

**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CZ4041 - MACHINE LEARNING

PROJECT REPORT

Northeastern Smile Lab - Recognizing Faces in the Wild

STUDENT NAME	MATRICULATION NUMBER
Tharakan Rohan Roy	U1822028E
Atluri Sai Mona	U1722267D
Pereddy Vijai Krishna Reddy	U1722037E
Jaswanth Pinnepu	U1822536F

Table of Contents

1. Introduction:	3
1.1 Problem statement:	3
1.2 Roles of individual team members:	4
2. Kaggle evaluation score and rank:	4
3. Proposed solution:	6
3.1 Initial research:	6
3.2 Preprocessing data:	6
3.3 Choosing the baseline model:	10
3.4 Feature extraction:	10
3.5 Methodologies:	11
3.6 Prediction and final output:	11
4. The rationale for the chosen solution:	12
4.1 10 Fold Cross-Validation:	13
4.2 Use of VGGFace pre-trained model:	13
4.3 Reduced Overfitting using EarlyStopping and Dropout:	14
4.4 Average ensembling	16
4.5 Choice of loss function and optimiser	18
5. Challenges faced:	19
5.1 Limited GPU time	19
5.2 Optimizing the training process	17
5.3 Scarcity of different study resources	20
6. Conclusion:	20
7. References:	21

1. Introduction:

Synergetic Media learning Lab (SMILE lab) of Northeastern University focuses on research regarding the application of various concepts like machine learning, Human-computer interactions, social media analytics along with high-level video and image understanding. Their research involves making use of multimedia available on the internet to design a learning system that can collect visual information from the environment, process it, and perform precise analysis as well as provide possible suggestions.

On August 1, 2019 SMILE lab organized a Kaggle competition “Recognizing faces in the wild” SMILE lab has been working on a kinship classifier since 2010, However, due to a limited database and the effect of miscellaneous factors on facial relationships, this software cannot be made practical. This competition aims to improve the kinship classifier designed by the smile lab.

1.1 Problem statement:

This competition requires the participant to solely make use of facial images to predict whether two people are blood-related or not by designing a complex model. The data used for training and testing is provided by an image database called families in the wild.

The files used for this project are:

Train.zip: This zip file is used to train the model. It consists of folders for individual families. Each family folder has separate folders for each individual of the family. This separate folder contains one or more images of that individual. The family names are of the format ‘FXXXX’, where XXX stands for the family number. Each member of the family is stored in a folder with the name of the format ‘MIDX’ where X stands for the member number.

Test.zip: This zip file contains a folder which contains the images of the faces of unknown individuals and is used for testing. Each of these images are named in the format ‘faceXXXXX’, where XXXXX stands for the image number.

Train_relationships.csv: This .csv file contains columns which denote the relations between 2 pairs of members of the family. This information is used to train the model.

Sample_submission.csv: This .csv file contains two columns, which help the participants determine what to perform the predictions on, so as to submit the file output file for the submission. The first column consists of image-image pairs from the test folder. The second column by default has a value of 0 for all the rows and is to be filled by the participant.

The aim of the competition is to predict the probability of the chances of the given pair of persons represented by the images being blood related. This is done by filling the second column in the sample submission file with a value between 0 and 1.

1.2 Roles of individual team members:

The team believed that an important part of the project is for each and every team member to share their knowledge with each other and step out of their comfort zone and learn something new. Hence every team member equally and effectively contributed at every stage of the project.

The additional concepts that are required to be learned are broken down into different parts and each team member learned their own part and explained it to the rest of the team for effective learning. This was done either virtually through video calls or in person.

Various strategies to improve the performance of the model were discussed and later designed and various modified versions of the code were run by all the members of the team parallelly to figure out the best possible way to solve the problem.

The report and the presentation video were completed by all the team members working together and bringing forth their knowledge. This was done with the help of online tools which helps editing simultaneously possible.

2. Kaggle evaluation score and rank:

We submitted multiple output files on kaggle and the following figure shows the Kaggle evaluation of score and rank of the best output after countless attempts of data processing and feature extraction as well as model parameters tuning.

Best submission: (after 10 cross validation and ensembling 20 different models)

Submission and Description	Private Score	Public Score	Use for Final Score
CSVFinal1.csv 2 hours ago by Mona Chowdary add submission details	0.908	0.909	<input type="checkbox"/>

Public Leaderboard					
Private Leaderboard					
This leaderboard is calculated with approximately 50% of the test data. The final results will be based on the other 50%, so the final standings may be different.					
				Raw Data	Refresh
16	Andreisun		0.911	134	9mo
17	BM_AI		0.911	84	9mo
18	dzw_		0.908	195	9mo
19	jazz7313		0.907	72	9mo
20	clds		0.907	106	9mo

Fig 1.1

It can be seen from Fig 1.1, on the public leaderboard that we stand at 18th place out of 528 teams which puts us in the top 4%
(Score: 0.909, Rank 18/528, **Top 4%**)

Public Leaderboard							
Private Leaderboard							
The private leaderboard is calculated with approximately 50% of the test data. This competition has completed. This leaderboard reflects the final standings.							
#	△pub	Team Name	Notebook	Team Members	Score ?	Entries	Last
21	▲16	YamPeleg			0.910	52	9mo
22	▲3	AKA不花现金			0.909	60	9mo
23	▲1	huiqin			0.909	21	9mo
24	▼5	jazz7313			0.908	72	9mo
25	▼5	clds			0.908	106	9mo

Fig 1.2

It can be seen from the figure 1.2, on the private leaderboard that we stand at 24th place out of 528 teams which puts us in top 5%.
(Score: 0.908, Rank 24/528, **Top 5%**)

3. Proposed solution:

3.1 Initial research:

The team went through a list of existing solutions available on Kaggle, Github, discussion forums, and other available resources online for the given problem and other similar problems. The solutions provided by the top performers were analyzed to figure out strategies that could help us achieve a good score. Each of the given methods and concepts were researched on and discussed.

3.2 Preprocessing data:

The following steps involved in the data preprocessing are listed below:

- I. **Getting the dataset:** The data provided by Families in the Wild (FIW), the largest and most comprehensive image database for automatic kinship recognition. We did the preprocessing, training and testing for the data and the model was done on the notebooks available on kaggle as it had 30 hours of GPU available per person. Since we used these notebooks, the output files were of the form .ipynb and we had the dataset preloaded. We also had to create and load datasets created for the ensembling.

Given below in figure 1.3, is the file structure for our notebooks:

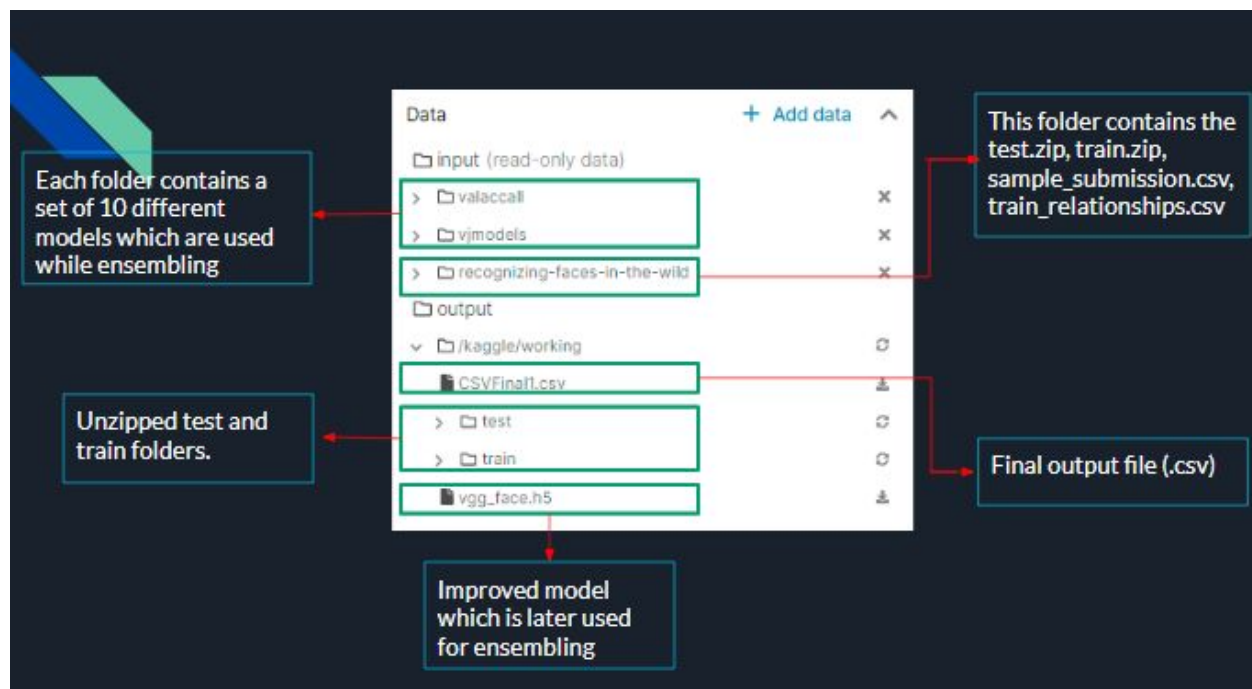


Fig 1.3

II. Importing libraries: The imported libraries that were imported for our model have been shown below in the given figure 1.4. We used tensorflow as our framework to run the machine learning model. We heavily relied on the built in functions from the keras library which is best suited to run on top of tensorflow. Python was the programming language we used to write all our code in.

```
import os
import cv2
import h5py
import zipfile
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from collections import defaultdict
from glob import glob
from random import choice, sample
from tqdm import tqdm
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from keras.layers import Input, Dense, GlobalMaxPool2D, GlobalAvgPool2D, Concatenate, Multiply, Dropout, Subtract
from keras.models import Model
from keras.optimizers import Adam
from keras.preprocessing import image
```

Fig 1.4

```
!pip install git+https://github.com/rcmalli/keras-vggface.git
```

Fig 1.5

We have installed the *keras-vggface* from this github repository mentioned in the Fig 1.5 (<https://github.com/rcmalli/keras-vggface>) using the following line of code as mentioned in Fig 1.5 . After running this cell, we could import the *keras_vggface* by running the import statements given in the Fig 1.6. The reason for choosing this model has been explained later in the report.

```
from keras_vggface.vggface import VGGFace
from keras_vggface.utils import preprocess_input
```

Fig 1.6

III. Splitting dataset into training and validation set: In the model the dataset was split into Training set (*train_person_to_images_map* list) and Validation set (*val_person_to_images_map* list). The ratio of the splitting was about 9:1 for the training set to the validation set. The validation set was changed every run to reduce any chance of bias being introduced to the model while training.

```
val_famillies = "F09"
```

```

all_images = glob(train_folders_path + "*/*/*.jpg")
# print(all_images) # list all the images
train_images = [x for x in all_images] # list all the images
val_images = [x for x in all_images if val_families in x] # images that belong to val_families
# print(val_images)
train_person_to_images_map = defaultdict(list)

ppl = [x.split("/")[-3] + "/" + x.split("/")[-2] for x in all_images]
# print(ppl)
# print(len(ppl)) # 12379 pics in total

for x in train_images:
    train_person_to_images_map[x.split("/")[-3] + "/" + x.split("/")[-2]].append(x) # total number of ppl in the data set

# print(train_person_to_images_map) # segregates pics of each person in each family of train images
# print(len(train_person_to_images_map)) # 2316 ppl in total

val_person_to_images_map = defaultdict(list)

for x in val_images:
    val_person_to_images_map[x.split("/")[-3] + "/" + x.split("/")[-2]].append(x)

```

Fig 1.7

IV. Eliminating the false relationships: In the Fig 1.8, we can observe from the *train_relationships.csv* file, that there are a total of 3598 relationships but out of which only 3362 are the true relationships, i.e only 3362 relations can be used for the training the other relations have data missing in the dataset and hence are not of any use for training the model.

	A	B
1	p1	p2
2	F0002/MID1	F0002/MID3
3	F0002/MID2	F0002/MID3
4	F0005/MID1	F0005/MID2
5	F0005/MID3	F0005/MID2
6	F0009/MID1	F0009/MID4
7	F0009/MID1	F0009/MID3
8	F0009/MID1	F0009/MID2
9	F0009/MID1	F0009/MID6
10	F0009/MID2	F0009/MID4
11	F0009/MID2	F0009/MID6
12	F0009/MID2	F0009/MID3
13	F0009/MID3	F0009/MID4
14	F0009/MID3	F0009/MID6
15	F0009/MID4	F0009/MID6
16	F0009/MID5	F0009/MID1
17	F0009/MID5	F0009/MID2
18	F0009/MID5	F0009/MID4
19	F0009/MID5	F0009/MID6
20	F0009/MID5	F0009/MID3
21	F0009/MID7	F0009/MID3
22	F0009/MID7	F0009/MID6
23	F0009/MID7	F0009/MID4

Fig 1.8

So, we eliminated the false relationships by running the following code present mentioned in Fig 1.9. After which we separated the relationships into *train_relationships* and *val_relationships*. These lists were then later used for building the model.

The reason for the missing data is not mentioned by the competition makers but the complications arising due to it has been resolved in the preprocessing stage of the model.

```
relationships = pd.read_csv(train_file_path)
# print(relationships)
relationships = list(zip(relationships.p1.values, relationships.p2.values))
# print(relationships)
# print(len(relationships)) # 3598 realtions or rows in the csv file
relationships = [x for x in relationships if x[0] in ppl and x[1] in ppl] # to eliminate the false relations
# relationships = [x for x in relationships if x[0] in ppl and x[1] not in ppl]
# print(relationships)
# print(len(relationships)) # 3362 true relations

train = [x for x in relationships if val_families not in x[0]] # all the relations without the val_families
# print(train)
val = [x for x in relationships if val_families in x[0]] # relations only with the val_families
# print(val)
```

Fig 1.9

The below snapshot of the console is the list of false relationships in the *train_realtionships.csv* file. These are false relationships because the individuals in these relations do not exist. For example if we look at the relationship in the 1st index ('F0228/MID2', 'F0228/MID3'), it can be observed that MID3 (member) does not exist in the F0288 (family) from Fig 1.10. Similarly in the 2nd index ('F0238/MID1', 'F0238/MID8'), it can be observed that MID8 (member) does not exist in the F0238 (family) from Fig 1.11. Therefore all these false relationships have been eliminated.

```
('F0228/MID2', 'F0228/MID3'), ('F0238/MID1', 'F0238/MID8'), ('F0238/MID3', 'F0238/MID8'), ('F0238/MID3', 'F0238/MID7'), ('F0238/MID5', 'F0238/MID7'), ('F0238/MID5', 'F0238/MID8'), ('F0238/MID6', 'F0238/MID8'), ('F0238/MID6', 'F0238/MID7'), ('F0241/MID1', 'F0241/MID4'), ('F0244/MID1', 'F0244/MID7'), ('F0253/MID2', 'F0253/MID4'), ('F0253/MID3', 'F0253/MID4'), ('F0262/MID2', 'F0262/MID7'), ('F0262/MID3', 'F0262/MID7'), ('F0262/MID4', 'F0262/MID7'), ('F0262/MID5', 'F0262/MID7'), ('F0264/MID1', 'F0264/MID5'), ('F0264/MID10', 'F0264/MID5'), ('F0264/MID9', 'F0264/MID5'), ('F0275/MID2', 'F0275/MID3'), ('F0299/MID1', 'F0299/MID4'), ('F0299/MID2', 'F0299/MID4'), ('F0299/MID3', 'F0299/MID4'), ('F0309/MID2', 'F0309/MID4'), ('F0309/MID3', 'F0309/MID4'), ('F0309/MID6', 'F0309/MID4'), ('F0363/MID1', 'F0363/MID4'), ('F0363/MID2', 'F0363/MID4'), ('F0363/MID3', 'F0363/MID4'), ('F0363/MID7', 'F0363/MID4'), ('F0393/MID1', 'F0393/MID3'), ('F0393/MID1', 'F0393/MID6'), ('F0393/MID2', 'F0393/MID6'), ('F0393/MID4', 'F0393/MID6'),
```



Fig 1.10



Fig 1.11

3.3 Choosing the baseline model:

A baseline model Keras VGGFace is chosen. VGGFace is a highly reputed model based on convolution neural networks and max-pooling layers. It also supports Tensorflow backend. Further developments were made to the baseline model to achieve the best possible performance. We loaded the feature extraction layers with the following line of code as initiation.

```
base_model = VGGFace(model='resnet50', include_top=False)
```

Fig 1.12

3.4 Feature extraction:

Using feature extraction the dimensionality reduction is done by which the initial set of raw data is reduced to more manageable groups for processing. The characteristic of the large data sets is a large number of variables that require a lot of computing resources to process. Here, we selected and combined the variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

1) Convolution Features:

```
from keras.engine import Model
from keras.layers import Input
from keras_vggface.vggface import VGGFace

# Convolution Features
vgg_features = VGGFace(include_top=False, input_shape=(224, 224, 3), pooling='avg')
```

2) Specific Layer Features:

```
from keras.engine import Model
from keras.layers import Input
from keras_vggface.vggface import VGGFace

# Layer Features
layer_name = 'layer_name' # edit this line
vgg_model = VGGFace() # pooling: None, avg or max
out = vgg_model.get_layer(layer_name).output
vgg_model_new = Model(vgg_model.input, out)
```

3.5 Methodologies:

The model is trained by adding various additional layers on top of the existing layers provided by the baseline model Vggface. These additional layers are concatenate, multiply, subtract, dense and dropout. To further optimize the model, *ModelCheckpoint*, *EarlyStopping*, and *ReduceLROnPlateau* are implemented.

The checkpoint function is used to monitor the *val_acc* as the model moves through different epochs and updates the model only when the validation accuracy improves.

Early stopping function monitors the *loss* value and is used to stop the training if the training does not show any improvement in reducing the *loss* value. The patience for the early stopping function is chosen to be 50. That is the training will be discontinued if the model does not show any improvement for 50 epochs straight.

Reduce_on_plateau is used to monitor the *val_acc* as the model moves through different epochs and if the training shows no improvement of the *val_acc* for 20 epochs straight, the learning factor is reduced by 0.1.

```
curr_model = baseline_model() # initializing model with the given layers
# curr_model.load_weights(file_path)
curr_model_hist = curr_model.fit_generator(gen(train, train_person_to_images_map, batch_size=16), use_multiprocessing=True,
                                           validation_data=gen(val, val_person_to_images_map, batch_size=16), epochs=150, verbose=2,
                                           workers=4, callbacks=callbacks_list, steps_per_epoch=200, validation_steps=10)
```

Here, the *baseline_model()* is initialised and then we run the *fit_generator* function to train the model. We have selected the batch size as 16 and steps per epoch as 200.

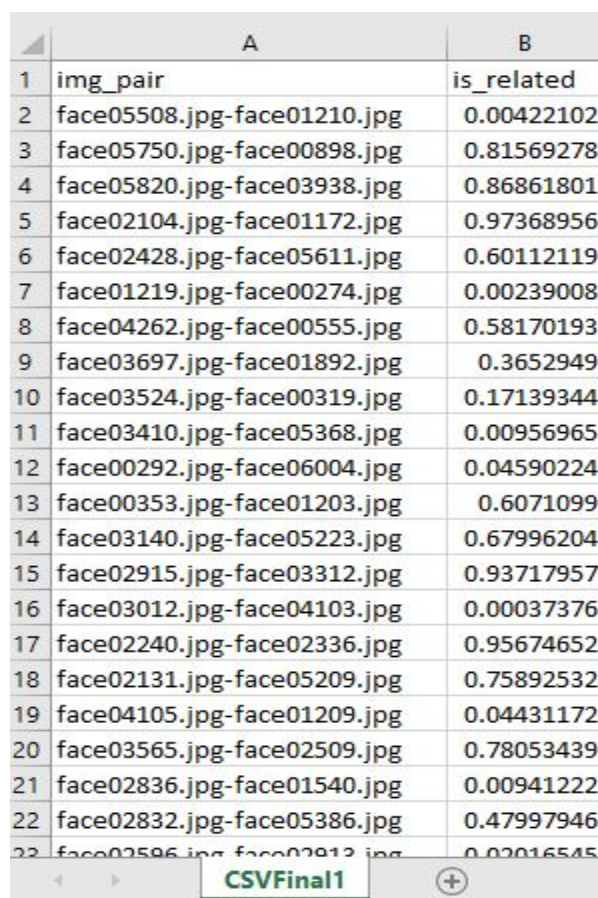
3.6 Prediction and final output:

```
prediction = 0
for i in ['00', '01', '02', '03', '04', '05', '06', '07', '08', '08_2', '09']:
    curr_model.load_weights('../input/valaccall/Val_acc_f'+ i + '.h5')
    prediction = prediction + curr_model.predict([X1, X2])
for i in ['00', '01', '02', '03', '04', '05', '06', '07', '08', '09']:
    curr_model.load_weights('../input/vjmodels/F'+ i + '_val_loss.h5')
    prediction = prediction + curr_model.predict([X1, X2])
prediction = prediction/10
pred = prediction.ravel().tolist()
predictions += pred

submission['is_related'] = predictions
submission.to_csv("CSVFinal1.csv", index=False)
```

Fig 1.13

For the prediction and the generation of the final output file, we used an average ensembling over 20 models. 20 models can be split into two groups of ten models for which each of the different models corresponds to a different validation set. This was done to implement the 10 fold cross validation. In reference to Fig 1.13, the first loop was to ensemble the models which were trained keeping the Validation accuracy as the determining metric to stop the training. While the second loop shown in the Fig 1.13 is used to ensemble the 10 models which were generated keeping the validation loss as the metric to stop the training. This point is further elaborated in the later sections of the report.



	A	B
1	img_pair	is_related
2	face05508.jpg-face01210.jpg	0.00422102
3	face05750.jpg-face00898.jpg	0.81569278
4	face05820.jpg-face03938.jpg	0.86861801
5	face02104.jpg-face01172.jpg	0.97368956
6	face02428.jpg-face05611.jpg	0.60112119
7	face01219.jpg-face00274.jpg	0.00239008
8	face04262.jpg-face00555.jpg	0.58170193
9	face03697.jpg-face01892.jpg	0.3652949
10	face03524.jpg-face00319.jpg	0.17139344
11	face03410.jpg-face05368.jpg	0.00956965
12	face00292.jpg-face06004.jpg	0.04590224
13	face00353.jpg-face01203.jpg	0.6071099
14	face03140.jpg-face05223.jpg	0.67996204
15	face02915.jpg-face03312.jpg	0.93717957
16	face03012.jpg-face04103.jpg	0.00037376
17	face02240.jpg-face02336.jpg	0.95674652
18	face02131.jpg-face05209.jpg	0.75892532
19	face04105.jpg-face01209.jpg	0.04431172
20	face03565.jpg-face02509.jpg	0.78053439
21	face02836.jpg-face01540.jpg	0.00941222
22	face02832.jpg-face05386.jpg	0.47997946
23	face02586.jpg-face02812.jpg	0.02016545

Fig 1.14

From Fig 1.14, the second column shows the probability of the persons represented by the image pairs being blood related. This is the output file which will be submitted to kaggle to get a test score for the model generated.

4. The rationale for the chosen solution:

Some Machine learning techniques were adopted to improve the overall performance of the model. Some of these techniques were taught in the lectures and some were discovered online

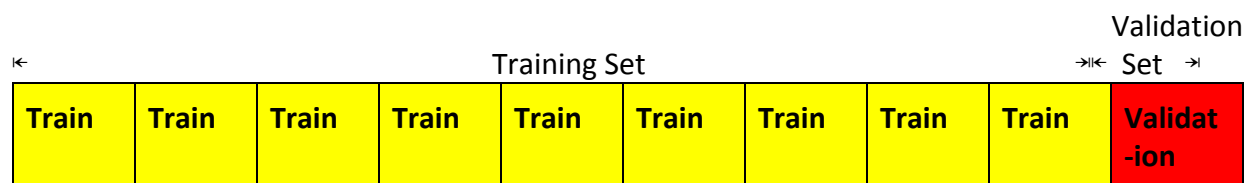
from other courses. Experimentations were done on these practices and the best parameters were used for the final run of the model. These techniques were mainly implemented to increase the accuracy on the Test set. Some of the techniques used are:

4.1 10 Fold Cross-Validation:

Since there was no clear validation set provided to the users to test the model on and determine whether the model has overfitting and if so, by what amount is the overfitting taking place and hence the decision to divide the training set into 10 equal parts was taken. The model was run 10 times keeping 9 parts of the training set for training and the remaining one part as a validation set. The validation set was changed every run to a different part from the 10 different parts.

The decision to divide the total training set into 10 parts was taken as there was not a lot of data for the training provided and hence using a different validation set per run would result in a small improvement for the overall model.

The `model.load_weights()` from the `keras.models` was used to help reduce the training time per run as the weights and biases were stored in the .h5 file and the model didn't need to be trained from randomised weights and biases, instead had weights that were obtained from the previous run.



4.2 Use of VGGFace pre-trained model:

The main reasons for using the VGGFace model were the limited training data provided for the competition and the lack of GPU resources to train a model from scratch.

The pretrained model VGGFace from `keras_vggface` library as it is based on a well-reputed model architecture (VGG16 and Res50) which is based on a numerous number of Convolution Neural Networks and MaxPooling layers. This architecture also preserves all the details in the images in spite of the convolutions as it uses the same padding. This is important as different parts of the face in the image is important for identifying kinship. Due to the GPU constraints we decided to use this model instead of training a model with convolutional layers.

The VGGFace model also has been trained over 1 million images and hence applying transfer learning to this model would provide the best results instead of training the model from scratch. Another added advantage of using the VGGFace was that the model was trained on

inputs of the size 224 x 224, the dataset provided for this competition also had images of the same size. This reduced the need to resize the image for the model.

The Res50 model from the VGGFace was chosen as a deep neural network was required to perform the identification and detection and hence without using a residual network, the accuracy of the model would eventually decrease. This model was symmetrically applied to both the images being passed to the model at once to train the relations.

4.3 Reduced Overfitting using EarlyStopping and Dropout:

Since a separate Validation set or a Test set was not provided to check the accuracy throughout training so as to determine when to reduce overfitting, a ten fold cross validation was implemented. To reduce the overfitting, some other techniques were also applied, this included EarlyStopping which is a callback function from the keras.callbacks library. A patience value of 50 epochs were used before stopping the training.

To determine which value to monitor for the EarlyStopping, we experimented by using the Validation Accuracy and the Validation Loss. Later all the 20 models were ensembled to generate the final output file. The trend of the accuracy and the loss before the EarlyStopping for the train and the validation set is shown below in Fig 1.15 and .

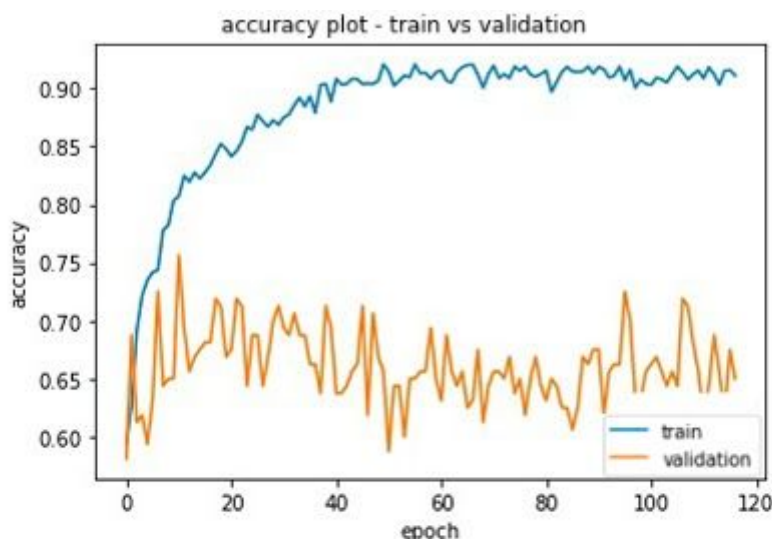


Fig 1.15

As seen from the above Fig 1.15, there is a large difference between the Validation accuracy and the Train accuracy. The Validation accuracy gradually decreases as the number of epochs are increased and while the training accuracy increases and is not reduced further, this suggests overfitting. We can also observe from Fig 1.16, the validation loss increases as the number of epochs increase while the training loss keeps decreasing and this suggests overfitting as well.

In order to find the best metric, we experimented with both the validation loss as well the validation accuracy for the EarlyStopping callback function.

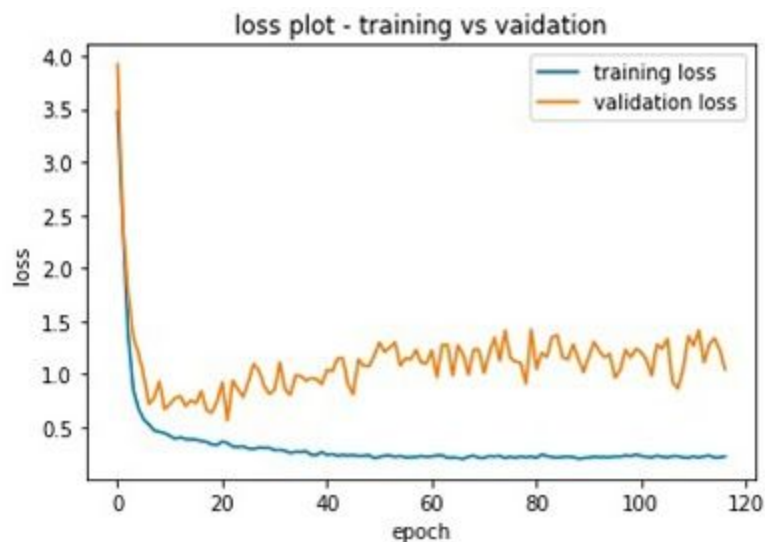


Fig 1.16

After implementing the EarlyStopping Function, the following graph for the accuracy and loss for the Validation and Train set is obtained. From Fig 1.17 and Fig 1.18, we can observe that the number of epochs was reduced from 120 to 60. We can observe that once there is no improvement in the validation accuracy or the validation loss, the training stops.

To reduce further overfitting, a dropout with rate = 0.01 was added in the final layer. We can see that the value of the deviation of the validation accuracy and validation loss from the training accuracy and training loss respectively has been reduced.

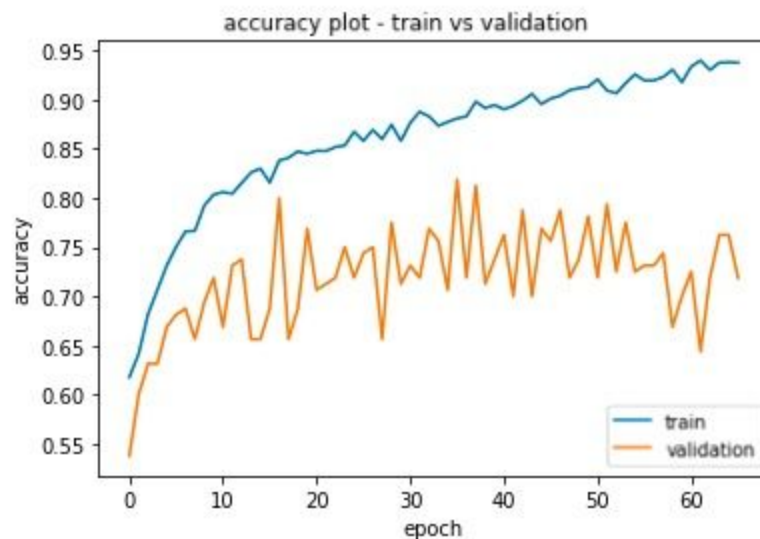


Fig 1.17

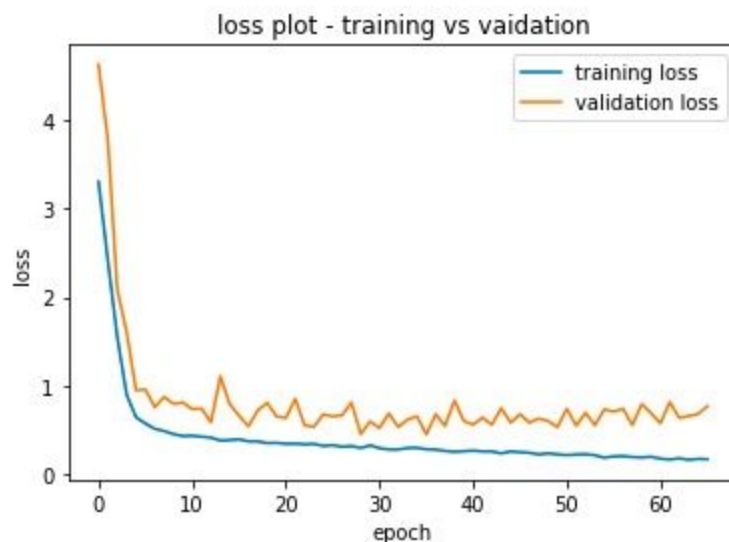


Fig 1.18

4.4 Average Ensembling:

After generating 10 layers using 10 fold cross validation using Validation accuracy as the metric for early stopping, the 10 different output files were tested on the test set and each of the models. Each of these files gave a score above 0.865 on the public leaderboard and when these models were assembled using average ensembling, the resultant output file gave a score of 0.902 on the public leaderboard.

Submission and Description	Private Score	Public Score	Use for Final Score
CSVF00.csv a minute ago by Rohan Roy add submission details	0.872	0.868	<input type="checkbox"/>
CSVF05.csv 2 days ago by Rohan Roy add submission details	0.879	0.878	<input type="checkbox"/>
CSVF04.csv 2 days ago by Rohan Roy add submission details	0.891	0.878	<input type="checkbox"/>
CSVF03.csv 2 days ago by Rohan Roy add submission details	0.881	0.872	<input type="checkbox"/>
CSVF01.csv 2 days ago by Rohan Roy add submission details	0.865	0.860	<input type="checkbox"/>
CSVF02.csv 2 days ago by Rohan Roy add submission details	0.868	0.872	<input type="checkbox"/>

CSVF07.csv 2 days ago by Rohan Roy add submission details	0.870	0.872	<input type="checkbox"/>
CSVF06.csv 2 days ago by Rohan Roy add submission details	0.877	0.878	<input type="checkbox"/>
CSVF09.csv 2 days ago by Rohan Roy add submission details	0.876	0.881	<input type="checkbox"/>
CSVF08.csv 2 days ago by Rohan Roy add submission details	0.880	0.874	<input type="checkbox"/>

Fig 1.17

CSVFinal.csv a day ago by Rohan Roy add submission details	0.902	0.902	<input type="checkbox"/>
---	-------	-------	--------------------------

Fig 1.18

As seen in Fig 1.17, the highest score obtained on the public leaderboard based on the single run models is 0.881 and the lowest score obtained is 0.860.

After average ensembling these models, the score obtained is 0.902 on the public leaderboard as shown in Fig 1.18.

A similar trend is observed in the case of the models obtained when using the Validation Loss as the monitor metric for the EarlyStopping callback function.

F00_val_loss.csv 3 days ago by pvijaikr	0.888	0.873	<input type="checkbox"/>
F01_val_loss.csv 3 days ago by pvijaikr	0.879	0.875	<input type="checkbox"/>
F02_val_loss.csv 3 days ago by pvijaikr	0.882	0.887	<input type="checkbox"/>
F03_val_loss.csv 3 days ago by pvijaikr	0.882	0.890	<input type="checkbox"/>
F04_val_loss.csv 3 days ago by pvijaikr	0.879	0.889	<input type="checkbox"/>
F05_val_loss.csv 2 days ago by pvijaikr	0.882	0.882	<input type="checkbox"/>

F06_val_loss.csv 2 days ago by pvijaikr	0.884	0.880	<input type="checkbox"/>
F07_val_loss.csv 2 days ago by pvijaikr	0.879	0.887	<input type="checkbox"/>
F08_val_loss.csv 2 days ago by pvijaikr	0.885	0.882	<input type="checkbox"/>
F09_val_loss.csv 2 days ago by pvijaikr	0.875	0.882	<input type="checkbox"/>

Fig 1.19

CSVFinal0.5.csv a few seconds ago by Mona Chowdary add submission details	0.898	0.899	<input type="checkbox"/>
---	-------	-------	--------------------------

Fig 1.20

As seen from the above Fig 1.19, the highest score and the lowest score obtained by using the single models when using the Validation loss as the metric for the EarlyStopping are 0.890 and 0.873.

When these 10 models are ensembled, we get a score of 0.899 on the public leaderboard as shown in Fig 1.20

Submission and Description	Private Score	Public Score	Use for Final Score
CSVFinal1.csv 2 hours ago by Mona Chowdary add submission details	0.908	0.909	<input type="checkbox"/>

Fig 1.21

When all the 20 models are ensembled together using average ensembling, a final score of 0.909 is obtained as seen in Fig 1.21. This was the final output file used to generate the output score.

4.5 Choice of Loss Function and Optimizer:

As the output value is the probability of the two people represented by the pair images being blood-related, the output value is between 0 to 1. This means that the output value is going to be in continuous range between 0 to 1, hence there two classes the output would get mainly classified to, hence the choice of the loss function which is the binary cross entropy function.

The Adam optimizer was used as it is a combination of the gradient descent with momentum algorithm and the RMS (Root Mean Square) Prop algorithm. This enables us to optimise multiple aspects of the models using the same optimizer and hence it was chosen for the final model.

5. Challenges faced:

Some of the major difficulties and limitations faced while working on this project are listed below:

5.1 Limited GPU time

As the training data is huge, running the code on google collab or regular python interfaces is extremely time-consuming. This puts the team at a greater disadvantage as it would take hours to run the code and see the results of the different modifications made by the team. Hence the team decided to use Kaggle notebooks and run the code on the GPU version. However, Kaggle only provides 30 hours of GPU per account. This resulted in a limitation of the number of new strategies that could be implemented.

As a result, the team had to constantly analyze the pros and cons of the different ideas proposed and choose the ideas that the team predicts would work. These ideas were implemented in the Kaggle notebook in order to ensure that the results are generated in the least possible time. These generated results are then analyzed and further strategies are planned.

Additionally, some ideas are also implemented on Google Collab. However, this resulted in a lot of time consumption as it would take almost 10 hours to generate a single .csv file.

5.2 Optimizing the training process:

During the training process, it is observed that in some situations the training does not show any improvement in the accuracy after a certain process. Hence, the model is being trained for a longer period of time without any significant improvement. This resulted in unnecessary consumption of valuable resources and wastage of time and memory.

In order to overcome this challenge, the team implemented the Early stopping function in some of its training models. The early stopping function monitored the 'loss' value and ensured that the training is automatically terminated if the model does not show any improvements for 50 Epochs in a row.

However, figuring out the variables that are to be monitored during the optimization required a lot of trial and error as the actual scores are generated only when the final .csv file is generated. Hence it is impossible to figure out if the model shows significant improvement during the running phase.

Different training models are executed and their resulting output file and score are analyzed to figure out the best way to implement the early stopping function in order to reduce overfitting and optimize the model.

This process was very challenging as it demanded a lot of time and resources.

5.3 Scarcity of relevant study resources:

Although face recognition has been a very prominent concept in the field of Machine Learning research, the concept of using facial recognition to verify kinship is still in the initial stages of development. As a result, it was a challenge to find relevant study resources that could help guide in the process of developing the network architecture.

This limited our knowledge to the limited information available on the discussion forms of Kaggle and a few other online platforms. As a result, the team had to rely on its own intuition and experiments to come up with creative ways of developing the model and increasing the performance and accuracy.

6. Conclusion:

The project provided many meaningful insights in terms of both knowledge in the field of machine learning as well as working together as a team with people from different ideologies and backgrounds. Some of the important concepts the team learnt during the different stages of completing the project are as follows:

Machine learning concepts:

The project helped refresh several different concepts of machine learning and helped us go through other innovative solutions for machine learning, especially for kaggle competitions. After going through such concepts we adopted them for our problem and modified it to attain the best results. Some of the implemented concepts are:

- Usage of Pretrained VGGFace
- Preprocessing the Data
- Feature extraction
- Different types of neural network layers
- Different types of callback functions
- Implementation of Machine Learning models using keras and tensorflow
- K Fold cross validation
- Types of Ensembling

Strategies to improve ranking:

Some of the strategies that were used to improve ranking were learnt through research as well as through trial and error experimentation. Some of the important lessons learnt are:

- The order in which additional layers are added to the existing VGGFace baseline model has a significant impact on the training and the overall score. This is because the image size changes when it is passed through certain layers. Some layers reduce the size of the image and thereby leading to the loss of some features.
- Sometimes, moving through multiple epochs does not show significant improvement in the model. Hence it is important to identify the areas of training that do not show any improvement and optimize the training by eliminating those areas.
- A combination of the output generated by multiple models in the form of .csv files shows a huge improvement in score.

Collaborative Skills:

Working together as a team during the project helped improve our people skills and throughout the course of the project all of us learnt that every person has individual strengths and weaknesses and that working together with humility and mutual respect can not only help improve the performance of the team as a whole but can also develop the skill of the individual team members and help them better prepare for their future workplace.

7. References:

- 1) [Northeastern SMILE Lab - Recognizing Faces in the Wild](#)
- 2) [SmileLab - Synergistic Media Learning Lab @ NEU](#)
- 3) <https://github.com/akuritsyn/kaggle-recognizing-faces>
- 4) [rcmalli/keras-vggface: VGGFace implementation with Keras Framework](#)
- 5) [Core Layers](#)