IT 298
FALL 2016
PROJECT REPORT

HANDWRITTEN DIGIT RECOGNIZER USING MACHINE LEARNING

By

Dawood Khan Mohammed
ID. No. 109155286

Under the guidance of
Dr. Daming Zhang

Dept. of Industrial Technology

Jordan College of Agricultural Sciences and Technology,

California State University, Fresno.

Abstract

Today, hand written recognition is a kind of important challenge in the technology industry. Handwriting recognition has many applications and variety of uses. For example, it is used in the mobile settings of mobile phones or tablets, where writing with a stylus or fingers makes us more convenient when compared to typing on the keypad. Handwritten digits are a common part of everyday life, one such kind is that zip codes. It is known that, usually a zip code consist of 5 digits, in some cases it might be more depending upon the trailing digits are included or not. It is also most important that the letter to be delivered to the correct location or destination. Few years back the postman used to read the zip code manually to deliver it to the specific location [1]. However due to the advancement in the technology and Artificial Intelligence, this type of work is done automated by using the Optical Character Recognition (OCR) and such a kind of similar type of solution is implemented in this project with the Machine Learning. The aim of this project is to take an image of a handwritten single digit, and then find out what that digit is using machine learning algorithms. The data for this project is taken from the MNIST datasets from Kaggle data science competition website, the MNIST ("Modified National Institute of Standards and Technology") dataset is a classic within the Machine Learning community and traditionally used for many research and testing purpose of the new machine learning algorithms. In this project, tensor flow library function is used for predicting the handwritten digits, the MNIST train data and test data is used, the algorithm used the train data to predict the test data.

Keywords: Machine Learning, Digit Recognition, Keras, Kaggle.

Acknowledgement

I would like to offer a special thanks to my advisor, Dr. Daming Zhang, for his help and guidance as I worked on this project this past semester. Also, I want to thank the Industrial Technology department of the California State University Fresno, especially to the faculty engaged in the MSIT program. Lastly I would like to thank the Kaggle for hosting Digit Recognizer competition to gain knowledge in the new concepts. Finally, I would also like to thank my family, fellow colleagues and all the people that have supported me.

Contents

Sl. No Title Page No

List of Tables & Figures

1.0 Introduction

Intelligent image analysis is an interesting research area in Artificial Intelligence and also important to a variety of current open research problems. Handwritten digits recognition is a well-researched subarea within the field, which is concerned with learning models to distinguish pre segmented handwritten digits. The application of machine learning techniques over the last decade has proven successful in building systems which are competitive to human performance, and which perform far better than manually written classical AI systems used in the beginnings of optical character recognition technology. However, not all aspects of such models have been previously investigated.

1.1 Machine Learning

Machine Learning belongs to the subfield of computer science which got developed from the study of computational learning and pattern recognition theory of Artificial Intelligence. It is defined as a set of methods which can automatically detects the patterns of data, and it also uses the uncovered patterns to anticipate the future data, or to execute other ways of decision making (like how to collect more data). Machine learning deals with several different disciplines such as control theory, reinforcement learning, statistics, probability theory, logic design, and combinatorial optimization. In general, machine learning algorithms are used to represent, evaluate and optimize the given data.

Machine learning techniques originated from computer science and statistics subject backgrounds, it got emerged during the path of the decline of neural networks systems and introduction to the knowledge-intensive systems. Methods of Machine Learning: Depending upon the algorithms there are three methods of machine learning available. They are supervised learning, unsupervised learning, and reinforcement learning. Artificial neural network, case-based reasoning, Bayesian statistics, learning automata, regression analysis, linear classifiers, decision trees, and hidden Markov models comes under supervised learning. Unsupervised learning had four types of algorithms; they are artificial neural network, association rule learning, hierarchical clustering, and partitioned clustering. Only two algorithms are under reinforcement learning they are Q-learning and learning automata.

The most commonly and widely used machine learning algorithms are induction, clustering, analogy, discovery, genetic algorithms and reinforcement. Using machine learning you can develop your own system that can automatically adapt and customize themselves to the users. (e.g. spam filtering). There are few examples of machine learning problems: The list below is few common classification problems, where the goal is to achieve the solution using the machine learning techniques, Face detection: Considering specific input features, machine learning identifies the particular faces in an image. In this project the machine learning is used to determine the handwritten single digit images, where the data for the project is taken from the Kaggle website.

To make it more clear how Machine Learning can be implemented in the Data Science, I will explain a small python script code where it shows the application of Machine Learning algorithm.
Example: Gender Classifier application:
Firstly, there are four steps which you have to follow:
Step 1: Install Python ~ Python 2.7/3.5, in this code Python 2.7 version is used

Step 2: Setup Environment ~ Sublime Text 3 (can also use other)
Step 3: Install Dependencies ~ Scikit learn
Step 4: Write Python Script.

The python script and explanation:

Firstly, import the dependencies for the algorithm. In this code, only one dependency Scikit learn is used.

from sklearn import tree

For all dependencies we're going to use a specific sub model of Scikit-learn call tree that will let us build a machine learning model called a decision tree which is like a flow chart that stores data. It asks each label data it receives a yes/no question does it contain X or not, if the answer is yes the data moves to one direction, if answer no it will moves in the other. It'll build every node in the tree the more data points it receives then we have a new unlabelled data, we can feed it to the tree it will ask you series of questions until it labels that label is our classification, the more data we traded on it the more accurate the classification will be.

#[height,weight,shoesize]

X = [185,45,44], [145,67,38], [156,56,41], [165,58,40], [159,61,41], [178,66,44], [189,69,45], [190,88,46]

X is as a list of lists a variable (it is a value that can change and sort list of lists in it this is a data type in python that can store a sequence)

Y stores a list of labels, each stores gender associated with a body metric in the previous list and will write them as strings

Y = ['male', 'female', 'female','male', 'female', 'male', 'female', 'male']

Decision tree is an entry model that classifies data by creating branches for every possible outcome. Now another variable to store our decision tree model, let's call it as clf and store our decision tree classifier and now we have tree variable and we can train it on our data set.

clf = tree.DecisionTreeClassifier()

Now, will call the fit method on the classifier variable that takes two arguments to store variables x and y. The result will be stored in the updated elf variable the fit method trains the decision tree on our dataset.

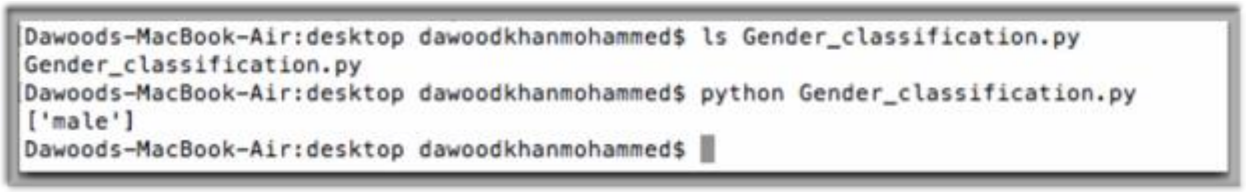
clf = clf.fit(X,Y)

Let's test it by classifying the gender of someone by giving a new list of body Metrix will create a variable called prediction to store the results

prediction =clf.predict([[165,54,42]])

Finally call the predict method of our decision tree to predict the gender given these value print prediction

The output will be as shown below, it has predicted the gender as “Male” with the Body Metrix that we have provided.



```
Dawoods-MacBook-Air:desktop dawoodkhanmohammed$ ls Gender_classification.py
Gender_classification.py
Dawoods-MacBook-Air:desktop dawoodkhanmohammed$ python Gender_classification.py
['male']
Dawoods-MacBook-Air:desktop dawoodkhanmohammed$
```

Figure 1. Output for gender classifier application

This is how the machine learning algorithm can be used to determine or predict the output by providing it the past or train data to learn. In the same way, this project uses machine learning algorithm to predict the test data by making it learn the train data.

1.2 Objective

- ➢Take an image of a handwritten single digit
- ➢Build Machine Learning algorithm using Convolutional Neural Network classifier
- ➢Download and import of different dependencies
- ➢Determine what that image is through Tensorflow and Theano deep learning packages
- ➢Compete in Digit Recognizer competition ~ Kaggle website

2.0 Literature Reviews

Handwritten digit recognition is the most important problem faced in optical character recognition, and it has been used widely as a test case for pattern recognition and also as a machine learning algorithms techniques for a long period of time. Historically, to promote machine learning and pattern recognition research, several standard databases have emerged in which the handwritten digits are preprocessed, including segmentation and normalization, so that researchers can compare recognition results of their techniques on a common basis. The freely available MNIST database of handwritten digits has become a standard for fast-testing machine learning algorithms for this purpose. Different approaches used for the recognition of handwritten characters, a few are mentioned below:

Handwritten Character Recognition using Neural Network
Chirag I Patel, Ripal Patel, Palak Patel
The main objective of this paper is to recognize the characters of a scanned documents, neural networks are most generally used for pattern recognition task. The paper also describes the behaviors of different models of neural network used in Optical Character Recognition (OCR), OCR is usually a widespread use of Neural Network [1]. In this paper, Multilayer Feed Forward network with Back propagation is used and in preprocess of data some basic algorithms are used for segmentation of characters, normalizing of characters and De-skewing. After that different models of neural network are applied to test set on each to find the best results based upon the accuracy of the respective neural network [1].

Character Recognition Using Matlab's Neural Network Toolbox
Kauleshwar Prasad, Devvrat C. Nigam, AshmikaLakhotiya and DheerenUmre.
Recognition of handwritten text is one of the most active and challenging areas of research in both the image processing and pattern recognition fields. It has many applications that include reading aid for blind, bank cheques and conversion of any hand written document into structural text form. In this project, recognition of English alphabet in a given scanned document is done with the help of Neural Networks [2]. Using Mat Lab Neural Network toolbox, recognition of handwritten characters is done by projecting them on different sized grids, the first step performed is image acquisition which helps in noise filtering of a scanned image, rendering image suitable for segmentation where image is decomposed into sub images. Also in this project character extraction and edge detection algorithm are used for training the neural network to determine and recognize the handwritten characters [2].

Handwritten Character Recognition using Gradient Features
Ashutosh Agarwal, Rajneesh Rani, RenaDhir.
For any recognition system feature extraction is an integral part, the purpose of feature extraction is to describe the pattern class by means of minimum number of features that are most effective and necessary in discriminating pattern classes [3]. The gradient is used to measure the magnitude and direction of the change in intensity even in a small pixel where the gradient refers to a magnitude and direct ion. Sobel operator, sometimes called the Sobel filter is used in image processing and computer vision, specifically within the algorithms of edge detection where it creates an emphasizing edges. In this paper recognition of English characters are obtained with a

recognition accuracy of 94%. Gradient Features are recommended for recognition purpose because of its of ease of use and high recognition rate [3].

Digit Recognizer Using Machine Learning to Decipher Handwritten Digits
Matheus M. Lelis
The goal of this research project is to create an artificial neural network machine able to recognize handwritten letters and numbers. Through multi-class classification and using the back propagation algorithm, the proposed machine should be able to learn the difference between each handwritten number and after proper training, correctly guess the number in the inputted image. The machine will be written in MATLAB® language using new original code and pre-existing code when it is applicable [4].

Machine Learning for digit recognition problem
Musings in Data Science (www.deblivingdata.net)
In this project approach Random Forest Classifier is used to train this data and test with Kaggle's test data. Random Forest is a learning technique that uses an ensemble of decision trees. An ensemble of classifiers has better classification performance than individual classifiers and is more resilient to noise. These decision trees are built from a sample drawn with replacement (Bootstrap sample) from the training set. This prevents overfitting since it decreases the variance without changing the bias. Bootstrapping is sometimes complemented with "Boosting" where at each step the weights given to data points are adjusted so as to give higher weight to misclassified data and retrain the sample. The disadvantage of "Boosting" vs "Bagging" is that its cannot be distributed easily. At each time step the weights depend on the previous time step. While splitting the decision tree the splitting is not done on the entire set of features, rather it is done on a random set of m features [5].

3.0 Project Statement

The purpose of this project is to take handwritten single digit image as input, process the data, train the machine learning algorithm, to recognize the digit image.

This project is aimed at developing software which will be helpful in recognizing handwritten single digit and compete in the Kaggle website to secure a high rank with good score and accurate digit recognizer output comma separated files. This project is restricted to English written single digit images only. It is also helpful in recognizing different format digits.

One of the primary means by which computers are endowed with humanlike abilities is through the use of a machine learning. Machine learning is particularly useful for solving problems that cannot be expressed as a series of steps, such as recognizing patterns, classifying them into groups, series prediction and data mining.

As is often the case when humans can't directly do something, we've built tools to help us. There is an entire, well-developed field, called dimensionality reduction, which explores techniques for translating high-dimensional data into lower dimensional data. Much work has also been done on the closely related subject of visualizing high dimensional data.

Digit recognition is perhaps the most common use of machine learning. The machine learning is presented with a target vector and also a vector which contains the pattern information, this could be an image and hand written data. The machine learning then attempts to determine if the input data matches a pattern that the algorithm or model that has memorized.

A machine learning trained for classification is designed to take input samples and classify them into groups. Machine learning often demands we work with thousands of dimensions – or tens of thousands, or millions! Even very simple things become hard to understand when you do them in very high numbers of dimensions. This project concerns detecting free handwritten digits and competing on an ongoing real competition through Kaggle website.

4.0 Methodology

- ❖ Learning about Digit Recognizer ~ Kaggle
- ❖ Approach to different methods of Digit Recognizer
- ❖ Selecting best method and look after tools
- ❖ Building Machine Learning Algorithm model
- ❖ Submission of results & improvements ~ Kaggle

4.1 Kaggle

Kaggle is a Silicon Valley start-up and Kaggle.com is its online platform hosting many data science competitions. Kaggle uses a crowdsourcing approach that relies on the fact that "there are countless strategies that can be applied to any predictive modelling task and it is impossible to know at the outset which technique or analyst will be most effective." Kaggle competitions are hosted by technology companies who post their data as well as a description of the problem on the website, Participants from all over the world can experiment with different techniques and submit their best results to a scoreboard to compete; After the deadline passes, the winning team or participant will receive a cash reward and the company obtains "a worldwide, perpetual, irrevocable and royalty-free license". Being participating in a Kaggle competition will make you to experience with large, complex, interesting, real data and the participant can learn new knowledge and skills and top of that participant can be a part of the data science community and also changes of getting hired by any company or could get to a job.

4.2 Different methods of Digit Recognizer

As we have seen in the literature review there are different methods and techniques used to determine the handwritten single digit image through computer. I haven't chosen those methods because even every method has its own advantages. The first method with Neural Networks was very basic approach and it differs a lot from the advanced artificial neural network system and it has no sub hidden layers and the layer connection was just the feed forward system and that's the reason that the method couldn't be able to execute high end data projects like the one which we selected from Kaggle.

The next method which uses MatLab Neural Network toolbox has a lot of steps to go through to decipher handwritten single digit image such as image acquisition, noise filtering, smoothing, normalizing, feature extraction, and edge detection method which used to detect the images through the algorithm as it has long way to predict the images and the data that provided by Kaggle is huge and the MatLab approach will take forever to complete. The other method with the gradient features it will usually measure the directional change or color intensity of the image with the magnitude of it and also it uses the Sobel operator which generally used in Image processing and computer vison for edge detection algorithm. This method gives accuracy level of 94.5 percent which is actually good from the above two methods, but again as our main goal is to compete in

Kaggle website that score is not going to fetch high rank that is the reason I have not chosen even this method.

Method chosen for the project

In this project to get the results more accurate and less error percentage rate few different types of dependencies and library functions are used, the coding of the machine learning algorithm is done by using Python script and based on the Keras dependency which works on both the TensorFlow and Theano open source licensed library functions the code runs. Convolutional Neural Networks or otherwise CNNS/ ConvNets are used in this project instead of using Neural Networks, CNNs are very similar to the ordinary Neural Networks, but they are made up of neurons that have learnable weights and biases. Better results are obtained using the above method compared to other traditional methods explained.

4.3 Software requirements & Tools

To complete this project, there is only need of a computer with the following software requirement.

Operating System: Mac OS/ Windows/ Linux
Technology: Python 2.7/3.5 version
IDE: Spyder 3.0 ~ Anaconda

PIP installer for installing packages and library functions that may also include dependencies for the code. The coding of the project will be done using the Python language. The only reason for selecting the python language is, it is well suited to solve data problems because of its quick, intuitive, well documented and pre library stored features. In this project, the setup will be Spyder 3.0 coupled with Anaconda. To get the best results with the good accuracy, less processing and execution time, the python language will be using some other tools like Tensor Flow, which uses data flow graphs for the numerical computation.

4.4 Build of a Machine Learning Algorithm

It’s a stepwise procedure to build a model, firstly we need which data which is the most important part of the project.

Data for the project

As already mentioned, the project is dealing with the MNIST data, the MNIST database was constructed out of the original NIST database; hence, it is named as modified NIST or MNIST [7]. There are 60,000 training images in which some of these training images can also be used for cross- validation purposes and 10,000 test images, both are drawn from the same distribution. The images used in this project as dataset are black and white digits which are size normalized and are centred in a fixed size image position where the center of gravity of the intensity lies at the center of the image with 28 #28 pixels. Thus, the dimensionality of each image sample vector is 28 * 28 = 784, where each element is binary. And also the image pixel data is already encoded into numeric values in a CSV file. A CSV is a comma separated values file, in which the data is saved in a table structured format. Generally, they take the form of a text file which contains information separated by commas, hence how the name it got.

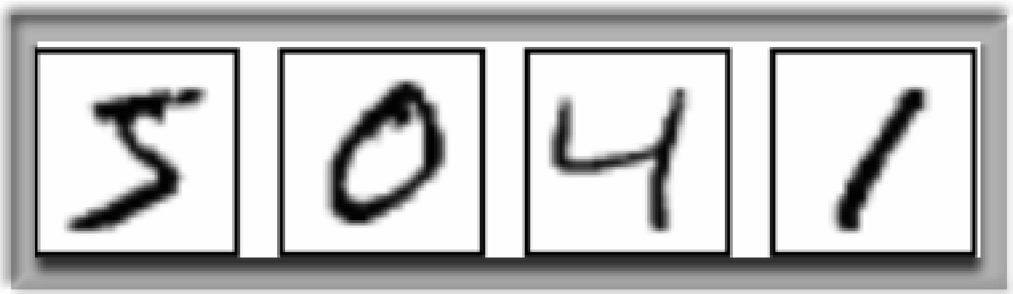
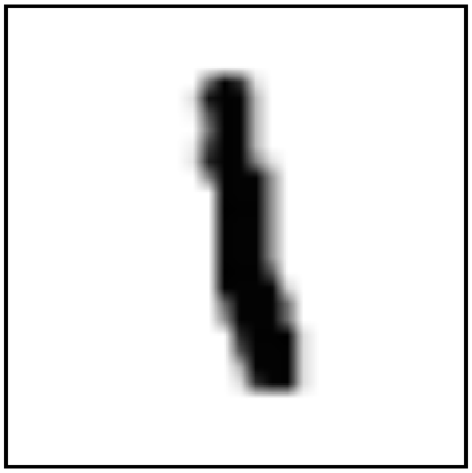


Figure 2. Handwritten single digit image
Image retrieved from http://colah.github.io/posts/2014-10-Visualizing-MNIST/

The two MNIST data CSV files that are used in this project are train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total as already mentioned. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.



$$\cong \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\ 0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\ 0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

Figure 3. Image as an array of numbers describing each pixel
Image retrieved from http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Every MNIST data point, every image can be thought of as an array of numbers describing how dark each pixel is. For example, we might think of 1 as something like shown in the Fig.2. In the Figure 3. above shown the pixel density of each component vector of the handwritten single digit image, but the pixel values are shown between 0 and 1 because when you run algorithm for the input image the steps that undergo to convert the image into arrays where the dependencies are used, theymake the conversion of integer value [0;255] will get convert into float values between [0;1] for better precision of reading the image and also more accuracy to predict the test image.

In the 784-dimensional space not all the vectors are MNIST digits. Few points in this space are very typical and different. To get more sense what a typical point looks like, we can randomly pick a few points and examine them. Generally, in a random point – a random 28*28 image- each pixel is randomly black, white or some shade of gray. The result of it is shown in the below figure 4 with noise [7].

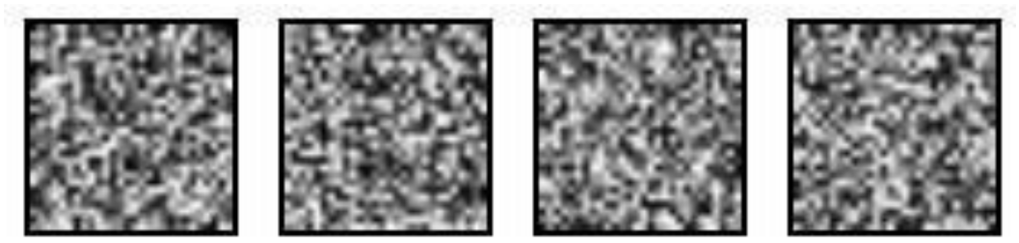
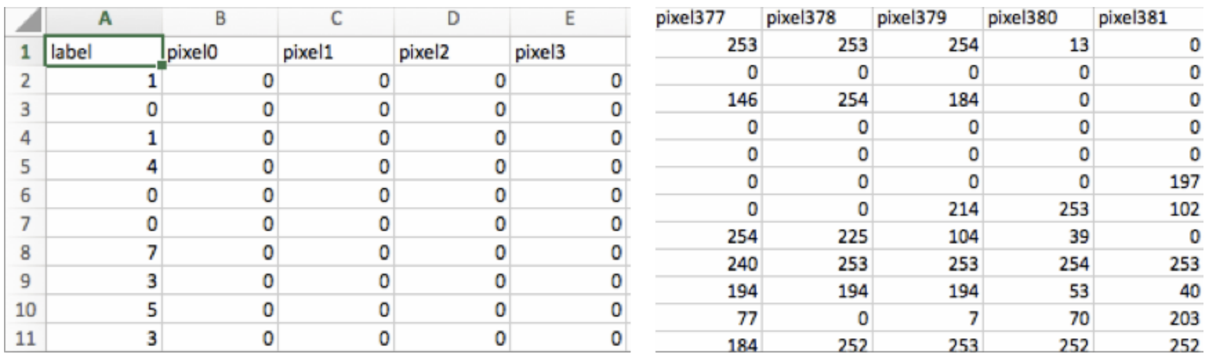


Figure 4. Random points in a random 28*28 image.
Image retrieved from http://colah.github.io/posts/2014-10-Visualizing-MNIST/

The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image. Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i * 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero) [6].



	A	B	C	D	E
1	label	pixel0	pixel1	pixel2	pixel3
2	1	0	0	0	0
3	0	0	0	0	0
4	1	0	0	0	0
5	4	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	7	0	0	0	0
9	3	0	0	0	0
10	5	0	0	0	0
11	3	0	0	0	0

pixel377	pixel378	pixel379	pixel380	pixel381
253	253	254	13	0
0	0	0	0	0
146	254	184	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	197
0	0	214	253	102
254	225	104	39	0
240	253	253	254	253
194	194	194	53	40
77	0	7	70	203
184	252	253	252	252

Figure 5. Train data csv file

The above figure 5 shows the train data csv file where you could see the label column as the first one, it is the actual handwritten single digit image and the other columns represent from pixel0 to pixel 784 because the images considered in this project are 28 x 28 dimensions and the pixel value ranges from 0 to 255 inclusive. The lower value represents lightness and the higher value represents darkness.

For example, pixel31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below [6].
Visually, if we omit the "pixel" prefix, the pixels make up the image like this:
000 001 002 003 ... 026 027

028 029 030 031 ... 054 055
056 057 058 059 ... 082 083
| | | | ... | |
728 729 730 731 ... 754 755
756 757 758 759 ... 782 783
The test data set, (test.csv), is the same as the training set, except that it does not contain the "label" column.

Your submission file should be in the following format: For each of the 28000 images in the test set, output a single line containing the Image Id and the digit you predict. For example, if you predict that the first image is of a 3, the second image is of a 7, and the third image is of a 8, then your submission file would look like:
Image Id, Label
1,3
2,7
3,8
(27997 more lines)
The evaluation metric for this contest is the categorization accuracy, or the proportion of test images that are correctly classified. For example, a categorization accuracy of 0.97 indicates that you have correctly classified all but 3% of the images [6].

4.5 Design and modelling of a Machine Learning Algorithm

Design of Convolutional Neural Networks (CNNs / ConvNets): Convolutional Neural Networks are very similar to ordinary Neural Networks and they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

The complete project and machine learning algorithm is written through python script. The only reason behind selecting python as a programming language for this project because it is a high-level, dynamically typed multiparadigm programming language. Python code is often said to be almost like pseudocode, since it allows you to express very powerful ideas in very few lines of code while being very readable. Also top of that python language can accept many library functions that are suitable for a machine learning algorithm to work. Python 2.7 version is used for the project.

Now after the data there is a need of dependencies and library functions to be used in the code. The different library functions that are used in this code are Numpy, Pandas, and Keras. The reason behind each library function that used:

Numpy is used as the core library for scientific computing in Python programming language [8]. It provides a high-performance multidimensional array object, and tools for working with arrays and it is also used to seamlessly and speedily integrate with a wide variety of databases. In this project Numpy is used for reproducibility as np.random.seed(1337). The reproducibility refers to the ability to run the same thing twice and get the same results. As this is necessary to train and check the algorithm that whether it is providing correct answers or not.

Pandas are the next library functions used in the code and they are used for high-performance [9], easy-to-use data structures and data analysis tools for the Python programming language. Python with *pandas* is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more. In the code, pandas are used to open, and read of the csv files. The general syntax used to read the csv files are pd.read_csv("/file path location/train.csv").

4.5.1 Keras library function

It is used to build the machine learning algorithm model. The machine learning algorithm model has the Keras sequential model imported with many other sub layers in it. Keras is a deep learning library function can be used on top of either Theano or TensorFlow [10].

Theano

It is a Python library that lets you to define, optimize, and evaluate mathematical expressions, especially ones with multi-dimensional arrays (numpy.ndarray). Using Theano it is possible to attain speeds rivaling hand-crafted C implementations for problems involving large amounts of data

Tensor Flow

One of the most popular and recent software library functions for numerical computation on machine learning is Tensor Flow[11]. Tensor Flow is an open source software library that uses data flow graphs for the numerical computation. It is represented by nodes and graph edges, nodes of the graph are used to represent the mathematical operations of the data, whereas graph edges in the graph represent the multidimensional data array, which are also called as tensors that are communicated between the node and graph. Tensor flow has a single second generation Application Programming Interface (API), which got a flexible architecture which helps you to deploy computation to more than one Central Processing Units or Graphics Processing Units in a computer desktop, server, or any mobile device[11].

Tensor Flow Features

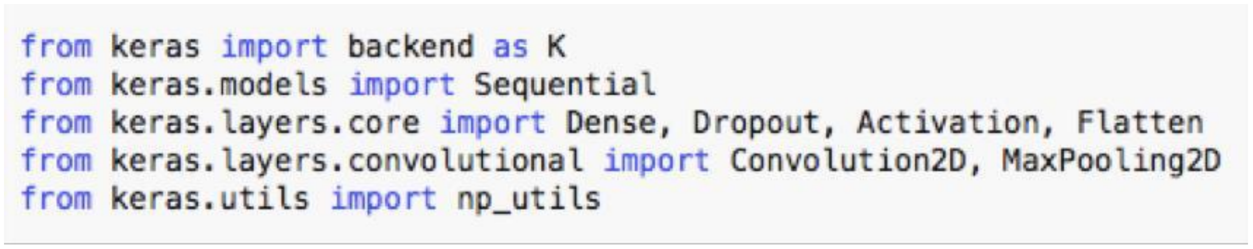
Tensor Flow has excellent features such as Deep Flexibility, In the Tensor Flow you don't need to follow a specific language to use. If you can express anycomputation in the form data flow graph then you are good to use it. Tensor Flow by default provides a tool that helps to assemble sub-graphs which are common to use in any neural networks. True Portability: Tensor Flow runs on both the CPU's and GPU's, and also on any desktop, and server. Research and Production by using Tensor flow: There is a lot of research undergoing using Tensor Flow. Many scientists at industries are conducting research on Tensor flow to make the ease and faster production by using this Artificial Intelligence program. It is also helpful to students to conduct research on the Tensor flow[11].

Keras is a high-level neural networks library, written in python programming language. The main idea for developing Keras is to focus on enabling fast experimentation, it is being able to go from idea to result with the minimum possible delay which helps a lot for good research in which digit recognizer is one of the example.

Keras has an outstanding features compared to other library functions that is why it is chosen in this project:

- It helps in fast and easy prototyping through total modularity, minimalism, and extensibility.
- It generally supports both type of famous artificial neural networks, they are: convolutional networks which is used in this project model and the other one is recurrent network.

- It also supports arbitrary connectivity schemes that includes multi-input and multi-output training.
- Keras can also run on both the CPU and GPU.



```
from keras import backend as K
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
```

Figure 6. Importing Keras function

The Figure 6. above shows importing of Keras library functions in the code, the sequential model imported from the Keras library function shown in the second line of the Figure 6. It is a linear stack of layers and it helps in creating a sequential model by passing a list of layer instances to the constructor. There are many other layers and models are imported that are shown in the third and fourth line of the figure 6. Each model and layer has its own advantages and roles.

The dense layer is used for processing the output of an Embedding layer or a recurrent layer with its inner sequence features. This feature also helps to transform the hidden representations at each timestep before applying further processing like pooling or another recurrent layers.

Generally, when deep neural networks with a large number of parameters are usually very powerful machine learning systems. However, it is a serious problem as overfitting is a serious problem in such networks and also large networks are also slow to use and this makes difficult to deal with overfitting by combining the predictions of many different large neural networks at test time. Dropout is a key technique that can be used in such cases to solve such problems, this helps from co-adapting too much. It can be simply imported by using keras.layers.core.Dropout(p).

4.5.2 Convolutional Layers

There are different types of convolutions such as convolution1D, convolution2D, and convolution3D. Generally, each convolutions changes little differ from the other and they have different methods to import and approach in the code through Keras and are written in python programming language. It is also named as the convolution operator; it is used for filtering neighborhoods of one-dimensional inputs. When using this layer as the first layer in a model, either provide the keyword argument input_dim (int, e.g. 128 for sequences of 128-dimensional vectors), or input_shape (tuple of integers, e.g. (10, 128) for sequences of 10 vectors of 128-dimensional vectors). Convolution is explained, but what is CNNs? They are basically just several layers of convolutions with nonlinear activation functions like ReLU that is applied to the results. In a traditional feedforward neural network, we connect each input neuron to each output neuron in the next layer. That's also called a fully connected layer, or affine layer. In CNNs we don't do that. Instead, we use convolutions over the input layer to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies different filters, typically hundreds or thousands like the ones showed below in Figure 7, and combines their results[12].

During the train data is provided, a CNN automatically learns the values of its filters based on the task you want to perform. For example, in Image Classification a CNN may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to deter higher-level features, such as facial shapes in higher layers. The last layer is then a classifier that uses these high-level features [12].

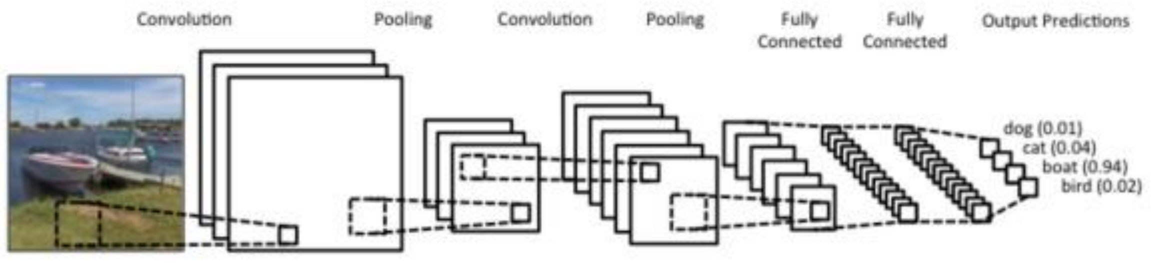
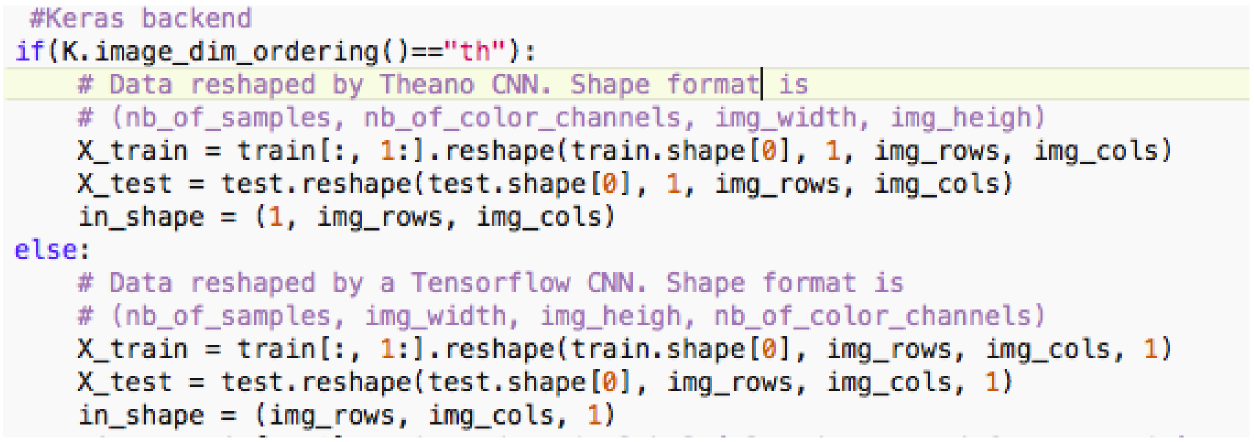


Figure 7. Internal layers of CNN that how detects image.
Image retrieved from http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

4.5.3 Keras backend

Keras is a model-level library, providing high-level building blocks for developing deep learning models. It does not handle itself low-level operations such as tensor products, convolutions and so on. Instead, it relies on a specialized, well-optimized tensor manipulation library to do so, serving as the "backend engine" of Keras. Rather than picking one single tensor library and making the implementation of Keras tied to that library, Keras handles the problem in a modular way, and several different backend engines can be plugged seamlessly into Keras. In this project both the Theano and Tensorflow backend is used.



```
 #Keras backend
if(K.image_dim_ordering()=="th"):
    # Data reshaped by Theano CNN. Shape format is
    # (nb_of_samples, nb_of_color_channels, img_width, img_heigh)
    X_train = train[:, 1:].reshape(train.shape[0], 1, img_rows, img_cols)
    X_test = test.reshape(test.shape[0], 1, img_rows, img_cols)
    in_shape = (1, img_rows, img_cols)
else:
    # Data reshaped by a Tensorflow CNN. Shape format is
    # (nb_of_samples, img_width, img_heigh, nb_of_color_channels)
    X_train = train[:, 1:].reshape(train.shape[0], img_rows, img_cols, 1)
    X_test = test.reshape(test.shape[0], img_rows, img_cols, 1)
    in_shape = (img_rows, img_cols, 1)
```

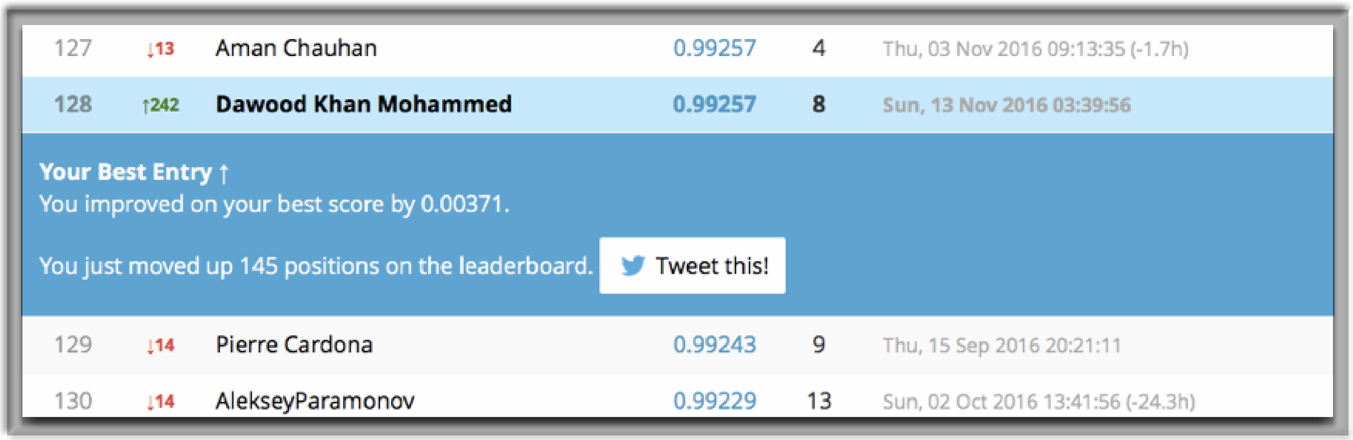
Figure 8. Checking of Keras backend (Theano and TensorFlow)

The Figure 8. Shows the import of Keras backend and the format that needed the backend engine to work, it rearranges/reshapes the data in its on format as shown in the above Figure 8. After the Keras backend, conversion of data (train and test) into value of floats [0;1] instead of integer in [0;255] is done, this helps for more or better precision. Later on calling of core layers of Keras that includes pooling layers, convolution. Next, the data is flattening to 1D tensor from 3D output to fully connect the layers that makes to accept the input data. The optimization of the model is done through cross entropy between the true label and output (softmax) of the model.

The model is made to learn the data through the command that we have seen the same in the gender classifier example in the introduction about machine learning model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch, verbose=1), after the model learns the data it is given the test data to predict. Finally, Numpy library function variable is used to save the predicted test data of the project.

5.0 Results:

- The time taken to complete run (execution) of the code is approximately 40 – 45 minutes and again that depends on the configuration of the computer and usage of dependencies.
- The output is an comma separated value file with two columns, one is the image ID and the other is the label.
- The label column of the test.csv file represents the predicted handwritten single digit image.
- After the test.csv file is saved, it is uploaded in the Kaggle competition website for evaluating the score and rank.
- The below figure.9 shows the rank holding in the Kaggle website as 128th position and with an accuracy score of 0.99257



127	↓13	Aman Chauhan	0.99257	4	Thu, 03 Nov 2016 09:13:35 (-1.7h)
128	↑242	**Dawood Khan Mohammed**	0.99257	8	Sun, 13 Nov 2016 03:39:56

Your Best Entry ↑
You improved on your best score by 0.00371.

You just moved up 145 positions on the leaderboard.

129	↓14	Pierre Cardona	0.99243	9	Thu, 15 Sep 2016 20:21:11
130	↓14	AlekseyParamonov	0.99229	13	Sun, 02 Oct 2016 13:41:56 (-24.3h)

Figure.9 Rank display on Kaggle website
Image retrieved from https://www.kaggle.com/c/digit-recognizer/leaderboard

6.0 Future Scope:

- The rank I achieved in the Kaggle competition is after an series of attempting and improving the code in the competition.
- The rank can improved to high by increasing the nb_epoch value in the machine learning algorithm model that is increasing the number of epoch value.
- Nb_epoch value means the number of times the whole data is used to learn by the algorithm. I have taken the epoch value as 142.
- As you keep increasing the epoch value you need high configuration computer or a server connected computer to execute the code.
- There is also another recommendation to improve the reproducibility value of the program that is what we import through Numpy.
- The reproducibility value will means ability to run the same thing twice and get the same results. That's why we seed our sources of randomness.

7.0 Conclusions:

- The scripting of the Machine Learning model was completely done through Python programming language, which is compact, easier to debug, and allows for ease of extensibility.

- The method is advantageous as it uses different sub layers of the convolutional network to train the neural network using Convolutional 2D, Pooling layer, and Maxpooling and other features imported through Keras dependency. The advantage lies in less computation involved in feature extraction, training and classification phases of the method.

- Obtained high accuracy rate as we had used both the TensorFlow and Theano open source library functions that were backend in the code through Keras dependency.

- As the code of the Machine Learning algorithm was based on Keras library function, the results were better and secured good score.

- The script of the Machine Learning algorithm model was reduced by the usage of Keras library function as it has a lot stack of layers and sub library functions which are easily imported in the script.

Appendix

```
#Importing Pandas and Numpy
import pandas as pd
import numpy as np
#For Reproducibility
np.random.seed(1337)

#Importing keras
from keras import backend as K
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils

# Assigning image dimensions
img_rows, img_cols = 28, 28

# Number of images used in each optimization step
batch_size = 128
# One class per digit
nb_classes = 10
# Number of times the whole data is used to learn
nb_epoch = 142

# Read of train and test datasets
train = pd.read_csv("/Users/dawoodkhanmohammed/Documents/Spyder/train.csv").values
test  = pd.read_csv("/Users/dawoodkhanmohammed/Documents/Spyder/test.csv").values

# Check Keras backend
if(K.image_dim_ordering()=="th"):
    # Reshape the data to be used by a Theano CNN. Shape is
    # (nb_of_samples, nb_of_color_channels, img_width, img_heigh)
    X_train = train[:, 1:].reshape(train.shape[0], 1, img_rows, img_cols)
    X_test = test.reshape(test.shape[0], 1, img_rows, img_cols)
    in_shape = (1, img_rows, img_cols)
else:
    # Reshape the data to be used by a Tensorflow CNN. Shape is
    # (nb_of_samples, img_width, img_heigh, nb_of_color_channels)
    X_train = train[:, 1:].reshape(train.shape[0], img_rows, img_cols, 1)
    X_test = test.reshape(test.shape[0], img_rows, img_cols, 1)
    in_shape = (img_rows, img_cols, 1)
y_train = train[:, 0] # First data is label (already removed from X_train)

# Conversion to floats [0;1] from int [0;255] values
X_train = X_train.astype('float32')
```

```
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

# One-hot Vector - conversion of class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)

#Check of images format
print('X_train shape:', X_train.shape)
print('Y_train shape:', Y_train.shape)
print('X_test shape:', X_test.shape)

# Import Sequential layer- Keras
model = Sequential()
model.add(Convolution2D(12, 5, 5, activation = 'relu', input_shape=in_shape, init='he_normal'))

# import of pooling layers from keras
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Convolution2D(25, 5, 5, activation = 'relu', init='he_normal'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten the 3D output to 1D tensor for a fully connected layer to accept the input
model.add(Flatten())
model.add(Dense(180, activation = 'relu', init='he_normal'))
model.add(Dropout(0.5))
model.add(Dense(100, activation = 'relu', init='he_normal'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes, activation = 'softmax', init='he_normal')) #Last layer with one
output per class

# The function to optimize is the cross entropy between the true label and the output (softmax) of
the model
model.compile(loss='categorical_crossentropy', optimizer='adamax', metrics=["accuracy"])

# Make the model learn
model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch, verbose=1)

# Predict the label for X_test
yPred = model.predict_classes(X_test)

# Save prediction in file for Kaggle submission
np.savetxt('Kaggle-pred.csv', np.c_[range(1,len(yPred)+1),yPred], delimiter=',', header =
'ImageId,Label', comments = '', fmt='%d')
```

References

[1] Chirag I Patel, Ripal Patel, Palak Patel, “Handwritten Character Recognition Using Neural Networks”, International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011.

[2] Kauleshwar Prasad, Devvrat C Nigam, AshmikaLakhotiya, DheerenUmre, “Character Recognition Using Matlab’s Neural Toolbox”, International Journal of u- and e- Service, Science and Technology Vol. 6, No. 1, February, 2013.

[3] AshutoshAggarwal, Rajneesh Rani, RenuDhir, ”Handwritten Character Recognition Using Gradient Features”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 5, May 2012.

[4] Matheus M.Lelis, “Digit Recognizer Using Machine Learning to Decipher Handwritten Digits”, MTH 499 CSUMS FALL 2012, University of Massachusetts.

[5] Musings in Data Science, “Machine Learning for digit recognition problem”, www.deblivingdata.net.

[6] Kaggle, “Digit Recognizer”, kaggle.com.

[7] Fabien Tence, Ankivil, Machine Learning Experiments, “MNIST Database and Simple Classification Networks” wwwankivil.com, 04 May, 2016.

[8] Scipy.org, “NumPy”, numpy.org.

[9] Pandas, “Python Data Analysis Library”, pandas.pydata.org.

[10] Keras, “Keras: Deep Learning library for Theano and TensorFlow”, keras.io

[11] TensorFlow, “TensorFlow is an Open Source Software Library for Machine Intelligence”, tensorflow.org.

[12] WildML, AI, Deep Learning, NLP, “Understanding Convolutional Neural Networks for NLP”, wildml.com.