MNIST DIGIT RECOGNIZER WITH BIG DATA SCIENCE ALGORITHMS

MNIST DIGIT RECOGNIZER WITH BIG DATA SCIENCE ALGORITHMS

A project report submitted in partial

fulfillment of the requirements for the degree of

Master of Science

By

Mien Nguyen

National Universities of Viet Nam, Natural Science College

Bachelor of Science. in Computer Science

April 2020

University of Colorado Denver

ABSTRACT

There are many algorithms that can be used to explore the data to yield the output such as prediction, classification, clustering. For a certain set of data (Ex: set of images, set of selling houses, set of tumor sizes, set of digits, ect.), and for a certain expected output (Ex: identify objects in the images, predict house selling price, classify if the tumor is benign or malignant, so on) how can one pick an efficient algorithm. Certainly, it all bases on experience of the data scientist. However, before becoming a data scientist, everybody needs some knowledge.

This project aims to help students/fresh data scientists to understand and apply MANY big data science algorithms on ONE dataset. Subsequently it will let him know which algorithm works best for the dataset to gain the expected output, and why the other algorithms don’t work.

This Project Report is approved for recommendation to the Graduate Committee.

Project Advisor:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Farnoush Banaei-kashani

©2020 by Mien H. Nguyen

All Rights Reserved

TABLE OF CONTENTS

[1. Introduction 1](#_Toc38487216)

[1.1 Problem 1](#_Toc38487217)

[1.2 Project Statement 1](#_Toc38487218)

[1.3 Approach 1](#_Toc38487219)

[1.4 Organization of this Project Report 2](#_Toc38487220)

[2. Background 3](#_Toc38487221)

[2.1 Key Concepts 3](#_Toc38487222)

[2.1.1 Key Concept 1 3](#_Toc38487223)

[2.1.2 Key Concept 2 3](#_Toc38487224)

[2.1.3 Key Concept 3 4](#_Toc38487225)

[2.2 Related Work or Literature Review 5](#_Toc38487226)

[2.2.1 Area 1 5](#_Toc38487227)

[2.2.2 Area 2 5](#_Toc38487228)

[2.2.3 Area 3 5](#_Toc38487229)

[2.2.4 Area 4 5](#_Toc38487230)

[2.2.5 Area 5 6](#_Toc38487231)

[3. Architecture (or Approach or Model …) 7](#_Toc38487232)

[3.1 High Level Design 7](#_Toc38487233)

[3.2 Implementation 7](#_Toc38487234)

[3.2.1 – Linear Regression 7](#_Toc38487235)

[3.2.2 – Logistic Regression 8](#_Toc38487236)

[3.2.3 – SSL Graph-based 10](#_Toc38487237)

[3.2.4 – CNN 11](#_Toc38487238)

[3.2.5 – Android - Tensorflow Lite 14](#_Toc38487239)

[4. Methodology, Results and Analysis (or similar title) 16](#_Toc38487240)

[4.1 Test Methodology 16](#_Toc38487241)

[4.1.1 – Linear Regression 16](#_Toc38487242)

[4.1.2 - Logistic Regression 16](#_Toc38487243)

[4.1.3 - SSL Graph-based 16](#_Toc38487244)

[4.1.4 – CNN 16](#_Toc38487245)

[4.1.5 – Android 17](#_Toc38487246)

[4.2 Result 17](#_Toc38487247)

[4.2.1 – Linear Regression 17](#_Toc38487248)

[4.2.2 – Logistic Regression 17](#_Toc38487249)

[4.2.3 – SSL Graph-based 18](#_Toc38487250)

[4.2.4 – CNN 18](#_Toc38487251)

[4.2.5 – Android 19](#_Toc38487252)

[4.3 Analysis 20](#_Toc38487253)

[4.3.1 – Linear Regression 20](#_Toc38487254)

[4.3.2 – Logistic Regression 21](#_Toc38487255)

[4.3.3 – SSL Graph-based 21](#_Toc38487256)

[4.3.4 – CNN 22](#_Toc38487257)

[4.3.5 – Apply CNN to Android 23](#_Toc38487258)

[5. Conclusions 24](#_Toc38487259)

[5.1 Summary 24](#_Toc38487260)

[5.2 Potential Impact 24](#_Toc38487261)

[5.3 Future Work 25](#_Toc38487262)

[References 26](#_Toc38487263)

[Appendix A 27](#_Toc38487264)

LIST OF FIGURES

Figure 1: Caption for Figure 1 via Insert/Reference/Caption 4

Figure 2: Caption for Figure 2 5

# 1. Introduction

## 1.1 Problem

A digit can be written in different ways, some were written clear and straight, some were more italic, bold ink, so on, depend on the human writing style. Although it will not always true, a human can easily regcognize if a hand written image is a number 2 or number 7 so on, it is not that simple for a computer to know what digit was that hand written image until we train the computer.

Same thought with above, a computer would not be able to know if the picture was a cat or a dog or if it has a human in it. In general, it is trivial for a man to recognize the objects in an image but it is not always trivial for a computer until human trains the computer.

## 1.2 Project Statement

Train the computer so that (after being trained), giving an image of a digit (an image of 0, 1, … ,9), the system is able to tell the user what the digit is (if the image was number 0 or number 1,… or number 9), with some percentage of accuracy.

## 1.3 Approach

Train the computer to recognize digits on the MNIST dataset, which will be described in Key Concept 1, using many Machine Learning, Deep Learning models, such as:

* Linear Regression
* Logistic Regression
* Graph-based SSL
* Convolutional Neural Network

The model which has highest accuracy will be used/applied to an Android application to recognize the digit in an image.

## 1.4 Organization of this Project Report

Chapter 2 covers background of the project including key comcepts and related academic research.

Chapter 3 describes the high-level architecture of the project, and detailed implementation.

Chapter 4 describes the Test Methodology, Results and Analysis.

Chapter 5 summarises the project, potential impact and future work.

# 2. Background

## 2.1 Key Concepts

### 2.1.1 Key Concept 1

MNIST ("Modified National Institute of Standards and Technology") is the de facto “hello world” dataset of computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike. [1]

Each image is a matrix of 28x28 pixels in black and white color.

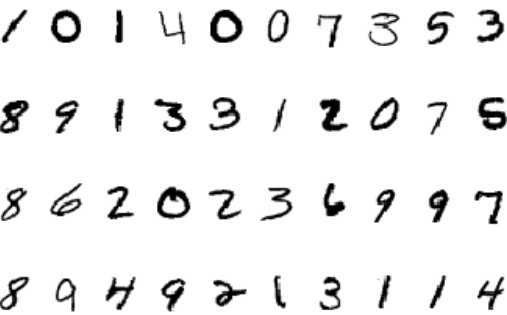


Figure 1: Hand-written digits

### 2.1.2 Key Concept 2

The input and output below are applied to all machine learning algorithms/models within this report.

Input: The 70,000 image samples, which is splitted in training set (42,000 images) and test set (28,000 images) in each algorithm.

Output: A model which can classify if an digit image is number 0, 1,… or 9.

### 2.1.3 Key Concept 3

CAPTCHA as an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart". Many registration forms use Captcha to prevent a robot to massively register to their application/website, subsequently break the application/website.

A human can easily recognize the number or letter in a gizzling image, but it is not trivial for a computer to do so (well, back then). If a "machine" can solve Captcha, Captcha does not really do its job. This project introduces a little about how a machine can break Captcha, in term of digit recognizer. The sample below shows you an alphabet Captcha.

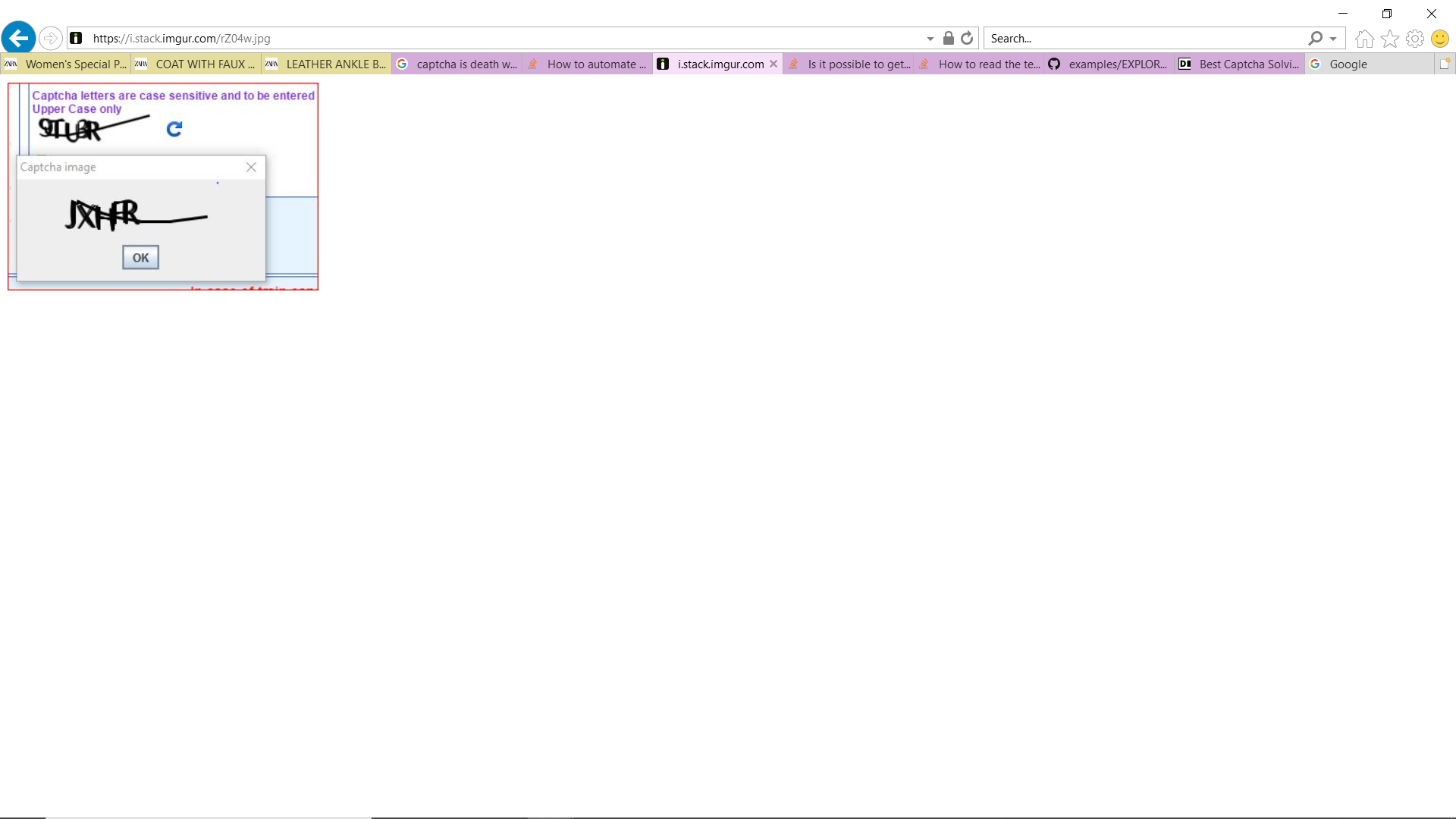


Figure 2: A sample Captcha

## 2.2 Related Work or Literature Review

### 2.2.1 Area 1

Big Data Science and Machine Learning course are taught in the University of Colorado Denver. This project refers to cited papers and materials that are provided during these two courses.

Big Data Science – Dr. Farnoush Banaei-kashani, UC Denver

Machine Learning – Dr. [Ashis Biswas](https://ucdenver.instructure.com/courses/405028/users/254721), UC Denver

### 2.2.2 Area 2

Machine Learning (ML) and Deep Learning (DL) models are applied in this project. ML and DL have been taught in many courses in the universities all over the world but one of the most mathematical oriented and intuitive courses was from professor Andrew Ng. – “Machine Learning — Andrew Ng, Stanford University [FULL COURSE]” [2].

### 2.2.3 Area 3

Kaggle has been posting a competition to recognize MNIST dataset and there are many successful high accuracy results posted in Kaggle. The “Introduction to CNN Keras – 0.997 (top 6%)” was on of the useful source for Convolutional Neural Network. [3]

### 2.2.4 Area 4

Convolutional Neural Network (CNN) model is the most effective image classifer so far, it was studied, described in many researches but the work in youtube from Edureka! team more visualized and well explained the intuition of an image recognizer in term of a CNN [4].

### 2.2.5 Area 5

Tensorflow team introduced a method (Tensorflow Lite) to help integrate a machine learning model to an IoT device such as Android or Apple phone.

It enables on-device machine learning inference with low latency and a small binary size. TensorFlow Lite consists of two main components [5]:

* The [TensorFlow Lite interpreter](https://www.tensorflow.org/lite/guide/inference), which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.
* The [TensorFlow Lite converter](https://www.tensorflow.org/lite/convert/index), which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

# 3. Architecture

## 3.1 High Level Design

MNIST dataset will be trained and tested through the models such as Linear Regression, Logistic Regression, SSL Graph\_based model, Convolutional Neural Network (CNN). Each model will have its own accuracy. The highest accuracy model will be used to recognize a digit image which is drawn in an Android application.

A screenshot of a cell phone

Description automatically generated

Figure 3: High level design

## 3.2 Implementation

### 3.2.1 – Linear Regression

1. Description

Linear Regression is a supervised learning. This project uses Batch Gradient Descent – one of the algorithms in Linear Regression, along with Close Form, Stochastic Gradient Descent, mini-batch Gradient Descent – to minimize the cost function and subsequently yield the optimal parameters for the linear model.

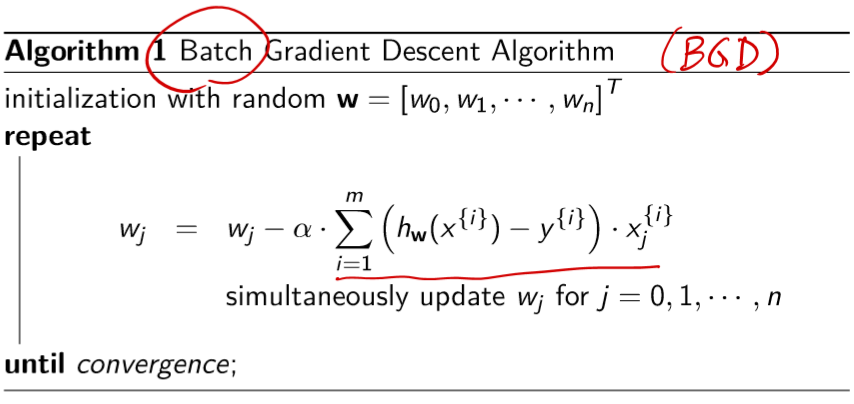


Figure 4: Batch Gradient Descent

1. Language: Python 3 on Anaconda base environment
2. Tools: Panda, Numpy
3. Code: The Batch Gradient Desccent is coded by Close Form, no library used

### 3.2.2 – Logistic Regression

1. Description:

Logistic Regression is a classification method. Like regression, classification is also a supervised learning. In this MNIST dataset, we expect many Y - “label”, it is the multi-class classification or multinomial classification, different from “binary” classification (Ex: The tumor is malignant: 1 or not - 0). We need to classify 10 classes:

Class 0: The image is for digit “0”

Class 1: // “1”

…

Class 9: The image is for digit “9”

When the new sample X come in, we just need to run the X though 10 classifiers. The classifier gives the maximum probability will be the decision classifier.

To learn the classifier for class 0: Learn the boundary to separate “class 0” from “not class 0”.

* Create a “fake” dataset, if the Y = 0 (digit “0”) then the sample is assigned to “class 0” - positive, otherwise, the sample is assigned to “not class 0” – negative.
* Use Logistic Regression method, run through the new training set, we will have a hypothesis function: (P(Y=0|(X,W)).

Do the same process for the classifier ofclass 1, class2, … class 9.

1. Language: Python 3 on Anaconda base environment
2. Tools: Panda, Numpy, Scikit-learn
3. Code: Use Scikit-learn Multinomial Logistic Regression (MLR) model as steps below
   1. Load data and process the data

Load and split the MNIST training dataset into X\_train, Y\_train, X\_val, Y\_val

* 1. Define the model

Create one-vs-rest logistic regression object

* 1. Fit the model

Fit the model on the training dataset (X\_train)

* 1. Predict the labels

Predict the labels (Y\_val\_predicted) for the validation dataset (X\_val)

* 1. Plot Confusion Matrix

Compute the confusion matrix and plot the confusion matrix on Y\_val and Y\_val\_predicted. The confusion matrix tells us how many samples were predicted right and how many samples were predicted wrong.

* 1. End.

### 3.2.3 – SSL Graph-based

1. Desciption

Graph-based SSL main idea follow steps below

1. Construct graph G with edges between very similar samples (both Labeled and Unlabeled). kNN graph is used in this project.
2. Unlabeled data can help to glue the objects of the same class together.
3. Run a graph partitioning algorithm to separate graph into pieces

The similarity between samples is represented by the thickness of the edges. The process of construction of the graph involves two stages (Cheng et al., 2010). The initial phase involves graph adjacency matrix construction and second phase deals with graph weight calculation.

The graph adjacency matrix is constructed either by the k-nearest neighbor or by ε nearest neighbor. For calculation of graph weights, one of the following equations is used.

1. The Gaussian similarity function which is represented as

g() = exp(- ) where σ controls the width of the neighborhoods.

1. The inverse Euclidean distance function is given as

g() = , … with normalized graph Laplacian, more complication.

### 3.2.4 – CNN

1. Description: There are many lectures, articles describe CNN, we do not explain CNN theory in this project but implement it.
2. Language: Python 3 on Anaconda tensorflow environment
3. Tools: Panda, Numpy, Scikit-learn, tensorflow
4. Code: The real code use Tensorflow Convolutional Neural Network (CNN) model
5. Load data and process the data

Load and split the MNIST training dataset into X\_train, Y\_train, X\_val, Y\_val

1. Define the model

Below is the recommended CNN architecture, figure 4. The first two convolutional layers (Conv2D) was set 32 filters, the other convolutional layer was 64 filters. The pooling layer (MaxPool2D) has a size of (2,2). Dropout is a regulization method, where a proportion of nodes in a layer is randomly ignored. As shown below, two convolutional layers are combined with a pooling layer and a dropout layer, allow the neural network to use ‘local features’ to learn more about ‘global features’ of the image.

The Flatten layer is to convert the final feature into a single 1D vector.

Next is a fully-connected layer (Dense) which is an Artificial Neural Network classifier.

The last layer is a fully-connected layer (Dense(10, activation = ‘softmax’)) will give the outputs of distributed probalitities of each class.

1. Define the optimizer

After defining the CNN model, the most important step is to define the optimizer, which is RMSprop. We could also use Stochastic Gradiend Descent (SGD)

1. Define the metric function (evaluate performance of the model)

The metric function is used here is crossentropy.

1. Compile the model

After the model, optimizer and metric function are designed, the model will be compiled.

1. Data augmentation

In order to avoid overfitting, some data transformation is introduced. The idea is to alter training data with small transformation to reproduce the variations when someone is writing a digit. Approach that alter the training data in ways that change the array representation while keeping the labels is known as data augmentation techniques. Some popular techniques are grayscales, horizontal flips, vertical flips, random crops, color jitter, translation, rotation and much more.

According to the Kaggle competion [3]:

* The model without augmentation got an accuracy of 98.114%
* The model with augmentation got an accuracy of 99.67%

1. Fit the model

Once the model is ready, we fit the model on the training (X\_train) dataset.

A screenshot of a social media post

Description automatically generated

Figure 5: CNN model

1. Predict the labels

Predict the values (Y\_val\_predicted) for the validation dataset (X\_val)

1. Plot the confusion matrix

Compute the confusion matrix and plot the confusion matrix on Y\_val and Y\_val\_predicted. The confusion matrix tells us how many samples were predicted right and how many samples were predicted wrong.

1. Plot the loss and accuracy curves for training dataset and validation dataset.
2. End.

### 3.2.5 – Android - Tensorflow Lite

1. Description: Tensorflow Lite website explains Tensorflow lite in details. Within this project, we do not discuss what the Tensorflow lite is but implement it.
2. Language: Python 3 on Anaconda tensorflow environment, Java on Android environment
3. Tools: Tensorflow Lite
4. Code: Use Tensorflow Lite as defined in steps below
5. Convert the model

Convert the machine leaning model that was created above into a Tensorflow Lite format model, using Tensorflow Converter. In this project, the CNN model is selected (over Multinomial Linear Regression or Graph-based SSL) to convert to a Tensorflow Lite model. This is coded via Python.

1. Deploy the model to the device

The Tensorflow lite model that was converted above will be stored, load and applied in an Android application using Tensorflow Interpreter. This step is coded in Android environment.

A screenshot of a cell phone

Description automatically generated

Figure 6: Tensorflow Lite architecture

# 4. Methodology, Results and Analysis

## 4.1 Test Methodology

### 4.1.1 – Linear Regression

Linear Regression is not applicable for digit recognizer.

### 4.1.2 - Logistic Regression

The MNIST training dataset (42,000 samples) was splitted into training and validation datasets X\_train, Y\_train, X\_val, Y\_val. The Multinomial Logistic Regression model is trained on X\_train and is validated on the X\_val.

### 4.1.3 - SSL Graph-based

In theory, SSL Graph-based method could work for digit recognizer but it fails in reality due to the large dimension. SSL Graph-based requires to construct a kNN graph of labeled and unlabeled samples, = (42000 + 28000) = 70,000 samples, which can not be done by a regular computer.

### 4.1.4 – CNN

The MNIST training dataset (42,000 samples) was splitted into training and validation datasets X\_train, Y\_train, X\_val, Y\_val. The CNN model is trained on X\_train and is validated on the X\_val.

### 4.1.5 – Android

The CNN model is tested on an Android application. Giving an image of a hand-written digit, the application needs to tell the user which digit the image presents for.

## 4.2 Result

### 4.2.1 – Linear Regression

Linear Regression is not applicable for digit recognizer.

### 4.2.2 – Logistic Regression

As in the confusion matrix below, we can see the implemented Multinomial Logistic Regression (MLR) model performs very well on all digits with a few errors considering the size of the validation dataset (4200 images).

However, in comparison to the CNN model which will be described in the next sections, MLR model introduces more errors, we can compare the two Confusion Matrix.

A picture containing keyboard

Description automatically generated

Figure 7: Multinomial Logistic Regression Confusion Matrix

### 4.2.3 – SSL Graph-based

This method requires to construct a kNN graph with weights. It is a graph of labeled and unlabeled samples, = (42000 + 28000) = 70,000 samples

The adjacent matrix of 70,000 \* 70,000 = 4,900,000,000 elements exceeds the capability of a python list on a 32 bits system, which only can handle 536,870,912 elements. Therefore, constructing a kNN graph with weights fails on a regular computer.

### 4.2.4 – CNN

The model reached almost 99% on the validation dataset after 2 epochs. The validation accuracy is higher than training accuracy almost every time during the training. That means our model does not overfit the training dataset. CNN model is very well-trained with the design depicted in the implementation section above.

As in the confusion matrix below, we can see the implemented CNN model performs very well on all digits with a few errors considering the size of the validation dataset (4200 images).

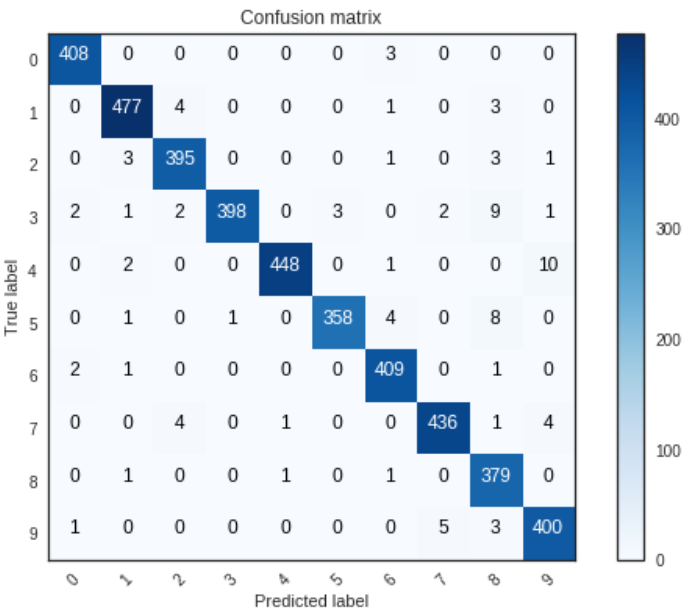


Figure 8: CNN Confusion matrix

### 4.2.5 – Android

The CNN model is implemented successfully in Android and able to recognize a hand-written digit, with some accuracy/confidence as the fugure below.

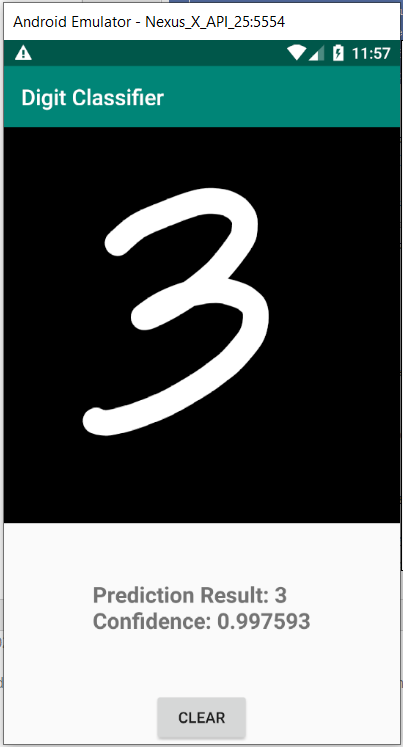


Figure 9: CNN in Android

## 4.3 Analysis

### 4.3.1 – Linear Regression

Linear Regression is a prediction model that the relationship between the “related parameter” (the “y”) and “unrelated parameter” (the “x”) is linear whereas the relationship of the lable (“y”) and the pixel (“x”) of the image is nonlinear. A linear relationship of “y” and “x” can be expressed as “y depends on some percentage of x” while the lable and the pixel of an image sample cannot be expressed so. Thus, the Linear Regression should not be applied. Moreover, as its name, Linear Regression is a prediction model while our task here is pretty much classification.

Another advice that is worth to mention that for a certain dataset, after shuffling the parameters of the dataset one still can recognize each parameter, for example, he still knows which parameter is the house size and which parameter is the house square footage, then the Linear Regression can be applied. In the MNIST dataset, after shuffling the pixels, we do not know which pixel is at which position previously.

### 4.3.2 – Logistic Regression

Logistic Regression or Multinomial Logistic Regression is a supervised learning to classify dataset into classes, therefore it can be applied to classify the digits. The implementation and result for Logistic Regression has been proved so and it does quite a good job.

However, MLR is still less than CNN in term of performance, i.e CNN gives less errors than MLR. Similar with the analysis of Linear Regression above, it has some suggestions that MLR is a classification method for linear parameters and CNN is a classification method for nonlinear parameters which works better for the MNIST dataset. However, there was not a proof for such that suggestion.

### 4.3.3 – SSL Graph-based

Apart from the difficulty of constructing a kNN graph that was described above, to determine the nearest neighbor of a new data point x, we must compute the distance to all m training examples which is expensive and slow, running time is O(mn) = (4,900,000,000 samples x 784 dimensions), where m is the number of samples, and n is the dimension of each sample.

The current semi-supervised learning methods have not yet handled large amount of data. The complexity of many elegant graph-based methods is close to O(n3). Speed-up improvements have been proposed (Mahdaviani et al. 2005; Delalleau et al. 2005; Zhu and Lafferty 2005; Yu et al. 2005; Garcke and Griebel 2005; and more), their effectiveness has yet to be proven on real large problems.

### 4.3.4 – CNN

Let us look at some errors. As the figure below, the model does not behave very unreasonable. Some errors can be human mistakes, especially ones for 4 are very closed to 9. The last 9 was very misleading to, it can be 0.

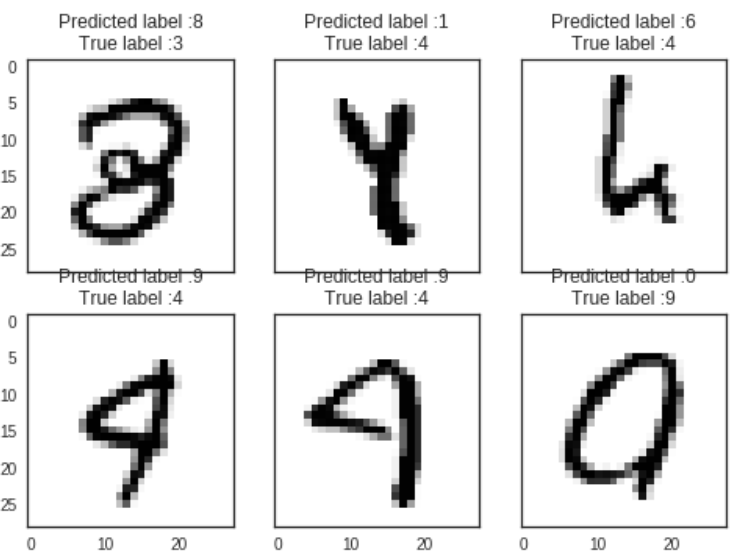


Figure 10: Some Errors

### 4.3.5 – Apply CNN to Android

The result in the Test section for Android shows that the CNN model to recognize digit images was applied to an Android application sucessfully. Thinking about Captcha, a security model that has been embeded to most of web registration forms to prevent a robot breaking the website’s database by registrating a large numbers of accounts, a hacker can use CNN model to recognize the Captcha digits (or images) which are asked during the website registration. Meaning, the hacker can register as many accounts as he wants as soon as he can teach the computer to recognize Captcha. Then, Captcha process is death by CNN. In fact, Captcha was reported death a long time ago by some human service that contains many human working non-stop at computers, remotely, receive Captcha from requesters and respond the answers back to the requesters [6].

In machine learning time, Captcha could die by CNN.

# 5. Conclusions

## 5.1 Summary

The project presents many machine learning models to recognize/classify the MNIST dataset which is a foundation step in computer vision science. Although not all models in this project go through all of the steps of “creating the model”, “testing the model” to the final step of “applying the model” (only CNN model has gone though this path), the project report has shown audiences the following:

* An idea of which machine learning models are “applicable” or “not applicable” for the MNIST dataset (or other images dataset). Linear Regression is “not applicable”, Multinomial Logistic Regression is “applicable” and does quite a good job but less accurate than Convolutional Neural Network. The Graph-based SSL is “applicabe” to classify MNIST in theory but is not coded in this project as building a graph of labled and unlabled samples is over the capacity of a regular computer.
* An idea of how the machine learning model can be applied in real life. This project applies CNN to Android but it does not limit to just Android, people can apply any model to iPhone or other IoT devices.

## 5.2 Potential Impact

As discussed in the Analysis - Apply CNN to Android section above, CNN can break Captcha. Within this project – recognize digits – it is easy to see CNN can break “digit” Captcha, then CNN can break “alphabet” Captcha, and then break “image” Captcha. Meaning the machine learning models are a big threat to the security process.

## 5.3 Future Work

This project stops at the “apply CNN to Android”, it has not implemented the real Captcha security process, which is out of scope of this project due to the time constrainst. Therefore, the next steps will be applying the CNN to break the real “digit” Captcha security system. The implementation will have a lot to do with pre-processing the input image to make it compatible with what CNN requires.

After that, it will be applying the CNN to break the real “alphabet” Captcha, and “image” Captcha and so on. Eventually, hopefully, we will have a complete platform to break all types of Captcha.

# References

[1] “Kaggle Digit Recognizer competion” - <https://www.kaggle.com/c/digit-recognizer/overview>

[2] Andrew Ng - “Machine Learning — Andrew Ng, Stanford University [FULL COURSE]” - <https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN>

[3] Yassine Ghouzam, “Introduction to CNN Keras - 0.997 (top 6%)” - <https://www.kaggle.com/yassineghouzam/introduction-to-cnn-keras-0-997-top-6>

[4] Edureka, “Convolutional Neural Network (CNN) | Convolutional Neural Networks With TensorFlow” - <https://www.youtube.com/watch?v=umGJ30-15_A>

[5] Tensorflow.org, “Android quickstart” - <https://www.tensorflow.org/lite/guide/android>?

[6] DeathByCaptcha website - <https://www.deathbycaptcha.com/user/login>

# Appendix A

This is the final page of a Project Report and should be a blank page