CSC3034 Assignment 2

Neural Networks & Hybrid Intelligence Systems Report

Ah Kua

17000894 Khor Ze Yao

17031865 Kieren Gan Khye Tzer

17033853 Loo Zhen Ming

18077537 Joey Kua Choon Tzse

Table of Contents

Table of Figures

Introduction

In this report, we will discuss the neural network architecture that can be applied to a real-world dataset, as well as enhancements that can be made to the neural network by a hybrid intelligent system.

Problem

In this assignment, we are tasked with identifying and understanding a real-world dataset, to identify the possible applications an Artificial Neural Network may have on the dataset, and to implement a neural network to solve one of the identified applications.

We have also been tasked with identifying 2 Hybrid Intelligent Systems that may be used to enhance the implemented neural network.

Dataset

The dataset we have chosen for this assignment is the Optical Recognition of Handwritten Digits Data Set from the UCI Machine Learning Repository. The dataset consists of 5620 images of handwritten digits that have been extracted as normalised 32x32 bitmaps, which were then divided into 4x4 nonoverlapping blocks, resulting in each image being represented as a 8x8 matrix of integers in the range of 0 to 16. Due to this conversion process, the images have reduced dimensionality and invariance to small distortions. The dataset was prepared by C. Kaynak [1].

The dataset is split into two sets, a training set of 3823 training images and 1797 testing images. The training and testing datasets were prepared by 2 groups of 30 and 13 people respectively, with no overlapping participants and each with a unique handwriting style.

The datasets are available as .csv files, with each line representing an image composed of 65 integer numbers — the first 64 number represents a pixel of the 8x8 image in the order of left to right and top to bottom; the last digit represents the class of that image (and answer key to the digit the image represents). With this information, the dataset can be affectively split into training and testing sets, with each set containing the image data and each image's corresponding answer key.

The distribution of classes are as following:

Class	Training Set	Testing Set
0	376	178
1	389	182
2	380	177
3	389	183
4	387	181
5	376	182
6	377	181
7	387	179
8	380	174
9	382	180

Table 1 — Dataset class distribution

One possible application for neural networks for such a dataset would be to create a neural network capable of taking an input image and determining the corresponding digit represented in the image. The purpose of the neural network as such would be to classify an input image as one of the 10 Arabic numerals.

Neural Network

Architecture

A neural network is a network of artificial neurons, each having weighted to other neurons in different layers of the network whose purpose is to convert a meaningless input into meaningful output.

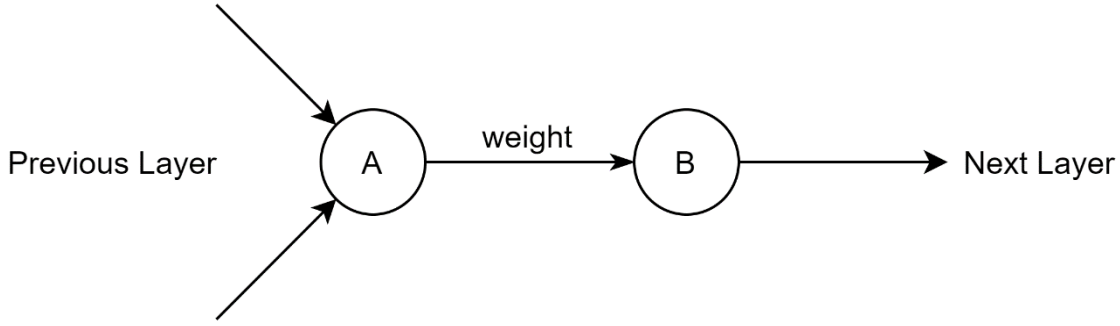

Figure 1 — Neuron

In a neural network, each neuron receives an input, and determines if it should activate based upon its activation function. Although there are various activation functions, one of the simplest forms is the summation & thresholding activation. That is to say, the neurons summarise the inputs it receives, and determines if the sum of the inputs is higher than its activation threshold. If true, the neuron is activated, and feeds its output to other neurons in the next layer. Neurons may feed forward either a constant value, for example 1 when activated and 0 when not, or a continuous value based upon the inputs it receives. For the sake of simplicity, this report will only discuss constant outputs.

Connections between neurons are generally weighted. As such, the output of a neuron must first have its value corrected by the weight on the connection before the corrected value is fed to the next neuron. This process is repeated, until the network reaches the output layer. Here, the final decision of the neural network is presented.

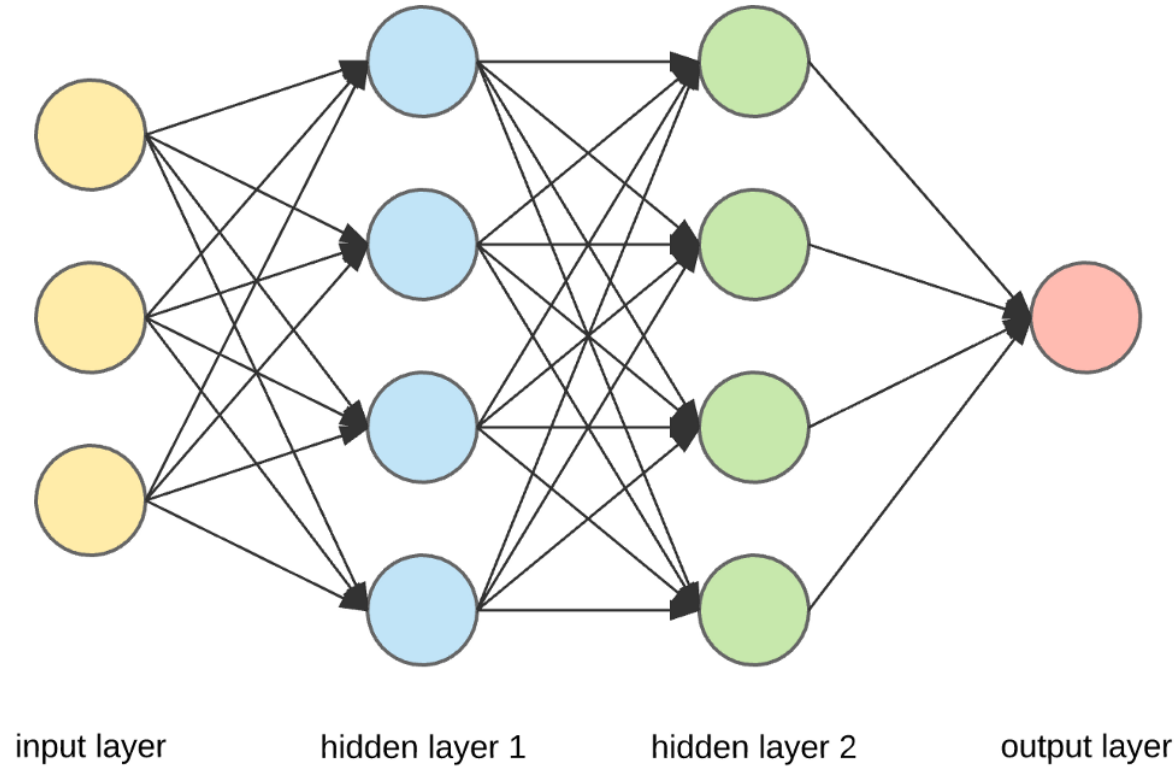

Figure 2 — Neural network architecture

Neural networks are generally constructed with 3 distinct segments — an input layer which receives raw input from the data source, an output layer which represents the decision made by the neural network, and one or more hidden layers of neurons between the input and output layers.

Different applications of neural networks may call for different configurations, such as different numbers of hidden layers or varying number of neurons in each hidden layer. In normal neural networks designed by humans, the number of hidden layers and neurons in each layer may be designated by the human arbitrarily, or via experience and trial & error. Variations such as evolutionary neural networks may have their architectures decided by other factors instead.

In order to train the neural network, the dataset must first be divided in training and testing sets. The training set is used to train the neural network and allow it to optimise the weights of the connections between neurons. Once training is

completed, the testing set is used to evaluate the performance of the neural network.

However, in order to train the neural network, we need a method to correct for errors and optimise the weights. As such, weight optimisation algorithms such as gradient descent are used to correct for errors in a process called backpropagation.

Implementation

In this assignment, we attempt to create a neural network capable of receiving an input image of a handwritten digit and determine the digit represented in the image.

First, we must determine the input parameters of the neural network. The 8x8 images in the dataset are stored as a list of 64 integer values ranging from 0 to 16, each value representing a pixel in the image in the order from top to bottom and left to right. As such, we can utilise this data by feeding the 64 pixels as our input parameters. As such, the input layer of our neural network shall be composed of 64 neurons, with each neuron's value being the brightness of a pixel in the image, represented as an integer value between 0 to 16.

As the purpose of our neural network is to classify the images into several classes (the numbers 0 to 9), we may represent them as neurons in the output layer. As such, the output layer will be composed to 10 neurons, each representing a class (digit) that the image may belong to.

We must also decide the flow of information in our neural network. Generally, information flow in neural networks can be categorised as:

- Feedforward Networks: In this model, the signals only travel in one direction, towards the output layer. Feedforward Networks have an input layer and a single output layer with zero or multiple hidden layers. They are widely used in pattern recognition.

- Feedback Networks: In this model, the recurrent or interactive networks use their internal state (memory) to process the sequence of inputs. In them, signals can travel in both directions through the loops (hidden layer/s) in the network. They are typically used in time-series and sequential tasks.

As the purpose of our neural network is to solve a classification problem, we will be using the feedforward network model.

We must also determine the activation function for our neural network. Traditionally, the logistic sigmoid function was used. In this activation function, the summation of activations and weights from the previous layer are scaled into a continuous number between 0 and 1. However, a sigmoid function results in a increasingly smaller gradient as the input size increases, thus resulting in loss of information and slower learning rates. As such, we will be using the reLU activation function for our implementation. In a reLU function, inputs larger than 0 are scaled linearly, as such the gradient remains constant even as the input increases. This allows for a faster learning rate and lowers convergence time in the neural network.

Finally, we must determine the number of hidden layers and neurons in each layer. Due to our lack of professional experience in implementing neural networks, we may either decide the number of neurons arbitrarily or through trial and error. As such, we decided to experiment by implementing a large number of neural networks with varying numbers of hidden layers and number of neurons, and evaluate their performance.

To decide the number of neurons in the network, we decided to follow the guidelines laid out by Heaton Research [2], which states the following:

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

In accordance to these guidelines, and with an input/output layer of 64 and 10 neurons respectively, we calculated the minimum number of neurons our network should contain to be 10 neurons, and the maximum to be 64. For the number of hidden layers, we tested an arbitrary number of layers between 1 to 4 layers. The number of neurons in each layer shall be the number of neurons currently being tested divided by the number of layers.

As the dataset we have acquired has already been split into training and testing datasets, we did not explicitly modify the dataset in anyway. In total, we tested a total of 113 network architectures with the provided datasets, the full results of which can be found here.

In summary, we found that a neural network with one hidden layer of either 29 or 62 neurons resulted in the best performance for our task. In our experiment, the network consisting of 62 neurons in its sole hidden layer converged faster than the network with 29 hidden neurons, so we chose the former as our architecture of choice. With this, the full architecture of our neural network is complete and can be implemented.

Results

Download Neural Network

Resultant neural network after 134 iterations of training.

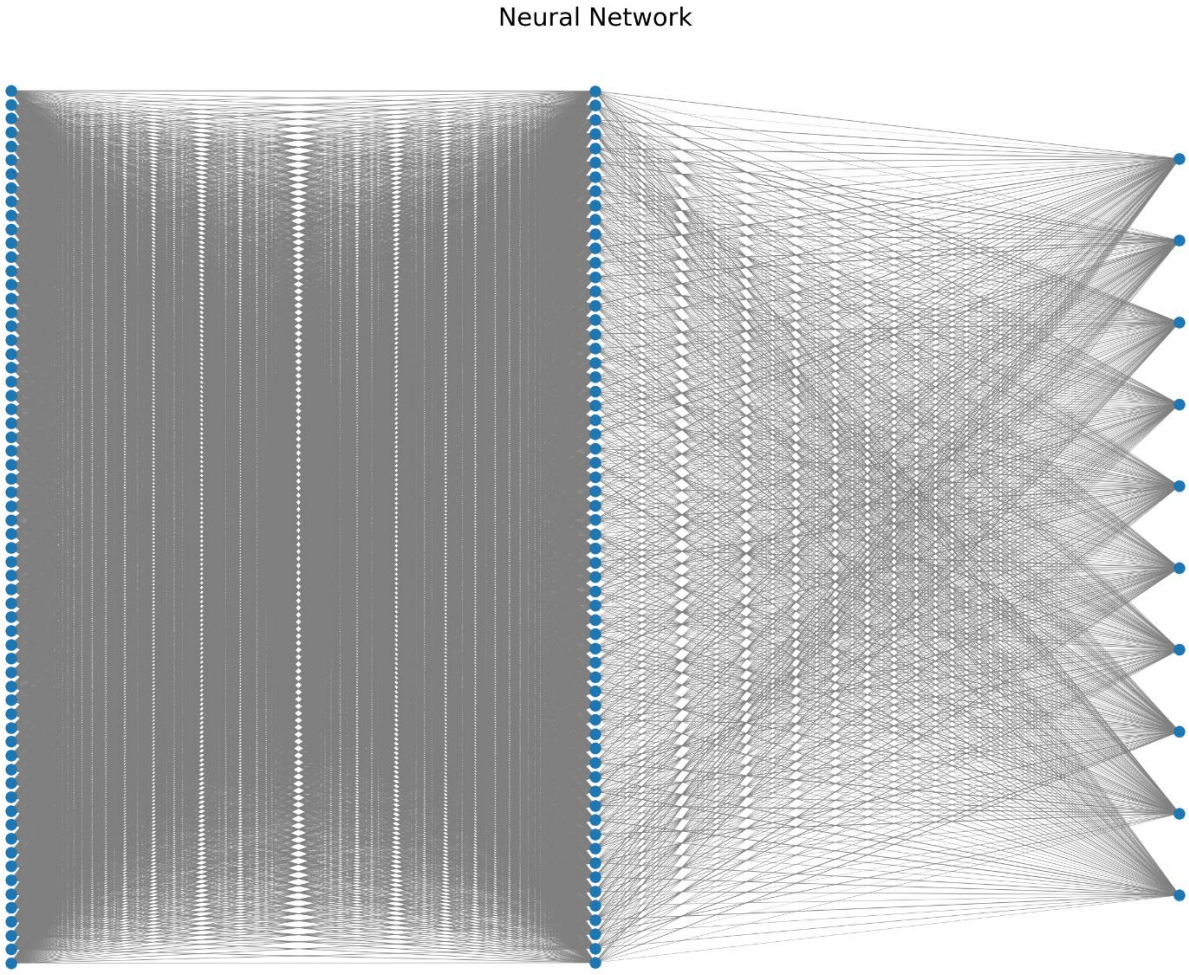

Figure 3 — Neural network result

Classification report

Class	Precision	Recall	f1-score	Support
0	1.00	0.99	1.00	178
1	0.96	0.99	0.98	182
2	0.97	0.98	0.97	177
3	1.00	0.95	0.97	183
4	0.97	0.98	0.98	181
5	0.93	0.99	0.96	182
6	0.99	0.99	0.99	181
7	0.99	0.97	0.97	179
8	0.95	0.94	0.94	174
9	0.94	0.95	0.95	180
Accuracy			0.97	1797
Macro Average	0.97	0.97	0.97	1797
Weighted Average	0.97	0.97	0.97	1797

Table 2 — Classification report

Confusion Matrix

Actual Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9	Total
0	177	0	0	0	0	1	0	0	0	0	178
1	0	181	0	0	0	0	1	0	0	0	182
2	0	1	173	0	0	0	1	0	2	0	177
3	0	0	5	174	0	2	0	0	1	1	183
4	0	1	0	0	177	0	0	1	2	0	181
5	0	0	0	0	0	180	0	0	0	2	182
6	0	0	0	0	2	0	178	0	1	0	181
7	0	0	0	0	0	6	0	170	1	2	179
8	0	5	0	0	1	2	0	0	160	6	174
9	0	0	0	0	2	2	0	1	1	174	180

Table 3 — Confusion matrix

Discussion

Overall, the neural network we have designed and implemented after 134 iterations of training on the 3823 training data produced an overall decent accuracy of 97.05% accuracy when tested with the 1797 testing data.

However, despite the neural network having achieved convergence in a relatively short number of iterations (some networks tested during the search for an appropriate network architecture required as many as 700+ iterations for convergence, none failed to converge), the network still failed to classify a number of images from the testing dataset.

Most notably, the neural network performed particularly badly when classifying 5s and 8s (relative to other digits). This presents a surprise to us. Before the implementation of the neural network, it was assumed that digits with high degrees of similarity, such as 3 and 8, would be more easily mistaken. However, the network was able to correctly distinguish 3s from 8s. From the confusion matrix, of the 9 false negatives for class 3, only 1 was misclassification as class 8. Similarly, none of the 14 false negatives for class 8 resulted in misclassification as class 3.

From the confusion matrix, it can be seen that class 5 accumulated the largest number of false positives at 13 of the 52 total misclassifications. The reason for this is unknown, and furthermore confusing as the network accumulated very few false negatives for class 5. In the dataset description, the image data provided for the training and testing datasets were written by different sets of participants. A possibility for such a high number of misclassifications may be due to one or multiple participants responsible for the training dataset had class 5s written in ways that looked similar to other classes. This finding is surprising to us, as human observers we see the number 5 as particularly unique, with little similarities to other classes, especially class 7 — the class that was misclassified as class 5 the most often.

On a more positive note, the network score particularly well when classifying 0s, and 1s. The high level of true positives for class 0 is unsurprising to us, however we had initially thought that class 1 may have resulted in a higher number of false positives for class 7. Overall, we are satisfied with the results of the implementation.

Hybrid Intelligent Systems

Now that we discussed the architecture and implementation of an Artificial Neural Network to solve the problem of classifying hand-written images, we will now discuss 2 hybrid intelligent systems that can be used to enhance the neural network.

Evolutionary Neural Networks

The first hybrid intelligent system that could be used to enhance the neural network are evolutionary neural networks. Whilst traditional neural networks such as that implemented in this report utilise backpropagation algorithms to identify optimal weights for connections between neurons. However, the selection of back-propagation algorithms suitable for different tasks is decided by the human, which may not always be appropriate. Furthermore, back-propagations cannot always guarantee an optimal solution for every problem.

The neural network topology is also generally decided arbitrarily or with limited heuristics by the human observer, and as such there is no guarantee that the chosen topology is ideal for the problem. An evolutionary neural network is able to rectify many of these concerns.

Evolutionary neural networks are neural networks where the optimal weights and topologies are determined using a genetic algorithm. In such a network, multiple neural networks are generated with varying weights and topologies. The parameters of each neural network are then encoded as chromosomes in the genetic algorithm, where the genetic algorithm will attempt to optimise its parameters using crossovers and mutations. The networks that performed best (highest fitness) are then used to improve and optimise future networks.

To implement such a network, the existing parameters must first be converted into a manner usable by a genetic algorithm. For this purpose, each connection with a

weight within the neural network can be represented as a weight. These weights could then be encoded as a chromosome that can be used by the genetic algorithm.

The fitness function of each chromosome can be determined by the performance of the network it represents. In the case of this implementation, the goal of the genetic algorithm can be to optimise the weights such that the maximum fitness (accuracy of the network) is achieved.

As an evolutionary algorithm shares many similarities to a traditional neural network in the way input/outputs are computed, these segments do not require any modifications. Instead, the goal of the evolutionary neural network is to provide an alternative way to optimise the weights and topology of the neural network (genetic algorithm vs backpropagation algorithm). As evolutionary neural networks are able to freely modify both the weights and topologies of the network as opposed to the backpropagation's sole purpose of optimising weights, it is assumed that an evolutionary neural network would be able to identify and form more meaningful connections between neurons and achieve a higher accuracy than a traditional neural network.

Neuro-Fuzzy systems

The second hybrid intelligent system that may be utilised to enhance the existing neural network such a task would be a Neuro-Fuzzy system. Neuro-Fuzzy systems perform in much the same way as a traditional fuzzy inference model, however instead of relying on expert information to determine the input/output membership functions and fuzzy rules, they are instead determined by a neural network. The advantage of such an architecture would be the dramatic enhancement to readability and understandability of the system. As we can see in the implementation of a traditional neural network above, it is hard to analysis and understand the decision-making process and reason for faults in a traditional neural network. A Neuro-fuzzy system can rectify this issue.

The implementation of such a system will require dramatic changes to the neural network architecture. A neuro-fuzzy system consists of 5 layers — the input layer, the input membership layer, the fuzzy rules layer, the output membership layer, and the defuzzification layer. For the purposes of this implementation, the input layer may remain the same as the traditional neural network we have already implemented, that being that each neuron represents the brightness of a given pixel of the provided image.

After the pixels are input into the system, the system must now determine the input membership functions of each input neuron in the input membership layer. In this layer, the weights of the connections between each input layer neuron and each input membership layer neuron will determine the membership function of the neuron for each member. Subsequently, each neuron in the input membership layer must forward its output to the fuzzy rule layer.

In this layer, the system must now determine the output membership of each input it has received. As before, each neuron in this layer will have a weighted connection to neurons in the next layer — in this case, the output membership layer. From this, the system is able to determine the membership function of each input image for each category. For this implementation, this membership function may represent the degree to which the system is confident that the input image is of a certain digit.

This output can then be fed to the defuzzification layer, where the system makes the final choice as to its decision for which class the input image belongs to.

With such an implementation, it can be easier to identify issues or flaws in the system, as well as increase readiblity and understandability of the system at large, whilst still able to solve the classification problem of this task.

Conclusion

Throughout this report, we have discussed the architecture and implementation of an Artificial Neural Network to solve the problem of classifying images of hand-written digits. After the implementation, we were able to discuss the results of the training and testing of the system and discover its flaws. Through this experience, we have also been able to identify 2 Hybrid Intelligent systems that may be implemented to supplement the system and to solve its issues.

References

[1] K. C., "UCI Machine Learning Repository: Optical Recognition of Handwritten Digits Data Set", *Archive.ics.uci.edu*, 2020. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits. [Accessed: 21- Nov- 2020]

[2] "The Number of Hidden Layers", *Heaton Research*, 2020. [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html. [Accessed: 21- Nov- 2020]