AI Project Report

Hardi Patel, Johann Winter, Aayushi Shah

1. Abstract:

We are interested in locating artifacts that are comparable to a query artifact in various areas. We're going to build a system that can discover artifacts when the queries are visible in this project. Images of faces and videos of individuals will be the kind of queries we will use to test our system. So our approach for part 1 started with importing libraries like tensorflow, matplotlib and sklearn. The key result is to get the efficient amount of related images of a query given.

2. Introduction:

It is important to work on this problem in order to find similar images that we have in our dataset related to a person. This is an important problem since we can use it in our daily lives. One can use this in the crime department in order to detect a person they are looking for and to check whether it is the same person or not. One can also use it to detect at many security checkpoints where they detect the face of the person to whether he or she is the same person or not. By performing the reverse visual search experiment one can detect similar images of a specific person. Our results for part 1 have somewhat similarity but

the results of part 2 are more accurate since we performed elasticsearch on the queries to get efficient results and more similarity.

3. Related Work:

There are various similar approaches in solving this problem. A published work contains the use of eigenvectors. Eigenvectors are unit vectors, which means that their length or magnitude is equal to 1.0. It is a characteristic vector of a linear transformation that changes at most by a scalar factor when that linear transformation is applied to it. Another published work included OpenCV for face detection. OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.

4. Data:

We are working with the “Labeled Faces in the Wild” dataset. It is a database of face photographs designed for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. Many groups are not well represented in LFW. For example, there are very few children, no babies, very few people over the age of 80, and a

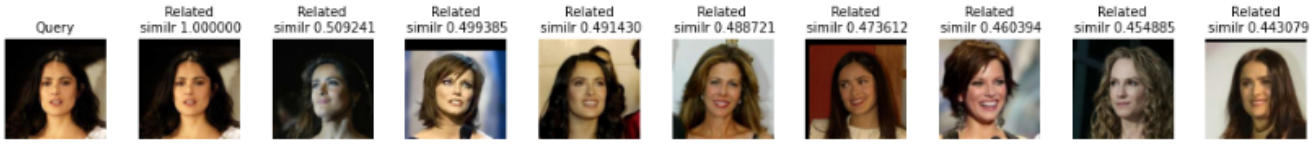
relatively small proportion of women. In addition, many ethnicities have very minor representation or none at all.

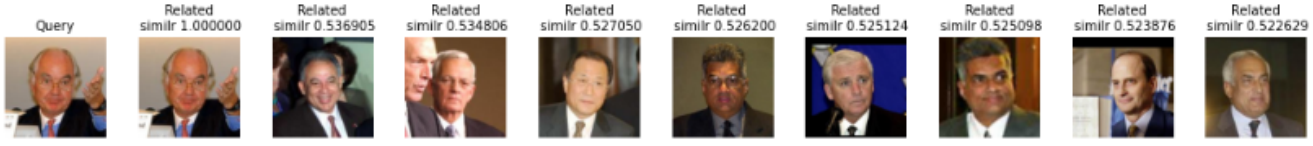
5. Methods:

To solve the problem we used tensorflow, sklearn, pickle, tqdm, and matplotlib. TensorFlow is an open-source library used primarily for deep learning applications. It also supports traditional machine learning. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network. tqdm is a library which is used for creating Progress Meters or Progress Bars. Matplotlib is a library used for creating static, animated, and interactive visualizations in Python.

```
import os
import pickle
import tensorflow
import matplotlib.pyplot as plt
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tqdm import tqdm
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

```
filenames_path = '/content/filenames.pickle'
features_path = '/content/features.pickle'
images_folder_path = '/content/images'
```





6. Experiments:

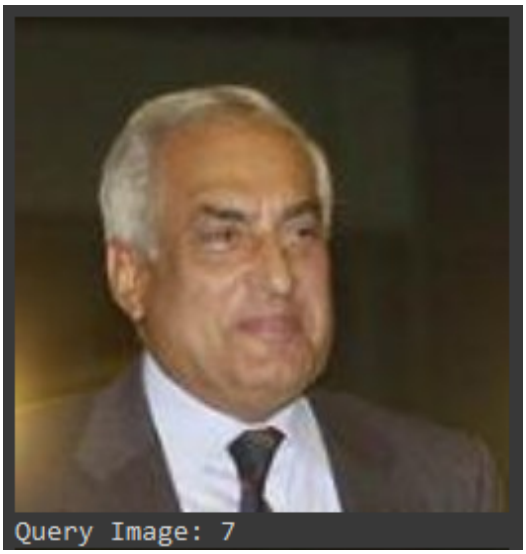
In order for an improvement in the baseline performance we used Elastiknn to get more similar image results based on the query image. A

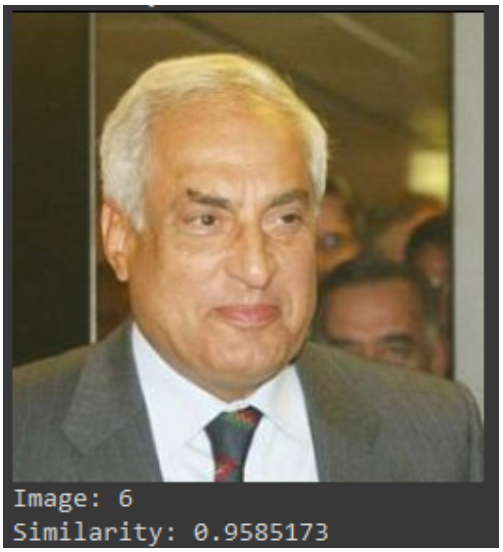
K-nearest neighbors search finds the K nearest vectors to a query vector, as measured by Euclidean distance. kNN is commonly used in the following scenarios:

i. Natural language processing (NLP) methods are used to rank relevance.
ii. Engines for product suggestions and product recommendations
iii. Image or video similarity searches

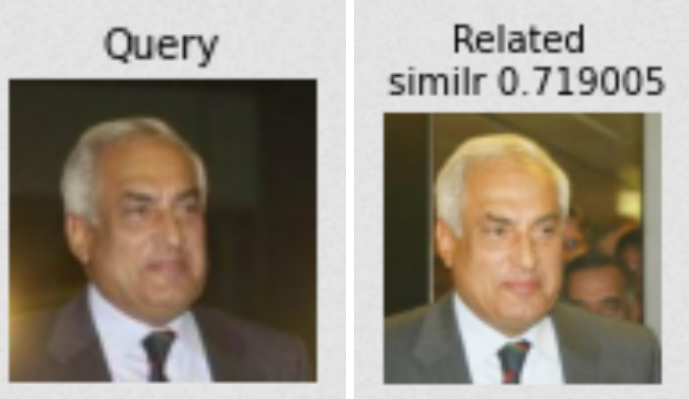
The Elastiknn plugin for Elasticsearch which implements kNN, bridges the gap by providing efficient exact and approximate vector search. This allows users to enhance their search experience by combining standard searches with vector search queries (e.g., a picture (vector)).

```
! pip install --upgrade elastiknn-client
```





From the above picture, we can see that using Elastiknn improves the embeddings of the images and improves the search/detection. The queries also ran in significantly less time compared to part one by several orders of magnitude. The difference in accuracy can be seen most clearly in the query shown above. The image returned using Elastiknn was scored at .95 while the response to the same query from the model in part one was scored at only .71 (shown below).



This is a large increase in accuracy of over 20%.

7. Conclusion:

We have learned how to use a big dataset to break it down in smaller segments and use different plugins to find similar images to the query images. In the future we can use these ideas to help recognize similar people or things if they need to be captured or studied. It can be used at many places which includes crime departments as well.