Neural Networks & Biometrics as an advanced form of authentication

Final Year Project (Design)

Institute of Technology, Tralee

Aoife Sayers T00170881

# Abstract

In recent years biometrics are becoming a popular form of authentication in conjunction with passwords.

Passwords have become key requirements for users who want to perform any online activity in a secure environment. The ever-changing complexity requirements of passwords and security breaches of passwords is becoming every changing in today’s world. Biometric Authentication is one method to overcome this issue. Linear algorithms are no longer the only manner to authenticate a user. Artificially Intelligent Neural Networks will also be a likely method of authenticating a user in years to come.

The aim of this study was to investigate if authenticating users biometric data with artificially intelligent neural networks would improve today’s authentication in biometrics.

Table of Contents

[Abstract 2](#_Toc507494722)

[Abbreviations 9](#_Toc507494723)

[1. Chapter 1 - Artificial Neural Network 10](#_Toc507494724)

[1.1. Artificial Intelligence 10](#_Toc507494725)

[1.2. Artificial Neural Networks 10](#_Toc507494727)

[1.3. Neurons 11](#_Toc507494728)

[1.4. Activation Function 11](#_Toc507494729)

[1.5. Bias 13](#_Toc507494730)

[1.6. Connection 13](#_Toc507494731)

[1.7. Threshold/Step Function 13](#_Toc507494732)

[1.8. Weights 13](#_Toc507494733)

[1.9. Neural Network Layers 15](#_Toc507494734)

[1.9.1. Input Layer 15](#_Toc507494735)

[1.9.2. Hidden layers 15](#_Toc507494736)

[1.9.3. Output Layer/Target 15](#_Toc507494737)

[1.10. Learning Rules 15](#_Toc507494738)

[1.10.1. Hebbian Learning Rule 16](#_Toc507494739)

[1.10.2. Delta Learning Rule/ Widrow-Hoff Rule/ Perceptron Learning Rule 16](#_Toc507494740)

[1.10.3. Competitive Learning Rule / Winner Takes All 17](#_Toc507494741)

[1.11. Epoch 17](#_Toc507494742)

[1.12. Training Set, Testing Set & Validation Set 17](#_Toc507494743)

[1.12.1. Training Set 17](#_Toc507494744)

[1.12.2. Testing Set 17](#_Toc507494745)

[1.12.3. Validation Set 17](#_Toc507494746)

[1.13. Neural Network Learning Strategies 18](#_Toc507494747)

[1.13.1. Unsupervised Learning 18](#_Toc507494748)

[1.13.2. Supervised Learning 18](#_Toc507494749)

[1.14. Applications of Neural Networks 19](#_Toc507494750)

[1.15. Neural Network Types 20](#_Toc507494751)

[1.15.1. Single Layer Perceptron 20](#_Toc507494752)

[1.15.2. Multi-Layer Perceptron/Deep Feedforward Neural Networks 20](#_Toc507494753)

[1.15.3. Recurrent Neural Network 20](#_Toc507494754)

[1.15.4. LSTM Neural Network 20](#_Toc507494755)

[1.15.5. Hopfield Neural Network 21](#_Toc507494756)

[1.15.6. Boltzmann Machine 21](#_Toc507494757)

[1.15.7. Convolutional Neural Network 22](#_Toc507494758)

[1.16. The dot product 24](#_Toc507494759)

[1.17. Feed forward Propagation 25](#_Toc507494760)

[1.17.1. Workthrough of an XOR in feed forward propagation 25](#_Toc507494761)

[1.17.2. Backpropagation work through for XOR 29](#_Toc507494762)

[1.18. Neural Network Libraries 33](#_Toc507494763)

[1.18.1. NNabla by Sony 33](#_Toc507494764)

[1.18.2. Tensorflow 33](#_Toc507494765)

[1.18.3. Caffe 36](#_Toc507494766)

[1.18.4. Keras 36](#_Toc507494767)

[2. Chapter 2 – Biometrics 37](#_Toc507494768)

[**2.1.** Introduction 37](#_Toc507494769)

[**2.2.** Unimodal & Multimodal systems 37](#_Toc507494770)

[**2.3.** Biometric Software/Hardware 37](#_Toc507494771)

[2.3.1. Fingerprint Scanners 37](#_Toc507494772)

[**2.4.** Biometric Authentication 37](#_Toc507494773)

[**2.5.** Physical Biometrics 38](#_Toc507494774)

[2.5.1. Fingerprint Identification 38](#_Toc507494775)

[2.5.2. Palm Vein 43](#_Toc507494776)

[2.5.3. Retina 43](#_Toc507494777)

[2.5.4. Iris scan 44](#_Toc507494778)

[2.5.5. FacIAL recognition 44](#_Toc507494779)

[2.5.6. Signature 45](#_Toc507494780)

[2.5.7. Voice 45](#_Toc507494781)

[2.5.8. DNA 45](#_Toc507494782)

[**2.6.** Behavioural Biometrics 45](#_Toc507494783)

[3. Chapter 3 – Authentication, Authorization and Encryption 46](#_Toc507494784)

[**3.1.** Authentication 46](#_Toc507494785)

[**3.2.** Sessions 46](#_Toc507494786)

[**3.3.** Passwords 46](#_Toc507494787)

[Prevent Brute-Force Attacks 47](#_Toc507494788)

[4. Methodology & Design 48](#_Toc507494789)

[**4.1.** Research Undertaken 48](#_Toc507494790)

[**4.2.** Research Question 48](#_Toc507494791)

[**4.3.** Vision Document 49](#_Toc507494792)

[Introduction 49](#_Toc507494793)

[Scope/Outline 49](#_Toc507494794)

[Product Overview/Features 49](#_Toc507494795)

[User requirements, functional and non-functional 50](#_Toc507494796)

[Datasets 53](#_Toc507494797)

[Solutions 55](#_Toc507494798)

[**4.4.** Project Specification 55](#_Toc507494799)

[User Stories 55](#_Toc507494800)

[Use Cases and Narratives 57](#_Toc507494801)

[Risk Analysis 60](#_Toc507494802)

[Class Diagrams 64](#_Toc507494803)

[Project Plan 64](#_Toc507494804)

[**4.5.** Design Phase 65](#_Toc507494805)

[Sequence Diagram 65](#_Toc507494806)

[Relational Schema/ Entity Relational Diagram 65](#_Toc507494807)

[Work Completed 65](#_Toc507494808)

[Screenshots 66](#_Toc507494809)

[Future Extensions 66](#_Toc507494810)

[2.1. Implementation 68](#_Toc507494811)

[References 69](#_Toc507494812)

# Abbreviations

ANN – Artificial Neural Networks

AI – Artificial Intelligence

RNN – Recurrent Neural Network

CNN – Convolutional Neural Network

CovNet – Convolutional Neural Network

NIST – National Institute of Standards & Technology

ITL - Information Technology Laboratory – research labratory within the NIST

NBIS - NIST Biometric Image Software

OWASP - Open Web Application Security Project

# Chapter 1 - Artificial Neural Network

## Artificial Intelligence



Artificial Intelligence(AI) is a methodology in Computer Science using computers/machines or computer systems mirroring the actions and intelligence of humans with algorithms. According to John McCarthy (founder of AI), he defines AI as “Machine Learning is a method of learning performed by a computer by using prediction algorithms”. (V.Chande, 2012)

## Artificial Neural Networks

An Artificial Neural Network (ANN) is a system that is loosely based on the learning processes found in the neural networks of the human brain. (Point, 2017) A neural network is a system of neurons or nodes (proccessing units) interconnected by connections containing weights. The neuron/ node is a processing element take takes inputs, adds weights to the connections, sums them up, adds a bias and used the outcome as the argument for a single-valued function (transfer function which results in the neuron’s output ( (M.Domnanovich, 2004). ANN’s are like Biological Neural Networks in that a ANN acquires knowledge through learning.



*Figure 1.2 – Figure of a Multi-Layer Neural Network* (cs231n, 2017)

## Neurons

The function of an artificial neuron involves summing its weighted input signal and applying an output or activation function. (Fausset, 1994). Neurons may also be reffered to nodes within a Neural Network.

## Activation Function

Activations of a neuron is a nodes/neurons level of activity. (Fausett, 1993). The function of an Activation function is to introduce non-linearity into the network to allow a neural network to compute complex problems. (VV, 2016). The activation mathematical function inside a neuron is used to produce an output referring to its input value. The input value should exceed a specified threshold value that determines, if an output to other neurons should be generated/fired. In summary the activation function determines whether the neurons should be activated. Activation functions may also be referred to as Transfer Functions.The most commonly used Activation functions carried out by a Neuron include:

**Linear Activation Function**

Including an linear activation function has the same effect of not applying an activation function to the ANN as the input and output signals of a neuron or node remain the same. The Activation function is denoted by: Function *f(x) = x* (K.Vijayarekha, 2013)

**Sigmoid Activation Function**

The Sigmoid Activation Function is a S Shaped Activation Function when graphed. The Logistic and hyperbolic tangent functions are commonly used sigmoid activation functions. Sigmoid activations functions are used for back propagation in Artificial Neural Networks (Fausset, 1994). Different types of sigmoid activation functions exist such as:

* Binary/Logistic
  + - Arctan
      * Range from -1 to 1
      * arctan(x);
    - Bipolar
    - Tanh

**Identity function/ Threshold/ Heavside Function**

The Identity function is denoted as . Single layer neural networks often use the step function to convert an input into a binary output (1 or 0) or bipolar (1 or -1). (K.Vijayarekha, 2013).

**ReLu – Rectified Linear Units**

The ReLu Activation Function most commonly used Activation function for deep learning models (K.Vijayarekha, 2013). It has proved to be 6 times better in convergence in comparison to the TanH layer activation function. The ReLu activation is only used within the hidden layers of the neural network. A Rectified linear unit has output 0 if the input is less than 0, When the input is positive, the derivative is just 1. The Relu Activation Function is denoted as follows: *f(x) = max(0)* (Walia, 2017)

## Bias

A "pseudo" input of a neural net with any value except zero. The function of the bias is to produce different inputs for different input patterns given to the net. The bias is an additional parameter which is used shift the output to the left or right along with the weighted sum of the inputs to the neuron.

## Connection

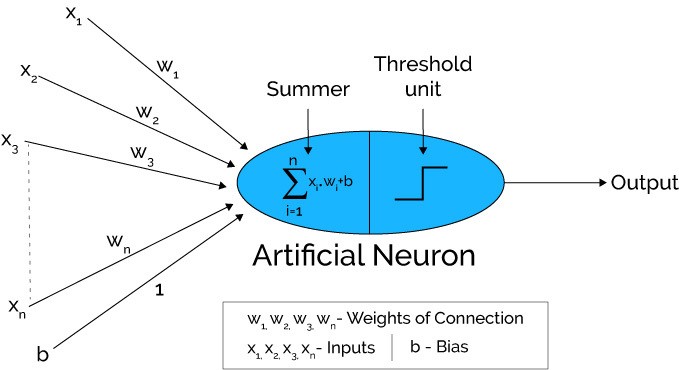
A connection is a path from one neuron to another to transfer information. These are also called synapses in biological neurons and they are often parameterised with weights (Hackernoon, 2017).

## Threshold/Step Function

This is a value used in some activation functions to determine the unit’s output. Changing the threshold’s purpose is to shift the graph of the activation function right or left. The threshold/step function has the same effect as a bias. (Ciaburro & Venkateswaran, 2017)

## Weights

Weights help to determine the strength of the signal that is transferred. Weights are parameters of integer values on the connections of a neural network. Setting the weight value distinguishes in different types of neural networks. During the training process of a neural network in Backward propagation - the weights are adjusted. The adjustment of weights continues until the individual or total errors in the responses exceed a specified level or until there are no measurable errors. This is known as convergence. (Fausset, 1994)



*Figure 1.8 Artificial Neuron with weights of connection, inputs and bias (Hacker Noon. 2017).*

## Neural Network Layers

A Neural Network is composed of various layers: an input layer, one or more hidden layers and Output layers – depending on the type of neural network. At each of the layers there are many interconnected neurons which contain the various Activation Functions. Each of the connections connected to the Neurons contain weights. (To be continued below). A layer is a pattern of weighted connections between two slabs of neurons and may also function in the same way. (Fausset, 1994)

## Input Layer

The input layer is the first layer/slab of a neural net that accepts certain input patterns and generates output values to the succeeding weight matrix. An Input layer typically transmits the input signal to all connected neurons. (Fausset, 1994)

## Hidden layers

A Hidden layer makes sense of complex patterns, like image recognition. The function of hidden layers allow a neural net's abilities to learn logical and more complex operations.

## Output Layer/Target

An Output layer is the last slab of neural Network. An ANN outputs a value or a matrix of values (pattern), generated by the neurons of the Artificial Neural Network. The output layer is used to measure the current error value/margin of error of the net.

## Learning Rules

As an Artificial Neural Network is a complex adaptive system, the weights are adjusted during training. Methods used to adjust weights are called learning rules. Learning Rules are mathematical algorithms to improve the Neural network performance. There are various learning rules for Neural Networks such as: Hebbian Learning Rule, Perceptron Learning Rule, Delta Learning Rule (Widrow-Hoff Rule) and the Competitive Learning Rule (Winner Takes All). (Point, 2017)

## Hebbian Learning Rule

One of the first learning rules developed in 1949, the Hebbian rule used it to identify how to improve the weights of neurons of an artificial neural network. The Hebian learning rule assumes that – If two neighbour neurons are activated and deactivated together, the weight values of the affected neurons should increase. For neurons working in opposite phases, the weights between the affected neurons should decrease. Otherwise in the event of no signal correlation, the weight should not change. (Team, 2017) The Hebian Rule for unsupervised learning is as follows:

[Mathematical Formula of Hebb Learning Rule in Artificial Neural Network.](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/hebbian-learning-rule.png)

## Delta Learning Rule/ Widrow-Hoff Rule/ Perceptron Learning Rule

The Delta Learning rule is the most common supervised learning rule in Artificial Neural Networks. It is a gradient descent learning rule for updating the weights of the inputs to artificial neurons in a neural network. “Delta” is the difference between actual and desired output. The Delta Learning also known as the Perceptron Learning Rule or the Widrow Hoff Rule.

## Competitive Learning Rule / Winner Takes All

## Epoch

An epoch refers to the number of iterations of providing the neural network with an input and updating the Artificial Neural Networks weights. A standard Neural Network may have in the region of thousands of epochs for sufficient training. (Ciaburro & Venkateswaran, 2017)

## Training Set, Testing Set & Validation Set

## Training Set

The Training set of a neural network involves adjusting the Neural Networks weights in order to increase the performance accuracy. A sufficient amount of data should be fed through the network in order for Training a neural Network.

## Testing Set

Once a neural network is trained, it is often tested to measure the neural network’s accuracy.

## Validation Set

The validation set is a dataset used to reduce the likelihood of overfitting of the neural network. The validation set verifies that any increase in accury of the training data includes and increase in accuracy in unseen data by the neural network.

## Neural Network Learning Strategies

There are 2 learning strategies for Neural Networks: Supervised Learning, Unsupervised Learning

REINFORCEMENT LEARNING

## Unsupervised Learning

In Unsupervised learning techniques no target patterns exist. An example of unsupervised learning technique may be a dataset without known answers. Unsupervised Learning typically uses unsupervised learning technique algorithms for self-organizing neural networks. Unsupervised Learning algorithms are thought to be “more closely aligned with what some call true artificial intelligence” (Castle, 2017) An ANN determines structures & patterns in data. A cost function alerts the neural network to how much it is deviating from its target. The network then adjusts its weights for each iteration/epoch of the neural network (AppliedGo, 2016).

## Supervised Learning

Supervised Learning techniques have an output pattern of the network which is compared with target output pattern. They are usually composed of a large set of test data with known results. Each iteration of the dataset, the target is compared with the output result and the weights are adjusted. For example, the teacher feeds student or the ANN with some example data about which the teacher already knows the answers. (Applied Go. 2017). Supervised Learning algorithms linear and logistic regression, multi-class classification, and support vector machines. (Castle, 2017)

## Applications of Neural Networks

Neural Networks have been applied across a range of industries. In financial industries neural networks have been utilised in stock market forecasts, prediction of bankruptcy, credit worthiness and detecting frauds. In medical industries applications of neural networks have been used in diagnosing data intensive diseases in detecting cancer. (Filippo Amato, 2013)

Neural Networks appear to be used most frequently in data mining applications for prediction, classification, identifying change and deviation and detection. (Alyuda.com, 2015)

## Neural Network Types

The various Neural Network types include:

## Single Layer Perceptron

A single layer neural network is composed of 2 or more input nodes and one output node. (Hackernoon, 2017)

## Multi-Layer Perceptron/Deep Feedforward Neural Networks

A Multilayer perceptron is composed of 2 or more input nodes, various hidden layers and 1 or more output nodes. (Hackernoon, 2017)

## Recurrent Neural Network

A Recurrent Neural Network is often referred to as a RNN. The hidden layer neurons in this network has self-connections. A recurrent neural network has larger memory requirements. Hidden layer neurons receive activation from the lower layer as well as it previous activation value. (Fausset, 1994)

## LSTM Neural Network

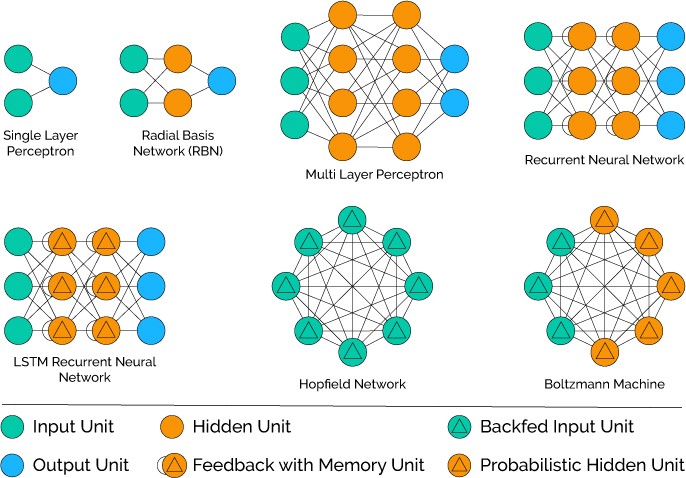
A LSTM Neural Network is a type of RNN neural network in which memory cell is incorporated inside hidden layer neurons is called LSTM network. (Hackernoon, 2017) These neural networks allow information to persist over various iterations. LSTM is an abbreviation for Long Short Memory Neural Networks.

## Hopfield Neural Network

This is a type of network where all neurons are connected to every other neuron via weights. The network is trained with input patterns by setting a value of neurons to the desired pattern. Then its weights are computed, however the weights do not change. Once trained a pattern, the network will converge to the trained pattern (Hackernoon, 2017).

## Boltzmann Machine

The Boltzmann Machine network can operate with or without learning. Without learning, the weights are fixed. The network solves operations by changing the activations (1 or 0) of the units based on a probability distribution (Fausset, 1994). A Boltzmann machine is similar to a Hopfield network except some neurons are input, while other are hidden in nature. The weights are initialized randomly and adjusted throughout back propagation algorithm.



*Figure 2.14 Diagrams of different types of Neural networks* (Hackernoon, 2017)

## Convolutional Neural Network

A Convolutional Neural network is composed of learning weights and biases. Every input layer is composed of a set of neurons where at every input a dot product is performed and move further with the concept of non-linearity. It is a kind of fully-connected type of network that uses SVM/Softmax function as a loss function. (Stack, 2017)

ConvNet architectures assume that the inputs are images, which allows the encoding of certain properties into the architecture. This makes the forward function more efficient to implement and vastly reduce the amount of parameters in the network. (cs231n, 2017)

With a regular fully connected neural network, a large image of size 200x200x3, (200w, 200h, 3 colour channels) would lead to neurons that have 200\*200\*3 = 120,000 weights. Often want several more neurons so the parameters would add up quickly! Full connectivity (Regular neural networks) is wasteful and the huge number of parameters would quickly lead to overfitting.

Convolutional Neural Network have 3D volume of neurons: width, height, and depth. Depth refers to an activation volume not the depth of a full ANN. The ConvNet architecture reduces the full image into a single vector of class scores, arranged along the depth dimension.





*Figure 2.14.1 Regular 3 layer neural network vs a Covnet* (Yangqing Jia∗, 2014)

*Top: A regular 3-layer Neural Network.*

*Bottom: A convolutional neural network* (Yangqing Jia∗, 2014)

A ConvNet arranges its neurons in three dimensions (width, height, depth) in the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. The red input layer holds the image, with its width and height dimensions the same as the image. The depth would be 3 holding the Red, Green, Blue channels of the image.

**Covnet Layers**

* The INPUT layer [32x32x3] stores the pixel values of the image. The width of the image (32 pixels), height of the image (32 pixels) and the Three colour channels RGB (Red Green Blue). (Yangqing Jia∗, 2014)
* The CONV layer computes the neuron output which is connected to a neuron input by performing the dot product between their weights. This may result in volume such as [32x32x12] with 12 filters. (Yangqing Jia∗, 2014)
* The RELU layer performs an elementwise activation function, such as the max(0,x) max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).(Yangqing Jia∗, 2014)
* The POOL layer will performs a down sampling operation of the width and height operations resulting in the volume such as [16x16x12]. (Yangqing Jia∗, 2014)
* The fully connected layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

## The dot product

The Dot Product is written a . b or the product of vectors a x b. The dot product is used to calculate the Dot Product of two vectors to produce a scalar result. (Fausset, 1994)

**a · b** = |**a**| × |**b**|

*Figure 2.15 Dot Product Formula* (Fun, 2014)

## Feed forward Propagation

One neuron layer may only have connections to neurons of other layers - e.g. Perceptron

One direction, from the input layer to the output layer.

## Workthrough of an XOR in feed forward propagation

XOR Truth Table

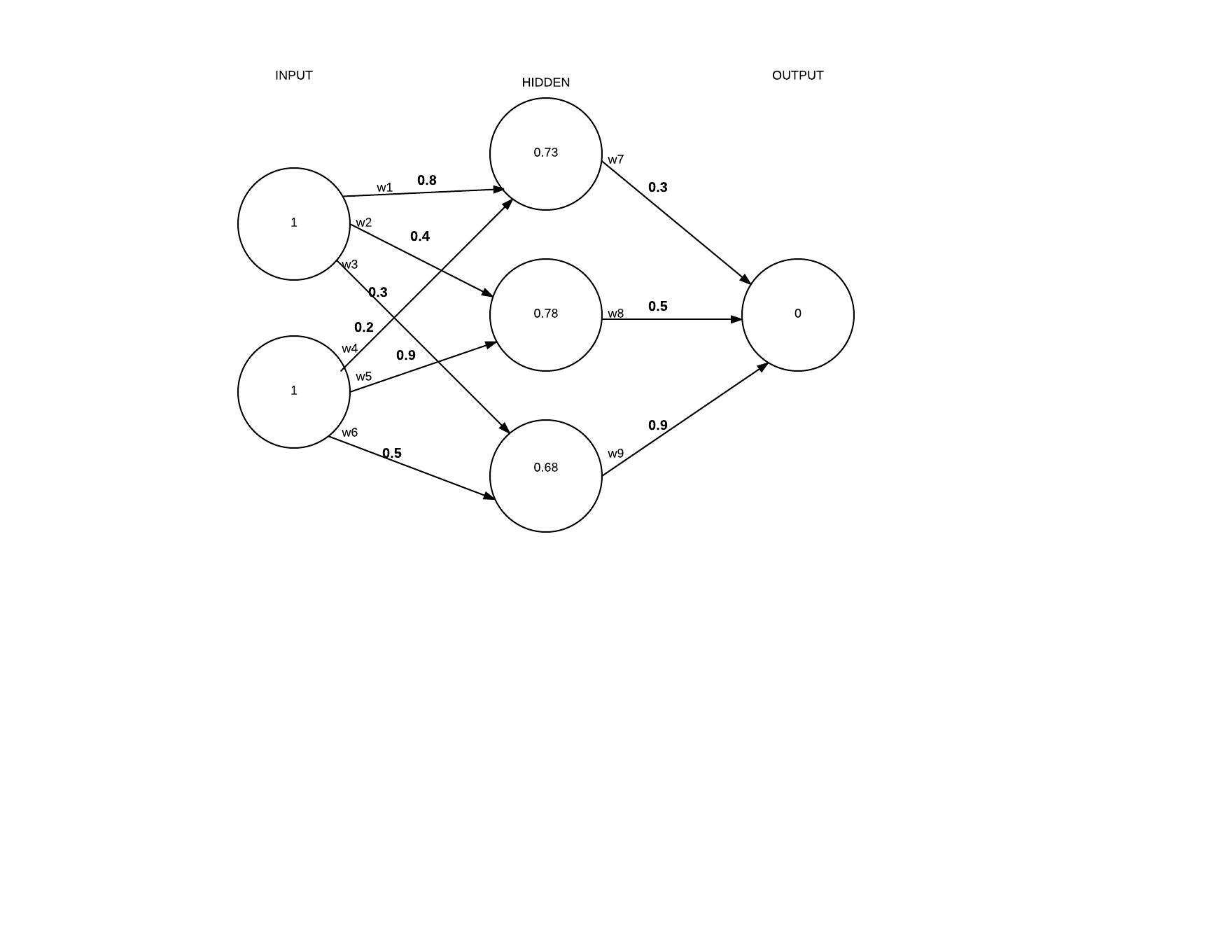
|  |  |  |
| --- | --- | --- |
| INPUTS | | OUTPUTS |
| X | Y | Z |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Steps**

1. Set of random input weights between 0 and 1
2. Random weights between 0 and 1
3. First values of the hidden layer = *Sum(Inputs X Corresponding Weights)*
4. Sigmoid of values of hidden layer from above (See sigmoid function below)
5. *Output = Sum(Hidden layer results X Corresponding Weights)*
6. Sigmoid of final output
7. Margin of Error = expected – calculated

**Sigmoid Function**

N3

**

|  |  |
| --- | --- |
| **INPUT → HIDDEN**  **N3:** ∑w1 . w4= (1 X 0.8) + (1 X 0.2) = 1.0  **N4:** ∑w2 . w9= (1 X 0.4) + ) + (1 X 0.9) = 1.3  **N5:**  ∑w3 . w6= (1 X 0.3) + (1 x 0.5) = 0.8  1.0  1.0 | **HIDDEN → OUTPUT**  **N6:** ∑(N3 . w7) + **:** ∑(N4. W8) + **:** ∑(N5 . w9)  = (1 X 0.3) + (1.3 X 0.5) + (0.8 X 0.9)  = 0.2 + 0.39 + 0.612 = 1.236 |
| **Sigmoid of input to hidden result**  S(1) = 0.73105857  S(1.3) = 0.79583498  S(0.8) = 0.68997448 | **Sigmoid of hidden to output**  S(1.236) = 0.774866989 |
| **Margin of Error:**  0 – 77 = (-0.77) | |

**Delta Learning Rule:** Learning rule for minimising the square error for each training pattern

Mean Squared Error – the squared error divided by the number of output components

**Learning Rate:** This parameter controls the amount by which weights are adjusted during training depending on the type of neural network.

Learning rates – constant

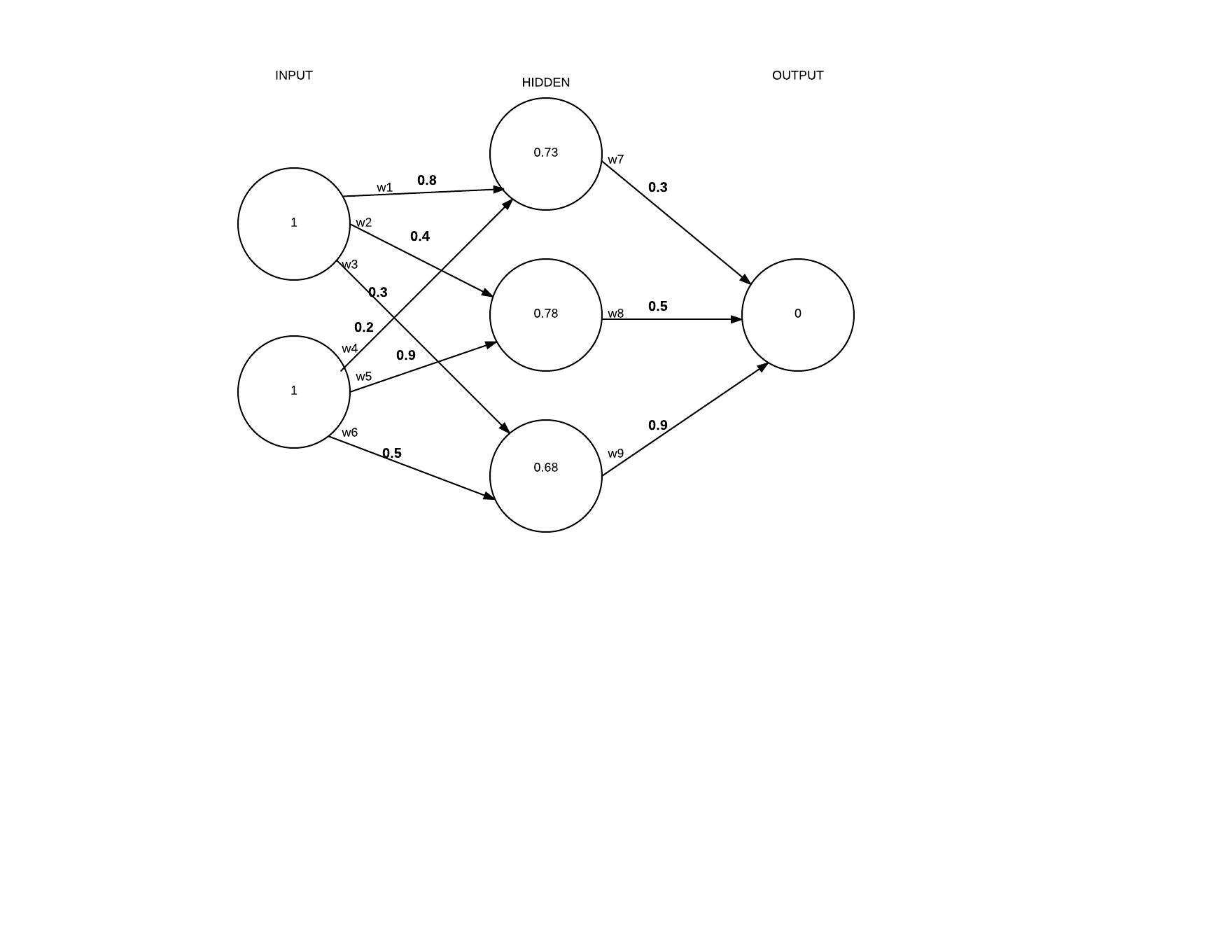
**Momentum** Each step weight adjustments are based on current weight adjustments and the weight change from the previous iteration (Fausset, 1994)

**Clustering:** Group of similar patterns together

**Convergence** Recurrent Neural nets reach convergence when the weights stop changing – state of consistency (Fausset, 1994)

## Backpropagation work through for XOR

A learning algorithm used by multilayer neural nets based on minimising the mean or total squared error over a number of iterations.

1. **Special form of the delta learning rule. Network has one input, one output and at least 1 hidden layer - mainly used for pattern association

**Sigmoid Function**

N3

N1

W7

1.0

|  |  |
| --- | --- |
| INPUT → HIDDEN  H1 = 0.73105857863  H2 = 0.78534983  H3 = 0.68997448 | HIDDEN → OUTPUT  1.235  Output layer MOE -0.77 |

Delta Output Sum = S(Output) X MOE (Margin of Error)

= S(1.235) x (-0.77)

= -0.134398

**Output to Hidden**

Delta Weights = Delta Output Sum x Hidden layer

-0.134398 . [0.731058, 0.78583983, 0.68997448] . [-0.09825, -0.10256, -0.092]

**W7 →** 0.3 - 0.09825 = 0.202

**W8 →** 0.5 - 0.10256 = 0.394

**W9 →** 0.9 – 0.094 = 0.806

**Hidden to Input**

Delta Hidden Sum = Delta Output Sum x Hidden To Outer Weights x S(HiddenSum)

-0.134 . [0.3, 0.5, 0.9] X S([1, 1.3, 0.8])

= [-0.0403, -0.0672, -0.1209] \* [0.1966, 0.1683, 0.2129]

= [-0.0079, -0.0113, -0.0259], [-0.0079, -0.0113, - 00.259]

**W1 →** 0.8 – 0.079 = 0.7921

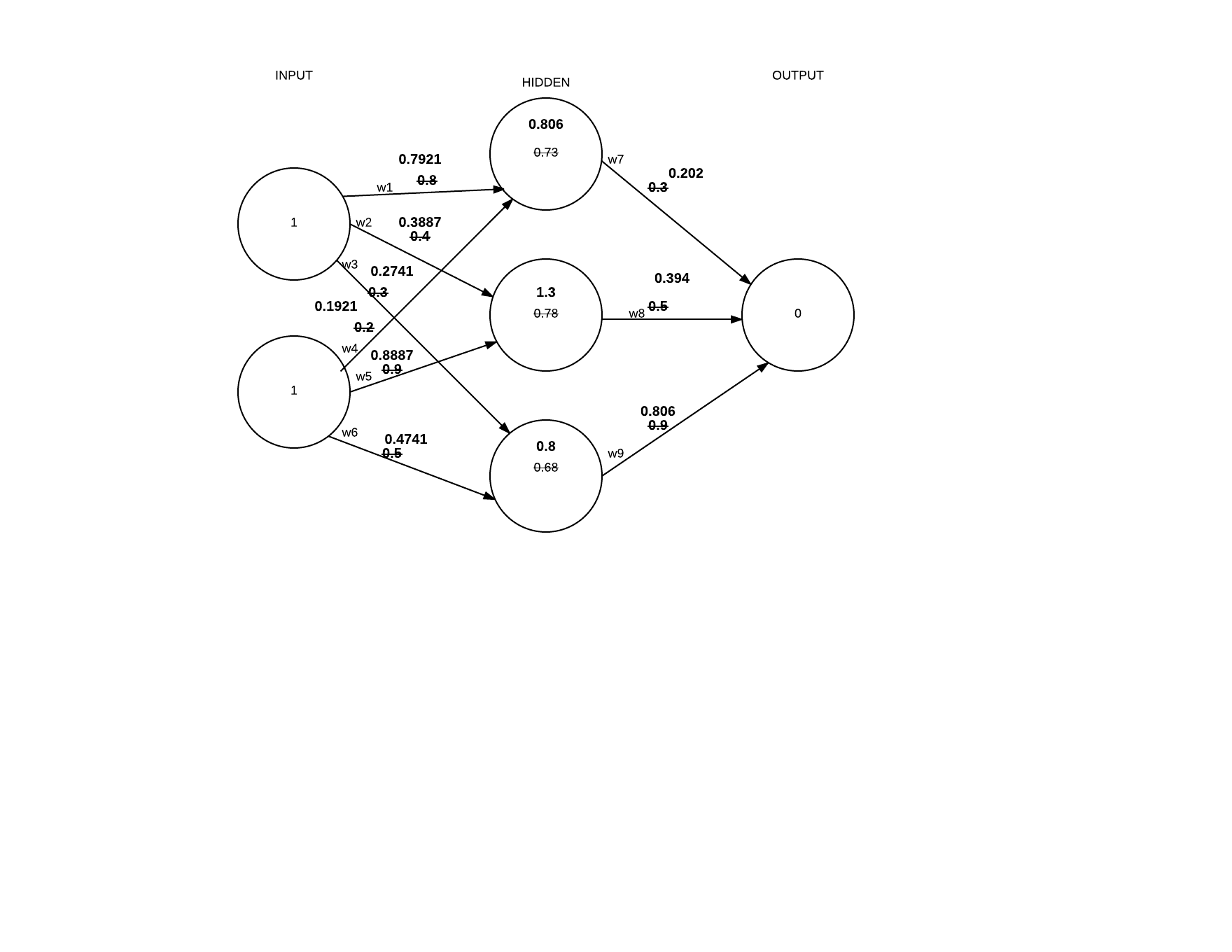
**W2 →** 0.4 – 0.0113 = 0.3887

**W3 →** 0.3 – 0.0259 = 0.274

**W4 →** 0.2 - 0.0079 = 0.1921

**W5 →** 0.9 – 0.0113 = 0.8887

**W6 →** 0.5 – 0.0259 = 0.4741



## Neural Network Libraries

Neural Networks are most commonly developed using Neural network libraries/APIs or frameworks. Libraries allows developers to use code in an already functional application in a stand-alone fashion. Libraries assist in abstracting the lower level more complex details of code and libraries allow for faster implementation.

## NNabla by Sony

Neural Network Libraries is a deep learning framework intended for research, development and Production on desktop PCs, HPC clusters, embedded devices and production servers. Developer can uses Python and C++ APIs for developing Logistic Regression, Multi-Layer Perceptron’s, Convolutional Neural Networks and Recurrent Neural Networks. However, this framework is relatively new since August 2017.

## Tensorflow

TensorFlow was originally developed by the Google Brain Team within Google's Machine Intelligence research organization. (TensorFlow, 2017). Nodes in Tensorflow represent the various activation functions, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. This API allows developers to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow is written with Python 3 and has many similarities to Python.

The TensorFlow library does numerical computation using data flow graphs. The graph nodes represent mathematical operations and graph edges represent Multidimensional data arrays/tensors which flow between then. As the computations can be heavy on one system, TensorFlow allows the developer deploy the computation to multiple CPUs or GPUs on desktops, servers or mobile devices easily.

TensorFlow also has a data visualization toolkit called TensorBoard.

**Tensors** are multidimensional arrays of primitive values. The Tensor object is  tf.Tensor. TensorFlow programs work by first building a graph of tf.Tensor objects, by describing how each tensor is computed based on the other available tensors and then by running parts of this graph to achieve the desired results.A tf.Tensor has the following properties: (TensorFlow, 2017).

* a data type (float32, int32, or string, for example)
* a shape/rank

[2., 3., 4.] # a rank 1 tensor; a vector with shape [3]  
[[2., 3., 4.], [5., 6, 7.]] # a rank 2 tensor; a matrix with shape [2, 3]  
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]

*Figure 2.17 – Examples of Tensors with Ranks -* (TensorFlow, 2017)

The core of TensorFlow programs include building and running a computational graph.

A **computational graph** is a series of TensorFlow operations arranged into a graph of nodes Each node takes 0 to multiples tensors(Multidimensional array) as inputs and produces a tensor as an output.

Tensorflow constnats take no inputs. The value is stored internally and does not change. Two floating point Tensors node1 and node2 as follows:

node1 = tf.constant(3.0, dtype=tf.float32)  
node2 = tf.constant(4.0) # also tf.float32 implicitly  
print(node1, node2)

 Estimators are high level utilities which manage Tensorflow graphs and sessions. Estimators encapsulate the following actions:

* training
* evaluation
* prediction
* export for serving

[Variables](https://www.tensorflow.org/programmers_guide/variables) detail how to represent shared, persistent state in your program. a tf.Variable exists outside the context of a single session.run call.

Higher-level APIs such as [tf.estimator.Estimator](https://www.tensorflow.org/api_docs/python/tf/estimator/Estimator) and Keras hide the details of graphs and sessions from the end user

Dataflow graphs are TensorFlow's representation of computations as dependencies between operations.

Sessions are TensorFlow's method of running dataflow graphs across one or more local or remote instances

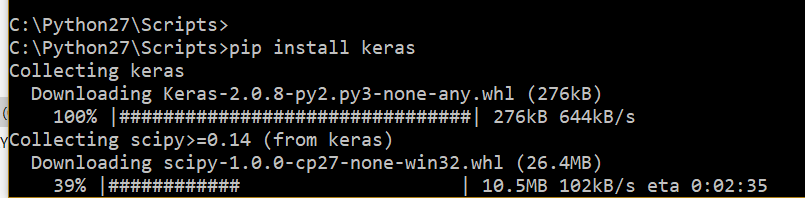
## Caffe

Caffe, a deep learning framework developed with a focus expression, speed and modularity by Berkely AI Research centre.

## Keras

Keras is a high-level neural networks API, written in Python and can be used with TensorFlow, CNTK, or Theano. Keras was developed with a focus on enabling rapid application development.  Keras is a useful deep learning library allowing for easy and fast prototype development. Keras also supports various neural network types such as Convolutional Neural Networks, Recurrent Networks etc. Keras also runs on CPU and GPU (Keras, 2017)

*Keras Installation via pip*



# Chapter 2 – Biometrics

## Introduction

Biometrics is the measurement and analysis of physical and behavioural characteristics a person. Biometrics are used for identification and access control of individual’s unique physical or behavioural characteristics. “The term "biometrics" is derived from the Greek words "bio" meaning life and "metric" meaning to measure”. (Target, n.d.)

## Unimodal & Multimodal systems

Unimodal systems use one physical or behavioural biometric trait. Multimodal systems use more than one physical or behavioural biometric traits for identification and verifications. (Thepade & Manade, 2017)

## Biometric Software/Hardware

## Fingerprint Scanners

## Biometric Authentication

In recent years, biometric authentication has been growing in use. Passwords represent a dated form of authentication requiring ever increasing frustrating parameters. For example, currently most passwords require 12 characters, contain capital letters, a number, a special character etc. (BrainTree & TechCrunch, 2016).

Physical and behavioural biometrics have been proven to solve issues that traditional forms of cybersecurity do not. Malware, Social Engineering, remote access Trojan attacks have been proven to compromise the effectiveness of traditional forms of authentication. Traditional forms of authentication include PINs, passwords, tokens, device ids, geolocation verification and so on. Biometric authentication coupled with the traditional forms of authentication may promise a more secure authentication for applications in the near future. (Turgeman, 2017) The comprimisation of PINs and passwords etc are paving a way for 'Automated Biometrics' When choosing a form of biometrics for authentication the following factors should be considered:

* Accurate Identification
* Accountability
* Easy & Safe to Use
* User Friendliness & Intrusiveness
* Security
* The physical characteristic cannot change over time
* The physical characteristic must identify a unique person

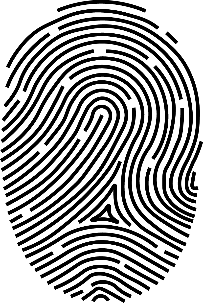
## Physical Biometrics

Physical Biometrics systems most often use DNA, Face, Hand geometry, vein, voice, retina and iris of the eye, ear features etc.

## Fingerprint Identification

Fingerprints are formed by the fourth month of foetal development and remain the same through life, however the fingers do grow and subsequently get larger. (technovelgy.com, 2015)

Fingerprint biometrics are detailed, unique, difficult to alter and durable over a lifetime. A fingerprint is defined as a “pattern of ridges and valleys on a fingertip”. (Thepade & Manade, 2017). The black areas of a fingerprint are called ridges and white space in between are called valleys. (Thakkar, 2015)



*Figure 3.5 Diagram of a fingerprint showing ridges (black marks) and valleys (whitespace)* (biometricsolution.com, 2018)

Fingerprint patterns are most commonly divided into 3 pattern types: arches (tented arch), loops (right loop, left loop) and whorls. The loop fingerprint pattern is the most common fingerprint pattern. (Shrein, 2018)

The many advantages of fingerprints as a form of biometric authentication include: very high accuracy, uniqueness, one of the most developed in biometrics, is easy to use, small database storage required and it is standardized. However, factors which may decrease the reliability of fingerprints uses in biometrics include dry or dirty skin and age. Children’s fingerprint size may change quickly. Also fingerprints are not identifiable from a distance, require user’s consent and may be considered intrusive by the individual.



*Figure 5.1 3 types of fingerprint patterns* (handlines.ie, 2005)

**Loops**

Loops are fingerprint pattern that curve back on themselves to form a loop shape. Loops are the most common fingerprint pattern accounting for approximately 60 percent of fingerprint pattern types. (forensicsciencesimplified.org, 2017)

**Whorls**

Whorls are fingerprint patters that form circular or spiral patterns. There are four categories of whorls: plain concentric circle, central pocket loops, double loops(S shaped) and irregular shaped whorls. Whorls account for 35% of fingerprint patterns.

**Arches**

Arches are fingerprint patters that form wave like patterns. There are 2 categories of arches: plain arches and tented arches. Arches are the rarest type of fingerprint accounting for 5% of all pattern types. (forensicsciencesimplified.org, 2017)

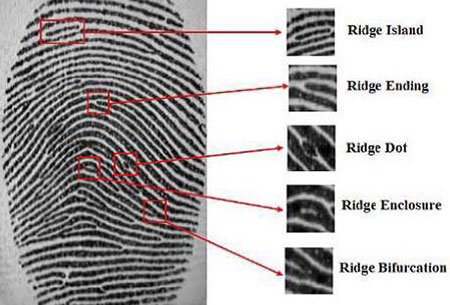
**Fingerprint uniqueness**

Fingerprints have been proven be unique as no two people have exactly the same fingerprints. Even identical twins, with identical DNA have different fingerprints. As fingerprints are so unique, they have been proven very effective in biometric security and forensic science. (Thepade & Manade, 2017)

**Minutiae Features**

Minutiae refer to specific points in a fingerprint, these are the small details in a fingerprint that are most important for fingerprint recognition. The Minutiae of a fingerprint are the points where the ridges lines end or branch out in a fingerprint. (Thakkar, 2015) There are various minutiae types such as:

* **Ridge ending** is where the ridge ends.
* **Ridge bifurcation** is where a ridge branches out into two or more ridges.
* **Ridge dots** are shorter than ridges.
* **Ridge islands** are slightly longer than dots and often reside between a ridge bifurication.
* **Ponds or Lakes** **or Valleys** are the white space in between two diverging ridges. (Thakkar, 2015)



*Common Minutiae patterns of a fingerprint* (Thakkar, 2015)

Hand geometry

Hand Geometry biometrics is based on the shape of a hand. Hand geometry considers the size of the palm, length and width of the fingers, distance between the knuckles, etc. Hand geometry data is inexpensive and not intrusive to collect. Lighting of image is not an issue, unlike obtaining fingerprint data. The hand geometry is not unique and cannot be used alone as a form of identification in identification systems. However, hand geometrics in biometrics are not unique and it is not ideal for growing children. Jewellery rings, arthritis, etc etc may pose a challenge in hand geometry biometrics. (360Biometrics.com, 2016)

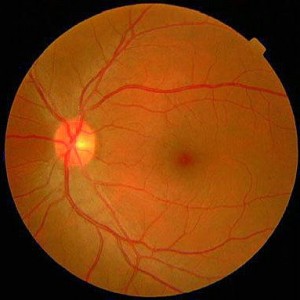
## Palm Vein

Palm Vein biometric data is obtained using infrared beams and the veins are returned as black lines. Palm veins have been proven to have a high-level authentication accuracy due the nature of complex patterns of the vein. Also, as vein patterns are internal, it would be difficult to forge a palm vein biometric system. (technovelgy.com, 2015)

## Retina

A retina scan analyses the capillary blood vessels located in the back of the eye using a low intensity light to take an image. (technovelgy.com, 2015) The pattern of capillary blood vessels in the retina remains the same throughout a person’s lifespan. The low intensity light traces a standardized path on the retina. (Trader, 2012)

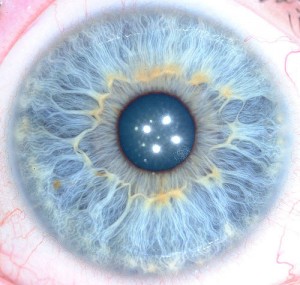
Retina scan data is equivalent in uniqueness to fingerprints, however, as retinae are internal organs, they’re less susceptible to modification. Also, certain eye-related medical conditions and diseases, such as cataracts and glaucoma, may compromise a retinal scan, as the blood vessels can be obscured. (King, 2013)



*Figure 3.5.1 The retina* (Trader, 2012)

## Iris scan

The iris is the coloured circle surrounding the pupil. The function of the Iris is to change the size of the pupil and allows different amounts of light to enter the eye. An iris scan analyses the rings, furrows and freckles in the iris. More than 200 points of the iris scan are used for comparison. (technovelgy.com, 2015).Iris scans are acquired using infrared illumination taking images of the iris. The iris is then fed into mathematical algorithms to determine an individuals identity. (Trader, 2012)



*Figure 3.5.2 The Iris* (Trader, 2012)

## FacIAL recognition

Facial recognition in biometrics constitute as the size and shape of facial characteristics and their relationship to one another. Facial recognition uses relatives between common landmarks on the face ie. Eyes to nose to generate a unique “faceprint”. (technovelgy.com, 2015). Facial recognition is frequently used in security analysis as user consent is not required. However, facial recognition is not the most unique form of biometrics: For example twins may have the same face. Also human’s faces are subjective to change as they age.

## Signature

Signatures are rarely used in biometrics as a user may not sign their name the same way each time and can easily be forged.

## Voice

Voice Biometrics involve analysing the pitch, tone, cadence, and frequency of a person's voice.

## DNA

DNA is a popular biometric used in forensics and healthcare for identifying a person. Every individual has its own individual map for every cell made and can be found in every body cell making it dificult to omit at crime scenes. DNA can be obtained via blood, hair, finger nails etc. and is unique to a person. However, obtaining DNA is an intrusive and costly process (ex-sight.com, 2009).

## Behavioural Biometrics

Biometric that analyses users by constantly collecting information with how a user interacts with a system. Behavioural biometric examples include: keystrokes, mouse movement, hand eye coordination, pressure, handshakes, navigation, finger movements etc. (Turgeman, 2017)

# Chapter 3 – Authentication, Authorization and Encryption

## Authentication

According to OWASP (Open Web Application Security Project ), Authentication is defined as “the process of verification that an individual, entity or website is who it claims to be. Authentication in the context of web applications is commonly performed by submitting a user name or ID and one or more items of private information that only a given user should know.”. (OWASP, 2017)

## Sessions

Servers are stateless meaning every request from a new client process is not maintained. Servers cannot handle a new request if they cannot identify its origin. (Pankaj, 2017). Session management is a process by which a server maintains the state of an entity interacting with it. The server stores the entity information to process requests and send responses. (OWASP, 2017)

## Passwords

The application must determine the minimum length and maximum length of a password. Passwords shorter than 10 characters are considered to be weak (NIST SP800-132). Passwords must also have a set of complexity requirements. In accordance with OWASP suggestions, a password must meet at least 3 out of the following 4 complexity requirements (OWASP, 2017):

* at least 1 uppercase character (A-Z)
* at least 1 lowercase character (a-z)
* at least 1 digit (0-9)
* at least 1 special character (punctuation) — do not forget to treat space as special characters too
* at least 10 characters
* at most 128 characters
* not more than 2 identical characters in a row (OWASP, 2017)

### Prevent Brute-Force Attacks

Attacker ability to guess passwords to compromise the account security.

# Methodology & Design

## Research Undertaken

Research undertaken has highlighted neural networks, the parts involved in a neural network such as neurons, activation functions, connections, bias, threshold, weights, different neural network layers, neural network learning rules, the training and testing process of a neural network, neural network learning strategies, feed forward and back propagation algorithms.

Research also undertaken researched into the area of biometrics. The research conducted analysed the different types of physical and behavioural biometrics and the advantages of each.

Throughout the process of research, research concluded that physical biometrics particularly, fingerprints are the most effective in the identification of a person. Fingerprints are also unique to an individual, difficult to copy/compromise and they are the least intrusive to obtain. In order to train and test a neural network, a large sample of biometric data is required.

## Research Question

Using Neural Networks, can biometrics be used with neural networks as an advanced form of authentication ? In addition, what type of neural network should be used to implement a biometric neural network system and do neural networks trump the traditional algorithms for detecting biometrics.

## Vision Document

### Introduction

The purpose of this Vision document is to give a high level overview of my final year project: Biometric Neural Networks as an advanced form of authentication. I plan to develop an Artificial Neural Network to authenticate a user using their biometric data such as fingerprints, facial recognition etc.

### Scope/Outline

Biometric Neural Networks identifies patterns in a training dataset to better identify a user’s biometric data e.g. fingerprints, facial recognition, and gestures etc. Biometric Neural networks will be written in Python using the Tensorflow Neural Network library which provides abstractions and prevent the developer from having to develop a neural network from scratch. The design phase will start in September 2017 and will produce a Project Vision Statement, a Project Proposal Document, a Project Plan, Project Specification, Formal Design, Prototype, and a Design Presentation. The design phase will terminate on December 13th 2017.

### Product Overview/Features

To prove the research question: Neural Networks as a form of authentication involves developing a convolutional neural network to classify images. On successful completion of developing a convolutional neural network with Tensorflow, I will feed in a biometric dataset of fingerprints and then possibly a dataset of irises(eye) to train the dataset. I will then feed in a random image of a fingerprint from the dataset for a positive result. I will then feed in a random image from online of a fingerprint to test the accuracy. I plan to compare activation functions in the Neural Network to compare the accuracy. I also hope to use algorithms for comparing fingerprints to test it against the Biometric neural network.

* Locate a dataset of biometric data – fingerprints
* Identify & Learn How to an API to create an Artificial Neural
* Identify the type of Neural Network required
* Develop a Neural Network using the Neural Network API
* Integrate a dataset into the Neural Network
* Train the Neural Network
* Test the Neural Network
* Test different Activation Functions
* Improve Neural Network accuracy

### User requirements, functional and non-functional

**Moscow Method**

MoSCoW is an abbreviation in terms of must, should, could and would. The Moscow Method is used to develop an understnding of the requirements and priority of a project.

M – Must states that this feature is a requirement in the project to guarantee project success. S - Should states that this requirement must be included if possible, but project success does not rely on it. C – Could means that this requirement may be included if it does not affect anything else on the project W - Would like to have this requirement later, but it won’t be a feautre in the project delivery at this time. (Haughey, 2014)

The user supplies their fingerprint

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement** | **Priority (Moscow)** |
| 001 | Locate a dataset of biometric data – fingerprints | Must |
| 002 | Identify & learn how to use an API to create an Artificial Neural Network | Must |
| 003 | Identify the type of Neural Network required | Must |
| 004 | Train the Neural Network | Must |
| 005 | Test the Neural Network | Must |
| 006 | Test different Activation Functions | Should |
| 007 | Improve Neural Network accuracy | Must |
| 008 | Test Neural Network with an image within the dataset and image from online | Should |
| 009 | Use fingerprint recognition algorithms to identify whether the neural network or traditional algorithms are more accurate. | Could |
| 010 | Use Different datasets | Could |
| 011 | Research fingerprint storage security | Could |
| 012 | Read in images from biometric scanner | Could |
| 013 | Create a UI | Could |

### Datasets

For my final year proect I will require a biometric datasets for fingerprints:

The various datasets available are all from the NIST which is The National Institute of Standards and Technology. The NIST is one of the US’s oldest and most developed physical science laboratories. The NIST encompasseses the ITL (Information Technology Laboratory), a research laboratory within the NIST. They supply various biometric fingerprint datasets for public use as listed below. (NIST, 2018)

The NIST Biometric Image Software (NBIS) was developed by the National Institute of Standards and Technology (NIST) for the FBI and Department of Homeland Security (DHS) in the United States. A feature of the NBIS was a neural-network based fingerprint pattern classification system called, PCASYS. PCAYS categorizes a fingerprint image into the class of arch, left or right loop, scar, tented arch, or whorl using an Artificial Neural Network. (NIST, 2018)

**Special Database 4** is a dataset of 400 8 bit Gray Scale Images of Fingerprint Image Groups. The Fingerprint Groups includes 400 samples each of the five fingerprint classifications: Arch, Left & Right Loops, Tented Arch and the Whorl. This dataset is commonly used for fingerprint classification research with algorithms and system training and testing.(NIST, 2018)

Other Datasets provided by the NIST include:

**Special Database 9** is a NIST dataset of 8-bit Gray Scale Images of Mated Fingerprint Card Pairs. (NIST, 2018)

**Special Database 10** is a NIST dataset of Supplemental Fingerprint Card Data (SFCD) for NIST Special Database 9 with include rolled fingerprints and harder to find fingerprint classifications like arch, tented arch and low count loops. (NIST, 2018)

**Special Database 14** is a NIST dataset of Mated Fingerprint Card Pairs 2.  
 This dataset includes 2,700 ten-print card pairs of rolled fingerprints (no plain impressions). (NIST, 2018)

**NIST Special Database 27 and 27A** is a dataset of fingerprint minutiae from latent and matching tenprint images. (NIST, 2018)

**Special Database 29** is a NIST dataset of Plain and Rolled Images from Paired Fingerprint Cards in 500 pixels per inch. (NIST, 2018)

### Solutions

For developing a neural network, I plan on using Tensorflow which is an open-source software library for Machine Intelligence. Tensorflow was initially developed by Google for their Google Brain project and then made opensource. Tensorflow is a Python language. I will use Anaconda. Anaconda is an open source distribution of Python which includes preinstalled modules for use. Using Tensorflow I will develop a Convolutional Neural Network as a CNN specialises in image recognition. I will feed in a dataset of biometric data to train the neural network. I will use NIST Special Database 4 which is a dataset of 400 8 bit Gray Scale Images of Fingerprint Image Groups. I will disregard the Image Groups from this dataset and use only the images and create my own set of labels for each unique fingerprint in the dataset.

I plan to develop the CNN using PyCharm – a Jet Brains IDE.

I will used git for version control.

## Project Specification

### User Stories

1. *ID = feature ID / user story ID*

|  |  |
| --- | --- |
| ID | 001 |
| TITLE | Obtain a biometric fingerprint dataset |
| DESCRIPTION | As a software engineer,  I want to obtain a biometric dataset of fingerprints,  So that I can train and test the Convolutional Neural Network |
| PRIORITISATION | Must |
| ACCEPTANCE CRITERIA | A data set of clean data of fingerprints 512px x 512px |

|  |  |
| --- | --- |
| ID | 002 |
| TITLE | Develop a Convolutional Neural Network using Tensorflow |
| DESCRIPTION | As a software engineer,  I want develop a Convolutional Neural Network using Tensorflow,  So that I can identify fingerprints |
| PRIORITISATION | Must |
| ACCEPTANCE CRITERIA | The software engineer develops a CNN in Tensorflow accepting an image as an input |

|  |  |
| --- | --- |
| ID | 003 |
| TITLE | Train an Artificial Neural Network using a training set of the fingerprint dataset by feeding it through the Convolutional Neural Network |
| DESCRIPTION | As a software engineer,  I want to train a Convolutional Neural Network using Tensorflow with a training set,  So that I can build up the accuracy for the test set |
| PRIORITISATION | Must |
| ACCEPTANCE CRITERIA | The software engineer obtains a successfully trained Convolutional Neural Network. |

|  |  |
| --- | --- |
| ID | 003 |
| TITLE | Test an Artificial Neural Network using the test set of the fingerprint dataset by feeding it through the Convolutional Neural Network |
| DESCRIPTION | As a software engineer,  I want to test a Convolutional Neural Network using Tensorflow with a test set,  So that I can accurately authenticate users by fingerprints |
| PRIORITISATION | Must |
| ACCEPTANCE CRITERIA | The software engineer obtains a successfully tested Convolutional Neural Network with a relatively high accuracy |

### Use Cases and Narratives

Use Case Description

User

<<includes>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** |  | |
| **Use Case Id** | 0001 | |
| **Priority** | High | |
| **Source** |  | |
| **Primary Business Actor** | User | |
| **Other Participating Actors** |  | |
| **Description** | The user will enter a biometric data – fingerprint from a data set or via a scanner. The application will be trained and use the fingerprint entered to test whether the user exists or not. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The user will launch the Biometric Neural Network UI application.  **Step 5:** The user will enter their biometric data. | **Step 2:** Th application will be displayed to the user upon a successful application start up.  **Step 3:** The application will train the neural network with the training data from the dataset.  **Step 4:** The application will ask the user to supply their biometric data – ie. Fingerprint.  **Step 6:** The Neural Network will test the user entered fingerprint against the trained network and test if the user is valid. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  | **Step 5:** The user enters invalid biometric data | **Step 6:** The system cannot authenticate a user |
| **Conclusions** | The Neural Network authenticates user with biometric data. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Risk Analysis

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Risk Source** | **Probability**  Low Med High  1-3 4-7 8-10 | | | **Impact**  Low Med High  1-3 4-7 8-10 | | | **Result**  Prob. X  Impact | **Impact Areas**  Cost. Sch. Perf. | | | **Risk Response** |
| 1 | Difficult learning Tensorflow |  | 6 |  |  | 7 |  | 42 |  | x | x | Start Tensorflow research as soon as possible.  Follow Tensorflow tutorials  Read Tensorflow documentation |
| 2 | Breakdown in communications between developer and supervisor |  | 5 |  |  | 5 |  | 25 |  | x | x | Have a clear understanding what the supervisor wants, complete work on schedule |
| 3 | Difficulty developing a Convolutional Neural Network |  | 6 |  |  | 4 |  | 24 |  | x | x | Follow tutorial and try get them working before the development phase commences |
| 4 | Difficulty finding a biometric dataset for training and testing the neural network | 3 |  |  |  |  | 8 | 24 |  | x | x | Develop own dataset of biometric data now by web scraping applications |
| 5 | Difficulty in incorporating dataset into neural network |  |  | 8 |  | 5 |  | 40 |  | x | x | Complete research and try to solve fitting of dataset into neural network |
| 6 | Neural network slow to train | 3 |  |  |  | 4 |  | 12 |  | x | x | Output training to an external file ie. Csv file or |
| 7 | Neural Network Accuracy |  | 4 |  |  |  | 5 | 20 |  | x | x | Add more layers to improve complexity or use different activation functions or research different neural network types |
| 8 | Difficulty developing with Tensorflow |  | 5 |  |  |  | 7 | 35 |  | x | x | Revert to a lower level neural network api such as TFLearn or Keras |
| 9 | Implementing a database time constraint | 3 |  |  |  | 5 |  | 15 |  | x | X | Leave the database – no real need when the dataset exists |
| 10 | Issues integrating a scanner |  | 7 |  | 4 |  |  | 28 |  | x | x | Drop the use of the scanner and get an image dataset from the web |
| 11 | Issues with CPU or training Neural Network |  | 7 |  | 5 |  |  | 35 |  | x | x | Use a GPU for training the neural network or train on a server |
| 12 | Dataset requires significant preprocessing to ensure clean data | 2 |  |  | 5 |  |  | 10 |  | x | x | Follow Big Data’s modules steps for ensuring the dataset is properly sanistised |
| 12 | Project too simple |  |  | 8 |  |  | 7 | 56 |  | x | x | Integrate a database and secure encryption of biometric data |
| 13 | Project Over Schedule |  |  | 8 |  |  | 7 | 56 |  | x | x | Work in accordance with the project plan |

### Class Diagrams

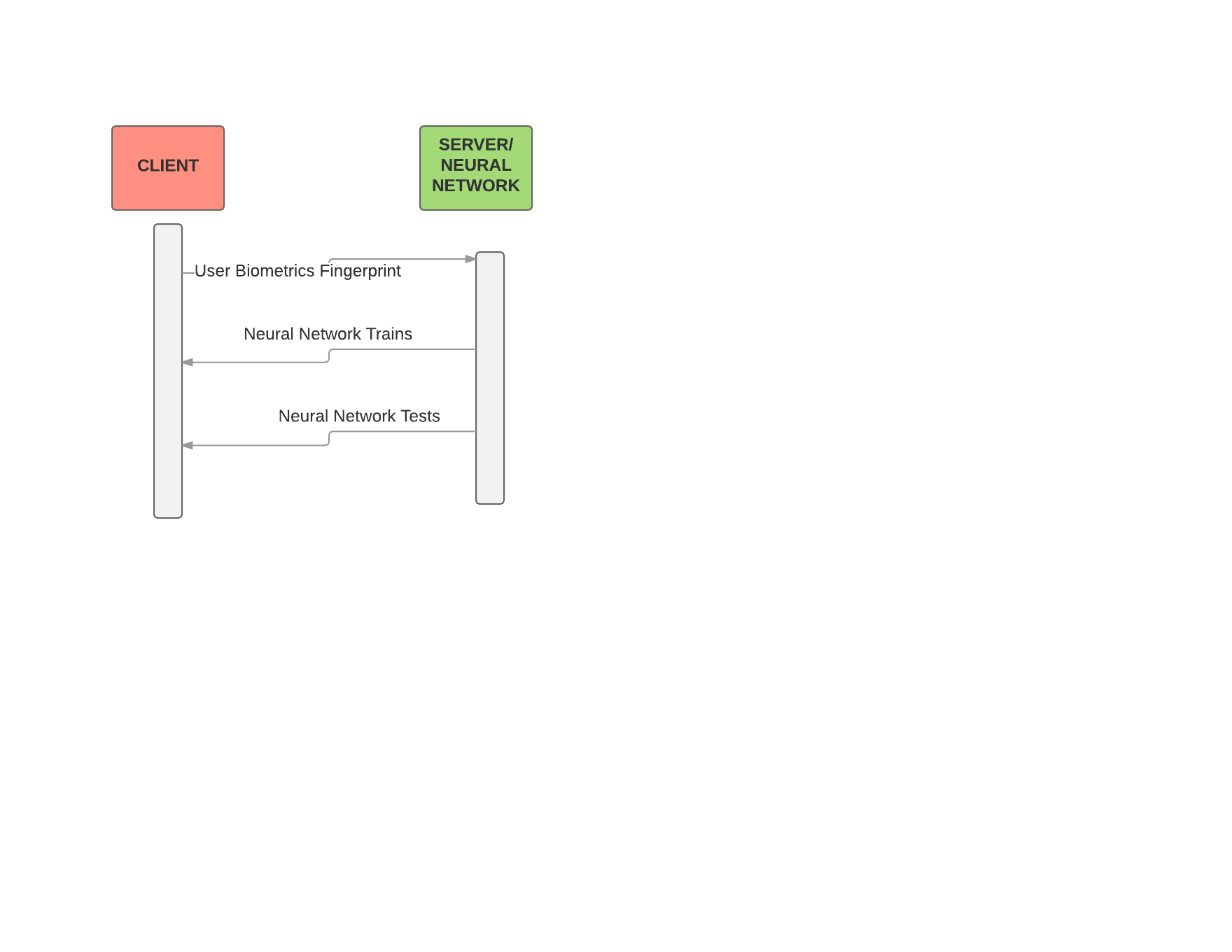
### Project Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Prototype | Start Date | End Date | Hours |
| Identify an API for Neural Networks | 1 October 2017 | 9 October 2017 | 14 hours |
| Learn Tensorflow | 10 October 2017 | 24 October 2017 | 24 hours |
| Develop a Multi Layer Neural Network | 25 October 2017 | 30 October 2017 | 12 hours |
| Convolutional Neural Network | 1 December 2017 | Ongoing |  |
| Integrate Dataset | 5 January 2017 | Ongoing |  |
| Improve Train & Test Accuracy |  |  |  |

## Design Phase

### Sequence Diagram

The following diagram shows the sequence diagram for the ANN. The user enters their biometric data to the neural network. The Neural Network is trained and then tested using the clients biometric data.



### Relational Schema/ Entity Relational Diagram

### Work Completed

|  |  |  |
| --- | --- | --- |
| Task Number | Details | Status |
| Use Keras | Used Keras to create a simple 3-layer neural network | Complete |
| Implement Keras Multilayer Neural Network | Developed a multilayer neural network using Keras | Complete |
| Create a simple Neural Network with Tensorflow | Developed a Neural Network to demonstrate the XOR | Complete |
| Develop a Convolutional Neural Network | Developed a Neural Network with MNIST Dataset | Complete |
|  |  |  |
|  |  |  |

### Screenshots

### Future Extensions

Integrate Hardware – Use fingerprint scanners, cameras etc. instead of using a dataset for a more real life project.

Develop a Database for the fingerprint obtained by scanners or web scraped

Ensure encryption of biometric data – encrypt biometric data ie. Fingerprint for storage in the database

Another mode of biometric authentication – maybe facial recognition or keystrokes for characteristic biometric authentication

# Implementation

Sprints

|  |  |  |  |
| --- | --- | --- | --- |
| Prototype | Start Date | End Date | Hours |
| Identify an API for Neural Networks | 1 October 2017 | 9 October 2017 | 14 hours |
| Learn Tensorflow | 10 October 2017 | 24 October 2017 | 24 hours |
| Develop a Multi Layer Neural Network | 25 October 2017 | 30 October 2017 | 12 hours |
| Convolutional Neural Network | 1 December 2017 | Ongoing |  |
| Integrate Dataset | 5 January 2017 | Ongoing |  |
| Improve Train & Test Accuracy |  |  |  |

# Sprint 1

Set up environment

### **Sprint 1: Project Setup (Software, Environments, Frameworks, Version Control System)**

|  |  |  |
| --- | --- | --- |
| Sprint No. | Start Date | Finish Date |
| 1 |  |  |

|  |  |  |
| --- | --- | --- |
| **Task Number** | **Details** | **Status** |
| **1** | Setup the Python environment of Python 3 /Anaconda | Complete |
| **2** | Install a Python IDE – Pycharm by JetBrains Community Edition | Complete |
| **3** | Configure the Python Interpreter for a Python Project | Complete |
| **4** | Install modules for project – Keras, Tensorflow etc. | Complete |
| **5.** | Set up local Version Control System on Git | Complete |
| **6.** | Set up remote Version Control System repository on Github | Complete |
| **7.** | Learn Python Basics – syntax, using modules, functions, control structures etc. | Complete |

Python 3 was released in 2008

Anaconda is a open source distribution of Python with 150 data science packages already installed for developers convenience.

Pycharm is a Python IDE by JetBrains. JetBrains have many other IDEs and tools such as JetBrains for Java, PhpStorm for PHP, Teamcity Continuous Integration tools etc. Pycharm offer a Professional, Educational and Community Edition of Pycharm. Pycharm Community Edition is free and open source. Pycharms features include

* Python Editor
* Graphical Debugger & Test runner Sprint
* Learn Tensorflow basics
* Create a multilayer neural network
* Develop a convolutional neural network

Python modules are scripts of Python code. A module can imported with specific functions, classes and variables. Python modules can be imported as follows:

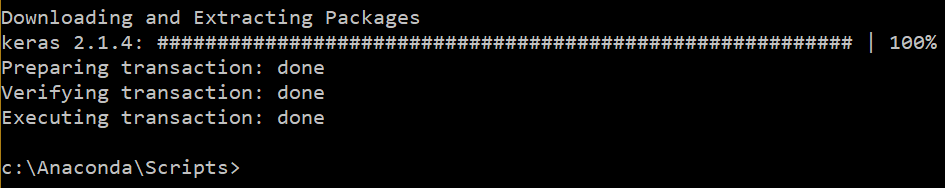
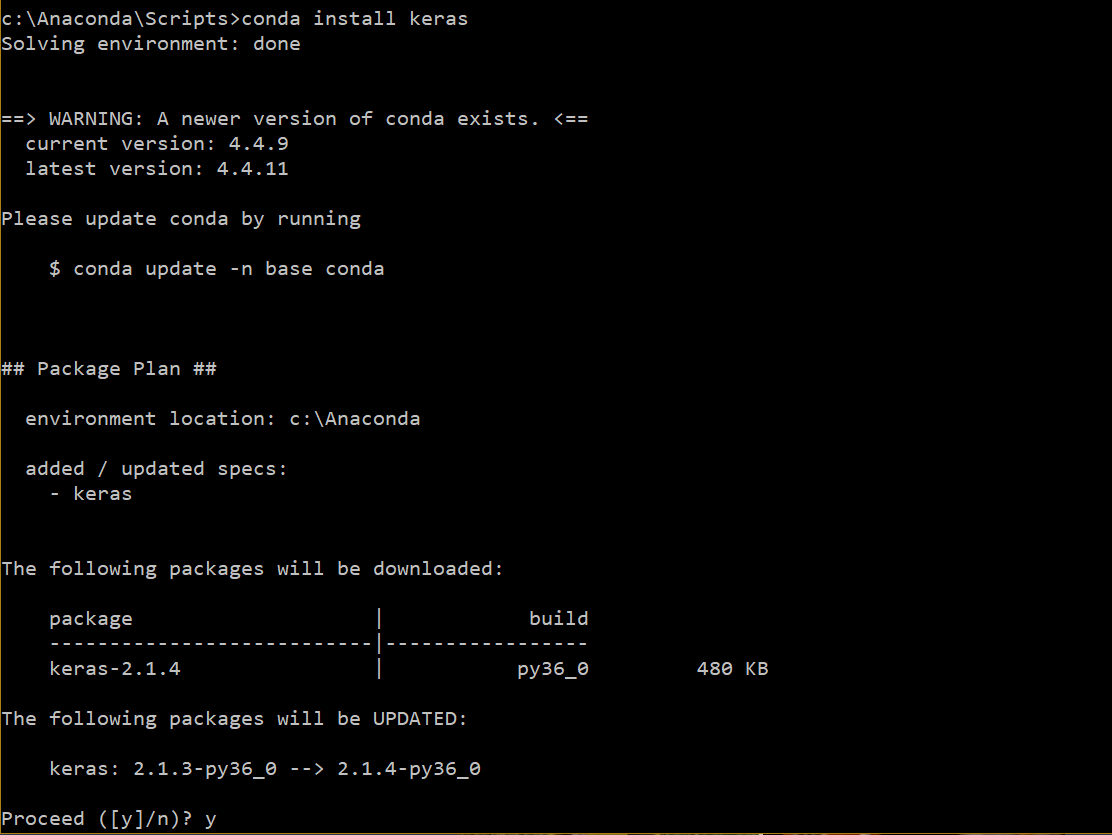




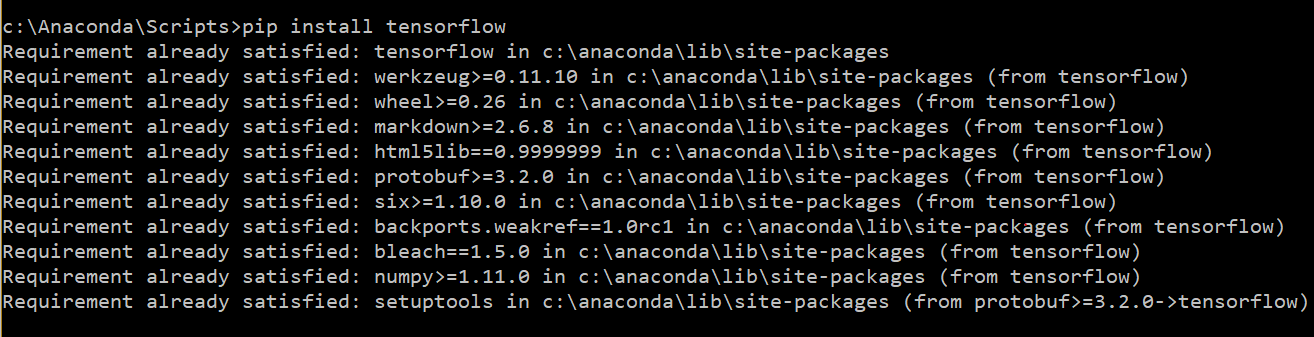
Examples of Python modules include: Keras, Tensorflow, shutil, os, scikit-learn, pandas, numpy etc.

Python modules must be installed via pip or conda if using Anaconda

conda install keras



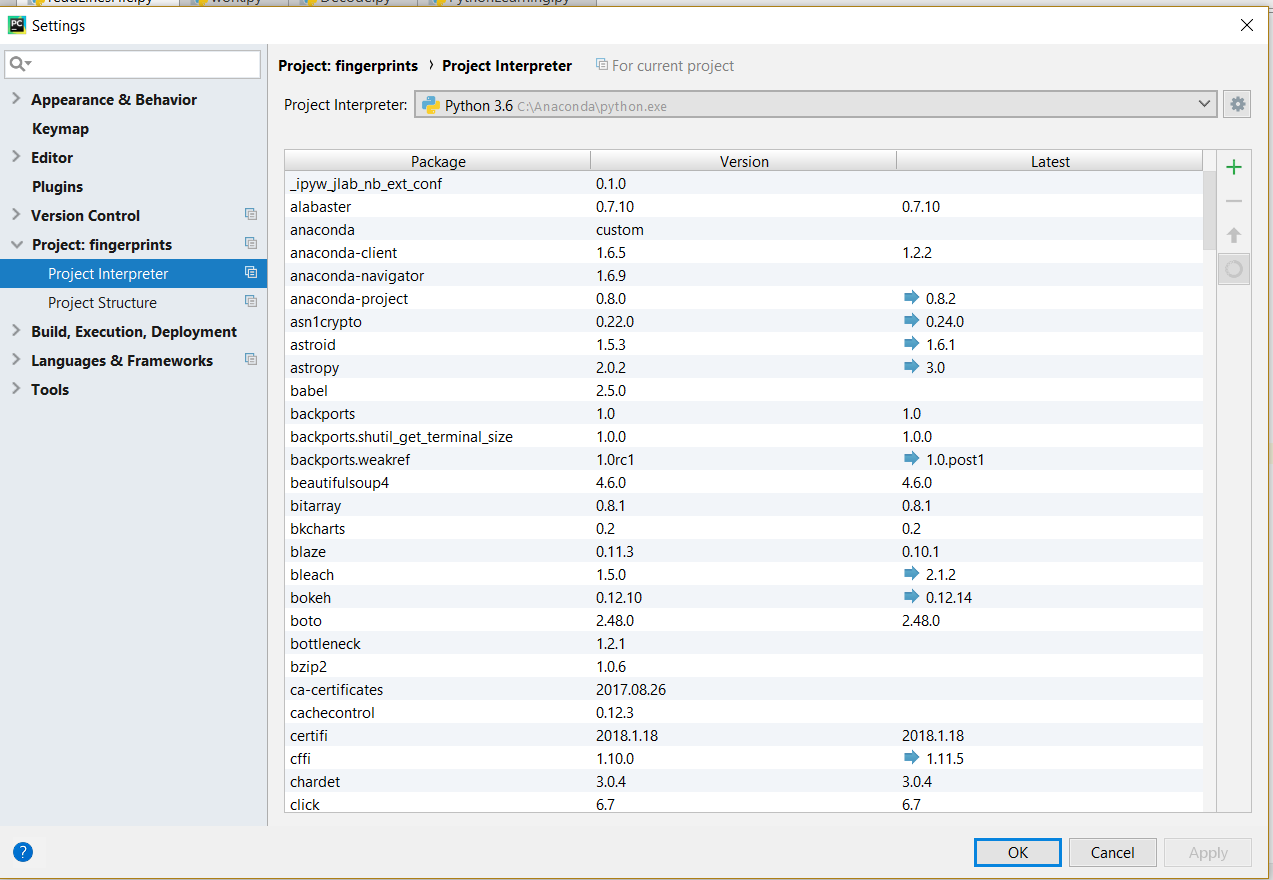
Pip install tensorflow



Or via Pycharm IDE

File > Settings > Project Interpreter

Click on plus to add a module



Learning Python Basics#

**Importing modules**

**import** time  
**import** calendar  
**Printing to console**print(**"Hello, Python"**)  
  
**Concatenation**item1 = **"Bread"**item2 = **"milk"**item3 = **"eggs"**total = item1 + \  
 item2 + \  
 item3

**Lists**  
days = [**"Monday"**, **"Tuesday"**, **"Wednesday"**,  
 **"Thursday"**, **"Friday"**]  
  
**String literals**word = **'word'**sentence = **"""This is a sentence made up \  
 of multiple lines and sentences"""  
  
User input**raw\_input**("\n\nPress the key to exit\n")**print(**"Hello my name is "** + firstName + **" "** + surname)  
print(**"\nI drove "** + str(miles) + **" miles today"**)  
*#can't combine float and strings*age = input(**"\nWhat is your age?\n"**)  
print(**"You said you were "** + str(age) + **" years old!"**)

**Variables** *#Standard data types: Number, String, List, Tuple, Dictionary*counter = 100  
miles = 1000.50  
firstName = **"Aoife"**surname = **'Sayers'**  
  
**List**list = [ **'abcd'**, 786 , 2.23, **'john'**, 70.2 ]  
tinylist = [123, **'john'**]  
print(list) *# Prints complete list*print(list[0]) *# Prints first element of the list*print(list[1:3]) *# Prints elements starting from 2nd till 3rd*print(list[2:]) *# Prints elements starting from 3rd element*print(tinylist \* 2) *# Prints list two times*print(list + tinylist) *# Prints concatenated lists***Tuple**print(**"Printing Tuples"**)  
tuple = (**"abcd"**, 793, 23.3, **"John"**, 30.3)  
tinyTuple = (123, **"john"**)  
print(tuple) *#complete list*print(tuple[0]) *#prints 1st index*print(tuple[1:3]) *# Prints elements starting from 2nd till 3rd*print(tuple[2:]) *# Prints elements starting from 3rd element*print(tinyTuple \* 2) *# Prints list two times*print(tuple + tinyTuple) *# Prints concatenated lists***Dictionary**dict = {}  
dict[**'one'**] = **"This is one"**dict[2] = **"This is two"**tinydict = {**'name'**: **'john'**,**'code'**:6734, **'dept'**: **'sales'**}  
print(dict[**'one'**]) *# Prints value for 'one' key*print(dict[2]) *# Prints value for 2 key*print(tinydict) *# Prints complete dictionary*print(tinydict.keys()) *# Prints all the keys*print(tinydict.values()) *# Prints all the values***Arithmetic operators**a = 10  
b = 20  
c = 2  
print(str(a) + **" + "** + str(b) + **" = "** + str((a+b)))  
print(str(a) + **" - "** + str(b) + **" = "** + str((a-b)))  
print(str(a) + **" \* "** + str(b) + **" = "** + str((a\*b)))  
print(str(b) + **" / "** + str(a) + **" = "** + str((b/a)))  
print(str(c) + **" to the power of "** + str(a) + **" = "** + str((2\*\*10)))  
  
**If Elif Else**var = 120  
  
**if** var <=100 :  
 print(**"The value is 100 or less"**)  
**elif** var >= 101 **and** var <= 110:  
 print(**"The value is greater than 100 & less than 110"**)  
**elif** var >= 111 **and** var <= 120:  
 print(**"The value is greater than 110 & less than 120"**)  
**else**:  
 print(**"The value is "** + str(var) + **" not 100"**)  
  
**While loop**count = 0;  
  
**while**(count <= 10):  
 print(**"Number: "** + str(count))  
 count = count + 1  
print(**"\n\t\tExited while loop"**)  
  
**For loop**tables = 12  
j = 0  
**for** j **in** range(0,13):  
 print(str(j) + **" x "** + str(12) + **" = "** + str((j\*12)))  
  
print(**"Number of ticks since 12 am: "** + str(time.time()))  
print(calendar.month(2017,3))  
num = 0

**Functions****def** makeTables(timesTables, HighestNum):  
  
 **for** timesTables **in** range(0, HighestNum+1):  
 print(str(num) + **" x "** + str(timesTables) + **" = "** + str(num\*timesTables))  
 **return**makeTables(12,12)

# Sprint 3

Integrate biometric fingerprint dataset

Clean & Preprocess data

Data Feature Extraction

Noise removal

# References

360Biometrics.com, n.d. *Hand Geometry Biometrics.* [Online]   
Available at: http://www.360biometrics.com/faq/Hand-Geometry-Biometrics.php  
[Accessed 15 Novmeber 2017].

Alyuda.com, 2015. *Neural Network Applications.* [Online]   
Available at: http://www.alyuda.com/products/forecaster/neural-network-applications.htm  
[Accessed 12 September 2017].

AppliedGo, 2016. *Perceptrons - the most basic form of a neural network.* [Online]   
Available at: https://appliedgo.net/perceptron/  
[Accessed 12 10 2017].

biometricsolution.com, 2018. *Fingerprint Recognition.* [Online]   
Available at: http://www.biometricsolution.com/  
[Accessed 24 January 2018].

BrainTree & TechCrunch, 2016. *Forget passwords how biometrics are transforming the security of mobile payments.* [Online]   
Available at: https://techcrunch.com/sponsored/forget-passwords-how-biometrics-are-transforming-the-security-of-mobile-payments/  
[Accessed 15 Novembber 2017].

cs231n, 2017. *Convolutional Networks.* [Online]   
Available at: http://cs231n.github.io/convolutional-networks/  
[Accessed 18 October 2017].

cs231n, 2017. *convolutional-networks.* [Online]   
Available at: http://cs231n.github.io/convolutional-networks/  
[Accessed 30 September 2017].

Fausett, L., 1993. *Fundamentals of Neural Networks.* 2nd ed. New Jersey: Pretence Hall.

Fausset, L., 1994. *Fundamentals of Neural Networks.* 1st ed. New Jersey: Pretence Hall.

Filippo Amato, A. L. E. M. P.-M. P. V. A. H. J. H., 2013. Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine,* Volume 11.

forensicsciencesimplified.org, 2017. *Principles of Fingerprint Analysis.* [Online]   
Available at: http://www.forensicsciencesimplified.org/prints/principles.html  
[Accessed 15 November 2017].

Fun, M. I., 2014. *Vectors Dot Product.* [Online]   
Available at: https://www.mathsisfun.com/algebra/vectors-dot-product.html  
[Accessed 18 October 2017].

Hackernoon, 2017. *Overview of artificial neural networks and its applications.* [Online]   
Available at: https://hackernoon.com/overview-of-artificial-neural-networks-and-its-applications-2525c1addff7  
[Accessed 18 October 2017].

handlines.ie, 2005. *Do you have unusual fingerprints.* [Online]   
Available at: http://handlines.blogspot.ie/2005/09/do-you-have-unusual-fingerprints.html  
[Accessed 15 November 2017].

Humphrys, D. M., n.d. *Single-layer Neural Networks (Perceptrons).* [Online]   
Available at: http://computing.dcu.ie/~humphrys/Notes/Neural/single.neural.html  
[Accessed 18 October 2017].

K.Vijayarekha, D., 2013. Activation Functions. *NPTEL – Electronics & Communication Engineering – Pattern Recognition,* Volume I, p. 3.

Keras, 2017. *Keras - Github.* [Online]   
Available at: https://github.com/fchollet/keras  
[Accessed 18 October 2017].

King, R., 2013. *Retinal Scan Technology.* [Online]   
Available at: http://www.biometricupdate.com/201307/explainer-retinal-scan-technology  
[Accessed 11 November 2017].

M.Domnanovich, L. R. P. D. P. S., 2004. Prediction of trace compounds in biogas from anaerobic digestion using the MATLAB Neural Network Toolbox. *Environmental Modelling & Software,* pp. 803-810.

OWASP, 2017. *Authentication Cheet Sheet.* [Online]   
Available at: https://www.owasp.org/index.php/Authentication\_Cheat\_Sheet  
[Accessed 22 November 2017].

Pankaj, 2017. *Session Management in Java – HttpServlet, Cookies, URL Rewriting.* [Online]   
Available at: https://www.journaldev.com/1907/java-session-management-servlet-httpsession-url-rewriting  
[Accessed 22 November 2017].

Point, T., 2017. *Artificial Intelligence - Neural Networks.* [Online]   
Available at: https://www.tutorialspoint.com/artificial\_intelligence/artificial\_intelligence\_neural\_networks.htm  
[Accessed 11 September 2017].

S. Agatonovic-Kustrin, R. B., 1999. Basic concepts of artificial neural network (ANN) modeling. *Journal of Pharmaceutical and Biomedical Analysis,* Volume II, p. 11.

Stack, J. X., 2017. *Log analytics with deep learning and machine learning.* [Online]   
Available at: https://www.xenonstack.com/blog/log-analytics-with-deep-learning-and-machine-learning  
[Accessed 18 October 2017].

Target, T., n.d. *Biometrics.* [Online]   
Available at: http://searchsecurity.techtarget.com/definition/biometrics  
[Accessed 15 November 2017].

Team, D., 2017. *Learning Rules In Neural Network.* [Online]   
Available at: https://data-flair.training/blogs/learning-rules-in-neural-network/  
[Accessed 11 December 2017].

technovelgy.com, n.d. *Biometric Security.* [Online]   
Available at: http://www.technovelgy.com/ct/Technology-Article.asp?ArtNum=11  
[Accessed 15 November 2017].

TensorFlow, 2017. *About Tensorflow.* [Online]   
Available at: https://www.tensorflow.org/  
[Accessed 18 October 2017].

Thakkar, D., 2015. Minutiae Based Extraction in Fingerprint Recognition. *Bayometric*, 27 May, p. 1.

Thepade, S. & Manade, M., 2017. Multifinger biometric identification with score level fusion using Thepade's sorted n-ary block truncation coding. *Electronics & Telecommunication Engineering (ICWET 2016), International Conference & Workshop on,* I(1), p. 6.

Trader, J., 2012. *Iris Recognition vs Retina Scanning What are the differences?.* [Online]   
Available at: http://www.m2sys.com/blog/biometric-hardware/iris-recognition-vs-retina-scanning-what-are-the-differences/  
[Accessed 15 November 2017].

Turgeman, A., 2017. Behavioral Biometrics Are Not New, So Why Are They So Hot Right Now?. *Forbes Tech*, 20 June, p. 2.

V.Chande, D. S., 2012. *A Survey on Generation of Test Cases and Test Data,* s.l.: Research Gate.

VV, P., 2016. *Mathematical Foundation for Activation Functions in Artificial Neural Networks.* [Online]   
Available at: https://medium.com/autonomous-agents/mathematical-foundation-for-activation-functions-in-artificial-neural-networks-a51c9dd7c089  
[Accessed 19 October 2017].

Walia, A. S., 2017. *Activation functions and it’s types-Which is better?.* [Online]   
Available at: https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f  
[Accessed 25 November 2017].

Yangqing Jia∗, E. S. ,. J. D. S. K. J. L. R. G. S. G. T. D., 2014. Caffe: Convolutional Architecture. *Caffe: Convolutional Architecture,* p. 4.