**Are these two people related ?**

**Venkata Pratyush Kodavanti [G01225485]**

**Abul Shariff [G00908050]**

## **File Structure:**

**Preprocessing\_Project.ipynb** : Contains the combimbinations of all the images, As we are storing the data in the drive we are accessing the folders from there and retrieving the path for each image and thereby creating the dataframes in this file.

**7\_Layers\_model.ipynb:** File contains the implementation of the basic CNN model.

**VGG from Scratch.ipynb:** File contains implementation of the VGG CNN discussed further in the report.

**Improvement\_Project.ipynb:** This file contains the implementation of the VGG on top of RESNET 50

## **Abstract:**

In this project we explore different neural network models to explore their learning ability to detect blood relations based on profile photos of different people. We have a dataset containing 1000 families which have multiple members and each member is provided with multiple images. The dataset also contains relationships stored in .csv files to help us determine which relations are blood related. We begin our experiments using a small 7 layer neural network on a small dataset of 500 images. We then use a self built VGG-CNN architecture to process a larger number of images. Following which we create pooled images to pass through a sequence of dense layers. And then finally we use a built in Reset-50 architecture to predict if its actually possible to predict blood relations based on profile images. For testing purposes we focus on Accuracy and AUC metrics provided by the tensorflow keras api and we use 250000 images for testing on our final models.

## **Introduction:**



DNA test is one of the sure shot methods for telling if two people are related by blood, but it is time consuming, expensive and sometimes the DNA samples are not readily available. Furthermore, not everyone is privileged to get a DNA test done. We humans can identify similarities between two people just by looking at them, and can tell if two people share similar facial features. As I can identify the similar Jaw structure in the above image. Can we develop a program which can look at the facial images of two individuals, identify any similarities between them and then decide if they are related by blood.

## **Literature Review:**

Turns out we can develop a program to do so, the pilot project(**Automatic Kinship Classifier**) [1] was done by NorthEastern University of Boston way back in 2011, They mainly focused on transfer learning. They would apply face points detection on all pairs of images. Once these points are found out they would align the faces according to the affine transformation, but this paper lacked in 2 particular ways

1. The dataset was not large enough
2. The accuracy was very low.

Multiple papers were published on this dataset and this area of research and few of the most recent are

**Families in the Wild [2]** and **Recognising families in the wild [3]** used SOTA techniques such as VGG Facenet on top of Resnet50 and then fine tuned models to get a good accuracy.

**VGG FaceNet:** is a pre-trained CNN with the topology of VGG-16: made-up of small convolutional kernels (i.e., 3 3) with a convolutional stride of 1 pixel. VGG-Face is trained on 2:6M face images of 2,622 different celebrities. By using VGG Face CNN the authors of the paper [5] got an accuracy of 73%.

**RESNET 50 (Residual Network):** Resnet 50 is a 50 layer residual network which is pre-trained on millions of Images to classify 1000 different sets of images. The network has an input size of 224 x 224. By using the RESNET along with CF the authors of the paper[5] got an accuracy of 86% for kinship verification.

Further the authors of these papers also tried to identify the type of the relation between two related people, which is beyond the scope of this project. Finally in 2019 a Kaggle challenge was posted by NorthEastern University SMILE LAB [4] which is where we got our dataset.

## **Dataset Description & Preprocessing:**

The dataset contains the folders of Families, which inturn contains the folders of individual members. Each member folder contains the raw facial images. In total there are 1000 families and on an average each family has 4 to 5 members. Each member has an average of 9 to 10 images in the dataset. Further, we are given a CSV file which contains all the pairs of images(image address) that have a kinship between them.

The training folder contains 1 folder(families) and a file (train\_relationships). The Families folder of a

family(​F0123​) contains ​the individuals (​MIDx​) in that family. Images in the same ​MIDx​ folder belong to the same person. Individuals in the same Family folder belong to the same family.

The train\_relationships.csv file contains the training labels of only the related people in a family.

As we are given the pairs of people who are related, we need to take this and make all the combinations of people who are related. Once we get all the combinations, we use the glob function to get all the images of a particular individual. From this we get all the possible combinations of each image of each user with all other images. For a 1000 families with an average of 4 members and an average of 8 images we got **422871** photo combinations.

After generating the possible combinations, if the individuals of the combination are a pair in the train relationship csv file we mark that combination as 1(related) else we mark it as 0(not related). This functionality is implemented in the **Preprocessing\_Project.ipynb file**

Working with **422871** pairs of images was too heavy for our processor and RAM so we sampled only 250000 pairs of image combinations and then split the training data into Training, Validation and Testing (70, 10, 20 percent respectively). To maintain a homogeneous ratio we used **Stratified sampling.**

## **Methods and Techniques**

**7 Layer Model:**

We initially created a **7 Layer Model**. The images are first read and then converted into grayscale(224 x 224.). We then reshaped the images to be of (224, 224,1) dimension. We cannot read multiple images in this model at a time, we then take the difference of the image arrays and then send it to the model as an input.

In this model, we input our image through 3 Convolutional layers and use a max pooling function to get the max features in the images. Following the Convolutional layers we use 3 Dense layers to take flattened features and output a decision of 1 - related and 0 - not related. We constructed this model for understanding the complexity of images. The model was not predicting properly as we are only sending the images array difference and the model was not properly understanding on which pair is a relation and which pair is not.

**VGG CNN:**

To improve this model, and overcome the above mentioned limitations, we then built a VGG -CNN model, as a part of preprocessing the images we are sending the individual image for each pair into the VGG model **preprocessor** first and then we send in the array difference between the to the model again as an input to the **classifier**. We realized that our images were extremely large ( 224 x 224 ) to process and it was taking a long time. In order to reduce the size we decided to separate out our VGG model. We used 16 convolutional layers to reduce 224 x 224 images to a size of 2000 features and then take subtractions of these images then input them into a series of Dense layers. Preprocessor refers to the model of reducing the size of the image and then classifier refers to the model which includes all the dense layers that helps in classifying the relation. By doing so we did observe a significant increase in the accuracy of the prediction from the 7 layer model(**“Visual Kinship Recognition of Families in the Wild.”**)[5].

**VGG on top of the RESNET 50 model:**

To further improve the above mentioned model we used a pre-built model for our project to see if relationships can be predicted via profile images. For this we used pre-built **Resnet-50** from tensorflow/keras api. Further, this model can take input of multiple images so we first trained the model in such a way that the Resnet model initially trains on each individual image for each pair and then on top of that we train the model with the difference of the images. Thereby giving more clarity to the model on which features are similar and which features are not similar. A major point for using this model was to compare how we can improve our model building capabilities and skills in the future. The Resnet-50 model took a lot of time to train but it only took less than 5 epochs to reach higher accuracy. This model achieved a better performance when compared with model described in the paper (**“Visual Kinship Recognition of Families in the Wild.”**)[5]

We implemented this code for comparison purposes and our learning purposes to test whether it's possible to actually find kinship amongst the dataset that we had. [8]

**Evaluation Metrics**

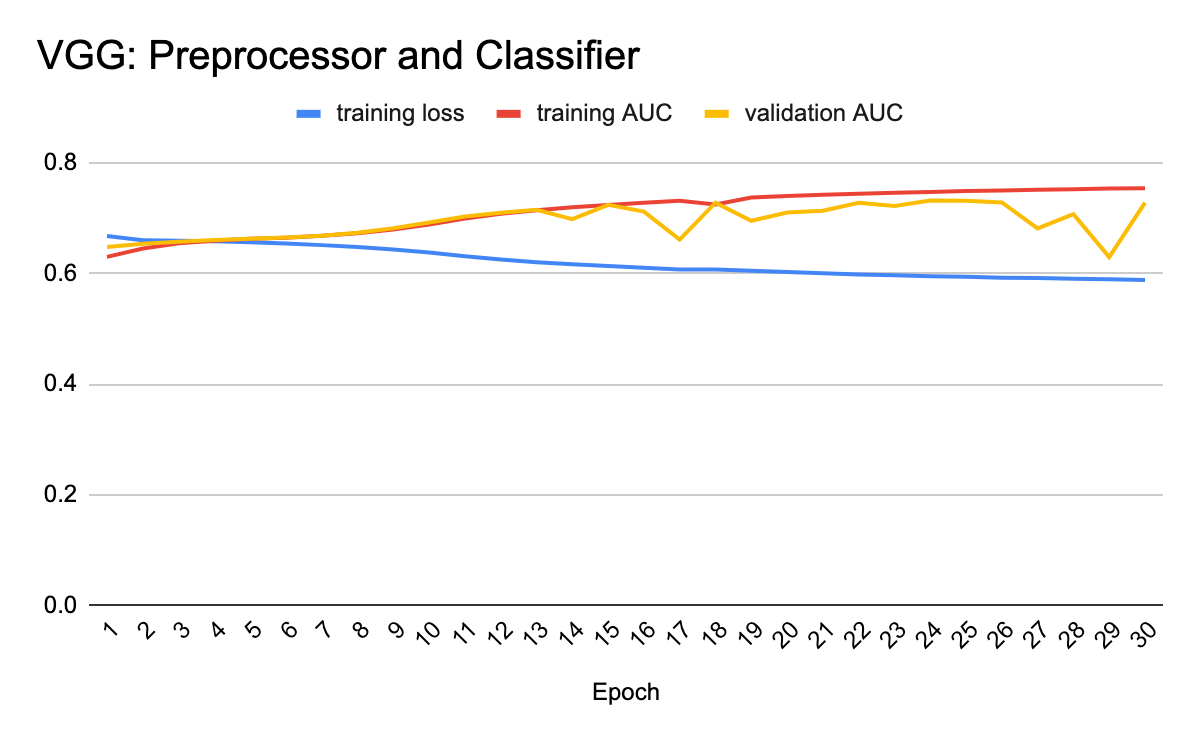
In this project we were using 2 metrics **Area Under the Curve** and **Accuracy**. We were performing binary classification. We didn’t want to limit ourselves to an accuracy metric which predicts which predictions were correct and incorrect but wanted to have a confidence level to see if the prediction was satisfactory. Hence we used AUC. Receiver Operating Characteristic (ROC) curve is the projection of TPR v FPR (True Positive rate v False positive rate) on a 2D plane. Based on each record and its classification we can calculate the TPR and FPR and project a graph. The area under the graph is referred to as AUC which provides a percentage of a classifier’s confidence based on their classification rates. AUC goes between 0.5 to 1. 0.5 reflects a random guessing and no learning of data 1 represents a good classifier.

**Tools Used**

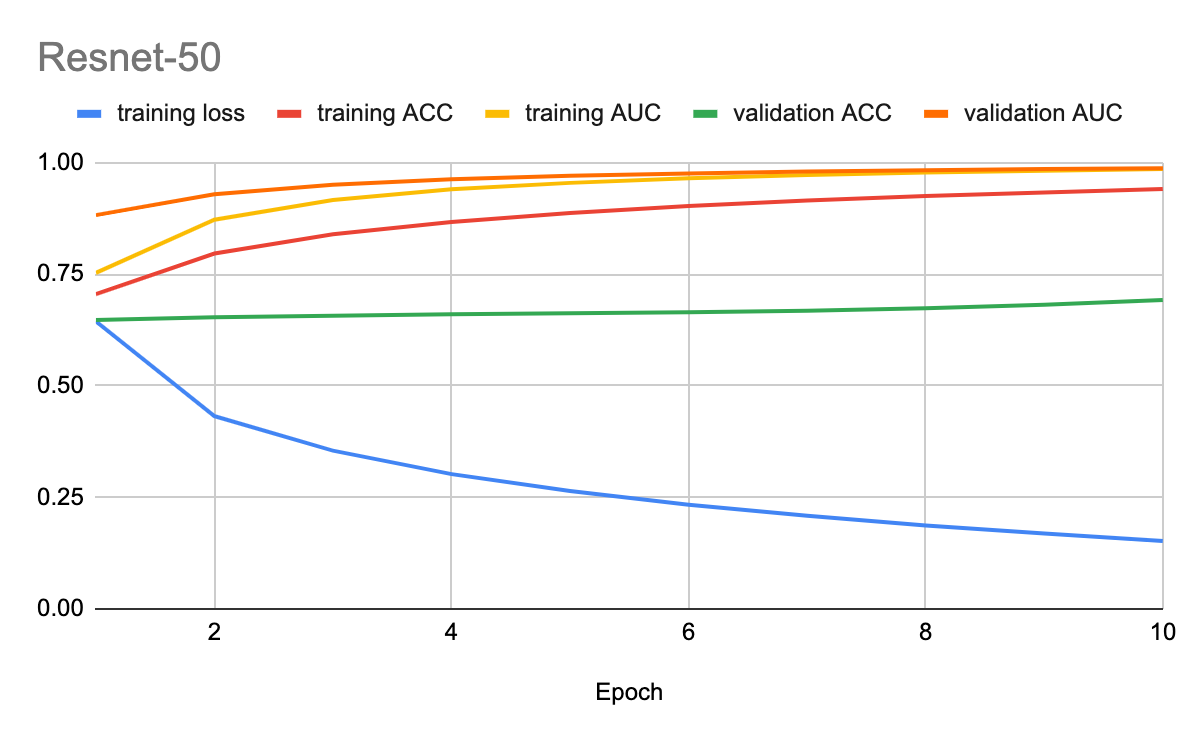
We used tensorflow/keras for model creation, numpy arrays for image representation, scikit learn tool kit for calculating AUC and Accuracy and pandas dataframe to store tuples of image pairs for kinship.

## **Discussion and Results:**

For the **7 layer model** we were converting all the images to a numpy arrays and then storing the difference of the numpy arrays in a single array, it consumed a lot of time and processing power which when trained on large number of images (> 2500000) images caused the system to break. For the 7 layer model, we only worked with 100000 images. The main limitation for this model is that as we are only sending the difference of the images so the model did not properly understand what are the similarity features for deciding whether there is a kinship or not. The model always gave an output of all ones (All the pairs are related). The other reason for failure of this model is that it was not deep enough, we have used 7 layers of CNN which when compared to current SOTA methods is very shallow and not suitable for kinship classification.



For the **VGG CNN layer model**, even though we are converting the images to numpy arrays we are first sending them to VGG preprocessor and use the difference between the numpy arrays to classify if as a kinship relation or not. We trained this model for 30 Epoch on a training dataset of 175000 pairs of images and validation dataset of 25000 pairs of images and then tested on 50000 pairs of images. This model gave us an accuracy of 66.5% and an Area under Curve of 66.34%, there is a significant improvement in the accuracy and AUC score when compared with the basic convolution layer but this improvement is not on par with the SOTA techniques. The limitations being that the layers were not deep enough, the current SOTA Resnet 50 uses 50 layers of residual network whereas our model is only 16 layers. Further, we preprocessing the images separately and then sending the difference of the images to train the model, the model is still unable to understand on what are the similarity features for deciding whether there is a kinship or not



For the VGG Resnet 50 model, we first train the model on individual images on the Resnet 50 layer and then on top of this we train the VGG layer with the difference of the images. This procedure gave the model 2 advantages

1. We used a pre built model which classified images properly
2. The model was separately trained on the images individually and then it is getting trained on the images difference(using tensor) thereby giving the model a clear understanding of what features are required to determine kinship.

The model achieved an accuracy of 94.62% and the AUC score of 98.73 for a training set of 1750000 pairs of images, a validation set of 25000 pairs of images and a test set of pairs of images.

The results for the 3 models are tabulated below:

|  |  |  |
| --- | --- | --- |
| Model Type | Area Under Curve | Accuracy |
| 7 Layer Model | 50% | 50% |
| Processor and Classifier | 66.34% | 66.5% |
| Resnet-50 | 98.73% | 94.62% |

From the above results we concluded we needed more convolutional layers to break down images to get more information. Initially the model was very conservative in max pooling different features in an image since most of the features are not being used or getting missed out.

The results didn’t get any better. We tried adding a lot more Dropout functions, reducing / increasing the dataset but the validation score remained stable. Down the line we began using a larger dataset compared to what we had in the previous models. Furthermore we tried looking at adding more layers and removing layers at this point but having 13 convolutional layers and 3 dense layers gave a stable result

With a change in our pipeline, in VGG CNN, we had better and stable results from our model. The time to preprocess the image still took a lot of time but it was much easier for the classification to finish. In this case we had mostly tested with 250000 pairs of images to get the above accuracy. In many cases our validation score was lower because of the amount of data. The model worked better as we added more data to it.

## **Conclusion:**

In this project we had a huge learning curve of understanding how perceptrons work and how they are used in different python libraries. Through research and examples we created multiple techniques to make better decision models. In our experiments we had to skip over some other techniques due time constraints. In comparison to our implementation the open-sourced pre-trained models were a lot better. The pre-trained model we used was already trained on millions of images and in comparison a more complex architecture. With available tools such as Resnet-50 it is in fact possible to find blood relationships via their profile images. Just as humans detect similarities between children and their parentings machines can be taught to learn different facial features to come to a conclusion whether 2 people are related.

## **Future Works:**

On the technical end we can alter the architecture to input images and subtract their differences from their neural parameters rather than their features[8]. In our implementation we can extend our project to include images that weren’t in our training dataset (profile images which include us and our parents). In families are training sets mostly included images of closed related families but we can extend this implementation to include grandparents which means neural networks can detect the best features to contrive how ancestors and successors would look like. Using a Generative Adversarial Network implementation we can add such models to find how missing people would look like in the present or in the future.

## **References:**

[1] ] S. Xia, M. Shao, and Y. Fu, “Kinship verification through transfer learning,”

[2] J. P. Robinson, M. Shao, Y. Wu, and Y. Fu, “Families in the Wild (FIW): Large-Scale Kinship Image Database and Benchmarks.”

[3] J. P. Robinson, M. Shao, H. Zhao, Y. Wu, T. Gillis, and Y. Fu, “Recognizing Families In the Wild

[4] https://www.kaggle.com/c/recognizing-faces-in-the-wild/

[5] J. P. Robinson, Y. Wu, T. Gillis, and Y. Fu, “Visual Kinship Recognition of Families in the Wild.”

[6] <https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/>

[7] <https://www.kaggle.com/shivamsarawagi/wildimagedetection-0-875/comments>

[8]<https://github.com/dennymarcels/AreTheseTwoPeopleRelated/blob/master/Notebook.ipynb>