Unsupervised face recognition using a video clip

Mhd Hussein Murtada

Abstract

In this paper, I am proposing a faces classifier that takes videoclips as input and applies unsupervised learning to cluster the extracted faces from that video. These clusters will be used to classify unknown images effectively. This classifier is able to classify faces regardless of face angle, facial expressions, facial wears (sunglasses, facemasks, etc..), and different levels of makeup. Moreover, it builds a labeled dataset that can be used to train other supervised learning classifiers.

1. Introduction

With all the advancements we are seeing in the Artificial Intelligence field, data is still the main ingredient for the AI classification system. Having few or not enough labeled data may lead to a very bad and inconsistent classifier. In face recognition for instance, it is very hard to find labeled faces data for specific people we want to classify. Moreover, different facial expressions and face wears make it harder on the classifier. However, by using normal videos that are available online (e.g. on YouTube), we can build a classifier that works perfectly. The video is divided into frames, then faces are extracted out using CNN and Caffe face detection models, their embeddings are generated and clustered into unique faces using DBSCAN clustering algorithm. Unknown faces are then recognized by finding the Euclidean distance from the center of these clusters giving consistent classes whatever the facial expression is, with makeup or without.

2. Related work

This project is based on several related works and algorithms:

Video framing: using FFmpeg Libavcodec library which provides a generic encoding/decoding framework and contains multiple decoders and encoders for audio, video and subtitle streams, and several bitstream filters. The shared architecture provides various services ranging from bit stream I/O to DSP optimizations [1]. It was used to convert the input video to frames.

Face extraction: an OpenCV Detection Neural Network was used to detect faces in the frames. The model is a pretrained Caffe model. Caffe: "Convolutional Architecture for Fast Feature Embedding" is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research [2].

Another model which was used to find the exact locations of the faces is a pretrained CNN model. Provided by the face_recognition python library API. The library was built using dlib's state-of-the-art face recognition. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark [3].

Face embeddings: using face_recognition's Deep Convolutional Neural Network which was provided by OpenFace [8], a vector that contains 128 different measures for each face. The used algorithm is FaceNet which was invented in 2015 by researchers at Google [4]. Any ten different pictures of the same person should give roughly the same measurements.

Face clustering: using DBSCAN - Density-Based Spatial Clustering algorithm with Euclidean distance as a metric. This algorithm was used because it is good for data which contains clusters of similar density, which are faces of the same person in our case. It views clusters as areas of high density separated by areas of low density. Clusters found by DBSCAN can be any shape, as opposed to k-means which assumes that clusters are convex shaped.[5][6]

Unknown face classification: face embeddings for the unknown face are generated. Then the Euclidean distance is calculated for each cluster. The distance tells where the face belongs.

3. Method

3.1. Dataset:

A one-minute part of an interview with the members of Blackpink KPop girl group was used as an input.

3.2. Summary:

The video was divided into 60 frames, with framerate of 1 fps using FFmpeg libraries. Faces are then extracted out from the frames using OpenCV's Caffe DNN model and written to disk as faces square images.
Faces are then passed to the CNN model that detects the exact coordinates of the face. Faces and their coordinates are given as input to the FaceNet model that outputs a vector of length 128 for each face. Vectors are then passed to the DBScan fitter using "Euclidian distance" as a metric with eps=0.32 and 10 minimum samples (minPts) in each cluster. A dataset of labeled faces is built and stored on drive afterwards.

3.3. Classification**:**

An image with one face at least is passed. The system will detect the face and its coordinates and generate its vector. Next, its vector's distance to each cluster is calculated and the cluster with the minimum distance is chosen. However, if the distance is larger than a specific threshold (which was set to 0.6 in our implementation), the face is considered unknown.

3.4. Behind the scenes:

Detecting Faces with different directions: to solve this, an algorithm called face landmark estimation was used [7]. The basic idea is coming up with 68 specific points (called landmarks) that exist on every face. Then a machine learning algorithm is trained to be able to find these 68 specific points on any face.

Face embeddings: it turns out that the measurements that seem obvious to us humans (like eye color) do not really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure. The solution is to train a Deep Convolutional Neural Network to generate 128 measurements for each face. The training process works by looking at 3 face images at a time:

- Load a training face image of a known person
- Load another picture of the same known person
- Load a picture of a totally different person

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart. This approach was created by Google researchers [4] and implemented by the developers of face_recognition Python package.

Generating encodings: a pretrained CNN by OpenFace [8] was used to generate the embeddings which are the 128 measurements for each face.

Faces clustering: DBSCAN requires two parameters: ε (eps) and the minimum number of points required to form a dense region (minPts). It starts with an arbitrary starting point that has not been visited. This point's ε-neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized ε-environment of a different point and hence be made part of a cluster. If a point is found to be a dense part of a cluster, its ε-neighborhood is also part of that cluster. Hence, all points that are found within the ε-neighborhood are added, as is their own ε-neighborhood when they are also dense. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise [5][6].

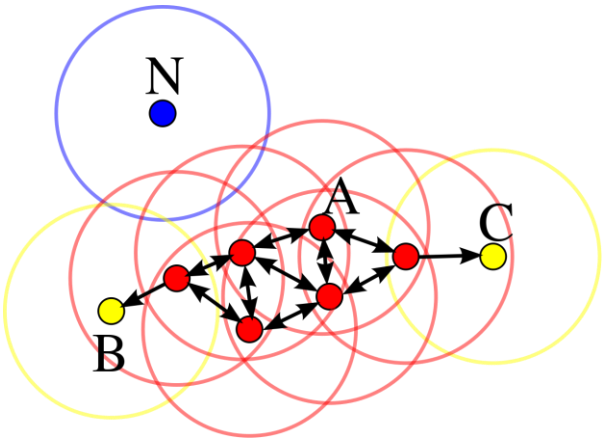
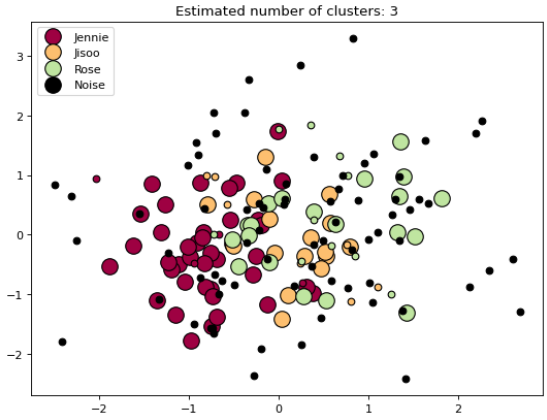


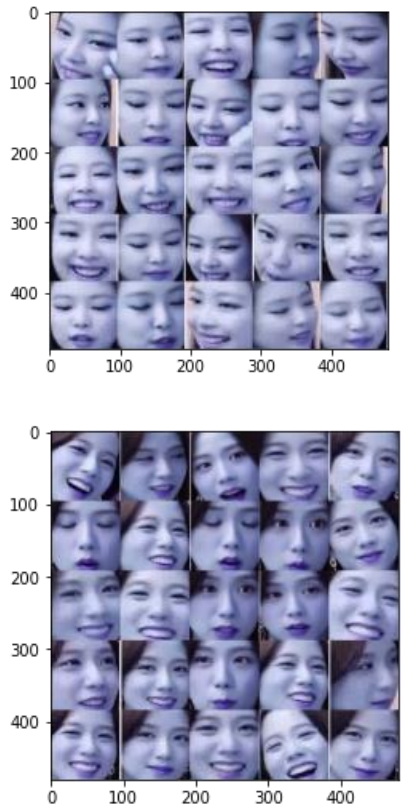
Figure and description from Wikipedia:
In this diagram, minPts = 4. Point A and the other red points are core points, because the area surrounding these points in an ε radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable.

4. Results:

The Blackpink interview video was framed and faces were extracted and then clustered. The clusters were as follows:



Each cluster has its own color which represents faces for the same person. Black dots are faces that were not classified into any cluster. For colored dots, big ones were chosen as core samples and small dots are just samples of the corresponding cluster (not core samples). Below are two samples of the clusters:



Successful classifications: the system was able to classify:

- Images with multiple faces.
- Images with different facial emotions and expressions (crying, laughing, angry, etc..)
- Unknown faces as unknown.
- Faces wearing face masks were correctly classified.
- Faces wearing sunglasses were correctly classified.
- With makeup / no makeup faces were correctly classified.

All results can be found in the attached Jupyter Notebook and in the results folder.

References

[1] "FFmpeg Libavcodec," *FFmpeg*. [Online]. Available: https://trac.ffmpeg.org/wiki/Using libav*.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe," Proceedings of the ACM International Conference on Multimedia - MM 14, 2014.

[3] Ageitgey,"ageitgey/face_recognition,"GitHub,20-Feb-2020. [Online].Available: https://github.com/ageitgey/face_recognition.

[4] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[5] Ester, M., H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 226-231. 1996

[6] Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. ACM Transactions on Database Systems (TODS), 42(3), 19.

[7] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[8] OprnFace.[Online].Available: https://cmusatyalab.github.io/openface/.