What effect does the architecture of a neural network have on its accuracy?

Computer Science NEA

Name: Callum [redacted]
Candidate Number: [redacted]

Centre Name: [redacted]
Centre Number: [redacted]

[redacted]

[redacted]

0 Contents

[redacted]

1 Analysis

1.1 Statement of Investigation

My investigation will explore how the architecture of a neural network affects its accuracy. I will do this by creating different basic neural networks that can be trained to detect digits 0-9 from a pre-drawn image.

The first thing I will be investigating is the fundamentals of how neural networks actually work along with how they are built. I will look at their structure and how the things they are composed of like nodes and layers are connected together to influence each other. I will be exploring the maths behind them such as the use of matrices and the calculations used to actually allow them to create an output from the data they are given. Studying further maths, I have a base knowledge of matrices and I already know it is a crucial thing that will be used through the project but I will need to research how they are actually applied in context. I also have a comprehension of graphs however again I will need to consider the complexities of using them to model neural networks reasonably as large networks contain many connections.

The next part of the investigation will involve me looking at the process of training neural networks. I will see how a network is first set up in order to start the training process, how the weights of the edges in the network determine which nodes are correlated and then how the system can correct itself and actually learn when it gets an answer wrong during training. This will involve researching the maths that allows networks to train such as how to create a cost function and use calculus to adjust the weights of arcs in a network based on its performance to cause the greatest improvement in accuracy.

The final stage of the investigation will look at different aspects used to answer my key question. I will consider different parts of a network that may affect accuracy such as:

- **The physical structure of the network e.g. the nodes and layers**
- **The maths e.g. using different functions**
- **The way the network is trained**

After comparing the accuracy of different trained networks I will be able to come to a conclusion that answers my main investigation question:
What effect does the architecture of a neural network have on its accuracy?

To help answer my key question I have decided on a set of more specific questions to look at:

1. **How is the rate that a network improves affected by changes in architecture?**
2. **What is the most accurate that I can train a neural network to be?**
3. **Which factors will actually affect the accuracy of a network?**
4. **Do specific changes e.g. more layers impact a network positively or negatively?**

1.2 Background

Neural networks are complex systems that artificially mimic the functions of biological neural networks in the brain. They have applications in artificial intelligence and machine learning. Overall the idea of neural networks is that they are able to learn from a set of training data by improving when they make a mistake. They can then accurately predict/answer something that they have never seen before. For example a network training on hand written images of digits 0-9 should be able to correctly identify a new image that it has never seen before after it has trained.

I have always been interested in the concept of machine learning as a whole and doing some basic research into it and finding out about neural networks has sparked curiosity in me to try out something involving them. This project should allow me to gain a greater understanding of neural networks.

From basic research, my current understanding of neural networks is that they are made up of layers which contain multiple nodes. Each node is connected to every other node on adjacent layers. These connections have weights which are adjusted during training to improve the accuracy of the network. As data runs through the network it gets interpreted going from the raw data at the first layer to the final result at the last layer. Usually the final result will be split between multiple nodes however the one with the greatest value will be used. The value of a node represents its activation which when looking at the last layer is interpreted as a percentage. This shows how likely the network thinks that what that node represents is the correct classification.

1.3 Expert

My expert, Eris, is a computer science student in the year above. Their NEA project was a survival simulation with a procedurally generated environment that includes enemies and collectables.They trained a learning algorithm which used neural networks to survive this environment. I think Eris will be helpful throughout my project as after doing her own investigation project relating to neural networks she will have a good understanding of some of the things that I will need to research. After briefly speaking to Eris about my project, they have confirmed that the performance of a network is definitely dependent on its structure and she has shown an interest in the project which leads me to believe that she will be a good choice as an expert.

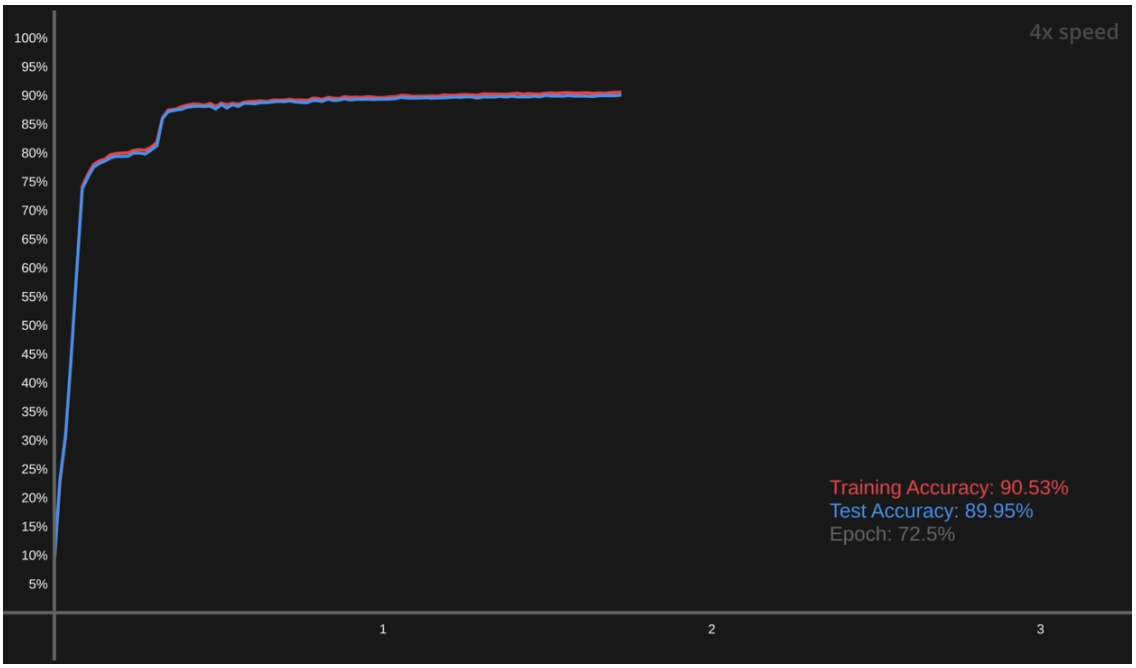
1.4 Research

1.4.1 Existing, similar programs

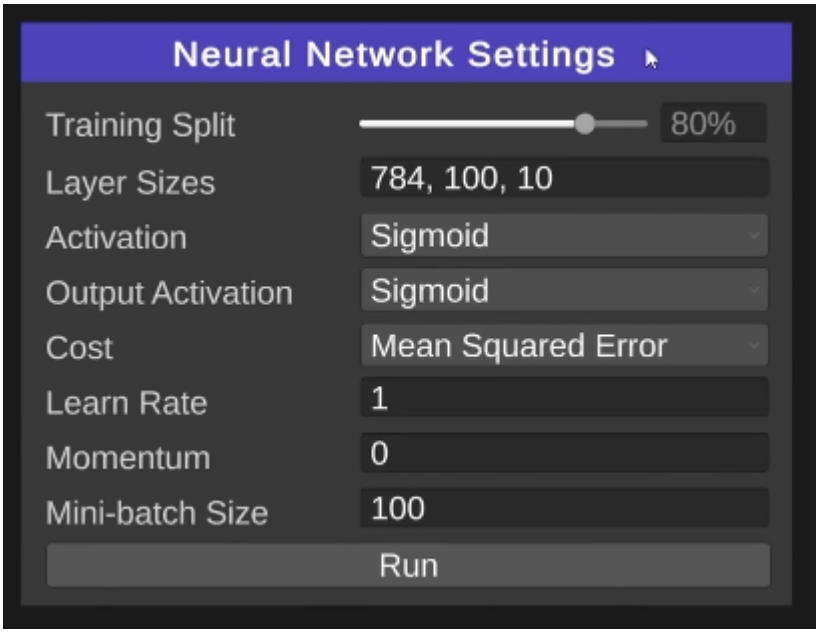
While researching neural networks I came across a video by a youtube creator called Sebastian Lague. In his video he explores creating a neural network and then he trains it to identify different sets of data including digits, doodles and images. The final product he released was a program which allows you to select between digits and doodles and the network will guess what you draw in the box provided.

Although I'm not aiming to make a final program like this I can gather useful information from the process he took to make it as well as looking at the project he has made available on GitHub which shows some other functionality.

When running the project from GitHub it presents a graph which shows the progress of the network as it trains. A feature similar to this would be useful for my project to log the rate of improvement in accuracy. This will allow me to better work out the impact of a change in architecture which is necessary to help answer question 1 and 4 in my statement of investigation.



Another thing included in his program that I will be adding to mine are a set of options for a specific network. This will be a crucial feature in my project as I will need to be able to change parameters and create different networks easily in order to compare them and look for trends. Options such as layer sizes and activation functions will be necessary parameters for my networks so I must include these.



Overall from looking at Sebastian's project I have found some features that will be absolutely necessary inclusions for my own investigation to be successful. I will however need to adapt these features to be more appropriate for my purpose for example menus will be needed to link these parts along with extra features that allow me more precise control of what the networks do.

Sources and credit:

- *Video - https://youtu.be/hfMk-kjRv4c*
- *GitHub Page - https://github.com/SebLague/Neural-Network-Experiments*
- *Demo - https://sebastian.itch.io/neural-network-experiment*

1.4.2 Potential abstract data types / algorithms

Graphs

Graphs are a data type that contains nodes which are points on the graph and edges/arcs which are the connections between the points. The nodes hold data and the edges can hold weights. A graph data structure will be an integral part of the project as it will be used to represent/model the neural network itself. A specialised type of graph will need to be used in order to include the layers that a neural network has. It is important that the graph is completely customizable in size (amount of layers and nodes in each layer) for this project as this will be the one of the main things that I intend to investigate and compare.

For my project I have decided the best way to use graphs will be to use nodes to represent layers in the network and use unweighted arcs to show which layers are connected. The nodes in the graph that represent layers will store information about the actual nodes in the network. Since all nodes in a neural network connect to each other based on their layers they don't need to be all individually connected as this would cause a problem with efficiency. Just connecting layers together as one abstracted graph is enough to show where there should be connections between the nodes of the network. Using other methods we can show these connections in a way that makes calculations easier.

Matrices

Matrices are another crucial data type that will be used throughout my program. They are similar to 2D arrays in the way that they hold values however they allow for operations to be carried out on them. There is a specific way of doing things such as multiplication which will need to be implemented.

Matrices are usually written like this:

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

An example of matrix multiplication:

$$\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix} = \begin{bmatrix} a_1a_2 + b_1c_2 & a_1b_2 + b_1d_2 \\ c_1a_2 + d_1c_2 & c_1b_2 + d_1d_2 \end{bmatrix}$$

For my project matrices will be a necessary companion to the graphs to show the more detailed connections between nodes and hold the weights of the arcs between them. Doing it this way will avoid wasting space with large adjacency matrices that would be used to represent the entire network as a lot of nodes can never be connected. They will also be used to perform big calculations which should be faster than doing it regularly.

Activation Functions

The number a node holds in a network is its activation value. The activation of one layer's nodes will determine the activation of the next layer's nodes through a set of calculations followed by an activation function.

A specific node's activation can be calculated by the sum of the previous layers activation values each multiplied by the weights of the connections between them and the node we are calculating and then a bias for activity/inactivity is added onto the result. This final result is what's put through an activation function to give the node its activation value. Since this calculation needs to be done many times on many different nodes it is more efficient to use matrices to handle the calculations.

It can be represented like this:

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$

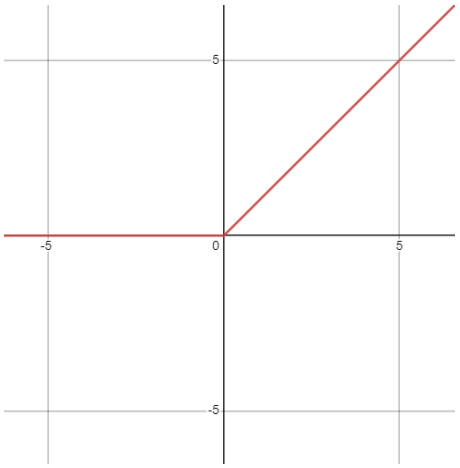
Sigma represents the activation function.
k and n represent the size of each layer respectively.

It can be written shorthand where the matrices are represented by letters:
$\sigma(\mathrm{Wa} + \mathrm{b})$

[redacted]

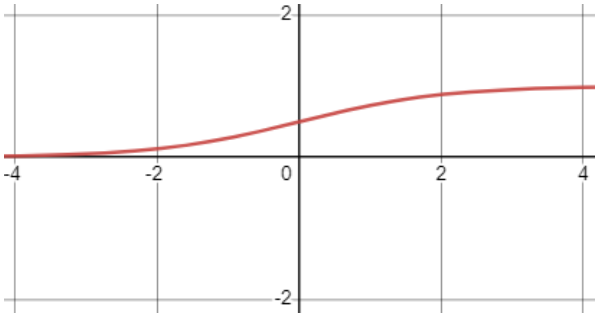
The activation functions I have researched are as follows:

ReLU - f(x) = max(0,x)



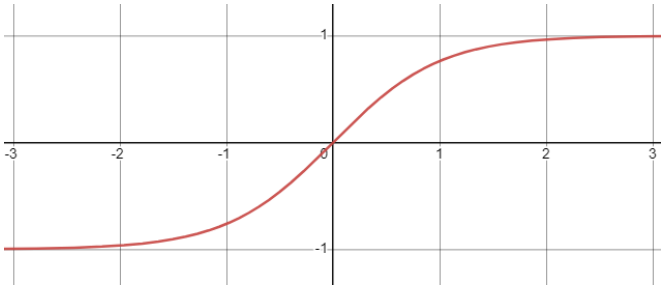
ReLU which stands for rectified linear unit takes the max value out of 0 and the input. This means any negative number will result in an output of 0 and anything positive will output as itself

Sigmoid - $\frac{1}{1+e^{-x}}$



The sigmoid function compresses the inputs between 0 and 1, The more negative the number the closer to 0 it will be and the more positive, the closer to 1 it will be.

Tanh - tanh(x)



The tanh function compresses the inputs between -1 and 1. The more negative the number the closer to -1 and the more positive the closer to 1.

[redacted]

Backpropagation
Backpropagation is the core algorithm behind how neural networks learn. It involves a lot of calculus. Firstly a cost function needs to be established. The cost is the sum of the squares of the differences between the result of any given output the neural network gives and the value we want it to be. The cost will be small when the network is correct and large when it is unsure or wrong. Taking the average cost of many different attempts will show overall how ‘bad’ the network is. The cost function aims to minimise the cost.

Using a column vector of all the weights and biases along with the cost function you can find how each of these values need to be adjusted to start to minimise the cost. For any sort of function the gradient of it at a point as a vector will give the direction of steepest increase. Since it needs to be minimised the negative of this gradient is what's important as this shows the direction of steepest decrease.

To sum this up, putting the weights and biases as a column vector into the negative gradient function of the cost function will output a column vector showing how each value needs to be changed in order to cause the biggest decrease in cost.

1.4.3 Interview

1. **What different factors do you think would be worth investigating to see if they have any impact on accuracy?**

   The simple parameters to investigate would be Learning Rate, and the Layer Structure / Size. You could quite easily set up a series of tests testing different parameters against each other. This would be best if automated by a manager program, which then collects and logs data for you which you can compare at a later date. You could also investigate different activation functions, but the more parameters you try to investigate, the more training time and data you will have. You may want to optimise your Network heavily such that you can collect data quickly.

   You could also investigate different types of Neural Networks such as Convolutional Neural Networks, but this would require more development time.

2. **How customisable should a particular network be, is there anything that would be best to keep consistent for every network?**

   I think it would be best to be able to change as much as you can within the network. I don’t believe there are parameters you’d want to keep consistent, other than maximum and minimum values for select parameters. So you don't end up with very large Neural Networks.

3. **Would you say that creating graphs to compare different networks and their accuracy is a good way to look for trends?**

   In my experience, storing as much data as you can will help you analyse the different effects of tuning parameters. If you can present this in a meaningful graphical way, such as plotting a variable against time, it will make it much easier to locate and write

about these trends. If you're storing data it may be sensible to store it in a Database or an appropriate file format such as JSON.

4. **What type of activation function do you think would be most suitable for the networks to use?**

   The most standard activation function would be Sigmoid, though I've found success with TanH, ReLU and ArSinH. This would certainly be a good thing to investigate, seeing as activation functions can drastically affect your Networks Performance between different applications.

5. **When saving data about a network after it has trained, do you think it is important to keep the entire network so that it can be loaded again or only the information that will be useful when looking at its accuracy?**

   If you wanted to resume the training of a particular Network with specific parameters, it would be best to store the entire weights and biases of the Network. This would allow you to retrieve additional information about a particular Network if you wanted.

1.4.4 Evaluation of interview

The interview with my expert has been helpful in clarifying certain things I was unsure about, helping to direct my research and establishing certain things my program should include.

My Observations from the interview:

- Main things to investigate are different learning rates and activation functions along with the structure and sizes of layers.
  Objectives 5.2 and 2.1 to 2.3

- Networks should have a lot of customisation in their parameters.
  Objectives 2.1 to 2.3

- Presenting the resulting data in a meaningful graphical way will be of use when locating and writing about trends.
  Objective 7.2

- Data should be stored in an appropriate format.
  Objective 4.3

- Sigmoid TanH and ReLU are good activation functions to try.
  Objectives 2.3 and 1.7 to 1.9

- Activation functions should impact performance of a network greatly.
  Objectives 2.3 and 1.7 to 1.9

- It is very important to store everything about a network so that it can be used again to obtain additional information.
  Objective 4.6

1.4.5 Further Discussion

After completing my prototype I spoke with Eris again to gather feedback and see if anything should be changed before settling on a final design. They agreed with my concerns about the way I had structured my network class and suggested I just store an array of layers rather than linking layers together in a sort of recursive tree. I think this change will be beneficial and allow the program to function more iteratively than recursively which may speed it up. It also will help me to avoid complications when addressing the layers in a network. They also provided me with some help in understanding the backpropagation algorithm which allowed me to start considering the best way to implement it.

1.5 Prototype

1.5.1 Prototype Objectives

For my prototype I want to focus on implementing some base features that will be crucial to the project moving forward. I also want to explore some different ways of implementing things to see which way would be the most useful.

The things I have decided to explore in my prototype are:

- Matrices
- Graphs
- Making the basis for a network

My plans are to implement matrices with all the basic calculations I know will be useful at the moment. I also need to work out the easiest way to create and access a matrix. After this I want to figure out how I can use a graph appropriately to represent a network. Then I can combine these aspects to create the basis of a network which I will later use to actually build onto and make functional.

1.5.2 Initial Matrix Implementation

For my first attempt at implementing a class for a matrix I wrote this code:

```
1  internal class Matrix
2  {
3      private int rows, columns;
4      public int[,] contents;
5
6      public Matrix(int rows,int columns)
7      {
8          this.rows = rows;
9          this.columns = columns;
10         contents = new int[rows,columns];
11     }
12 }
```

It includes the basic features that are necessary for a matrix, however after writing code using the class and testing it I decided this implementation was difficult to use when assigning values to matrices due to the clunky way of accessing specific digits in the matrix.

```
myMatrix.contents[i, j]
```

I also wrote some static methods for this class which allowed for basic mathematical operations that included addition, subtraction and multiplication.
My code for the addition:

```
1  public static Matrix add(Matrix a, Matrix b)
2  {
3      if (!(a.rows == b.rows && a.columns == b.columns))
4      {
5          throw new Exception("Invalid Addition");
6      }
7      else
8      {
9          Matrix c = new Matrix(a.rows, a.columns);
10         for (int i = 0; i < a.rows; i++)
11         {
12             for (int j = 0; j < a.columns; j++)
13             {
14                 c.contents[i, j] = a.contents[i, j] + b.contents[i, j];
15             }
16         }
17         return c;
18     }
19 }
```

Again although the code did what it was supposed to it was difficult to use efficiently due to the way it had to be written out.

```
Matrix.add(myMatrixA,myMatrixB)
```

From this initial implementation I had already coded the specific ways some calculations would function along with finding out what parts were not ideal and needed to be refined.

1.5.3 Final Matrix Implementation

Using what I had gathered from my initial implementation I worked out some solutions to help clear up any code I would need to write to access things. I decided to start from fresh and use the other code for reference when creating my new code.

Firstly to combat having to define every single digit in a matrix I decided to have a constructor which just took a 2D array in and built the rest of the matrix from that information. This doesn't have a huge impact on the whole as ultimately the values in each matrix will not have to be assigned by hand however it is useful for making testing faster.

The new code:

```
class Matrix
{
    int rows;
    int columns;
    double[,] data;

    public Matrix(double[,] data)
    {
        rows = data.GetLength(0);
        columns = data.GetLength(1);
        this.data = data;
    }
}
```

Next to allow for easy access to the contents of a matrix I implemented an indexer which lets me treat the matrix as if it was just the 2D array it holds when looking at its contents.
The code for this:

```
private double this[int i, int j]
{
    get { return data[i, j]; }
        set { data[i, j] = value; }
}
```

Instead of using static methods for the calculations I decided to use operator overloading as it would allow for simple expressions to be written more smoothly like a mathematical calculation would. I also implemented my own exception which would let me know if any calculations were not possible to avoid any errors.
The exception:

```
class MatrixNonConformableException : Exception
{
    public MatrixNonConformableException(string operation) :
base($"{operation} cannot be performed on these matrices")
    {

    }
}
```

[redacted]

Addition:

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1 + b_1 & a_2 + b_2 \\ a_3 + b_3 & a_4 + b_4 \end{bmatrix}$$

```
1  public static Matrix operator +(Matrix a, Matrix b)
2  {
3      if (a.rows != b.rows || a.columns != b.columns)
4      {
5          throw new MatrixNonConformableException("Addition");
6      }
7      double[,] result = new double[a.rows, a.columns];
8      for (int i = 0; i < result.GetLength(0); i++)
9      {
10         for (int j = 0; j < result.GetLength(1); j++)
11         {
12             result[i, j] = a[i, j] + b[i, j];
13         }
14     }
15     return new Matrix(result);
16 }
```

Subtraction:

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} - \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1 - b_1 & a_2 - b_2 \\ a_3 - b_3 & a_4 - b_4 \end{bmatrix}$$

```
1  public static Matrix operator -(Matrix a, Matrix b)
2  {
3      if (a.rows != b.rows || a.columns != b.columns)
4      {
5          throw new MatrixNonConformableException("Subtraction");
6      }
7      double[,] result = new double[a.rows, a.columns];
8      for (int i = 0; i < result.GetLength(0); i++)
9      {
10         for (int j = 0; j < result.GetLength(1); j++)
11         {
12             result[i, j] = a[i, j] - b[i, j];
13         }
14     }
15     return new Matrix(result);
16 }
```

Multiplication:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix}$$

```
1  public static Matrix operator *(Matrix a, Matrix b)
2  {
3      if (a.columns != b.rows)
4      {
5          throw new MatrixNonConformableException("Multiplication");
6      }
7      double[,] result = new double[a.rows,b.columns];
8      for (int i = 0; i < result.GetLength(0); i++)
9      {
10         for (int j = 0; j < result.GetLength(1); j++)
11         {
12             for (int k = 0; k < a.columns; k++)
13             {
14                 result[i, j] += a[i, k] * b[k, j];
15             }
16         }
17     }
18     return new Matrix(result);
19 }
```

I also added two extra functions which I knew might be useful at some point.

Scalar Multiplication:

$$a \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} = \begin{bmatrix} ab_1 & ab_2 \\ ab_3 & ab_4 \end{bmatrix}$$

```
1  public static Matrix operator *(int a, Matrix b)
2  {
3      for (int i = 0; i < b.rows; i++)
4      {
5          for (int j = 0; j < b.columns; j++)
6          {
7              b[i, j] = a * b[i, j];
8          }
9      }
10     return b;
11 }
```

Transverse:

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix}^T = \begin{bmatrix} a_1 & a_4 \\ a_2 & a_5 \\ a_3 & a_6 \end{bmatrix}$$

```
1  public static Matrix operator ~(Matrix a)
2  {
3      double[,] result = new double[a.columns, a.rows];
4      for (int i = 0; i < result.GetLength(1); i++)
5      {
6          for (int j = 0; j < result.GetLength(0); j++)
7          {
8              result[j, i] = a[i, j];
9          }
10     }
11     return new Matrix(result);
12 }
```

1.5.4 Networks

To create the skeleton of a network which is easily customisable I decided to combine a graph data structure using object oriented programming along with the matrices I have previously implemented. The first class I created was a node which would hold an activation value.
The code:

```
1  class Node
2  {
3      public double activation;
4  }
```

After this I created a layer class. A layer contains a reference to the previous and next layers in the tree as it is defined recursively. It contains an array of nodes that the layer holds along with two matrices which show the weights of the connections to the previous and next layer. There is also a random variable which is used to randomly fill the weights in a network.
The code for this:

```
1  class Layer
2  {
3      public Layer prev;
4      public Layer next;
5      public Node[] nodes;
6      public Matrix prevweights;
7      public Matrix nextweights;
8      public static Random rnd = new Random();
9  }
```

The class also contains two constructors one of which is called originally to create the first layer in the tree and the second which runs until the required number of layers have been created.
The Code:

```
public static Matrix operator ~(Matrix a)
{
    double[,] result = new double[a.columns, a.rows];
    for (int i = 0; i < result.GetLength(1); i++)
    {
        for (int j = 0; j < result.GetLength(0); j++)
        {
            result[j, i] = a[i, j];
        }
    }
    return new Matrix(result);
}
```

To hold everything together I created a Network class which is what I would define in the code to create a new network.
The code for this:

```
public Layer(int layers, int[] nodesPerLayer)
{
    nodes = new Node[nodesPerLayer[nodesPerLayer.Length - layers]];
    prev = null;
    prevweights = null;
    double[,] nextdata = new double[this.nodes.Length,
nodesPerLayer[nodesPerLayer.Length - layers + 1]];
    nextweights = new Matrix(Fill(nextdata));
    next = new Layer(this, layers - 1, nodesPerLayer);
}
public Layer(Layer prev, int layers, int[] nodesPerLayer)
{
    this.prev = prev;
    prevweights = ~(prev.nextweights);
    nodes = new Node[nodesPerLayer[nodesPerLayer.Length - layers]];
    if (layers - 1 == 0)
    {
        next = null;
        nextweights = null;
    }
    else
    {
        double[,] nextdata = new double[this.nodes.Length,
nodesPerLayer[nodesPerLayer.Length - layers + 1]];
        nextweights = new Matrix(Fill(nextdata));
        next = new Layer(this, layers - 1, nodesPerLayer);
```

```
25          }
26      }
```

This allows for things to be neatly defined in just one line of code in the main program.

1.5.5 Prototype evaluation

Overall I think my prototype achieved the goals set in the prototype objective. I am happy with my implementation of matrices and it will be easy to add any extra functionality that I find necessary later on. I have explored creating a network however I may have to adapt this in the future when I am actually implementing the algorithms that will make the networks functional as the implementation may be too complicated to address values such as weights and biases of a specific layer easily.

1.6 Objectives

1.6.1 Program overview

I aim for my program to have three main functions:

- Creating, loading and saving neural networks
- Training neural networks using the MNIST data set
- Displaying Information about a networks structure and accuracy after training

1.6.2 Objectives

1. **Matrices**
    - 1.1. The program will implement dynamically sized square and non square matrices
    - 1.2. The program will allow matrices to be multiplied together
    - 1.3. The program will allow matrices to be added/subtracted together
    - 1.4. The program will allow matrices to be multiplied by a scalar
    - 1.5. The program will allow the transpose of a matrix to be calculated
    - 1.6. The program will allow the determinant of a matrix to be calculated
    - 1.7. The program will allow the sigmoid function to be applied to a matrix
    - 1.8. The program will allow the tanh function to be applied to a matrix
    - 1.9. The program will allow the ReLU function to be applied to a matrix
    - 1.10. The program will throw an appropriate exception when a calculation is not possible
2. **Creating new networks**
    - 2.1. The program will ask the user for the number of hidden layers in the network
    - 2.2. The program will ask the user for the number of nodes in each hidden layer
    - 2.3. The program will ask the user what type of activation function they want the network to use

3. **Using MNIST data**
    - 3.1. The program can read in training images
    - 3.2. The program can read in test images
    - 3.3. The program correctly interprets the image data to provide an input to the network
    - 3.4. The program can read the corresponding label to an image and compare it to the networks result
4. **Storing/Accessing created networks**
    - 4.1. The program will allow the user to save a network to a file
    - 4.2. The program will allow the user to choose a name for the file
    - 4.3. The program will write the data about a network to a binary file
    - 4.4. The program will allow the user to load a network from a file
    - 4.5. The program will ask the user which file it should load
    - 4.6. The program will read the data from the binary file to reconstruct the network
5. **Training**
    - 5.1. The program will allow the user to train the network
    - 5.2. The program will ask the user to set a learn rate for the training session
    - 5.3. The program will use a set of training data
    - 5.4. The program will use forward propagation to calculate the activations of nodes in the network for a specific image
    - 5.5. The program will implement the backpropagation algorithm to increase a networks accuracy
    - 5.6. The program will calculate the derivatives of each weight and bias based on the cost function
    - 5.7. The program will calculate the average derivative for each weight and bias across a batch of data
    - 5.8. The program will adjust each weight and bias by the negative of their average derivative multiplied by the learn rate
6. **Testing**
    - 6.1. The program will allow the user to test the network
    - 6.2. The program will use a set of test data
    - 6.3. The program will use forward propagation to run the data through the network and determine a result
    - 6.4. The program will compare the result the network has determined to the correct label
    - 6.5. The program will return an overall accuracy from the test
7. **Presenting and logging data**
    - 7.1. The program will allow the user to view the information about the currently loaded network
    - 7.2. The program will display a graph of the accuracy against the number of batches completed in the training session
    - 7.3. The program will allow the user to save the graph of a training session

2 Design

2.1 High Level Overview

The main aim for the program is to allow the user to:

- Create networks
- Save and load networks
- Test networks
- Train networks

The user can access these functions through a main menu in a console application.

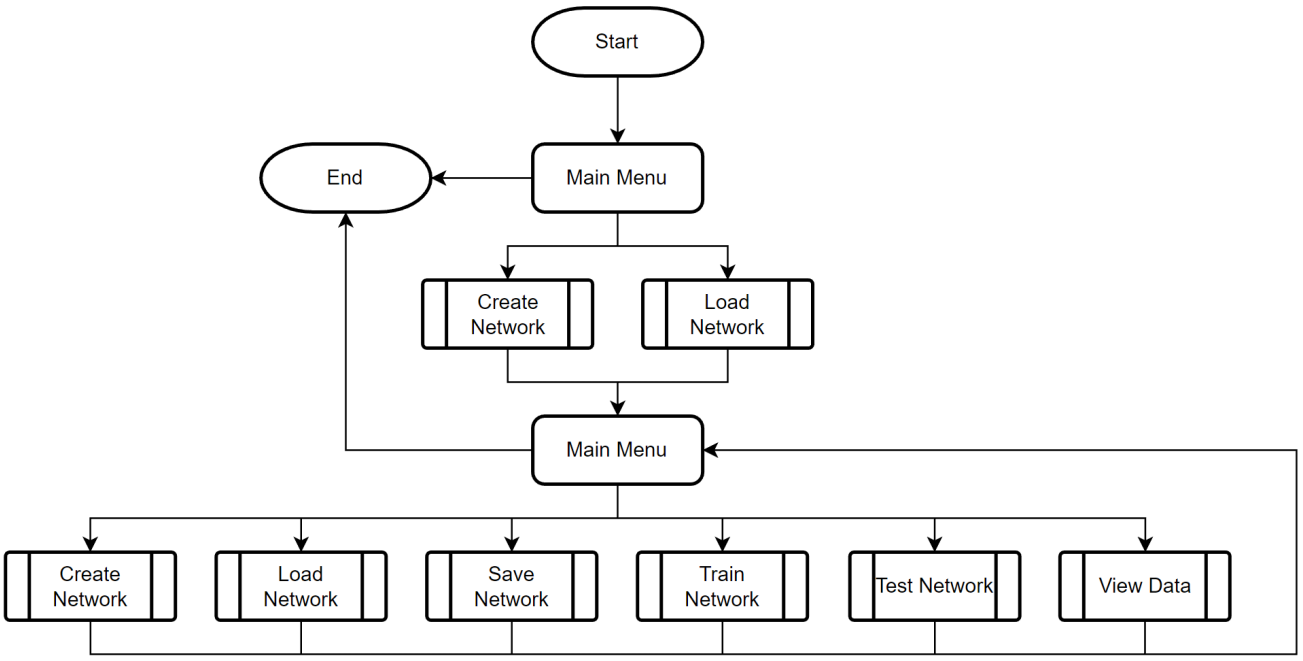


Figure 1 - A Flow Chart illustrating how the user will move through the program

Figure 1 displays the way the user will move through the program and access the different features. This abstracted diagram only shows the options available to the user when the program is in one of the two different states at the main menu, having no network loaded and having a network loaded. The Main Menu boxes show when the user is being displayed the main menu and the various subroutine boxes branching from this show the selectable options given to the user.

Figure 2 shows some of the selectable options in more detail. These are still abstracted but display how the user will have to interact with each option such as the inputs they have to give or things that will be displayed to them. These options do more behind the scenes to actually handle the interaction with networks such as creating a new one but the diagram displays only what the user will see and interact with.

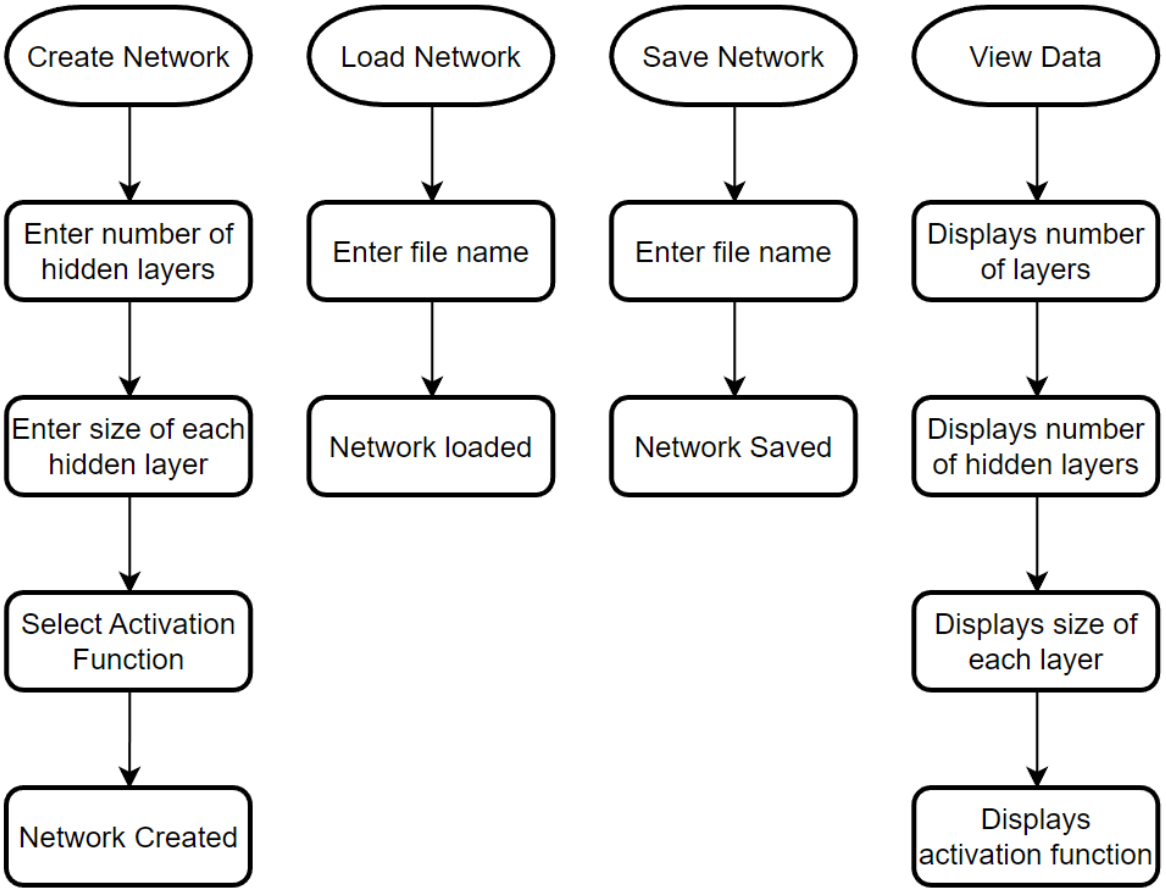


Figure 2 - Option Subroutines

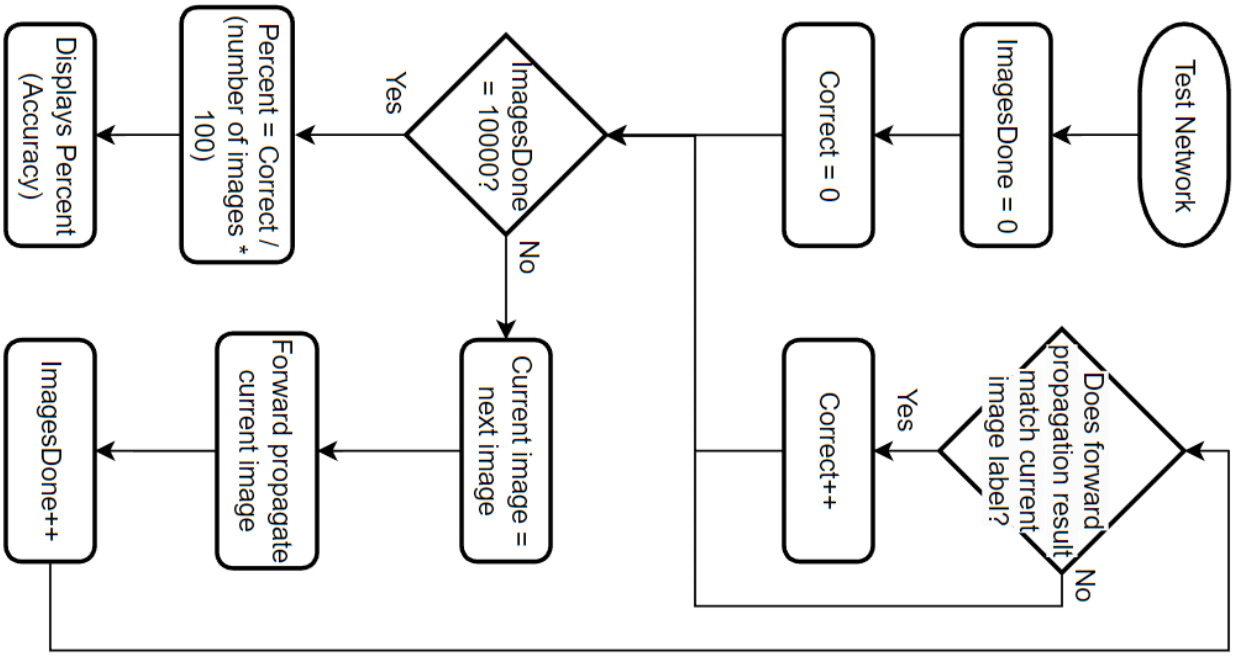


Figure 3 - Testing

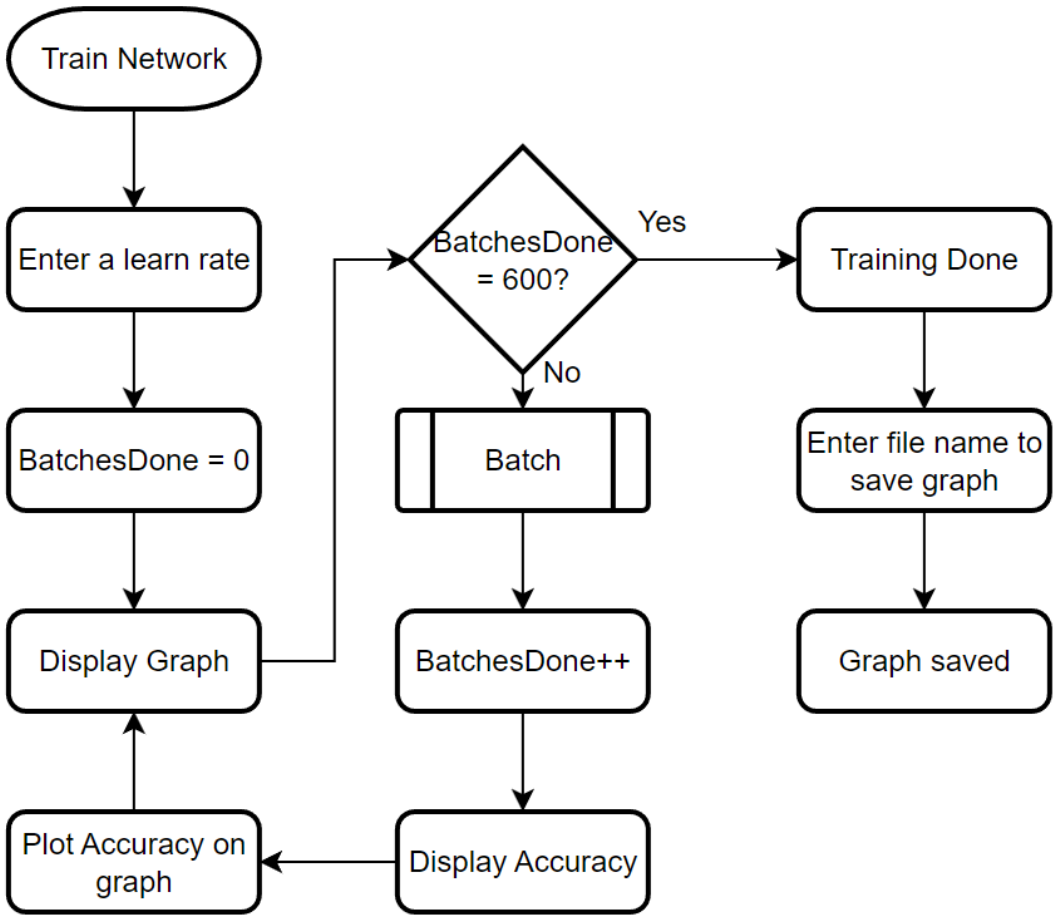


Figure 4 - Training

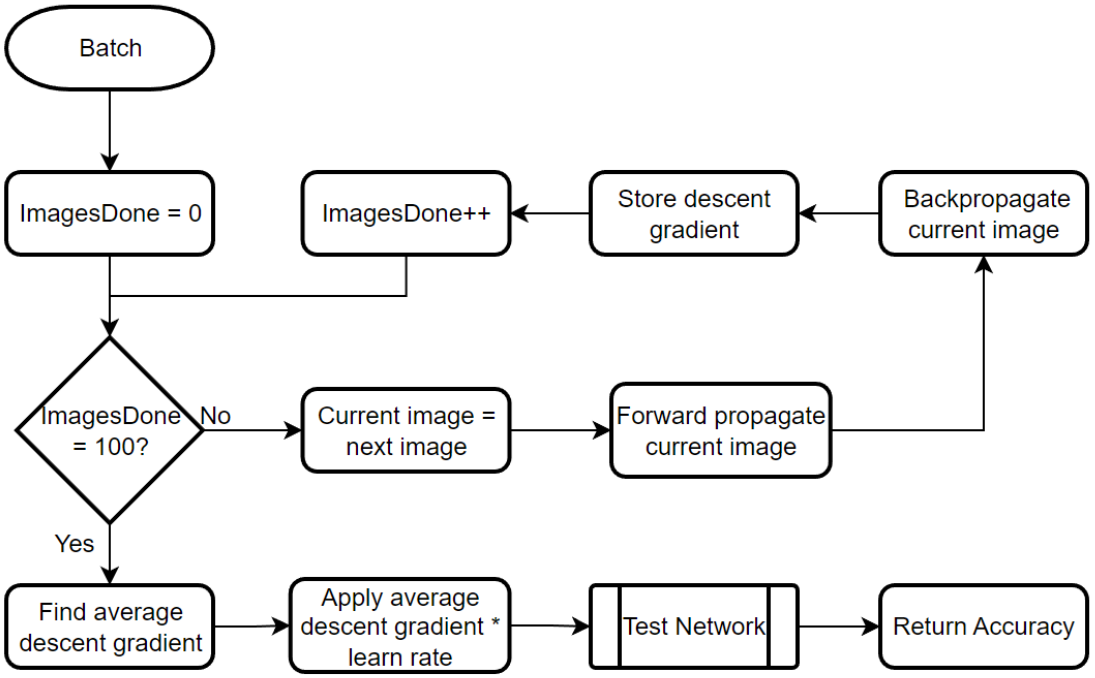


Figure 5 - Batch

[redacted]

Figure 3 and *Figure 4* are more subroutines that the user can select but these are given in more detail. They still display an abstracted version of what is going on but show off some of the loops and decisions the program is making to come up with a result for the user to see. I have chosen to add more detail to these diagrams to represent the fact that running these while the network is in different states will have different effects. For example creating a network and testing it will have a different output to testing it after it has been trained.

Figure 5 is a subroutine used by *Figure 4* which I again created a diagram for to show more detail as to how the program is coming up with certain results.

2.2 Programming Language and Libraries

2.2.1 Programming Language

To program the project I decided to use C# as it is the programming language that I am most familiar with. Due to the time spent learning the language in college I am comfortable and skilled enough to complete my project without needing to spend time learning the language. C# is a high level object oriented programming language which makes it ideal for my project as OOP will be necessary to achieve the objectives I have set.

2.2.2 Libraries

Since my program needs to display graphs to the user I found a graphing library which works with windows forms to display a graph. I used the ScottPlot library as It offered many different graph options which gave me freedom to explore what would be best for my project. The ScottPlot library allows me to easily graph the training sessions that I run my neural networks through and allows me to save an image of the graph afterwards both with only a small amount of code required. This is useful as it allows me to spend time focusing on the more important technical parts of the project but still allows me to implement a useful output for the user.

ScottPlot - https://scottplot.net/

2.3 Algorithm design

2.3.1 Matrix Multiplication

Matrix Multiplication works differently to multiplication with two scalar numbers. There are a few different matrix multiplication algorithms but I have chosen to implement the most intuitive algorithm even though it has complexity $O(n^3)$. When multiplying two matrices they must be conformable in order to get a result. Two matrices are conformable if the first matrix has the same number columns as the number of rows in the second matrix. Because of this matrix multiplication is **not** commutative.
An example of how matrix multiplication works:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix}$$

[redacted]

The result of Matrix multiplication will always have the dimensions of rows of the first matrix x columns of the second matrix. For example the 3x1 matrix multiplied by the 1x3 matrix above results in a 3x3 matrix. Multiplication with more than one column in the first matrix will result in terms being added together in each spot of the matrix.

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1 b_1 + a_2 b_2 \\ a_3 b_1 + a_4 b_2 \end{bmatrix}$$

Overall In Pseudocode the algorithm works as such:

```
Function MatrixMultiplication(Matrix a, Matrix b)
    If (a.columns != b.rows)
        Output Error
    Else
    Matrix result = Matrix(a.rows, b.columns)
    For i = 0 to result.rows
        For j = 0 to result.columns
            For k = 0 to a.columns
                result(i, j) += a(i, k) * b(k, j)
                k = k+1
            End For
            j = j+1
        End For
        i = i+1
    End For
    Return result
End Function
```

2.3.2 Forward Propagation

Forward Propagation is the algorithm that is used in neural networks to propagate data through a network. It allows the network to come to a result which is determined based on the activations of the final layer in the network. The network will have a set number of nodes on the final layer and each node will correspond to a result. My networks are determining digits so there are 10 nodes on the final layer, one corresponding to each digit. The activations of these nodes after forward propagation determine which digit the network has decided. It does this by picking the node with the largest activation value.

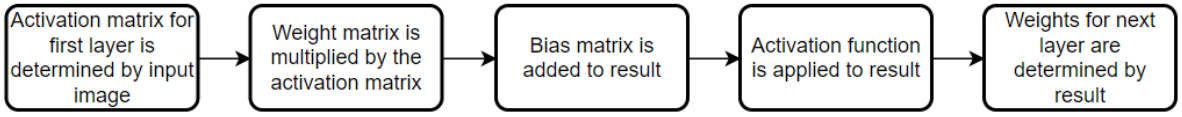
The general idea of forward propagation is that the activations of the first layer are determined by the input data. This is then multiplied by the weight matrix for that layer and the biases for that layer are added on to this result too. The result with the biases added is called the weighted sum. The weighted sum is used to calculate the activations for the next layer. To keep the activations at sensible values an activation function is applied to the weighted sum. The activation function is usually a function that limits activations between a certain range. For my networks the function that is used on the last weighted sum is always sigmoid as this means the activations on the final layer will always be between 0 and 1 making it easy to work out which value is the one determined by the network.

As a maths expression the forward propagation calculation looks like this:

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$

The equivalent expression simplified:
$\sigma(\mathrm{Wa} + \mathrm{b})$

An overview of how forward propagation works:



2.3.3 Backpropagation

The main idea of the Backpropagation algorithm is to find the fastest way to improve a network. Firstly a cost function needs to be established. This cost function is the mean squared error of the actual outputs vs the expected outputs when an image is forward propagated. Since the actual outputs are based on the activations weights and biases of the layers before that we can substitute that into the function in place of the actual output. This gives us a link between cost and the parameters which allows us to use multivariable calculus. Multivariable calculus is used to find lots of partial derivatives and these work out how to nudge each parameter (weight or bias) to have the most impact on improving the cost function.

The partial derivatives are in the form of the cost function with respect to whichever parameter we are looking at. Letting the cost function be represented by C and the parameter be represented by x
This is shown as:

$$\frac{\partial C}{\partial x}$$

Taking the negative of this gradient will tell us the downwards direction and this is how we adjust the parameters in the network. We do this as this gives us the fastest way down which will allow us to settle at a minimum point therefore minimising the cost function. A minimised cost function will mean that the network is achieving results close to the desired output and this therefore means that the network is more accurate.

To calculate the derivative of the cost function with respect to a variable the chain rule must be applied to other connected derivatives to get the whole derivative. The things that need to be found are the weighted sum with respect to the weight, the activation with respect to the weighted sum and the cost with respect to the activation function. Letting C and x mean the same as before and letting z and a be the weighted sum and activation function respectively.

The derivatives we need to find are:

$$\frac{\partial z}{\partial x}, \frac{\partial a}{\partial z}, \frac{\partial C}{\partial a}$$

We can then use these to find the whole derivative we want:

$$\frac{\partial C}{\partial x} = \frac{\partial z}{\partial x} \times \frac{\partial a}{\partial z} \times \frac{\partial C}{\partial a}$$

2.4 User Interface Design

2.4.1 Console Application

Most of the user interaction takes place in the console where the user will navigate a text based menu and enter data. I decided it was best to make a console application as the project will not be used by a lot of people making aesthetics not a concern.

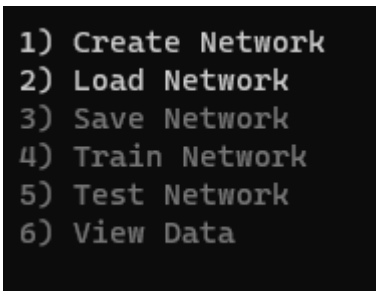


Figure 1 - Menu when program first runs

When the program is first started the user is met with the main menu screen as shown in *Figure 1*. The user has the option to either create a network or load a network. The other options that the program has are still displayed but are greyed out as the user cannot select them. To select an option the user types in the number corresponding to the option they want to select and then presses enter. This clears the console and takes them to the next screen.

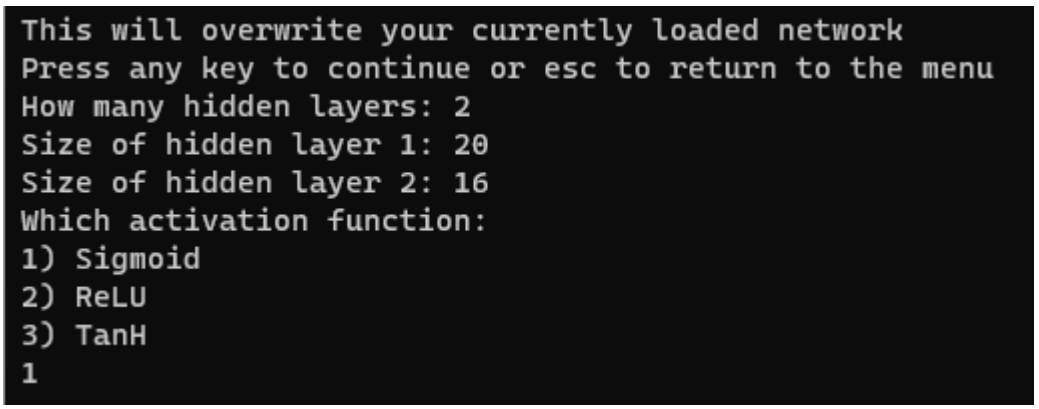


Figure 2 - User creating a network

Figure 2 shows what would happen if the user entered 1 on the screen from *Figure 1*. They are asked to confirm if they want to continue to create a network as it would override the current network, this is asked regardless of if a network is loaded or not. The user is then asked for how many hidden layers and they can input a number. This number determines how many layers they will be asked to choose a size for. They are then asked to decide on an activation function by picking the corresponding number just like how the main menu functions. The user interface does not have any input validation and it is assumed that the users will enter in values that the program can interpret. After this the user is returned to the main menu.

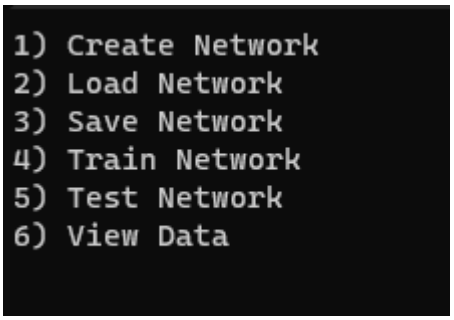


Figure 3 - Menu when network is loaded

When a network is loaded in the other options will become available to the user. Almost all of these other options have user interaction in the same way except for option 4) Train Network which partially uses windows forms to display a graph to the user.

2.4.2 Windows Forms

When option 4) Train Network is selected from the main menu along with the console interaction the user is also presented with a windows form during training which displays a graph. As the network completes batches data is plotted onto the graph so the user can see how the network is improving.

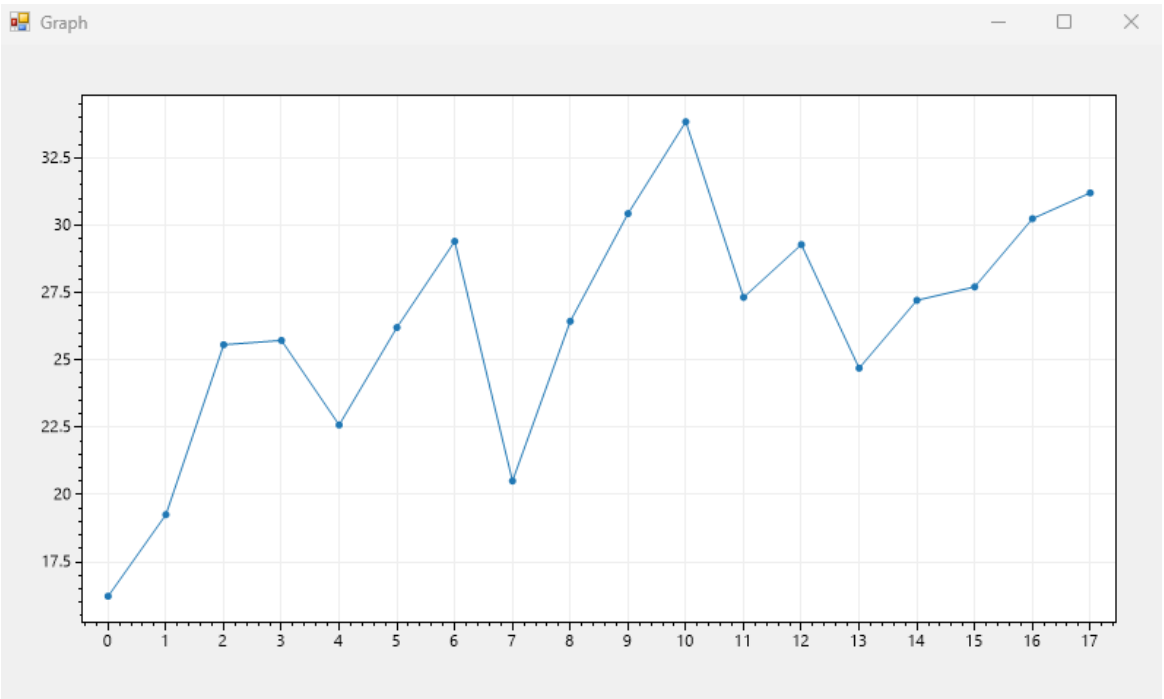


Figure 1 - Windows forms graph

As seen in *Figure 1* the current batch is plotted along the x axis and the current accuracy is plotted on the y axis. *Figure 2* shows the graph with more data points plotted. Once the network has finished its training session the graph will turn into a dialogue box which allows the user to move around and zoom in and out of the graph. When the user closes this window the application will continue running in the console with more interactions for the user.

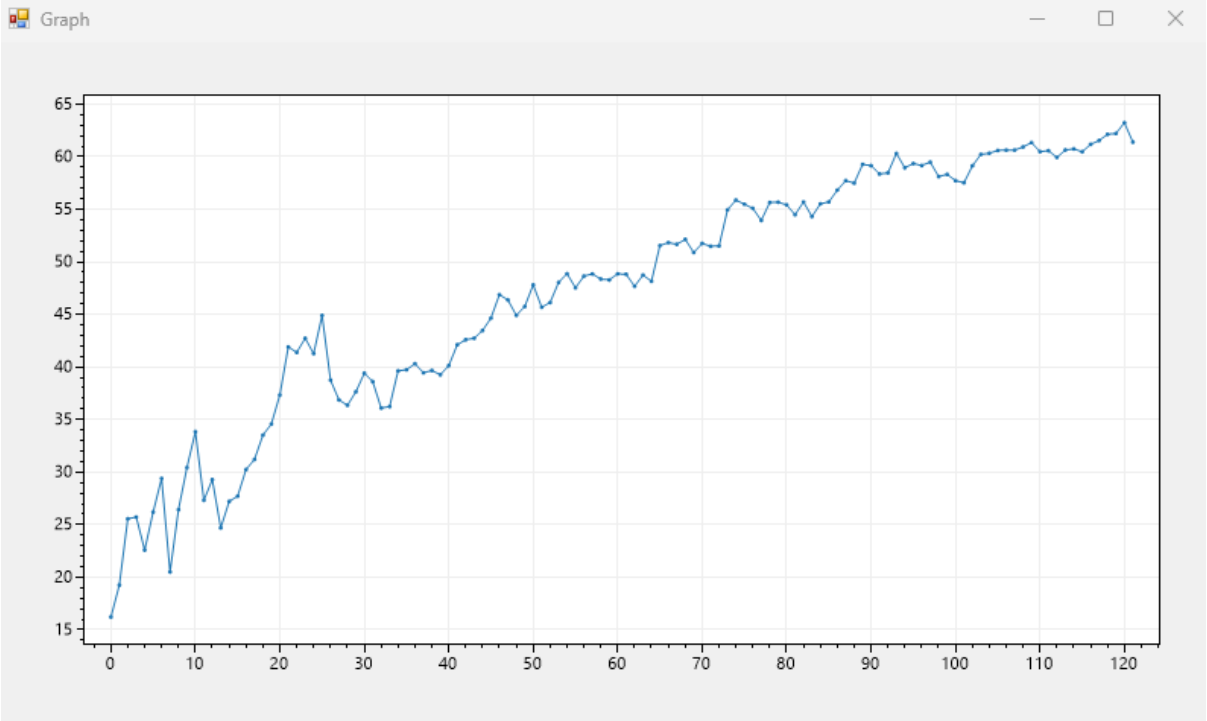
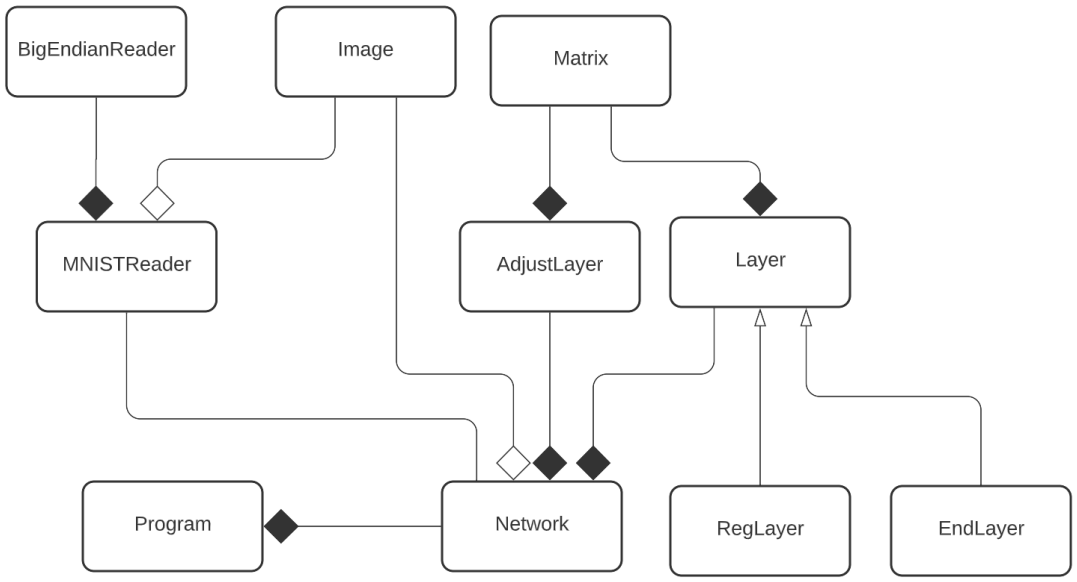


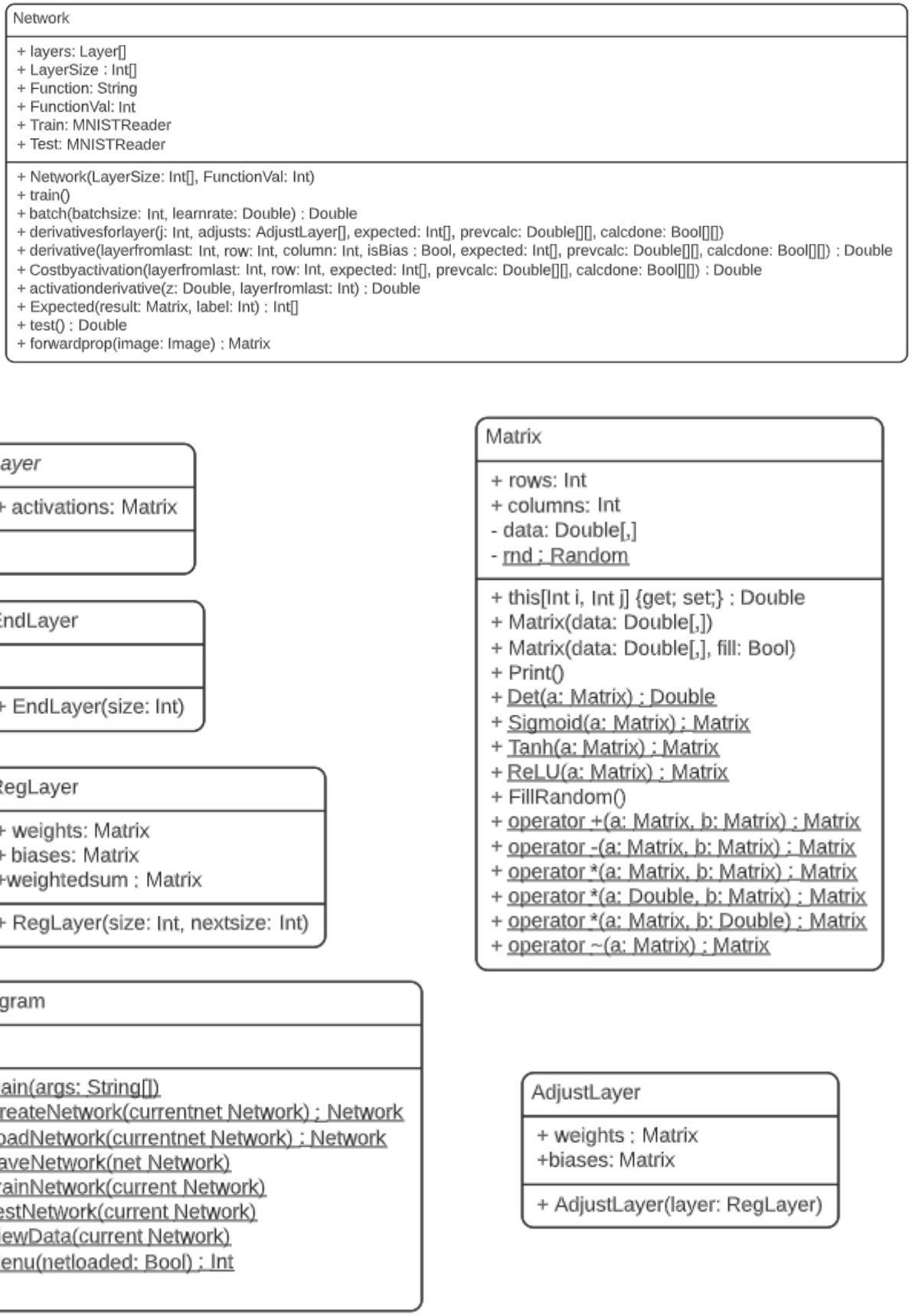
Figure 2 - Windows forms graph with more data

2.5 Object-oriented Design

2.5.1 Overview



2.5.2 Detailed UML Diagrams



2.6 Files and Reading Data

2.6.1 MNIST Data Files

My project uses the MNIST data set to train and test the networks. The data set is stored in four files, two are allocated to training and two are allocated to testing. For each part one of the files contains the image data itself while the other contains the labels to identify the images. They have a specific structure for which I have created a reader to interpret the files. Since the image and label files line up and store related data the reader handles both these files in one go to match up the images with their labels.

The Layout of the files:

Image File			Label File		
Offset	Type	Description	Offset	Type	Description
0000	32 bit integer	Magic number	0000	32 bit integer	Magic number
0004	32 bit integer	Number of images Training - 60000 Test - 10000	0004	32 bit integer	Number of labels Training - 60000 Test - 10000
0008	32 bit integer	Rows (28)	0008	Unsigned byte	Label
0012	32 bit integer	Columns (28)	0009	Unsigned byte	Label
0016	Unsigned byte	Pixel	....	Unsigned byte	Label
0017	Unsigned byte	Pixel	XXXX	Unsigned byte	Label
....	Unsigned byte	Pixel			
XXXX	Unsigned byte	Pixel			

Because the MNIST data is stored in a big endian format and most processors, including mine, use little endian the bytes need to be flipped when reading in the 32 bit integers in order to correctly interpret them.

2.6.2 MNIST Reader and Image class

To read data from the MNIST files I first created a BigEndianReader which inherits from the System.IO class BinaryReader. It implements a subroutine to deal with the big endian data of the MNIST data set by reading integers as sets of bytes, flipping them and converting to an integer.

My MNISTReader class has a composition relationship with two BigEndianReaders which are a part of it. One is to read the label file and the other is to read the image file. The MNISTReader also holds an integer called values which represents how many pieces of data there are and an image size which represents the size of each image in the data (28 pixels in both directions). When an instance of the MNISTReader class is created the BigEndianReaders are opened and the data at the start of the MNIST files is assigned to the variables or discarded as required. From then the MNISTReader can return an Image object using the ReadNext method.

An Image class is used to store the image data and its corresponding label together and is the object returned from a call to ReadNext. It stores the label as an integer and the data as an array of doubles. This is used to neatly move multiple pieces of data from the MNISTReader out into the Network and from there use the data as an input to the network.

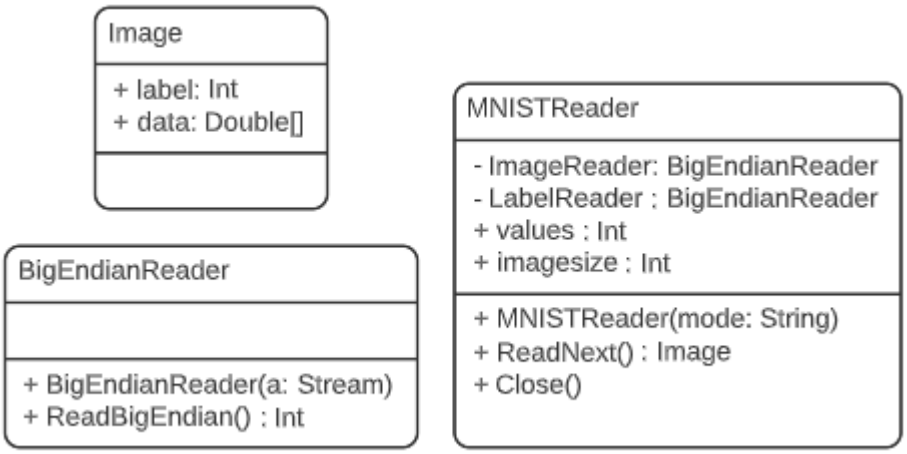


Figure 1 - UML Diagrams

2.6.3 Network File

In order to save and load networks I created my own file structure to hold the necessary information. The program saves information about networks into a binary file such that if the file is read back in the same way it was written the program can exactly reconstruct the network and make it functional.

The layout of my binary file:

<table>
<tr><td colspan="2">Amount</td><td>Type</td><td>Description</td></tr>
<tr><td colspan="2">1</td><td>32 bit integer</td><td>Activation function the network uses</td></tr>
<tr><td colspan="2">1</td><td>32 bit integer</td><td>Number of layers in the network</td></tr>
<tr><td colspan="2">Number of layers in the network</td><td>32 bit integer</td><td>Size of a layer</td></tr>
<tr><td rowspan="7">Number of layers in the network</td><td>1</td><td>32 bit integer</td><td>Number of rows in weights matrix</td></tr>
<tr><td>1</td><td>32 bit integer</td><td>Number of columns in weights matrix</td></tr>
<tr><td>number of rows in weights matrix multiplied by the number of columns in weights matrix</td><td>double</td><td>weight</td></tr>
<tr><td>1</td><td>32 bit integer</td><td>Number of rows in biases matrix</td></tr>
<tr><td>1</td><td>32 bit integer</td><td>Number of columns in biases matrix</td></tr>
<tr><td>number of rows in biases matrix multiplied by the number of columns in biases matrix</td><td>double</td><td>bias</td></tr>
</table>

3 Testing

3.1 Testing Information

Due to the nature of my project lots of calculations and data interpretation is happening behind the scenes that the user never sees. In order to evidence that these parts of the program were working correctly I wrote some extra code to display things. The programs running in the Matrix test video and Image test video use different code than my main program but both interact with the classes written for the main program.

3.2 Testing Table

Testing Videos:	**Video QR Codes:**
Matrix test video: https://youtu.be/QZ23gIz-uVk	
General use video: https://youtu.be/rmiR1oSLZ3I	
Image test video: https://youtu.be/IE5mkPnJgtY	

Objective	Description	Input	Expected Output	Outcome	Evidence
Objective 1 - Matrices					
1.1	Create a square matrix	2D array { {1, 2}, {1, 2} }	$\begin{bmatrix}1 & 2\\1 & 2\end{bmatrix}$	Pass	Matrix test video 0:00
	Create a non square matrix	2D array { {1, 2, 3}, {1, 2, 3} }	$\begin{bmatrix}1 & 2 & 3\\1 & 2 & 3\end{bmatrix}$	Pass	Matrix test video 0:00
	Create a matrix filled randomly	Empty 2D array size 2x2 Random Fill true	Matrix with 2 rows and 2 columns with random data	Pass	Matrix test video 0:00
1.2	Multiply two conformable matrices	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}1 & 2\\5 & 0\end{bmatrix}$	$\begin{bmatrix}11 & 2\\23 & 6\end{bmatrix}$	Pass	Matrix test video 0:04
1.2 **1.10**	Multiply two non conformable matrices	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$	Matrix non conformable exception	Pass	Matrix test video 0:07
1.3	Add two conformable matrices	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}+\begin{bmatrix}4 & 3\\2 & 1\end{bmatrix}$	$\begin{bmatrix}5 & 5\\5 & 5\end{bmatrix}$	Pass	Matrix test video 0:10
	Subtract two conformable matrices	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}-\begin{bmatrix}0 & 1\\2 & 3\end{bmatrix}$	$\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$	Pass	Matrix test video 0:10
1.3 **1.10**	Add two non conformable matrices	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}+\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$	Matrix non conformable exception	Pass	Matrix test video 0:14
	Subtract two non conformable matrices	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}-\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$	Matrix non conformable exception	Pass	Matrix test video 0:14
1.4	Multiply by a scalar	$2\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}2 & 4\\6 & 8\end{bmatrix}$	Pass	Matrix test video 0:17
1.5	Calculate the transpose of a square matrix	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^T$	$\begin{bmatrix}1 & 3\\2 & 4\end{bmatrix}$	Pass	Matrix test video 0:20
	Calculate the transpose of a non square matrix	$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}^T$	$\begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix}$	Pass	Matrix test video 0:20

1.6	Calculate the determinant of a square matrix	$Det\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	-2	**Fail**	Matrix test video 0:23
	Retest Calculate the determinant of a square matrix	$Det\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	-2	Pass	Image test video 0:00
	Calculate the determinant of a non square matrix	$Det\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$	Matrix non square exception	Pass	Matrix test video 0:23
1.7	Apply the sigmoid function	$\sigma\begin{bmatrix}1 & -2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}0.731 & 0.119\\0.953 & 0.982\end{bmatrix}$	Pass	Matrix test video 0:26
1.8	Apply the tanh function	$tanh\begin{bmatrix}1 & -2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}0.762 & -0.964\\0.995 & 0.999\end{bmatrix}$	Pass	Matrix test video 0:29
1.9	Apply the ReLU function	$ReLU\begin{bmatrix}1 & -2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}1 & 0\\3 & 4\end{bmatrix}$	Pass	Matrix test video 0:31
Objective 2 - Creating networks					
2.1	The user is prompted for the number of hidden layer in the network	User selects create network in the menu	User is prompted to enter a number	Pass	General use video 0:05
2.2	The user is prompted for the size of each hidden layer	User has entered a number of hidden layers	User is prompted to enter a number for each hidden layer	Pass	General use video 0:06
2.3	The user is prompted to choose an activation function	User has entered a number for each hidden layer	User is prompted to select an activation function	Pass	General use video 0:09
2.1 **2.2** **2.3**	The created network has the parameters specified by the user	User has created a network and choses to view the network data	The information the user inputted to create the network is displayed back	Pass	General use video 0:16

Objective 3 - Using MNIST data					
3.1	Reading in training images	Training images file	Image of a number	Pass	Image test video 0:05
3.2	Reading in testing images	Testing images file	Image of a number	Pass	Image test video 0:27
3.3	The program correctly interprets the image data	Training and testing image files	Image of a number	Pass	Image test video 0:05
3.4	The program can read the image label	Training and testing label files	Number which matches the number in the image	Pass	Image test video 0:05
Objective 4 - Storing/accessing created networks					
4.1	The user can save a network to a file	User selects save network in the menu	A file is created with the networks data	Pass	General use video 0:22
4.2	The user is prompted for a file name to save the network as	User selects save network in the menu	User is prompted to choose a file name	Pass	General use video 0:22
4.3	The program creates a binary file containing the network	User has chosen a file name	A file is created with the specified name is created and contains the networks data	Pass	General use video 0:32
4.4	The user can load a network from a file	User selects load network in the menu	A network is loaded into the program	Pass	General use video 0:58
4.5	The user is prompted for a file name	User selects load network in the menu	User is prompted to enter a file name	Pass	General use video 1:00
4.6	The loaded network is the same network which has been saved to the file	User has loaded a network and choses to view the network data	The information displayed to user is the same information as the saved network	Pass	General use video 1:05

Objective 5 - Training					
5.1	The user can train a network	User selects train network in the menu	The program runs a training session	Pass	General use video 1:17
5.2	The user is prompted to pick a learn rate for the session	User selects train network in the menu	User is prompted to choose a learn rate	Pass	General use video 1:18
5.5	The user can run a training session which will increase the networks accuracy	User tests the network before training	User tests the network after training and the accuracy has increased	Pass	General use video 1:22
Objective 6 - Testing					
6.1	The user can test a network	User selects train network in the menu	The program runs a testing session	Pass	General use video 1:13
6.5	The program displays the percentage of a tests accuracy	User selects train network in the menu	The program outputs an accuracy result for the test	Pass	General use video 1:14
Objective 7 - Presenting and logging data					
7.1	The user can view data about a loaded network	User choses view data in the menu	The correct data about a current network is displayed	Pass	General use video 0:16
7.2	The program displays a graph during the training session	User selects train network in the menu	A graph is displayed tracking the accuracy percentages which are displayed in the console	Pass	General use video 1:22
7.3	The program prompts the user to save the graph at the end of the training session	Training session finishes	User is prompted to enter a filename to save the graph as	Pass	General use video 10:25

3.3 Failed Tests

When testing objective 1.6 the program did not output the determinant of the matrix as expected. The result came out to be the negative of what it should have been as I had mixed up the positive and negative parts in the calculation. This bug was simple to fix by switching the signs in the code.

4 Evaluation

4.1 Overall Effectiveness

My system is designed to help investigate the effect of a neural network's architecture on its accuracy. It allows for the creation of networks with architecture that is defined by the user and this is a crucial aspect for the investigation. The system trains and tests networks which shows the change in accuracy over time. It plots a graph during training which gives the user a visual representation of how the accuracy is increasing. All of these aspects are useful for the investigation and help the user to come to an understanding as to how the architecture affects accuracy.

I think that my system overall is effective for the investigation given the scope of the project however there are improvements that could be made to make my solution more useful as a tool to answer the proposed investigation question.

4.2 Evaluation of objectives

1. **Matrices**

   As a whole my objectives for implementing matrices were achieved. Both the determinant and the transpose methods ended up not being used for my project but I originally added them as I thought they may have been needed. Regardless of this they make additions which may need them in the future relatively easy and it also means the whole class could be used in other contexts where those methods might be necessary. The program successfully implements dynamic matrices which can be square and non square and allows for all the operations to take place between them provided that the matrices are conformable. To combat any issues where matrices may be non-conformable, exceptions are implemented to tell the user if a calculation cannot be performed. Since the program handles everything to do with matrices and the user does not directly create them the exceptions never occur however their implementation was important to allow me to test the code sufficiently.

2. **Creating new networks**

   The program successfully allows users to create a network asking them for the number of hidden layers, the number of nodes in each of those layers and the activation function for the network to use. There is no input validation as a part of this process which could cause errors. Adding this could improve the robustness of the program however this wasn't my priority due to the nature of the project being so specific and the fact that a user would likely know what were sensible values due to this. The user can choose between three activation functions however adding more would expand the opportunity to investigate other functions and may make the program more useful.

3. **Using MNIST data**

   Overall MNIST data is used successfully to both test and train networks. The files are read from correctly and data is passed into the network to use. The network can come to a conclusion as to what it thinks the data is and compare whether it is right or wrong. It may be helpful for the program to have some kind of option to display the data to the user so they can look through the data set.

4. **Storing/Accessing created networks**

   My objectives for storing and accessing created networks have been achieved. The program uses binary files to store the data about a network and can use the file to load the network for continued use. To load a network the program creates a new network using parameters stored in the file and then copies the values from the file into the network. This creates a network identical to the one saved to the file. The user is only able to load one network at a time and this is a limitation of the program. It may have been beneficial to be able to load multiple networks at a time and select between them. This would allow for multiple training sessions to happen at once and overall allow data to be gathered faster.

5. **Training**

   When training networks the program works as intended and the accuracy is usually overall increased by the end of the session. Progress in training is sometimes slow and accuracy may not increase between every batch due to the nature of the backpropagation algorithm. Being able to train multiple networks at the same time would definitely be a useful feature to speed up how fast data can be collected. Options like exiting during the middle of training or pausing a session would also make the program more flexible.

6. **Testing**

   The objectives set for testing are met well. The program allows the user to test networks using the testing set of data and it returns the overall accuracy. A visual element during testing could be useful to see which images the network fails to identify correctly.

7. **Presenting and logging data**

   Although the program does log data and present it to the user I feel there are a lot of ways this could be improved to make the program better. It only presents basic data about training sessions of a network and this limits the usefulness for the investigation. During training the user is displayed a graph which plots accuracy against the training batch number. This is somewhat useful but it can be hard to compare how different networks perform. Keeping an overall graph plotting how accuracy improves over different training sessions would be useful to display a bigger picture of what is happening. An option to plot multiple network's sessions onto one graph would also show how training differs between them. The program does successfully display information about a network such as the layers and activation function being used.

4.3 Feedback

To gain more insight into what could be improved with my project I showed the program to my expert mentioned in the analysis section and asked them some questions to help them give me meaningful feedback.

4.3.1 Questions

1. **To what extent do you think the program has met the objectives?**

   I feel that the program has met the majority of the objectives well. The main features that the program needs are implemented and working correctly. Objective 7, presenting and logging data, is probably the objective that needs the most improvement to it as the data presented by the program is only of limited use. Objective 1, matrices, is well implemented and is the most clearly met objective as the whole program's functionality relies on this implementation.

2. **How useful do you think the program is in the context of the investigation question?**

   The program is useful in the context of the investigation but only to a certain extent. The most obvious flaws in the system are that it is slow to collect data and there is not really any way to compare different networks. This does limit how useful it is to help answer the investigation question but the program is overall still a useful tool.

3. **What features do you think could be added to improve the program?**

   To improve the program it would be useful to present more data about the networks training and the ability to compare networks with each other. Adding a feature to collect lots of data by allowing different networks to be automatically created, trained and compared with each other with differences specified by the user could help to answer the investigation question more thoroughly.

4.3.2 Evaluating Feedback

From the feedback given I can see that although my program has met the objectives set It struggles in aspects that help to answer my question set at the beginning of the project. I think the observation that the program is not the best at comparing networks is a valid criticism and It is helpful to see that that is the area where the system needs improvement. The idea for some kind of automatic training Is one that I think would have very large benefits to the program as a whole and would help it to be better suited for the investigation.

4.4 System Improvements

After evaluating the objectives and receiving feedback I have decided on some things that I think would be good additions to improve the system. The things that I find to be the most important are an auto training feature as suggested by my expert and more in depth graphing and comparison. There are also a handful of usability improvements that may be beneficial to make which I have added here too.

4.4.1 Auto Training

For the auto training feature I think it would be best implemented by allowing the user to pick one varying trait for a set of networks and then running training on all of them. Plotting the different networks simultaneously on one graph would be a good way to show which networks are doing the best and would therefore make it quite obvious as to what the best performing network is from a selected batch.

An example of how this would function is the user may pick to test the size of a layer. They could choose to have one hidden layer on all their networks and then change the size of this layer as their variable trait to test. They could set minimum and maximum values for example testing 10 networks by specifying for the program to test networks with a hidden layer of sizes ranging from 10 to 20. The program would then create and train each of these networks and plot the progress of all 10 on one graph for comparison.

4.4.2 Graphing and comparing

To improve the graphing and comparison of networks I think it is first important to be able to load at least two networks at one time and be able to select between them. Ideally holding a list of all the networks loaded would be good so the user can have many loaded at once. A simple menu to select between the network you are currently working with and the other loaded ones would be a good addition. To ensure this list is meaningful, networks could be given names by the user or have default names that list their characteristics.

From this ability to have multiple networks loaded there could then be a comparison option to compare two or possibly more networks together and display their stats. Some extra information could be saved about the number of training sessions they've had, the accuracy on their last test and the data about their last training session. A graph could be plotted with all the networks last training sessions to compare them together. Most of the system would have to be slightly adapted to allow for multiple networks to be loaded.

4.4.3 Usability Improvements

To make the system more user friendly a GUI rather than a console window would be a good addition. This wasn't ever something that I had planned for the project however if it is to be used in the future I think it would be greatly beneficial to have this. Another usability improvement would be a built in way to view the data set. During my testing I wrote some code to turn data from the data set into actual images that could be viewed however this was not a feature in the final solution. Being able to look at the data would be good for a user who is not familiar with the data set. This could also allow for an option to look at which specific images a network failed to identify as a small amount of the images in the data set are difficult for even a human to classify correctly.

5 Technical Solution

5.1 Contents page for code

5.1.1 CS Files

5.1.2 Testing Code

5.2.3 Key Algorithms and Data structures

5.2 Code

5.2.1 Exception.cs

```
1  using System;
2
3  namespace Program
4  {
5      class MatrixNonConformableException : Exception
6      {
7          public MatrixNonConformableException(string operation) :
   base($"{operation} cannot be performed on these matrices")
8          {
9
10         }
11     }
12     class MatrixNonSquareException : Exception
13     {
14         public MatrixNonSquareException(string operation) :
   base($"{operation} cannot be performed on this matrix")
15         {
16
17         }
18     }
19 }
```

5.2.2 Image.cs

```
1  namespace Program
2  {
3      internal class Image
4      {
5          public int label;
6          public double[] data;
7      }
8  }
```

5.2.3 Graph.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Windows.Forms;
4
5  namespace Program
6  {
7      public partial class Graph : Form
8      {
9          public Graph()
10         {
11             InitializeComponent();
12         }
13
14         private void formsPlot1_Load(object sender, EventArgs e)
15         {
16
17         }
18
19         public void updategraph(List<double> data)
20         {
21             double[] points = data.ToArray();
22             formsPlot1.Plot.Clear();
23             formsPlot1.Plot.AddSignal(points);
24             formsPlot1.Refresh();
25         }
26         public void savegraph(string filename)
27         {
28             formsPlot1.Plot.SaveFig($"{filename}.png");
29         }
30     }
31 }
```

5.2.4 Matrix.cs

```
1   using System;
2
3   namespace Program
4   {
5       class Matrix
6       {
7           public int rows;
8           public int columns;
9           double[,] data;
10          static Random rnd = new Random();
11
12          public Matrix(double[,] data)
13          {
14              rows = data.GetLength(0);
15              columns = data.GetLength(1);
16              this.data = data;
17          }
18          public Matrix(double[,] data, bool fill)
19          {
20              rows = data.GetLength(0);
21              columns = data.GetLength(1);
22              this.data = data;
23              if (fill)
24              {
25                  this.FillRandom();
26              }
27          }
28
29          public void Print()
30          {
31              for (int i = 0; i < rows; i++)
32              {
33                  for (int j = 0; j < columns; j++)
34                  {
35                      Console.Write($"{this[i, j]} ");
36                  }
37                  Console.WriteLine();
38              }
39              Console.WriteLine();
40          }
41
42          public double this[int i, int j]
43          {
44              get { return data[i, j]; }
45              set { data[i, j] = value; }
```

```
46          }
47
48          public static Matrix operator +(Matrix a, Matrix b)
49          {
50              if (a.rows != b.rows || a.columns != b.columns)
51              {
52                  throw new MatrixNonConformableException("Addition");
53              }
54              double[,] result = new double[a.rows, a.columns];
55              for (int i = 0; i < result.GetLength(0); i++)
56              {
57                  for (int j = 0; j < result.GetLength(1); j++)
58                  {
59                      result[i, j] = a[i, j] + b[i, j];
60                  }
61              }
62              return new Matrix(result);
63          }
64          public static Matrix operator -(Matrix a, Matrix b)
65          {
66              if (a.rows != b.rows || a.columns != b.columns)
67              {
68                  throw new MatrixNonConformableException("Subtraction");
69              }
70              double[,] result = new double[a.rows, a.columns];
71              for (int i = 0; i < result.GetLength(0); i++)
72              {
73                  for (int j = 0; j < result.GetLength(1); j++)
74                  {
75                      result[i, j] = a[i, j] - b[i, j];
76                  }
77              }
78              return new Matrix(result);
79          }
80          public static Matrix operator *(Matrix a, Matrix b)
81          {
82              if (a.columns != b.rows)
83              {
84                  throw new
    MatrixNonConformableException("Multiplication");
85              }
86              double[,] result = new double[a.rows, b.columns];
87              for (int i = 0; i < result.GetLength(0); i++)
88              {
89                  for (int j = 0; j < result.GetLength(1); j++)
90                  {
91                      for (int k = 0; k < a.columns; k++)
92                      {
```

```
93                          result[i, j] += a[i, k] * b[k, j];
94                      }
95                  }
96              }
97              return new Matrix(result);
98          }
99          public static Matrix operator *(double a, Matrix b)
100         {
101             for (int i = 0; i < b.rows; i++)
102             {
103                 for (int j = 0; j < b.columns; j++)
104                 {
105                     b[i, j] = a * b[i, j];
106                 }
107             }
108             return b;
109         }
110         public static Matrix operator *(Matrix b, double a)
111         {
112             for (int i = 0; i < b.rows; i++)
113             {
114                 for (int j = 0; j < b.columns; j++)
115                 {
116                     b[i, j] = a * b[i, j];
117                 }
118             }
119             return b;
120         }
121         public static Matrix operator ~(Matrix a)
122         {
123             double[,] result = new double[a.columns, a.rows];
124             for (int i = 0; i < result.GetLength(1); i++)
125             {
126                 for (int j = 0; j < result.GetLength(0); j++)
127                 {
128                     result[j, i] = a[i, j];
129                 }
130             }
131             return new Matrix(result);
132         }
133
134         public static double Det(Matrix a)
135         {
136             if (a.columns != a.rows)
137             {
138                 throw new MatrixNonSquareException("Determinant");
139             }
140             if (a.rows == 1)
```

```
            {
                return a[0, 0];
            }
            else
            {
                double det = 0;
                for (int i = 0; i < a.rows; i++)
                {
                    double[,] b = new double[a.rows - 1, a.columns - 1];
                    int count = 0;
                    for (int j = 0; j < a.rows; j++)
                    {
                        if (i != j)
                        {
                            for (int k = 1; k < a.rows; k++)
                            {
                                b[count, k - 1] = a[j, k];
                            }
                            count++;
                        }
                    }
                    if (i % 2 == 1)
                    {
                        det += -a[i, 0] * Det(new Matrix(b));
                    }
                    else
                    {
                        det += +a[i, 0] * Det(new Matrix(b));

                    }
                }
                return det;
            }
        }

        public static Matrix Sigmoid(Matrix a)
        {
            for (int i = 0; i < a.rows; i++)
            {
                for (int j = 0; j < a.columns; j++)
                {
                    a[i, j] = 1 / (1 + Math.Exp(-a[i, j]));
                }
            }
            return a;
        }
        public static Matrix Tanh(Matrix a)
        {
```

```
189             for (int i = 0; i < a.rows; i++)
190             {
191                 for (int j = 0; j < a.columns; j++)
192                 {
193                     a[i, j] = Math.Tanh(a[i, j]);
194                 }
195             }
196             return a;
197         }
198         public static Matrix ReLU(Matrix a)
199         {
200             for (int i = 0; i < a.rows; i++)
201             {
202                 for (int j = 0; j < a.columns; j++)
203                 {
204                     a[i, j] = Math.Max(0, a[i, j]);
205                 }
206             }
207             return a;
208         }
209 
210         public void FillRandom()
211         {
212             for (int i = 0; i < this.rows; i++)
213             {
214                 for (int j = 0; j < this.columns; j++)
215                 {
216                     this[i, j] = (2 * rnd.NextDouble()) - 1;
217                 }
218             }
219         }
220     }
221 }
```

5.2.5 MNISTReader.cs

```
1   using System;
2   using System.IO;
3
4   namespace Program
5   {
6       internal class MNISTReader
7       {
8           BigEndianReader ImageReader;
9           BigEndianReader LabelReader;
10          public int values;
11          public int imagesize;
12
13          public MNISTReader(string mode)
14          {
15              ImageReader = new
    BigEndianReader(File.Open($"{mode}-images", FileMode.Open));
16              ImageReader.ReadBigEndian();
17              values = ImageReader.ReadBigEndian();
18              imagesize = ImageReader.ReadBigEndian();
19              ImageReader.ReadBigEndian();
20
21              LabelReader = new
    BigEndianReader(File.Open($"{mode}-labels", FileMode.Open));
22              LabelReader.ReadBigEndian();
23              LabelReader.ReadBigEndian();
24          }
25          public Image ReadNext()
26          {
27              Image currentimage = new Image();
28              currentimage.label = LabelReader.ReadByte();
29              currentimage.data = new double[imagesize * imagesize];
30              for (int i = 0; i < currentimage.data.Length; i++)
31              {
32                  currentimage.data[i] = ImageReader.ReadByte();
33              }
34              return currentimage;
35          }
36          public void Close()
37          {
38              ImageReader.Close();
39              LabelReader.Close();
40          }
41      }
42      internal class BigEndianReader : BinaryReader
```

```
    {
        public BigEndianReader(Stream a) : base(a) { }
        public int ReadBigEndian()
        {
            byte[] data = this.ReadBytes(4);
            byte[] num = new byte[4];
            for (int i = 0; i < 4; i++)
            {
                num[i] = data[3 - i];
            }
            return BitConverter.ToInt32(num, 0);
        }
    }
}
```

5.2.6 Network.cs

```
using System;
using System.Collections.Generic;

namespace Program
{
    internal class Network
    {
        public Layer[] layers;
        public int[] LayerSize;
        public string Function;
        public int FunctionVal;
        public MNISTReader Train;
        public MNISTReader Test;

        public Network(int[] LayerSize, int FunctionVal)
        {
            Train = new MNISTReader("train");
            Test = new MNISTReader("test");
            this.FunctionVal = FunctionVal;
            string[] Functions = { "Sigmoid", "ReLU", "TanH" };
            Function = Functions[FunctionVal - 1];

            this.LayerSize = LayerSize;
            layers = new Layer[this.LayerSize.Length];
            for (int i = 0; i < layers.Length; i++)
            {
                if (i == layers.Length - 1)
                {
```

```
29                      layers[i] = new EndLayer(this.LayerSize[i]);
30                  }
31                  else
32                  {
33                      layers[i] = new RegLayer(this.LayerSize[i],
this.LayerSize[i + 1]);
34                  }
35              }
36          }
37          public void train()
38          {
39              Console.Write("Enter a learn rate for this training session:
");
40              double learnrate = double.Parse(Console.ReadLine());
41              const int batchsize = 100;
42              Graph graph = new Graph();
43              List<double> datapoints = new List<double>();
44              graph.Show();
45              graph.BringToFront();
46              for (int i = 0; i < 600; i++)
                {
47                  double output = batch(batchsize, learnrate);
48                  datapoints.Add(output);
49                  graph.updategraph(datapoints);
50                  Console.WriteLine($"{i}: {output}% accurate");
51              }
52              Train.Close();
53              Train = new MNISTReader("train");
54              Console.WriteLine("Training Complete");
55              graph.Hide();
56              graph.ShowDialog();
57              Console.Write("Enter a file name to save the graph (leave
blank to skip):");
58              string filename = Console.ReadLine();
59              if (filename != "")
60              {
61                  graph.savegraph(filename);
62              }
63          }
64          public double batch(int batchsize, double learnrate)
65          {
66              AdjustLayer[] adjusts = new AdjustLayer[layers.Length - 1];
67              for (int i = 0; i < adjusts.Length; i++)
68              {
69                  adjusts[i] = new AdjustLayer((RegLayer)layers[i]);
70              }
71              for (int i = 0; i < batchsize; i++)
72              {
```

[redacted]

```
73                  Image currentimage = Train.ReadNext();
74                  int[] expected = Expected(forwardprop(currentimage),
    currentimage.label);
75                  double[][] prevcalc = new double[adjusts.Length][];
76                  bool[][] calcdone = new bool[adjusts.Length][];
77                  for (int j = 0; j < adjusts.Length; j++)
78                  {
79                      prevcalc[j] = new double[adjusts[j].weights.rows];
80                      calcdone[j] = new bool[adjusts[j].weights.rows];
81                  }
82                  for (int j = adjusts.Length - 1; j >= 0; j--)
83                  {
84                      derivativesforlayer(j, adjusts, expected, prevcalc,
    calcdone);
85                  }
86              }
87              for (int i = 0; i < layers.Length - 1; i++)
88              {
89                  adjusts[i].weights *= (-learnrate) / batchsize;
90                  adjusts[i].biases *= (-learnrate) / batchsize;
91                  ((RegLayer)layers[i]).weights += adjusts[i].weights;
92                  ((RegLayer)layers[i]).biases += adjusts[i].biases;
93              }
94              return test() * 100;
95          }
96          public void derivativesforlayer(int j, AdjustLayer[] adjusts,
    int[] expected, double[][] prevcalc, bool[][] calcdone)
97          {
98              for (int k = 0; k < adjusts[j].weights.rows; k++)
99              {
100                 for (int l = 0; l < adjusts[j].weights.columns; l++)
101                 {
102                     adjusts[j].weights[k, l] +=
    derivative(adjusts.Length - j, k, l, false, expected, prevcalc,
    calcdone);
103                 }
104                 adjusts[j].biases[k, 0] += derivative(adjusts.Length -
    j, k, 0, true, expected, prevcalc, calcdone);
105             }
106         }
107         public double derivative(int layerfromlast, int row, int column,
    bool isbias, int[] expected, double[][] prevcalc, bool[][] calcdone)
108         {
109             double wholederivative = 0;
110             double sumbyvalue = 0;
111             double activationbysum = 0;
112             double costbyactivation = 0;
113             if (isbias)
```

```
114             {
115                 sumbyvalue = 1;
116             }
117             else
118             {
119                 sumbyvalue = (layers[layers.Length - 1 -
    layerfromlast]).activations[column, 0];
120             }
121             activationbysum =
    activationderivative(((RegLayer)layers[layers.Length - 1 -
    layerfromlast]).weightedsum[row, 0], layerfromlast);
122 
123             costbyactivation = Costbyactivation(layerfromlast, row,
    expected, prevcalc, calcdone);
124 
125             wholederivative = sumbyvalue * activationbysum *
    costbyactivation;
126             return wholederivative;
127         }
128         public double Costbyactivation(int layerfromlast, int row, int[]
    expected, double[][] prevcalc, bool[][] calcdone)
129         {
130             if (calcdone[layers.Length - 1 - layerfromlast][row])
131             {
132                 return prevcalc[layers.Length - 1 - layerfromlast][row];
133             }
134             if (layerfromlast == 1)
135             {
136                 prevcalc[layers.Length - 1 - layerfromlast][row] = 2 *
    ((layers[layers.Length - 1]).activations[row, 0] - expected[row]);
137                 calcdone[layers.Length - 1 - layerfromlast][row] = true;
138                 return prevcalc[layers.Length - 1 - layerfromlast][row];
139             }
140             else
141             {
142                 double total = 0;
143                 for (int j = 0; j < layers[layers.Length - layerfromlast
    + 1].activations.rows; j++)
144                 {
145                     total += ((RegLayer)layers[layers.Length - 1 -
    layerfromlast]).weights[j, row] *
    activationderivative(((RegLayer)layers[layers.Length - 1 -
    layerfromlast]).weightedsum[j, 0], layerfromlast) *
    Costbyactivation(layerfromlast - 1, j, expected, prevcalc, calcdone);
146                 }
147                 prevcalc[layers.Length - 1 - layerfromlast][row] =
    total;
148                 calcdone[layers.Length - 1 - layerfromlast][row] = true;
```

```
                return total;
            }
        }
        public double activationderivative(double z, int layerfromlast)
        {
            switch (FunctionVal)
            {
                case 1:
                    //sigmoid
                    return (1 / (1 + Math.Exp(-z))) * (1 - (1 / (1 +
Math.Exp(-z))));
                case 2:
                    //relu
                    if (layerfromlast == 1)
                    {
                        return (1 / (1 + Math.Exp(-z))) * (1 - (1 / (1 +
Math.Exp(-z))));
                    }
                    else
                    {
                        if (z > 0)
                        {
                            return 1;
                        }
                        else
                        {
                            return 0;
                        }
                    }
                case 3:
                    //tanh
                    if (layerfromlast == 1)
                    {
                        return (1 / (1 + Math.Exp(-z))) * (1 - (1 / (1 +
Math.Exp(-z))));
                    }
                    else
                    {
                        return 1 - (Math.Tanh(z) * Math.Tanh(z));
                    }
                default:
                    return 0;
            }

        }
        public int[] Expected(Matrix result, int label)
        {
            int[] expected = new int[result.rows];
```

```
            for (int i = 0; i < expected.Length; i++)
            {
                if (i == label)
                {
                    expected[i] = 1;
                }
                else
                {
                    expected[i] = 0;
                }
            }
            return expected;
        }
        public double test()
        {
            double correct = 0;
            Image currentimage;
            for (int i = 0; i < Test.values; i++)
            {
                currentimage = Test.ReadNext();
                Matrix result = forwardprop(currentimage);
                double largest = result[0, 0];
                int index = 0;
                for (int j = 1; j < result.rows; j++)
                {
                    if (result[j, 0] > largest)
                    {
                        largest = result[j, 0];
                        index = j;
                    }
                }
                if (index == currentimage.label)
                {
                    correct++;
                }
            }
            Test.Close();
            Test = new MNISTReader("test");
            return correct / Test.values;
        }
        public Matrix forwardprop(Image image)
        {
            for (int j = 0; j < layers[0].activations.rows; j++)
            {
                layers[0].activations[j, 0] = image.data[j];
            }
            for (int j = 0; j < layers.Length - 1; j++)
            {
```

```
242                 Matrix weights = ((RegLayer)layers[j]).weights;
243                 Matrix activations = ((RegLayer)layers[j]).activations;
244                 Matrix biases = ((RegLayer)layers[j]).biases;
245                 ((RegLayer)layers[j]).weightedsum = (weights *
    activations) + biases;
246                 switch (FunctionVal)
247                 {
248                     case 1:
249                         layers[j + 1].activations =
    Matrix.Sigmoid(((RegLayer)layers[j]).weightedsum);
250                         break;
251                     case 2:
252                         if (j == layers.Length - 2)
253                         {
254                             layers[j + 1].activations =
    Matrix.Sigmoid(((RegLayer)layers[j]).weightedsum);
255                         }
256                         else
257                         {
258                             layers[j + 1].activations =
    Matrix.ReLU(((RegLayer)layers[j]).weightedsum);
259                         }
260                         break;
261                     case 3:
262                         layers[j + 1].activations =
    Matrix.Tanh(((RegLayer)layers[j]).weightedsum);
263                         break;
264                 }
265             }
266             return layers[layers.Length - 1].activations;
267         }
268 
269     }
270     abstract class Layer
271     {
272         public Matrix activations;
273     }
274     class RegLayer : Layer
275     {
276         public Matrix weights;
277         public Matrix biases;
278         public Matrix weightedsum;
279         public RegLayer(int size, int nextsize)
280         {
281             weights = new Matrix(new double[nextsize, size], true);
282             biases = new Matrix(new double[nextsize, 1]);
283             weightedsum = new Matrix(new double[size, 1]);
284             activations = new Matrix(new double[size, 1]);
```

```
285         }
286     }
287     class EndLayer : Layer
288     {
289         public EndLayer(int size)
290         {
291             activations = new Matrix(new double[size, 1]);
292         }
293     }
294     class AdjustLayer
295     {
296         public Matrix weights;
297         public Matrix biases;
298         public AdjustLayer(RegLayer layer)
299         {
300             weights = new Matrix(new double[layer.weights.rows,
    layer.weights.columns]);
301             biases = new Matrix(new double[layer.biases.rows,
    layer.biases.columns]);
302         }
303 
304     }
305 }
```

5.2.7 Program.cs

```
1   using System;
2   using System.IO;
3
4   namespace Program
5   {
6       class Program
7       {
8           static void Main(string[] args)
9           {
10              Network current = null;
11              while (true)
12              {
13                  bool netloaded = false;
14                  if (current != null)
15                  {
16                      netloaded = true;
17                  }
18                  int choice = Menu(netloaded);
19                  switch (choice)
20                  {
21                      case 1:
22                          current = CreateNetwork(current);
23                          break;
24                      case 2:
25                          current = LoadNetwork(current);
26                          break;
27                      case 3:
28                          SaveNetwork(current);
29                          break;
30                      case 4:
31                          TrainNetwork(current);
32                          break;
33                      case 5:
34                          TestNetwork(current);
35                          break;
36                      case 6:
37                          ViewData(current);
38                          break;
39                      default:
40                          break;
41                  }
42              }
43
44          }
```

```
45          static Network CreateNetwork(Network currentnet)
46          {
47              Console.Clear();
48              Console.WriteLine("This will overwrite your currently loaded
    network");
49              Console.WriteLine("Press any key to continue or esc to
    return to the menu");
50              if (Console.ReadKey(true).Key == ConsoleKey.Escape)
51              {
52                  return currentnet;
53              }
54              if (currentnet != null)
55              {
56                  currentnet.Train.Close();
57                  currentnet.Test.Close();
58              }
59              Console.Write("How many hidden layers: ");
60              int layers = int.Parse(Console.ReadLine());
61              int[] LayerSize = new int[layers + 2];
62              LayerSize[0] = 28 * 28;
63              LayerSize[LayerSize.Length - 1] = 10;
64              for (int i = 1; i < LayerSize.Length - 1; i++)
65              {
66                  Console.Write($"Size of hidden layer {i}: ");
67                  LayerSize[i] = int.Parse(Console.ReadLine());
68              }
69              Console.WriteLine("Which activation function: ");
70              string[] Functions = { "Sigmoid", "ReLU", "TanH" };
71              for (int i = 0; i < Functions.Length; i++)
72              {
73                  Console.WriteLine($"{i + 1}) {Functions[i]}");
74              }
75              int FunctionVal = int.Parse(Console.ReadLine());
76              return new Network(LayerSize, FunctionVal);
77          }
78          static Network LoadNetwork(Network currentnet)
79          {
80              Console.Clear();
81              Console.WriteLine("This will overwrite your currently loaded
    network");
82              Console.WriteLine("Press any key to continue or esc to
    return to the menu");
83              if (Console.ReadKey(true).Key == ConsoleKey.Escape)
84              {
85                  return currentnet;
86              }
87              if (currentnet != null)
88              {
```

```
                currentnet.Train.Close();
                currentnet.Test.Close();
            }
            Console.Write("Enter the file name you want to load: ");
            string filename = Console.ReadLine();
            BinaryReader Reader = new BinaryReader(File.Open(filename +
".bin", FileMode.Open));
            int FunctionVal = Reader.ReadInt32();
            int[] LayerSize = new int[Reader.ReadInt32()];
            for (int i = 0; i < LayerSize.Length; i++)
            {
                LayerSize[i] = Reader.ReadInt32();
            }
            Network net = new Network(LayerSize, FunctionVal);
            for (int i = 0; i < net.layers.Length - 1; i++)
            {
                RegLayer current = (RegLayer)net.layers[i];
                double[,] weights = new double[Reader.ReadInt32(),
Reader.ReadInt32()];
                for (int j = 0; j < weights.GetLength(0); j++)
                {
                    for (int k = 0; k < weights.GetLength(1); k++)
                    {
                        weights[j, k] = Reader.ReadDouble();
                    }
                }
                current.weights = new Matrix(weights);
                double[,] biases = new double[Reader.ReadInt32(),
Reader.ReadInt32()];
                for (int j = 0; j < biases.GetLength(0); j++)
                {
                    for (int k = 0; k < biases.GetLength(1); k++)
                    {
                        biases[j, k] = Reader.ReadDouble();
                    }
                }
                current.biases = new Matrix(biases);
                net.layers[i] = current;
            }
            return net;
        }
        static void SaveNetwork(Network net)
        {
            Console.Clear();
            Console.Write("Enter the file name you want to save the
network as: ");
            string filename = Console.ReadLine();
            BinaryWriter Writer = new BinaryWriter(new
```

```
FileStream(filename + ".bin", FileMode.Create));
            Writer.Write(net.FunctionVal);
            Writer.Write(net.LayerSize.Length);
            foreach (var layer in net.LayerSize)
            {
                Writer.Write(layer);
            }
            for (int i = 0; i < net.layers.Length - 1; i++)
            {
                RegLayer current = (RegLayer)net.layers[i];
                Writer.Write(current.weights.rows);
                Writer.Write(current.weights.columns);
                for (int j = 0; j < current.weights.rows; j++)
                {
                    for (int k = 0; k < current.weights.columns; k++)
                    {
                        Writer.Write(current.weights[j, k]);
                    }
                }
                Writer.Write(current.biases.rows);
                Writer.Write(current.biases.columns);
                for (int j = 0; j < current.biases.rows; j++)
                {
                    for (int k = 0; k < current.biases.columns; k++)
                    {
                        Writer.Write(current.biases[j, k]);
                    }
                }
            }
            Writer.Close();
            Console.Write("Save complete! Press any key to continue ");
            Console.ReadKey();
        }
        static void TrainNetwork(Network current)
        {
            Console.Clear();
            current.train();
        }
        static void TestNetwork(Network current)
        {
            Console.Clear();
            Console.WriteLine("Testing Network");
            Console.WriteLine(current.test() * 100 + "% Accurate");
            Console.ReadKey();
        }
        static void ViewData(Network current)
        {
            Console.Clear();
```

```
180             Console.WriteLine($"Number of layers:
    {current.LayerSize.Length}");
181             Console.WriteLine($"Number of hidden layers:
    {current.LayerSize.Length - 2}");
182             for (int i = 0; i < current.LayerSize.Length; i++)
183             {
184                 if (i == 0)
185                 {
186                     Console.WriteLine($"Size of first layer:
    {current.LayerSize[i]}");
187                 }
188                 else if (i == current.LayerSize.Length - 1)
189                 {
190                     Console.WriteLine($"Size of last layer:
    {current.LayerSize[i]}");
191                 }
192                 else
193                 {
194                     Console.WriteLine($"Size of hidden layer {i}:
    {current.LayerSize[i]}");
195                 }
196             }
197             Console.WriteLine($"Activation Function:
    {current.Function}");
198             Console.ReadKey();
199         }
200         static int Menu(bool netloaded)
201         {
202             Console.Clear();
203             if (netloaded)
204             {
205                 Console.WriteLine("1) Create Network");
206                 Console.WriteLine("2) Load Network");
207                 Console.WriteLine("3) Save Network");
208                 Console.WriteLine("4) Train Network");
209                 Console.WriteLine("5) Test Network");
210                 Console.WriteLine("6) View Data");
211                 return int.Parse(Console.ReadLine());
212             }
213             else
214             {
215                 Console.WriteLine("1) Create Network");
216                 Console.WriteLine("2) Load Network");
217                 Console.ForegroundColor = ConsoleColor.DarkGray;
218                 Console.WriteLine("3) Save Network");
219                 Console.WriteLine("4) Train Network");
220                 Console.WriteLine("5) Test Network");
221                 Console.WriteLine("6) View Data");
```

```
                Console.ForegroundColor = ConsoleColor.Gray;
                return int.Parse(Console.ReadLine());
            }
        }
    }
}
```

5.3 Testing Code

5.3.1 Matrix Test Video

```
Console.WriteLine("Test 1a:");
new Matrix(new double[,] { { 1, 2 }, { 1, 2 } }).Print();

new Matrix(new double[,] { { 1, 2, 3 }, { 1, 2, 3 } }).Print();

new Matrix(new double[2, 2], true).Print();
Console.ReadKey();

Console.Clear();
Console.WriteLine("Test 1b:");
new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
new Matrix(new double[,] { { 1, 2 }, { 5, 0 } }).Print();
(new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }) * new Matrix(new
double[,] { { 1, 2 }, { 5, 0 } })).Print();
Console.ReadKey();

Console.Clear();
Console.WriteLine("Tests 1b, 1j:");
new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
new Matrix(new double[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 }
}).Print();
try
{
    (new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }) * new Matrix(new
double[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } })).Print();
}
catch (MatrixNonConformableException ex)
{
    Console.WriteLine(ex.Message);
}
Console.ReadKey();

Console.Clear();
Console.WriteLine("Test 1c:");
new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
new Matrix(new double[,] { { 4, 3 }, { 2, 1 } }).Print();
```

```
34  (new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }) + new Matrix(new
    double[,] { { 4, 3 }, { 2, 1 } })).Print();
35
36  new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
37  new Matrix(new double[,] { { 0, 1 }, { 2, 3 } }).Print();
38  (new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }) - new Matrix(new
    double[,] { { 0, 1 }, { 2, 3 } })).Print();
39  Console.ReadKey();
40
41  Console.Clear();
42  Console.WriteLine("Test 1c , 1j:");
43  new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
44  new Matrix(new double[,] { { 1, 2, 3 }, { 4, 5, 6 } }).Print();
45  try
46  {
47      (new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }) + new Matrix(new
    double[,] { { 1, 2, 3 }, { 4, 5, 6 } })).Print();
48  }
49  catch (MatrixNonConformableException ex)
50  {
51      Console.WriteLine(ex.Message);
52  }
53
54  new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
55  new Matrix(new double[,] { { 1, 2, 3 }, { 4, 5, 6 } }).Print();
56  try
57  {
58      (new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }) - new Matrix(new
    double[,] { { 1, 2, 3 }, { 4, 5, 6 } })).Print();
59  }
60  catch (MatrixNonConformableException ex)
61  {
62      Console.WriteLine(ex.Message);
63  }
64  Console.ReadKey();
65
66  Console.Clear();
67  Console.WriteLine("Test 1d:");
68  new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
69  (2 * new Matrix(new double[,] { { 1, 2 }, { 3, 4 } })).Print();
70  Console.ReadKey();
71
72  Console.Clear();
73  Console.WriteLine("Test 1e:");
74  new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
75  (~new Matrix(new double[,] { { 1, 2 }, { 3, 4 } })).Print();
76
77  new Matrix(new double[,] { { 1, 2, 3 }, { 4, 5, 6 } }).Print();
```

```
78  (~new Matrix(new double[,] { { 1, 2, 3 }, { 4, 5, 6 } })).Print();
79  Console.ReadKey();
80
81  Console.Clear();
82  Console.WriteLine("Test 1f:");
83  new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
84  Console.WriteLine(Matrix.Det(new Matrix(new double[,] { { 1, 2 }, { 3, 4
    } })));
85
86  new Matrix(new double[,] { { 5, -6 }, { 7, 4 } }).Print();
87  Console.WriteLine(Matrix.Det(new Matrix(new double[,] { { 5, -6 }, { 7,
    4 } })));
88
89  new Matrix(new double[,] { { 1, 2, 3 }, { 4, 5, 6 } }).Print();
90  try
91  {
92      Console.WriteLine(Matrix.Det(new Matrix(new double[,] { { 1, 2, 3 },
    { 4, 5, 6 } })));
93  }
94  catch (MatrixNonSquareException ex)
95  {
96      Console.WriteLine(ex.Message);
97  }
98  Console.ReadKey();
99
100 Console.Clear();
101 Console.WriteLine("Test 1g:");
102 new Matrix(new double[,] { { 1, -2 }, { 3, 4 } }).Print();
103 Matrix.Sigmoid(new Matrix(new double[,] { { 1, -2 }, { 3, 4 }
    })).Print();
104 Console.ReadKey();
105
106 Console.Clear();
107 Console.WriteLine("Test 1h:");
108 new Matrix(new double[,] { { 1, -2 }, { 3, 4 } }).Print();
109 Matrix.Tanh(new Matrix(new double[,] { { 1, -2 }, { 3, 4 } })).Print();
110 Console.ReadKey();
111
112 Console.Clear();
113 Console.WriteLine("Test 1i:");
114 new Matrix(new double[,] { { 1, -2 }, { 3, 4 } }).Print();
115 Matrix.ReLU(new Matrix(new double[,] { { 1, -2 }, { 3, 4 } })).Print();
116 Console.ReadKey();
```

5.3.2 Image Test Video

```
Console.WriteLine("Test 1f Retest:");
new Matrix(new double[,] { { 1, 2 }, { 3, 4 } }).Print();
Console.WriteLine(Matrix.Det(new Matrix(new double[,] { { 1, 2 }, { 3, 4
} })));
Console.ReadKey();

MNISTReader train = new MNISTReader("train");
for (int T = 0; T < 5; T++)
{
    Console.Clear();
    Console.WriteLine("Test 3a:");
    Console.WriteLine("Training Data");
    Image current = train.ReadNext();
    Bitmap image = new Bitmap(train.imagesize, train.imagesize);
    for (int i = 0; i < train.imagesize; i++)
    {
        for (int j = 0; j < train.imagesize; j++)
        {
            image.SetPixel(i, j, Color.FromArgb((int)current.data[i +
train.imagesize * j], (int)current.data[i + train.imagesize * j],
(int)current.data[i + train.imagesize * j]));
        }
    }
    Console.WriteLine($"Image Label: {current.label}");
    Picture display = new Picture(image);
    display.ShowDialog();
}
train.Close();
Console.ReadKey();

MNISTReader test = new MNISTReader("test");
for (int T = 0; T < 5; T++)
{
    Console.Clear();
    Console.WriteLine("Test 3b:");
    Console.WriteLine("Test Data");
    Image current = test.ReadNext();
    Bitmap image = new Bitmap(test.imagesize, test.imagesize);
    for (int i = 0; i < test.imagesize; i++)
    {
        for (int j = 0; j < test.imagesize; j++)
        {
            image.SetPixel(i, j, Color.FromArgb((int)current.data[i +
```

```
test.imagesize * j], (int)current.data[i + test.imagesize * j],
(int)current.data[i + test.imagesize * j]));
        }
    }
    Console.WriteLine($"Image Label: {current.label}");
    Picture display = new Picture(image);
    display.ShowDialog();
}
test.Close();
Console.ReadKey();
```