Handwritten Digit Recognizer using Deep Learning

Project Overview

Business Overview

The Modified National Institute of Standards and Technology dataset popularly known as MNIST dataset is widely used as a standard dataset in deep learning. The dataset was released in 1999. The classic dataset serves as a basis for testing most of the classification algorithms. In the emerging field of machine learning the MNIST dataset remains one of the reliable resources to test the new techniques. The training dataset contains 60,000 and the testing dataset contains 10,000 grayscale images of handwritten digits between 0 to 9 each of 28x 28 pixels.

Aim: To correctly identify handwritten digits from the MNIST dataset.

Data Description

The Modified National Institute of Standards and Technology dataset popularly known as MNIST dataset is widely used as a standard dataset in deep learning. The dataset was released in 1999. The training dataset contains 60,000 and the testing dataset contains 10,000 grayscale images of handwritten digits between 0 to 9 each of 28x 28 pixels.

Tech Stack

- ➔ Language : Python
- ➔ Libraries : pandas, numpy, tensorflow, matplotlib, seaborn, scikit-learn

Approach

1. Data visualization

2. Data preprocessing
    a. Reshaping data
    b. One hot encoding
    c. Feature Scaling

3. CNN model creation and building on training data

4. Model validation
    a. Confusion matrix
    b. Classification report
    c. Loss over training and validation dataset
    d. Accuracy over training and validation dataset

5. Model prediction over individual images

Modular code overview

```
lib
  |_ Handwritten_Digit_Recognizer.ipynb

src
  |_engine.py
  |_ ML_pipeline
            |_ model_evaluation.py
            |_ model.py
            |_ preprocessing.py

output
  |_ mnist_cnn_model.h5
  |_ mnist_cnn_model.json

requirements.txt
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. src
2. output
3. Lib
4. requirements.txt

1. The src folder is the heart of the project. This folder contains all the modularized code for all the above steps in a modularized manner. It further contains the following.
    a. ML_pipeline
    b. engine.py

The ML_pipeline is a folder that contains all the functions put into different python files which are appropriately named. These python functions are then called inside the engine.py file

2. The output folder contains the fitted CNN model that we trained for this data. This model can be easily loaded and used for future use and the user need not have to train the model from the beginning.

3. The lib folder is a reference folder. It contains the original ipython notebook that we saw in the videos.

4. The requirement.txt file has all required packages that need to be installed for this project. The packages can be installed by “pip install -r requirements.txt” command in the python interpreter.

Note: As the data is directly available in tensorflow library, there is no input folder.

Project Takeaways

1. Understanding the problem statement
2. Importing the dataset and libraries
3. Data Visualization
4. Data Preprocessing
5. One Hot Encoding
6. Feature Scaling
7. Introduction to CNN model
8. Building CNN model
9. Model Performance during training and validation
10. Model Evaluation
11. Model Prediction
12. Visualization of Confusion matrix
13. Predictions over individual images
14. Making Production ready code
15. Cloud AI services
16. Introduction and demonstration of Google Vision API
17. Introduction and demonstration of Amazon Rekognition
18. Introduction and demonstration of Azure Computer Vision