University of West Florida

MAP 6114 Machine Learning

Fall 2021

Handwritten Classification of Characters and Digits Using ANN

Gabriel Rivera-Alvarez

ID: 970607252

Data Science – 12/08/2021

Prof. Mizanoor Rahman, Ph.D.

Table of Contents

[Abstract 2](#_Toc89901167)

[Background and Motivation 3](#_Toc89901168)

[Objectives 5](#_Toc89901169)

[Databases 5](#_Toc89901170)

[Materials and Methods 6](#_Toc89901171)

[Results and Analysis 6](#_Toc89901172)

[Limitations 15](#_Toc89901173)

[Conclusions 15](#_Toc89901174)

[Bibliography 16](#_Toc89901175)

[Appendix I: Jupyter Notebook Code 17](#_Toc89901176)

# Abstract

The area of computer vision has evolved to become an important field in today’s world. It is used in many industries and has many applications. This project consists of using handwritten images of characters and numbers. Characters come in a variation lower-case letters and upper-case letters while numbers go from 0 to 9. The data comes from the EMNIST Dataset. This dataset is split into different classes. From those classes, this project uses EMNIST ByClass, EMNIST Letters, EMNIST Digits, and EMNIST MNIST. The model that will be developed is an Artificial Neural Network that consists of an input layer, two hidden layers and the output layer for a total of 5 layers. Once the models were created and evaluated, they were tested. The models that yielded the lowest accuracies for the testing data were EMNIST ByClass and EMNIST Letters with an accuracy of 0.1667 for both. Both models only classified correctly one of the testing images, while misclassifying the rest. The model based on the EMNIST Digits was the best model with an accuracy of 0.8889 of the testing data and one misclassification. On the other hand, the EMNIST MNIST had only two misclassifications and an accuracy of 0.7778. Using more hidden layers or a different approach like Convolutional Neural Network could improve substantially the results.

# Background and Motivation

The area of computer vision has evolved to become an important field in today’s world. It is used among many different types of industries and has many applications. Computer vision is a subcategory of Artificial Intelligence that focuses on building and using digital systems to process, analyze and interpret visual data. The goal of computer vision is to enable computing devices to correctly identify an object or person in a digital image and take appropriate action [1].

Computer Vision is a field employed by many Electrical Engineers and I feel motivated to do a project related to this topic because my undergraduate studies are in Electrical Engineering. This project consists of using handwritten images of characters and numbers. Characters come in a variation lower-case letters and upper-case letters while numbers go from 0 to 9.

The data comes from the EMNIST Dataset. The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28x28 pixel image format and dataset structure that directly matches the MNIST dataset.

The MNIST dataset has become a standard benchmark for learning, classification and computer vision systems. The MNIST database was derived from a larger dataset known as the NIST Special Database 19 which contains digits, uppercase and lowercase handwritten letters. [2]

There are six different splits provided in this dataset. A short summary of the dataset is provided below: [2]

* EMNIST ByClass: 814,255 characters. 62 unbalanced classes.
* EMNIST ByMerge: 814,255 characters. 47 unbalanced classes.
* EMNIST Balanced:  131,600 characters. 47 balanced classes.
* EMNIST Letters: 145,600 characters. 26 balanced classes.
* EMNIST Digits: 280,000 characters. 10 balanced classes.
* EMNIST MNIST: 70,000 characters. 10 balanced classes.

This project will use the EMNIST ByClass, EMNIST Letters, EMNIST Digits, and EMNIST MNIST. A similar model will be implemented for each one of the mentioned splits and their accuracies will be compared. The model that will be developed is an Artificial Neural Network. Because the model has two hidden layers, it can be considered a Deep Learning Neural Network.

An artificial neural network (ANN) is a system of hardware and/or software patterned after the operation of neurons in the human brain. An ANN usually involves a large number of processors operating in parallel and arranged in layers. The first layer receives the raw input information -- analogous to optic nerves in human visual processing. Each successive layer receives the output from the layer preceding it, rather than the raw input -- in the same way neurons further from the optic nerve receive signals from those closer to it. The last layer produces the output of the system. [3]

Some advantages of artificial neural networks include: [3]

* Parallel processing abilities mean the network can perform more than one job at a time.
* The ability to learn and model nonlinear, complex relationships helps model the real-life relationships between input and output.
* Fault tolerance means the corruption of one or more cells of the ANN will not stop the generation of output.

Some disadvantages of ANNs include: [3]

* The lack of rules for determining the proper network structure means the appropriate artificial neural network architecture can only be found through trial and error and experience.
* The network works with numerical information; therefore, all problems must be translated into numerical values before they can be presented to the ANN.

Figure 1 shows a basic diagram of an ANN. First, we have the input layer that gets the raw data that will be processed. Then, we have the hidden layers that convert the input into something that can be used by the output. Finally, the output layer which consists of the output to the system. Typically, an ANN is initially trained or fed large amounts of data. Training consists of providing input and telling the network what the output should be. [3]

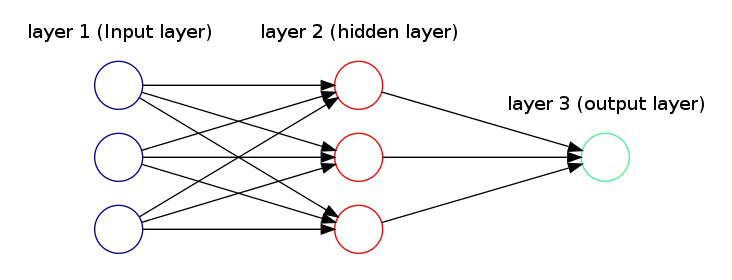


Figure 1. Basic Artificial Neural Network [4]

Normally, we call it ANN when dealing with only one hidden layer. When we have two or more hidden layers, we call that Deep Learning. Therefore, figure 2 is a basic diagram of a Deep Learning because we have 2 hidden layers.

Diagram

Description automatically generated

Figure 2. Artificial Neural Network [5]

Deep learning is a subset of machine learning, which is essentially a neural network with two or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy [6]. In this project a Deep Learning Neural Network will be implemented to develop a model capable of getting an image with a character or a number and it is able to classify it.

# Objectives

As mentioned before, the main goal of this project is to develop a model that capable of classifying input data into characters or numbers. The model will be developed using Artificial Neural Network with two hidden layers or Deep Learning Neural Network. The programming language that will be used to achieve this goal is python. The goals are to get the data, develop a model and train it, and finally test it. An objective of this project is to compare various of the splits of the EMNIST Dataset and compare their accuracy to one another.

# Databases

The main database for this project is the EMNIST Database. It is a very large file of over 500MB of data. Therefore, it will not be physically downloaded into the computer but instead it will be brought to Jupyter Notebooks using the command import. For testing the models, a small dataset was created to verify the accuracy of the models. Bellow are the images that will be used to test the models.



Figure 3. Numeric Dataset for Testing



Figure 4. Character Dataset for Testing

# Materials and Methods

To be able to achieve this project successfully several materials are needed. These materials are:

* Computer
* Python
* Jupiter Notebooks
* EMNIST Dataset
* TensorFlow

The methods, or procedure, that will be implemented is as follows:

1. Import the Data and Libraries that will be used
2. Normalization of the Data
3. Model Development and Compile
4. Model Training
5. Model Evaluation
6. Model Testing

Please keep in mind that more than one split will be used, therefore the first step will be implemented once but steps 2-6 will be implemented multiple times.

# Results and Analysis

The goal is to develop a Deep Learning Neural Network capable of identifying characters and numbers. The full code can be observed in in Appendix I. The first step is to import the EMNIST Database into the environment using the command import. Some other libraries that are also imported that will aid in the development of the models are: TensorFlow, os, cv2, numpy and matplotlib.

Once we have imported the necessary libraries. We start working with the data. We will be using the EMNIST ByClass, EMNIST Letters, EMNIST Digits, and EMNIST MNIST. Each split of the data will be worked in different blocks.

Now, our first interest is to develop a model using the EMNIST ByClass split. First, the training and testing data is extracted. Then, we normalize the data in terms of x\_train and x\_test. We normalize only these values to have the pixels in a range that goes from 0 to 1 and having this is easier for the Neural Network to work with. The y\_test and y\_train is not normalized because it is not the interest to put between 0 and 1 the actual results.

That is the pre-processing of the data. Now, the model is developed. The model implemented from TensorFlow Keras model is the Sequential model which means that each layer of the model will be right after the next. Please refer to figure 2 for a graphical view of what the code is performing. An input layer, two hidden layers, and output layer are used.

The first layer is Flatten which will flat certain input shapes and get the 28x28 images and covert those pixels into a line of 784 lines of pixels for easier computation. The next two layers that are used is the Dense layer which is a basic layer in Neural Networks. Each Dense layer has 256 neurons that use the rectified linear unit (relu) activation function. This is a basic and widely used activation function that gives 0 for negative numbers and grows linearly for positive numbers. The last later also uses a Dense layer but it has as output the amount of classes in EMNIST ByClass. That is, 62 classes. This last layer uses softmax as activation function because then all the outputs will add to 1. The logic behind this is that the highest value will have the highest probability of being the character/number that was the input. For example, if we put a 0 as its input, the neuron 0 of the output should have a higher value compared to the rest if there is no misclassification.

Then, the model is compiled with the adam optimizer, sparse\_categorical\_crossentropy as the loss function and accuracy as metrics. Model was compiled using variations of these, but the ones mentioned yielded the best results. After that, the model is trained and evaluated. Figure 5 shows the result of the training of the model in each epoch. A total of 10 epochs were used to train the model.

Table

Description automatically generated

Figure 5. Training results by Epoch for ByClass Split

From figure 5 it can be seen that as more epochs go by, the loss is decreased, and the accuracy increases. At the end of the training, the loss was around 0.41 and the accuracy of the model 0.85. The evaluation of this model yielded a 0.51 loss and 0.83 accuracy. This is a desired result. Now, the model is tested with the data created and shown in the Database Section.

Probably, this number is: 7 Probably, this letter is: g

A picture containing chart

Description automatically generated A picture containing qr code

Description automatically generated

Probably, this number is: 5 Probably, this letter is: g

Text, logo

Description automatically generated Logo

Description automatically generated with low confidence

Probably, this number is: 1 Probably, this letter is: M

A picture containing icon

Description automatically generated A picture containing chart

Description automatically generated

Figure 6. ByClass Model Testing Results

From figure 6, it can be observed that most of the testing data was misclassified. We tested the model with characters A, b, G, h, l, and M. From those the only one that was classified correctly was M while the rest were classified incorrectly. This is due to the fact that we have many characters that are alike in between them and some characters and numbers are also alike. Therefore, this yielded not the intended results despite the 85% accuracy.

Next step was to develop a model using the EMNIST Letters split. Like the previous model, the training and testing data is obtained and normalized. Then, the model is developed, compiled, fitted, and evaluated using the same format as EMNIST ByClass. In this case, it is observed that the accuracy and losses improved with respect to the ByClass split. In the EMNIST Letters split, the final accuracy of the model is 0.95 and the loss is 0.11. After evaluating the model, the accuracy was 0.90 and the loss was 0.40. Figure 7 shows the result of the training for this model.

Table

Description automatically generated

Figure 7. Training results by Epoch for Letters Split

In figure below, figure 8, we have the results of testing the model. In this case, the only correct classification was the character G, the rest was misclassified. Even though there is no confusion between characters and numbers in this model, the similarities between certain letters, like M and N, produced so many misclassifications.

Probably, this letter is: O Probably, this letter is: I

A picture containing chart

Description automatically generated A picture containing qr code

Description automatically generated

Probably, this letter is: G Probably, this letter is: I

Text, logo

Description automatically generated Logo

Description automatically generated with low confidence

Probably, this letter is: J Probably, this letter is: N

A picture containing icon

Description automatically generated A picture containing chart

Description automatically generated

Figure 8. Letters Model Testing Results

Next, a model using the EMNIST Digits split was developed. Like the previous model, the training and testing data is obtained and normalized. Then, the model is developed, compiled, fitted, and evaluated using the same format as previous models. In this case, it is observed that the accuracy and losses improved with respect to previous two models. In the EMNIST Letters split, the final accuracy of the model is 0.99 and the loss is 0.01. After evaluating the model, the accuracy was 0.98 and the loss was 0.06. Figure 9 show the results of the epoch training.

Table

Description automatically generated

Figure 9. Training results by Epoch for Digits Split

In figure below, figure 10, we have the results of testing the model. In this case, the only incorrect classification was the number 6, which was classified as 5. The rest were classified correctly. Which means that the so far, the digit split has been more accurate. That is, classifying numbers yields better results than classifying characters. There are less similarities between numbers than between characters.

Probably, this digit is: 1 Probably, this digit is: 2

Icon

Description automatically generated A picture containing icon

Description automatically generated

Probably, this digit is: 3 Probably, this digit is: 4

A picture containing chart

Description automatically generated A picture containing chart

Description automatically generated

Probably, this digit is: 5 Probably, this digit is: 5

Logo

Description automatically generated A picture containing text

Description automatically generated

Probably, this digit is: 7 Probably, this digit is: 8

A picture containing chart

Description automatically generated Qr code

Description automatically generated

Probably, this digit is: 9

A picture containing logo

Description automatically generated

Figure 10. Digits Model Testing Results

Lastly, a model using the EMNIST MNIST split was developed. Like the previous model, the training and testing data is obtained and normalized. Then, the model is developed, compiled, fitted, and evaluated using the same format as previous models. In this case, it is observed that the accuracy and losses improved with respect to the first two models and about the same with the Digits model. In the EMNIST MNIST split, the final accuracy of the model is 0.99 and the loss is 0.01. After evaluating the model, the accuracy was 0.98 and the loss was 0.08. Figure 11 show the results of the epoch training.

Table

Description automatically generated

Figure 11. Training results by Epoch for MNIST Split

Figure below, figure 12, we have the results of testing the model. In this case, two incorrect classifications occurred. The number 6 was misclassified as a 5 and the number 7 was misclassified as 1, the rest were classified correctly. Which means that the MNIST split yielded similar results to the Digits Split.

Probably, this digit is: 1 Probably, this digit is: 2

Icon

Description automatically generated A picture containing icon

Description automatically generated

Probably, this digit is: 3 Probably, this digit is: 4

A picture containing chart

Description automatically generated A picture containing chart

Description automatically generated

Probably, this digit is: 5 Probably, this digit is: 5

Logo

Description automatically generated A picture containing text

Description automatically generated

Probably, this digit is: 1 Probably, this digit is: 8

A picture containing chart

Description automatically generated Qr code

Description automatically generated

Probably, this digit is: 9

A picture containing logo

Description automatically generated

Figure 12. MNIST Model Testing Results

To summarize the results please take a look at table 1. It can be seen that the most accurate model was the digits model which only yielded one misclassification. The testing accuracy was calculated diving the correct classifications by the testing size.

Table 1. Models Summary

|  |  |  |
| --- | --- | --- |
| Model | Loss | Accuracy |
| ByClass Trained | 0.4148 | 0.8513 |
| ByClass Evaluated | 0.5183 | 0.8351 |
| ByClass Tested | - | 0.1667 |
| Letters Trained | 0.1193 | 0.9544 |
| Letters Evaluated | 0.4058 | 0.9038 |
| Letters Tested | - | 0.1667 |
| Digits Trained | 0.0138 | 0.9957 |
| Digits Evaluated | 0.0626 | 0.9882 |
| Digits Tested | - | 0.8889 |
| MNIST Trained | 0.0148 | 0.9952 |
| MNIST Evaluated | 0.0839 | 0.9828 |
| MNIST Tested | - | 0.7778 |

# Limitations

The limitations of this project is based on the simplicity of the model. A more complicated model with Convolutional Neural Networks could yield much better results. In addition, another limitation was the amount of hidden layers used and the amount of neurons per layer. For more accurate models, these should be improved but it will be at the cost of time because it would take a long time to train the model. Lastly, another limitation was the type of layer used. Maybe more complex layers could result in a better model. Removing these limitations would improve considerably the models develop. This is left as future work.

# Conclusions

In this project, a handwritten model was created to evaluate the recognition of handwritten characters and numbers. The EMNIST Dataset was used to obtain the handwritten information. From the different splits in EMNIST Dataset, the ones used were: ByClass, Letters, Digits, and MNIST. It was observed that due to similarities within characters and numbers, ByClass and Letters yielded the lowest testing accuracies of 0.1667 in both cases. They were only able to classify correctly one of the six testing images. On the other hand, Digits and MNIST yielded the best accuracies, and they were both similar. The testing accuracies were 0.8889 and 0.7778, respectively. Some limitations were observed like not using a CNN to train the model which was left as future work. Based in all this, it can be concluded that Artificial Neural Networks can be use in the classification of handwritten digits and characters.

# Bibliography

|  |  |
| --- | --- |
| [1] | Techopedia, "Computer Vision," Techopedia, 25 October 2021. [Online]. Available: https://www.techopedia.com/definition/32309/computer-vision. [Accessed 07 December 2021]. |
| [2] | G. Cohen, S. Afshar, J. Tapson and A. van Schaik, "The EMNIST Dataset," The MARCS Institute for Brain, Behaviour and Development , [Online]. Available: https://www.nist.gov/itl/products-and-services/emnist-dataset. [Accessed 07 December 2021]. |
| [3] | E. Burns, "What is a neural network? Explanation and examples," TechTarget, [Online]. Available: https://searchenterpriseai.techtarget.com/definition/neural-network. [Accessed 07 December 2021]. |
| [4] | T. G. Martins, "How to draw neural network diagrams using Graphviz," 12 June 2013. [Online]. Available: https://tgmstat.wordpress.com/2013/06/12/draw-neural-network-diagrams-graphviz/. [Accessed 07 December 2021]. |
| [5] | L. Dormehl, "What is an artificial neural network? Here’s everything you need to know," EMERGING TECH, 05 January 2019. [Online]. Available: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/. [Accessed 07 December 2021]. |
| [6] | IBM Cloud Education, "Deep Learning," IBM, 01 May 2020. [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning. [Accessed 07 December 2021]. |

# Appendix I: Jupyter Notebook Code

*# Import Libraries to be Used*

**from** emnist **import** extract\_training\_samples

**from** emnist **import** extract\_test\_samples

**import** tensorflow **as** tf

**import** os

**import** cv2

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

*# SPLIT: BYCLASS*

*# Import the Data to be Used*

x\_train\_byclass, y\_train\_byclass **=** extract\_training\_samples('byclass')

x\_test\_byclass, y\_test\_byclass **=** extract\_test\_samples('byclass')

*# Normalize the data*

x\_train\_byclass **=** tf**.**keras**.**utils**.**normalize(x\_train\_byclass, axis **=** 1)

x\_test\_byclass **=** tf**.**keras**.**utils**.**normalize(x\_test\_byclass, axis **=** 1)

*# Model Development*

model\_byclass **=** tf**.**keras**.**models**.**Sequential()

model\_byclass**.**add(tf**.**keras**.**layers**.**Flatten(input\_shape **=** (28, 28)))

model\_byclass**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_byclass**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_byclass**.**add(tf**.**keras**.**layers**.**Dense(62, activation**=**'softmax'))

model\_byclass**.**compile(optimizer **=** 'adam', loss **=** 'sparse\_categorical\_crossentropy', metrics **=** ['accuracy'])

*# Train the Developed Model*

model\_byclass**.**fit(x\_train\_byclass, y\_train\_byclass, epochs **=** 10)

*# Model Evaluation*

loss\_byclass, accuracy\_byclass **=** model\_byclass**.**evaluate(x\_test\_byclass, y\_test\_byclass)

*# Model Testing: ByClass*

letter\_array **=** ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','V','W','X','Y','Z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

image\_letter **=** ['A','b','G','h','l','M']

image\_number **=** 0

**while** os**.**path**.**isfile(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Letters/{image\_letter[image\_number]}.png"):

**try**:

img **=** cv2**.**imread(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Letters/{image\_letter[image\_number]}.png")[:,:,0]

img **=** np**.**invert(np**.**array([img]))

prediction **=** model\_byclass**.**predict(img)

letter\_number **=** np**.**argmax(prediction)

**if** letter\_number **<=** 10:

print('Probably, this number is: ',letter\_array[letter\_number])

**else**:

print('Probably, this letter is: ',letter\_array[letter\_number])

plt**.**imshow(img[0], cmap **=** plt**.**cm**.**binary)

plt**.**show()

**finally**:

image\_number **+=** 1

*# SPLIT: LETTERS*

*# Import the Data to be Used*

x\_train\_letters, y\_train\_letters **=** extract\_training\_samples('letters')

x\_test\_letters, y\_test\_letters **=** extract\_test\_samples('letters')

*# Normalize the data*

x\_train\_letters **=** tf**.**keras**.**utils**.**normalize(x\_train\_letters, axis **=** 1)

x\_test\_letters **=** tf**.**keras**.**utils**.**normalize(x\_test\_letters, axis **=** 1)

*# Model Development*

model\_letters **=** tf**.**keras**.**models**.**Sequential()

model\_letters**.**add(tf**.**keras**.**layers**.**Flatten(input\_shape **=** (28, 28)))

model\_letters**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_letters**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_letters**.**add(tf**.**keras**.**layers**.**Dense(27, activation**=**'softmax'))

model\_letters**.**compile(optimizer **=** 'adam', loss **=** 'sparse\_categorical\_crossentropy', metrics **=** ['accuracy'])

*# Train the Developed Model*

model\_letters**.**fit(x\_train\_letters, y\_train\_letters, epochs **=** 10)

*# Model Evaluation*

loss\_letters, accuracy\_letters **=** model\_letters**.**evaluate(x\_test\_letters, y\_test\_letters)

*# Model Testing: Letters*

letter\_array **=** ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','V','W','X','Y','Z']

image\_letter **=** ['A','b','G','h','l','M']

image\_number **=** 0

**while** os**.**path**.**isfile(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Letters/{image\_letter[image\_number]}.png"):

**try**:

img **=** cv2**.**imread(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Letters/{image\_letter[image\_number]}.png")[:,:,0]

img **=** np**.**invert(np**.**array([img]))

prediction **=** model\_letters**.**predict(img)

letter **=** np**.**argmax(prediction)

print('Probably, this letter is: ',letter\_array[letter])

plt**.**imshow(img[0], cmap **=** plt**.**cm**.**binary)

plt**.**show()

**finally**:

image\_number **+=** 1

*# SPLIT: DIGITS*

*# Import the Data to be Used*

x\_train\_digits, y\_train\_digits **=** extract\_training\_samples('digits')

x\_test\_digits, y\_test\_digits **=** extract\_test\_samples('digits')

*# Normalize the data*

x\_train\_digits **=** tf**.**keras**.**utils**.**normalize(x\_train\_digits, axis **=** 1)

x\_test\_digits **=** tf**.**keras**.**utils**.**normalize(x\_test\_digits, axis **=** 1)

*# Model Development*

model\_digits **=** tf**.**keras**.**models**.**Sequential()

model\_digits**.**add(tf**.**keras**.**layers**.**Flatten(input\_shape **=** (28, 28)))

model\_digits**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_digits**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_digits**.**add(tf**.**keras**.**layers**.**Dense(10, activation**=**'softmax'))

model\_digits**.**compile(optimizer **=** 'adam', loss **=** 'sparse\_categorical\_crossentropy', metrics **=** ['accuracy'])

*# Train the Developed Model*

model\_digits**.**fit(x\_train\_digits, y\_train\_digits, epochs **=** 10)

*# Model Evaluation*

loss\_digits, accuracy\_digits **=** model\_digits**.**evaluate(x\_test\_digits, y\_test\_digits)

*#Model Testing: Digits*

image\_number **=** 1

**while** os**.**path**.**isfile(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Digits/{image\_number}.png"):

**try**:

img **=** cv2**.**imread(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Digits/{image\_number}.png")[:,:,0]

img **=** np**.**invert(np**.**array([img]))

prediction **=** model\_digits**.**predict(img)

print('Probably, this digit is: ',np**.**argmax(prediction))

plt**.**imshow(img[0], cmap **=** plt**.**cm**.**binary)

plt**.**show()

**except**:

print('Error!')

**finally**:

image\_number **+=** 1

*# SPLIT: MNIST*

*# Import the Data to be Used*

x\_train\_mnist, y\_train\_mnist **=** extract\_training\_samples('mnist')

x\_test\_mnist, y\_test\_mnist **=** extract\_test\_samples('mnist')

*# Normalize the data*

x\_train\_mnist **=** tf**.**keras**.**utils**.**normalize(x\_train\_mnist, axis **=** 1)

x\_test\_mnist **=** tf**.**keras**.**utils**.**normalize(x\_test\_mnist, axis **=** 1)

*# Model Development*

model\_mnist **=** tf**.**keras**.**models**.**Sequential()

model\_mnist**.**add(tf**.**keras**.**layers**.**Flatten(input\_shape **=** (28, 28)))

model\_mnist**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_mnist**.**add(tf**.**keras**.**layers**.**Dense(256, activation**=**'relu'))

model\_mnist**.**add(tf**.**keras**.**layers**.**Dense(10, activation**=**'softmax'))

model\_mnist**.**compile(optimizer **=** 'adam', loss **=** 'sparse\_categorical\_crossentropy', metrics **=** ['accuracy'])

*# Train the Developed Model*

model\_mnist**.**fit(x\_train\_mnist, y\_train\_mnist, epochs **=** 10)

*# Model Evaluation*

loss\_mnist, accuracy\_mnist **=** model\_mnist**.**evaluate(x\_test\_mnist, y\_test\_mnist)

*# Model Testing: Mnist*

image\_number **=** 1

**while** os**.**path**.**isfile(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Digits/{image\_number}.png"):

**try**:

img **=** cv2**.**imread(f"/Users/gabrielrivera/Documents/Maestria/Courses/2- Fall 2021/MAP6114/Project/Digits/{image\_number}.png")[:,:,0]

img **=** np**.**invert(np**.**array([img]))

prediction **=** model\_mnist**.**predict(img)

print('Probably, this digit is: ',np**.**argmax(prediction))

plt**.**imshow(img[0], cmap **=** plt**.**cm**.**binary)

plt**.**show()

**except**:

print('Error!')

**finally**:

image\_number **+=** 1