Product recommendation using Image Search
Project Overview

Business Objective

We are all aware of how online shopping and e commerce is growing rapidly. Hence, it is imperative for computer vision systems to automatically and accurately recognize products based on images at the stock keeping unit (SKU) level. This project mainly focuses on meeting this market need. The core idea of this project is search and find images of products similar to any given image of a product.

Goal

To find images similar to any given image from the database

Tech Stack

- ➔ Language : Python
- ➔ Cloud support : AWS
- ➔ Libraries : Elasticsearch, Tensorflow, Keras, Numpy, Pandas, Requests, Scikit-learn

Data Overview

The dataset includes images from 2,019 product categories with one ground truth class label for each image. It includes a total of 1,011,532 images for training, 10,095 images for validation and 90,834 images for testing. Train and validation sets have the same format as shown below

```
{
    "images": [
        {
            "id" : string,
            "url": string,
            "class": int
        },
        ...
    ]
}
```

id : A unique id given to each image
url : The url to download the image
class : The class to which the image belongs to

It is to be noted that for each image,only the URL is provided. Users need to download the images by themselves. It is also to be noted that the image URLs may become unavailable over time.

Data Source : https://www.kaggle.com/c/imaterialist-product-2019/overview

Approach

1. Download images from label_id
   Downloading all the images using the given URLs of images.
2. Indexing using ElasticSearch
   Feature extraction is done using the weights of imagenets from MobileNetV2
3. Image2Image Query
   Use K Nearest Neighbour in Elastic search to find K nearest vectors which are having maximum similarity for the queried image.

Modular code overview

```
input
  |__train.json
src
  |__engine.py
  |__ML_pipeline
           |__downloading_images.py
           |__elasticsearch_indexing.py
           |__feature_extraction.py

lib
  |__notebooks
           |__Downloading-Images.ipynb
           |__elasticsearch-indexing.ipynb
           |__Feature-Extraction.ipynb
           |__test-ImageSearch.ipynb
  |__references
output
  |__12
  |__response.json
```

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

1. input
2. src
3. output
4. lib
5. requirements.txt - list of all the packages used in the project module.

1. The input folder contains all the data that we have for analysis which in our case is train.json
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 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

3. The output folder contains all the images that we have scraped from each url.
4. The lib folder is a reference folder. It contains two folders namely
    a. Resources - Contains all the learning resources
    b. Notebooks - Contain the jupyter notebook of the project

Project Takeaways

1. KNN Overview
2. Higher Dimensional Database - Overview
3. ANN BenchMarks and libraries of HDDB
4. Downloading Imaterialist using Python Script
5. Understanding MobileNet Architecture
6. Understanding Feature Extraction
7. Setting up ElasticSearch with a plugin for KNN
8. How to connect to ElasticSearch using Python
9. Indexing Using ElasticSearch with Python
10. Querying ElasticDb over Knn with Python
11. ElasticSearch API in action and understanding ImageSearch Response