Image Search Engine

Rutvik Rupapara
Oct 10 · 2 min read

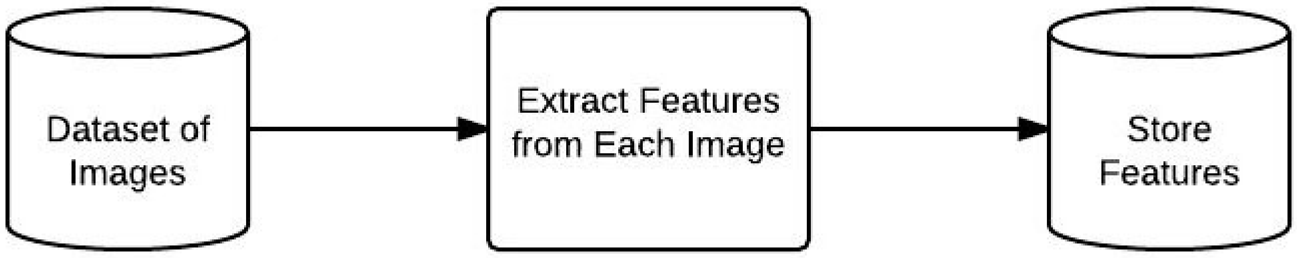
Image Search Types

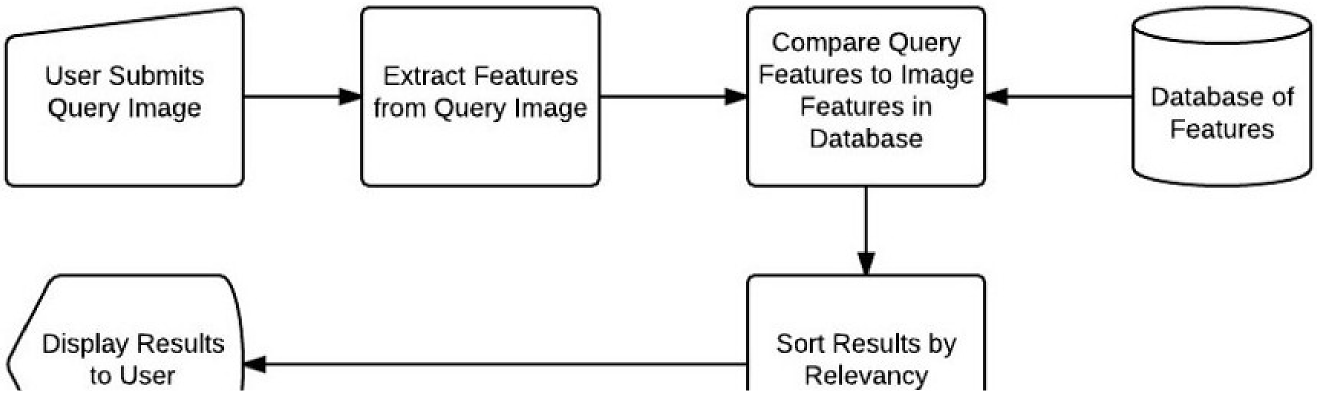
1. Search by Meta-Data
2. Search by Example

Image search engines that quantify the contents of an image are called **Content-Based Image Retrieval** (CBIR) systems.

The 4 Steps of Any CBIR System

1. Defining your image descriptor
2. Indexing your dataset
3. Defining your similarity metric
4. Searching




User Submits Query Image
Extract Features from Query Image
Compare Query Features to Image Features in Database
Database of Features
Sort Results by Relevancy
Display Results to User

1. **Import libraries**

2. Read Dataset

Now, we will convert the images in the dataset to feature vectors using Resnet50. We chose resnet because of the relatively small feature size. It converts images into 2048 convolutional features compared to 25088 features in vgg 19 or 51200 features in inception architectures. It will be easy for the nearest neighbor algorithm to find neighbors and also it will minimize the effects of Curse_of_dimensionality.

```
img_size =224

model = ResNet50(weights='imagenet', include_top=False,input_shape=
(img_size, img_size, 3),pooling='max')
```

Here are creating a Keras image data generator object and preprocessing the images.

```
batch_size = 64
root_dir = '101_ObjectCategories'

img_gen =
ImageDataGenerator(preprocessing_function=preprocess_input)

datagen = img_gen.flow_from_directory(root_dir,
                                        target_size=(img_size,
img_size),
                                        batch_size=batch_size,
                                        class_mode=None,
                                        shuffle=False)

num_images = len(datagen.filenames)
num_epochs = int(math.ceil(num_images / batch_size))

feature_list = model.predict_generator(datagen,
num_epochs)print("Num images   = ", len(datagen.classes))
print("Shape of feature_list = ", feature_list.shape)
```

Output:

```
Found 21 images belonging to 5 classes.
Num images   =  21
Shape of feature_list =  (21, 2048)
```

Now, We will fit the nearest neighbor algorithm to the extracted features.

```
neighbors = NearestNeighbors(n_neighbors=5,
                             algorithm='ball_tree',
                             metric='euclidean')
neighbors.fit(feature_list)
```

Once the image is uploaded extract the features for the image using the ResNet-50 and find the nearest neighbors for it.

A function that plot the images

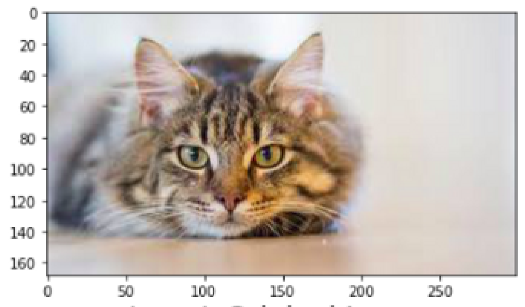
```
def similar_images(indices):
    plt.figure(figsize=(15,10), facecolor='white')
    plotnumber = 1
    for index in indices:
        if plotnumber<=len(indices) :
            ax = plt.subplot(2,4,plotnumber)
            plt.imshow(mpimg.imread(filenames[index]),
interpolation='lanczos')
            plotnumber+=1
    plt.tight_layout()
```

With the nearest neighbors, indices provided by the model let’s find out the images similar to the one we uploaded.

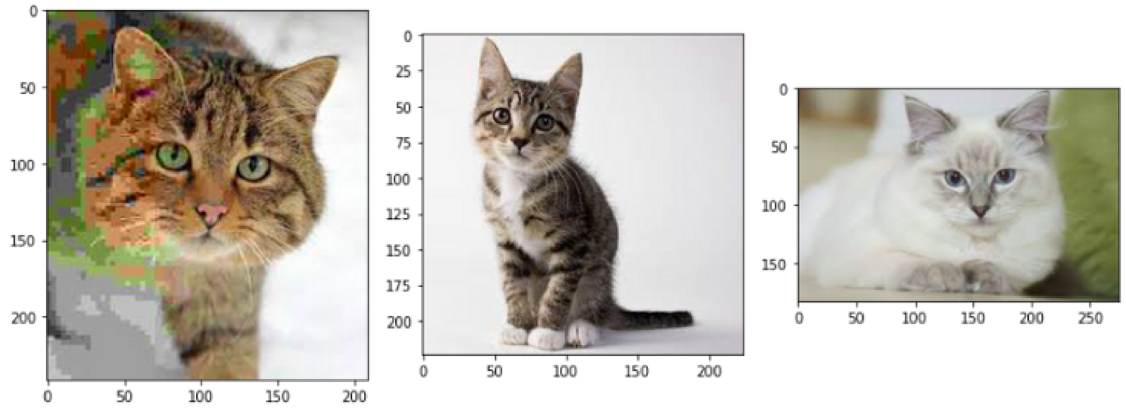
```
print(indices.shape)

plt.imshow(mpimg.imread(img_path), interpolation='lanczos')
plt.xlabel(img_path.split('.')[0] + '_Original Image',fontsize=20)
plt.show()
print('********* Predictions ***********')
similar_images(indices[0])
```

Output:

```
(1, 3)
```



```
trycat_Original Image
********* Predictions ***********
```



References:

https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/