Build CNN Image Classification Models for Real Time Prediction

Overview

Convolutional Neural Networks are part of deep neural networks which makes working with images and videos pretty easy. Convolutional Neural Networks capture useful information from images which helps to correctly classify them.

This project will give you a brief idea about CNN and related concepts. It also builds a CNN model for image classification and the model will also be tested for real-time prediction. If you haven't already visited, here is the previous project of the series Build Deep Autoencoders Model for Anomaly Detection in Python.

Aim

- To understand the basic ideas related to CNN
- To develop a Convolutional Neural Network model to classify images into different classes
- To deploy the model using Flask

Why use CNN?

Before CNN came into existence, image classification was a very hectic and time-consuming job as features for images needed to be created and fed to the model. Whereas CNN uses automatic feature extraction using convolution to make a feature map containing useful information of the image which is used by CNN to classify the images.

When to use CNN?

- Image Classification
- Image Segmentation
- Video Analysis
- Object Detection

Prerequisites

- Tensorflow
- Flask
- Deep learning and Activation functions
- Build Deep Autoencoders Model for Anomaly Detection in Python

Tech Stack

- ➔ Language: Python
- ➔ Libraries: tensorflow, pandas, matplotlib, keras, flask, pathlib

Data Description

Dataset used in this project are images of driving license, social security, and others categorized into respective categories. The images are of different shapes and sizes which are preprocessed before modeling.

Approach

1. Data loading
2. Data Preprocessing
3. Model building and training
4. Data Augmentation
5. Deployment

Modular Codes Overview:

```
input
  |_Testing_Data
  |_Training_data

output
  |_api_input
  |_cnn-model.h5

src
  |__init__.py
  |_Engine.py
  |_ ML_pipeline
        |__init__.py
        |_deploy.py
        |_Preprocess.py
        |_Train_Model.py
        |_Utils.py
        |_wsgi.py
        |_wsgi.sh

Convolution-Neural-Network.ipynb
Model_Api.ipynb
README.md
requirements.txt
```

Once you unzip the modular_codes.zip file the following folder can be found

1. input
2. output
3. src
4. Convolutional-Neural-Network.ipynb
5. Model_Api.ipynb
6. README.md
7. requirements.txt

1. The input folder contains the training and testing folder. The training and testing are further divided into three folders namely driving license, social security, and others which contains the respective images.

2. 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 includes 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.

3. The output folder contains the image for the testing and the cnn-model.h5 file which is saved model after the training.

4. The requirements.txt file has all the required libraries with respective versions. Kindly install the file by using the command **pip install -r requirements.txt**

5. **All the instructions for running the code are present in readme.md file**

Project Takeaways

1. What is CNN?
2. Deep Neural Network vs Convolutional Neural Network
3. What is Kernel in CNN?
4. Understanding CNN architecture
5. What is the use of CNN?
6. What is Pooling and Padding?
7. What is the use of Pooling and Padding?
8. What is Convolution?
9. What is Tensorflow?
10. What is a feature map?
11. What is Data Augmentation?
12. How to load the data using tensorflow?
13. How to build a CNN model using tensorflow?
14. How to do data preprocessing?
15. How to build an API using Flask?
16. How to do real time prediction using Unicorn Platform?