DEEP LEARNING FACE RECOGNITION WITH LIMITED TRAINING DATA

Oliver Wiech
University of Nottingham

ABSTRACT

The following research aims to accomplish successful recognition of faces having limited training data through the application of two Deep Learning algorithms and compare their performance with a traditional Computer Vision method of template matching. Pretrained model of VGG16 architecture is retrained to learn the classification of new samples of faces where each person is represented by only one image. To reduce the time needed to evaluate the large test set, the feature extraction properties of FaceNet were combined with Support Vector Machine classifier. Augmentation techniques and Haar feature-based Cascades Classifiers were used to improve the accuracy further. The study has managed to show that Deep Learning approaches greatly outperform the template matching method in face recognition task at the cost of the effective speed of the algorithms.

Index Terms: Deep Learning, FaceNet, VGG, Siamese Network, Face Recognition, Haar Cascade Classifier, Support Vector Machine

I. INTRODUCTION

Facial recognition used to be a challenging but very important problem in pattern recognition. Face recognition systems are incorporated into many existing systems mainly for the purpose of safety and security with identity verification and authentication. The goal of facial recognition is to find a match between one-to-one or one-to-many facial images by comparing extracted features. The pipeline usually consists of three steps: detecting faces in an image, extracting crucial features of faces, and classification which is assigning the features to a proper person. Images coming from cameras have different qualities, such as lighting, occlusions which may substantially affect the algorithm speed and accuracy, it is crucial to use robust techniques capable of handling these differences. In addition, the algorithm must be quick enough to be able to compare faces for the identification purposes. In the past traditional Computer Vision algorithms were used such as template matching, eigenfaces and Fisherface algorithm. This study aims to look at modern Deep Learning approaches and look at their balance between accuracy and speed.

II. METHODOLOGY

A. Dataset and Image Augmentation

The dataset consists of 100 test images, one per class. The test dataset has 1344 images with associated labels. Both datasets have a mix of color images (RGB) and black/white images (one channel) with differing quality. During reading process, the images are resized to fit the model input layer. FaceNet input requires images to be size of (160, 160, 3), where VGG takes inputs of size (100, 100, 3). For VGG training, the labels are transformed into one-hot encoding vector.

The images are augmented prior to training. Each image is flipped and then the augmentation process uses Keras ImageDataGenerator class to produce random observations. They include transformations of rescaling, rotating, zooming, shifting and brightness manipulation as shown on Fig. 1. This artificially enlarges the training feature space of each class improving model training.

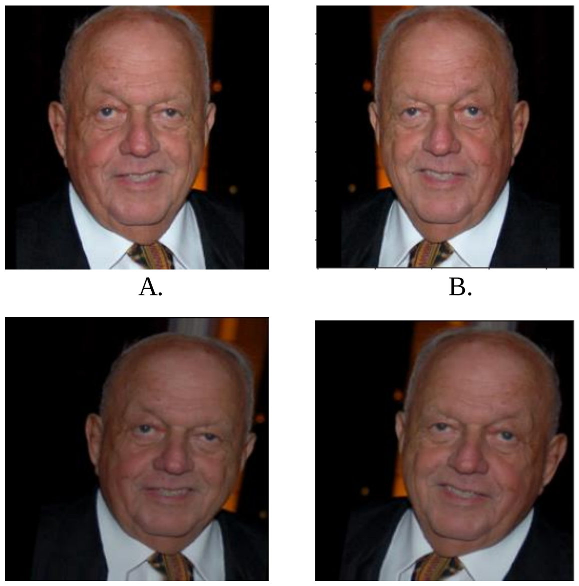

Figure 1. Augmented images used in VGG classification training. A. – original input image. B. – horizontally flipped image. C. and D. – randomly augmented images by rescaling, shifting, zooming and change of brightness.

B. Deep Learning Architectures

1) FaceNet: is a Siamese Network specifically target to solve one-shot face recognition problems implemented by Google in 2015 [1]. Siamese Neural Networks work on two different inputs to rank similarity between them. FaceNet feature extraction part outputs 128-long embedding vectors for two images and then the feature vectors are compared using a triplet loss. It is a unique training procedure where the model is fed by a random image from a class, called an anchor image, and then it compares the embedding of this image to two more images: another one belonging to the same class (positive pair) and one from different class (negative pair). The weights and biases are adjusted so the positive pairs embeddings are closer in the Cartesian system compared to negative pairs. The pre-trained model [2] used in this project in the final version is used only as embedding extractor which then are used as input to a Support Vector Machine. However, experiments were made where the model is retrained and used as both a feature extractor and a classifier. The FaceNet architecture consists of 22 layers, and is based on Inception ResNet v1, a model which combines inception modules with ResNet's residual networks. The inception modules are blocks that take an input and apply several convolutions (1x1, 3x3, 5x5) and a Max Pooling, the outputs are concatenated into one final result. It allows a multi-level feature extraction and has advantage of focusing on many different-level details from the input image. The ResNet's main idea is to combat vanishing gradient problem in which backpropagated gradient to earlier layers becomes very small which degrades performance of very deep networks. The shortcut connections skip some layers which then are trained using a residual mapping: the difference between input I and the output of a function H(I).

2) VGG-Face: is a model that bases on VGG16 which was the winner of ImageNet 2014 competition. The architecture originally consisted of 16 layers but VGG-Face is slightly expanded. The model uses weights pretrained on Labeled Faces in the Wild and YouTube Faces datasets taken from Oxford's Robotic Group [3]. In total, there are 36 layers, including ZeroPadding Layers that adds 0s around the image to make the kernels fit better. The general block of the architecture consists of a sequence of two or three Convolutional Layers, each preceded by a ZeroPadding Layer. At the end of the block there is always a MaxPooling Layer to reduce the dimensions and computational power needed. The number of kernels starts from 64 in the first block and doubles in every next block up to 512 in the two final blocks. However, the last three Convolution Layers have 4096, 4096 and 2622 (equal to number of classes of the original task) kernels respectively, first with size of 7 x 7 and second and third with 1 x 1. The first two layers are followed by Dropout with probability of 0.5 to further stimulate the generalization and combat the overfitting problem. The output of the last Convolution is flattened and given into the changed classification part. All the layers from feature extraction part are frozen and the classification part is adjusted: two Dense Layers are added, one with 512 neurons and 'ReLU' activation function and the final classification Layer of 100 neurons, equal to the number of classes, with 'Softmax' probability distribution function. This resulted in a final model with 147 million parameters, where only 1.5 million are trainable.

C. Supporting Methods

1) Haar Cascades Classifier: one way to boost the performance of FaceNet as an embedding extractor is to input cut-out images of only faces, without the background or any noise that can skew the results. To do this, OpenCV's Haar Cascades Classifier was used [4]. It is a machine learning model which trains on a large number of positive and negatives images of the class we try to detect. Different types of Haar Cascades are provided by OpenCV, the project uses one pretrained on frontal faces. The output of the algorithm is a bounding box that contains the face, which is used to crop the image, as in Fig. 2.

2) Support Vector Machine: it is a supervised machine learning classification algorithm used as the classification method for the FaceNet output embeddings. SVM takes as an input a set of vectors and their corresponding labels and then creates a hyperplane that separates different classes. SVM uses different kernels, mathematical functions, which are used to create the hyperplane. The project uses linear kernel to separate the data since there is only one example per class making it linearly separable.

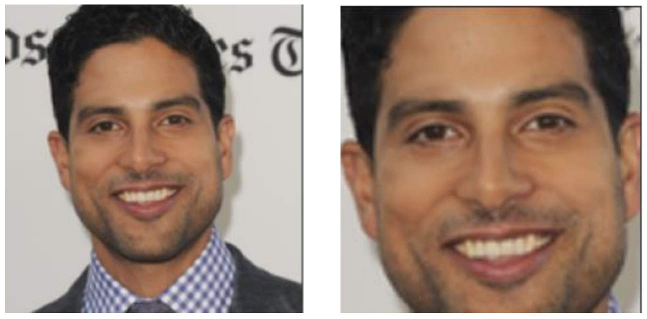

Figure 2. Original image (left), cropped and resized face image using Haar Cascade face detector (right).

D. Hyperparameters, Training and Prediction Process

The retraining of VGG required to set up many different hyperparameters. Starting with the loss function, Categorical

Cross-Entropy is the matching loss to Softmax output. CCE calculates the error by taking the negative sum of the probability for each class (output y_i of the model) multiplied by the log of the probability of the class (target value $\bar{y}_i$), as in Eq. 1. The optimiser used is Adam, which is the most commonly optimiser used due to its little memory requirements, being computationally efficient and quickly finding the global minima of the loss function by taking advantage of moving momentum. The learning rate is set to 1e-3 which is reduced over time using ReduceLROnPlateu class with patience of 2 epochs by a factor of 0.8. The training is done only for 6 epochs because only the classifier part needs to be retrained. The batch size is set only 8 due to memory issues and the data is shuffled after each iteration, a good practice to improve the model'. The embedding vectors before passing to the SVM are standardised by subtracting the mean and divide by the standard deviation to reduce the influence of large mean on the accuracy.

$$\text{CCE} = -\sum_{i=1}^{class\ total} y_i * log\bar{y}_i \tag{1}$$

III. RESULTS

The training loss and accuracy are key metrics to identify the model performance improvement during the training process, shown on Fig. 3. Usually, the dataset is split into train and validation, but due to very small number of data samples acquired, not such split was made.

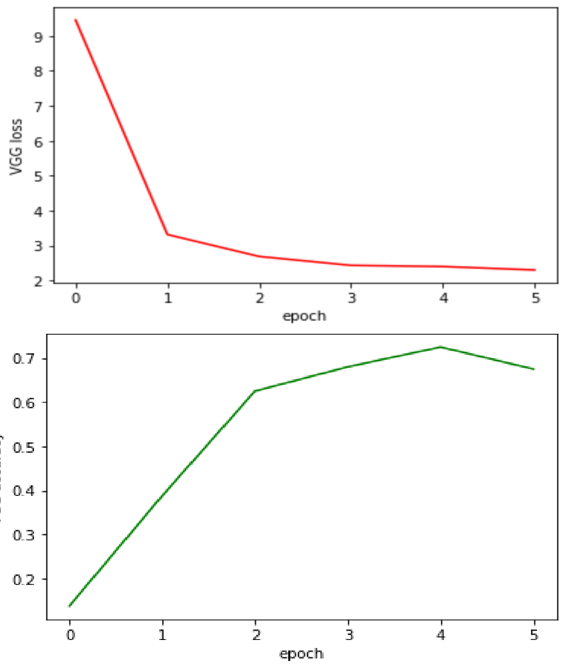

Figure 3. VGG-Face training loss (top) and resulting accuracy on train set (bottom).

The FaceNet model was tested as a stand-alone feature extractor and classifier in one, and in second variant only as embedding extractor with separate SVM classifier, the time difference to execute both approaches is shown on Fig. 4.

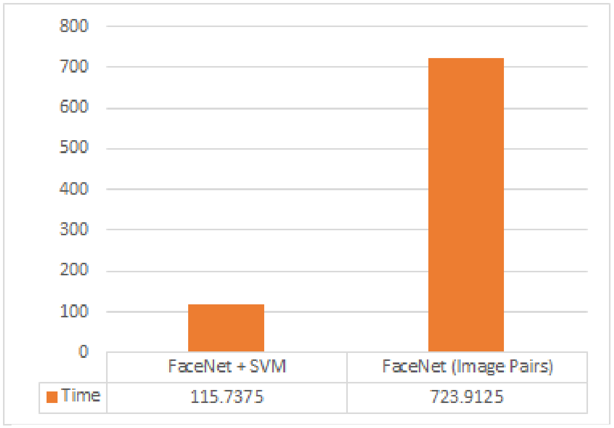

Figure 4. The time [ms] of execution of the two FaceNet approaches.

The study aimed to compare the accuracy between a traditional Computer Vision method which is template matching and two Deep Learning methods, which is presented on Fig. 5.

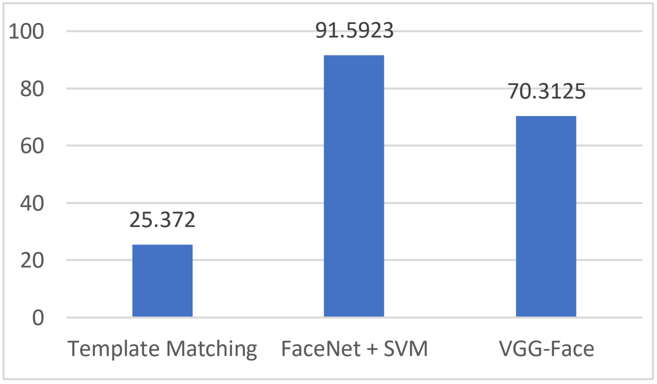

Figure 5. The test data accuracy [%] using the three algorithms: template matching, FaceNet, VGG-Face.

As important as the accuracy, is the speed of the algorithms. The Fig. 6 shows comparison of time [MS] between the three approaches. Table 1 shows the summary of the findings.

	Template Matching	FaceNet + SVM	VGG-Face
Time	29.565	115.7375	220.1347
Accuracy	25.372	91.5923	70.3125

Table 1. Summary of the findings (time [ms] and accuracy [%]) using the three algorithms.

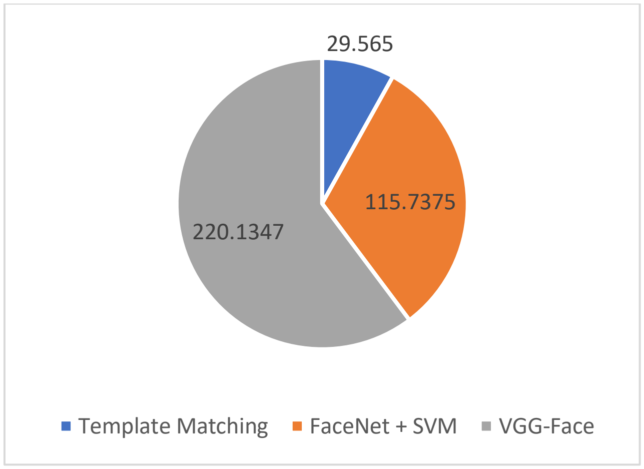

Figure 6. Execution time of the algorithms and their pipeline, including data loading, augmentation, and test-set evaluation.

IV. DISCUSSION

Finding convergence is often a difficult task in training of Deep Learning models. Fig. 3 shows that VGG-Face model training with just few epochs is enough for the classificator to retrain itself to be suitable for the new classes. The weakness of models like VGG is that it is often costly to train them from the scratch – it is a long process that require a large dataset which was not available in this task. Transfer learning used here was one of the solutions. The strength of the VGG-Face model was that it contains both feature extractor and classifier and that it was pretrained on two very large face datasets, meaning it is very well suited in solving the face recognition task. Notable was the ease of use of the model, it worked well as a stand-alone extractor and classifier, it did not need any supporting methods to achieve good balance of execution time to accuracy. The only requirements were specific input size and extensive data-augmentation since VGG-Face was not made for one-shot learning.

FaceNet is, however, a model that was fundamentally made for one-shot face recognition task, and hence it achieved very good accuracy score. The major weakness of this implementation as a sole model was that it requires a lot of time to evaluate test dataset. Since the input is a pair of images, and the output is a similarity score, the model had to go over entire train set for each test sample to compare every test image with all the train classes. It means that if the train set consists of 100 classes and test has size of 1344 images, 134 400 comparisons were required to made, resulting in a very long execution time as shown on Fig. 4. To solve this issue, FaceNet was used only as a feature extractor, and then combined with SVM which resulted in a significant time reduction, going from 724 ms to 116 ms. Another weakness of this method was that any background noise, including items like caps, hats or clothes highly influenced the model score. To prevent that, Haar Cascades Classifier had to be used to give only face regions as the input to the model.

Out of the three tested methods, FaceNet with SVM classifier scored the best accuracy of 91.5 percent on the test dataset. To compare, VGG-Face scored 70.3 percent, and template matching only 25.3 percent, Fig. 5. Table 1 and Fig. 6. show that the baseline method was the fastest with 29.5 secs execution time while FaceNet with SVM was almost four times slower with 115 secs, and VGG-Face required 220 secs to retrain itself and evaluate the test set. It is worth noting that the FaceNet model had to have an external classifier to achieve this time, otherwise the test set evaluation was very long.

V. CONCLUSION

This work has discussed different methods for recognizing faces with limited training data, putting emphasis on Deep Learning approaches, and comparing them to a traditional Computer Vision template matching. A compromise between speed and accuracy of predictions was taken into account in which FaceNet with SVM classifier achieved the best results with its 91.5 percent prediction accuracy and 115 secs execution time. The study has proven that augmentation enables successful results with limited training dataset even on models that were not built with solving one-shot problems in mind, like the VGG-Face. Overall, the Deep Learning algorithms achieved significant improvement in the test accuracy compared to the baseline method, and both FaceNet and VGG-Face could be successfully applied to an identity verification task as long as the model's weaknesses are taken into consideration, which for FaceNet were the necessity of implementing the Haar Cascades Classifier to prepare the input data and implementing separate SVM model to reduce the prediction time. VGG had to have extensive data augmentation, long execution time which included retraining, and even after that it achieved lower score than FaceNet.

The possible further improvements of the project include using a different dataset to verify models applicability to working with different observations – retraining only the classification part as needed. A real-time face recognition system can be created for example by taking input from a camera, which then could be tested by different participants. Another improvement is to use more different types of input data like thermal camera images, also the models could be adapted to 3D face recognition instead of 2D, the test data could be added with the same classes but more different emotions faces. The final suggested improvement could be verifying the model performance against anti-facial recognition systems, which would allow models to test against confusions like different hair, makeup or added noise, making the task more challenging.

REFERENCES

[1] 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.

[2] H. Taniai. FaceNet Pre-Trained Model. Available at: github.com/nyoki-mtl/keras-facenet [accessed 4th May 2021]

[3] O. Parkhi., A. Vedaldi, and A. Zisserman. "Deep face recognition." BMVC. Vol. 1. No. 3. 2015. Available at: www.robots.ox.ac.uk/~vgg/software/vgg_face [accessed 4th May 2021]

[4] R. Lienhart, Frontal Face Haar Cascade.
https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html XML available at: https://github.com/opencv/opencv/tree/3.4/data [accessed 4th May 2021]