Capstone Project

Machine Learning Engineer Nanodegree

Arwa Ibrahim
January 12, 2020

1. Definition

1.1 Project Overview

Face verification in unconstrained conditions has obtained increasing attention and encouraging progress in recent years. Biologists find that human facial appearance is an important cue for genetic similarity measurement. Motivated by this finding and related applications such as social media analysis, missing children searching, **Family Classification** and **kinship verification** through facial image analysis has attracted more and more attention over the past few years.

The goal of kinship verification is to determine whether a pair of faces are blood relatives or not, while the goal of family classification is to determine the family that an unseen subject belongs to. That's why they have several applications such as locating relatives in public databases, determining the kin of a victim or suspect by law enforcement agencies, screening asylum applications where kinship relationships are to be determined, organizing and resolving identities in photo albums.

In the seek of solving these problems there are many datasets proposed for this task such as: UB KinFace Dataset, Cornell Kinship Dataset, KinFace-I, KinFace-II and finally the FIW database. The largest public database that contains 11 different kin relations and provides annotation for both relationships and spatial information of each family member and pairwise kin relations in the family photo which are helpful in social media analytics.

1.2 Problem Statement

The problem statement can be divided into 2 main parts described as follows:

- **Kinship Verification:** It's a one_to_one classification problem that intended to determine whether or not a pair of facial images are blood relatives. This is a classical boolean problem with system responses being either kin and non-kin (i.e., related and unrelated, respectfully). Thus, this task tackles the one-to-one view of automatic kinship recognition.

- **Family Classification:** It's a one_to_many classification problem that intended to determine the family that an unseen subject belongs to. This is done by referencing faces, families modeled using facial images of all but the held-out family members, then at test-time the held out members are used to evaluate on. Types of held out family members vary by type from the youngest boy in a family tree that spans back several generations to a member whom assumes the role of being a mother, a sister, a daughter, and sits right in the middle of the family tree. Hence, this task is formulated as a one-to-many, closed-form classification problem.

In order to solve those tasks a comparison between the existed databases is made as shown in the table below, which ends up with a decision of using the FIW database.

Dataset	No. Families	No. People	No. Faces	Age Variation	Family Structure
CornellKin	150	300	300	No	No
UB KinFace-I	90	180	270	Yes	NO
UB KinFace-II	200	400	600	Yes	NO
KFW-I	___	1066	1066	NO	NO
KFW-II	___	2000	2000	NO	NO
TSKinFace	787	2589	14816	Yes	Yes
Family 101	101	607	14816	Yes	Yes
WVU	__	226	904	Yes	__
FIW	936	10676	30725	Yes	Yes

Table 1. Comparison of FIW with related datasets.

1.3 Metrics

As there are two different tasks in this project the metrics chosen will differ according to which task to solve as follows:

1.3.1 Kinship Verification

The evaluation metric chosen is the AUC - ROC curve. AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. Where ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s (non-related) as 0s and 1s (totally related) as 1s.

1.3.2 Family Classification

The evaluation metric chosen is the confusion matrix. A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.
The confusion matrix shows the ways in which your classification model is confused when it makes predictions and gives insight not only into the errors being made by the classifier but more importantly the types of errors that are being made.
It is this breakdown that overcomes the limitation of using classification accuracy alone.

2. Analysis

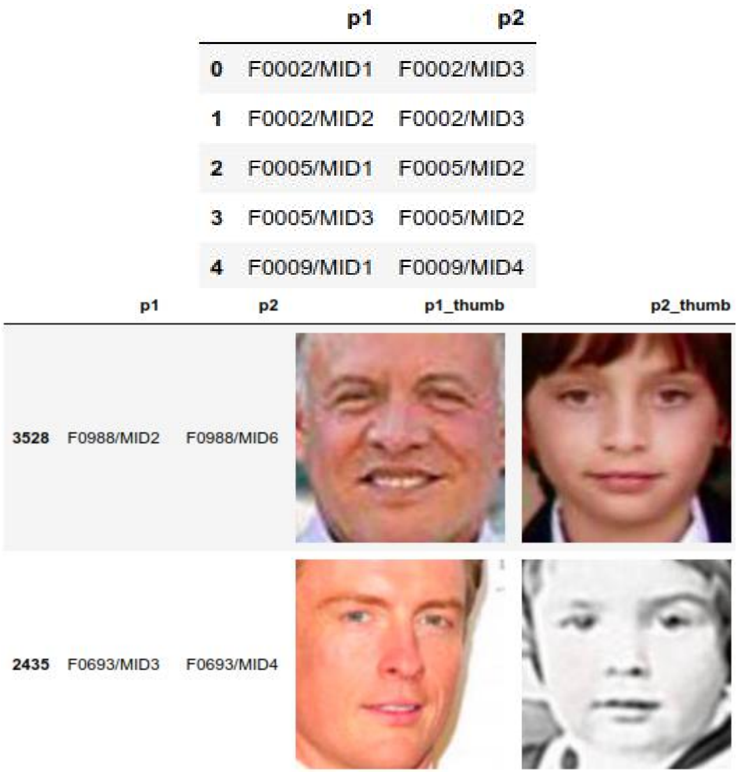
2.1 Data Exploration

The project works on 2 different versions of the FIW database, the provided database in the FIW kaggle competition is used for the kinship verification problem while the database that provided in the FIW2020 codalab competition is used for family classification problem. Both databases are almost identical to the original FIW database. The decision of working with a competition versions instead of the original is taken in order to test the model and compare it against a fixed testing mechanism.

2.1.1 Kinship Verification

The kaggle database consists of 4 files as follows: train databases, test database, and 2 csv files one of them contains training pairs and the other for test pairs.

The train_relationships.csv file consists of 3598 data records each record represent a pair of images. So it is consists of 2 columns labeled as **p1** (the first image code) and **p2** (the second image code).



	p1	p2
0	F0002/MID1	F0002/MID3
1	F0002/MID2	F0002/MID3
2	F0005/MID1	F0005/MID2
3	F0005/MID3	F0005/MID2
4	F0009/MID1	F0009/MID4

	p1	p2	p1_thumb	p2_thumb
3528	F0988/MID2	F0988/MID6		
2435	F0693/MID3	F0693/MID4		

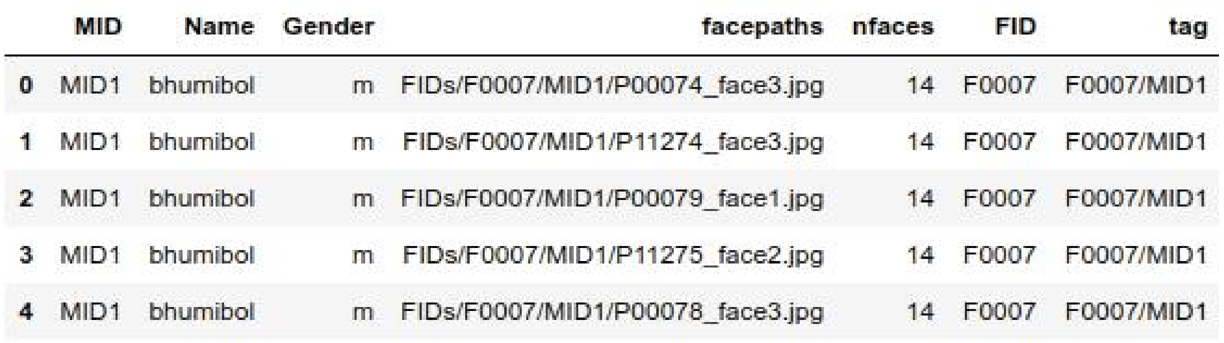
Fig. 1 The head of the train pairs dataframe along with a sample pairs of photos.

2.1.2 Family Classification

The codalab database consists of 4 files as follows: train/gallery databases, test/FIDs database, and 2 csv files one of them contains training pairs and the other for validation pairs.

The val_gallery_face_list.xlsx file consists of 4030 data records each record represent a family member image and some information about it. So it is consists of 7 columns described as follows:

- **MID** contains the member id also represents the name of the folder that contains the member photos.
- **Name** of the person in the photo.
- **Gender** of the person in the photo.
- **Facepaths** column contains a path to a photo of the person.
- **Nfaces** represent the number of photos for that person that exists in the database.
- **FID** contains the family id of the person.
- **Tage** contains the tag of the image that represents who is the person (the MID part) and from which family is the person.



	MID	Name	Gender	facepaths	nfaces	FID	tag
0	MID1	bhumibol	m	FIDs/F0007/MID1/P00074_face3.jpg	14	F0007	F0007/MID1
1	MID1	bhumibol	m	FIDs/F0007/MID1/P11274_face3.jpg	14	F0007	F0007/MID1
2	MID1	bhumibol	m	FIDs/F0007/MID1/P00079_face1.jpg	14	F0007	F0007/MID1
3	MID1	bhumibol	m	FIDs/F0007/MID1/P11275_face2.jpg	14	F0007	F0007/MID1
4	MID1	bhumibol	m	FIDs/F0007/MID1/P00078_face3.jpg	14	F0007	F0007/MID1

Fig. 2 The head of the val_gallery_face_list dataframe.

2.2 Exploratory Visualization

Families In the Wild (FIW) is the most comprehensive image database for automatic kinship recognition with the data distributed fairly with average of about 12 images per family, each with at least 3 and as many as 38 members. This data supports all prior pair-wise types and also introduces grandparent-grandchild pairs (GF-GD, GF-GS, GM-GD, GM-GS). In addition of it is a public and noncommercial database.

2.2.1 Kinship Verification

For this task the kaggle version was used which contains 1000 family, each of them contains from 3 to 38 member. For visualization purpose the 1000 families divided into 10 groups each of which contains 100 family.

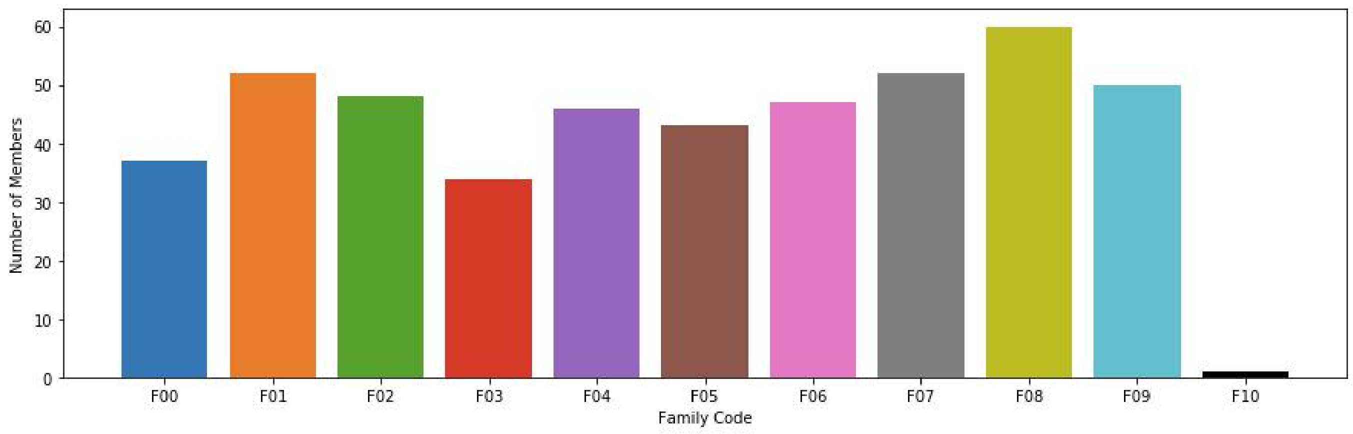


Fig. 3 The image shows the number of accumulative number of family members in each group.

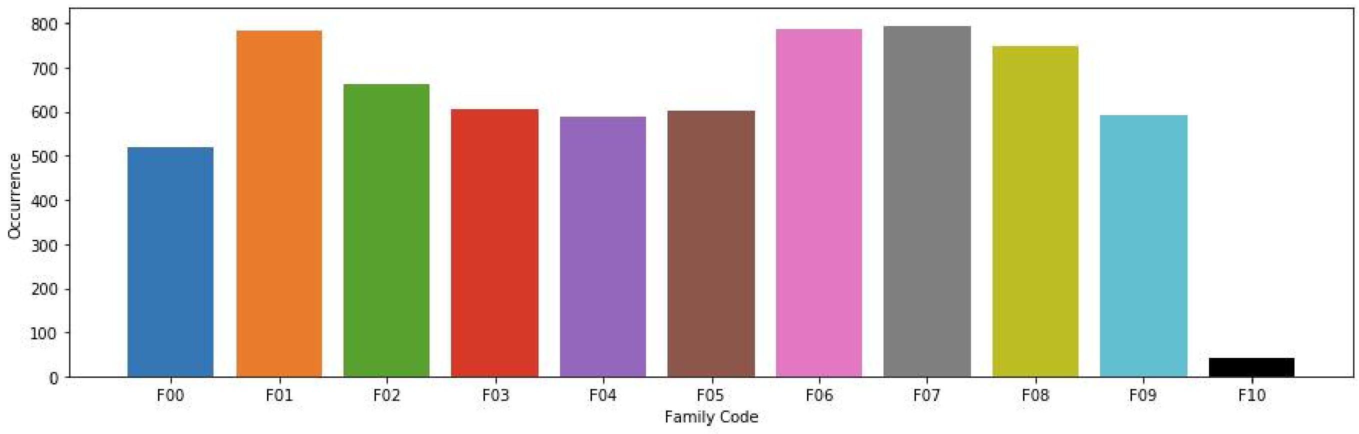


Fig. 4 The image shows the number of concurrence in the training of each group.

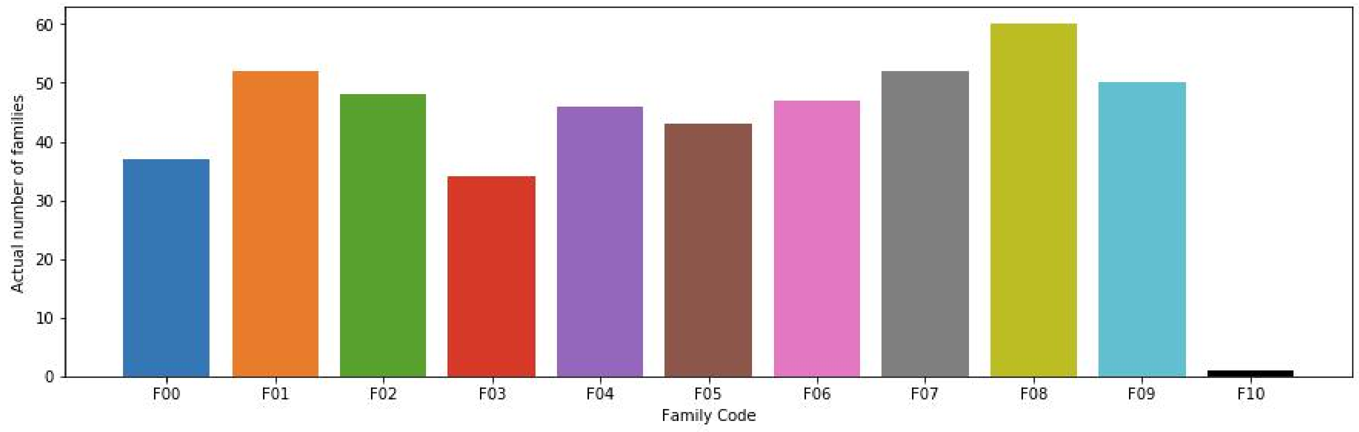


Fig. 5 The image shows the actual number of families in each group**.**

2.2.2 Family Classification

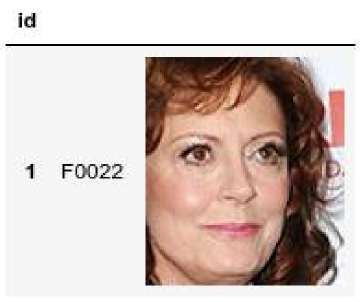
For this task the codalab version was used which contains 192 family, each of them contains up to 5 members (before excluding a member for validation).

Fig. 6 shows a family of five members which have 4 members included in the training and the excluded member that is used in the validation.

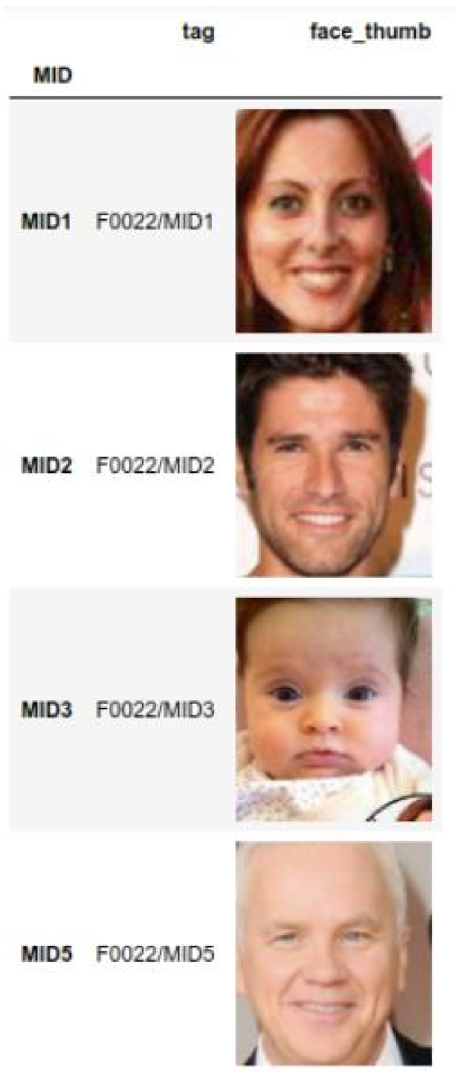
It also shows that the FIW dataset supports age variations as described above along with a gender variation.

It also shows that the FIW dataset supports all prior pair-wise types and also introduces grandparent grandchild pairs such as:

- Grandfather Granddaughter GF-GD.
- Grandfather Grandson GF-GS.
- Grandmother Granddaughter GM-GD.
- Grandmother Grandson GM-GS.



id		
1	F0022	



	tag	face_thumb
MID		
MID1	F0022/MID1	
MID2	F0022/MID2	
MID3	F0022/MID3	
MID5	F0022/MID5	

2.3 Algorithms and Techniques

2.3.1 Kinship Verification

The algorithm followed is use the Vgg-FaceNet to construct a siamese network using the accuracy as metrics and the triplet loss.

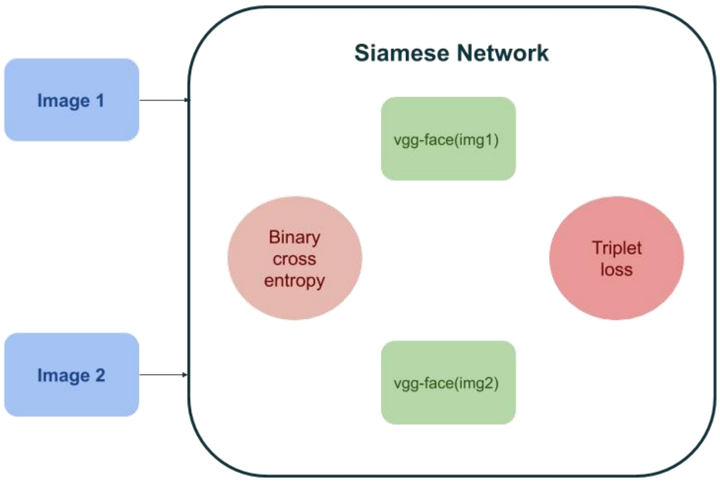


Fig. 7 Describes the network in a simple way.

2.3.2 Family Classification

A simple CNN is used that it is consists of around 8 main layers.

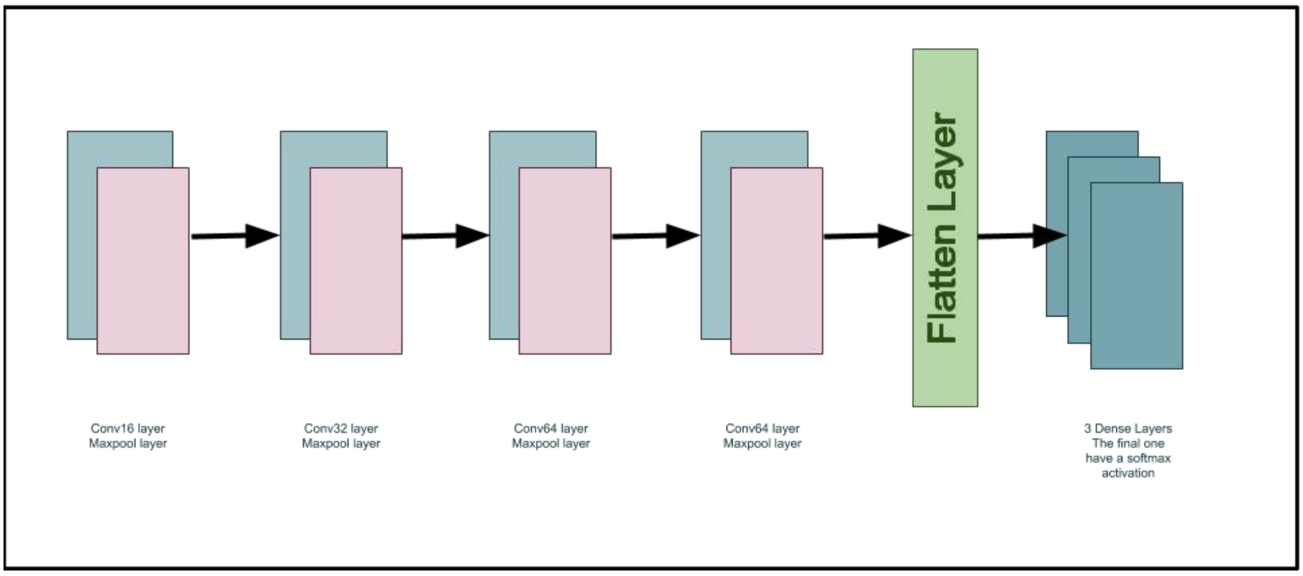
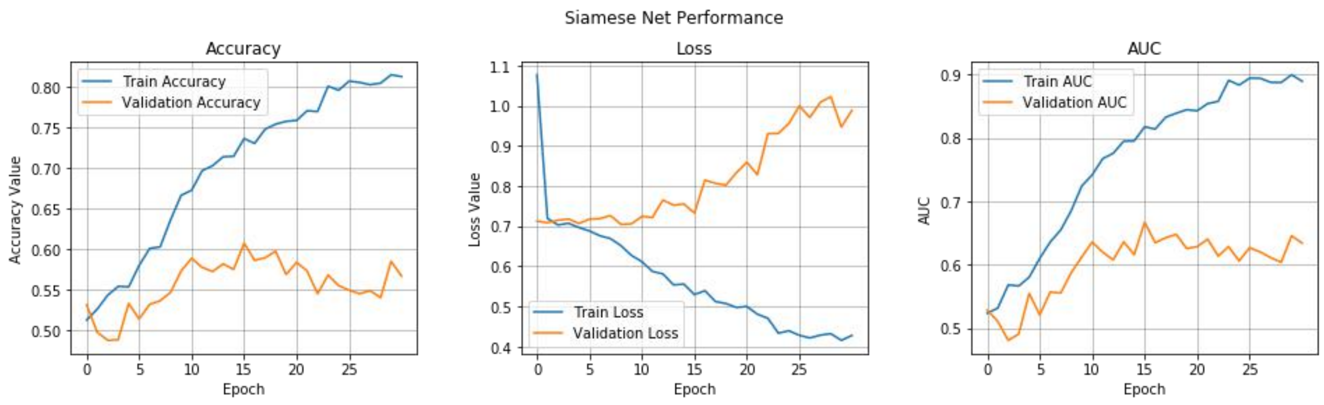


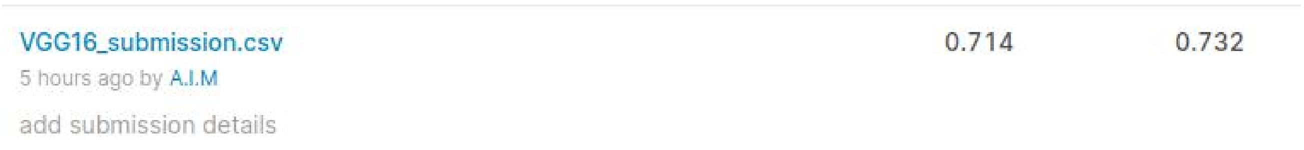
Fig. 8 Describes the network in a simple way.

2.4 Benchmark

2.4.1 Kinship Verification

The used benchmark is the VGG - Face VGG16 model trained as a siamese network on each pair of images then the triplet loss and the auc curve are calculated.





VGG16_submission.csv 0.714 0.732

5 hours ago by A.I.M

add submission details

Fig. 9 Shows the auc curves and the test accuracy for the VGG16 model.

2.4.2 Family Classification

The used benchmark is the random guessing of the excluded member family (class) which gives an accuracy of 0.6% with a confusion matrix shown below.

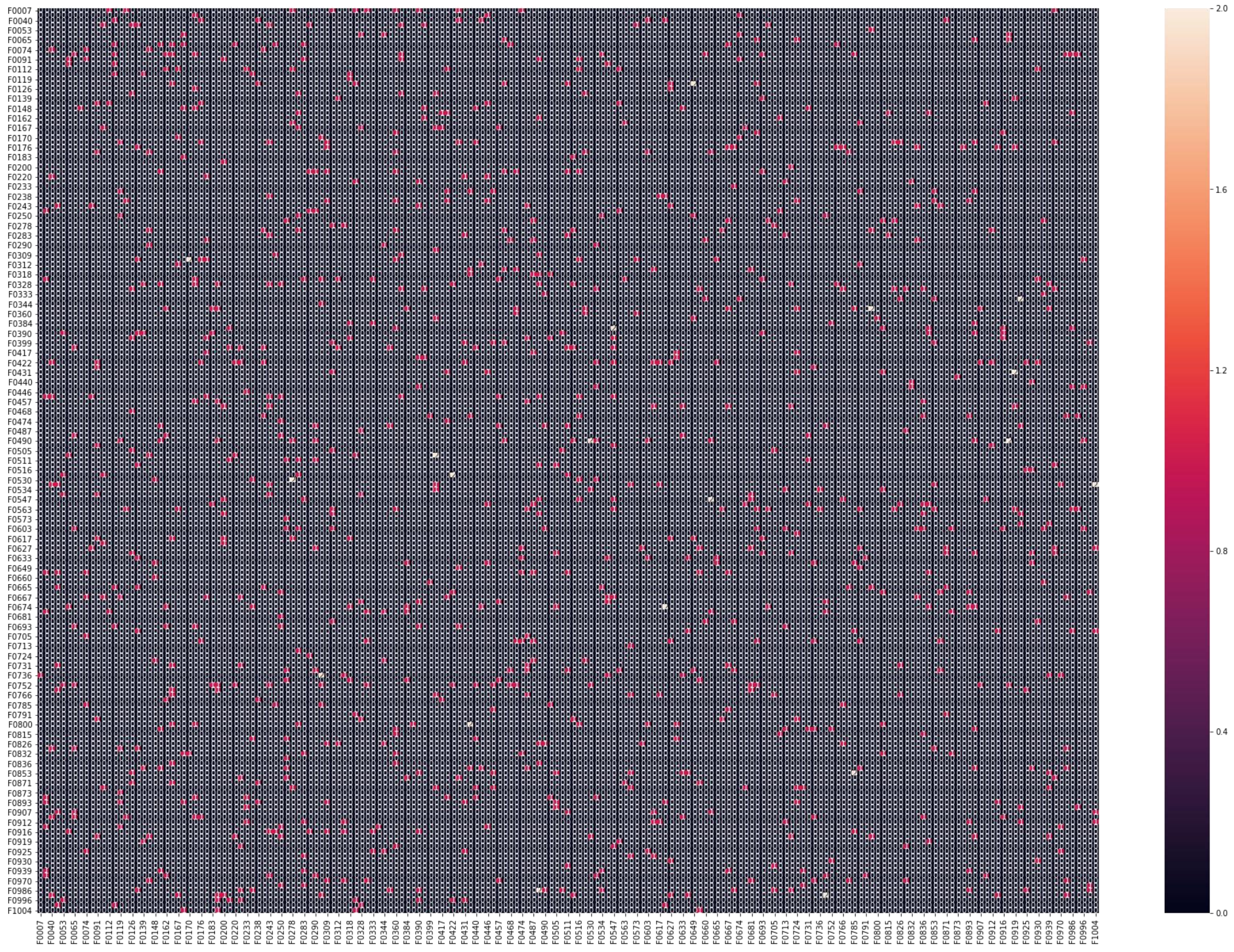


Fig. 10 Shows the confusion matrix of the random guess output.

3. Methodology

3.1 Data Preprocessing

3.1.1 Kinship Verification

- Firstly 2 columns added to the dataframe of the training pairs that contains a path for an image of each of the member pairs.
- As we using a smaller version of the FIW database, for each record of the pair the existence of that pair is tested and any of the records that has a non-existing pair is excluded.
- By dividing the families into 10 groups each contains 100 families, the number of occurrence of each group was calculated then this analysis used to exclude the F06 and F09 groups to be used as a validation data as they have about 20% of the training pairs.

3.1.2 Family Classification

For training data:

- Firstly the dataframe of a training data contains some additional information that haven't a high impact for this problem (ex, Name, Gender,..).
- For the facepaths column an update for all paths was made with our files path.
- As we using a smaller version of the FIW database, there were a need to check if the training dataframe have a valid paths (refers to an existing image). Fortunately, all images of the training were existing.
- Using the previous dataframe we split it into 2 lists, X that contains the input images to the CNN and y that contains the labels of those images in 1 hot encoding format.
- Silting the previous described X and y into a training (80%) and a validation (20%) sets.

For testing data:

- Firstly the dataframe of a testing images contains some additional information that haven't a high impact for this problem (ex, nfaces).
- For the facepaths column an update for all paths was made with our files path.
- As we using a smaller version of the FIW database, there were a need to check if the testing images exists on this version of FIW. Fortunately, all images of the training were existing.
- A column for the family id (FID) was added.
- In the seek of making the data consistent we check the number of unique FID (classes) then exclude all the training records that have FID that doesn't exist in the testing data. It's founded that all of those families contains a number of members less than 5.
- Finally, using the previous dataframe we split it into 2 lists, X_pred that contains the input images to the model to predict and t_y_test that contains the true labels of those images in 1 hot encoding format to be compared with the predicted labels.

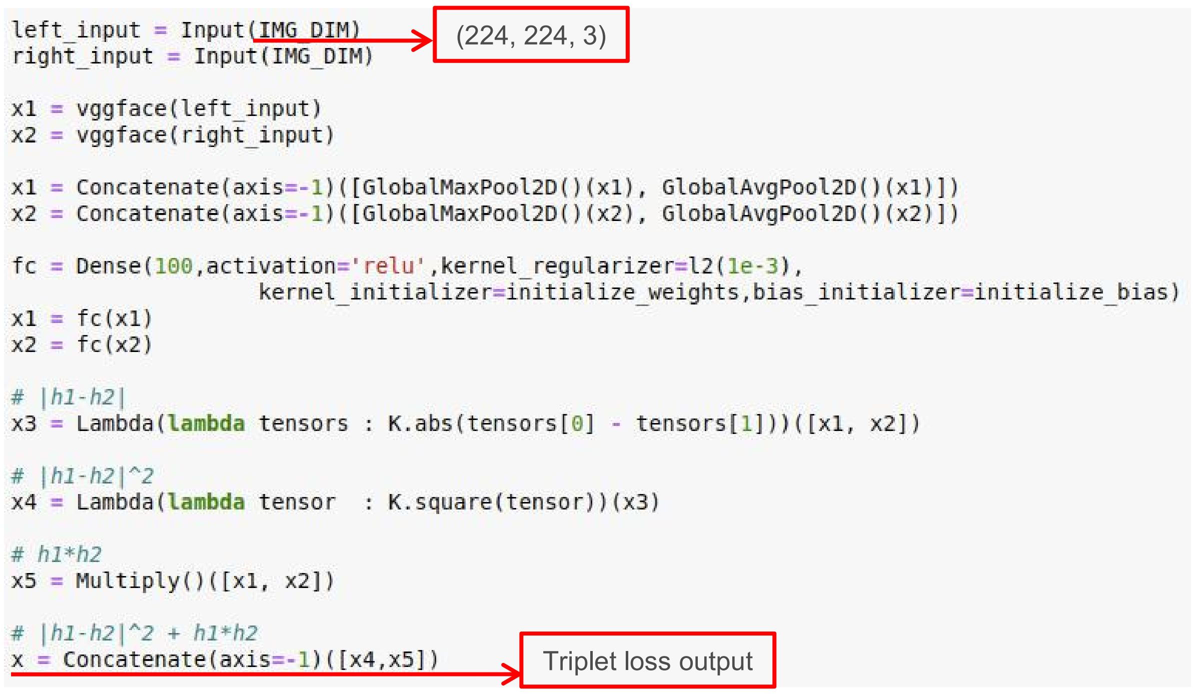
3.2 Implementation

3.2.1 Kinship Verification

- The first step for us is to determine the bias and weights values that may lead to a better training of the model so **initialize_bias** and **initialize_weights** are defined that used to initialize the bias with mean as 0.5 and standard deviation of 0.01, and weights with mean as 0.0 and standard deviation of 0.01 respectively.
- Initialize a vgg-face framework with the ResNet50/SENET50 model.
- Define a function that calculates the AUC - ROC curve as shown below.

```
def auc(y_true, y_pred):
    return tf.py_func(roc_auc_score, (y_true, y_pred), tf.double)
```

- Feed the images to the vgg-face followed by 2 pooling layers and a final dense layer. Then calculate the triplet loss function between the output of the pairs as shown in the below figure.



```
left_input = Input(IMG_DIM)
right_input = Input(IMG_DIM)

x1 = vggface(left_input)
x2 = vggface(right_input)

x1 = Concatenate(axis=-1)([GlobalMaxPool2D()(x1), GlobalAvgPool2D()(x1)])
x2 = Concatenate(axis=-1)([GlobalMaxPool2D()(x2), GlobalAvgPool2D()(x2)])

fc = Dense(100,activation='relu',kernel_regularizer=l2(1e-3),
                   kernel_initializer=initialize_weights,bias_initializer=initialize_bias)
x1 = fc(x1)
x2 = fc(x2)

# |h1-h2|
x3 = Lambda(lambda tensors : K.abs(tensors[0] - tensors[1]))([x1, x2])

# |h1-h2|^2
x4 = Lambda(lambda tensor  : K.square(tensor))(x3)

# h1*h2
x5 = Multiply()([x1, x2])

# |h1-h2|^2 + h1*h2
x = Concatenate(axis=-1)([x4,x5])
```

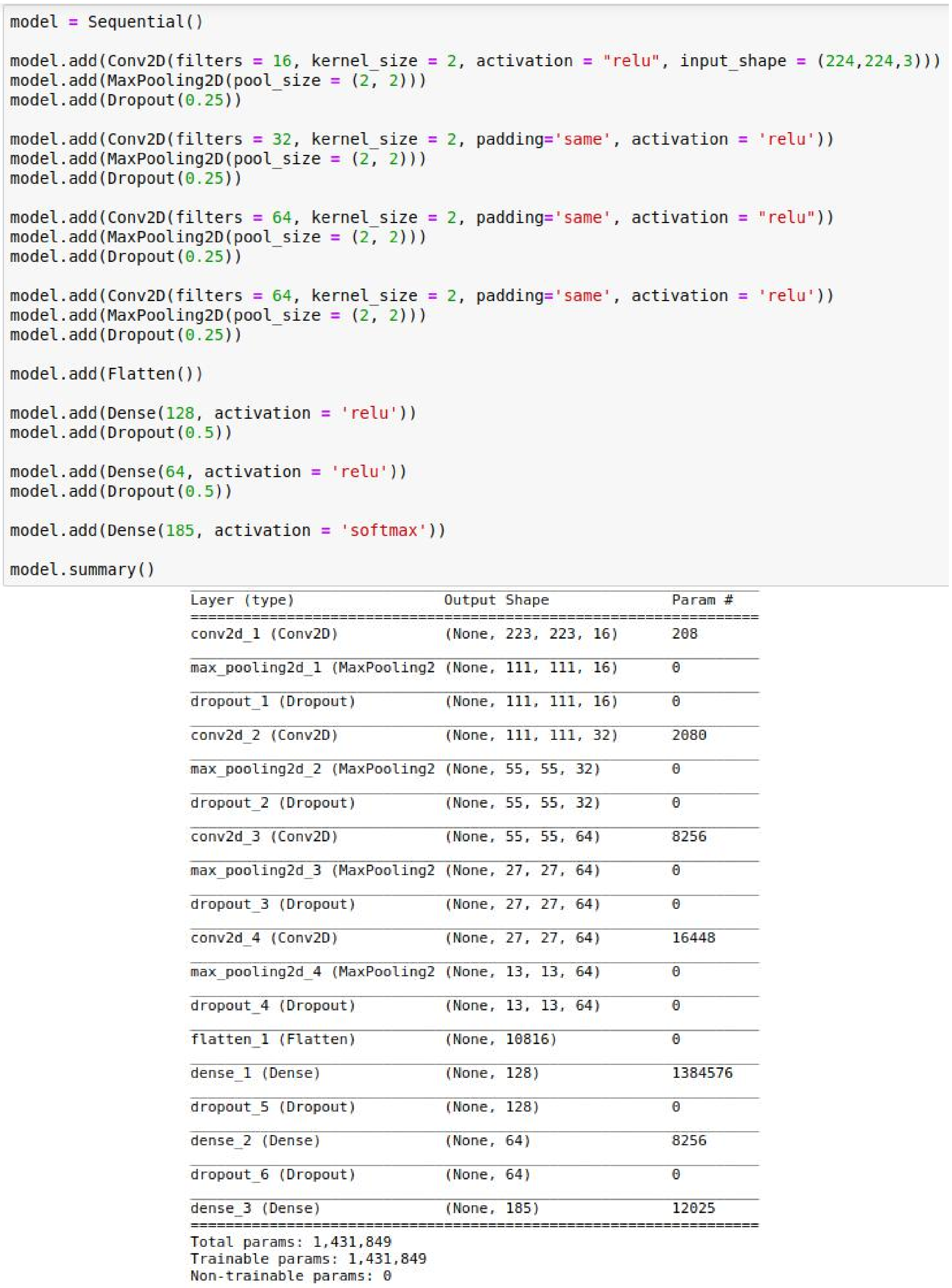
Fig. 11 Shows the steps of feeding the pair to the vgg-facenet and calculating the triplet loss function.

- Feed the output of triplet loss to a dense layer and construct the siamese network as shown below.

```
x = Dense(100,activation='relu',kernel_regularizer=l2(1e-3),
                  kernel_initializer=initialize_weights,bias_initializer=initialize_bias)(x)
x = Dropout(0.2)(x)


prediction = Dense(1,activation='sigmoid',bias_initializer=initialize_bias)(x)

siamese_net = Model(inputs=[left_input,right_input],outputs=prediction)

optimizer = Adam(1e-5)

siamese_net.compile(loss="binary_crossentropy",optimizer=optimizer,metrics=['accuracy',auc])
```

3.2.2 Family Classification

As mentioned in 2.3.2 we used a simple CNN to train the model. The construction of the used CNN shown in the **Fig. 7** described as shown in the below figures.



```
model = Sequential()

model.add(Conv2D(filters = 16, kernel_size = 2, activation = "relu", input_shape = (224,224,3)))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 32, kernel_size = 2, padding='same', activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size = 2, padding='same', activation = "relu"))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size = 2, padding='same', activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.5))

model.add(Dense(64, activation = 'relu'))
model.add(Dropout(0.5))

model.add(Dense(185, activation = 'softmax'))

model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 111, 111, 32)      2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 55, 55, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 27, 27, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 27, 27, 64)        16448
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 13, 13, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 10816)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1384576
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 185)               12025
=================================================================
Total params: 1,431,849
Trainable params: 1,431,849
Non-trainable params: 0
```

Fig. 12 Shows the proposed CNN and the model summary.

3.3 Refinement

Kinship verification task:

As mentioned in the previous sections, the faceNet models trained with the siamese network which gives 77% accuracy on the testing data set according to the kaggle submissions output.

The following figures contains plots of the training/validation accuracy, losses and AUC for both models (SENET50 / ResNet50). Divergence indicates that models overfitting, which can be addressed by intensifying the weight decay, adding dropout, or reducing the model complexity (e.g. reducing the number of layers), among other techniques.

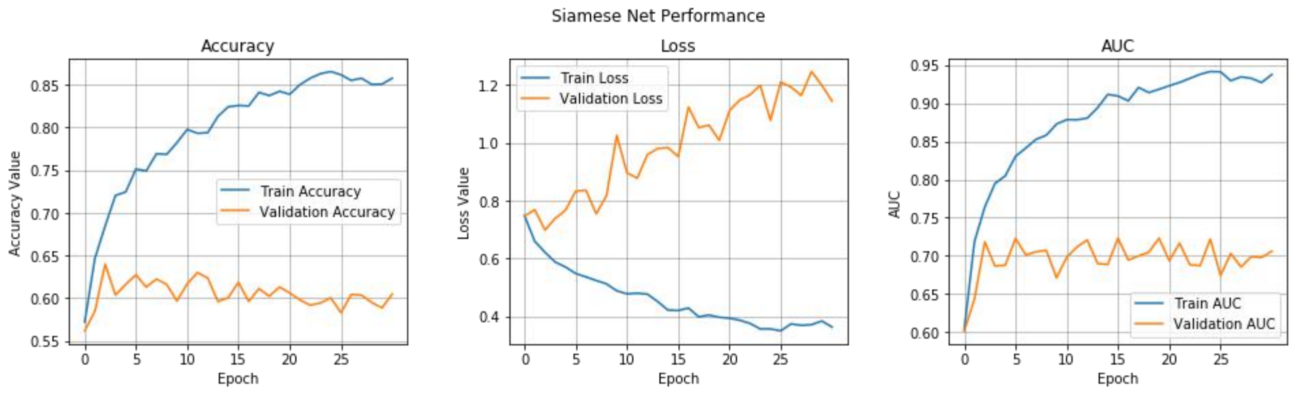


Fig.13 SENET50 model performance.

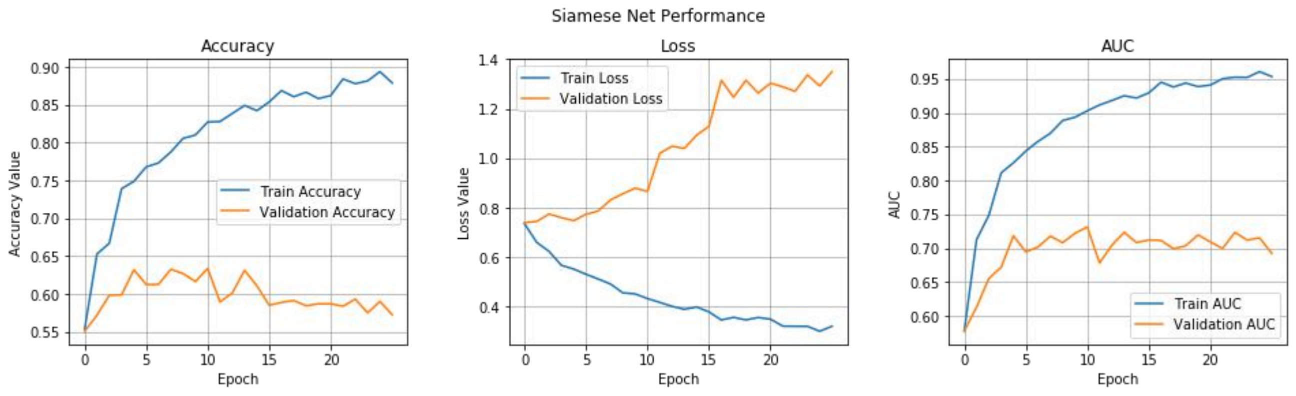


Fig. 14 ResNet50 model performance.

4. Results

4.1 Model Evaluation and Validation

4.1.1 Kinship Verification

To verify the robustness of the final model, a test was conducted using testing data and submitting the results against the kaggle submission(see **Fig. 11**, **Fig. 12**).

The following observations are based on the results of the test:

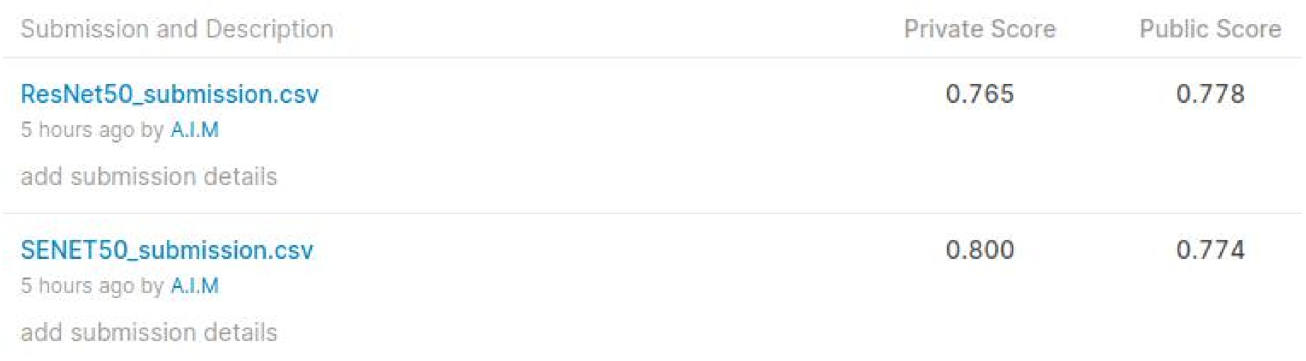
- ❖ The classifier can reliably detect the kin related pairs.
- ❖ Images with peoples of different ages can be applied and detect fairly.
- ❖ False positives are rare but present.

4.2 Justification

4.2.1 Kinship Verification

To test the models and compare there results, the kaggle judging is used and the results were as shown in the below figure.

The best accuracy is achieved by the ResNet50 model while the worst is that achieved by the VGG16 model which works as our benchmark model.



Submission and Description	Private Score	Public Score
ResNet50_submission.csv 5 hours ago by A.I.M add submission details	0.765	0.778
SENET50_submission.csv 5 hours ago by A.I.M add submission details	0.800	0.774

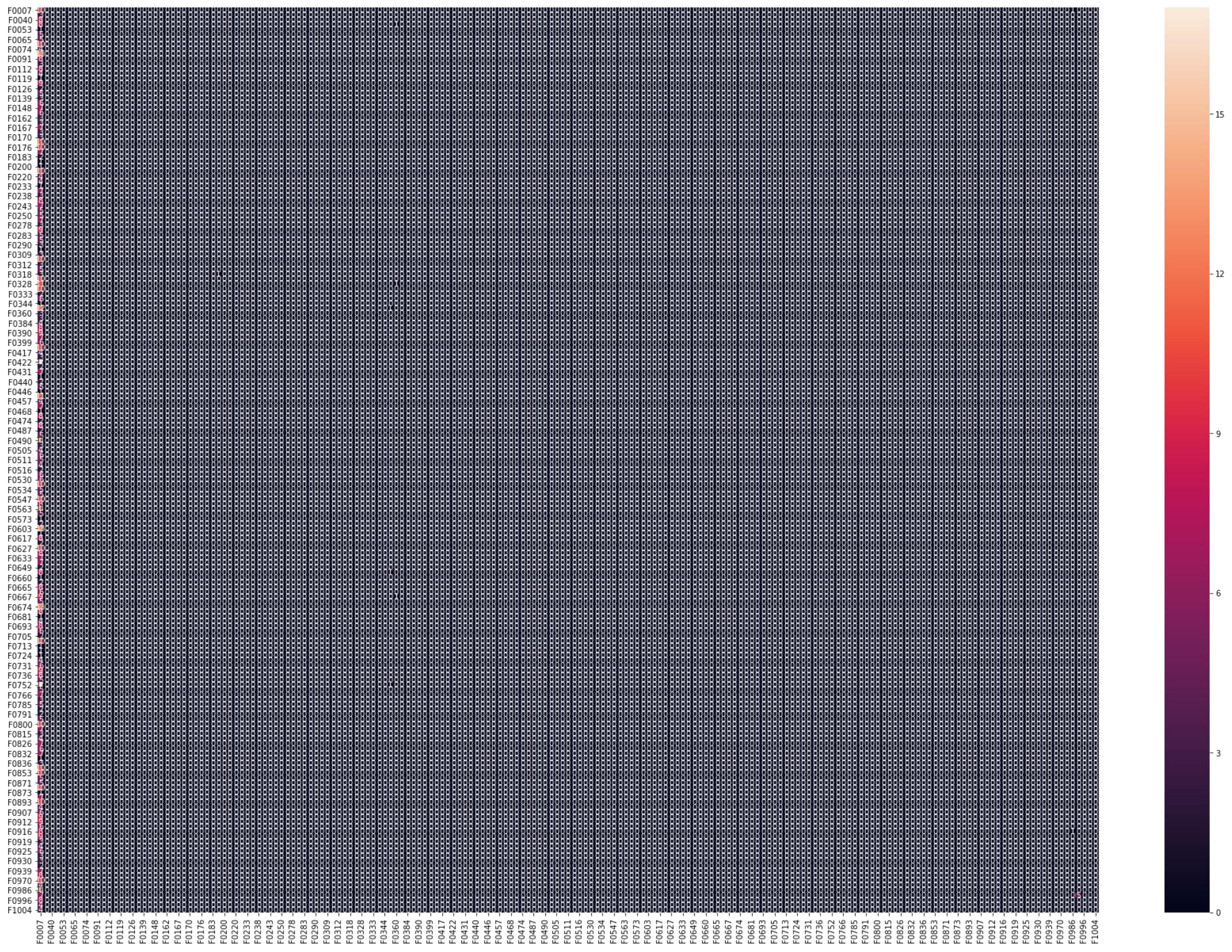
Fig. 15 Shows the results achieved by the different models.

4.2.2 Family Classification

To test the models and compare there results, the model was applied on a list of excluded family members to predict the class/family for each of them. The CNN have a low accuracy even lesser than the random guessing!. The achieved accuracy reaches 0.5524%.

Fig. 16 Shows the confusion matrix along with a shortest version of classification report.

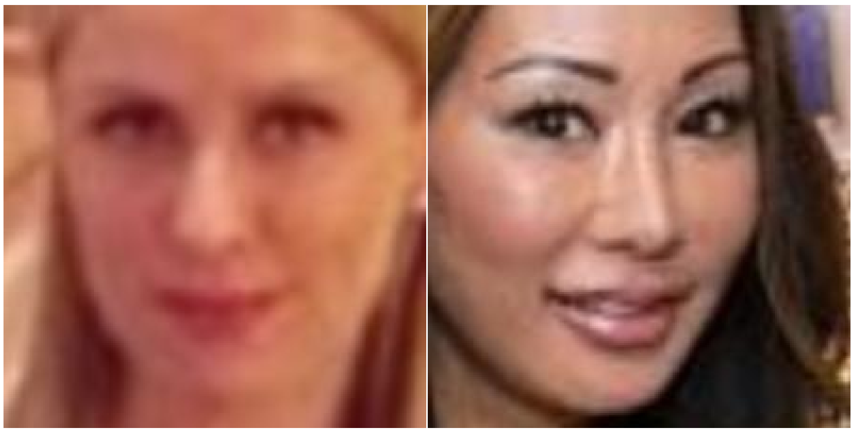
```
Classification Report           precision     recall   f1-score     support
          F0007          0.00         0.00         0.00          11
          F0087          0.00         0.00         0.00          13
          F0328          0.00         0.00         0.00          12
          F0358          0.33         0.08         0.12          13
          F0422          0.00         0.00         0.00          17
          F0448          0.00         0.00         0.00          11
          F0490          0.00         0.00         0.00          16
          F0563          0.00         0.00         0.00          16
          F0603          0.00         0.00         0.00          13
          F0674          0.00         0.00         0.00          13
          F0752          0.00         0.00         0.00          18
          F0986          0.00         0.00         0.00          16
          F0990          1.00         0.42         0.59          12
avg / total          0.02         0.01         0.01         1086
```



5. Conclusion

5.1 Free-Form Visualization

Here are some samples from the tested image pairs of kin related and non kin related pairs.



Model	Is related ?
VGG16	0.09
SENET50	0.086
ResNet50	0.0036

Fig. 17 Shows an examples of non_kin related pair that classified as non kin pair by the application.

Model	Is related ?
VGG16	0.994
SENET50	0.938
ResNet50	0.946

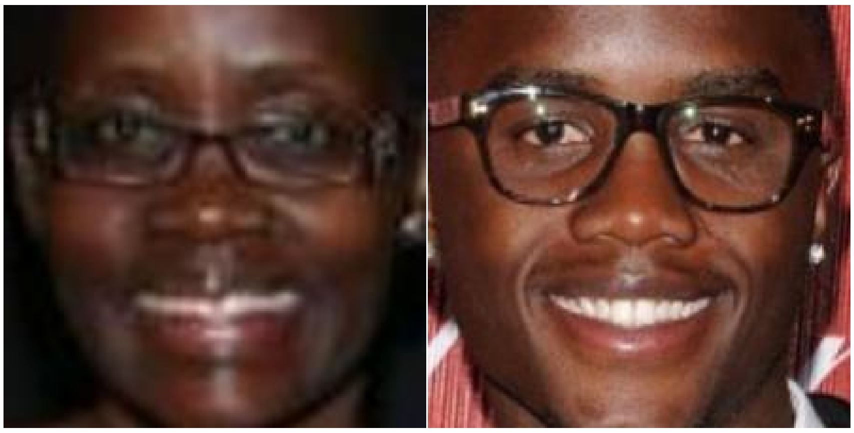


Fig. 18 Shows an examples of a kin related pair that classified as a kin pair by the application.

5.2 Reflection

- For **family classification** part there were two main challenges:

  I. The high number of classes that represents the families codes. This problem makes the visualization and analysis a harder problem to deal with.

  II. The small number of samples in each of those classes. Comparing with the ordinary one_to_many classification problems there were lesser number of classes and a larger number of samples per class which help the network to learn the common feature much better.

- For **kinship verification** part the initial challenge was the model training but after some searching it’s found that the triplet-loss along with the siamese will solve this for me as:

  I. In siamese networks, we take an input image of a person and find out the encodings of that image, then, we take the same network without performing any updates on weights or biases and input an image of a other person of the training-pair and again predict it’s encodings.

  II. Then compare these two encodings to check whether there is a similarity between the two images. These two encodings act as a latent feature representation of the images. Images with the same person (or in this case a pair with kin relation) have similar features/encodings. Using this, we compare and tell if the two persons/images have kin relation or not.

5.3 Improvement

5.3.1 Kinship Verification

There were multiple of interesting improvements such as:

I. Discover the effect of previous face recognition on the performance.
II. Discover the effect of previous facial points detection.

More accuracy advancement can be achieved by developing the following:

a) In building databases for kinship verification process, often different individuals’ images have been cropped from larger family photos. While this is a good way to find people in the same family it also builds in another clue – a bias that people from the same photo are related. This has been overlooked as a potential issue, as it is far easier to identify whether two face images are from the same original photo than it is to determine if they are kin. So, this ‘from same photograph’ (FSP) signal can be used to achieve results comparable to the state of the art.

b) Most of the methods proposed so far are to verify kinship from images. So, it’s interesting to try a method using facial dynamics to verify kinship from videos that enable us test the approach of that people may not look like their parents, until they smile. Furthermore, findings of shows that the appearance of spontaneous facial expressions of born-blind people and their sighted relatives are similar. However, the resemblance between facial expressions depends not only on the appearance of the expression but also on its dynamics, as each expression is created by a combination of voluntary and involuntary muscle movements.

5.3.2 Family Classifications

This part of problem is a new research topic. So, there are much more interesting things to try that may lead to a big movement in that area of research.

As we proposed a simple network for the purpose of figuring out how to deal with this kind of problems, there are some basic improvements such as:

I. Using a pretrained network such as Xcepthin, ResNet50, ... etc.
II. Use image augmentation to increase the number of samples per each class.
III. Dealing with images of group of people from different families.

6. References

[1] R. Fang, K. D. Tang, N. Snavely, and T.Chen, “Towards computational models of kinship verification”, 2010.

[2] S. Xia, M. Shao, J. Luo, and Y. Fu, “Understanding kin relationships in a photo” , 2012.

[3] J. Lu, X. Zhou, Y.-P. Tan, Y. Shang, and J. Zhou, “Neighborhood repulsed metric learning for kinship verification”, 2014.

[4] Joseph P Robinson, Ming Shao, Yue Wu and Yun Fu, “Family in the Wild (FIW): A Large-scale Kinship Recognition Database”, 2016.

[5] Naman Kohli, “Automatic Kinship Verification in Unconstrained Faces using Deep Learning”, 2019.

[6] Joseph P Robinson, Ming Shao, Yue Wu and Yun Fu, “Recognizing Families In the Wild (RFIW)”, 2017.

[7] Joseph P Robinson, Ming Shao, Yue Wu and Yun Fu, “Visual Kinship Recognition of Families in the Wild”, 2018.

[8] Shuyang Wang, Joseph P. Robinson, and Yun Fu, “Kinship Verification on Families in the Wild with Marginalized Denoising Metric Learning”, 2017.

[9] Naman Kohli, Mayank Vatsa, Richa Singh, Afzel Noore and Angshul Majumdar, “Hierarchical Representation Learning for Kinship Verification”, 2018.

[10] Kaihao Zhang, Yongzhen Huang, Chunfeng Song, Hong Wu and Liang Wang, “Kinship Verification with Deep Convolutional Neural Networks”, 2015.

[11] Hamdi Dibeklioglu, Albert Ali Salah, and Theo Gevers, “Like Father, Like Son: Facial Expression Dynamics for Kinship Verification”, 2013.

[12] Mitchell Dawson, Andrew Zisserman and Christoffer Nellåker, “FROM SAME PHOTO: CHEATING ON VISUAL KINSHIP CHALLENGES”, 2018

[13] Mitchell Dawson, Andrew Zisserman, and Christoffer Nellåker. “From Same Photo: Cheating on Visual Kinship Challenges”, 2018.

[14] G. Peleg, G. Katzir, O. Peleg, M. Kamara, L. Brodsky, H. Hel-Or, D. Keren, and E. Nevo. “Hereditary family signature of facial expression”, 2006.

[15] Hamdi Dibeklioglu, Albert Ali Salah, and Theo Gevers. “Like Father, Like Son: Facial Expression Dynamics for Kinship Verification”, 2013.

[16] Gregory Koch, “Siamese Neural Networks for One-Shot Image Recognition”, 2015.