**Phase 4 Report**

Team Number: 727-E

John Balawejder (20231865)

Ethan Parliament (20220776)

Thomas Tesselaar (20233228)

Farhan Naushad (202404707)

Dinali Assylbek (20242924)

Eric Chaput (20230041)

Client: Travis Cossarini

Faculty Advisor: Dr. Greenspan

Project Manager: Adam Farley

Date submitted: February 22nd, 2021

*“We do hereby verify that this written report is our own individual work and contains our own original ideas, concepts, and designs. No portion of this report has been copied in whole or in part from another source, with the possible exception of properly referenced material.”*

# Executive Summary

This report is written by the design team about a computer program, written in Python, a programming language, that was made to analyze handwritten digits and output the given digits using machine learning techniques. The research done on the subject included a variety of techniques and how they work, with each group member researching a separate type of machine learning model. The researched models were K-Nearest Neighbors, Naïve Bayes Predictor, Multi-Layer Perceptron, Convolutional Neural Network, Linear Classifier, and Support Vector Machine.

The design of the project included building three different programs written using three of the researched methods of machine-learning. Three models were designed rather than one to increase the chance that at least one of them is successful. It also allowed our group to gain a wider and deeper knowledge of machine learning. However, it meant higher chances for error in any one of the programs since only 2 people were working on it. The models were chosen based on the highest probability of being successful according to accuracy, a history of similar projects, and the difficulty of coding and implementing each model. The models chosen were Logistic Regression, Convolutional Neural Network, and Multi-Layer Perceptron.

Logistic Regression is a Linear Classifier, which is a type of machine learning that uses line and plane separators to divide up data. It is mainly used in data classification. It uses the lines and planes to learn the position of data points.

Convolutional Neural Networks use a 3-D array of pixels consisting of height, width, and color. The color dimension will be an RGB value. This means that each pixel will have a height, width, and color value, and each pixel will be represented in the 3-D array.

Multi-Layer Perceptron is a simple algorithm that performs binary classification. It is primarily used in data encryption, data visualization and data compression.

The modelling of each type was done by two group members. Each pair worked on it for 3 weeks with an objective of reaching an accuracy of 90%. Once the modelling stage was completed, the group members began the testing and evaluation phase. For one week, the pairs worked on refining the final models. Once each model was complete, the group decided on a model to present to the client.

Each of the solutions worked well and surpassed our initial objective 90% accuracy. Using a weighted evaluation matrix (WEM), they were each compared. The WEM compared accuracy, with a weight of 5, code readability and scalability, with a weight of 3, and the time it takes for the program to run, with a weight of 4. The final score for linear classifiers is 41, multi-layer perceptron is 47.5, and convolutional neural network is 47.5. The final product chosen was the program written using a Convolutional Neural Network.

The cost for the project was $0. This is because the research and implementation were free online. There were no physical pieces or online libraries that needed to be purchased.

Table of Contents

[Executive Summary i](#_Toc68552541)

[List of Tables iii](#_Toc68552542)

[List of Figures iii](#_Toc68552543)

[Part I - Key Information 1](#_Toc68552544)

[Problem Statement and Scope Definition 1](#_Toc68552545)

[Background Information 1](#_Toc68552546)

[Design Solution 2](#_Toc68552547)

[Conclusions 3](#_Toc68552548)

[Part II - Technical Information 3](#_Toc68552549)

[Conceptual Design Solutions 3](#_Toc68552550)

[Logistic Regression-John/Eric 3](#_Toc68552551)

[Multi-Layer Perceptron 4](#_Toc68552552)

[Convolutional Neural Network 6](#_Toc68552553)

[Decision Making 6](#_Toc68552554)

[Implementation – Farhan 7](#_Toc68552555)

[Project Plan 7](#_Toc68552556)

[Work Breakdown Structure 7](#_Toc68552557)

[Financial Analysis 9](#_Toc68552558)

[Evaluation 10](#_Toc68552559)

[References 12](#_Toc68552560)

[Appendix I 13](#_Toc68552561)

[Appendix II 14](#_Toc68552562)

[Appendix III 14](#_Toc68552563)

[Appendix IV 17](#_Toc68552564)

[Appendix V 18](#_Toc68552565)

# List of Tables

Table 1: A confusion matrix of a sample model of the MLP. Model used 4 hidden layers of 100 neurons in each layer. Along the top axis is the values the model guessed, and along the left axis is the actual values of the dataset. For example, the model predicted a 2 when it was a 7, 12 times. 5

Table 2: Displays the criteria and their weights, along with the weighted score of each developed model 7

Table 3: The work breakdown structure for this project. Includes task numbers, task description, expected duration of the task, and the activity leader. 8

Table 4: The total costs required to run this project, broken down month by month. 9

Table 5: The evaluation rubric developed by the team used to evaluate the different proposed models. Includes the different requirements for the model to be successful. 10

Table 6: A list of tasks required to complete this reports along with a description, duration, and the member responsible for each one 13

Table 7: The criteria that were used to evaluate the three proposed solutions and the meaning of each score 14

# List of Figures

[Figure 1: Results from the 10th epoch of the Logistic Regression model 4](#_Toc68552572)

[Figure 2: Shows the accuracy(blue) and time(orange) of each model based on 100 Neurons in each hidden layer. Each model was tested on the same computer and in the same conditions to remove any sources of error 4](#_Toc68552573)

[Figure 3: Shows the accuracy(blue) and time(orange) of each model based on 5 neurons in each hidden layer. Each model was tested under the same conditions to remove any sources of error. 5](#_Toc68552574)

[Figure 4: The results of the final Convolutional Neural Network model 6](#_Toc68552575)

[Figure 5: The Gantt chart based on the project timeline. Includes the different tasks, who was assigned to the task, how complete the task is, and the expected start and finish dates for each task. 8](#_Toc68552576)

[Figure 6: The results of the original Convolutional Neural Network model 19](#_Toc68552577)

# Part I - Key Information

# Problem Statement and Scope Definition

The categorization of handwritten characters by computer algorithms has been a challenge for a long time for programmers. The client, QMIND (Queen’s Machine Intelligence and Neuro-evolution Design), contracted the team to develop a model to help categorize different handwritten numerical digits from the MNIST (Modified National Institute of Standards and Technology) database which is a collection of 70,000 images of handwritten digits, broken into a 60,000-image training set and a 10,000-image testing set, that are each 28 pixels wide and 28 pixels tall. The objective of this project was to have a finished program, which should be able to recognize the handwritten digits from the MNIST’s data set with an accuracy above 90%, and the overall project should be completed in a time-appropriate manner. The key stakeholders involved in this project include the engineering students, the client (QMIND), programming language companies, potential industries interested in the product (Medical, Judicial, Law Enforcement).

The engineering students will be using Python to code the product based on the specific requirements and needs of the client (QMIND). The modeling concepts used for this handwritten digital analysis project can also be used by QMIND for like-minded projects such as facial recognition, bioinformatics, and the classification of images.

This problem contains constraints that can be dealt with in a few diverse ways. One constraint is the lack of technical knowledge and computing power available to carry out the task. The way to deal with that is to use a model that is both simple to design and operate, as well as being able to be properly run on a laptop. Another constraint is the time required. Building successful machine learning models take long amounts of time due to the way the models are typically built. It takes time to research what the model needs to do, and it also takes time to build and train the model. Because the program is only being exposed to images that are 28x28, it will only work on images that are of that size. Based on these constraints, the scope of this project includes: Researching different techniques for machine learning, building the different models, and comparing the effectiveness of each one, and having a working program be able to identify hand-written digits with at least 90% accuracy.

The problem statement and project scope have been updated from the Phase 1 document to ensure better results from the team. More constraints have been added that were not thought of before, and the project scope has been made more realistic to better fit the timeline and still follow the requirements.

# Background Information

There are many ways to approach the task of digit analysis, although the theory in which builds these different approaches is the same. Machine learning is fundamental in the solution to the problem, it enables the programmer to utilize test data to train an algorithm for it to learn and come to decisions on its own. There are many ways in which machine learning can be accomplished, one being neural networks. Filters are applied to the input data and multiplied using the dot product, which is then summed and referred to as the “scalar product”. This filter is smaller than the array of input to allow it to be applied at multiple points, ensuring all the data is covered. This process of cross-correlation is called convolution and it is used to generate the feature map. The feature map is then used to distinguish the differences between different potential numbers [1].

# Design Solution

Due to the nature of the project, it was decided that three models would be built, and as explained in the Decision Making section of this report it was decided to go with the Convolution Neural Network as the final model. The code for this model is shown below.

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train /= 255

x\_test /= 255

model = Sequential()

model.add(Conv2D(28, kernel\_size=(3,3), input\_shape=input\_shape))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation=tf.nn.relu))

model.add(Dropout(0.2))

model.add(Dense(10,activation=tf.nn.softmax))

model.compile(optimizer='adam',

              loss='sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

model\_log = model.fit(x=x\_train,y=y\_train, epochs=8, validation\_data=(x\_test, y\_test))

score = model.evaluate(x\_test, y\_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

The model begins by importing the necessary libraries and reshaping the dataset into a form that is useable by the “learning” part of the model. The “learning” part of the model is sequential, consisting of 6 layers which it runs through. The first layer is a Conv2D layer. This layer is used to make the convolutional network that deals with the images. The second layer is MaxPooling2D which is used to reduce the spatial dimensions of the output volume [2]. The third layer is a flatten layer. Flatten is a function which transforms the feature matrix into a single column which can be fed into the neural network for processing [3]. The fourth and sixth layers are Dense layers. These layers add the flattened layer to the neural network. The fifth layer is a dropout layer which will “drop out” a certain portion of the dataset, in this case 20% each time, to prevent overfitting to the training data. Overall, the model runs through 8 epochs. Each epoch is a time the entire dataset is run through the model. After 8 epochs it can achieve a 98.6% accuracy on the testing set and the time it takes to run the entire model is about 4 minutes.

# Conclusions

The group designed three different models to solve the given problem. Each solution uses a different machine learning method to analyze the digits. The three models are logistic regression, multi-layer perceptron, and convolutional neural networks. Using a weighted evaluation matrix, the three models were compared, and the convolutional neural networks was chosen. Although each final design performed above the base standard of having an accuracy above 90%, CNN reached the highest accuracy, at 98.6%. However, each model can still be fine-tuned to perform even better. The run time for convolutional neural networks is about 4 minutes, which can be brought down significantly. Other improvements can also be made, such as combining two of the models to test if the accuracy increases.

# Part II - Technical Information

# Conceptual Design Solutions

In the Phase 3 report three proposals on how to proceed were considered, and using the criteria and evaluation shown in the two tables Appendix II it was decided to create three unique types of machine learning models, one linear classifier and two neural networks, which would then be evaluated. The three models chosen were Logistic Regression, Multi-layer Perceptron, and Convolutional Neural Network. All models were successful by the original 90% benchmark that was set as a scale.

### Logistic Regression-John/Eric

Due to the nature of the project, it was important to realize the goal of statistical classification. Linear classifiers are used to make classification decisions based on linear combinations, this would include many different methods such as: logistic regression, perceptron, and support vector analysis. Logistic regression was chosen because it is known to be a simple and efficient method to solve classification problems. Logistic regression uses the Sigmoid Function to determine the classification of the handwritten digit. The data is trained by using different layers starting with a convolutional layer and ending with a max pooling layer. The feature map is then flattened so that the classifier can easily read the unique feature values and use them to generate classification. The logistic regression code is found in Appendix III. The algorithm takes advantage of 10 epochs to train the classifier which yields an accuracy of 91.2%, which is above the initial benchmark of 90%. The final epoch can be seen below in Figure 1.

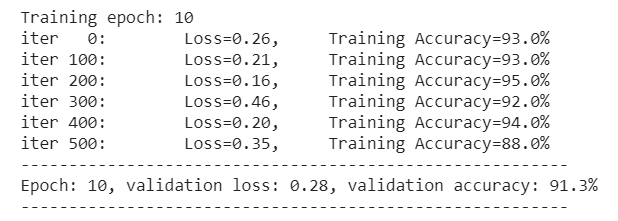


Figure : Results from the 10th epoch of the Logistic Regression model

### Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) was chosen to be one of the proposed models because of the previous success it has had for this problem [4]. A MLP works by using a series of hidden layers of neurons to classify each of the digits. By doing some preliminary testing, it was found that adding more hidden layers increases the accuracy, but also takes longer to run.

Figure : Shows the accuracy(blue) and time(orange) of each model based on 100 Neurons in each hidden layer. Each model was tested on the same computer and in the same conditions to remove any sources of error

Shown in Figure 2 is how models with hidden layers of 100 neurons performed. Generally, as you add more hidden layers the accuracy gets better, but after 3 hidden layers the accuracy hardly changes, while the time increases.

Figure : Shows the accuracy(blue) and time(orange) of each model based on 5 neurons in each hidden layer. Each model was tested under the same conditions to remove any sources of error.

Figure 3 shows a chart similar to Figure 2, but with five Neurons in each hidden layer instead 100. These models outperformed the 100 Neuron models due to their much shorter run time, and slightly higher accuracy scores. However, you can see the effects of over fitting the model once there are six hidden layers, as the accuracy drops significantly. This happens when the model becomes too sensitive to new information and will not recognize anything it has not seen already.

Through many iterations of the model, the highest accuracy score achieved was 98.59% in 450 seconds (7 minutes 30 seconds). This model consists of four hidden layers with five neurons in each layer. However, another model was constructed that was slightly less accurate, but was much faster. A model of four hidden layers, with 100 neurons in the first three layers, and ten neurons in the final layer had 97.61% accuracy and took 152 seconds (2 minutes 32 seconds).

**Confusion Matrix and Classification Report for MLP**

Table : A confusion matrix of a sample model of the MLP. Model used 4 hidden layers of 100 neurons in each layer. Along the top axis is the values the model guessed, and along the left axis is the actual values of the dataset. For example, the model predicted a 2 when it was a 7, 12 times.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 965 | 1 | 3 | 1 | 1 | 2 | 3 | 1 | 2 | 1 |
| 1 | 2 | 1120 | 3 | 2 | 0 | 0 | 1 | 0 | 7 | 0 |
| 2 | 6 | 1 | 988 | 14 | 3 | 0 | 6 | 4 | 9 | 1 |
| 3 | 1 | 0 | 14 | 970 | 1 | 5 | 1 | 4 | 14 | 0 |
| 4 | 0 | 5 | 2 | 1 | 939 | 0 | 7 | 8 | 5 | 15 |
| 5 | 4 | 0 | 0 | 22 | 2 | 834 | 9 | 1 | 18 | 2 |
| 6 | 4 | 2 | 2 | 0 | 2 | 7 | 936 | 0 | 4 | 1 |
| 7 | 2 | 3 | 12 | 15 | 6 | 0 | 0 | 978 | 5 | 7 |
| 8 | 12 | 2 | 1 | 4 | 3 | 5 | 3 | 2 | 938 | 4 |
| 9 | 5 | 4 | 1 | 6 | 16 | 2 | 3 | 10 | 22 | 940 |

As seen in Table 1 is the confusion matrix for one of the MLP models. This shows where the model was making incorrect guesses. Any guess that is not along the diagonal from (0,0) to (9,9) is an incorrect guess. As you can see, the most common mistake was guessing a 3 when it a 5, this happened 22 times. Part of the reason for the number 5 having the worst results is because it was the number with the least amount of training. Through finer precision and tweaking of the model, this could be greatly improved on.

### Convolutional Neural Network

Convolutional Neural Networks were chosen to model because it is considered one of the best types of models for computer vision and image classification [5], and because of the abundance of available information. The initial version took 1420 seconds (24.67 minutes) to run and only had an accuracy of 82.2%. This model and its results are shown in Appendix V. Throughout the building process, the design was iterated upon significantly until it was able to achieve a final accuracy of 98.6% and ran in 254 seconds (4.23 minutes). The components of the model are described in depth in the Design Solution section of the report. Overall, the model runs through 8 epochs. Each epoch is a time the entire dataset is run through the model. With each consecutive epoch the accuracy increases until 10 epochs where it hits a maximum accuracy of about 98.7% and past that the accuracy fluctuates between 98.65% and 98.75%. It was decided that 8 was the ideal number of epochs since it takes 80% of the time to run as 10 and the difference in accuracy is marginal. The results of the model are shown in Figure 4 below.

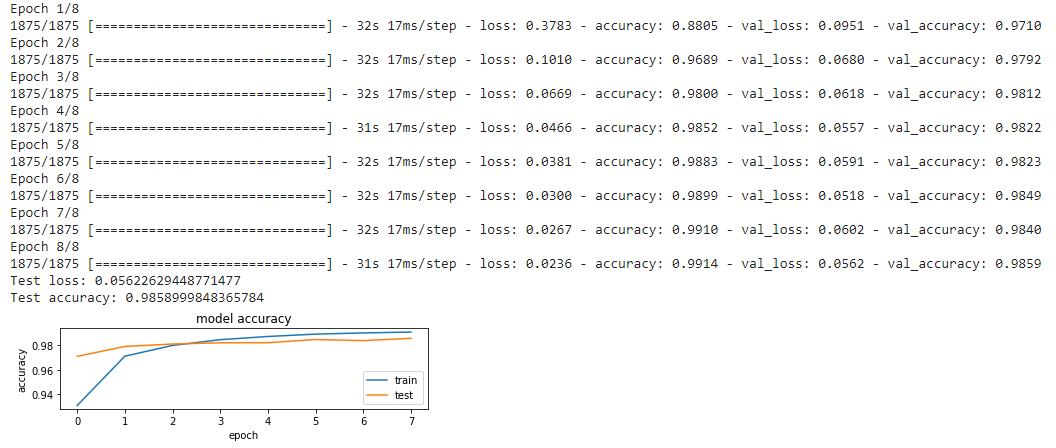


Figure : The results of the final Convolutional Neural Network model

K Cross-validation, a technique where part of the training set is removed and used as a test set in order to prevent over-fitting and increase test-set accuracy at the expense of training set accuracy [6], was used however did not increase the testing accuracy, most likely due to the fact that the model already had a dropout function to prevent over fitting.

# Decision Making

Table 2 displays a weighted matrix of the three models developed as solutions. The three models are logistic regression, multi-layer perceptron, and convolutional neural network. The three models were evaluated using three criteria points, as described in Table 5. The final decision was Model 3, Convolutional Neural Network.

Table 2: Displays the criteria and their weights, along with the weighted score of each developed model

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Model 1: Logistic Regression** | | **Model 2: Multi-Layer Perceptron** | | **Model 3: Convolutional Neural Network** | |
| **Criteria** | **Weight** | **Score** | **Weighted** | **Score** | **Weighted** | **Score** | **Weighted** |
| Accuracy | 5 | 3 | 15 | 4.5 | 22.5 | 5 | 25 |
| Code Readability and Scalability | 3 | 2 | 6 | 3 | 9 | 3.5 | 10.5 |
| Time to Run | 4 | 5 | 20 | 4 | 16 | 3 | 12 |
| Total (Sum) |  |  | 41 |  | 47.5 |  | 47.5 |

While the linear classifier ran significantly quicker than either neural network, the increased accuracy with which the two neural networks ran and the increased readability they had made them the preferred options and of the two neural networks, the Convolutional Neural Network had a slightly higher accuracy. Thus, it was decided that the Convolution Neural Network would be the final model to be presented to the client.

# Implementation – Farhan

The initial procedure was to code the 3 top models we chose, which were Multi-Layer Perceptron, Linear Classifier, and Convolutional Neural Network. If the accuracy was above 90% then the model was classified as successful, and the model which had the greatest accuracy would then be chosen as the final model. These models were coded using the programming language Python and were done using text editors like JupyterNotebook, Pycharm, etc. and used libraries like Sci-Kit Learn, and TensorFlow to import the algorithms, NumPy for data calculation, Matplotlib and Seaborn for data visualization and other libraries like Pandas. Then the models were trained using the MNIST data set, which contains 60,000 training images and were tested using their datasets 10,000 testing images. During the testing, the models identify the images based on what it thinks they are and then cross reference them to the test labels file to find the accuracy of the model. The final design and procedure have not been changed from what was planned originally.

The people who could use the model for handwritten analysis are people who works in different departments such as Medical, Judicial, Law Enforcement, etc. It would be used in their respective fields of work. The model has not gone through any specific changes to help the society, the final model is efficient and accurate as it is.

# Project Plan

## Work Breakdown Structure

The work breakdown structure is the first major step in project management as it contains and organizes the main goals of the project along with their components, expected duration, and the task leader. The work breakdown structure for the teams Handwritten Digit Analysis project can be seen below in Table 3. A visualization of the timeline and work breakdown can be seen in Figure 5.

Table : The work breakdown structure for this project. Includes task numbers, task description, expected duration of the task, and the activity leader.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task or Phase** | **Task Description** | **Expected Duration (Days)** | **Activity Leader** |
| **1** | **Define the scope of the project** | **14** |  |
| 1.1 | Meet with client to discuss shareholder needs. | 1 | Group |
| 1.2 | Determine constraints and available resources. | 2 | Dinali, Eric |
| 1.3 | Decide scope. | 4 | Ethan, Farhan |
| 1.4 | Conduct research on potential methods/options. | 2 | John, Thomas |
| 1.5 | Finish Phase 2 Report. | 5 | All |
| **2** | **Design the prototype** | **19** |  |
| 2.1 | Evaluate the potential options. | 4 | John, Ethan, Eric |
| 2.2 | Choose which option to proceed with. | 3 | Thomas, Farhan, Dinali |
| 2.3 | Conduct in depth research on the method that will be used. | 5 | Group |
| 2.4 | Draft and edit Phase 3 proposal report. | 7 | All |
| **3** | **Model and Build Display Prototype** | **33** |  |
| 3.1 | Determine a draft of algorithm in pseudocode. | 4 | John, Ethan, Eric |
| 3r.2 | Familiarize ourselves with libraries and Google Colab . | 7 | Farhan, Dinali, Thomas |
| 3.3 | Code the prototype. | 19 | Group |
| **4** | **Test and deliver final product** | **11** |  |
| 4.1 | Identify ways to increase the accuracy. | 2 | Dinali, Farhan, Thomas |
| 4.2 | Work on fine tuning the algorithm to enhance accuracy. | 2 | John, Ethan, Eric |
| 4.3 | Draft the final Phase 4 report and presentation. | 7 | Group |

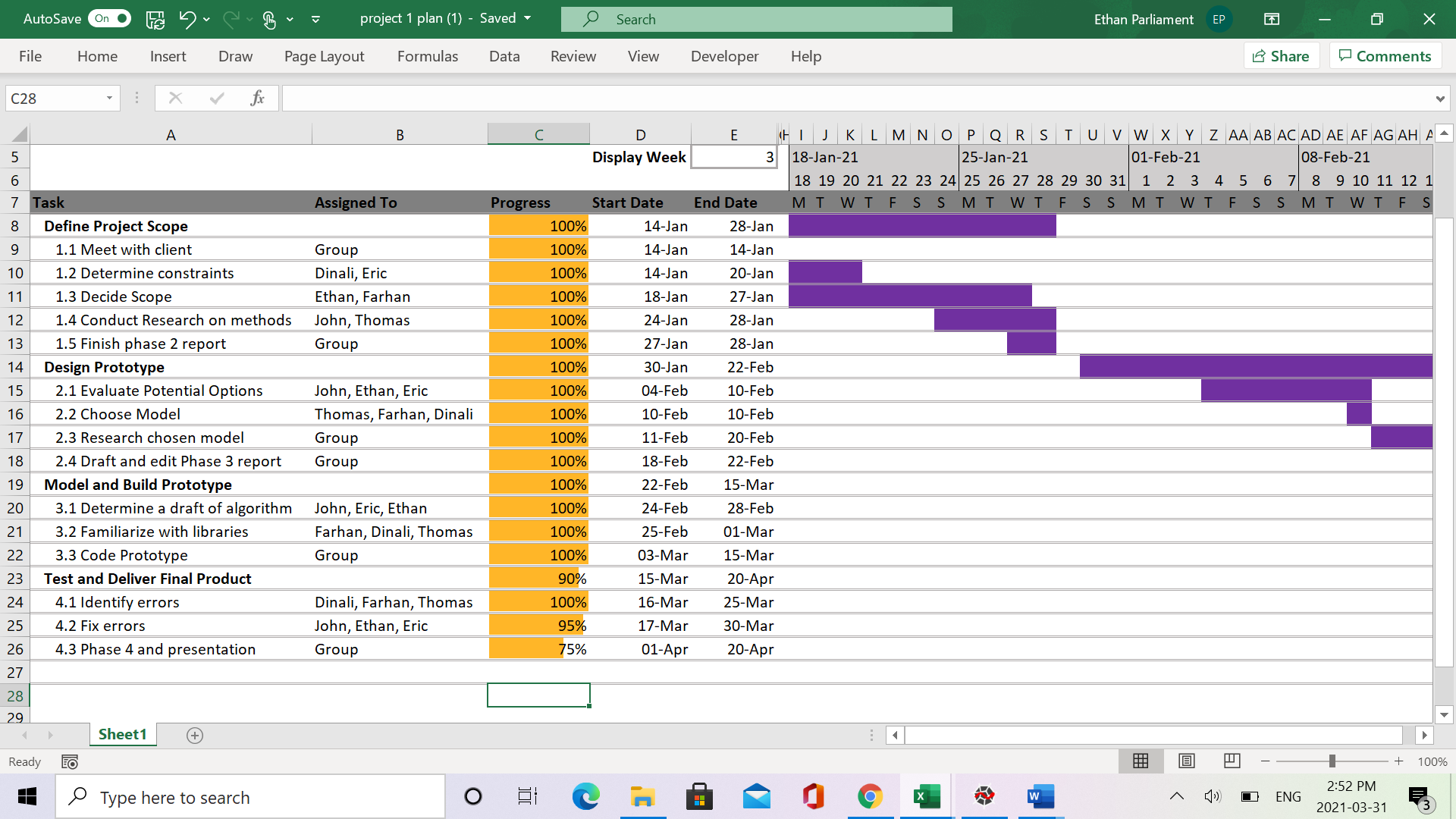


Figure : The Gantt chart based on the project timeline. Includes the different tasks, who was assigned to the task, how complete the task is, and the expected start and finish dates for each task.

As seen in Figure 5, the modeling and programming of this project is now complete. The project encountered very few problems with regards to the timeline proposed in Phase 1. The original timeline, shown in Table 3, allowed sufficient time to complete tasks as well as produce a quality product. No major changes were made to the project timeline, only who was assigned to certain tasks changed.

# Financial Analysis

Table : The total costs required to run this project, broken down month by month.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Costs involved in $ (Canadian dollar)** | January 2021 | February 2021 | March 2021 | April 2021 | Total expenditure in $ (Canadian dollar) |
| Workforce |  |  |  |  |  |
| * Training | 0 | 0 | 0 | 0 | 0 |
| * Student Expenditure | 0 | 0 | 0 | 0 | 0 |
| Hardware |  |  |  |  |  |
| * Laptop/PC | 0 | 0 | 0 | 0 | 0 |
| * Furniture | 0 | 0 | 0 | 0 | 0 |
| Software |  |  |  |  |  |
| * Python | 0 | 0 | 0 | 0 | 0 |
| * MNIST data set | 0 | 0 | 0 | 0 | 0 |
| * Libraries | 0 | 0 | 0 | 0 | 0 |
| Other expenditures |  |  |  |  |  |
| * Marketing | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  | Total: 0 |

**Cost benefit analysis**

Total expenditure [0] / Annual returns [0] = Payback (in years) [0]

A financial analysis conducted based on various criteria such as Manpower, Hardware, Software and Other expenditures in Table 4 show that the total expenditure for the project is $0. The reason financial factors play no role in this project is because the methods used to research and code the machine learning program was already available free of cost. [7]

# Evaluation

Table : The evaluation rubric developed by the team used to evaluate the different proposed models. Includes the different requirements for the model to be successful.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **1** | **2** | **3** | **4** | **5** |
| **Accuracy** | The model provides an accuracy of less than 84.9% | The model provides an accuracy ranging from 85-89.9% | The model provides an accuracy ranging from 90-93.9% | The model provides an accuracy ranging from 94-97.9% | The model consistently provides an accuracy of >98% |
| **Code Readability** | The code is extremely difficult to comprehend and would be unable to be implemented in an external application. | The code is difficult to comprehend and could be used in an external application. | The code is understandable to an extent and could be used in an external application. | The code is easily understandable and could be used in an external application. | The code is extremely clear and could easily be implemented in an external application. |
| **Time to run model** | The model takes more than 6 minutes to run. | The model takes 4.5 to 6 minutes to run. | The model takes 3 to 4.5 minutes to run. | The model takes about 2 to 3 minutes to run. | The model takes less than 2 minutes to run. |

Table 5 describes the criteria that the models were evaluated on. The criteria of the following table include the accuracy, code readability, and the time it takes to run the model, each criterion being evaluated out of 5. Then by using the weighted evaluation matrix in Table 1, the models were ranked to display the effectiveness of the model and its algorithm. The accuracy of the model was the biggest factor for the client and was weighted the most. The models needed to score a 90% or higher accuracy to meet the requirements of the client and be considered a success. The code readability was used to display the simplicity of building a set model. Code readability was considered essential by the team so that other stakeholders could easily understand the model and build the model into an external application if desired. The time to run the models was also chosen for the evaluation. The models needed to be able to run in under 5 minutes to be considered successful.

Through multiple iterations of code and constant revision to obtain the best results, each proposed design solution was put through the evaluation rubric and a weighted evaluation matrix to decide which model would be the final model. After evaluating using the matrix the results, as shown in Table 2, was that the Convolutional Neural Network and Multi-Layer Perceptron models tied for highest score so due to its greater accuracy Convolution Neural Network was chosen. It combined high accuracy with reliable speeds, and perfectly suits the clients’ needs. It delivers on the 90% accuracy goal set at the beginning of the project with 98.59% accuracy and runs in only 254 seconds (4.23 minutes). The other models performed to the client’s expectations as well but had some downside to them. While the logistic regression was able to run faster than the other models, it had a lower accuracy score of 91.7%. The Multi-Layer Perceptron was faster at 152s but had a slightly lower accuracy of 97.61%.

# References

|  |  |
| --- | --- |
| [1] | J. Brownlee, "How Do Convolutional Layers Work in Deep Learning Neural Networks?," Machine Learning Mastery, 17 April 2020. [Online]. Available: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/. [Accessed 17 March 2021]. |
| [2] | geeksforgeeks, "Keras.Conv2D Class," 18 May 2020. [Online]. Available: https://www.geeksforgeeks.org/keras-conv2d-class/. [Accessed 02 March 2021]. |
| [3] | D. Mwiti, "Convolutional Neural Networks: An Intro Tutorial," 8 May 2018. [Online]. Available: https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed. [Accessed 01 March 2021]. |
| [4] | X. Zuo, "Technical Report: Multilayer Perceptron on MNIST," Department of Computer Science University of Freiberg, 2016. [Online]. Available: https://share.cocalc.com/share/32b94ee413d02759d719862907bb0a85a76c43f1/2016-11-07-175929.pdf. [Accessed 30 March 2021]. |
| [5] | I. Hasabo, "Image Classification using Machine Learning and Deep Learning," Medium, 2 May 2020. [Online]. Available: https://medium.com/swlh/image-classification-using-machine-learning-and-deep-learning-2b18bfe4693f. [Accessed 16 February 2021]. |
| [6] | J. B. PhD, "A Gentle Introduction to k-fold Cross-Validation," Machine Learning Mastery, 3 August 2020. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/. [Accessed 15 March 2021]. |
| [7] | "Cost Benefit Analyis," Smart Sheet, [Online]. Available: https://www.smartsheet.com/expert-guide-cost-benefit-analysis. [Accessed 30 March 2020]. |
| [8] | S. Swaminathan, "Logistic Regression - Detailed Overview," Towards Data Science, 18 March 2018. [Online]. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc. [Accessed 3 March 2021]. |

# Appendix I

Table : A list of tasks required to complete this reports along with a description, duration, and the member responsible for each one

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Description of Activity** | **Activity Duration (hours)** | **Individual Responsible For Activity** |
| Executive Summary | Drafting and finalizing a copy of the executive summary | 1 | Eric |
| Problem Statement | Drafting and finalizing a copy of the problem statement and project scope | 1 | Dinali |
| Background Information | Describing the background information needed to understand the report. | 2 | John |
| Design Solution | Describing and explaining the final design solution | 2 | Thomas |
| Conclusions | Summarizing the findings into a concise but comprehensive conclusion | 2 | Eric |
| Conceptual Design Solutions | Describe and illustrate the possible design solutions, or approaches to subsets of your overall problem. | 1.5 each | Group |
| Decision Making | Describing the process and results of the decision between the three models | 2 | Thomas |
| Project Plan | Identify any changes to the timeline for compared to the original proposal | 1 | John, Ethan |
| Financial Analysis | Analysis of associated costs with the model | .5 | Farhan |
| Evaluation | Describe the evaluation process for the final model | 3 | Ethan, Dinali |
| Editing and formatting | General formatting and editing. | 2 | Group |

# Appendix II

Table 7: The criteria that were used to evaluate the three proposed solutions and the meaning of each score

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **1** | **2** | **3** | **4** | **5** |
| **Likelihood of Success** | Unlikely to have a model with a 90% accuracy. | Somewhat likely to have a model with 90% accuracy. | Very likely to have a model with 90% accuracy. | Likely to have a model with an accuracy over 90%, possibility of multiple models with a 90% accuracy. | Likely to have multiple models with an accuracy over 90%. |
| **Time Required** | All models can be completed in over 2.5 hours. | All models can be made in over 2 hours. | All models can be completed in over 1.5 hours. | All models can be completed in over an hour. | All models can be completed in under an hour. |
| **Modelling/ Implementation Difficulty** | Significant coding required and difficult to implement. | Prior projects in coding are needed to complete the project. | Moderate coding required and | A small amount of coding required but still easy to implement. | Minimal coding required and can be easily implemented. |
| **Collaboration Difficulty** | All 6 members working together, very difficult for scheduling, productivity | Most work done as a full group, limited individual work, lots of overlap | Some full group work, some individual/ small group work | Most work done individu-ally or in pairs, everyone has chance to code | All work is individual so no issues with scheduling, time zones, productivity |

# Appendix III

**Code for Logistical Regression Model**

%tensorflow\_version 1.x

import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

img\_h = img\_w = 28             # MNIST images are 28x28

img\_size\_flat = img\_h \* img\_w  # 28x28=784, the total number of pixels

n\_classes = 10                 # Number of classes, one class per digit

def load\_data(mode='train'):

    from tensorflow.examples.tutorials.mnist import input\_data

    mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)

    if mode == 'train':

        x\_train, y\_train, x\_valid, y\_valid = mnist.train.images, mnist.train.labels, \

                                             mnist.validation.images, mnist.validation.labels

        return x\_train, y\_train, x\_valid, y\_valid

    elif mode == 'test':

        x\_test, y\_test = mnist.test.images, mnist.test.labels

    return x\_test, y\_test

def randomize(x, y):

    permutation = np.random.permutation(y.shape[0])

    shuffled\_x = x[permutation, :]

    shuffled\_y = y[permutation]

    return shuffled\_x, shuffled\_y

def get\_next\_batch(x, y, start, end):

    x\_batch = x[start:end]

    y\_batch = y[start:end]

    return x\_batch, y\_batch

# Load MNIST data

x\_train, y\_train, x\_valid, y\_valid = load\_data(mode='train')

# Hyper-parameters

epochs = 10             # Total number of training epochs

batch\_size = 100        # Training batch size

display\_freq = 100      # Frequency of displaying the training results

learning\_rate = 0.001   # The optimization initial learning rate

def weight\_variable(shape):

    initer = tf.truncated\_normal\_initializer(stddev=0.01)

    return tf.get\_variable('W',

                           dtype=tf.float32,

                           shape=shape,

                           initializer=initer)

def bias\_variable(shape):

    initial = tf.constant(0., shape=shape, dtype=tf.float32)

    return tf.get\_variable('b',

                           dtype=tf.float32,

                           initializer=initial)

sess = tf.InteractiveSession()

# Initialize all variables

sess.run(init)

# Number of training iterations in each epoch

num\_tr\_iter = int(len(y\_train) / batch\_size)

for epoch in range(epochs):

    print('Training epoch: {}'.format(epoch + 1))

    # Randomly shuffle the training data at the beginning of each epoch

    x\_train, y\_train = randomize(x\_train, y\_train)

    for iteration in range(num\_tr\_iter):

        start = iteration \* batch\_size

        end = (iteration + 1) \* batch\_size

        x\_batch, y\_batch = get\_next\_batch(x\_train, y\_train, start, end)

        # Run optimization op (backprop)

        feed\_dict\_batch = {x: x\_batch, y: y\_batch}

        sess.run(optimizer, feed\_dict=feed\_dict\_batch)

        if iteration % display\_freq == 0:

            # Calculate and display the batch loss and accuracy

            loss\_batch, acc\_batch = sess.run([loss, accuracy],

                                             feed\_dict=feed\_dict\_batch)

            print("iter {0:3d}:\t Loss={1:.2f},\tTraining Accuracy={2:.01%}".

                  format(iteration, loss\_batch, acc\_batch))

    # Run validation after every epoch

    feed\_dict\_valid = {x: x\_valid[:1000], y: y\_valid[:1000]}

    loss\_valid, acc\_valid = sess.run([loss, accuracy], feed\_dict=feed\_dict\_valid)

    print("Epoch: {0}, validation loss: {1:.2f}, validation accuracy: {2:.01%}".

          format(epoch + 1, loss\_valid, acc\_valid))

# Test the network after training

# Accuracy

x\_test, y\_test = load\_data(mode='test')

feed\_dict\_test = {x: x\_test[:1000], y: y\_test[:1000]}

loss\_test, acc\_test = sess.run([loss, accuracy], feed\_dict=feed\_dict\_test)

print('---------------------------------------------------------')

print("Test loss: {0:.2f}, test accuracy: {1:.01%}".format(loss\_test, acc\_test))

print('---------------------------------------------------------')

# Appendix IV

**Code for Multi-Layer Perceptron Model**

#importing the data

print('Getting Data')

train=pd.read\_csv(r'C:\Users\13432\Documents\Mod 3\MNIST\mnist\mnist\mnist\_train.csv')

test=pd.read\_csv(r'C:\Users\13432\Documents\Mod 3\MNIST\mnist\mnist\mnist\_test.csv')

#preparing the data

#breaking off the labels seperate from the data so the model doesnt know what it is

print('Preparing data')

train\_label=train['label']

test\_label=test['label']

test.drop(columns=['label'])

train.drop(columns=['label'])

#training the data using sk.learn

print('Training data')

# the numbers indicate the hidden layers, in this case 3 hidden layers of 100 neurons, and a hidden layer #of 5 neurons

mlpc=MLPClassifier(hidden\_layer\_sizes=(100,100,100,5))

#fitting the data

print('fitting data')

mlpc.fit(train,train\_label)

#running the model on the test set

print('Predicting...')

prediction=mlpc.predict(test)

cm=accuracy\_score(test\_label,prediction)

print(cm)

#optional

#

confusion=confusion\_matrix(test\_label,prediction)

report=classification\_report(test\_label, prediction)

# Appendix V

**Initial Code for Convolutional Neural Network**

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train /= 255

x\_test /= 255

print('x\_train shape:', x\_train.shape)

print('Number of images in x\_train', x\_train.shape[0])

print('Number of images in x\_test', x\_test.shape[0])

num\_category = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_category)

y\_test = keras.utils.to\_categorical(y\_test, num\_category)

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=input\_shape))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num\_category, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy,

              optimizer=keras.optimizers.Adadelta(),

              metrics=['accuracy'])

batch\_size = 128

num\_epoch = 10

model\_log = model.fit(x\_train, y\_train,

          batch\_size=batch\_size,

          epochs=num\_epoch,

          verbose=1,

          validation\_data=(x\_test, y\_test))

score = model.evaluate(x\_test, y\_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

**Initial Results for Convolutional Neural Network**

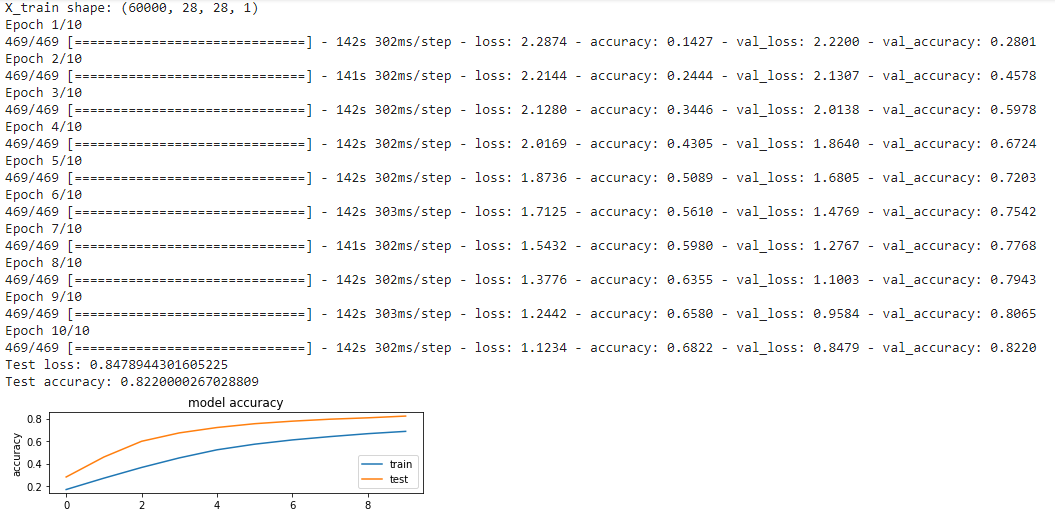


Figure : The results of the original Convolutional Neural Network model