

**BAHRIA UNIVERSITY ISLAMABAD**

**Department of Software Engineering**

**CNN Classifier**

**Project By:**

Muhammad Ali Raza Javaid (01-131172-043)

Pakeeza Saleem (01-131172-021)

Maira Babar (01-131172-011)

**Submitted to:** Sir Abdul Hanan

**Class:** BSE - VI

**Dated:** 25th June, 2020

Table of Contents

[**Problem Statement** 3](#_Toc44001438)

[**CNN Classifier** 3](#_Toc44001439)

[**Working of the Project** 3](#_Toc44001440)

[**Mnist** 3](#_Toc44001441)

[**Implementation with Keras using TensorFlow** 3](#_Toc44001442)

[**Setup** 3](#_Toc44001443)

[**Convolution** 4](#_Toc44001444)

[**Full Connection** 4](#_Toc44001445)

[**Compiling CNN** 4](#_Toc44001446)

[**Training the model:** 5](#_Toc44001447)

[**Training Accuracy** 5](#_Toc44001448)

[**Predictions** 5](#_Toc44001449)

[**Conclusion** 6](#_Toc44001450)

**CNN Classifier**

# **Problem Statement**

Your task is to collect 10 different image categories and build a CNN classifier using those images.

# **CNN Classifier**

A Convolutional Neural Network (CNN) is a multilayered neural network. CNNs have been used in image recognition, powering vision in robots, and for self-driving vehicles. An image classifier CNN can be used to differentiate between a cat and dog image and whether a brain image consists of tumor or not. Once a CNN is built, it can be used to classify the contents of different images this can be done when the images are feed into the model. CNNs are able to classify images by detecting features.

# **Working of the Project**

In this project we took the image dataset from *mnist.*

## **Mnist**

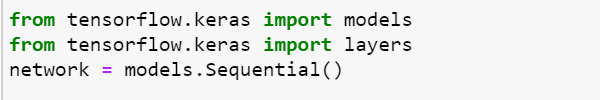
The MNIST database is a large [database](https://en.wikipedia.org/wiki/Database) of handwritten digits that is commonly used for [training](https://en.wikipedia.org/wiki/Training_set) various [image processing](https://en.wikipedia.org/wiki/Image_processing) systems. The MNIST database contains 60,000 training images and 10,000 testing images. The dimension of training images are 2. The shape of training images are (60000, 784). The type of training images are uint8

## **Implementation with Keras using TensorFlow**

First we give appropriate names i.e. the training set and the test set to the folders containing the images. This makes it easier to import the images into Keras.

### **Setup**

After this we need to import Keras and other packages that are going to be used in building the CNN. To initialize the neural network we used the Sequential model. In this project we created a Sequential model incrementally through add () method.

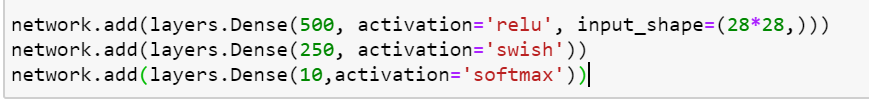


### **Convolution**

### **Full Connection**

The *Dense* function adds the fully connected layer to the neural network.

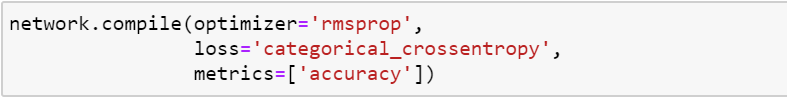
* For the first layer we used the ‘*relu*’ activation function. Relu stands for Rectified Linear Unit. It is used to increase non-linearity in the CNN.
* For the second layer we used the ‘*swish*’ activation function. As it performs better than ReLU with a similar level of computational efficiency.
* For the third layer we used the ‘*softmax*’ activation function. The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.



### **Compiling CNN**

We compiled the CNN using the compile function. This function expects three parameters: the optimizer, the loss function, and the metrics of performance.

* In our project, we used ‘rmsprop’ optimizer. The optimizers shape and mold the model into its most accurate possible form. RMSProp is Root Mean Square Propagation.
* The loss function is the guide to the terrain, telling the optimizer when it’s moving in the right or wrong direction. In our project we used *categorical\_crossentropy* loss function. This function is used for multiclass problem.
* A metric is a function that is used to judge the performance of the model. We used ‘*accuracy’*.



### **Training the model:**

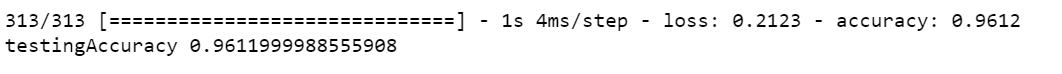
We categorized our data. The function fit consists of four parameters.

* The first parameter is the training images.
* The second parameter is the training labels.
* The *batch\_size*is the number of images that will go through the network before the weights are updated. Our batch\_size is 128.
* The epochs is 5.



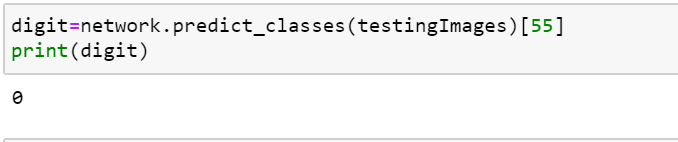
### **Training Accuracy**

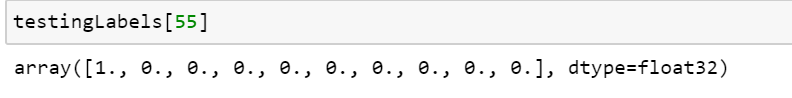
The training accuracy of our project is 96%

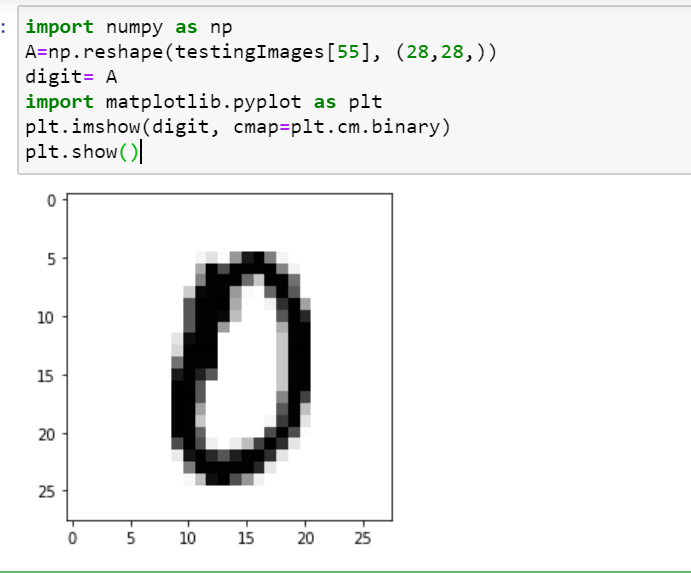


## **Predictions**

Now the model is fitted, we can used predict method to make predictions using new images. For this first we preprocess the images before we pass them to the predict method. To achieve this we use some functions from numpy. We use the *predict* method to predict which class the image belongs to.







# **Conclusion**

In this project, we used the handwritten dataset from mnist. The model we build did the image classification. For this purpose all we have to do is to put the training images and testing images in separate folders. We passed our images through 3 different layers having different activation functions. After this we compiled, fitted (trained) the model and checked the accuracy of the model which turned out to be 96%. We performed predictions on our model which were correct and helped us achieve the goal of our project.

The project is successfully made and helped us have a better understanding of CNN classification.