Webcam face recognition system

Assignment #3

Image Based Biometrics 2019/20, Faculty of Computer and Information Science, University of Ljubljana

Jure Zajc

***Abstract*—A facial recognition system is a technology capable of identifying and verying person from a digital image or a video fram from a video source. There are multiple avaliable methods in which facial recognition systems work, but in general they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artifical Intelligence based application that can uniquely identify a person by analysing patterns based on the person's facial textures and shape.**

I. Introduction

A real time face recognition system is capable of identyfing and verifying a person from a video frame. To recognize the face in a frame, first we need to detect whether the face is present in the frame. If it is present we marked it as a region of interest, extract it and process it for facial recognition. Comparison is based on a feature similarity metric and the label of the most similar database entry is used to label the input image. If the similarity value is below a certain threshold the input image is unlabeled. Comparing two face images to determine if they show the same person is known as face verification.

It turns out that the characteristics that seem obvious to us humans (for example, eye color) do not make sense for a computer analyzing individual pixels in an image. The researchers found that the most appropriate approach is to enable the computer to determine the characteristics that need to be collected. Deep learning, in turn, allows much better and faster identification.

In our final project we used deep convolutional neural network (CNN) [1] to extract features from input images. Our model follows the approach describeb in [2] with modification inspired by the OpenFace [3]. Keras [4] is used for implementing the CNN. OpenCV [5] is used for alligning faces on input images. Face recognition performance was evaluted on a small subset, provided with pictures of my brother, my girfriend and myself. We used around 30 pictures from every person.

A. Related works

Most famoust facial recognition systems based on deep convolutional networks are:

- DeepFace [6], created by research group at Facebook in 2014. It identifies human faces in digital images. With an accuracy of 97%, it was a major leap forward using deep learning for face recognition.
- The DeepID [7], the system support both identification and verification tasks by training via contrastive loss.
- The VGGFace, In addition to a better-tuned model, the focus of their work was on how to collect a very large training dataset and use this to train a very deep CNN model for face recognition that allowed them to achieve then state-of-the-art results on standard datasets.
- FaceNet [8], also achived state-of-art results on a range of face recognition benchmark dataset. The FaceNet system can be used broadly thanks to multiple third-party open source implementations of the model and the availability of pre-trained models.

II. Methodology

OpenFace is lightweight and minimalist model for face recognition. OpenFace model including model structure and weights is public.

We can see pipeline of the face recognition on figure 1.

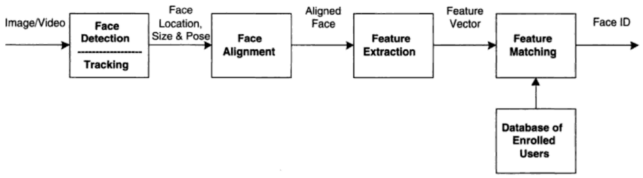

Figure 1. Face detection and getting Face ID

Our program works in 3 stages:

- Detect, transform and crop faces on input images.
- Use the CNN to extract 128 dimensional embeddings.
- Compare input embedding vectors to vectors on video camera.

A. Detection, transformation and cropping

Firstly we needed to set paramater of input to $96x96$ pixels with 3 channels. We then used `CascadeClassififier` to detect faces and then we generated embedding of images inside `images` folder. Firstly we needed to resize input image to $96x96$ pixels and then we needed to change it to `grayscale`. We used `cv` library from `OpenCV` and their methods `resize` and `cvtColor`. Then we used `numpy` [9] to get permuntations of image, rounded to 12 decimals. Last but not least we put that array into `numpy` array. Lastly embidding were calculated from function `predict_on_batch` which assumes that the data we passed is in exactly one batch and thus feeds it to the network. It won't try to split it, that might be problematic for our GPU memory if array is too big.

Second method we wrote was `create_input_image_embeedings` which creates 128D vectors of images inside images folder. This method simpy go through all images and convert each and every one to grayscale and then uses `face cascade` filter to detect faces. We then save each persons embeedings to dictionary.

We then create model and load weights from our `csv` file. Then for each and every picture put those weights on and save our embeedings to `embeedings.npy` file.

B. Use of CNN

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons.

A convolutional layer within a neural network should have the following attributes:

- Convolutional kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Typical structure of convolutional neural network can be seen on figure 2.

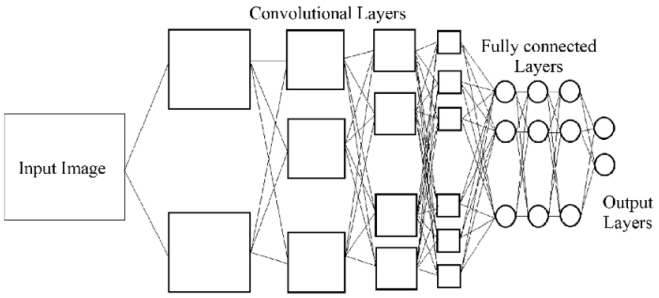

Figure 2. Convulutional Layers

OpenFace model expects 96×96 greyscale images as input. Its has a 128 dimensional output. The model is built on Inception Resnet V5. The Inception network was an important milestone in the development of CNN classifiers. Prior to its inception, most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance. We can see representation of Inception in figure 3

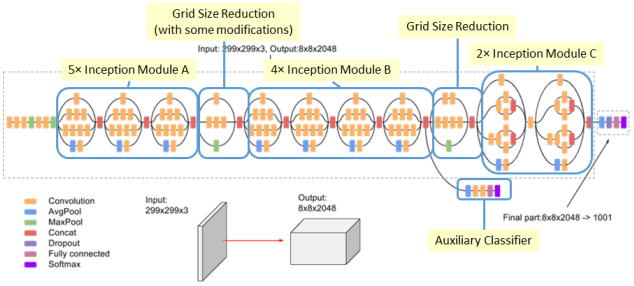

Figure 3. inception

1) Model: Our model have next parameters:

- Total parameters: $3,743,280$
- Trainable parameters: $3,733,968$
- Non-trainable parameters: $9,312$

Model go through 8 inceptions. Model was trained on GPU - GTX 1060. On table I we can see small example of trained model with it's parameters and kernels.

Layer	Kernel	Parameters
conv1	48,48,64	9472
pool1	24,24,64	0
bn1	48,48,64	256
conv2	24,24,64	4160
bn2	24,24,64	256
conv3	24,24,192	110784
bn3	24,24,192	768
conv4a	12, 12, 96	18528
conv4	12, 12, 128	110720
conv5	3, 3, 256	188672
conv6	6, 6, 256	295168
conv7	6, 6, 192	166080
conv8	6, 6, 256	164096
conv10	6, 6, 160	102560
conv11	3, 3, 384	332160
conv12	3, 3, 256	368896
conv13	3, 3, 128	204928
conv14	3, 3, 384	332160
conv15	3, 3, 128	262400
denseLayer	128	94336

Table I
SMALL EXAMPLE OF MODEL

`bn` is Batch Normalization, which normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1. [10]. 2D convolutional layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.

It should be noted, that during training a 128 dimensional float vector is used, but it can be quantized to 128-bytes without loss of accuracy. Thus each face is compactly represented by a 128 dimensional byte vector, which is ideal for large scale clustering and recognition. Smaller embeddings are possible at a minor loss of accuracy and could be employed on mobile devices.

C. Dataset

Dataset for custom identification and verification of person contained around 60 pictures of every person that needed to be detected. Pictures were taken on different positions, different angles and different light settings. Pictures were taken on iPhone 6s Plus [11].

D. Recognizing and identyfing faces

Faces can be recognized from Webcam. Firstly model needs to be created again and weights need to be loaded again from `csv` file, from `keras openFace`.

For recognizing faces we first calculate similiraty between faces with Euclidian distance [12]. If cartesian coordinates p and q are two points in Euclidean n-space then the distance d from p to q is given by Pythagorean formula 4

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2 + \cdots + (q_n-p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

Figure 4. Euclidian distance.

If distance is lower or equal than 0.75 face is correctly detected.

To get results we first need to open Webcam, that is done with help of `OpenCV` library and it's method VideoCapture. We

then save that video to computer in `.avi` format. We apply Cascade Classifier filter and we start detecting faces. When face is detected we check whether identity is in our embeddings or not. If it is then it prints name of the person above the rechtangle of persons face.

III. Results

Assignment detects correct person most of the times. Model recognizes and identifies faces from screen on mobile phone aswell. Face recognition works as shown in Figure 5.

Figure 5. Face recognition on 2 faces.

Model trains on GPU, but it could train on CPU aswell, because datasets are small. If we would use any larger dataset, it should always be used GPU for learning.

Results could be better, if dataset contained more images, or if tweak some additional parameters in CNN training model. Because in current model it rarely changes my face for my brother's.

IV. Conclusion

Without any doubt, Deep learning shows its great power now in relation to Face recognition delivering superhuman performance and great accuracy of output results.

OpenFace is a lightweight face recognition model. It is not the best but it is a strong alternative to stronger ones such as VGG-Face or Facenet. It has 3.7M trainable parameters. This was 145M in VGG-Face and 22.7M in Facenet. Besides, weights of OpenFace is 14MB. Notice that VGG-Face weights was 566 MB and Facenet weights was 90 MB. This comes with the speed. That's why, adoption of OpenFace is very high. You can deploy it even in a mobile device.

Additionally it would be interesting to train small networks that can run on a mobile phone and are compatible to a larger server side model.

References

[1] Wikipedia contributors, "Convolutional neural network — Wikipedia, the free encyclopedia," 2020, [Online; accessed 14-January-2020]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=934584283

[2] "[1503.03832] facenet: A unified embedding for face recognition and clustering," https://arxiv.org/abs/1503.03832, (Accessed on 13.01.2020).

[3] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[4] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[5] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Wikipedia contributors, "Deepface — Wikipedia, the free encyclopedia," 2019, [Online; accessed 14-January-2020]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=DeepFace&oldid=924890754

[7] "Yisun_cvpr14.pdf," http://mmlab.ie.cuhk.edu.hk/pdf/YiSun_CVPR14.pdf, (Accessed on 14.01.2020).

[8] "[1503.03832] facenet: A unified embedding for face recognition and clustering," https://arxiv.org/abs/1503.03832, (Accessed on 14.01.2020).

[9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python," *arXiv e-prints*, p. arXiv:1907.10121, Jul 2019.

[10] "Normalization layers - keras documentation," https://keras.io/layers/normalization/, (Accessed on 14.01.2020).

[11] "Apple iphone 6s plus - full phone specifications," https://www.gsmarena.com/apple_iphone_6s_plus-7243.php, (Accessed on 14.01.2020).

[12] Wikipedia contributors, "Euclidean distance — Wikipedia, the free encyclopedia," 2020, [Online; accessed 14-January-2020]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Euclidean_distance&oldid=934122069