**BUILDING A CLASSIFIER FOR MNIST DATASET USING CONVOLUTIONAL NEURAL NETWORK**

Emeka David Amadi

Introduction to Data Science & Machine Learning (CSC-559-M01)

Computer Science and Engineering,

South Dakota School of Mines & Technology, Rapid City, SD 57701, USA

**INTRODUCTION**

This project aims to build a classifier and train a machine learning model to be able to classify and solve the rotated MNIST dataset using a Convolutional Neural Network approach.

We have a MNIST dataset containing 70,000 28x28 images showing handwritten digits.

First, we load our dataset using Keras and split into train and test sets.

Text, application

Description automatically generated

X\_train and x\_test contains our train and test images. Y\_train and y\_test contains our targets (labels) which are numbers between 0 and 9 that indicates the digit shown in the corresponding image.

X\_train has a total of 60,000 28x28 pixel images to train the model, and x\_test has 10000 28x28 images to test it.

The problem involves building a classifier for the rotated dataset images, therefore I use the **rotate** function from **scipy.ndimage** to rotate the x\_train and x\_test datasets because these datatset are what we will use to train our model and test our model while the y\_train are our targets.

I rotate the datatset images and convert them into a numpy array and reshaping it back to a 28x28 pixel image, then normalizing the pixel scale and

A picture containing text

Description automatically generated

Because we are dealing with an image dataset, we must take into consideration the shape of the tensor that we will build. Ideally, a tensor with 4 dimensions is suitable for our model capturing the batch size, width, height, and color channels.

Therefore, there is need to reshape our train set (x\_train\_set) to accommodate the color channel because they are grayscale images and we normalize the pixels because neural networks works much better with smaller values (dividing each pixel by 255 to get 0-1 pixels) and selecting a random image from the x\_test\_rotated dataset and plot it to see the rotated image.

The targets (labels) are values from 0-9 indicating our expected output and we use the one-hot encoding for these values for our targets.



Now, we define the architecture of our model. We are using the Sequential API from Keras which will be a sequence of layers.

Defining our input shape is very essential when creating our model. Using a model input with shape 28x28x1 tensor (width, height, color channels). We define our first layer as a Conv2D layer with 32 filters and a 3x3 kernel which will generate 32 different representations using the train set.

Next, we define the activation function for this layer as ReLU which will allow us to solve non-linear problems. We define a max-pooling operation which its purpose is to down sample the amount of information that is bring collected by the convolutional layer (discarding unimportant details and retaining the details required for our training).

Next, I used a Flatten layer to flatten the output because I want everything in a continuous list of value (flat tensor). Dense layers were also added and the final output layer has a size of 10 which represents one for each possible value and a softmax activation to ensure the probability distribution that will indicate the most likely digit in the image.

Graphical user interface, text, application

Description automatically generated

After creating the model, I compile it and use the Stochastic Gradient Function (SDG) as the optimizer. The loss is categorical cross-entropy because this is a multi-classification problem.  
I record the accuracy of the model as it trains using the metrics.

Graphical user interface, text

Description automatically generated

Next, I fit the model and it starts to train using a batch size of 32 and training over 3 epochs.

Text

Description automatically generated with low confidence

Finally, I predict the value of the image that was randomly selected from the x\_test\_rotated image dataset. The result is a one-hot encoded vector. Therefore, I perform the argmax value to get the position with the highest probability and return it as the result.

Graphical user interface, text, application, email

Description automatically generated