Avantari

Machine Learning Challenge
FINDING SIMILAR IMAGES FROM DATASET

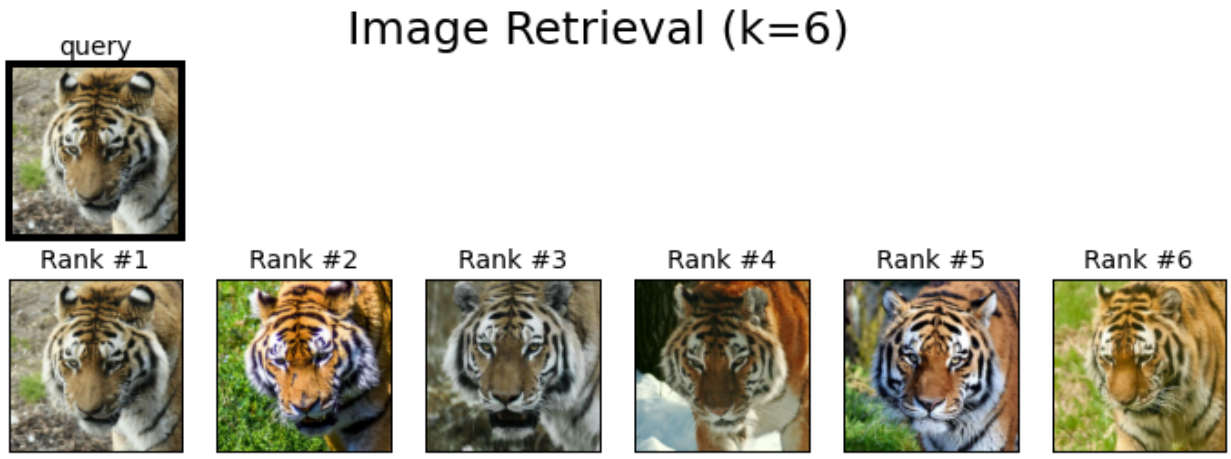


Fig : Actual final result.

Introduction

This task was aimed at finding the similar images from a dataset based on a query image. In this solution we try to find :

- Best similar images based on query image as whole.
- Also aim to find similar images based on the query which is not in the dataset, this will highlight the models performance on extracting features from the query image.
- Also as a subtask we try to find the image clusters and actual number of it.

Main Task :- Finding Similar Images.

Our approach was to use the ML Model to get the features from images instead of hard coded cv techniques.

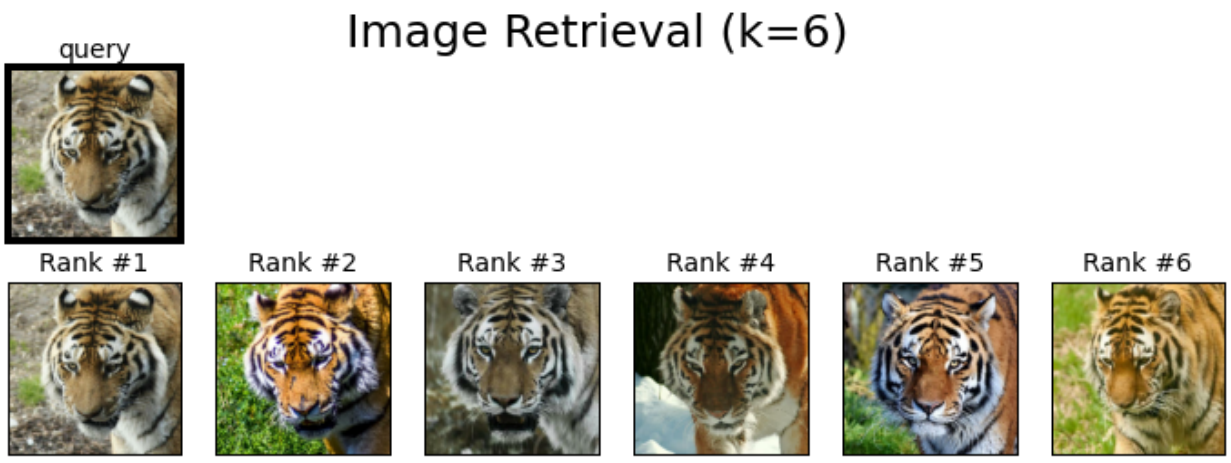
In this approach we used 2 major ways:

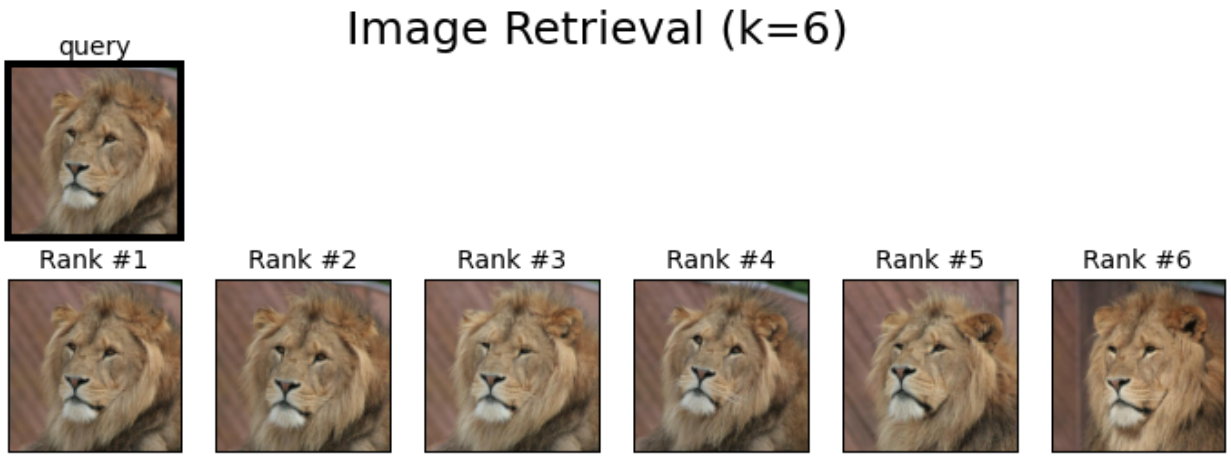
- Transfer Learning : Using the pre-trained model to extract features.
- Model from scratch: Create a simple CNN model to get features.

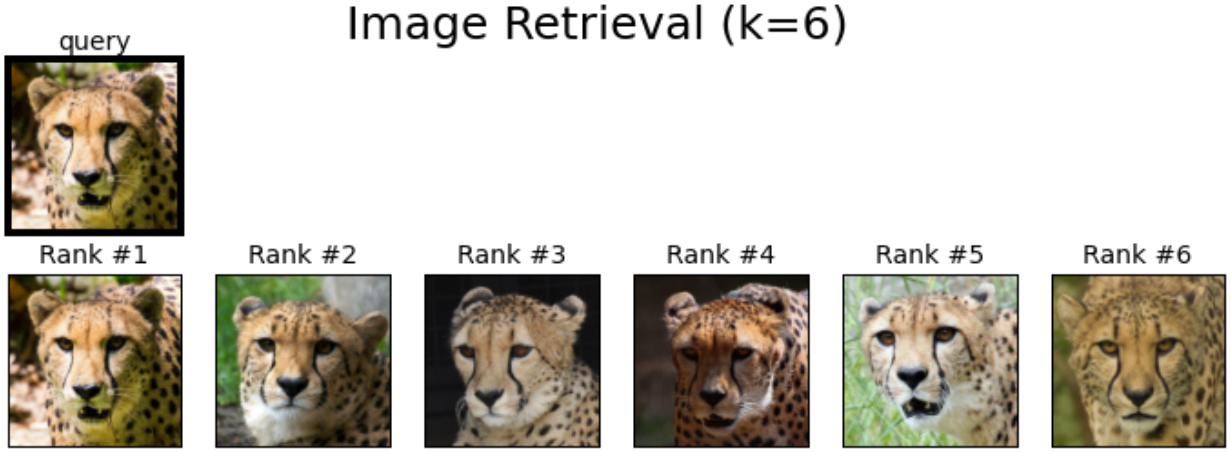
Transfer Learning:-

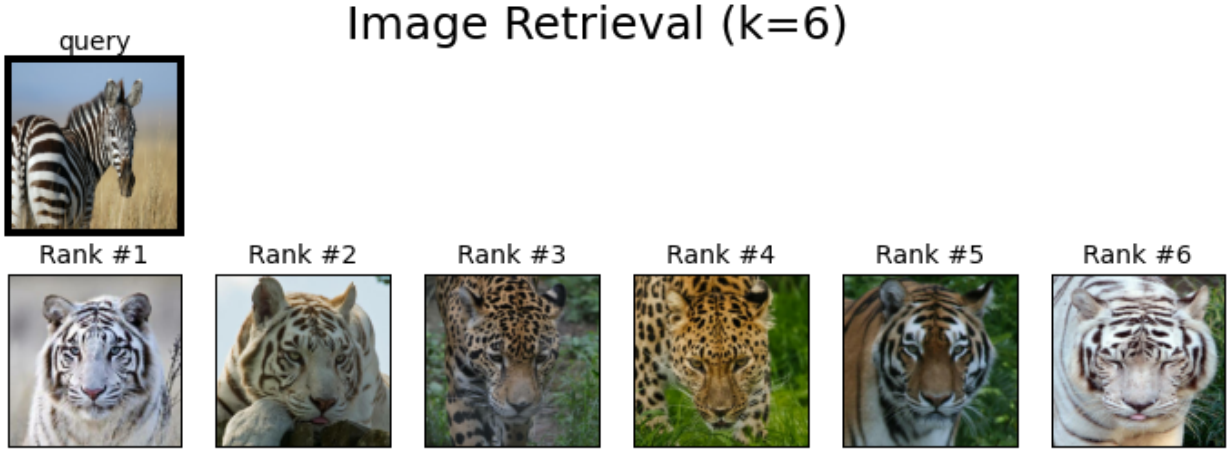
For this we use the “Mobile Net” model with imagenet weights, and extracted the feature for all images in the dataset. (As there is no need to split the dataset). The obtained feature set is then used to compare two images. The cosine similarity is used for comparison.

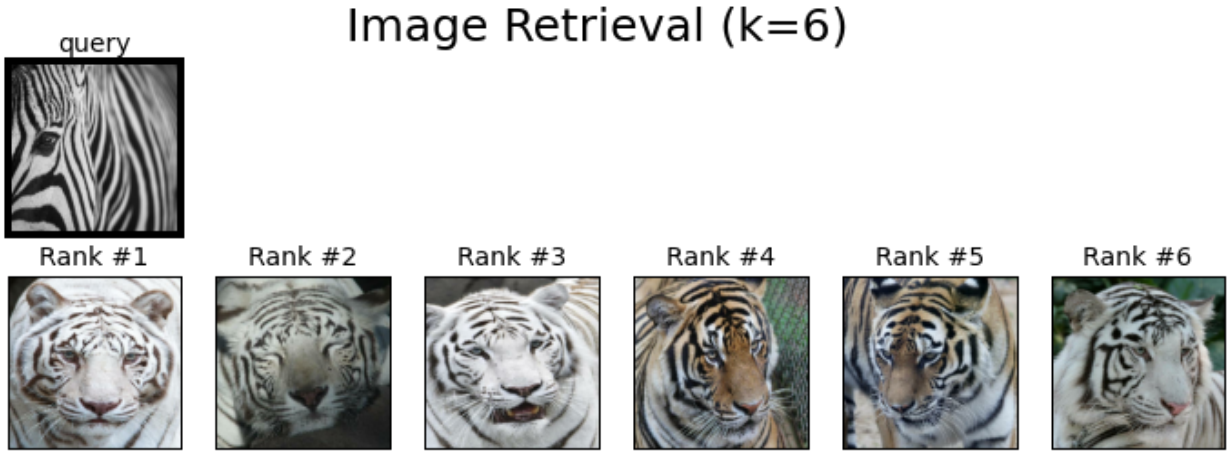
The results obtained from this are shown below.











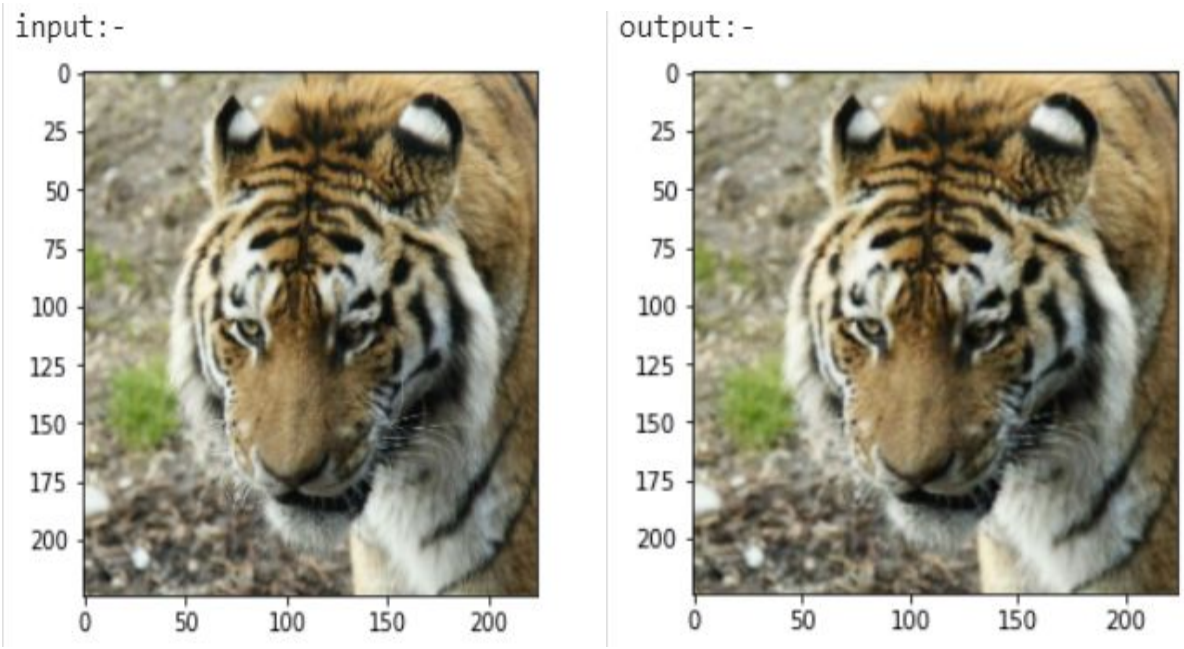
The Zebra images above were used to find the models selectivity towards the pattern in the image or say Unique feature. As the zebra mainly has two features first Black and white and second striped. The results justify both the patterns.

We tried this approach for both 512x512 and 224x224. The later model was best in terms of resource and speed. [Code is here]

Scratch model

To train models to extract the features from the images. To train such models we cannot rely on conventional X,Y pairs as here we don't have any labels or tags to compare to i.e. no Y, Thus we use the techniques/model structure called autoencoders for this task. Also as the data is of Image nature we need to extract features rather than pixel representation we would use CNN based autoencoders.

So the final reconstruction Vs. the Query image is shown below :



So the model works good as encoding and decoding the images.

We have tried 3 versions of the autoencoders and the code for the final version is here.

So then the encoder model is used to extract features of images which then used to infer. The results can be shown below with the same query as above.

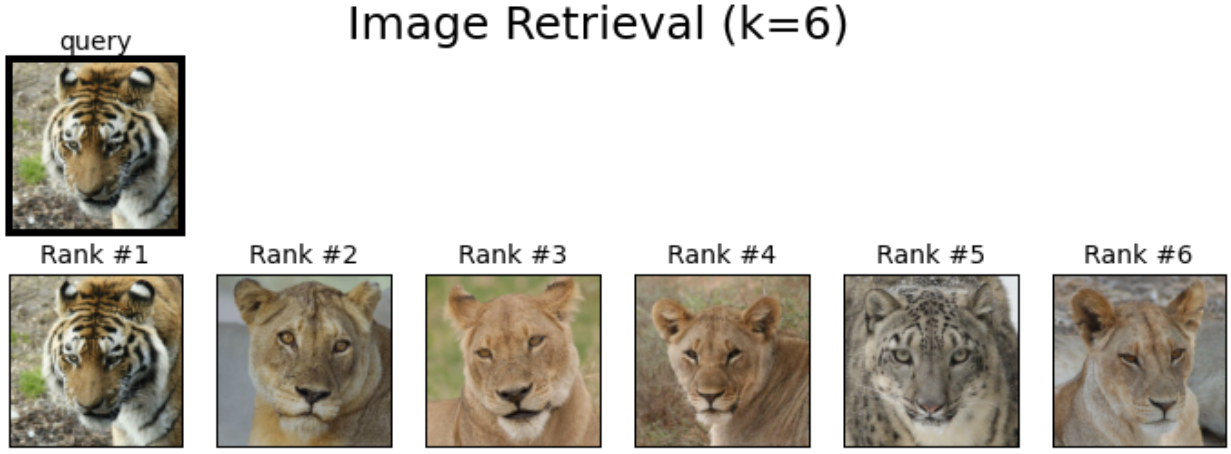


Image Retrieval (k=6)
query
Rank #1
Rank #2
Rank #3
Rank #4
Rank #5
Rank #6


Image Retrieval (k=6)
query
Rank #1
Rank #2
Rank #3
Rank #4
Rank #5
Rank #6

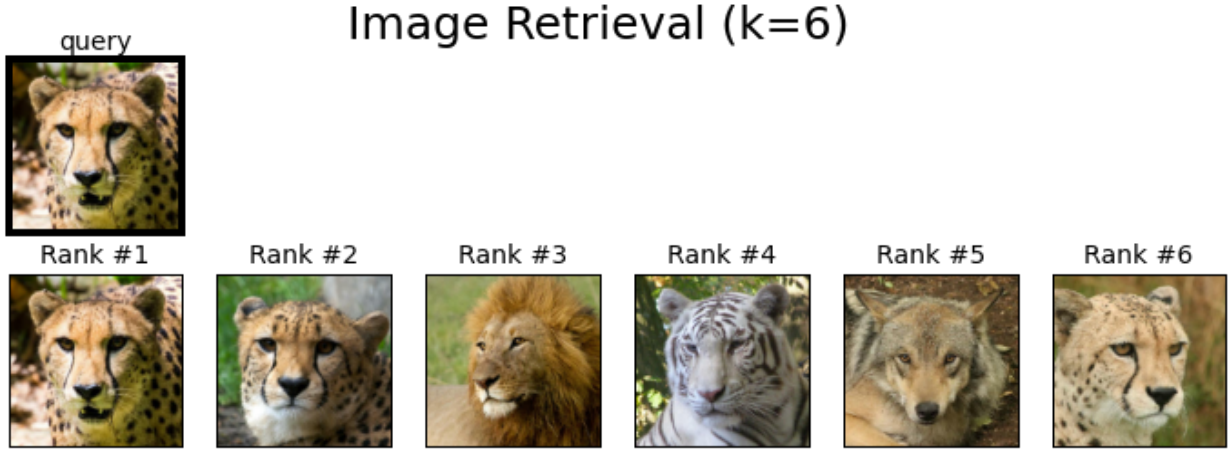
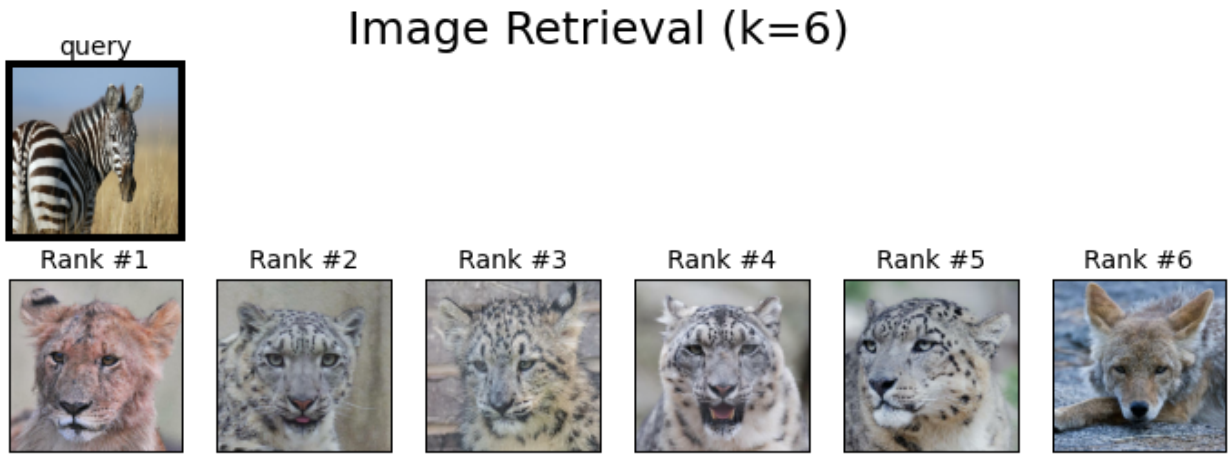
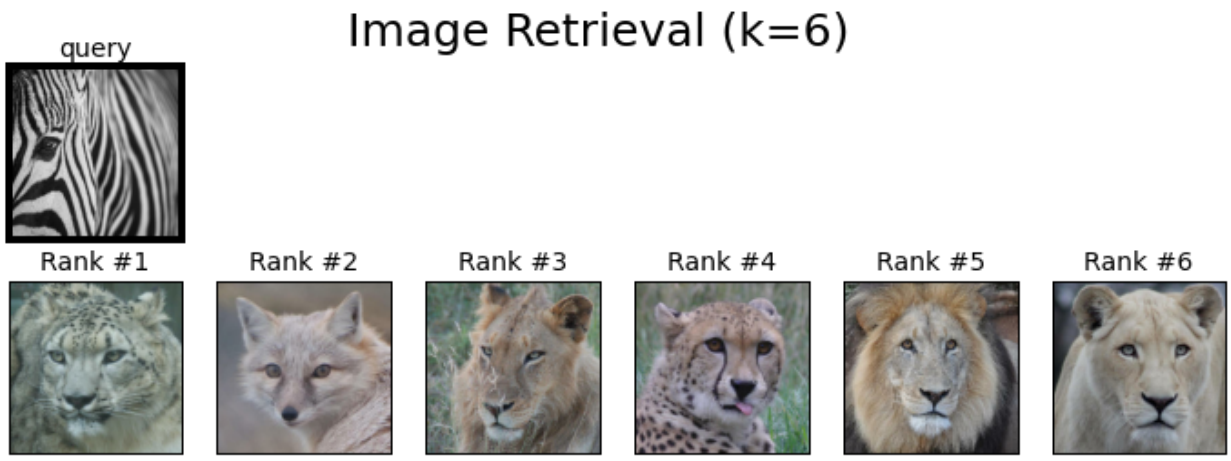


Image Retrieval (k=6)
query
Rank #1
Rank #2
Rank #3
Rank #4
Rank #5
Rank #6





As we can see the result is much sub standard that from the mobile net model as the CNN encoder model was very small we can make it deeper to get the better results. But this would mean longer training time.

Sub Task :- Image clustering.

Now we move towards the image clustering problem. The methodology of the clustering is to make similar images together. So the method we follow is that,

- We use the mobile net model to create embeddings or extract features, rather than directly using the image itself.
- Use a clustering algorithm, here we used the DBSCAN method as the input data is large with a high number of features as well as we don't know the final value of

clusters ahead of the problem. Thus we use DBSCAN with cosine distance to create the clusters. The results were impressive at the first run , as shown below.

```
Compute db scan on...
Estimated number of clusters: 4
Estimated number of noise points: 779
```

Thus it was able to find 4 unique clusters, with 779 images as unidentified to any of 4 clusters. [Code Here]

Code :-

All the coding was done on colab free instances, and all the notebooks are self sufficient to run.
All the notebooks with all the versions can be found on Github repo.