Ethnicity Identification using Machine Learning Models - Report

Ankur Vineet*, Aman Sharma†, Vaibhav Nautiyal‡
Computer Science & Information Systems, Birla Insitute of Technology & Science, Pilani, IND.
Email: *h20180144@pilani.bits-pilani.ac.in, †h20180137@pilani.bits-pilani.ac.in, ‡h20180136@pilani.bits-pilani.ac.in

***Abstract*—In recent times, Ethnicity Identification has become an important routine in profiling of an individual, mostly with the perspective of law and order. The rise in the number of image capturing devices with advent of high performing hardware has enabled us to perform a wide variety of image processing task. This has led to novel applications in the field of machine learning. Identifying the ethnicity of a person is one of the hot topics that has been around but has gained interest with the popularity of deep learning algorithm. Faces of individuals of different ethnicity show variations along the lines of bone structure, eyes, shape and size of nose however, there is only a broad group of ethnic population that can be successfully identified through machine learning algorithms.**

***Index Terms*—Ethnicity, CNN, SVM , Deep Learning**

I. Introduction:

A person's face is able to communicate a lot of implicit personal information about him/her without a verbal communication. For instance, their gender, age, emotional state etc. can be implicitly known. There's sufficient amount of research that shows that 'race' is arguably the most projecting trait of an individual. The topic has been studied in different disciplines from different angles.

In computer science, the topic is of interest from the perspective of profiling an individual. This profiling could be because of variety of reasons for example an investigative study (forensics), for aggregating demographic data (Aadhar, Social Security Number) etc. This task can be realized (to some success) by combining the strength of image processing and machine learning algorithms.

There has been some research on this topic in order to accomplish the task. The general perspective is that, ethnicity of an individual can be defined in terms of certain biometric features, especially of that of face. For example, certain jaw-structure is more common in major population of a certain ethnic group, shape and size of nose of middle-eastern population is very different from that of east Asians. These fundamental and stark differences help us to model the problem as a pattern recognition task which is exceptionally carried out by computers. Along the same lines, we modeled the problem as a machine learning task and used different approaches to recognize the best suited method to implement it. This report has been divided into various sections. Section-II describes the dataset which was used. Section-III reports the preparation of the dataset. It is followed by two different approaches, one being the traditional approach called "Support Vector Machine" while the other which has recently gained popularity called Deep Learning, specifically the Convolutional Neural Networks. In Section V, we compare the results of the two approaches. Section VI is the conclusion of the experiment.

II. Dataset:

The name of the dataset used by us to model different techniques is "UTKFace", which is a large scale face dataset. It has 20K+ facial images of people which is labeled with the age of the person in the image, gender of that person and most importantly for us which ethnic group that person belongs to. The ethnic groups were categorized as five groups namely "White", "Black", "Asians" (which in general reperesents East Asians like Chinese, Japanese,Koreans etc.), "Indians" (which broadly consists of South Asians that is belonging to the Indian subcontinent), and "Others" (which includes images of rest of the ethnic groups like Latinos, Hispanics, Arabs etc.) The filename of each image contained all the above mentioned details. The information was encoded as follows:

TABLE I
Dataset Image Properties

Image Attribute	Value
Resolution	200*200*3
Color Mode	RGB
Alignment	In-Plane Rotation with eyes

- [age] $\in$ (0, 116)
- [gender] - 0 denotes male , 1 denotes female
- [race] - [0,1,2,3,4] denoting White, Black, Asian, Indian and Others respectively
- [date & time] - showing the date and time that image was collected in the dataset

Fig. 1. Sample Image

Example of a file name is 21_0_0_20170110232137372 denoting the image shown in Figure 1, which represents a male who is 21 years old and belongs to white ethnic group. The initial dataset which has 20000+ images consisted almost 50 percent images belonging to white ethnic group, which is not a correct distribution for training, so we sampled 17,205 images randomly having almost equal distribution for each ethnic group. The summary of distribution is given in Table 2, while a general overview of variety of dataset is given in Figure 2.

TABLE II
SUMMARY OF DATASET USED

Ethnic Group	Number of Images
White	3578
Black	4526
Asian	3434
Indian	3975
Other	2692

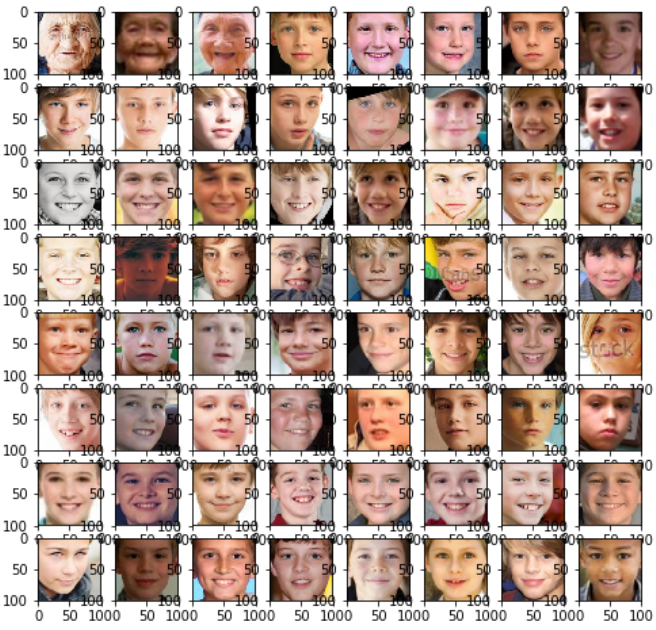

Fig. 2. Images in UTKFace

III. PREPARATION OF DATASET

Preparation of dataset is an important step in the machine learning pipeline in order to reduce redundancy and thus reducing the load on hardware for model processing. The images were in a 200 * 200 * 3 resolution in RGB color model. The images were downscaled to 100 * 100 * 3 resolution. This was done taking the hardware into consideration, as the original dimension of image would increase the training time significantly without contributing much in terms of feature of face. Almost all significant features can be extracted from 100 * 100 image, without much information loss. From this set of images futher two more datasets were created one maintaining the color model and other without it i.e. Grayscale. The labels were extracted from the filename of images and ported to a vector of dimensions 17205 * 1, while the images were stored in an array of dimension 17205 * 30000. These images and labels were converted to a platform independent array format so that it can be used to train different model without dealing with image files again. The code snippet of performing the preprocesing steps are shown in Figure 3.

```
import cv2
import numpy as np
from PIL import Image
import os, sys


path = "/home/FDUSER/Downloads/UTKFace/"
dirs = os.listdir( path )
Y_data = []
X_data = []
def resize():
    for item in dirs:
        if os.path.isfile(path+item):
            f, e = os.path.splitext(path+item)
            itemname, e = os.path.splitext(item);
            race = itemname.split('_');
            Y_data.append(race[2]);
            #print(item)
            im = Image.open(path+item)
            imResize = im.resize((100,100), Image.ANTIALIAS)
            imResize.save(  path+'resized/'+ itemname+ '.jpg', 'JPEG', quality=90)
            image = cv2.imread (path+'resized/'+item)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            X_data.append (image)

resize()
Y_data = np.array(Y_data,dtype='uint16')
Y_data.shape
```

```
(23705,)
```

Fig. 3. Preprocessing

IV. MACHINE LEARNING MODELS

The problem is a multi-class classification task. Some promising methods for problems under this category are:

- SVM
- Decision Trees
- k-Nearest Neighbors
- Naïve Bayes

A. K-Nearest Neighbour

k-nearest neighbor is one of the popular non parametric method employed for classification. The input to the classifier consist of k closest training data in the feature space. The neighbors are taken from a set of known instances of the data.

B. Support Vector Machines

Support Vector Machines look to maximize the margin separating the different classes. It is quite a popular technique but originally was meant for binary classification. However, additional parameters and constraints added to it, enables multi-class classification problem.

There are two kinds of SVM, Linear and Kernelized. Kernelized SVMs use a kernel function to transform the data such that resulting input data is linearly separable. For the

image, kernelized SVM perform better owing to their high dimensions.

We train our model with a Kernelized SVM using a Radial Basis Function (rbf) kernel.

$$K(x, x`) = e^{\left(\frac{-\|x-x`\|^2}{2\sigma^2}\right)}$$

SVM method can be applied in two ways to the dataset, with Principal Component Analysis (PCA) and without PCA. Without PCA the SVM is trained without any prior dimensionality reduction on image data, whereas with PCA dimensionality reduction is performed prior to model training.

C. Deep Learning: Convolution Neural Network

Deep learning has proved its mettle when it comes to deal with high dimension complex data such as images. Convolutional Neural Network, specifically are most suited to the task. They are basically a variation of neural networks consisting of convolutional layer, pooling layer, fully connected layer. The traditional Neural Nets suffer from "Curse of Dimensionality" and do not scale well to higher resolution images. One of the popular architecture is VGG-16 which we have used to train our CNN model, which is shown in Figure 11 in Appendix.

V. Building Models

Based of the models discussed in Section IV we built five models for our dataset which were namely:

1) K-Nearest Neighbour
2) SVM - without PCA
3) SVM - with PCA
4) CNN - RGB images
5) CNN - B/W images

A. KNN

It is the simplest of the model that was trained for the task. The input to the model was a 2-D array of size 17,205 * 30000 with labels. This input was further divided into train and test splits in 85:15 ratio giving 14,625 instances for training and 2580 instances for testing. As their were five output classes the parameter, number of neighbours for KNN was also 5. The model gave an accuracy of 0.48, but still a huge scope of improvement is there.

B. SVM - without PCA

In this model also similar kind of input that was given in KNN model was used that is 85:15 ratio of input and array size as 17,205 * 30000. The model gave an accuracy of 0.65 which was better than KNN but less than the acceptable accuracy measure.

C. SVM - with PCA

With PCA first the dimensionality reduction was applied to the input array, then the reduced dimension array was fed into the SVM model. The model gave an accuracy of 0.73, interesting thing was that the training time got significantly reduced because of reducing the dimensions and accuracy also got increased by a little bit.

D. CNN - RGB image

As CNN is specifically designed for image classification, it applies a number of filters to identify features in images. Hence, it takes input in form of array of size 17,205 * 100 * 100 * 3 which is a 4-D tensor. The filter size was (3,3) and four layer of convolution layers were there in the model as shown in Figure . We ran the model for 25 epochs and final validation accuracy was 0.77.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(256, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

Fig. 4. CNN Model

E. CNN - Black and White image

The motivation for this model was to see whether skin colour is an important feature for ethnicity identification. So,

we first converted all the RGB images into grayscale format. Same model that we used for CNN with RGB was used for it, the only difference being the input array was of size 17,205 * 100 * 100 * 1. Running this model for 25 epochs we got the final validation accuracy as 0.76, which shows that skin colour is not an important feature to detect ethnicity.

VI. Result and Analysis

We will first analyse the Convolution neural network as it gave the best accuracy among all the models. We investigated the loss and accuracy parameters of CNN with number of epochs. We can observe that initially both validation that

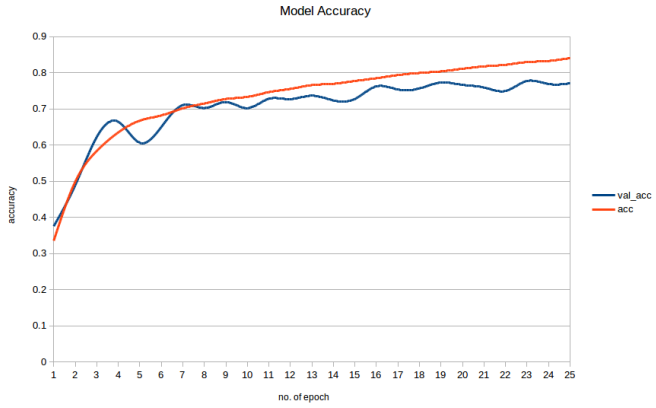

Fig. 5. Model Accuracy for CNN

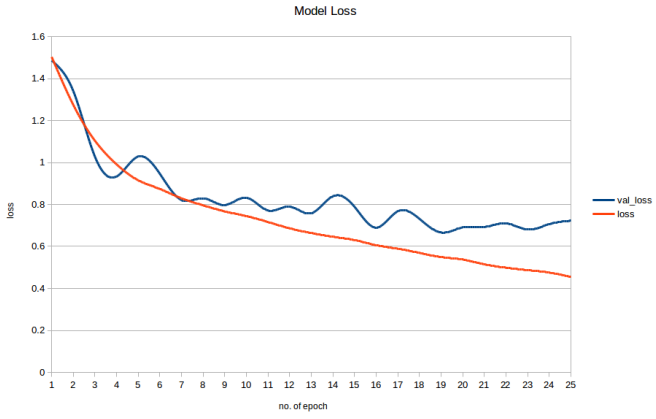

Fig. 6. Model Loss for CNN

is test accuracy and train accuracy is around 0.35 which is very low. But the model got better and better with each epoch and finally at around 15 epochs the validation accuracy got stabilized but training accuracy was getting improved so we can stop there as beyond that point the model may overfit. Same observation can be drawn from the model loss that around 15 epochs the validation loss got stabilized but training loss continued decreasing. So we can fix around 20 epoch as a good number to stop training the model with around 0.77 validation accuracy.

Finally we plotted the confusion matrix to observe the pattern of which ethnic group the model is not sure about and for which ethnic group it is most confident. We can observe from

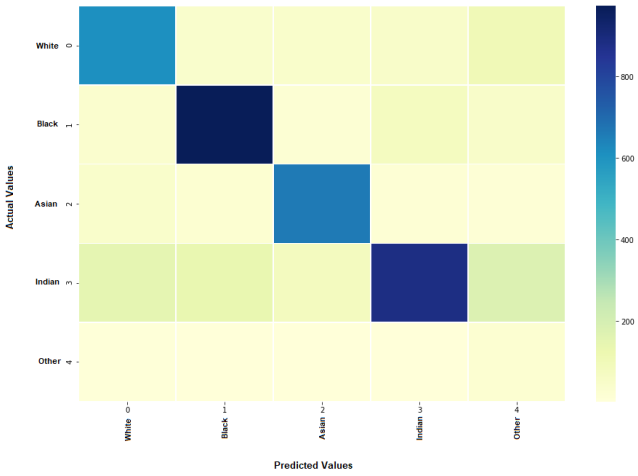

Fig. 7. Confusion Matrix for CNN

the confusion matrix that model is highly confident for White, Black and Asian. But for Indians the model was giving result from different ethnic groups. The Other is not that prominent also but that is due to less number of images being from that group. We tried similar kind of observation with SVM also and got the following result. In this also Other group is not

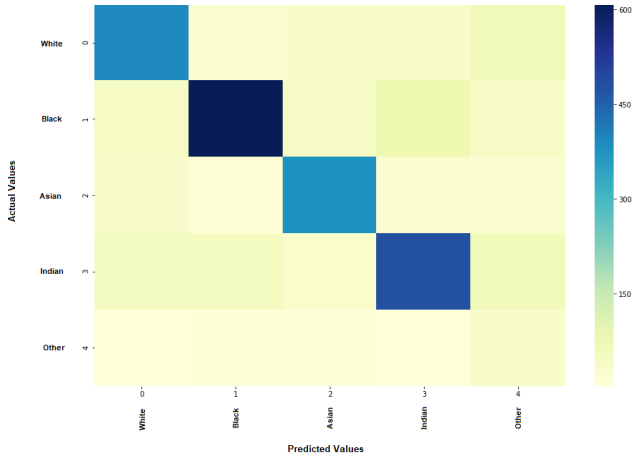

Fig. 8. Confusion Matrix for SVM

that prominent due to less number of images. But rest of the misclassification is distributed over all ethnic groups. The summary of the results is given in Table 3.

TABLE III
SUMMARY OF RESULTS

Models	Accuracy
k-NN	0.47
SVM - (without PCA)	0.65
SVM - (with PCA)	0.73
CNN - (grayscale)	0.76
CNN - (coloured images)	0.77

VII. PREDICTION

For prediction of any image using the model built using CNN, the following code was used in which any image can be given as the imput and the ethnic group label of that person was given as the output of the code.Image used for above code is given in Figure 10 which was not part of the dataset.

```
import os
os.environ['KERAS_BACKEND'] = 'theano'
import keras
import cv2
import numpy as np
from keras.models import load_model
model = load_model('ML/saved_models/complete_data_trained.h5')
Xdata = []
```

```
image = cv2.imread('ML/shin.jpg')
image = cv2.resize(image, (100,100))
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

Xdata.append(image)
```

```
array = ["white","Black","Asian","Indian","Others"]
print(image.shape)
arr = np.reshape(image,(1,100,100,3))
pd = model.predict(arr)
```

```
(100, 100, 3)
```

```
array[np.asscalar(pd.argmax(axis=1))]
```

```
'Asian'
```

Fig. 9. Prediction Code

Fig. 10. Input Image

VIII. CONCLUSION

We started out with modelling the problem of "Ethnicity Identification" as a multi-class classfication task. We built various models to effectively detect the ethnicity of a person just by their face. Out of all the models, deep learning seemed most promising. The highest accuracy was acheieved by a deep learning ConvNet. Our model was able to predict correctly 77 images out of 100 on unseen images. This is quite satisfactory considering the dataset was not big enough for a deep learning task. A very large and well curated dataset could enable us to acheive further accuracy.

The results in the end convinced us that face does reveal a lot about a person's origin. Also the fact that there was not much difference in the accuracy of the model on grayscaled images and color images highlights the observation that "Skin Color" plays an insignificant role when it comes to broad classification. Its importance in sifting ethnicity at finer granularity is an interesting topic of research, and can be carried out in future.

APPENDIX

A. KNN

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
y_pred_knn = knn.predict(X_test)
knn_score = accuracy_score(y_test, y_pred_knn)
knn_score
```

```
0.4746222394420767
```

B. SVM

```
#Applying RBF kernel SVM
svc = svm.SVC()
svc.fit(x_train_pca, y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

```
y_pred_svm = svc.predict(x_test_pca)
svc_score = accuracy_score(y_test, y_pred_svm)
svc_score
```

```
0.7342115459124371
```

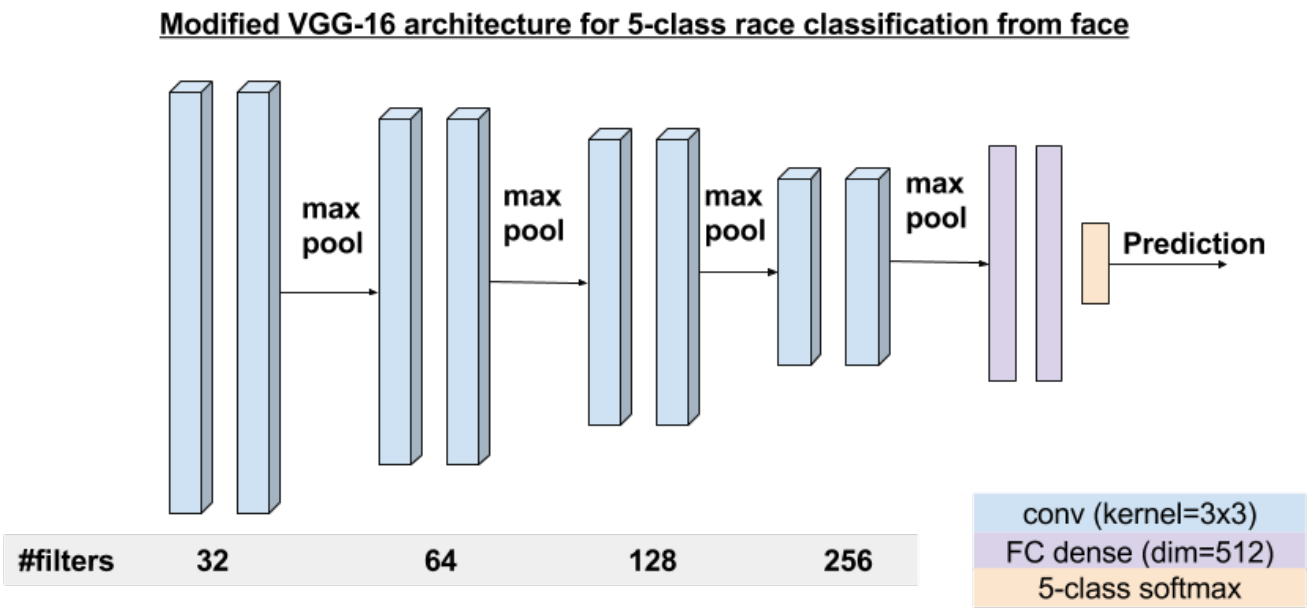

Fig. 11. VGG16 architecture for Ethnicity Identification

C. CNN

```
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
x_train = X_train
x_test = X_test
# Let's train the model using RMSprop
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=25,
              validation_data=(x_test, y_test),
              shuffle=True)

if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
    model_path = os.path.join(save_dir, model_name)
    model.save(model_path)
    print('Saved trained model at %s ' % model_path)
```