Image Similarity and Clustering via Transfer Learning

Given a database with random images of animals, we perform image retrieval on database images to get the top-N most similar database images using kNN on the image embedding with various similarity metrics.

Approach:

- Generate image embedding using a pre-trained network (trained on ImageNet) such as VGG19, VGG16, ResNet50, Inception etc. by removing its last few layers, and performing inference on our images vectors for the generation of flattened embedding.
- No training is needed throughout this entire processing, only the loading of the pre-trained weights.
- This approach takes single test image/multiple test images and find top-N similar images from the database using K-Nearest Neighbours with various similarity metrics.
- This approach applies KMeans clustering on the dataset and cluster them into N clusters
- This approach has been tested on 400 images from dataset. Works better for large datasets also.

Code workflow:

1. Reads images from specified train and test paths.
2. Load the desired pre-trained model.
3. Apply transformations on train and test images (resizing and normalizing)
4. Convert images into numpy array and creating image embedding using pre-trained model.
5. Fit the KNN model with train dataset information.
6. Perform image retrieval on test image(s).
7. Apply KMeans clustering on the dataset provided and clusters them into desired number.
8. Display of image retrieval and clustered image results.

Pre-trained models:

VGG16: The "**VGG-16 Neural Network**" which is trained for a more than a million type of images with respect to the database of the "ImageNet". The network consists of various types of the layers, and to be specific, there are 16 layers of additional data of media content available.

VGG19: The "**VGG-19 Neural Network**" looks same as VGG-16. The network consists of various types of the layers, and to be specific, there are 19 layers of additional data of media content available.

ResNet50: **ResNet-50** is a convolutional neural network that is 50 layers deep trained on more than a million images from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

Clustering methods:

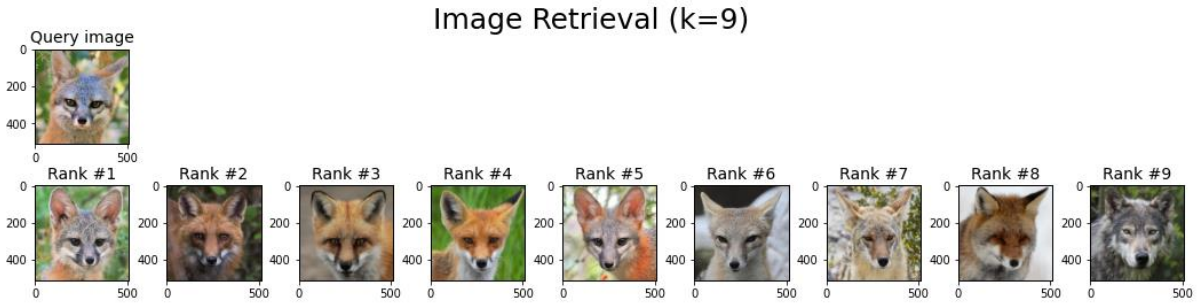
KMeans: The **KMeans** algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the *inertia* or within-cluster sum-of-squares. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

Spectral Clustering: Spectral Clustering is very useful when the structure of the individual clusters is highly non-convex or more generally when a measure of the centre and spread of the cluster is not a suitable description of the complete cluster.

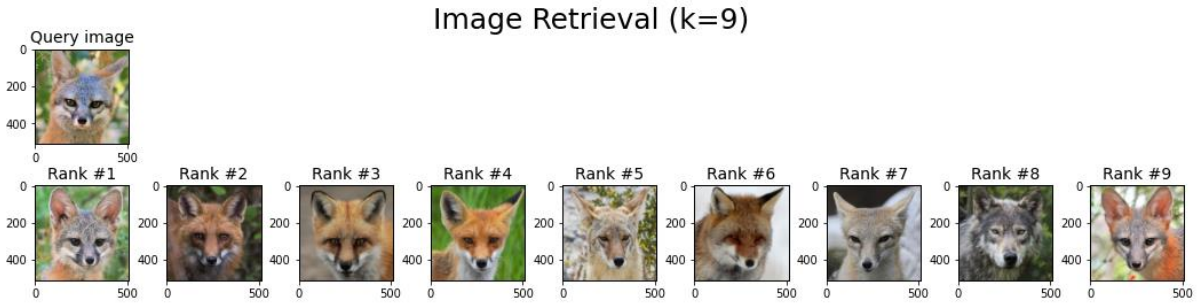
Silhouette scores are calculated for each clustering method. The silhouette value is a measure of how similar an object is to its own cluster compared to other clusters. The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.

Retrieval Results:

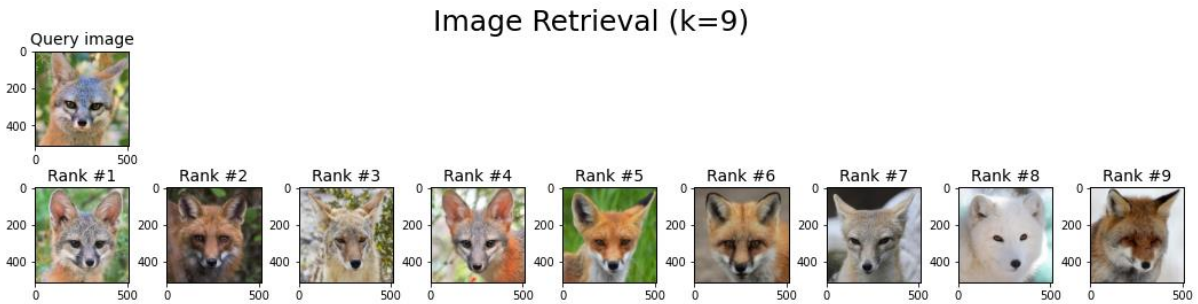
Example 1(a): Pre-trained model: VGG16, KNN Metric: Cosine



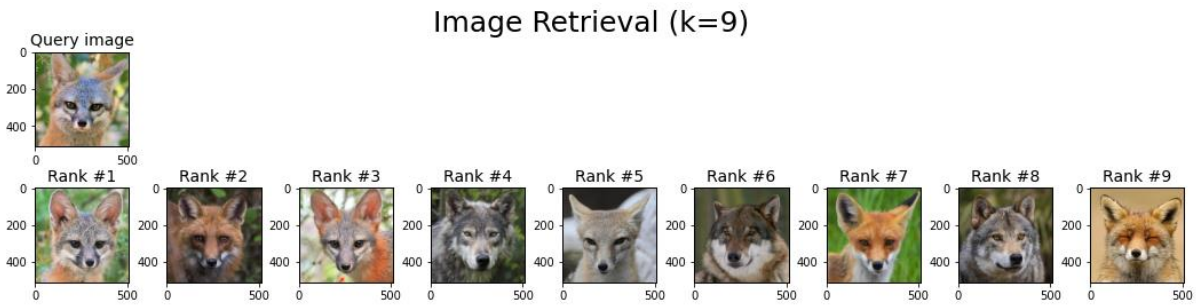
Example 1(b): Pre-trained model: VGG16, KNN Metric: Euclidean



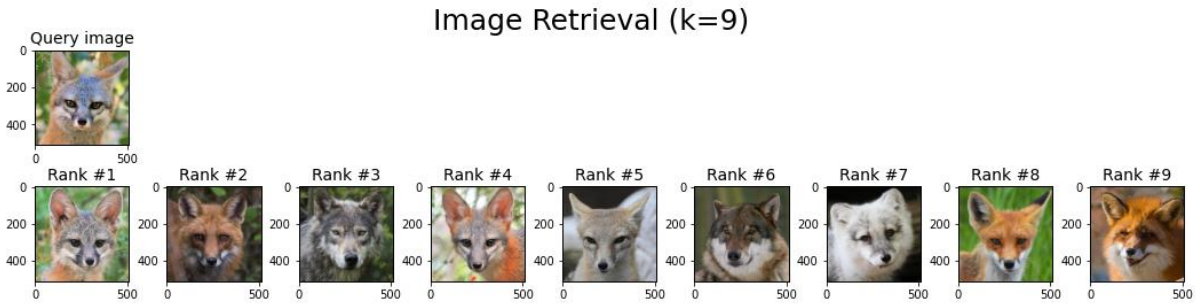
Example 1(c): Pre-trained model: VGG16, KNN Metric: Manhattan



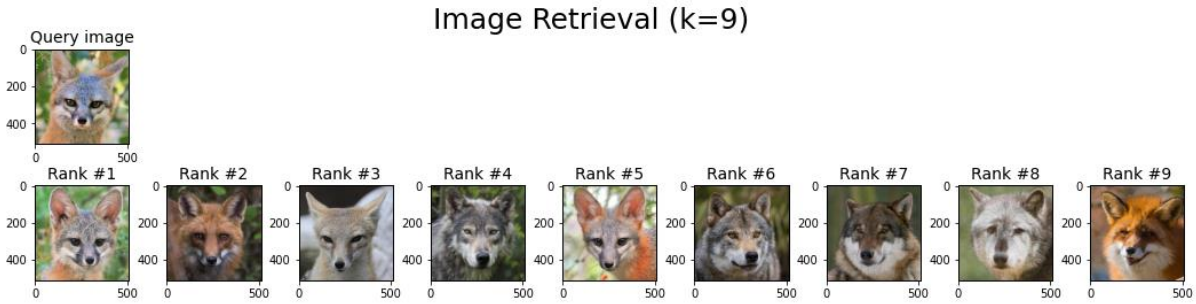
Example 2(a): Pre-trained model: VGG19, KNN Metric: Cosine



Example 2(b): Pre-trained model: VGG19, KNN Metric: Euclidean



Example 2(c): Pre-trained model: VGG19, KNN Metric: Manhattan



Clustering results:

For 31 samples,

KMeans clustering results

Cluster_1 has 18 samples

Cluster_2 has 6 samples

Cluster_3 has 4 samples

Cluster_4 has 3 samples

Dependencies:

- Tensorflow, skimage, sklearn, multiprocessing, numpy, matplotlib