

Case Study in Face Recognition: Identification of Celebrity faces

Anand Krishnamoorthy

Final Project - CSCI S-89 Introduction to Deep Learning (Summer 2020)

Abstract

Face recognition has lots of application in our daily life. One example is the Face ID, which is a facial recognition system developed by Apple Inc. to authenticate access to an apple device. The user's face is read by the apple device and it generates a coding for the image. This coding is compared with the codings learnt during the initial face registration. The device then grants access to the system if the codings are a match.

The task of identification of faces is similar. A Neural Network model is developed which learns the features important for distinguishing between different faces. Since the model must distinguish between multiple faces, the task becomes much harder. More the number of faces the model must identify, harder the task becomes.

The dataset "bollywood-celebrity-faces" is a publicly available dataset on Kaggle which has 33 different celebrities and has approximately 100 sample images for each celebrity. Intention of this project was to apply 'Deep Learning' techniques to identify with maximum accuracy the match in facial features to the 33 celebrities.

The project mainly deals with Transfer Learning of the VGGFace model, which identifies 2622 different faces with high accuracy. The project also evaluates the performance of techniques like:

- Image Augmentation
- Image cropping
- Freezing weights of different layers in the VGGFace model
- Different classifier architecture on top of VGGFace model

Table of Contents

1. Goal
2. Hardware and software
3. Dataset
4. Implementation
5. Results
6. References

1. Goal

Goal of the project is to apply ‘Deep Learning’ techniques to recognize celebrity faces. During the training phase the model tries learns features important to recognize different faces based on the training data provided. Using the trained model, the model produces the top 3 celebrity matches for a new input image.

This project can also answer the question ‘which common celebrity do you resemble?’ The project can be modified to produce a model similar to the ‘Face ID’ by Apple Inc.

2. Hardware and Software

The project was run on Google Colab with a GPU setting (Tesla p100). Although it is not required to run on GPU, running the code on GPU reduces training time.

Python 3 was utilized for development of the code. Make sure below packages are installed:

- Keras 2.3
- Tensorflow 2.2
- MTCNN

All other packages come pre-install with google colab.

3. Dataset

The dataset used during training is available publicly in [Kaggle](#). The dataset consists of 33 celebrities with approximately 100 images for each celebrity. Fig. 1 shows the sample images from the dataset used during training.

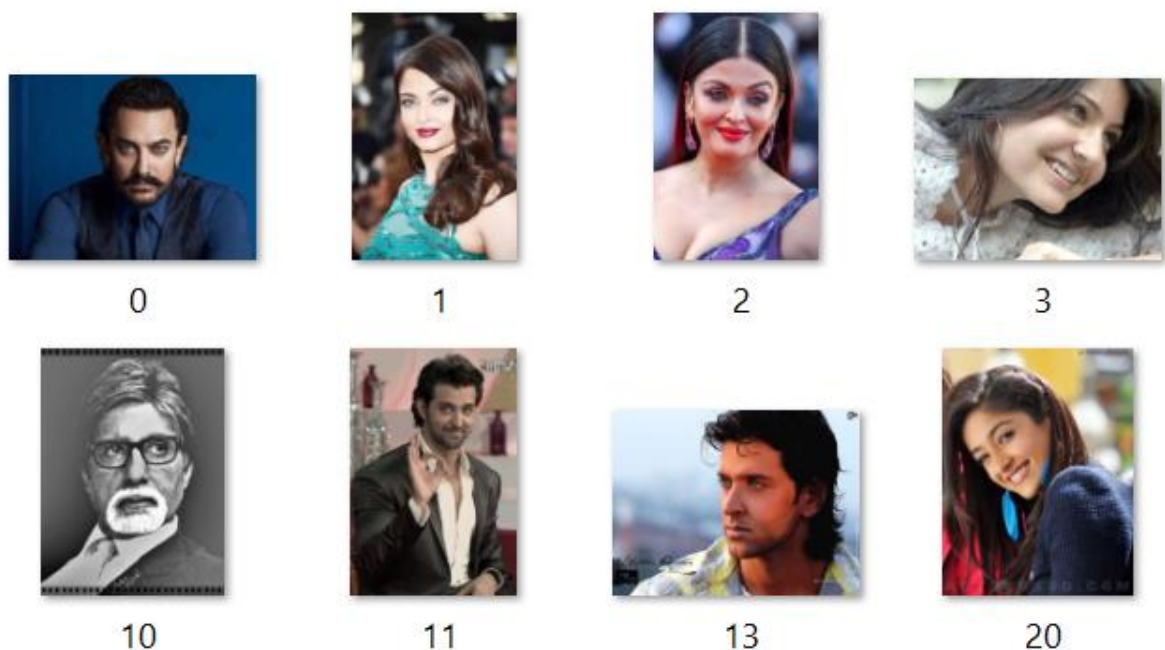


Fig. 1. Sample images used during training

4. Implementation

Step 1:

The data is split into train and test using the code from ‘Train Test Split.ipynb’ ([Original code.](#))

Step 2:

Crop faces of the celebrity and save the images using the ‘ExtractFace.ipynb’ script. The cropped images will be used for further training processes. It uses the MTCNN package as a blackbox to crop the face from images ([Package developer](#)).

Step 3:

The VGGFace is used as the base model for training the images. The weights of the initial layers of VGGFace are frozen and weights of the last few layers are fine tuned to achieve the desired accuracy. The original implementation of the VGGFace model was built to classify 2622 faces. Fig. 2 shows the visual representation of the VGGFace model.

Build a classifier model on top of the base model. Fig. 3 shows the architecture used for the classifier. The project also takes advantage of ‘ImageGenerator’ class in keras for image augmentation to improve classification accuracy.

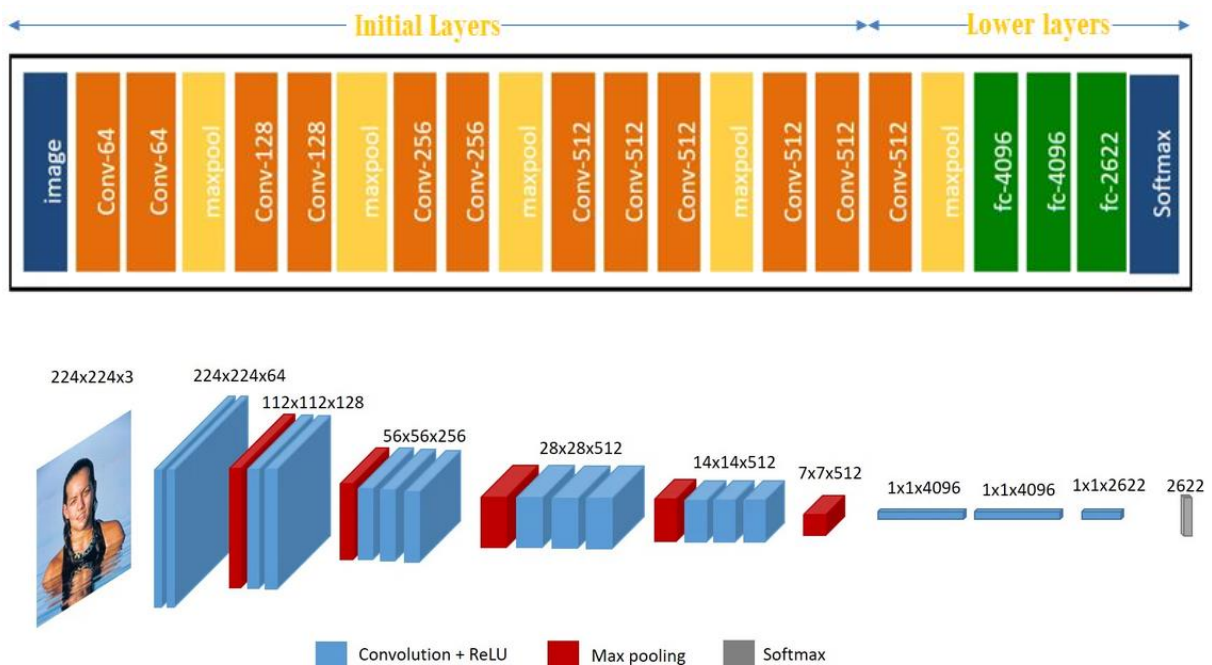


Fig. 2. Visualization of VGG face. [Original article](#)

Step 4:

Input test data and get the top 3 celebrity matches.




Layer (type)	Output Shape	Param #	
model_1 (Model)	(None, 2622)	145002878	 VGGFace Model
dense_1 (Dense)	(None, 1024)	2685952	 Classifier
leaky_re_lu_1 (LeakyReLU)	(None, 1024)	0	
batch_normalization_1 (Batch Normalization)	(None, 1024)	4096	
dropout_3 (Dropout)	(None, 1024)	0	
dense_2 (Dense)	(None, 512)	524800	
leaky_re_lu_2 (LeakyReLU)	(None, 512)	0	
batch_normalization_2 (Batch Normalization)	(None, 512)	2048	
dense_3 (Dense)	(None, 128)	65664	
leaky_re_lu_3 (LeakyReLU)	(None, 128)	0	 Output
dense_4 (Dense)	(None, 33)	4257	

Fig. 3. Classifier architecture

5. Results

- Model Performed best with Fine Tuning last 8 layers of VGGFace with Cropped images and using Data Augmentation.
- Fully connected Neural Network as classifier added on top of VGGFace.
- Model performed much better compared to few implementations posted online.
- Accuracy achieved is lesser compared to the original VGGFace implementation.

Model achieved 86% accuracy in the validation set. Table 1 shows the accuracy comparisons for various techniques used in the project. Fig. 4 shows the loss and accuracy for each epoch by the model during training.

Specification	Images Cropped	Image Augmentation	# Classes	Val. accuracy obtained	Val. Top 2 classes	Train accuracy obtained	Train Top 2 classes
Partial Retraining of VGGFace Layers + Fully Connected NN	No	Yes	33	23.00%	34.50%	33.50%	49.00%
Partial Retraining of VGG Layers + Extra 1D CNNs and Fully Connected NN	Yes	Yes	33	64.00%	74.00%	66.00%	80.00%
Partial Retraining of VGG Layers-training last 6 layers + Fully Connected NN	Yes	Yes	33	79.00%	80.00%	94.00%	98.00%
Partial Retraining of VGG Layers-training last 8 layers sparse + Fully Connected NN	Yes	Yes	33	86.00%	93.00%	97.00%	99.00%

Table 1. Accuracy comparison for various techniques

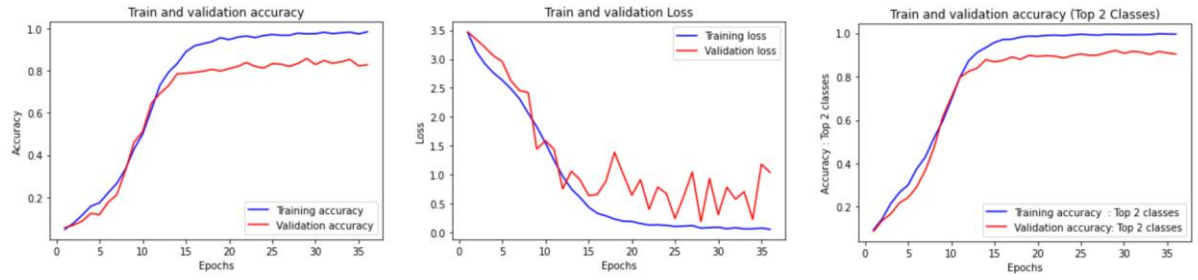


Fig. 4. Model accuracy and loss obtained

6. References

- <https://gist.github.com/EncodeTS/6bbe8cb8bebad7a672f0d872561782d9>
- <https://drive.google.com/file/d/0B4ChsjFJvew3NkF0dTc1OGxsOFU/view>
- <https://machinelearningmastery.com/how-to-perform-face-recognition-with-vggface2-convolutional-neural-network-in-keras/>
- <https://medium.com/analytics-vidhya/face-recognition-with-vgg-face-in-keras-96e6bc1951d5>