**PROJECT REPORT**

**Artificial Neural Network Based Optical Character Recognition**

**Subject:** ADEO1 -Singal & Information - Application

**Professor:** Guy Almouzni

**Group member:**

Fangnuo cai

Pham Quoc Viet

Yenchu chen

**Table of Contents**

[***1***](#_30j0zll) ***Introduction 4***

[***2***](#_1fob9te) ***Image Datasets 5***

[***3***](#_tyjcwt) ***Image Pre-processing 6***

[**3.1**](#_3dy6vkm) **Image Acquisition and Binarization 6**

[**3.2**](#_2s8eyo1) **Normalization 7**

[**3.3**](#_17dp8vu) **Whitening 8**

[**3.4**](#_lnxbz9) **Remove small objects 9**

[***4***](#_2jxsxqh) ***Feature Extraction 10***

[**4.1**](#_z337ya) **Closed area 10**

[**4.2**](#_4i7ojhp) **Walsh-Hadamard transform 11**

[**4.3**](#_2bn6wsx) **Horizontal Symmetry, Vertical Symmetry 13**

[**4.4**](#_3as4poj) **Sum of pixel at H30, H50, H80, V30, V50, V80 14**

[***5***](#_1pxezwc) ***Artificial Neural Networks 15***

[**5.1**](#_147n2zr) **Artificial neuron 16**

[**5.2**](#_23ckvvd) **Artificial Neural Networks 18**

[***6***](#_32hioqz) ***Convolution Neural Network Applied On OCR 18***

[**6.1**](#_1hmsyys) **Introduction 18**

[**6.2**](#_2grqrue) **Procedure of CNN 19**

[**6.3**](#_28h4qwu) **Model 22**

[**6.4**](#_37m2jsg) **Results 23**

[***7***](#_1mrcu09) ***Appendix 26***

[***8***](#_3l18frh) ***Reference 29***

**List of Figures**

[Figure 2-1:ETL-1 datasets 6](#_3znysh7)

[Figure 2-2: ETL-1 Description 6](#_2et92p0)

[Figure 2-3: Image Dataset of Numbers](about:blank) 6

[Figure 3-1: Image Acquisition and Binarization-Python code 7](#_1t3h5sf)

[Figure 3-2: Output (Matrix) after Acquisition and Binarization 7](#_4d34og8)

[Figure 3-3: Normalization](about:blank) 8

[Figure 3-4: Normalization- Python code](about:blank) 8

[Figure 3-5: Whitening 8](#_3rdcrjn)

[Figure 3-6: Whitening-Python code 9](#_26in1rg)

[Figure 3-7: Remove small objects 9](#_35nkun2)

[Figure 3-8: Output-Remove small objects 10](#_1ksv4uv)

[Figure 3-9: Apply remove small objects - Python code 10](#_44sinio)

[Figure 4-1: Closed area - Python code 10](#_3j2qqm3)

[Figure 4-2: Closed area- Labelled image output 11](#_1y810tw)

[Figure 4-3: Walsh-Hadamard transform- Python code 11](#_2xcytpi)

[Figure 4-4: WHT for comparison characters 12](#_1ci93xb)

[Figure 4-5: WHT for comparison characters 13](#_3whwml4)

[Figure 4-6: Horizontal symmetry, Vertical Symmetry 14](#_qsh70q)

[Figure 5-1: Working principle of an artificial neuron 15](#_49x2ik5)

[Figure 5-2: Example of simple artificial neural network 16](#_2p2csry)

[Figure 5-3: Biological and artificial neuron design 16](#_3o7alnk)

[Figure 5-4: Feed - forward topology of an artificial neural network 18](#_ihv636)

[Figure 6-1: The concept of CNN 19](#_41mghml)

[Figure 6-2: Mathematically explanation 19](#_vx1227)

[Figure 6-3: Graphically explanation 20](#_3fwokq0)

[Figure 6-4: Demo from our dataset with one convolution and two different activation function 20](#_1v1yuxt)

[Figure 6-5: Demo from our data with 2\*2 max pooling function 21](#_4f1mdlm)

[Figure 6-6: Flatten 21](#_2u6wntf)

[Figure 6-7: Dropout 21](#_19c6y18)

[Figure 6-8: Hidden layer to predict output 22](#_3tbugp1)

[Figure 6-9: Model 22](#_nmf14n)

[Figure 7-1: Overfitting](about:blank) 26

[Figure 7-2: Data Processing 28](#_46r0co2)

[Figure 7-3: Model Building 28](#_2lwamvv)

[Figure 7-4: Compile and training 29](#_111kx3o)

# Introduction

OCR stands for Optical Character Recognition, which is a technology to recognise characters include numbers, letters.. etc from images, such as handwriting, scanned documents and photos.

Nowadays, OCR is widely used throughout the entire spectrum of industries, bring a big innovation on the document management process. OCR has enabled scanned documents to become more than just image files, turning into fully editable and searchable documents with text content that is recognized by computers.

Both handwriting and printed characters could be recognized, but the performance is dependent upon the quality of the input documents. The more constrained the input is, the better result will receive of the OCR system be. However, when it comes to totally unconstrained handwriting, OCR are still a long way from recognite as well as humans. However, the technical advances in machine learning are bringing the technology closer to its ideal.

Artificial neural network(ANN) is a machine learning algorithm based on the working of human brain by making the right connections. The human brain consists of millions of neurons. It sends and process signals in the form of electrical and chemical signals. These neurons are connected with a special structure known as synapses. Synapses allow neurons to pass signals.

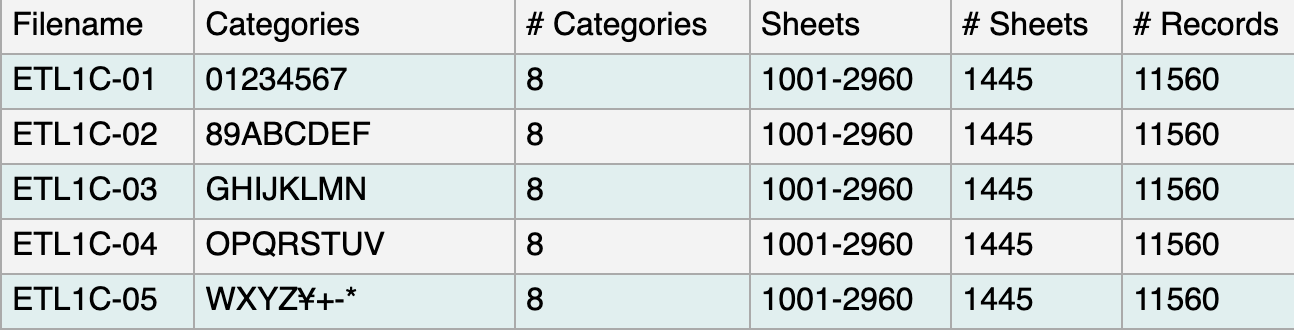
An ANN is an information processing technique. It ‘s working mechanism like the way human brain processes information. ANN includes a large number of connected processing units which work together to process input information and generate output results from it.

Neural networks have great application in data mining used in sectors. For example economics, for pattern recognition, etc. It could be also used for data classification in a large amount of data after careful training.

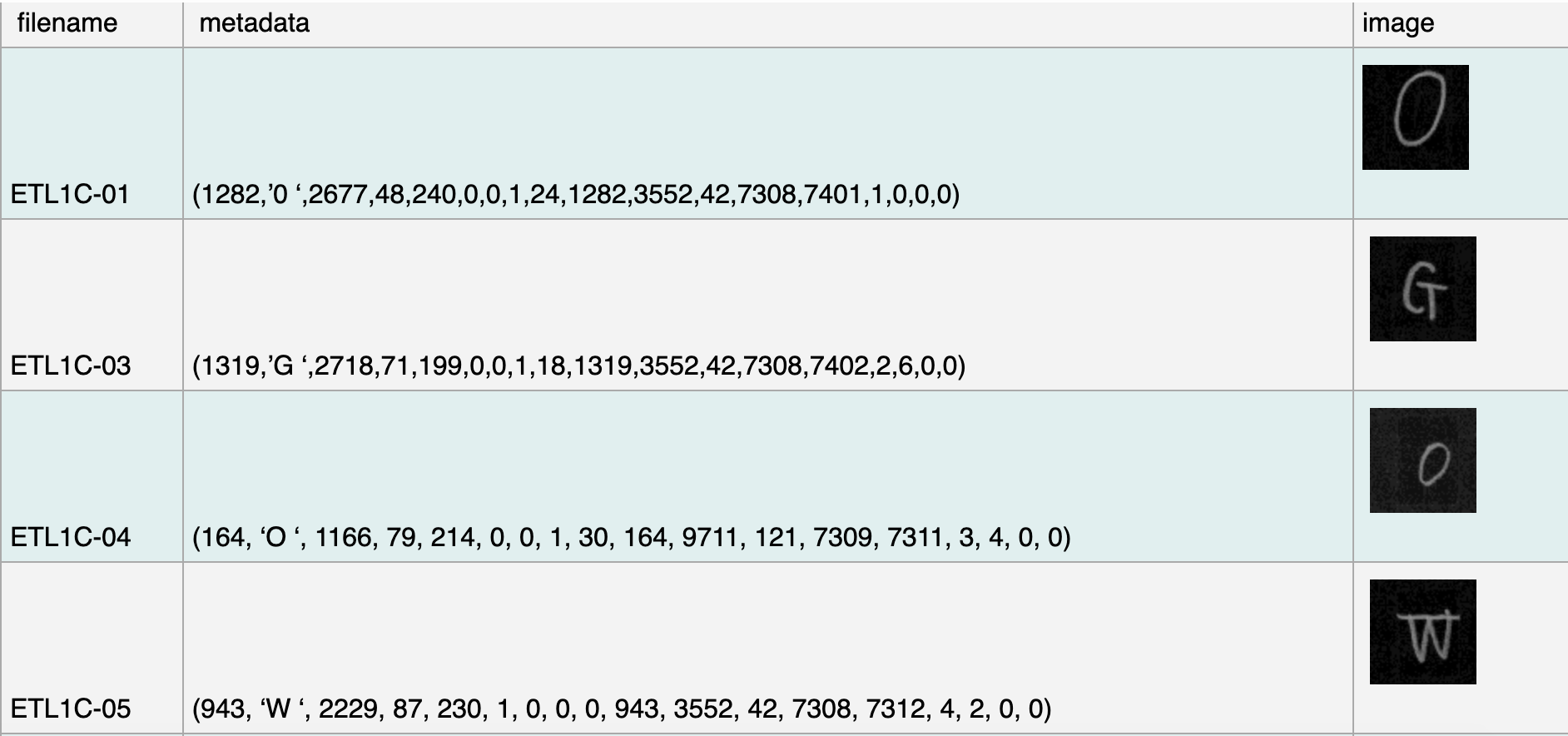
With the help of the combination ANN and OCR, people no longer need to manually retype important documents when entering them into electronic databases. Instead, OCR extracts relevant information and enters it automatically. The result is accurate, the processing time is less.

# Image Datasets

In this project, we use a dataset of images named “ETL Character Database” that is a collection of images of about 1.2 million hand-written and machine-printed numerals, symbols, Latin alphabets and Japanese characters and compiled in 9 datasets (ETL-1 to ETL-9). This database has been collected by Electrotechnical Laboratory (currently reorganized as the National Institute of Advanced Industrial Science and Technology (AIST)) under the cooperation with Japan Electronic Industry Developement Association (currently reorganized as Japan Electronics and Information Technology Industries Association), universities and other research organizations for character recognition researches from 1973 to 1984.

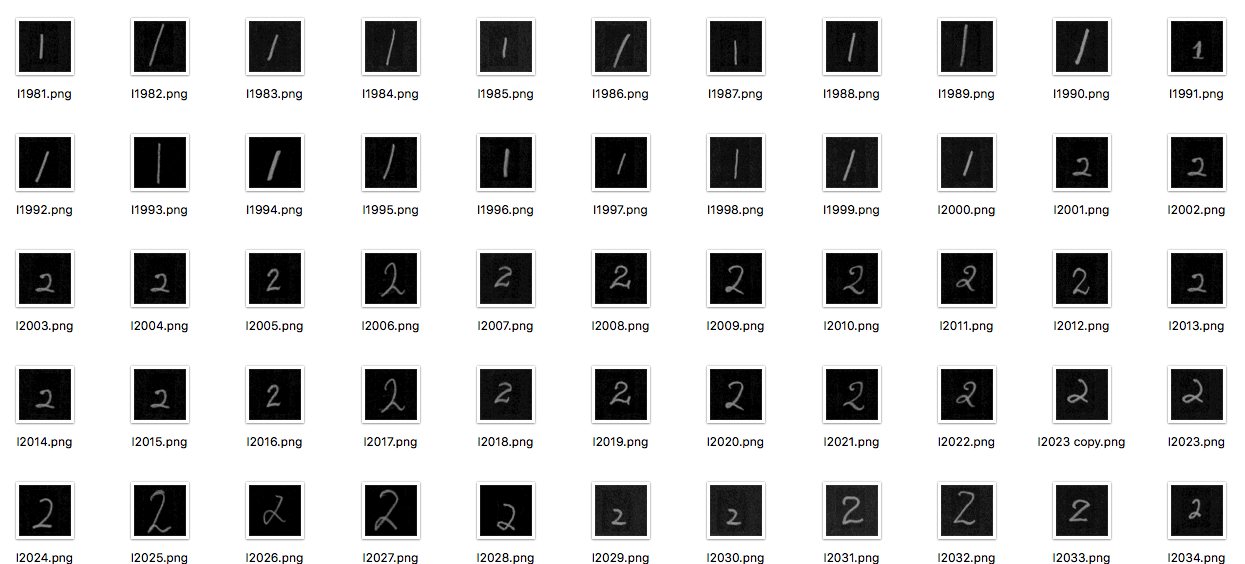


*Figure 2-1:ETL-1 datasets*



*Figure 2-2: ETL-1 Description*

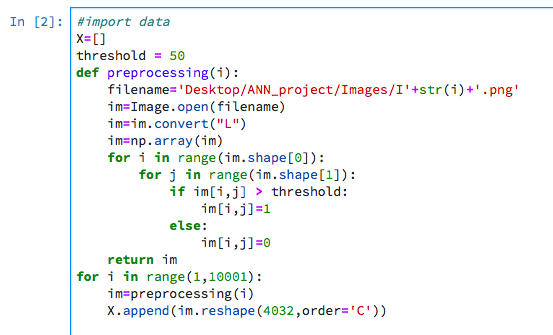
In the scope of the project, we use 1000 images of each number from 0 to 9 which are extracted from ETL1C-01 dataset and rename each image by the format I.png. (For example, the images with names I2001.png to I3000.png will be the number 2) every images of character are in the same size 63\*64 pixels.



# Image Pre-processing

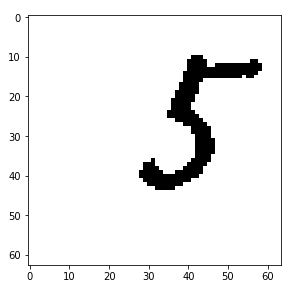
## Image Acquisition and Binarization

Here we use Python PIL library to read every image, transfer every image to matrix and binarize the matrix with the threshold that we chose. If the pixel of a point is bigger than the threshold, we set it to 1, otherwise we set it to 0.



*Figure 3-1: Image Acquisition and Binarization-Python code*

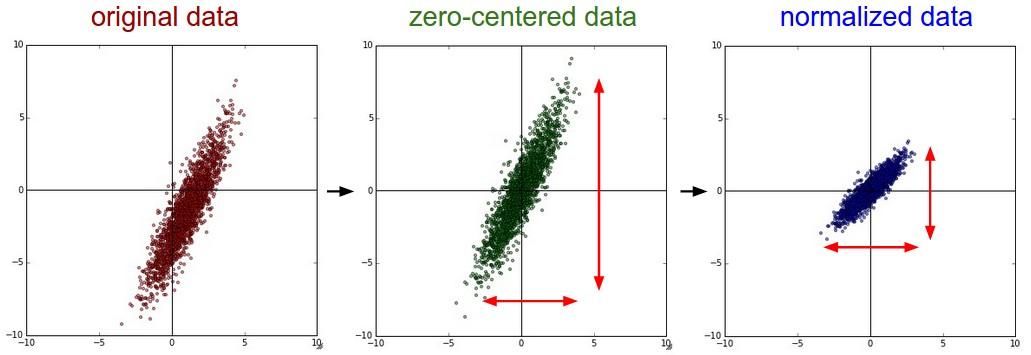
An example of the image(matrix) after Acquisition and Binarization



*Figure 3-2: Output (Matrix) after Acquisition and Binarization*

## Normalization

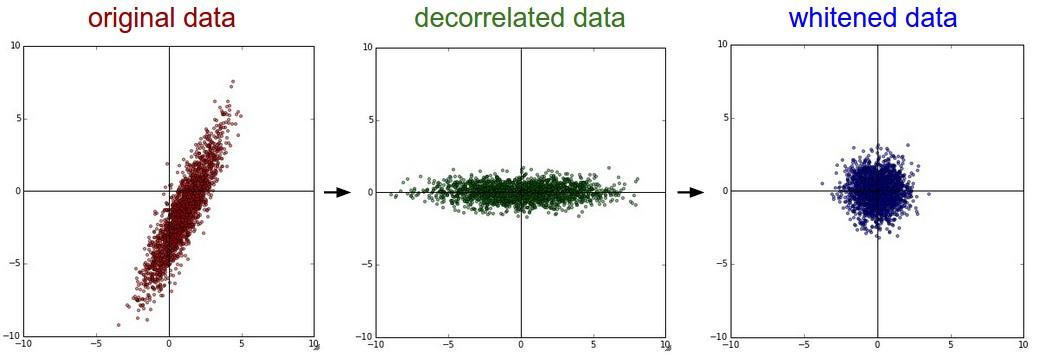
The idea of normalization is to avoid the influence of high frequency noise and very low noise. And on the other hand, it makes image data satisfy normal distribution.



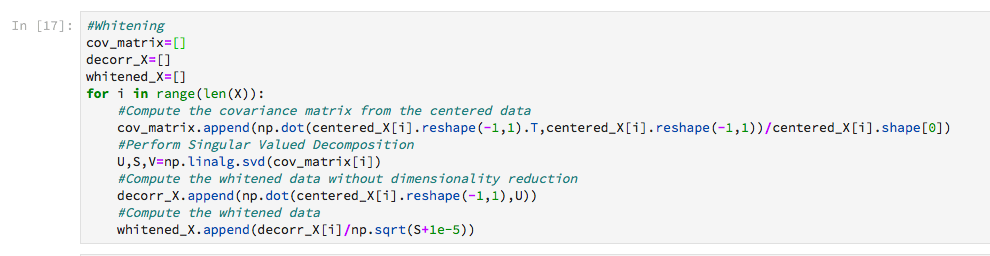


## Whitening

The idea of whitening all image patches is to remove global correlations and to normalize the variance



*Figure 3-5: Whitening*

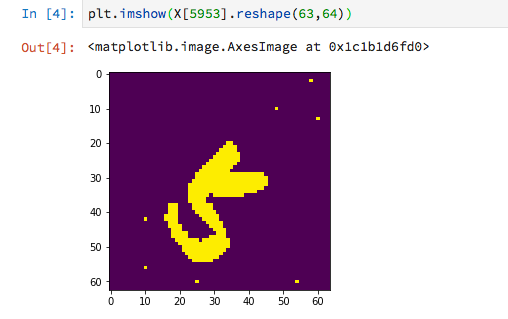


*Figure 3-6: Whitening-Python code*

## Remove small objects

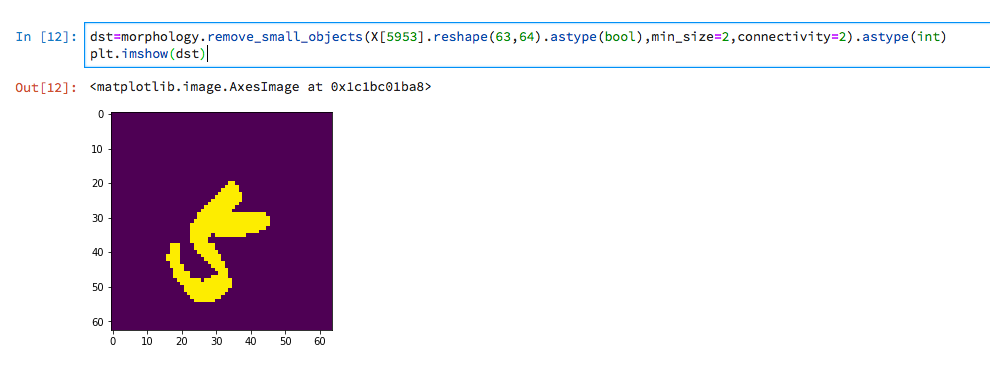
When images are transferred to matrix with a threshold, there is a risk that some small areas are also transferred to 1.

For example, as we can see in this image:



*Figure 3-7: Remove small objects*

We use the remove\_small\_objects function in the library skimage to remove these small objects. We only need to set the area of objects that we need to remove and the type of connectivity.



*Figure 3-8: Output-Remove small objects*

We apply this function to every image



*Figure 3-9: Apply remove small objects - Python code*

# Feature Extraction

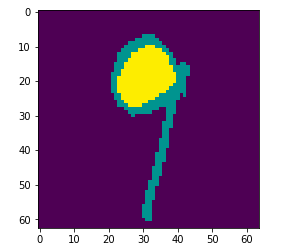
## Closed area

We use the measure.label function in the skimage library to calculate the closed area. This function will give a label to each area. The label.max() function will give the maximum lable in the image and thus label.max()-2 will give the exact number of closed area.



*Figure 4-1: Closed area - Python code*

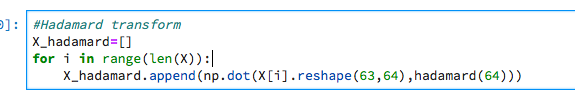
An example of the labelled image



*Figure 4-2: Closed area- Labelled image output*

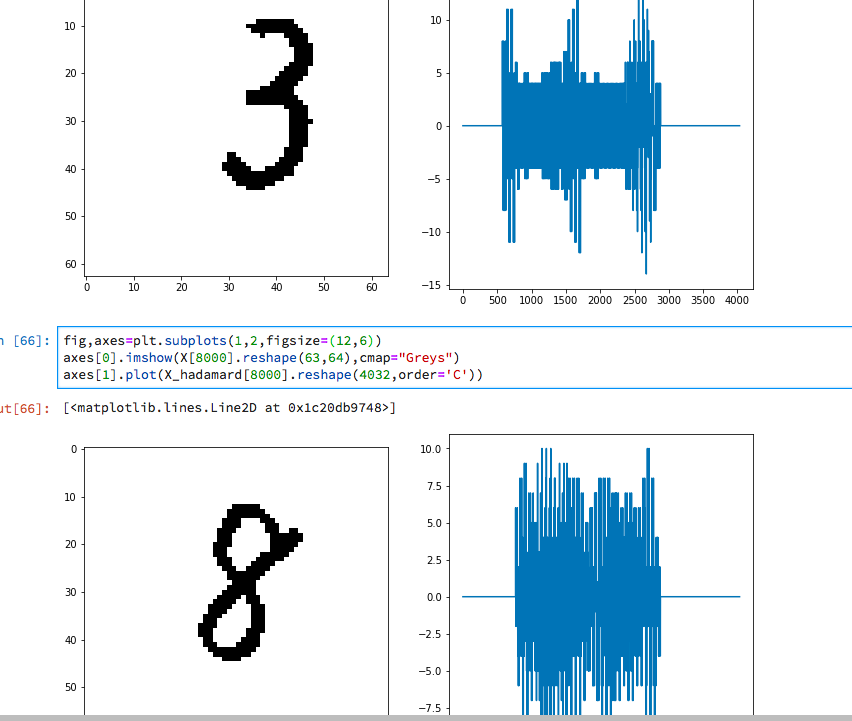
## Walsh-Hadamard transform

The Walsh-Hadamard transform of a character shows the variation in the magnitude of the WHT coefficients. It can be used to find similarity of different inputs

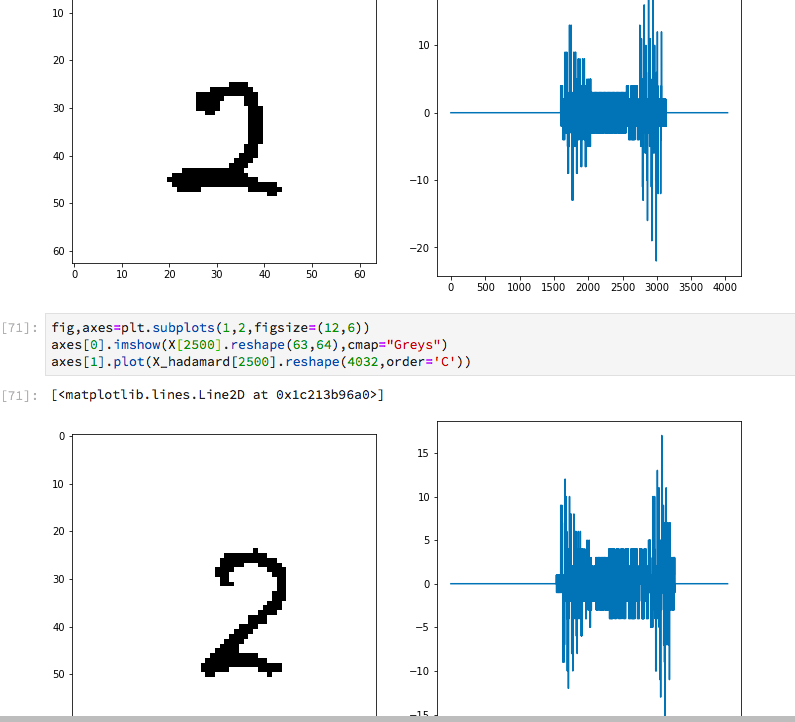


*Figure 4-3: Walsh-Hadamard transform- Python code*

Comparisons:



*Figure 4-4: WHT for comparison characters*



*Figure 4-5: WHT for comparison characters*

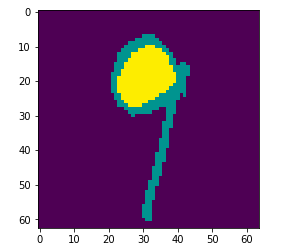
As we can see from the above comparisons, the two different images of character 2 shows a high similarity in the WHT transform, which is different from the WHT transform of character 3 or 8.

## Horizontal Symmetry, Vertical Symmetry

To detect the symmetry of an image, we use a simple idea of the centroid.From the labelled image below, we can use the function measure.regionprops in the skimage library to calculate the properties of each area. So, in order to find the horizontal symmetry of our image, we just need to find the centroid of the area labelled 1 (the green area in the picture), and calculate the distance between this centroid to the minimum column of this area and the distance between this centroid to the maximum column of this area.

If the two distances are close to each other, then the character is horizontal symmetry.

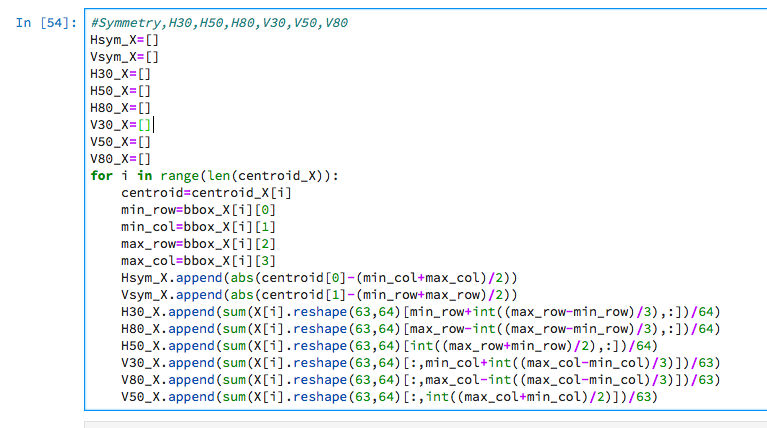
(We apply the same concept for vertical symmetry)



*Figure 4-6: Horizontal symmetry, Vertical Symmetry*

## Sum of pixel at H30, H50, H80, V30, V50, V80

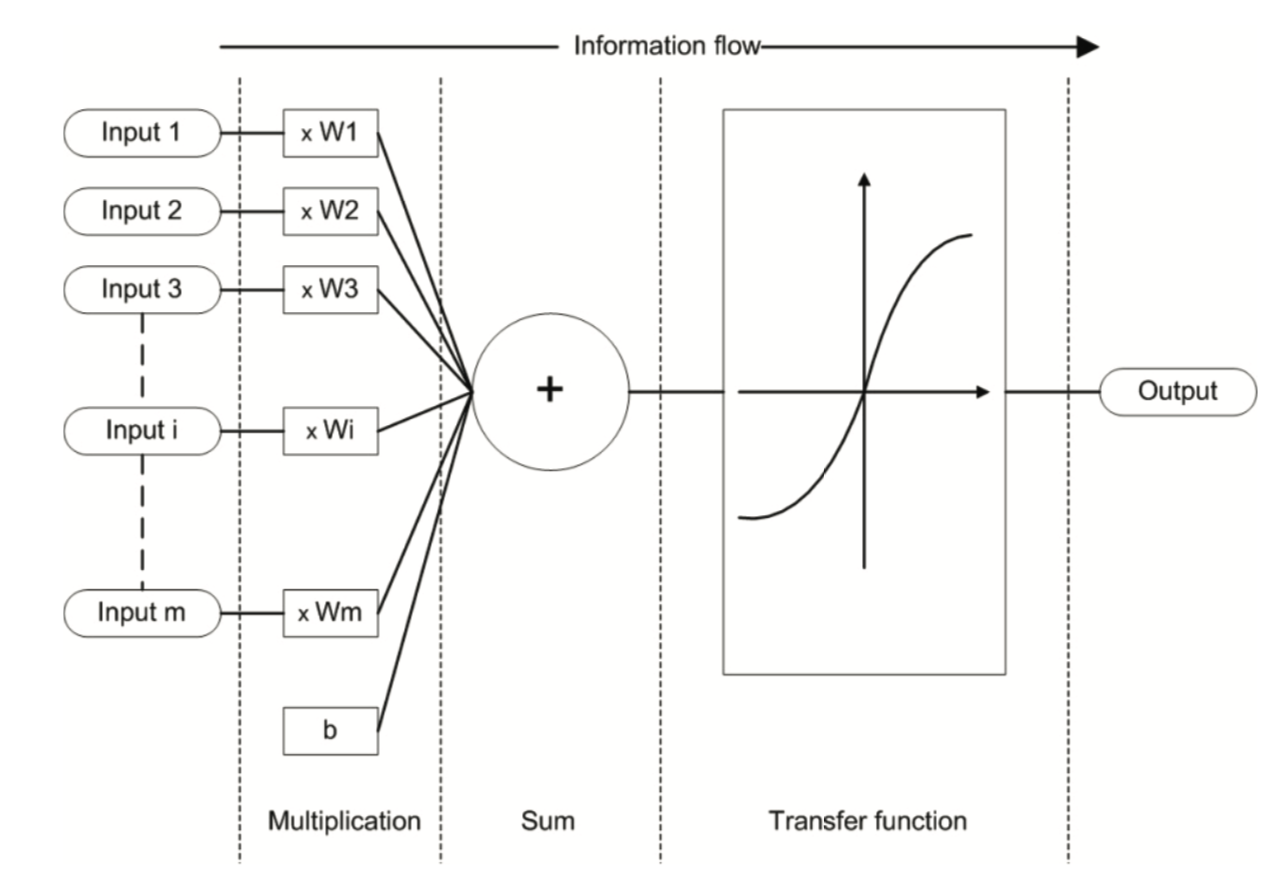
Here we just need to calculate the sum of pixel with values 1 at every line



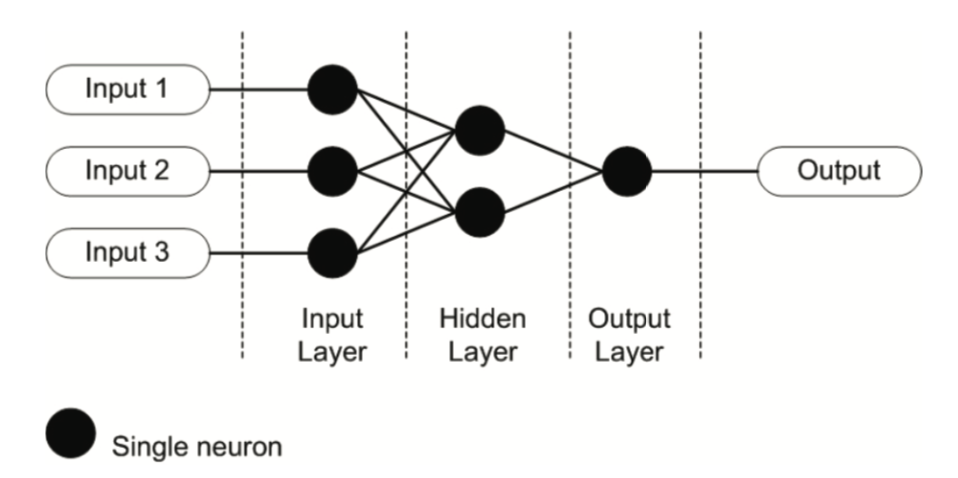
# Artificial Neural Networks

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function).

Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias. At the exit of artificial neuron the sum of previously weighted inputs and bias is passing trough activation function that is also called transfer function



*Figure 5-1: Working principle of an artificial neuron*

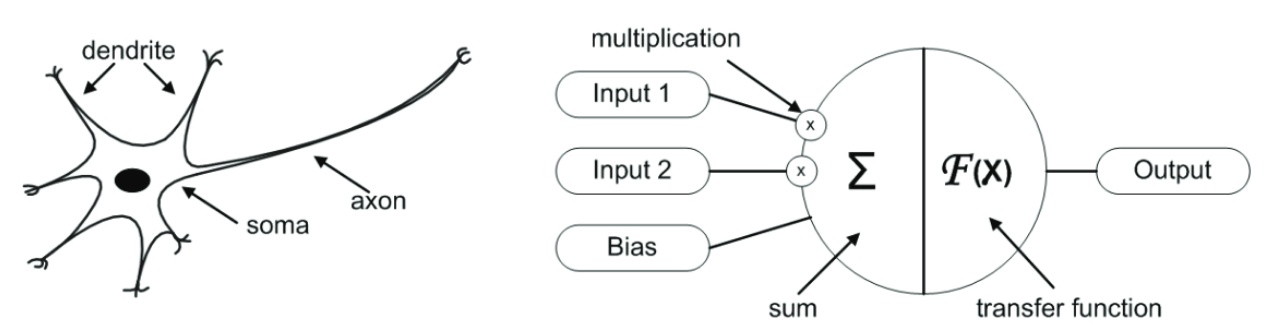


*Figure 5-2: Example of simple artificial neural network*

## Artificial neuron

Artificial neuron is a basic building block of every artificial neural network. Its design and functionalities are derived from observation of a biological neuron that is basic building block of biological neural networks (systems) which includes the brain, spinal cord and peripheral ganglia.

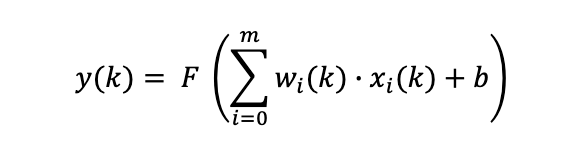
Similarities in design and functionalities can be seen in figure where the left side of a figure represents a biological neuron with its soma, dendrites and axon and where the right side of a figure represents an artificial neuron with its inputs, weights, transfer function, bias and outputs.

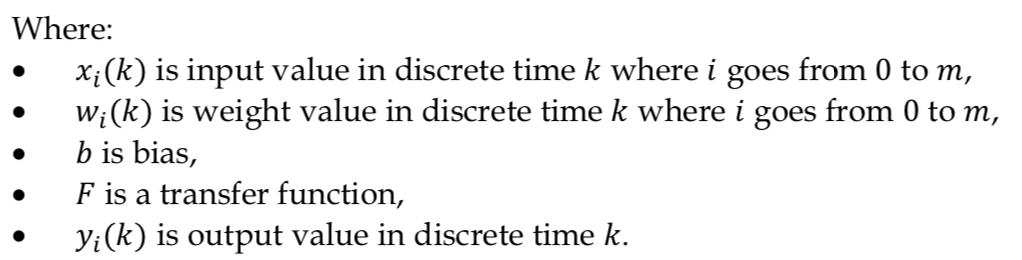


*Figure 5-3: Biological and artificial neuron design*

In biological neuron information comes into the neuron via dendrite, soma processes the information and passes it on via axon.

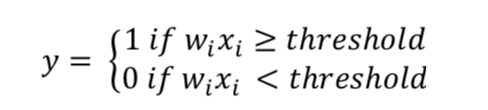
With artificial neuron, the information comes into the body of an artificial neuron via inputs that are weighted (each input can be individually multiplied with a weight). The body of an artificial neuron then sums the weighted inputs, bias and “processes” the sum with a transfer function. At the end an artificial neuron passes the processed information via output(s). Benefit of artificial neuron model simplicity can be seen in its mathematical description below:





Transfer function defines the properties of artificial neuron and can be any mathematical function. In most cases we choose it from the following set of functions: Step function, Linear function and Non-linear (Sigmoid) function.

Step function is binary function that has only two possible output values (e.g. zero and one). That means if input value meets specific threshold the output value results in one value and if specific threshold is not meet that results in different output value.



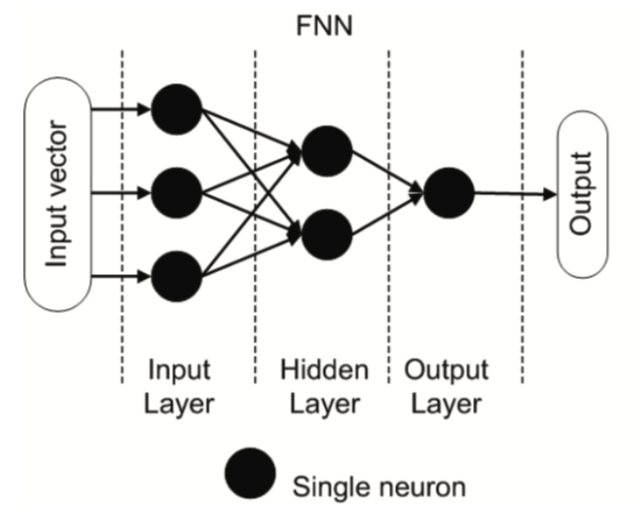
When this type of transfer function is used in artificial neuron we call this artificial neuron perceptron. Perceptron is used for solving classification problems and as such it can be most commonly found in the last layer of artificial neural networks.

In case of linear transfer function artificial neuron is doing simple linear transformation over the sum of weighted inputs and bias. Such an artificial neuron is in contrast to perceptron most commonly used in the input layer of artificial neural networks.

When we use non-linear function the sigmoid function is the most commonly used. Sigmoid function has easily calculated derivate, which can be important when calculating weight updates in the artificial neural network.

## Artificial Neural Networks

When combining two or more artificial neurons we are getting an artificial neural network. The way that individual artificial neurons are interconnected is called topology, architecture or graph of an artificial neural network. we group individual neurons in layers: input, hidden and output layer.



*Figure 5-4: Feed - forward topology of an artificial neural network*

# Convolution Neural Network Applied On OCR

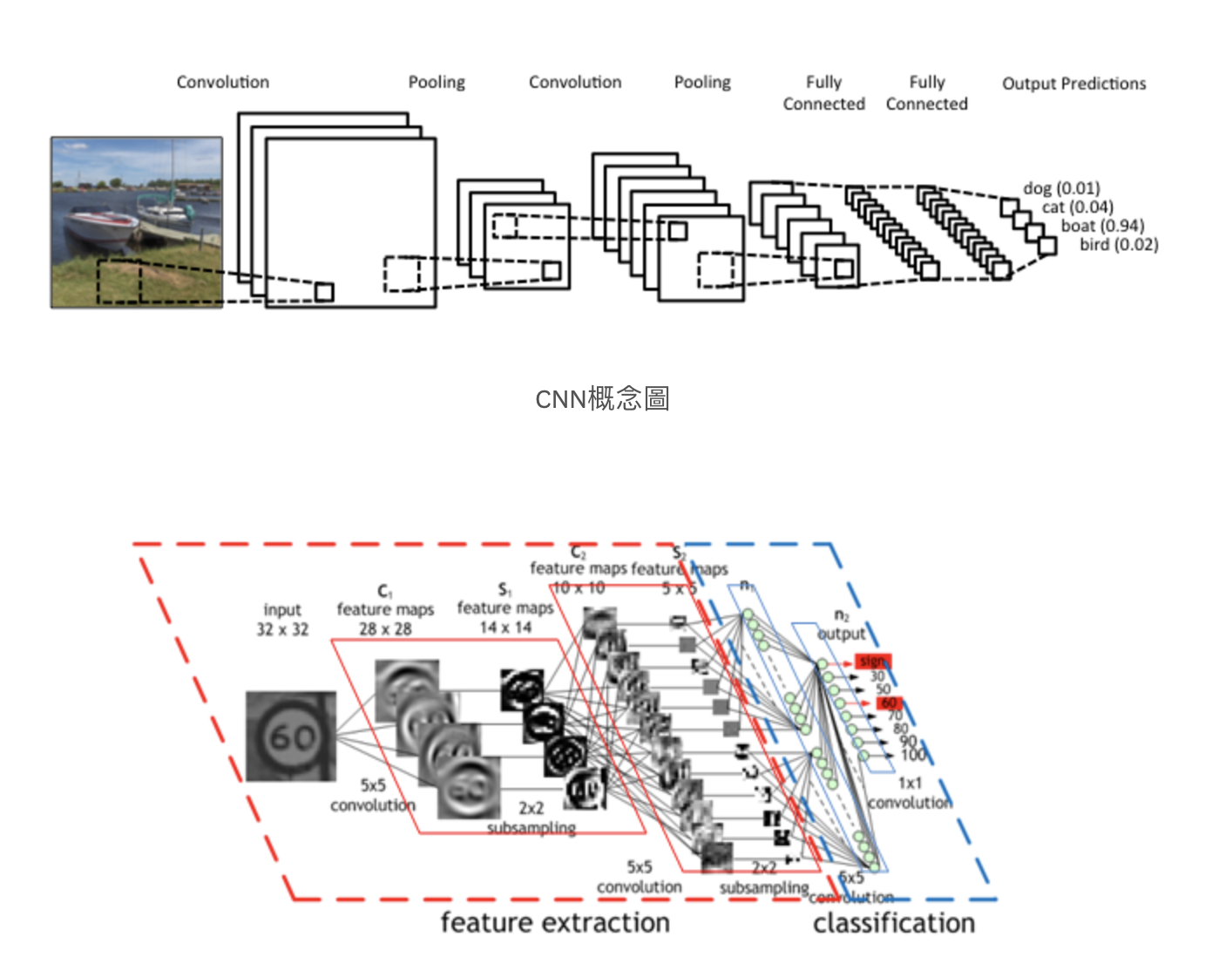
## Introduction

CNN is a supervised learning and a subclass if ANN1, the different between convolution network is that it uses at least one layer one convolution method to process data.

A convolution unit receives its input from multiple units from the previous layer which together create a proximity. Therefore, the input units share their weights, in another way is all the points in one feature map shares the same filter.

And by choosing the CNN instead of ANN, we can get the following benefits:

1. Many-to-one mappings(pooling) : In this way we can use fewer parameter, and it will reduce the complexity(points) and also the chance of overfitting2.
2. Locally connected networks: Convolution has the property to combine neighbor points, therefore, when applied to pictures it shows the essence of the graph more precisely.
3. Filter: Diminishing the noise and extract the character of picture.



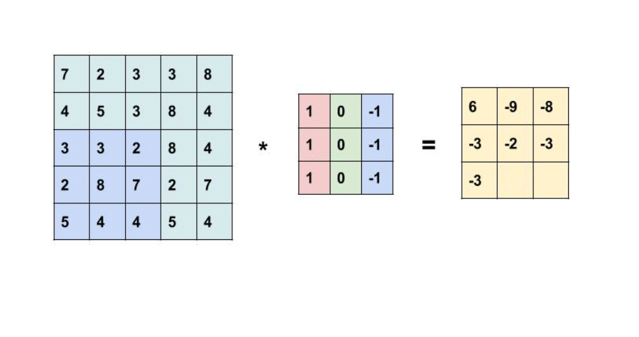
*Figure 6-1: The concept of CNN*

## Procedure of CNN

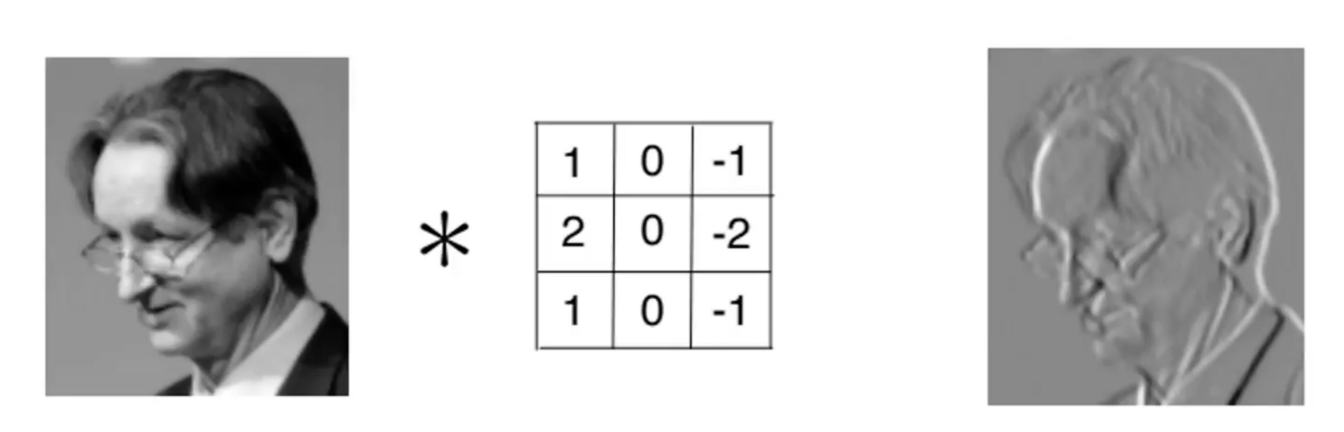
The essential layers in a convolution network has convolution layer, pooling layer, flatten layer and hidden layer.

**Convolution layer:**

The concept of this is to put the original matrix through a filter matrix, it can extract the feature of the original picture. The purpose of this procedure is to make a feature map that can shows the different feature of each graph.



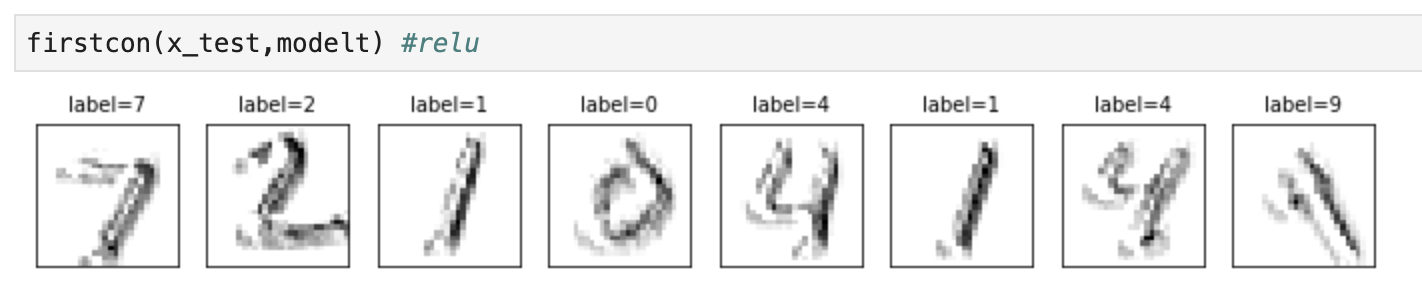
*Figure 6-2: Mathematically explanation*



*Figure 6-3: Graphically explanation*

From the left side of the matrix, multiply the blue part of first matrix with the colorful filter, and put it in the relative position in the feature map. The left side of pictures visualize the procedure of the convolution.

And to the notice, since the graph we show is the original graph with the initial random filter matrix, the feature map on the top still can be recognize, but with more and more numbers and interactions of optimization for the filter, we will not be able to recognized the feature map at the end.





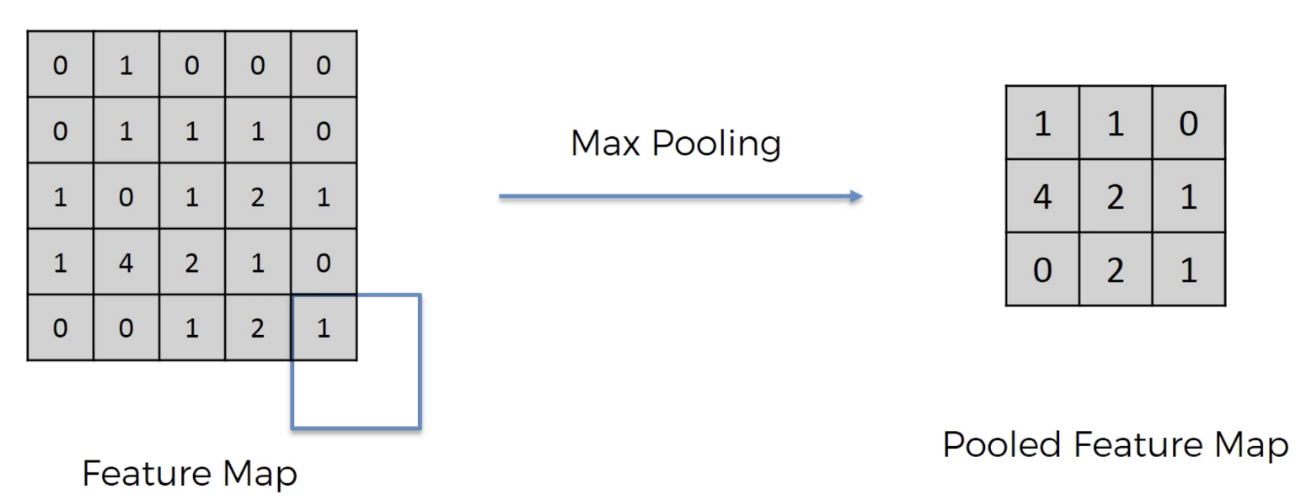
*Figure 6-4: Demo from our dataset with one convolution and two different activation function*

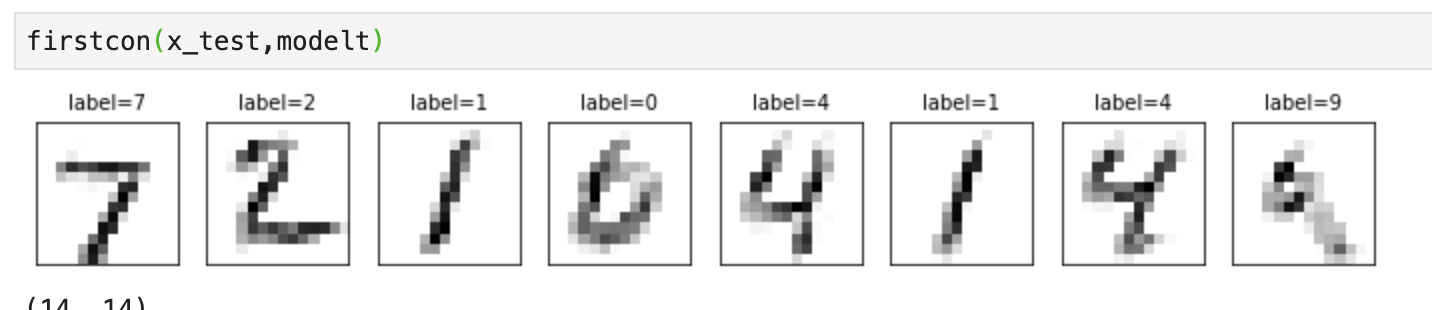
**Pooling Layer:**

In general, CNN use the max pooling function to extract the max number in the range, and in our project, there is no exception.

The propose of the pooling is to downsize the points to lower the memory used and the complexity and it is emphasized on whether the graph has the feature than where is the feature.

We use 2\*2 range pooling to downsize the points in the graph



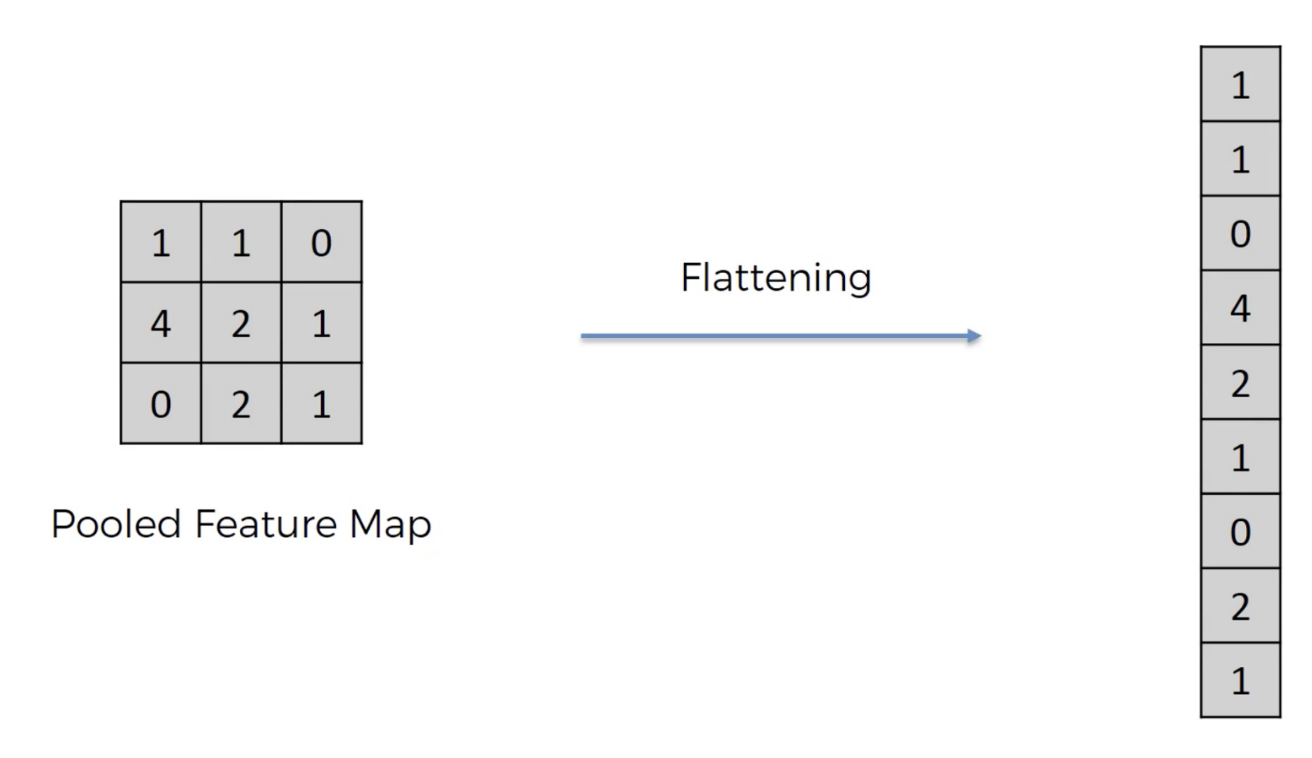




*Figure 6-5: Demo from our data with 2\*2 max pooling function*

**Flatten layer**

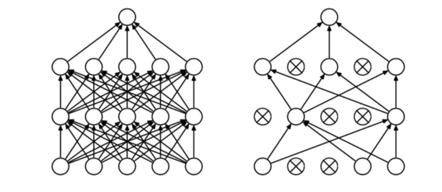
It is a preparation for the next level of analysis, we flatten the 2D table into 1D column, so that we can start to do the ANN(here we call it hidden layer analysis).



*Figure 6-6: Flatten*

**Dropout layer**

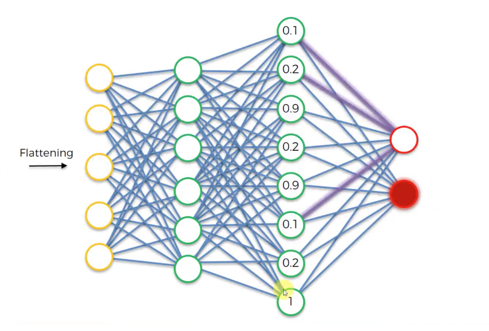
The purpose of the dropout is to reduce the probability of overfitting during the training session. By setting the dropout percentage (0-1) the optimization process will randomly neglect the percentage of the weight. Rather delete it, the weight still would be saved for the prediction.



*Figure 6-7: Dropout*

**Hidden layer**

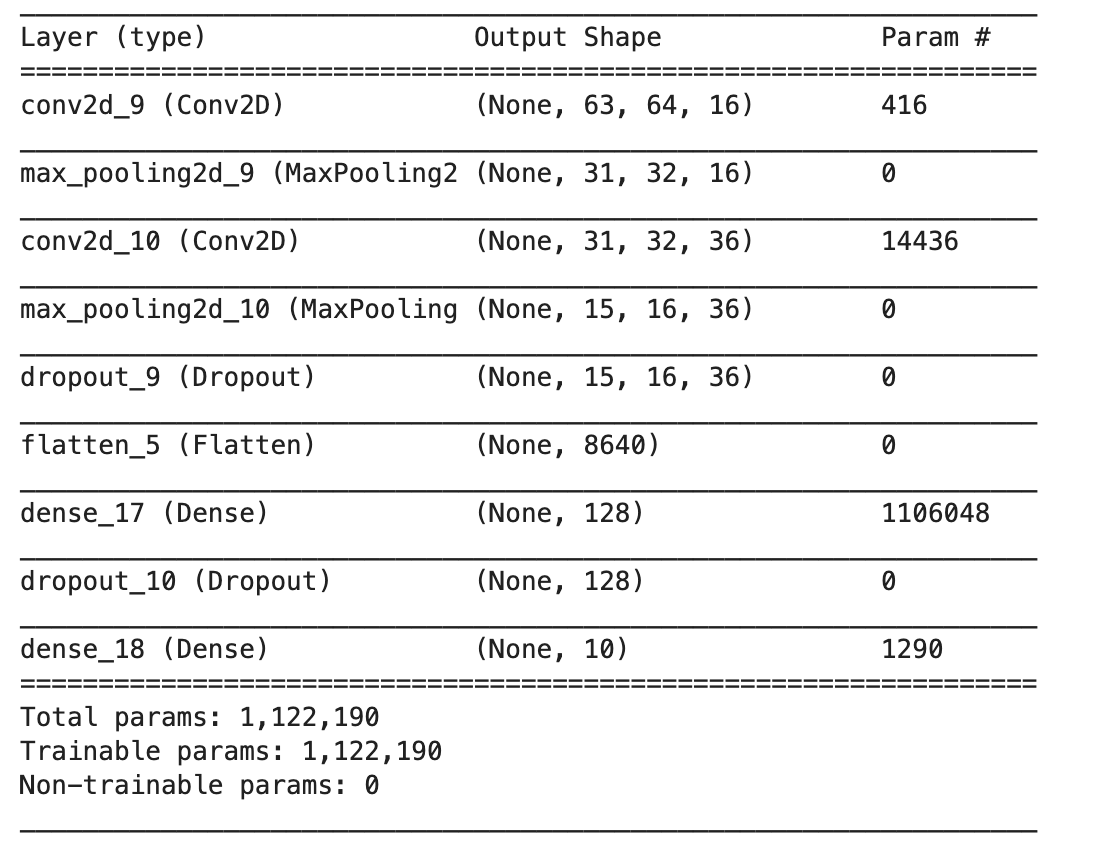
It’s an ANN layer that links the previous points to the next layer’s with the parameter we called ‘weight’. And it’s also the last layer of the function, therefore the final outpoint will have the same number like the class of classification.



*Figure 6-8: Hidden layer to predict output*

## Model

In our project we build the model with two convolution layer and two hidden layers.

****

*Figure 6-9: Model*

* As you can see the data set is a 63\*64 graph.
* At the first two convolution layers, we each use 16 and 36 filters with the 5\*5 pooling mask.
* In the middle, we use two dropout layers to reduce the overfitting.
* In the last part we use two hidden layers with 128 and 10 neural each.
* The total parameter in this model is 1,122,190.

**Parameter of the neural network**

We will explain this by study the model we build.

In all the filter and hidden layer nodes, we all have to put the bias as parameters.

* In the convolution, the parameters is decides by the filter matrix, therefore, we multiply the size of filter with the number of filter.

Example:

For the first convolution we use 16 different of 5\*5 filter, so we get (25+1)\*16 =

416.

In second convolution we use 36 different 5\*5 filter, but from the last layer there has already 16 of new feature map, so in each one will generate 36 more, so we get (25\*16+1)\*36 = 14436.

* In the hidden layer, because we use a fully connected network, so the total parameter will be multiplying the last layer points with the next.

Example:

In the dense\_18, there are 128 nodes at the last layer and it has 10 nodes in its own layer. So the parameters equals (128+1)\*10 = 1290.

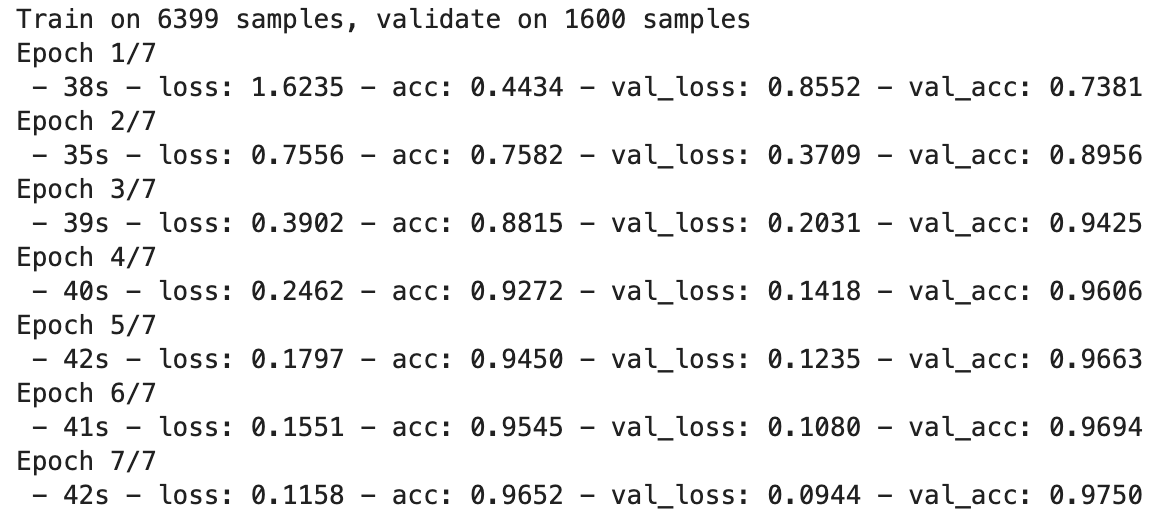
**Calculation method**

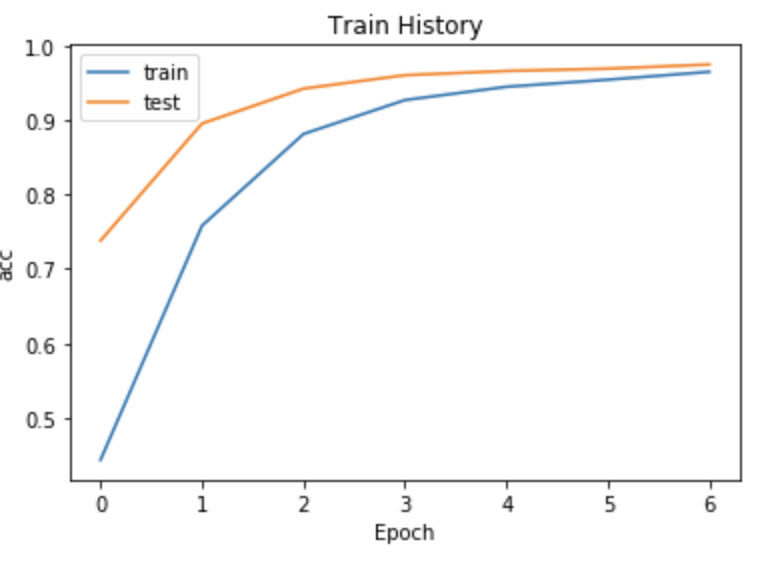
At first ,we will randomly assigned the parameter following a certain distribution, for example the neural cell in the convolution part will be arbitrarily generated to the number around zero.

In every interaction we have will use the ***Feedforward process*** to run all parameter in the network to and by compare to the label we will get the ***loss cost4***(the error between real value and predict value). Then, in our project we use ***batch gradient descent*** method to renew the parameter, the way to get the loss cost and gradient descent is called ***Backpropagation.***

## Results

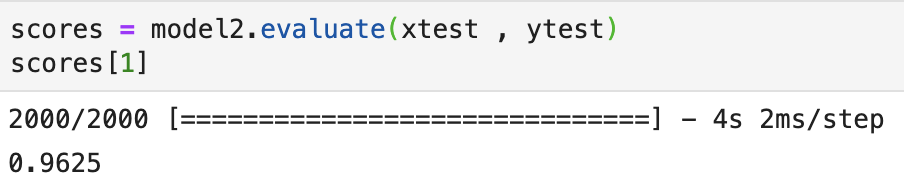
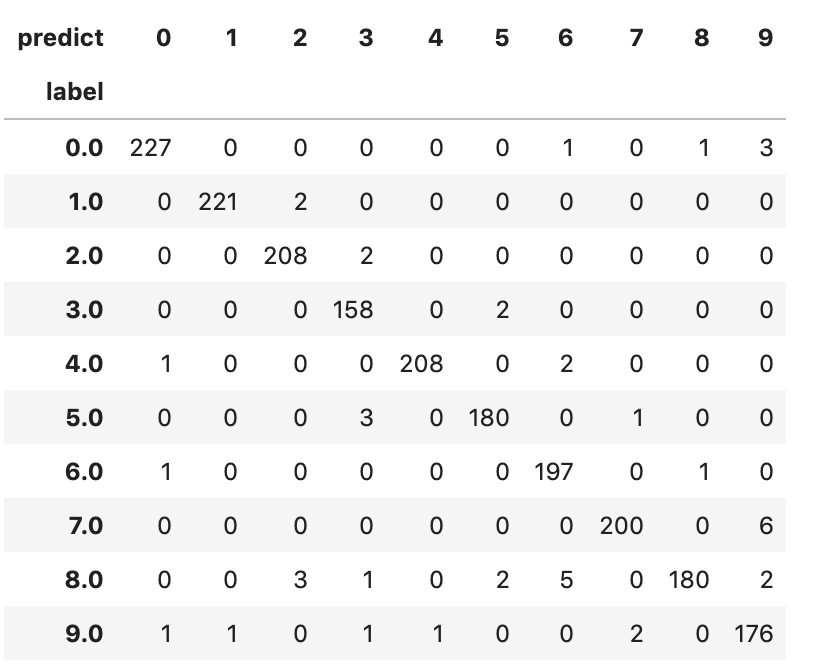
* The training sets has 8000 data and the testing sets has 2000 data.
* We use 7 interactions each with batch size of 300.
* Each interaction took about 40 seconds
* In the end we get 96.5% of accuracy.

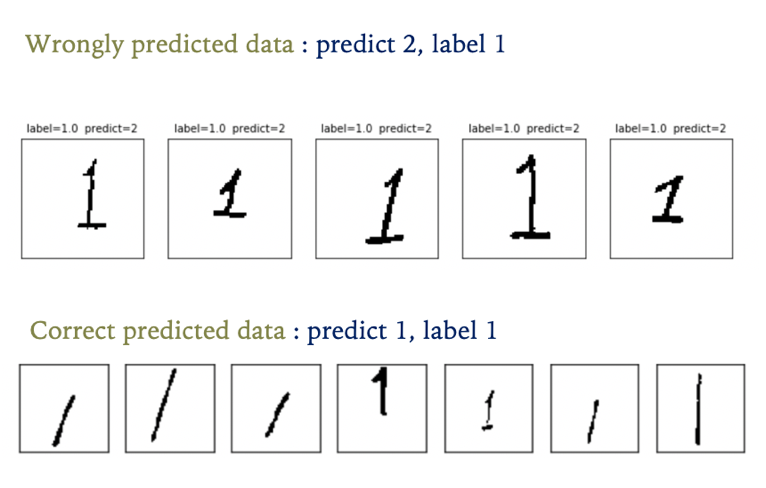


****

**Prediction**

By using to the prediction, we get 96% of accuracy. And the mostly confused number are 8-6 and 9-7. We supposed that is because the dataset we use is the hand writing numbers, so it is also confusing by the human eyes.





We extract the data that has label with 1, we can see the in the wrongly predicted number one it all has the head and the bottom line with doesn’t exist in the correctly predicted set, therefore, we can also suggest that the accuracy of prediction will affected by the training data set.

# Appendix

1. ANN

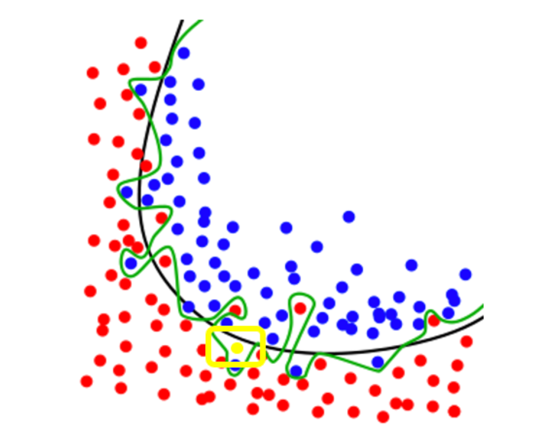
ANN is a collection of connected and tunable units (a.k.a. nodes, neurons, and artificial neurons) which can pass a signal (usually a real-valued number) from a unit to another. The number of (layers of) units, their types, and the way they are connected to each other is called the network architecture

1. Overfitting

The major reason to cause overfitting and the way to solve it is by:

1. There are too less data for training.
2. Too many parameters in the function.

* Dropout function
* Downsize the neural cells
* Regularization（weight decay）,add punishment for those who has oversized weight in the optimization.

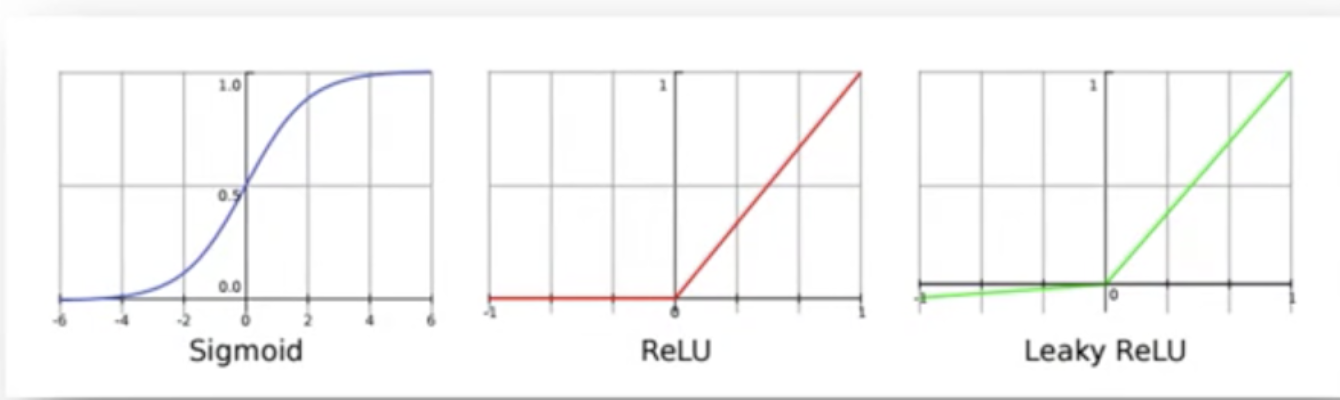


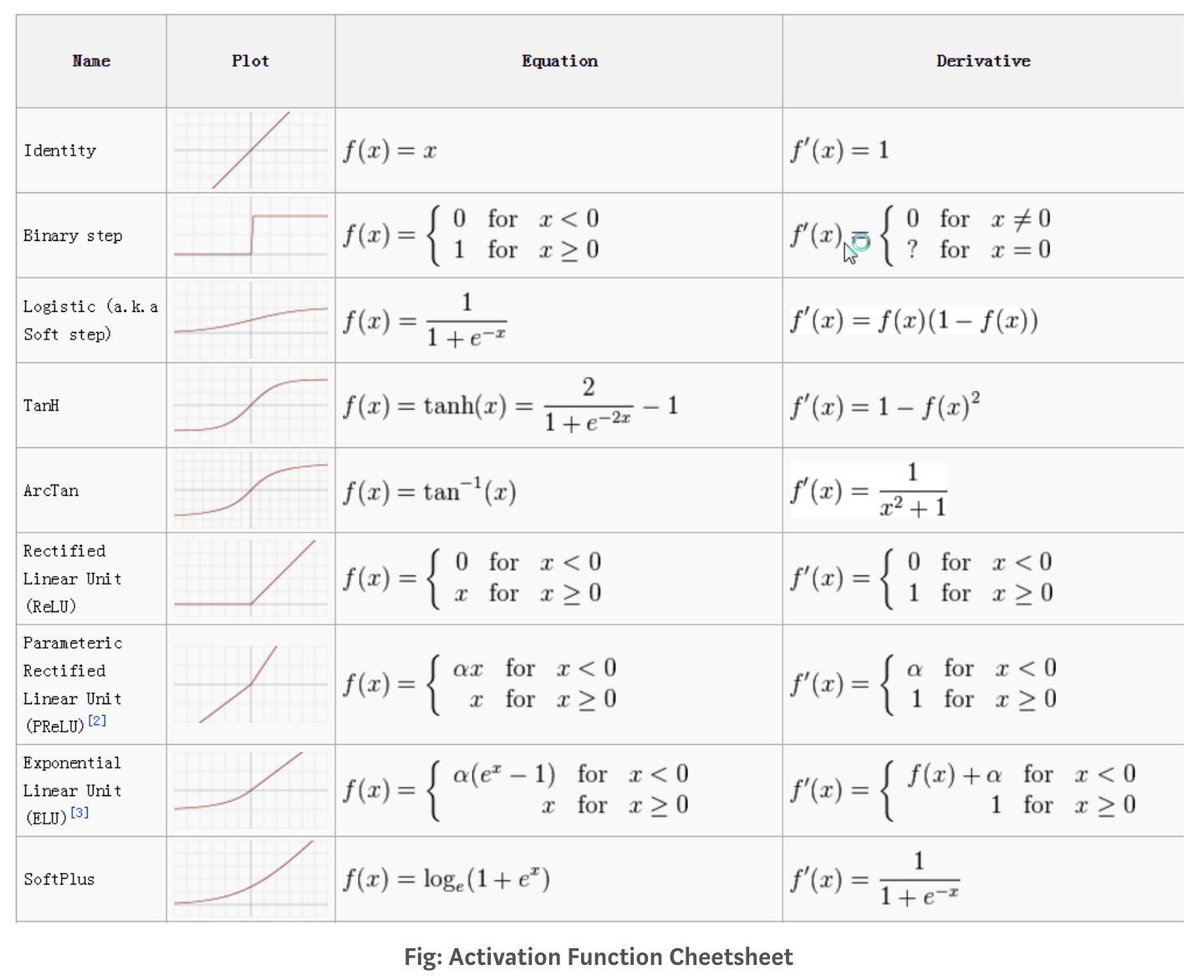
The green line is the overfitting curve,It overly verifies the date, the precision in the training set might be high, but if you put in prediction function, there will be errors.

1. Activation function

To transform the output by the specify function(mostly non-linear) in to a range. The purpose is to reduce the complexity and make it easy for the gradient descent.







1. Loss cost

By choosing one of the evaluation method to improve the error of prediction.

Ex: Accuracy, precise, f1-measure

For example:

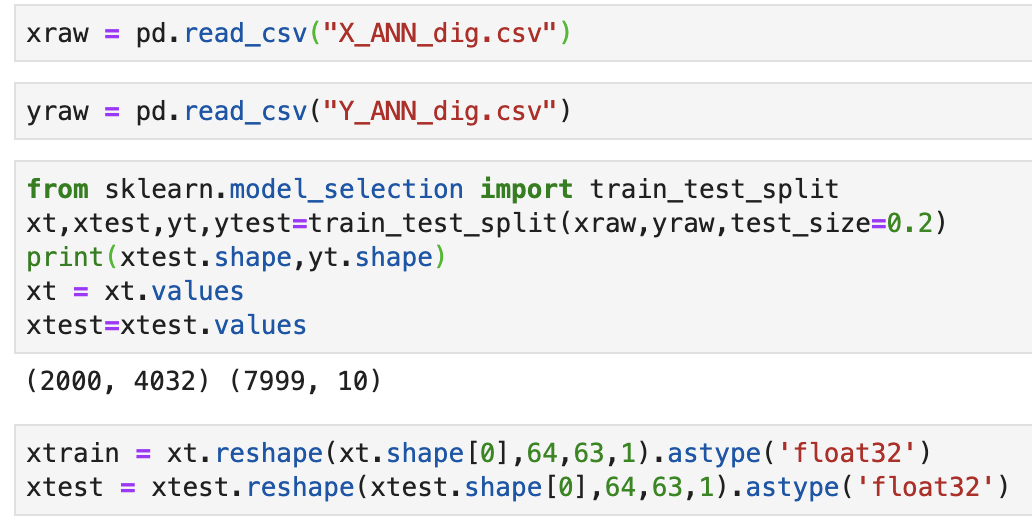
There are 100 students20girls and 80boys, we pick 50 students and there are 20girls and 30 boys.

* Accuracy : the number of correctly classified in the sample size/size of population（20+50）/100 =0.7

| Confusion m. | true | false |
| --- | --- | --- |
| positive | 20 | 30 |
| negative | 50 | 0 |

* Precise: the correctly predicted number in positive predicted set / positive predicted set, tp /(tp+fp) = 20/50=0.4.
* Recall: the positive correctly predictive set/ the correctly predicted set, = tp/(tp+nf) = 20/20=1.
* F1-measure: the mean of precise and recall. 2/f=1/r+1/p = rp2/(r+p) = 57.14%

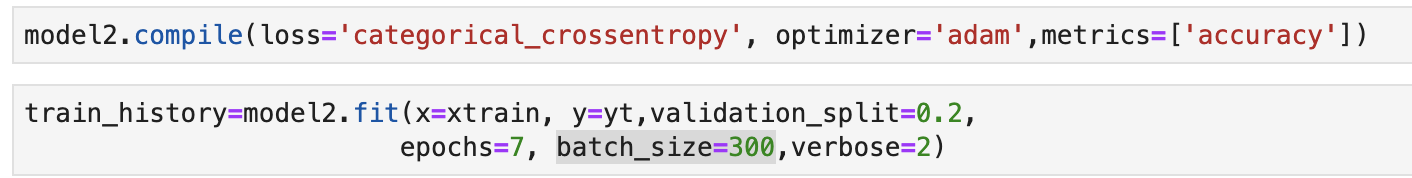
**Code**

****

*Figure 7-2: Data Processing*

****

*Figure 7-3: Model Building*



*Figure 7-4: Compile and training*

# Reference

* Artificial Neural Networks – Methodological Advances and Biomedical Applications (Anrej Krenker, Janez Bester and Andrej Kos)
* [https://medium.com/jameslearningnote/](https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC5-1%E8%AC%9B-%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E7%B5%A1%E4%BB%8B%E7%B4%B9-convolutional-neural-network-4f8249d65d4f)
* <https://mropengate.blogspot.com/2017/02/deep-learning-role-of-activation.html>
* <https://ithelp.ithome.com.tw/users/20001976/articles?page=3>
* <https://ai.stackexchange.com/questions/5546/what-is-the-difference-between-a-convolutional-neural-network-and-a-regular-neur>