Reverse Visual Search Report
Group 20

Danrong Li
New York University
dl4111@nyu.edu AWS Contact Person

Yuxiong (Cody) Cheng
New York University
yc4909@nyu.edu

Xuanhua Zhang
New York University
xz3426@nyu.edu

Abstract

Nowadays, it is not uncommon for users to search for pictures through typing keywords inside search engines, like Google. However, in the situations where words cannot quite capture the picture properties, an alternative searching method seems crucial. With this as our motivation, our team strives to find a solution in searching pictures with pictures. With pre-trained CNN model VGG19 as our baseline in this project, we also tried to utilize the facenet model and elastic search KNN approach to improve the baseline searching accuracy. As a result, the aforementioned techniques successfully beat the baseline performance and this improvement can be observed inside our github code results by human evaluators.

1. Introduction

Our team tried to produce an image searching machine that can output several similar pictures based on the user-input pictures. To be more precise, the goal was to output 20 similar human faces with the user-input human face. The reason why such search machines are important is that it induces a convenient and efficient searching process. In the face recognition area, typing features like "middle-aged blue-eyed caucassian male with a van dyke beard" is more tedious and possibly erroneous than simply inputting the potential person of interest and waiting for the computer algorithms to output several matches. In addition, the image reverse search model can be not only applied to faces, but also other categories of pictures. For instance, the emerging e-commerce industries are researching ways to recommend clothes to customers based on the input clothing pictures. As for our project results, we were able to beat the baseline accuracy by a lot with the facenet models and elastic search knn approach. To be more specific, the improved method could output all the faces for the user-input person in the dataset. Since in this project, the 10 faces that professor Pantellis chose only has 6 faces inside the dataset, the improved method could output these 6 images accurately along with 14 other similar faces.

2. Related Work

There are several publications related to our work. In [3] Gaillard, M., Egyed-Zsigmond, E. (2017), the authors focused on large scale reverse image search, with motivation in intellectual property protections. With [1] Araujo et.al. (2018), the authors were trying to retrieve similar scientific images. Although these related works are not aligned with our model that is specifically designed to face images, the general approaches share similar steps. In[1] Araujo et.al. (2018), the authors used a CNN model with max pooling, convolutions and fully connected layers called LeNet. In our baseline, we used VGG, which has a similar framework with LeNet, but with more parameters inside. In[3] Gaillard, M., Egyed-Zsigmond, E. (2017), the authors used hashing to index the images inside the database, while in our improvement step, we utilized elastic search to index the stored features. As for the image matching process, both Araujo et.al. (2018) and our project used KNN model to get the similar results, the k-nearest neighbors with pre-set k values, however, Gaillard, M., Egyed-Zsigmond, E. (2017) tried to output results based on pre-set threshold values. In a nutshell, our work follows the general principles when it comes to machine learning implementation with images, and the choices in specific models and parameters are tuned to fit our project goals better.

3. Data

The data utilized in this project is called: Labeled Faces in the Wild[4].This dataset is a database of face photographs designed for studying the problem of unconstrained face recognition. It 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 dataset. For our

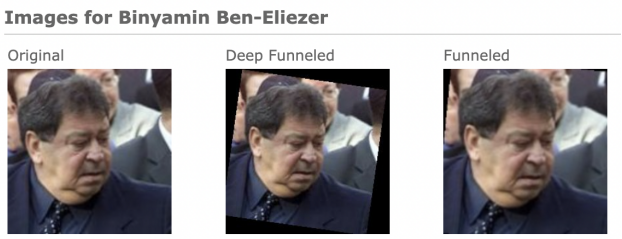

Figure 1. [4]Three versions of the dataset

project, we used deep funneled version images. As figure 1 shown below, deep funneled images are not represented as axis-aligned rectangles. The reason why we chose this version of the dataset is that the rotated images would better resemble the real-life input images from users.

Other than the property of being deep-funneled, our group did not introduce any other extra artificial alterations to the dataset, including translation, gaussian blur or augmentation. Due to the size of the dataset, our group downloaded the zip file and uncompressed it inside the google colab environment. We used the entire dataset in the baseline step. During the improvement step with facenet model feature embeddings, since some faces were not successfully detected by the facenet model, we excluded those faces. These exclusions are trivia since the amount of images with features successfully extracted is still above 13,000.

4. Methods

For the baseline, we used pre-trained CNN model VGG19. The architecture of the VGG family is shown below in figure 2. VGG 19 architecture is in column E. The only pre-processing inside the model was taking the mean of the RGB values of each pixel, and computed over the entire training dataset. The use of kernel with size 3*3 and stride 1 ensured the coverage of the entire image. The spatial padding is used to preserve the spatial resolution of the image. Each convolution layer is followed by relu operation to ensure the nonlinearity of the model. Another reason for choosing relu over sigmoid or tanh is due to the better computation efficiency. To be more precise, sigmoid and tanh require computed real number results, and relu is a choice of maximum value with 0. Figure 3 illustrates the differences between sigmoid and relu functions.

For every image we feed into the VGG19 model, we receive feature vectors. Then we utilized the sklearn package NearNeighbor to get the 20 nearest neighbors for each query image. For each query, the model would output the 20 index of the images as well as its distance to the query image. Not surprisingly, the first index returned is always the query image with distance 0. It makes sense since every picture would have the distance of 0 to itself. Figure 4 shows a way to visualize how KNN model helps every query image

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2. [6]VGG family model architecture

finding its neighbors. To be more specific, given the query image, the model would find its distance with other images inside the dataset. This distance can be represented in figure 4. The smaller the distance, the closer is the neighbor. And the model would output the top 20 images that are the nearest. Note the parameter 20 is chosen due to professor Pantellis instructions. Since the goal of the project was to output similar faces instead of face verification, the choice of using unsupervised KNN is a valid approach. This is because with supervised KNN, the aim would be to match the input face with a target, and then test if the match is aligned with the given label.

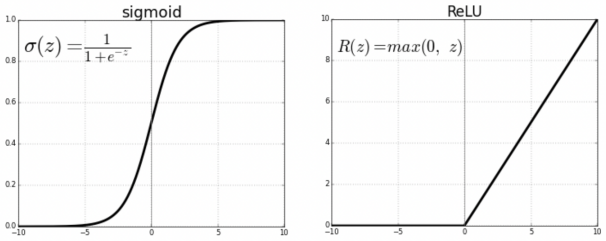

Figure 3. [5]sigmoid and relu functions

During the improvement step, our group used the facenet and elastic knn approach. Facenet is a Siamese network. Figure 5 shows an example of the Siamese neural network. It is a type of neural network architecture that learns how to differentiate between two inputs. In this way, the facenet model can learn which images are similar and which are

Figure 4. neighbors-with-distance visualization

not. The Siamese network consists of 2 identical neural networks, each with the same exact weights. First, each network takes one of the two input images as input. Then, the outputs of the last layers of each network are sent to a function that determines whether the images contain the same identity.

In facenet, this is done by calculating the distance between the two outputs. Siamese networks are commonly used in one-shot learning face recognition tasks. It is suitable for our project to utilize facenet with Siamese network because of the nature of the dataset. Since the dataset is very imbalanced, with a lot of people with just around 1 or 2 pictures, our project qualified as a one-shot learning task. And it makes sense for us to use facenet.

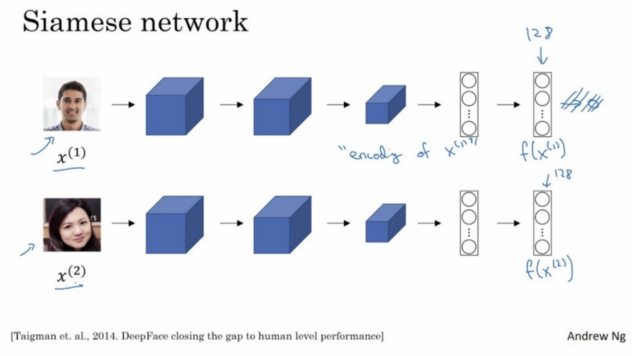

Figure 5. [7]Siamese network example from Deepface: Closing the gap to human-level performance in face verification

Dimensionality reduction is the approach that Siamese networks use to address one-shot learning problems. Unlike other loss functions that may evaluate the performance of a model across all input examples in the training dataset, contrastive loss is calculated between pairs of inputs, such as between the two inputs provided to a Siamese network. Pairs of examples are provided to the network, and the loss function penalizes the model differently based on whether the classes of the samples are the same or different. Specifically, if the classes are the same, the loss function encourages the models to output feature vectors that are more similar, whereas if the classes differ, the loss function encourages the models to output feature vectors that are less similar. The loss function requires that a margin is selected that is used to determine the limit to which examples from different pairs are penalized. Choosing this margin requires careful consideration and is one downside of using the loss function.[2]

5. Experiments

For the baseline step, with unsupervised KNN, there are multiple ways for neighbor-choosing inside the sklearn package, like KD tree or brute-force, and our group chose "auto", which asks the model to pick the most efficient algorithm depending on the seen training data. As for the professor-chosen 10 queries, we showcased them all inside the github code by using the first image of each person as the input image, and then output the 20 similar images. Figure 6 shows the partial baseline results with the first query. For the improvement step, after feeding the dataset to the

Figure 6. partial baseline query results

facenet model, we receive the feature vectors (dimension 128), and then output the vectors to a single data frame containing both the path to the image and the image feature vectors. The reason why the path is important is because during the feature vector generation process, the facenet built-in functions were called several times and the input parameter is the path to image. In addition, the path contains the person name as well as the picture index, which both could be useful in the later stage as in precision checking . Figure 7 shows the generated data frame. Before large-scale using the feature vectors to select the 20 similar faces, we tried to write from scratch the distance-comparing functions to see if the approach is valid. Figure 8 shows that 2 faces from different people have a Euclidean distance of 18.54. Figure 9 shows that 2 faces from the same people have a Euclidean distance of 7.57. And this suggests that our facenet feature vectors are valid.

	Path	Representations
0	drive/My Drive/nyu second semester/artificial ...	[-0.61650825, -0.36295557, 0.29242897, -0.4140...
1	drive/My Drive/nyu second semester/artificial ...	[0.23787875, 0.42983374, -1.3669863, 1.167738,...
2	drive/My Drive/nyu second semester/artificial ...	[-1.9144613, 0.4721296, -2.3119712, -0.8226508...
3	drive/My Drive/nyu second semester/artificial ...	[-1.0064372, -0.3309246, -2.275216, 1.0204922,...
4	drive/My Drive/nyu second semester/artificial ...	[-0.6732321, -0.06661366, 0.96323204, -0.54453...

Figure 7. partial baseline query results

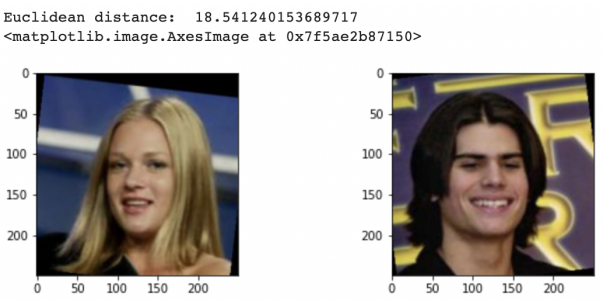

Figure 8. different people with distance 18.54

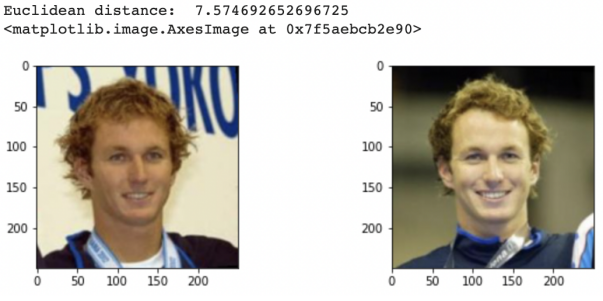

Figure 9. same people with distance 7.57

Now we're creating the Elasticsearch index (figure 10) and importing all the generated image facenet vectors and image path name into the index. The 'l2_norm' in similarity tab computes similarity based on the L2 distance (also known as Euclidean distance) between the vectors. The document '_score' is computed as

$$1/(1 + l2\_norm(query, vector)^2) \quad (1)$$

As for the professor-chosen 10 queries in improvement part, we showcased them all inside the github code(at the end of Elasic_KNN.ipynb) by using the first image of each person as the input image, and then output the most similar 20 images / the 20 images with highest '_score' value based on the computation of L2 normalization inside Elasticsearch.By using the provided k-nearest neighbors method Approximate KNN inside Elasticsearch, we're be able to return the most similar 20 image of given object. Figure 11 and figure 12 some examples of the asked professor-chosen

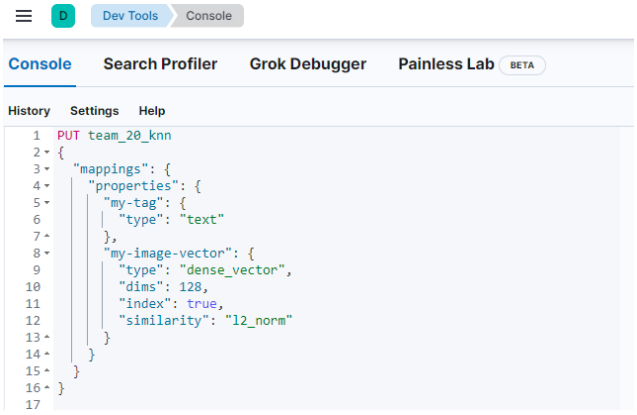

Figure 10. create elasticsearch index in kibana console

10 queries. The first image which is located at the top-left corner is the given input image and it's also the most similar image in the similarity search result by KNN search. Continued per row from left to the right, the similarity score decreased.

Compare the improvement result figure 11 with the baseline result figure 13 of the same person Armino Fraga. Improvement results are much more accurate as 5 of 6 total images of the given person are listed as top 5 in similarity compared to almost random returns in the baseline model's result. All the other returned results of improvement models are highly accurate as the majority of the given people's image will be listed in the top 10 of the 20 image query. See example figure 11 and figure 12.

6. Conclusion

From this project, we learned that there are many available models for face recognitions, or other image recognition tasks. And it is important to choose the one that is suitable for the task. By using the facenet model, we were able to improve the accuracy by a lot. As a future extension to our work, fine-tuning the VGG19 model before feeding into the KNN model would be a good idea. Now we are at the end of a successful expedition where we explore locating similar faces in images with the help of facenet embeddings and approxicate knn search.

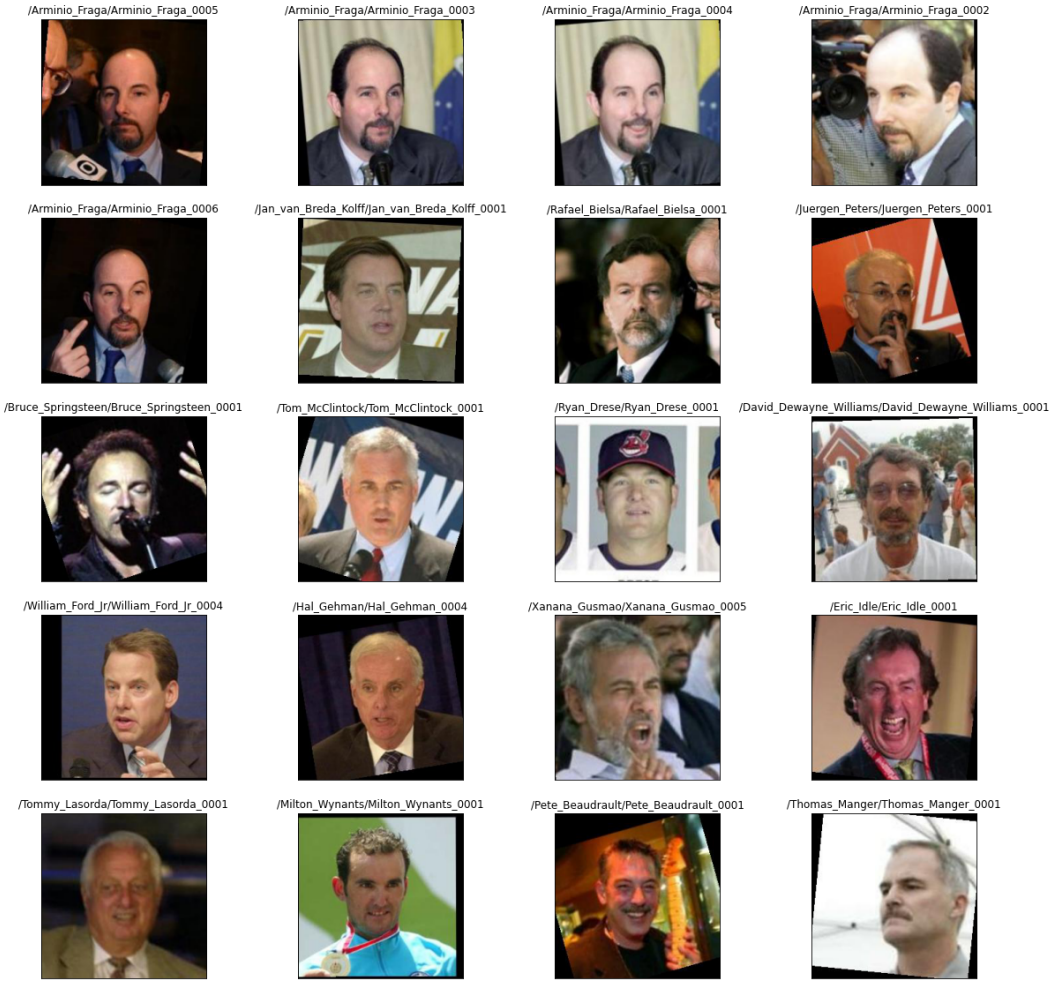

Figure 11. Improvement result of 20 similar face given person Armino Fraga

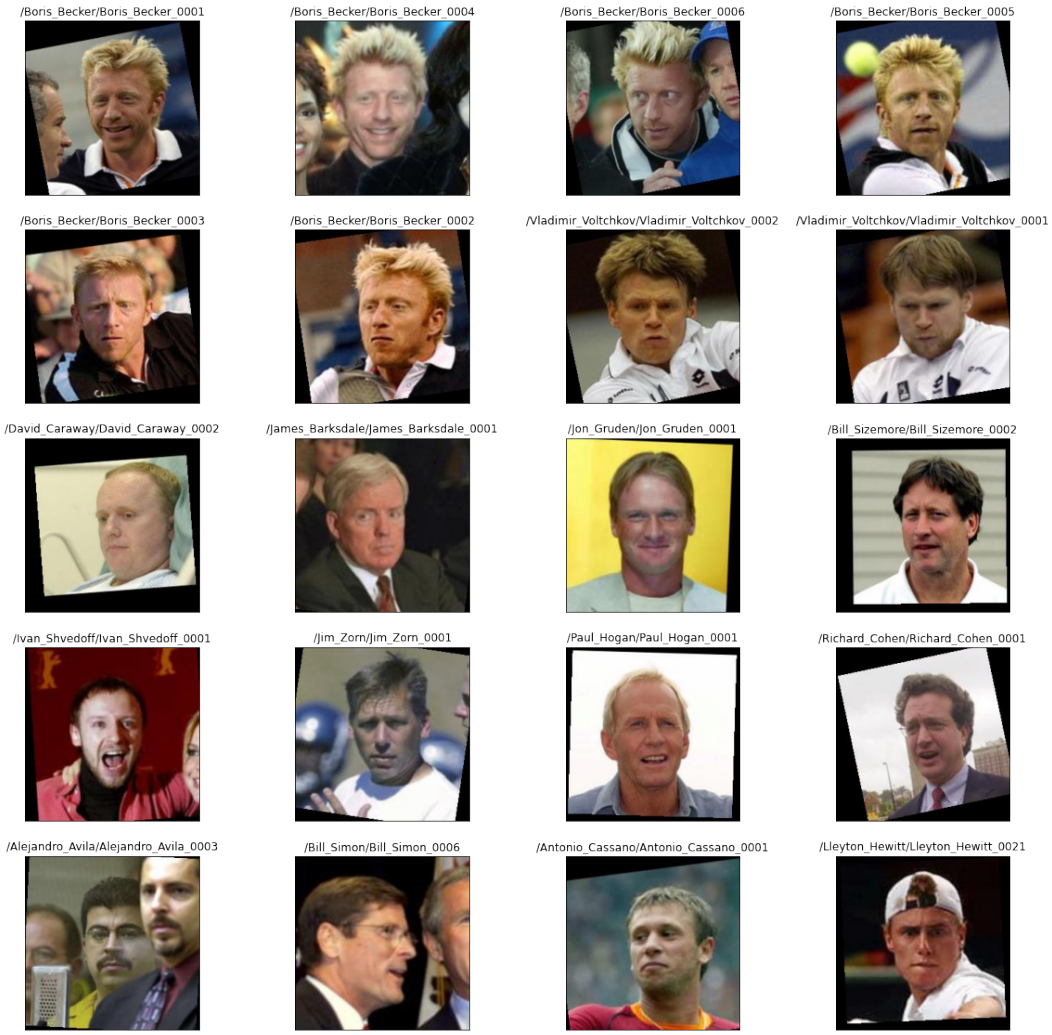

Figure 12. Improvement result of 20 similar face given person Boris Becker

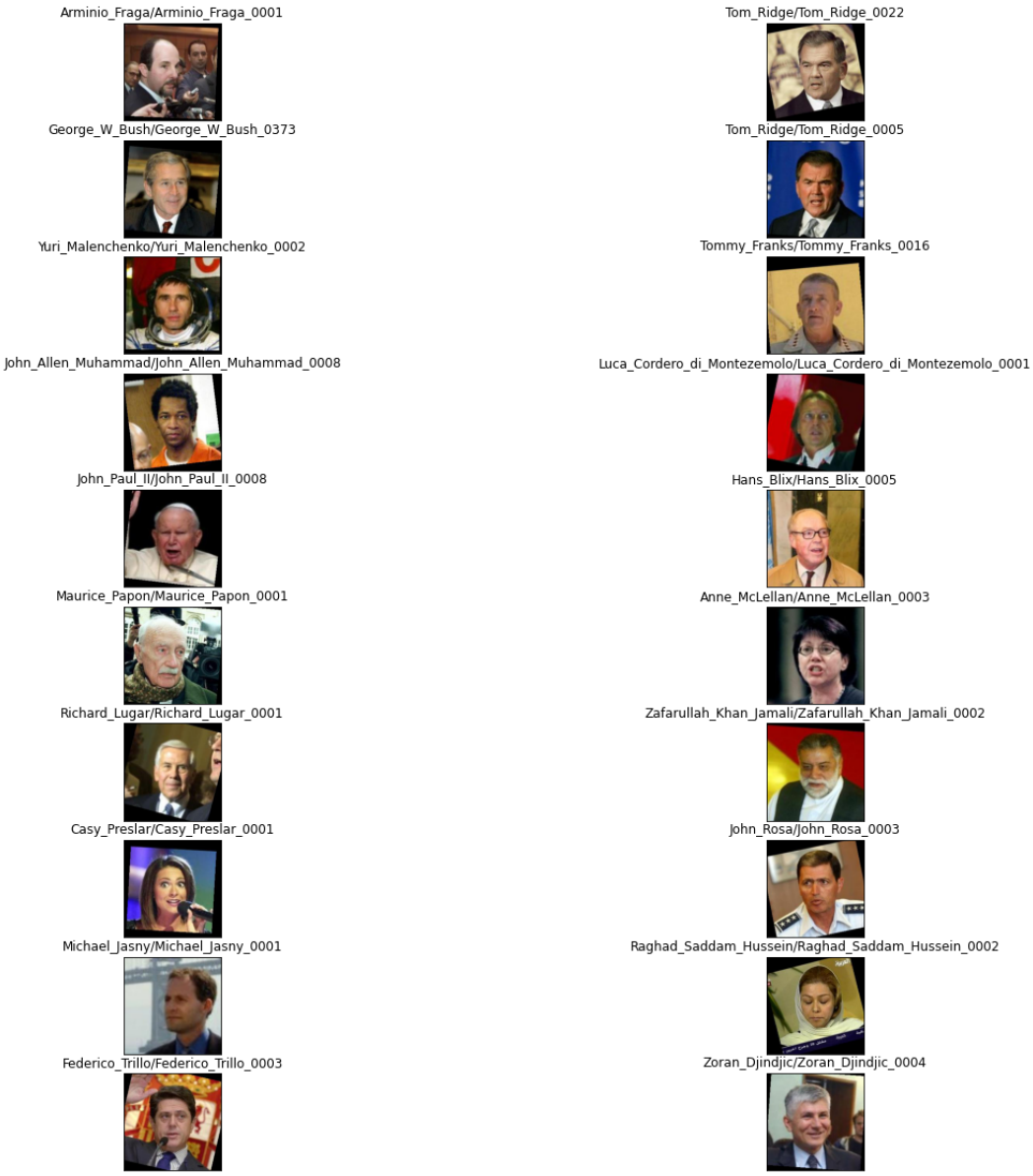

Figure 13. Baseline result of 20 similar face given person Armino Fraga

References

[1] F. H. Araujo, R. R. Silva, F. N. Medeiros, D. D. Parkinson, A. Hexemer, C. M. Carneiro, and D. M. Ushizima. Reverse image search for scientific data within and beyond the visible spectrum. *Expert Systems with Applications*, 109:35–48, 2018.

[2] J. Brownlee. One-shot learning for face recognition, June 2019.

[3] M. Gaillard and E. Egyed-Zsigmond. Large scale reverse image search. *XXXVème Congrès INFORSID*, page 127, 2017.

[4] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[5] S. Sharma. Activation functions in neural networks, September 2017. Sigmoid, tanh, Softmax, ReLU, Leaky ReLU EXPLAINED !!!

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.