Handwritten Digit Recognition Using Convolutional Neural Networks

Elif Duran
[1] Department of Computer Engineering, Eskişehir Technical University, Eskişehir
elif_duran@eskisehir.edu.tr

***Abstract*— A handwritten Digit Recognition problem is the most curial problem of machine learning and computer vision applications nowadays. The goal of my project is that creating a model which helps to identify the handwritten digit numbers with high accuracy. While completing my project, I have used Convolutional Neural Networks and MNIST dataset. I have chosen 2 for epochs to reach high accuracy and get less losses while I have fitted my Sequential model.**

Keywords-component; MNIST dataset; Convolutional Neural Networks; Tensorflow; Keras

I. INTRODUCTION

Nowadays, Convolutional Neural Networks is the most popular approach to deal with machine learning problems such as challenge of image processing, classification, speech recognition and object detection. For this reason, I have used CNN to identify handwritten digits with less training time. I have preferred to load MNIST dataset which contains 60.000 images to work on it. I have chosen one of the images to visualize within MNIST dataset. 6666 as an image index is corresponded to ''seven'' in dataset that I have loaded. I have chosen ''seven'' but it is possible to work with other digits in project. We just need to decide which digit we want to recognize in project. There are many applications of handwriting digit recognition in our real life. We can use it in banks for reading checks, post offices for sorting letter, and many other related works.

A. *MNIST Database*

The MNIST database of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. The images are of size 28*28 pixels.

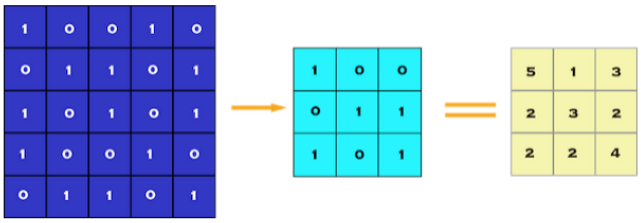
B. *Convolutional Neural Network*

Convolutional neural networks are deep artificial neural networks. We can use it to classify images, cluster them by similarity and perform object recognition within scenes. Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters Pooling, fully connected layers and apply Softmax function to classify an object with probabilistic values between 0 and 1.

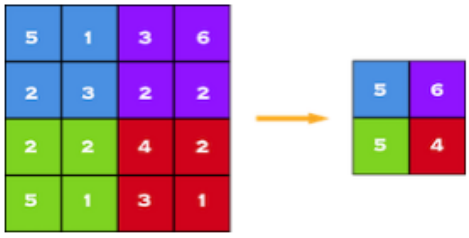
Identify applicable sponsor/s here. *(sponsors)*

II. ARCHITECTURE OF CNN

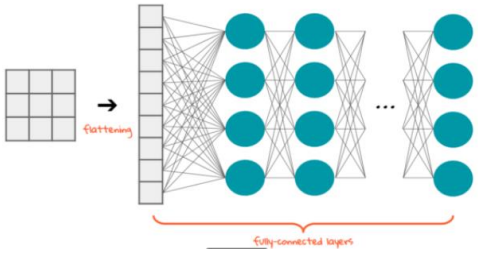
The *convolutional layer* is the first layer which can extract features from the images. Each filter extracts different features from the image. It has been preferred 3x3 filter size for kernel filters over 28x28 size input images in project. It is common to insert *pooling layers* after each convolution layer to reduce the spatial size of the features maps. Pooling layers also help with the overfitting problem.



Max pooling is the most popular pooling approach for Convolutional Neural Networks. It selects the maximum element from the region of the feature map covered by the filter. It has been achieved by using MaxPooling2D layer in project.



Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. I flatten the output of the convolutional layers to create a single long feature vector.



At the same time, I have added *dropout layer* because dropout layers fight with the overfitting by disregarding some of the neurons while training while Flatten layers flatten 2D arrays to 1D arrays before building the fully connected layers.

After all, we can construct a *fully connected network* in the end to classify the images. I have preferred softmax transfer function for output layer to get two outcomes.

III. IMPLEMENTATION

The steps which are taken by me are shown in below:

1. Firstly, I have loaded MNIST dataset via keras.
2. I have identified image index that I want to work on it.
3. I have printed the digit which is corresponded image index that I have chosen in previous step.
4. Then, I have visualized the number via matplotlib library.
5. I have printed the size of images to be sure that it is 28x28 size images like it has mentioned in MNIT dataset.
6. I have realized that I have 3D arrays but I have reshaped it to 4D arrays to work with keras.
7. I have controlled that the values are float to get decimal points after division.
8. I have normalized the RGB codes to maximum RGB codes.
9. Then, I have printed new shape of x_train images , number of x_train which is 60000 and the number of x_test which is 10000.
10. After I have completed reshape and normalization of images, I have created Sequential Model thanks to Keras library.
11. I have added Convolution layer for 2D thanks to Conv2D(), 28 is the output filters in convolution, 3x3 is the kernel size for height and width of 2D convolution windows.
12. I have added Pooling layer for 2D and I have preferred max pooling approach for pooling layer thanks to MaxPooling2D. 2x2 is the pool size for max pooling operation.
13. I have added Flatten layer to converts the pooled feature map to a single column that is passed to the fully connected layer.
14. I have added fully connected layer to the neural network thanks to Dense. 128 is the number of nodes in hidden layer and relU is an activation function in hidden layer. RelU is the most used activation function for many kinds of neural networks.
15. I have prevented overfitting with Dropout layer with 0.2 rate.
16. I have added output layer thanks to Dense. We deal with digits from 0 to 9. So i need to 10 neurons in output layer. I have chosen softmax activation function to get two outcomes.
17. Then I have compiled CNN using compile function with three parameters; optimizer, loss function, the metrics of performance. Optimizer is the gradient descent algorithm. Loss function that I have preferred computes the crossentropy loss between the labels and predictions.
18. Then, I have fitted my model by using train data. 2 epochs are the number of complete passes through the training dataset.
19. Lastly, I have evaluated trained model with x_test and y_test.

IV. RESULTS

After I run prediction.py, we can see the results for our model.

```
[0.08622708616983145, 0.973]
```

Loss and Accuracy for epoch is equal to 1

```
[0.07263960417844355, 0.9767]
```

Loss and Accuracy for epoch is equal to 2

```
[0.057632365197315814, 0.9812]
```

Loss and Accuracy for epoch is equal to 3

```
[0.05409538557237247, 0.9853]
```

Loss and Accuracy for epoch is equal to 5

```
[0.05699737581011614, 0.9851]
```

Loss and Accuracy for epoch is equal to 7

```
[0.061677003711520045, 0.9852]
```

Loss and Accuracy for epoch is equal to 10

V. REQUIREMENTS

It has to installed tensorflow 2.1.0, keras 2.3.1 and matplotlib 3.1.1 libraries. It has been reached MNIST dataset via tensorflow.keras.datasets.