CS299 INNOVATION LAB

END TERM REPORT

HUMAN FACE CLASSIFICATION ON THE BASIS OF THE COUNTRY THROUGH DEEP LEARNING TECHNIQUES

TEAM MEMBERS-

ADITYA PRAKASH PATRA(1601CS03)

VEDAVIKAS POTNURU(1601CS31)

Abstract

Face classification has been a field of interest since a decade .Variety of research,Articles have been published on the same.Face conveys the most direct and fastest impression of an individual. We can often guess a person's country by the way he or she looks.National identification unlocks important demographic information with many application in biomedical and sociological research.Studies have shown that race is subconsciously one of the first information retrieved by a human brain upon perceiving a face.This code finds a wide application in tourism , e-commerce ,social media marketing and criminal justice,and counter terrorism. Recently deep learning methods especially Convolutional Neural Network(CNN) have shown promising result in computer vision with the breakthrough results in image classification and recognition.These advancements in deep learning happens due to availability of more computational power and more datasets for training.Further here we trained multiple facial and attribute classifier to identify the most distinctive features for each country.

AIM AND OBJECTIVES:

This project aims to distinguish a collection of faces countrywise using Deep learning techniques.Our objective is to classify the human face which provides wealth of info about identity,nationality,ethnicity which remains invariant throughout the lifetime and supports face recognition systems and thus received large attention in these years.Gleaned the power of the recent advances in computer vision and machine learning, We took the challenge to investigate how accurate is it possible to classify locals of a group..Further here we trained multiple facial and attribute classifier to identify the most distinctive features for each country.

NOVELTY:

As we know that recognition of faces countrywise is not an easy task due to the images possessing different facial features,colour and having large population.This project will tackle one of the difficult problems in race classification. It is in a subset of race classification classification.it is difficult to classify the faces which brings the necessity of our algorithm to classify the faces which might be used in e-commerce,security,marketing,tourism as well. This finds application in human computer interaction,face based recognition , surveillance and defence

The above photos can be classified as per their respective countries

Classification Pipeline

- RAW DATA COLLECTION
- NOISE REMOVAL
- FACE DETECTION,CROPPING
- FEATURE SELECTION
- SPLITTING INTO TRAIN,VALIDATION AND TEST DATA

- MODEL DEVELOPMENT
- PREDICTION OF OUTPUTS AND ACCURACY ,LOSS VISUALIZATION
- PERFORMANCE TUNING THROUGH DIFFERENT OPTIMIZATION METHODS

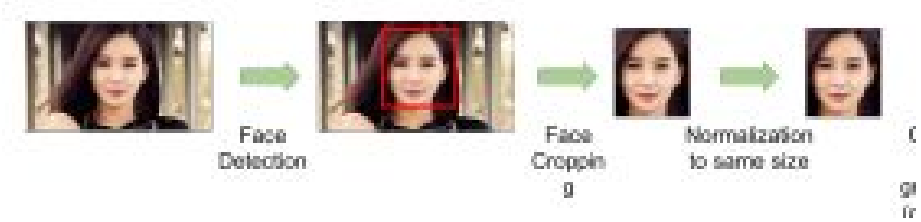
Face Dataset Collection:

Despite there are large-scare facial image databases available, such as Chicago Face Database,CAS PEAL database etc.., these databases are not applicable to our objective, as the images from these datasets are not labeled by the nationality. Therefore, we manually harvested images which contains profile photos of public figures from the six countries. All images were directly downloaded from the internet and also we utilized the Google API and other govt sites to collect people faces from the country after which we have around 2500 photos each which comprises of six countries (India,Korea,USA,philippines,kenya and Indonesia.)

Image Preprocessing:

Image preprocessing was conducted before training a classifier.The input facial images can be of varying sizes. We normalized the faces so that each training datapoint would have the same dimension. We first cropped the face using Haar Cascade Face Detection algorithm packaged with OpenCV[9] such that the noise from the image background is eliminated. And then we reframed the faces to 100 *100 images for the 2-layer Neural networks and CNN This is because a larger image significantly decreases the computational speed for neural

network approaches. Afterwards we applied different preprocessing methods, including normalization,whitening,dimensionality reduction.



Data Split:

We split our dataset into training set, validation set, and test set (8: 1: 1) and experiment with different architectures from shallow networks (3-5 layers) to the 16-layer VGG and Xception layer. In our experiments, all networks would converge (Figure 4), but we observe that as the network gets deeper, we are able to achieve better results , from 60% accuracy with shallow networks to an overall accuracy of 65% with CNN .

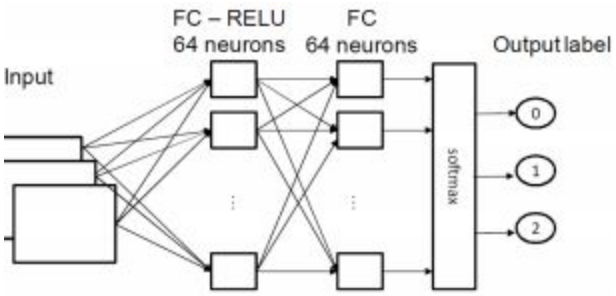
Classifiers:

I have tried both Two-Layer Neural Networks , CNN models from scratch and also used the Fine-tuned pre-trained networks such as VGG-16 and Xception net.

Two-Layer Neural Network:

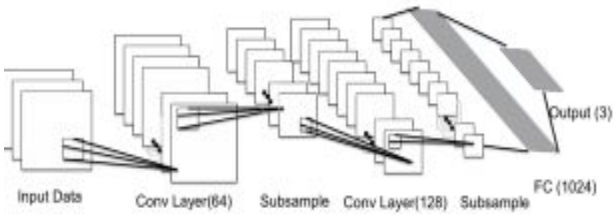
We proposed two neural network approaches to predict nationality. First we constructed a two-layer neural network. Each pre-processed facial image is flatten to a row vector which has a dimension of (100 * 100 * 3) (RGB) then all hidden layers are stacked together and fed into

the network. The network has a Rectified Linear Unit (ReLU) layer sandwiched by two fully connected layers. The ReLU layer computes the function(f(x) = max(0, x)).Due to large no of parameters it is difficult to train and it might also lead to overfitting/underfitting.



Convolutional Neural Network:

Next, we built and trained a convolutional neural network (CNN) using keras and TensorFlow Library works as a backend. We built a 5-layer CNN whose architecture is shown in Figure 3. The CNN consists of three convolutional layers (with 32 and 64 filters respectively) with Relu Activation, one fully-connected layer (with 500 neurons) and a dropout layer, followed by an output layer . By tuning the hyperparameters such as learning rate, dropout rate and optimizers we have tried to improve the accuracy of our basic CNN model.



Using the pre-trained imagenet networks:

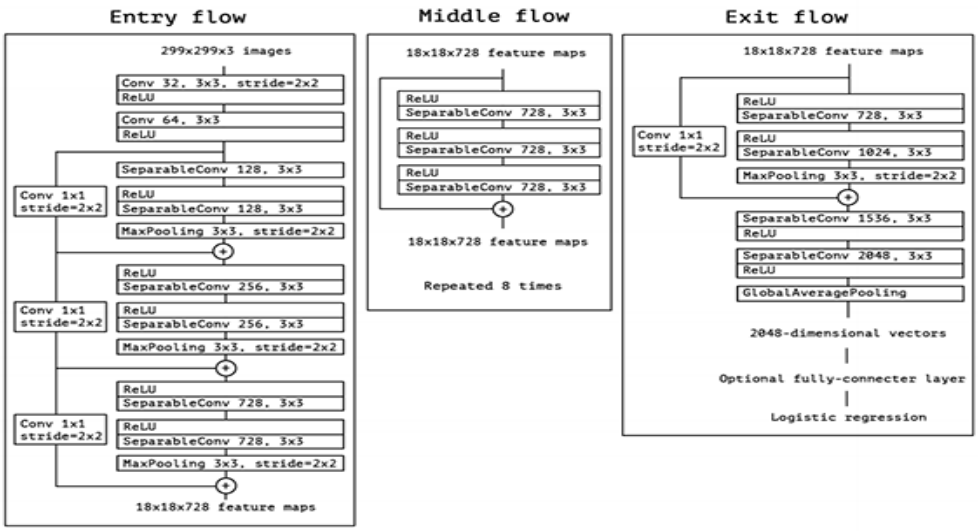
We have used different pre-trained networks such as VGG-16 ,Xception net,etc.. and fine-tuned the networks by making top-layers trainable and also added the FC-layers so as to make to implement on the six outputs and trained on our data which increased the accuracy score of the data.

VGG-16 Architecture

The VGG network architecture was introduced by Simonyan and Zisserman.This network is characterized by its simplicity, using only *3×3* convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier (below).Itt consists of 16 convolutional layers and is very appealing because of its very uniform architecture. It consists of 3*3 convolution layers ,282 pooling layers. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. VGGNet consists of 140 million parameters, which can be a bit challenging to handle.In this net we tried to fine-tune the last 2 blocks of convolution and pooling layers which would try to combine the bottle neck features of the VGG and also some high level features of the specific data provided.which resulted in increase in prediction rate.

Xception Architecture

Xception was proposed by none other than Francischollet, the creator and chief maintainer of the Keras library. Xception is an extension of the Inception architecture which replaces the standard Inception modules with depth wise separable convolutions.In this net we tried to fine-tune the network which would try to combine the bottle neck features of the Xception and also some high level features of the specific data provided.which resulted in increase in accuracy rate.



TRAINING AND TESTING THE DATASET:

The cropped images and labels of the training ,validation and testing data set obtained after the pre-processing methods were used to train the data based on four CNN Nets i.e. 2 -D Neural net and CNN classifiers ,finetuned on imagenet models such as VGG-16 and Xception net separately in four different files. Some important frameworks were used to carry out the training process are Keras,tensorflow,matplotlib,numpy etc..were used and ranthe code in the Google Collab server(GPU).

FINAL PREDICTION :

After training the dataset we saved the model and made the saved model to run on a different image final accuracy was predicted and displayed on the terminal The outputs are based on the country number(0,1,2,3,4,5) which 0 describes that person in the image is korea similarly 1-India,2-USA,3-Indonesia,4-Philippines,5-Kenya respectively.Some important libraries used were Keras,tensorflow,numpy,sklearn which has functions to predict the accuracy score and also confusion matrix through which we can get an idea of the classification of the more data.

TOOLS AND TECHNOLOGY USED:

- Python 3.6
- Keras
- Tensorflow (backend)
- Scikit-learn
- Numpy
- MatplotLib for plotting Graphs
- Google Collabs server(GPU accessibility)

LIMITATIONS:

The lack of face datasets labeled with nationality is a general problem in face recognition. Hence, in order to build a classifier that is robust to these changes, a bigger dataset has to be constructed and also other better pre-processing methods has to be needed to train the classifier so as to get better predictive results.Some models are been either overfitted/underfitted.

KEY INNOVATION ACHIEVED:

As we know that recognition of faces countrywise is not an easy task due to the images possessing different facial features,colour and having large population.There have been several attempts on

nationality classification until now and further research is going on in this field.In this field until now probably a great accuracy had not been achieved yet.But our project tends to attain good accuracy of more than 70% on test data which can be further increased with more data and using better pre-processing techniques.

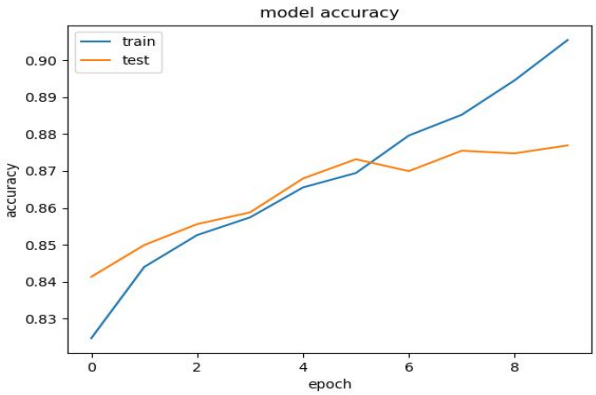
RESULTS:

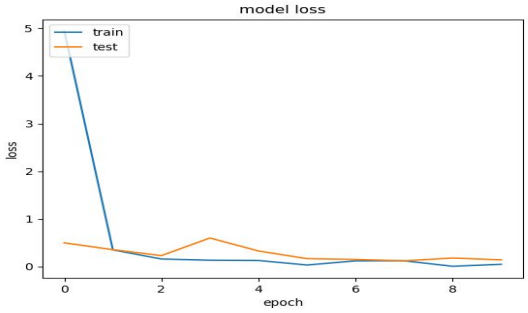
The table below summarizes the results obtained from various models. Which were tested against the datasets. We took 10% of the dataset to test on the model trained using training and validation datasets.

Comparison between different approaches

MODEL	ACCURACY(%)
Two-Layer Neural Network	50%
CNN(convolutional neural networks)	65%
VGG-16 Net(Pre-trained Network)	69.5%
Xception Net(Pre-trained Network)	72.6%

Accuracy & Loss Vs epochs - Graph for VGG-16 net





Model Summary & Simulation

```
Found 1201 images belonging to 6 classes.
Epoch 1/20
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:23: UserWarning: The semantics of the Keras 2 argument `steps_per_epoch` is not the same as the Keras 1 argument `samples_per_epoch`. `steps
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:23: UserWarning: Update your `fit_generator` call to the Keras 2 API: `fit_generator(<keras.pre..., validation_data=<keras.pre..., steps_per
550/550 [==============================] - 125s 227ms/step - loss: 0.1525 - acc: 0.9317 - val_loss: 0.2736 - val_acc: 0.9156
Epoch 2/20
257/550 [=============>................] - ETA: 46s - loss: 0.1339 - acc: 0.9367550/550 [==============================] - 125s 227ms/step - loss: 0.1383 - acc: 0.9375 - val_loss: 0.1793 - val_acc: 0.9
Epoch 3/20
353/550 [==================>...........] - ETA: 31s - loss: 0.1386 - acc: 0.9366550/550 [==============================] - 125s 227ms/step - loss: 0.1354 - acc: 0.9368 - val_loss: 0.1711 - val_acc: 0.9
Epoch 4/20
388/550 [====================>.........] - ETA: 25s - loss: 0.1162 - acc: 0.9457550/550 [==============================] - 125s 227ms/step - loss: 0.1169 - acc: 0.9467 - val_loss: 0.2107 - val_acc: 0.9
Epoch 5/20
401/550 [====================>.........] - ETA: 23s - loss: 0.1207 - acc: 0.9467550/550 [==============================] - 125s 227ms/step - loss: 0.1164 - acc: 0.9495 - val_loss: 0.3294 - val_acc: 0.9
Epoch 6/20
406/550 [====================>.........] - ETA: 22s - loss: 0.1136 - acc: 0.9541550/550 [==============================] - 124s 226ms/step - loss: 0.1112 - acc: 0.9542 - val_loss: 0.2232 - val_acc: 0.9
Epoch 7/20
408/550 [====================>.........] - ETA: 22s - loss: 0.0918 - acc: 0.9630550/550 [==============================] - 124s 226ms/step - loss: 0.0976 - acc: 0.9618 - val_loss: 0.2501 - val_acc: 0.9
Epoch 8/20
409/550 [====================>.........] - ETA: 22s - loss: 0.0866 - acc: 0.9668550/550 [==============================] - 124s 226ms/step - loss: 0.0930 - acc: 0.9668 - val_loss: 0.2347 - val_acc: 0.9
Epoch 9/20
409/550 [====================>.........] - ETA: 22s - loss: 0.0837 - acc: 0.9721550/550 [==============================] - 125s 228ms/step - loss: 0.0895 - acc: 0.9711 - val_loss: 0.2670 - val_acc: 0.9
Epoch 10/20
407/550 [====================>.........] - ETA: 22s - loss: 0.0789 - acc: 0.9738550/550 [==============================] - 125s 227ms/step - loss: 0.0784 - acc: 0.9742 - val_loss: 0.2281 - val_acc: 0.9
Epoch 11/20
407/550 [====================>.........] - ETA: 22s - loss: 0.0797 - acc: 0.9744550/550 [==============================] - 125s 227ms/step - loss: 0.0734 - acc: 0.9765 - val_loss: 0.2528 - val_acc: 0.9
Epoch 12/20
407/550 [====================>.........] - ETA: 22s - loss: 0.0699 - acc: 0.9788550/550 [==============================] - 125s 227ms/step - loss: 0.0780 - acc: 0.9795 - val_loss: 0.3256 - val_acc: 0.9
Epoch 13/20
406/550 [====================>.........] - ETA: 23s - loss: 0.0715 - acc: 0.9818550/550 [==============================] - 125s 227ms/step - loss: 0.0681 - acc: 0.9818 - val_loss: 0.3243 - val_acc: 0.9
Epoch 14/20
406/550 [====================>.........] - ETA: 23s - loss: 0.0601 - acc: 0.9856550/550 [==============================] - 125s 228ms/step - loss: 0.0549 - acc: 0.9859 - val_loss: 0.3289 - val_acc: 0.9
Epoch 15/20
406/550 [====================>.........] - ETA: 22s - loss: 0.0571 - acc: 0.9850550/550 [==============================] - 125s 227ms/step - loss: 0.0587 - acc: 0.9845 - val_loss: 0.3435 - val_acc: 0.9
Epoch 16/20
406/550 [====================>.........] - ETA: 23s - loss: 0.0538 - acc: 0.9876550/550 [==============================] - 125s 227ms/step - loss: 0.0489 - acc: 0.9881 - val_loss: 0.3268 - val_acc: 0.9
Epoch 17/20
406/550 [====================>.........] - ETA: 22s - loss: 0.0732 - acc: 0.9861550/550 [==============================] - 124s 226ms/step - loss: 0.0629 - acc: 0.9874 - val_loss: 0.4233 - val_acc: 0.9
Epoch 18/20
406/550 [====================>.........] - ETA: 22s - loss: 0.0736 - acc: 0.9867550/550 [==============================] - 124s 225ms/step - loss: 0.0807 - acc: 0.9869 - val_loss: 0.3825 - val_acc: 0.9
Epoch 19/20
406/550 [====================>.........] - ETA: 22s - loss: 0.0705 - acc: 0.9882550/550 [==============================] - 125s 227ms/step - loss: 0.0658 - acc: 0.9886 - val_loss: 0.3958 - val_acc: 0.9
Epoch 20/20
```

References:

● Lu Xiaoguang and Jain Anil K 2004 Ethnicity Identification from Face Images Proceedings of SPIE

● Tariq Usman, Hu Yuxiao and Huang Thomas S 2009 Gender and Ethnicity Identification from Silhouetted face profiles 16th IEEE International Conference.

● Using Deep Learning to classify Dogs and Cats Ravish Chawla Conghui Fu Qiqi Ai Tianyu Li

● Keras Documentation(https://keras.io/)

● Deep Learning with Keras by Antonio Gulli and Sujit Pal

● Do They All Look the Same?Deciphering Chinese,Japanese by

Deep learning by Yu wang,Liao,Feng-https://arxiv.org/pdf/

● Handwriting Digit Recognition Using Convolutional Neural Network In python with keras - By Jason Brownie.

● Learned Features are better for Ethnicity Classification Inzamam Anwar, Naeem Ul Islam

●https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

●https://www.kaggle.com/crawford/monkey-classifier-cnn-xception-0-90-acc