Name: Shahriar Farabi

Id: 18-39140-3

Section: B

Course name: COMPUTER VISION AND PATTERN RECOGNITION

Course teacher name: Dr. Debajyoti Karmaker

Midterm Project Report

Semester: Fall 21-22

Implement a CNN architecture to classify the MNIST handwritten dataset

Contents

Abstract

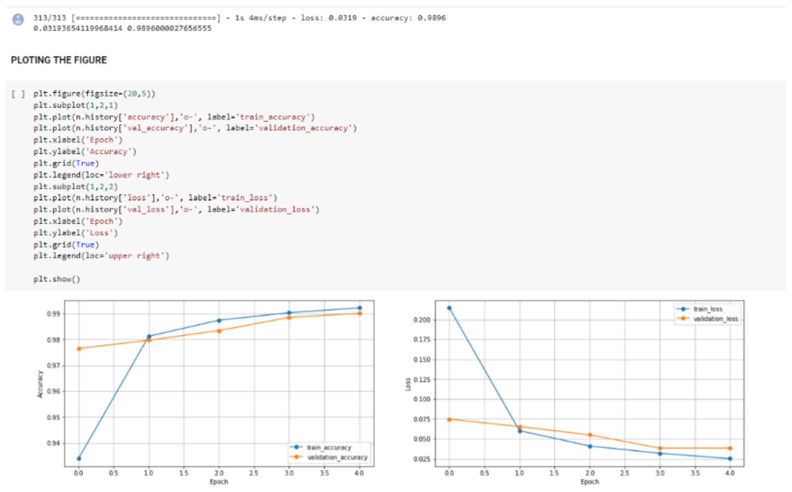
A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g., holding the class scores) through a differentiable function. A few distinct types of layers are commonly used. The MNIST handwritten digit classification problem is a standard dataset used in computer vision and deep learning. Humans can see and visually sense the world around them by using their eyes and brains. Computer vision works on enabling computers to see and process images in the same way that human vision does. Several algorithms developed in the area of computer vision to recognize images. The goal of our work will be to create a model that will be able to identify and determine the handwritten digit from its image with better accuracy. We aim to complete this by using the concepts of Convolutional Neural Network and MNIST dataset. We will also show how MatConvNet can be used to implement our model with CPU training as well as less training time. Though the goal is to create a model which can recognize the digits, we can extend it for letters and then a person's handwriting. Through this work, we aim to learn and practically apply the concepts of Convolutional Neural Networks.

Introduction

Convolutional neural networks (CNN) – the concept behind recent breakthroughs and developments in deep learning. CNNs have broken the mold and ascended the throne to become the state-of-the-art computer vision technique. Among the different types of neural networks (others include recurrent neural networks (RNN), long short-term memory (LSTM), artificial neural networks (ANN), etc.), CNNs are easily the most popular. These convolutional neural network models are ubiquitous in the image data space. They work phenomenally well on computer vision tasks like image classification, object detection, image recognition, etc. MNIST (Modified National Institute of Standards and Technology) is a well-known dataset used in

Computer Vision that was built by Yann Le Cun et. al. It is composed of images that are handwritten digits (0-9), split into a training set of 50,000 images and a test set of 10,000 where each image is of 28 x 28 pixels in width and height. This dataset is often used for practicing any algorithm made for image classification as the dataset is fairly easy to conquer. Hence, I recommend that this should be your first dataset if you are just foraying in the field.

Results

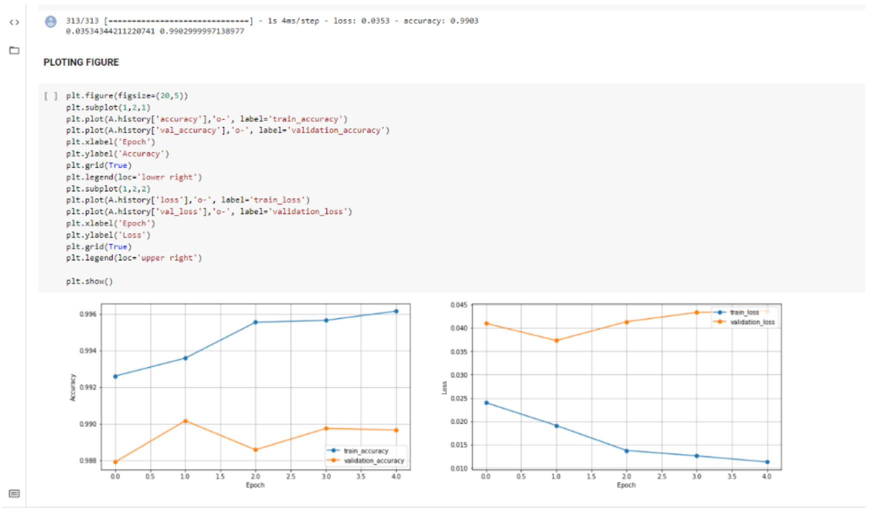


```
313/313 [==============================] - 1s 4ms/step - loss: 0.0319 - accuracy: 0.9896
0.03193654119968414 0.9896000027656555
```

PLOTING THE FIGURE

```
[ ] plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    plt.plot(n.history['accuracy'],'o-', label='train_accuracy')
    plt.plot(n.history['val_accuracy'],'o-', label='validation_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.grid(True)
    plt.legend(loc='lower right')
    plt.subplot(1,2,2)
    plt.plot(n.history['loss'],'o-', label='train_loss')
    plt.plot(n.history['val_loss'],'o-', label='validation_loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid(True)
    plt.legend(loc='upper right')

    plt.show()
```

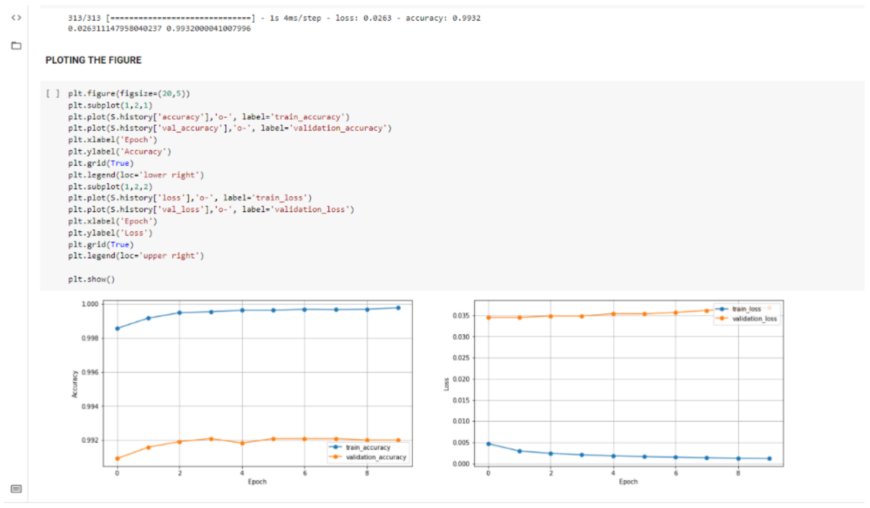


```
313/313 [==============================] - 1s 4ms/step - loss: 0.0353 - accuracy: 0.9903
0.03534344211220741 0.9902999997138977
```

PLOTING FIGURE

```
[ ] plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    plt.plot(A.history['accuracy'],'o-', label='train_accuracy')
    plt.plot(A.history['val_accuracy'],'o-', label='validation_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.grid(True)
    plt.legend(loc='lower right')
    plt.subplot(1,2,2)
    plt.plot(A.history['loss'],'o-', label='train_loss')
    plt.plot(A.history['val_loss'],'o-', label='validation_loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid(True)
    plt.legend(loc='upper right')

    plt.show()
```



```
313/313 [==============================] - 1s 4ms/step - loss: 0.0263 - accuracy: 0.9932
0.026311147958040237 0.9932000041007996
```

PLOTING THE FIGURE

```
[ ] plt.figure(figsize=(20,5))
    plt.subplot(1,2,1)
    plt.plot(S.history['accuracy'],'o-', label='train_accuracy')
    plt.plot(S.history['val_accuracy'],'o-', label='validation_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.grid(True)
    plt.legend(loc='lower right')
    plt.subplot(1,2,2)
    plt.plot(S.history['loss'],'o-', label='train_loss')
    plt.plot(S.history['val_loss'],'o-', label='validation_loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid(True)
    plt.legend(loc='upper right')

    plt.show()
```

Discussion

We got the results using the three types of optimizers to solve the picture classification problem with handwritten MNISY data set. ADAM optimizer is used at first which gave 98.96% accuracy for MNIST dataset. Using SGD, the accuracy obtained is 99.03% and test loss is 3.53%. Lastly from RSMProp optimizer test accuracy obtained is 99.32% and test loss is 2.63%. From here it is visible that maximum accuracy is obtained from RMSProp optimizer. On the other hand, using SGD minimum accuracy is obtained. Therefore, we can easily get faster and high accuracy from RMSProp optimizer compared to ADAM and SGD optimizer.