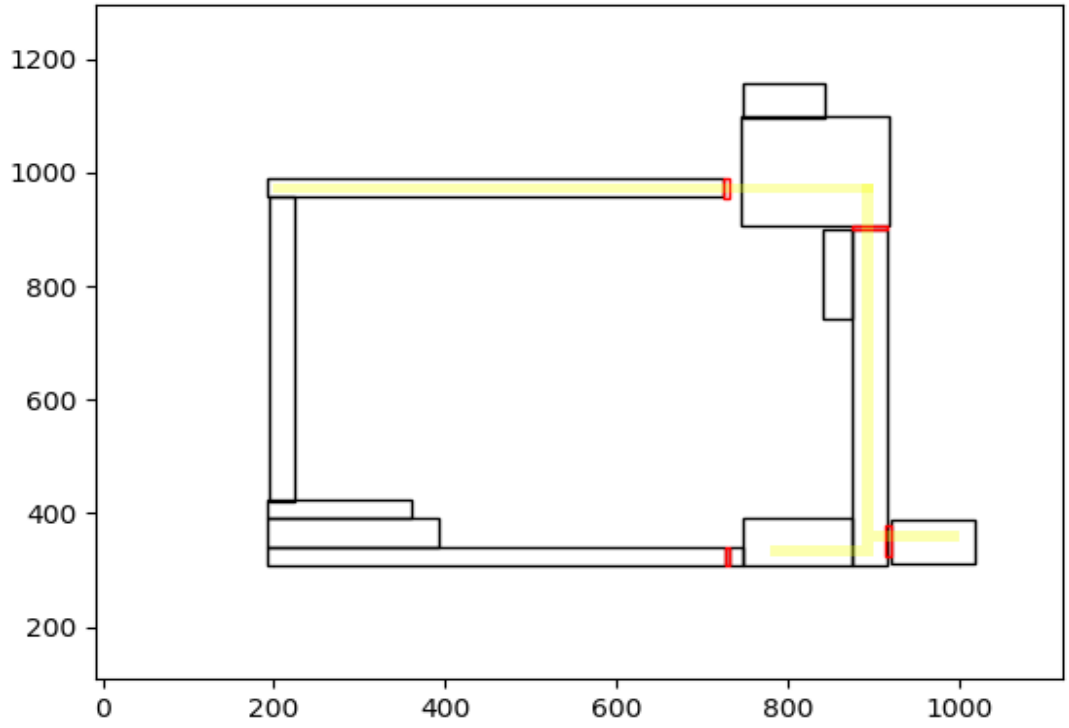

Figure 20 Route with level 2 complexity

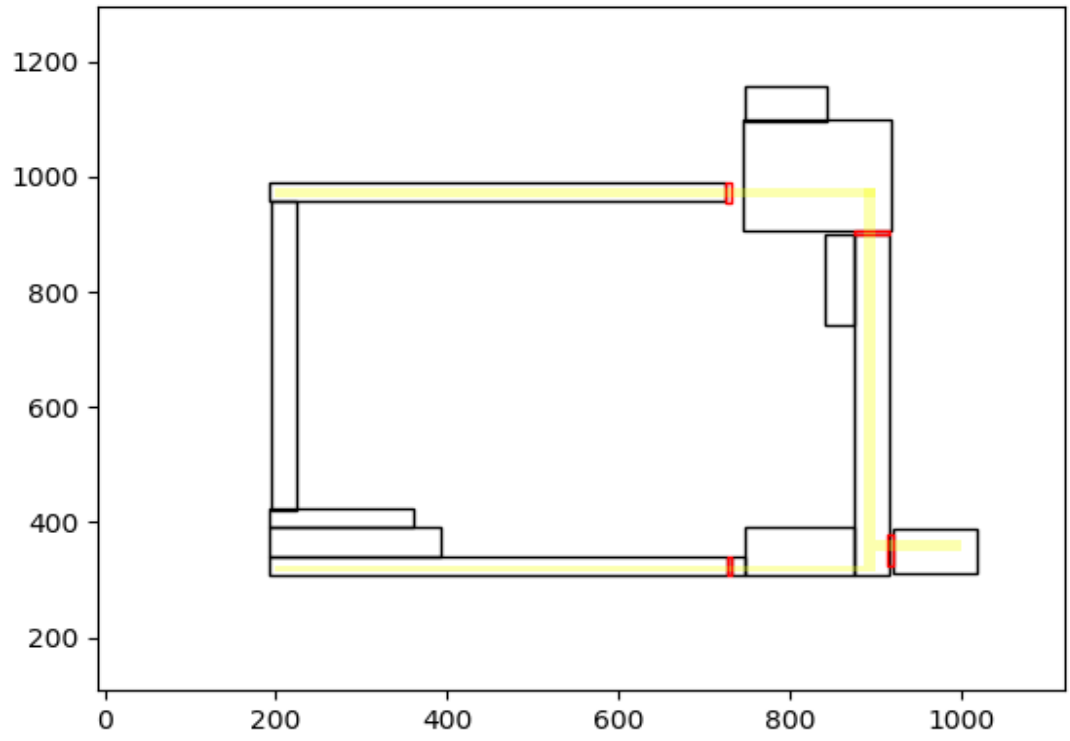

Figure 21 Route with level 3 complexity

Level 3 complexity result

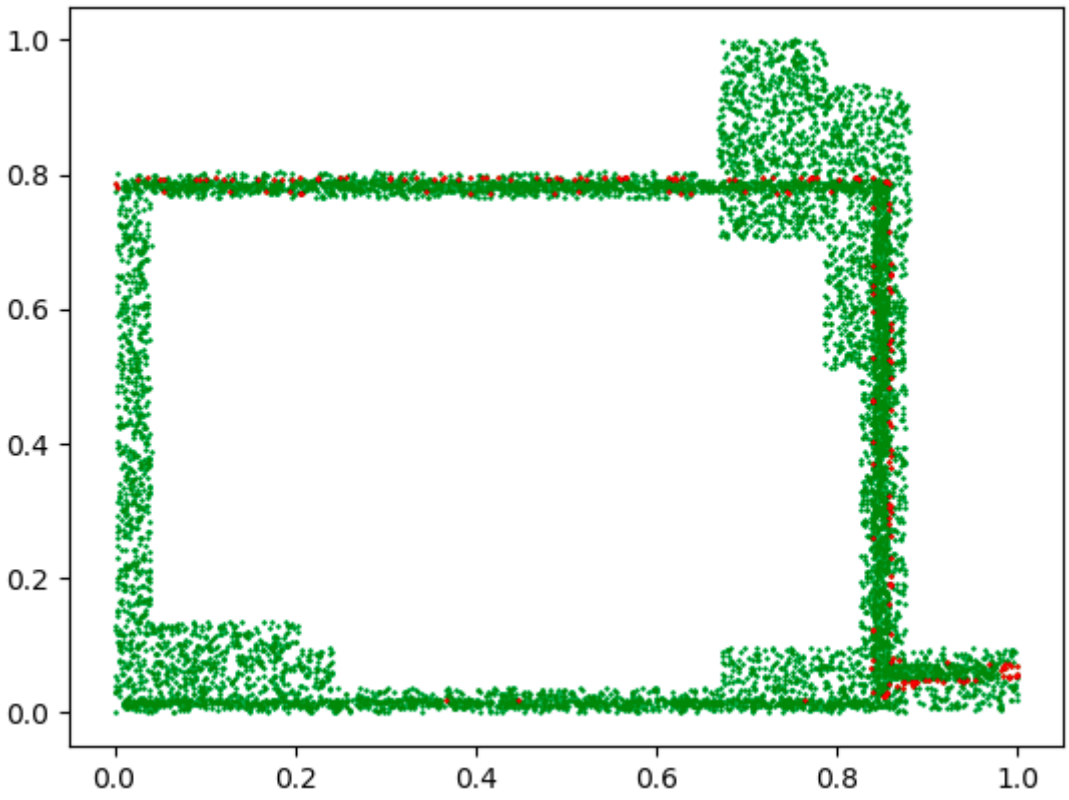

Figure 22 Result with route at level 3 complexity

ANN with different topologies

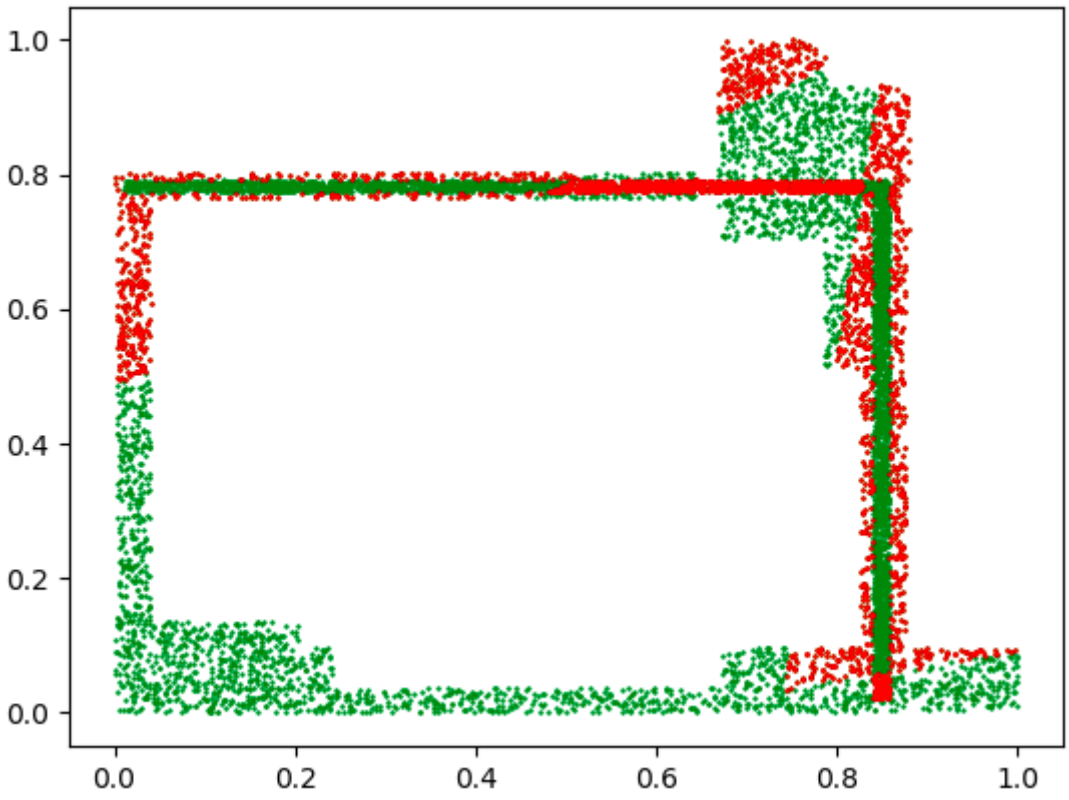

Figure 23 Result with the ANN topology of in-8-out

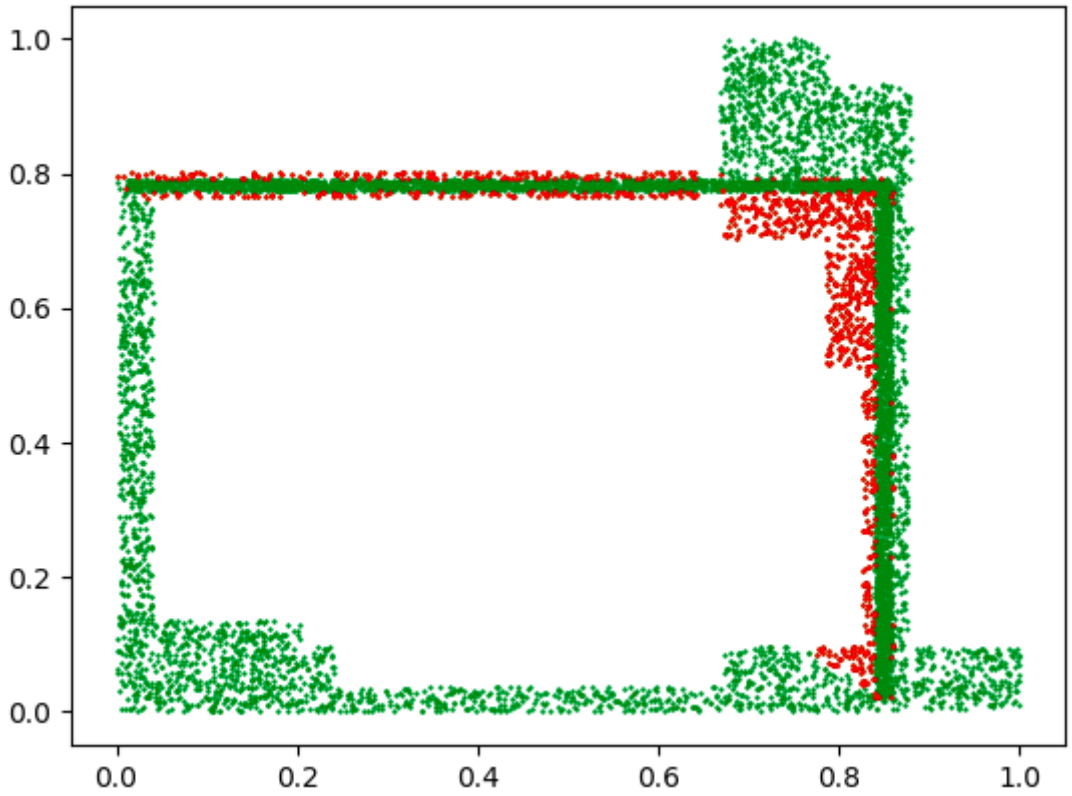

Figure 24. Result with the ANN topology of in-444-out