



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

DEEP LEARNING - PRINCIPLES AND PRACTICES (CSE4088)

DIGITAL ASSIGNMENT 2

Team Members:

Madhuvanthi Sankar Ganesh (20BRS1113)

Pooja R (20BRS1091)

Title: Visual Product Recognition Challenge

GitHub link:

<https://github.com/pooja-172/Visual-Product-Recognition---Deep-Learning>

Competition link:

<https://www.aicrowd.com/challenges/visual-product-recognition-challenge-2023>

Introduction:

In retail, there has always been a need to improve customers' shopping experience and automate corporate procedures. Retail operations and customer experiences may be enhanced significantly by incorporating product identification capabilities in businesses. This solution can be used for multiple applications like identifying which products are in high demand, detecting counterfeit products, stock/inventory updation, and so on. The main components required to design a product recognition system are object detection and image recognition.

Transfer learning and **ResNet** are two popular techniques in deep learning that have been used successfully for product recognition. Transfer learning is the process of

leveraging the knowledge acquired while solving one problem to solve a distinct but related problem. Transfer learning in deep learning involves adapting a neural network trained on a large dataset to a new task by fine-tuning its weights on a smaller dataset. By doing so, the network can learn more efficiently and effectively, as it has already learned useful features from the large dataset that can be applied to the new task.

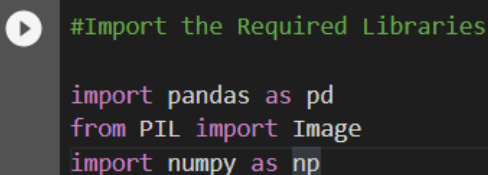
ResNet, or residual network, is a type of architecture for deep neural networks that facilitates the training of networks that are significantly deeper than was previously feasible. ResNet introduces a new type of layer known as a residual block, which enables the direct transfer of data between earlier and later layers. This alleviates the vanishing gradient problem that can occur in very deep networks when gradients become too small to effectively update weights.

Transfer learning and ResNet can be combined to increase accuracy and reduce training time for product recognition. To recognize specific products, for instance, a pre-trained ResNet model can be fine-tuned on a reduced dataset of product images. The pre-trained ResNet model has already learned the general features of images, such as edges, corners, and textures, which can be useful for recognizing products. A smaller dataset can help the model acquire more particular characteristics pertinent to the goods being recognized.

Overall, the combination of transfer learning and ResNet is a powerful technique for product recognition, enabling high accuracy while reducing training time and computational resources.

Implementation:

1) Importing the necessary libraries:

A code editor snippet with a dark background and light-colored text. It shows the import of several Python libraries: pandas as pd, PIL Image, and numpy as np. The text is color-coded: comments are green, and code is white. A play button icon is visible on the left side of the code block.

```
#Import the Required Libraries  
  
import pandas as pd  
from PIL import Image  
import numpy as np
```

2) Reading the Datasets:



#Read the Datasets

```
gallery_df = pd.read_csv('gallery.csv')
queries_df = pd.read_csv('queries.csv')
```

3)Unzipping the gallery and queries folder



```
!unzip gallery.zip
```



Archive: gallery.zip

```
inflating: gallery/abiding-debonair-viper-of-downpour.jpg
inflating: gallery/able-smoky-echidna-of-discourse.jpg
inflating: gallery/aboriginal-maroon-flamingo-of-criticism.jpg
inflating: gallery/aboriginal-organic-sturgeon-of-fury.jpg
inflating: gallery/aboriginal-rational-kiwi-of-priority.jpg
inflating: gallery/abstract-maize-cuscuta-of-growth.jpg
inflating: gallery/accelerated-enthusiastic-pegasus-of-attraction.jpg
inflating: gallery/accelerated-merciful-mongrel-of-exercise.jpg
inflating: gallery/accomplished-meaty-ladybug-from-hell.jpg
inflating: gallery/accomplished-strict-copperhead-from-mars.jpg
inflating: gallery/accurate-hypersonic-clam-of-philosophy.jpg
inflating: gallery/acrid-belligerent-mongrel-of-fantasy.jpg
inflating: gallery/acrid-horned-peacock-of-current.jpg
inflating: gallery/acrid-myrtle-tuatara-of-joy.jpg
inflating: gallery/acrid-optimal-mule-of-unity.jpg
inflating: gallery/acrid-prophetic-scallop-from-hell.jpg
inflating: gallery/acrid-vivacious-skink-of-happiness.jpg
inflating: gallery/active-almond-mastodon-of-democracy.jpg
inflating: gallery/active-flat-dragon-of-agreement.jpg
inflating: gallery/active-jade-hawk-of-radiance.jpg
inflating: gallery/active-wasp-of-awesome-tempest.jpg
inflating: gallery/adamant-belligerent-lionfish-of-popularity.jpg
inflating: gallery/adaptable-sepia-dragon-of-love.jpg
inflating: gallery/adaptable-striped-dugong-of-virtuosity.jpg
inflating: gallery/adept-resourceful-antelope-of-serendipity.jpg
inflating: gallery/adorable-kestrel-of-phenomenal-relaxation.jpg
inflating: gallery/adorable-lilac-agama-of-calibration.jpg
inflating: gallery/adorable-pygmy-poodle-of-variation.jpg
inflating: gallery/adventurous-orthodox-pig-of-sunshine.jpg
inflating: gallery/airborne-skinny-herring-of-action.jpg
inflating: gallery/airborne-steadfast-crab-of-superiority.jpg
inflating: gallery/alluring-quiet-peacock-of-temptation.jpg
```

```
!unzip queries.zip

Archive: queries.zip
  inflating: queries/abiding-inchworm-of-ultimate-freedom.jpeg
  inflating: queries/abiding-industrious-bullfinch-from-heaven.jpeg
  inflating: queries/abiding-tomato-oarfish-of-excellence.jpg
  inflating: queries/abiding-warm-buffalo-of-vitality.jpeg
  inflating: queries/abiding-warping-gibbon-of-acceptance.jpeg
  inflating: queries/able-cherubic-zebu-from-ganymede.jpeg
  inflating: queries/able-fervent-pheasant-of-feminism.jpeg
  inflating: queries/able-fluffy-labrador-of-philosophy.jpeg
  inflating: queries/able-fuzzy-ladybug-of-tempest.jpeg
  inflating: queries/able-natural-boa-of-force.jpeg
  inflating: queries/able-talented-cuttlefish-of-cookies.jpeg
  inflating: queries/able-tasteful-dalmatian-of-glory.jpeg
  inflating: queries/aboriginal-purple-viper-of-tempering.jpeg
  inflating: queries/aboriginal-sparkling-magpie-of-agility.jpg
  inflating: queries/abstract-radical-lorikeet-of-intensity.jpeg
  inflating: queries/accelerated-glorious-fennec-of-reward.jpg
  inflating: queries/accelerated-mysterious-chihuahua-of-love.jpeg
  inflating: queries/accomplished-clever-scallop-of-efficiency.jpeg
  inflating: queries/accomplished-savvy-dragon-of-success.jpeg
  inflating: queries/accomplished-talented-cricket-of-felicity.jpeg
  inflating: queries/accurate-skilled-mosquito-of-serendipity.jpeg
  inflating: queries/acoustic-brainy-lobster-from-hyperborea.jpeg
  inflating: queries/acoustic-civet-of-silent-chivalry.jpeg
  inflating: queries/acoustic-pompous-dugong-of-contentment.jpeg
  inflating: queries/acrid-fat-bullfrog-of-emphasis.jpeg
  inflating: queries/acrid-hospitable-toad-of-exercise.jpeg
  inflating: queries/acrid-spiked-gorilla-of-prestige.jpeg
  inflating: queries/acrid-thistle-hawk-of-tempest.jpeg
  inflating: queries/acrid-thundering-tortoise-of-tempest.jpeg
  inflating: queries/acrid-venomous-shrimp-of-might.jpeg
  inflating: queries/active-ambitious-quetzal-of-fury.jpeg
  inflating: queries/active-bright-cravfish-from-union.jpeg
```

4)Preprocessing the images from gallery and Query folder:

We are resizing the images to 224x224, converting the images to a numpy array and normalizing the pixel values to [0,1]

```
#Pre-process the Gallery Images - Resize the image to 224x224, Convert the image to a numpy array, Normalize pixel values to [0, 1]

gallery_images = []
for img_path in gallery_df['img_path']:
    img = Image.open(img_path)
    img = img.resize((224, 224))
    img = np.array(img)
    img = img / 255.0
    gallery_images.append(img)
gallery_images = np.array(gallery_images)

# Preprocess the query images - Resize the image to 224x224, Convert the image to a numpy array, Normalize pixel values to [0, 1]

query_images = []
for i, row in queries_df.iterrows():
    img = Image.open(row['img_path'])
    img = img.resize((224, 224))
    img = np.array(img)
    img = img / 255.0
    query_images.append(img)
query_images = np.array(query_images)
```

5)Extract the bounding boxes from the queries dataframe:

A bounding box is a rectangular structure superimposed over an image including all important features of a particular object residing in it. It is one of the simplest

and low time taking techniques of image annotation. The annotator outlines the objects of the images in a box as per the project requirements.

Its purpose is to reduce the range of search for the object features and thereby conserve computing resources. It not only helps to classify the objects but also helps in object detection.

```
#Extract the Bounding Boxes from the Queries Dataframe
query_boxes = queries_df[['bbox_x', 'bbox_y', 'bbox_w', 'bbox_h']].values
```

6) Feature Extraction based on pre-trained ResNet 50 model [ImageNet Database]

```
#Extract Features Based on Pre-trained ResNet50 Model [ImageNet Database]

import tensorflow as tf
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input

model = ResNet50(weights='imagenet', include_top=False, pooling='avg')

gallery_features = model.predict(preprocess_input(gallery_images))
query_features = model.predict(preprocess_input(query_images))

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [=====] - 1s 0us/step
34/34 [=====] - 199s 6s/step
61/61 [=====] - 365s 6s/step
```

7) Compute the cosine similarities between the query features and the gallery features (Cosine similarity is a similarity metric that can be used to find similarity between two images) and ranking them.

```
#Compute Cosine Similarities between the Query Features and the Gallery Features
from sklearn.metrics.pairwise import cosine_similarity

similarities = cosine_similarity(query_features, gallery_features)

#Rank the Gallery Images based on Similarity Scores

sorted_indices = np.argsort(similarities, axis=1)[: , ::-1]
ranked_results = sorted_indices[: , :1000]
```

8) Creating the csv file to submit to the contest



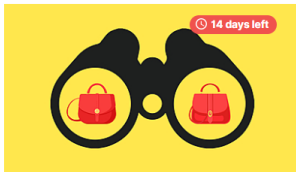
#Submission.csv

```
submission = np.zeros((len(queries_df), 1000))
for i, indices in enumerate(ranked_results):
    submission[i, :] = gallery_df.iloc[indices]['seller_img_id'].values



np.savetxt('submission.csv', submission, delimiter=',')
```

Submission:

CHALLENGES ENTERED



Visual Product Recognition Challenge 2023

By  Alcrowd  Machines Can See Summit
Identify user photos in the marketplace

LATEST SUBMISSIONS

submitted

213989

Sun, 2 Apr 2023 21:39:24

See All

