CLOUD APPLICATION DEVELOPMENT

Image Recognition with IBM Cloud Visual Recognition

PHASE 4: DEVELOPMENT PART 2

Introduction:

In this second development phase, we venture into the integration of face detection and emotion recognition, forming a comprehensive image recognition system.

At the heart of this endeavour lies the "Haar Cascade Classifier," a cutting-edge technology in computer vision primarily designed for detecting frontal faces within images.

Its role in identifying and locating faces is pivotal, providing a foundation for subsequent emotion recognition tasks.

The phase also involves a series of code snippets designed to facilitate the development of this system.

It begins with the preparation of a machine learning model for facial emotion recognition, configuring data generators, creating and fine-tuning the deep learning model, and preparing it for training.

Users have the option to select from a range of pre-trained deep learning models for emotion recognition, each with its unique architecture.

The training and evaluation process monitors the model's performance, with early stopping mechanisms and insightful visualizations of accuracy and loss.

To implement the image classification process using the IBM Cloud Visual Recognition API, you can follow these

Steps:

Sign up for IBM Cloud: **Go to the IBM Cloud website and create an account if you don't have one already. Note**

That there may be a free tier available which you can use for testing and small-scale implementation.

Create a Visual Recognition Service: **Once logged in to IBM Cloud, navigate to the "Catalogue" and search for**

"Visual Recognition" in the search bar. Then, select the Visual Recognition service from the options and create it.

Get API Key: **After the service is created, click on it from the "Cloud Services" section in your IBM Cloud**

Dashboard. Then, go to the "Service Credentials" tab and click "View Credentials" to get your API key. Make sure to

Save the API key as it will be needed for authentication.

Install IBM Watson Python SDK: Open your Python IDE or terminal and install the IBM Watson Python SDK using

The following command:

```
Pip install ibm-watson
```

~~~

Import dependencies and initialize the Visual Recognition service:

```python

From ibm_watson import VisualRecognitionV3

Apikey = 'YOUR_API_KEY'

Version = '2018-03-19'

Visual_recognition = VisualRecognitionV3(version=version, iam_apikey=apikey)

Classify an Image: To classify an image, you can use the

`classify()` method provided by the Visual Recognition service.

Here's an example:

```python

With open('image.jpg,' 'rb') as image_file:

Classes = visual_recognition.classify(images_file=image_file,

Threshold='0.6,' classifier_ids='default').get_result()

Print(classes)
~~~

```
```

Replace `’image.jpg’` with the path to your image file. The

`threshold` parameter is used to filter the results based on their

Confidence score.

Analyze the Result:

The `classify()` method will return a JSON

Response containing the results of the image classification. You

Can extract and use the information as needed, such as the class

Labels and their associated confidence scores.

These are the basic steps to start the image classification process

Using the IBM Cloud Visual Recognition API. You can explore more

Advanced features and customize the process based on the API

Documentation and your requirements.

Data Sources and Setup

Before running the scripts for the image recognition project, it’s crucial to set up the required datasets. This document provides an overview of the datasets used in our project and how to obtain them.

Datasets:

CK+ (Cohn-Kanade Extended+):

Source:

https://www.kaggle.com/datasets/shawon10/ckplus

Description: The CK+ dataset contains facial expressions captured in lab-controlled environments. It includes seven different emotion labels, making it suitable for training and testing emotion recognition models.

FER-13 (Facial Expression Recognition 2013):

Sources:

https://www.kaggle.com/datasets/msambare/fer2013

https://www.kaggle.com/datasets/deadskull7/fer2013

Description: The FER-13 dataset is a collection of images representing facial expressions. It contains various emotional states, enabling comprehensive training and testing for emotion recognition.

FERPlus:

Source:

https://github.com/microsoft/FERPlus

Description: FERPlus is an extension of the FER-13 dataset, providing a more refined annotation of emotions. It includes additional labels, offering improved granularity in emotion recognition.

Data Setup:

To run the image recognition scripts successfully, follow these steps:

Download the CK+, FER-13, and FERPlus datasets from their respective sources.

Organize the dataset files according to your project's directory structure.

Data Preprocessing for Emotion Recognition Model

In the field of emotion recognition using deep learning, data preprocessing plays a pivotal role in shaping the effectiveness of the models.

This document outlines essential data preprocessing steps to prepare the FERPlus dataset for training emotion recognition models.

Data Cleaning and Transformation:

Read and Clean CSV:

The process begins with reading the FERPlus dataset's CSV file, which contains labels and information about the images.

Any rows with missing values (NaN) are removed to ensure data integrity.

Mapping Emotions:

The FERPlus dataset provides emotion labels in a detailed format.

Emotions are mapped into seven primary categories: neutral, happy, surprise, sad, angry, disgust, and fear.

Data Reorganization:

Transfer Images:

Images are transferred from the original FERPlus directory structure to match the FER-2013 structure.

Images are categorized into training and test sets based on the "Usage" attribute in the CSV file.

Emotion-Based Sorting:

Images are further sorted into subfolders within the training and test sets based on the dominant emotion category they represent.

Execution:

The Python script provided automates the data preprocessing steps mentioned above. It ensures that the FERPlus dataset aligns with the FER-2013 dataset's structure and emotion categories.

```
Import os
Import shutil

Import cv2

Import numpy as np

Import pandas as pd


Def get_best_emotion(list_of_emotions, emotions):

   Best_emotion = np.argmax(emotions)

   If best_emotion == “neutral” and sum(emotions[1::]) > 0:

      Emotions[best_emotion] = 0

      Best_emotion = np.argmax(emotions)

   Return list_of_emotions[best_emotion]


Def read_and_clean_csv(path):

   # we read the csv and we delete all the rows which contains NaN

   Df = pd.read_csv(path)

   Df = df.dropna()

   Return df


Def rewrite_image_from_df(df):

   Print(“Moving images from FERPlus inside FER-2013”)

   # we setup an accumulator to print if we have finished a task

   Acc = “”

   Emotions = [

      “neutral”,

      “happy”,

      “surprise”,

      “sad”,
      “angry”,
      “disgust”,
      “fear”,
      “contempt”,
      “unknown”,
      “NF”,
   ]
  # we rewrite all the image files
   For row in range(len(df)):
      Item = df.iloc[row]
      If item[“Usage”] not in [“”, acc]:
          Print(f”{item[‘Usage’]} done”)
      If (item[‘Usage’] == “Training”):
          Image = cv2.imread(f”./FERPlus/output/FER2013Train/{item[‘Image
name’]}”)
      Elif item[‘Usage’] == “PublicTest”:
          Image = cv2.imread(f”./FERPlus/output/FER2013Valid/{item[‘Image
name’]}”)
      Else:
          Image = cv2.imread(f”./FERPlus/output/FER2013Test/{item[‘Image
name’]}”)
      Acc = item[“Usage”]
      If acc == “Training”:
          Cv2.imwrite(
              F”./FER-2013/train/{get_best_emotion(emotions,
item[2::])}/{item[‘Image name’]}”,
              Image,
          )
    Else:
        Cv2.imwrite(
            F"./FER-2013/test/{get_best_emotion(emotions,
item[2::])}/{item['Image name']}",
            Image,
        )
If __name__ == "__main__":
  Os.system('python ./FERPLUS/src/generate_training_data.py -d
./FERPLUS/output -fer ./FER-2013/fer2013.csv -ferplus
./FERPLUS/fer2013new.csv')
  Df = read_and_clean_csv("./FERPlus/fer2013new.csv")
  Rewrite_image_from_df(df)
```

Feature Engineering:

For image recognition, this often involves pre-processing images (resizing, normalization, etc.) and extracting features. However, with deep learning, this step is typically handled by the neural network.

Model Evaluation:

Use a validation dataset (separate from training data) to assess the performance of your model.

Metrics like accuracy, precision, recall, and F1-score can be used to evaluate its effectiveness.