

**CZ4041 Machine Learning**

| **Name** | **Matric Number** |
| --- | --- |
| Ong Hui Lee Grace | U1921756L |
| Joelle Thng | U1920277K |
| Wang Anyi | U1922992C |

**Contents Page**

[**1. Introduction**](#_a15ruh6m361m) **3**

[1.1 Problem Statement](#_4ezlmuh69dp0) 3

[1.2 Related works](#_46t3ydcq5ozc) 3

[1.3 Challenges](#_b6ssveyl3077) 3

[**2. Methodology**](#_cxt4mj7laapn) **4**

[2.1 Data Preprocessing](#_cj6lig18yjmg) 4

[2.2 Siamese Network](#_lop4ymvbh43t) 4

[2.2.1 Motivation](#_cy4x09uhqt0e) 4

[2.3 Transfer Learning](#_a6nhljduxivb) 6

[2.3.1 Resnet-50](#_kkrjdqcr56tx) 7

[2.3.2 SE-ResNet-50](#_kwth5mxw90mu) 9

[2.3.3 VGG-16](#_3andobz30d1w) 9

[2.3.4 EfficientNet-B4](#_rask6j91i0px) 10

[2.3.5 InceptionV3](#_x9u1empp8q46) 11

[2.3.6 FaceNet](#_rn3i53492c3) 13

[2.4 Ensemble Models](#_epgla4n04k70) 14

[2.4.1 Majority voting](#_utm04arytgyh) 14

[2.4.2 Simple Average](#_prdztynhc23m) 14

[2.4.3 Weighted average](#_g0y118mw0g7) 14

[2.4.4 Stacking](#_oz0zncgdlmwk) 15

[2.5 Parameters](#_sz4g4k7nf0t) 15

[**3. Results**](#_25653u9slkq3) **16**

[3.1 Summary of Kaggle leaderboard performance:](#_q07atabg6r01) 16

[3.2 Screenshots from Kaggle](#_w3g0w5vt74de) 18

[**4. Conclusion**](#_vyv5uy5rmwh6) **19**

[4.1 Discussion](#_xnffqb7rvxw6) 19

[4.2 Reflection and Future Improvements](#_uiqjvdhyrkoq) 22

[4.3 Contributions](#_8hcn7tlgni8x) 22

[**5. References**](#_id3bw7rutvus) **23**

# Introduction

## 1.1 Problem Statement

People who are biologically related to each other often share some sort of similarities in facial features, including the eyes and shapes of the face. However, it is often difficult even for the human eye to identify related people from images alone.

Furthermore, researchers have been faced with the challenge of the unavailability of a dataset that is large enough to be able to reflect the true distribution of families around the world. For this Kaggle competition, the dataset provided is the Families in the Wild (FIW) dataset, which is known to be the largest and most comprehensive database for kinship recognition tasks. It was made from over 11,000 family photos of 1,000 families, resulting in almost 657,000 image pairs, and 11 unique relationships (including father-daughter (F-D), father-son (F-S), brother-brother (B-B), grandfather-grandson (GF-GS) etc.). For our project, we experimented with different deep learning architectures and ensemble techniques to derive an optimal model for this binary classification task.

Through our research and experimentation, we were able to achieve a top accuracy score of 0.886 for kinship classification on pairs of face images. The establishment of computer vision based kinship recognition could be beneficial in practical applications such as automated digital photo album organisation, search of lost family members or even facilitate crime investigation such as human trafficking.

## 1.2 Related works

Many researchers have also tried traditional approaches for kinship recognition, including handcrafted feature extraction methods like the Local Binary Pattern (Ahonen et al., 2006) and metric learning methods like the Neighbourhood Repulsed Metric Learning (NRML) (Lu et al., 2014). Deep learning based approaches have also been an active research area. Dehghan et al., 2014, introduced an approach of fusing using gated auto-encoders while Zhang et al., 2015 used deep convolutional networks to extract high-level features for kinship verification.

## 1.3 Challenges

The main challenges faced were due to the difference in environments and package versions from those that were recommended for the pretrained models, which caused delays in executing the models.

# 2. Methodology

In this section, we will explain the data preprocessing and methods used. Specifically, the methods that will be elaborated on are the Siamese Network, Transfer Learning, and Ensemble models.

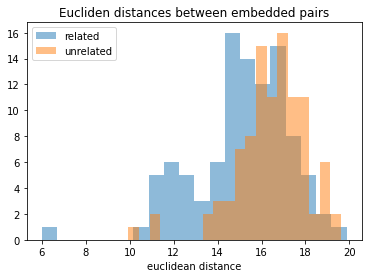
## 2.1 Data Preprocessing

The data provided contains information on pairs of images of related faces to be used for training. As part of preprocessing, we also created pairs of unrelated face images to be included in the training data. In each batch of 16 pairs, 8 of the pairs are related face images fetched from the training list provided (label = 1) while another 8 pairs are randomly sampled and checked for unrelatedness (label = 0). Additionally, certain pretrained models were trained on images of a certain size, hence performing better for that specific image size, as such the images had to be reshaped and resized into the relevant parameters.

## 2.2 Siamese Network

### 2.2.1 Motivation

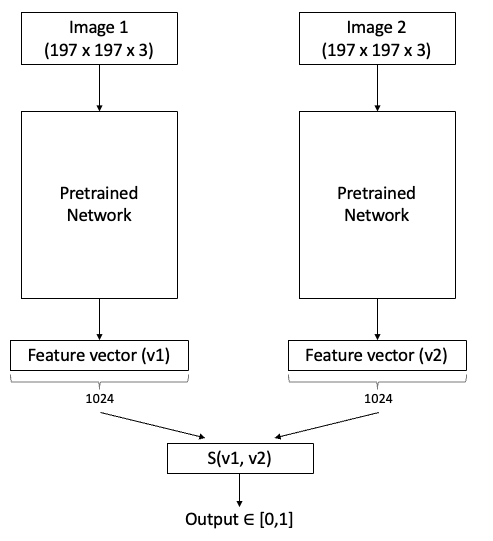
Since blood relatives share similar features, this suggests feature mappings of related faces may be closer to each other than unrelated people. Hence, we seek to explore the embeddings of face images and the similarities between them. This similarity could be represented in the form of a distance or loss function. During initial exploration, we sampled 100 pairs of images of related people and 100 pairs of images of unrelated people. We then passed each image into a pretrained FaceNet model to output an embedding vector. For each pair of feature/embedding vectors from images of related people, we calculated the Euclidean distance between them, which results in an array of 100 distance values in total. This was repeated for the image pairs of unrelated people. The distribution of the Euclidean distances of the two groups (related and unrelated) are plotted below.



*Figure 2.2.1.1. Histogram to visualise distributions of euclidean distances of related vs unrelated vectors*

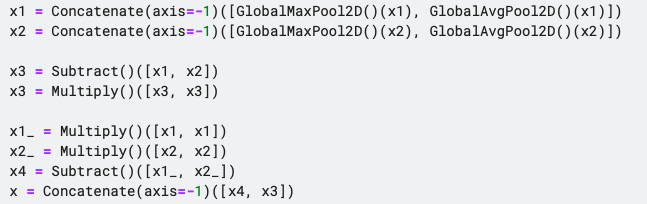
It was observed that the Euclidean distances of unrelated image pairs were generally larger than that of related pairs. This motivated us to use the feature vectors from pretrained models for this classification task. It is also evident that there is a significant overlap in the histograms of the two groups, which means that classification will be rather inaccurate if we distinguish kinship purely based on Euclidean distance of feature vectors. This led us to a hypothesis that perhaps learning a similarity function from the vectors would be more effective in capturing deeper relationships between the two vectors, compared to merely a simple Euclidean distance.

Therefore, to perform kinship classification on the pairs of images, we adopted a Siamese network consisting of 2 identical sub-networks. Each image in a pair is then fed into one of the two sub-networks. These two sub-networks output two feature vectors, to represent the two images. Then, a loss function is applied to learn the semantic similarity between the two inputs. At the end, a sigmoid layer is applied for binary classification.

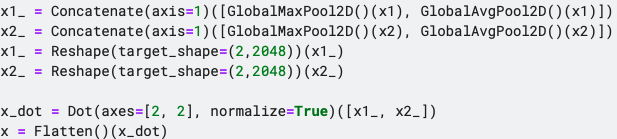


*Figure 2.2.1. Architecture of Siamese network used*

In order to learn the semantic similarity between the 2 feature vectors v1 and v2, we introduced several computational layers:



The operation x3 = (v1 - v2)2 computes the euclidean distance between the two feature vectors, while x4 = (v12 - v22) is another version of distance computation, similar to the manhattan distance but squared at each point.



We also experimented with replacing the above solution with a layer computing the dot product between the 2 feature vectors. The result of this experiment is documented as ResNet50-2 in the results table (section 3). It achieved a less optimal evaluation score of 0.797 as compared to the previous score of 0.866. Hence we decided to stick with the previous layers of (v1 - v2)2 and (v12 - v22) instead of the dot product.

## 2.3 Transfer Learning

Since the dataset is relatively small, with about 12379 images of 2316 people, in 3362 relationships, training could lead to overfitting, where the model has low error on the training data but higher error on the test data, as the model may learn certain dependencies that are not generalisable to other data points. Apart from fine tuning the model with a validation dataset and including dropouts and regularizations, one of the ways to improve the reliability of the model is to train the model on more data, such as “generating” more data by augmenting the current data, or using similar data from other datasets. However, due to time constraints, it may not seem feasible to label and train a model on another dataset. Fortunately, we can exploit the benefits of transfer learning. Transfer learning is a powerful solution that allows us to transfer learned deep layer representations from pretrained models and use these representations for our target task, which would help us achieve much faster convergence rate and accuracy compared to training from scratch. The source tasks would be image classification, specifically face recognition and object recognition, while our target domain is kinship classification in this case. Hence, the most relevant pretrained embeddings in this project that we could build upon would be feature mappings of other face or object datasets. Thus, the networks we have chosen were pre-trained on 3 image datasets:

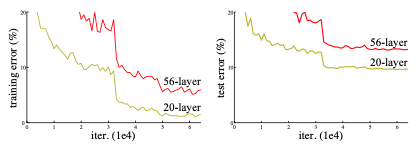
* ImageNet: the ImageNet is a visual database of 14 million manually labelled images used for multiclass object classification. The full dataset contains 21,841 classes, including ‘balloon’, ‘strawberry’ etc.
* VggFace2: the VggFace2 is a set of 3.3 million face images and 9000+ unique persons of a wide range of ethnicity and ages.
* Labelled Faces in the wild (LFW): the LFW is a database of face photographs detected by the Viola-Jones face detector, designed for studying the problem of unconstrained face recognition. The dataset contains more than 13,000 images of faces, labelled with the person’s name. 1680 of the people pictured have two or more distinct photos in the data set.

For image classification and recognition tasks, CNNs were designed for this purpose and are commonly used as it is able to capture higher-level representations in the images’ raw pixel data content. In the following sections, we will explain the architecture and background of the various models used for our submissions.

### 2.3.1 Resnet-50

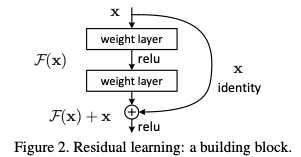
The Resnet-50 is a 50 layer convolutional network, consisting of 48 convolutional layers, 1 max pool layer, and 1 average pooling layer. Being a type of residual network, it is known to be particularly effective compared to the traditional complex CNN models (such as VGG-16) in mitigating the problem of vanishing gradient and degradation. The development of VGG models (Simonyan et al., 2014), has proven that steadily increasing the depth in convolutional networks (while keeping other parameters fixed) yields substantial benefits for large scale image classification tasks. The VGG architecture has since become the basis of ground-breaking image / object recognition models.

However, a deep architecture creates the notorious problem of vanishing gradients, where the value of gradient decreases exponentially as it is propagated to the initial layers during back propagation. In other words, the parameters of earlier layers will not be upgraded effectively, hampering the model from converging.

*Figure 1. Training (left) and test (right) error of 20-layer and 56-layer networks*

While the problem of vanishing gradient could mostly be solved by normalisation layers, the degradation problem arised. This problem was observed in deep neural networks, whereby the accuracy of models becomes ‘saturated’ as the number of layers increases, then degrades rapidly. Theoretically, we would expect that a deeper model (n + m layers) would perform at least as well as its shallower counterpart (n layers), since the deeper model is expected to at least learn the first n layers in the same way as the shallower network, then learn the remaining m layers as an identity mapping. In practice, however, both training and test error were shown to increase when more layers are added to an already sufficiently deep model.

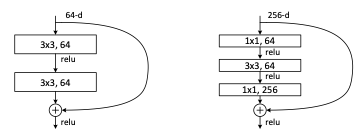
This problem of degradation was due to a loss of information through layers. It hence inspired the development of the deep residual network (He et al., 2016), which contains ‘shortcut connections’ (simply identity mapping) to skip over one or more layers, then recast it together with the stacked non linear layers (F(x) + x). These shortcuts do not incur any additional complexities to the model (since no extra parameters).



*Figure 2. Residual learning: a building block*

This novel idea allows networks to continue to yield gains in accuracy by increasing its layers, while not being affected by the high training error. It is also similar to the cell state (memory belt) in LSTMs (Long short term memory) to address the issue of vanishing gradient in recurrent neural networks.

On top of this, the ResNet-50 replaced the original 2 layer blocks in the original 34 layer architecture with 3 layer blocks. In addition, pointwise convolution layers (1x1 kernels) were introduced to reduce the time complexity of model training (similar to depthwise separable convolution). This is known as the ‘bottleneck’ block.



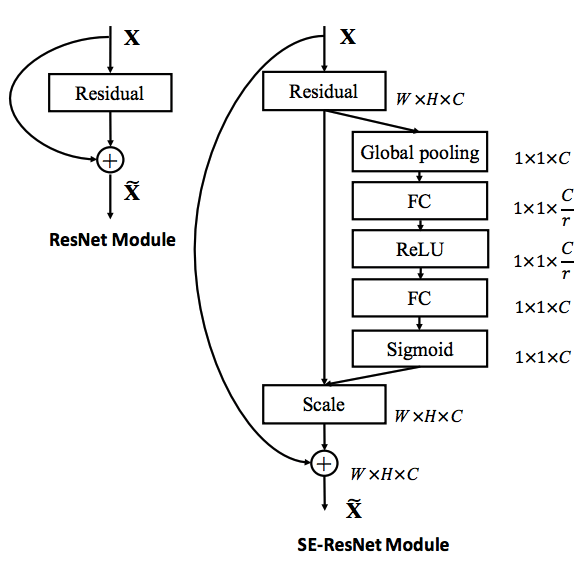
*Figure 3. Original block in ResNet-34 (left) and ‘bottleneck’ block in ResNet-50 (right)*

The ResNet-50 model we used was pretrained on VGGFACE2 (Cao et al., 2017) and the feature extraction layers were loaded (using VGGFace(include\_top=False)).

### 2.3.2 SE-ResNet-50

A Squeeze-and-Excitation (SE) block adaptively recalibrating channel-wise features by modelling interdependencies between channels. We used a SE-ResNet-50 model that was pre-trained on the VGGFace dataset. SE blocks improve channel interdependencies at almost no computational cost, and are designed to improve the model’s representation power of a network by allowing it to perform dynamic channel-wise feature calibration. With a convolutional block as input, each channel is first “squeezed” into a single number using global average pooling. Then, output channel complexity is reduced and non=linearity is introduced because of the Fully Connected layer and ReLU. Another dense layer with sigmoid function results in each channel having a smooth gating function.

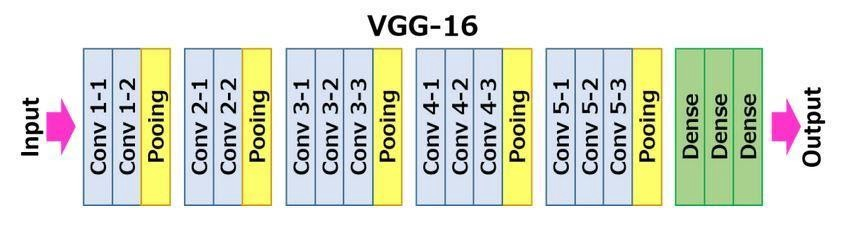
The architectures before and after adding the SE block are shown in Figure 2.3.2.1 below.



*Figure 2.3.2.1 ResNet vs SE-ResNet Model Architectures*

### 2.3.3 VGG-16

VGG-16 is a type of CNN that is unique for focusing on convolution layers of 3x3 filters instead of having a large number of hyperparameters. The architecture involves stacks of multiple convolution layers of size 3x3 with stride 1 and padding, followed by a max-pooling layer of size 2x2 (Figure 2.3.3.1). Different configurations of this stack are then repeated in the network configurations. At the end, 3 FC (Fully-Connected) layers are added followed by a softmax for output. Even though there are 21 layers in total, there are 16 layers with learnable weights since there are 5 layers of max-pooling in total. All the hidden layers use the ReLU activation function as it results in faster learning, decreases the likelihood of vanishing gradient problems, and is more computationally efficient.

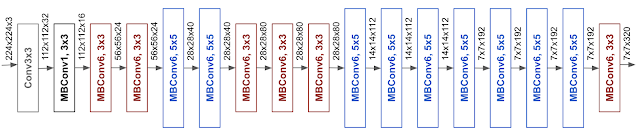


*Figure 2.3.3.1 VGG-16 Architecture*

VGG-16 is a powerful architecture for object localisation and classification tasks. However, it is very slow to train, and due to the large size with 138 million parameters, it requires a significant amount of space and bandwidth, and leads to exploding gradients.

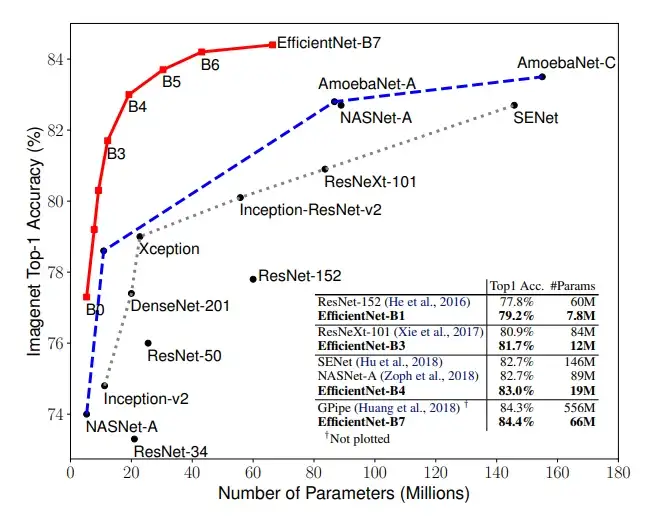
### 2.3.4 EfficientNet-B4

EfficientNet is a family of models that make use of compound model scaling to achieve better accuracy and efficiency. The traditional approach of model scaling for CNNs is to increase the depth or width, or even use larger input image resolution for training and testing. Although these methods improve accuracy, they tend to require long training periods and a lot of manual tuning. EfficientNets aim to solve the issue of randomly scaling models by proposing a compound coefficient to scale CNNs in a more structured manner. The figure below shows the architecture of the simplest EfficientNet-B0 model, upon which all the other EfficientNet models were scaled using different compound coefficients.



*Figure 2.3.4.1 EfficientNet-B0 Architecture*

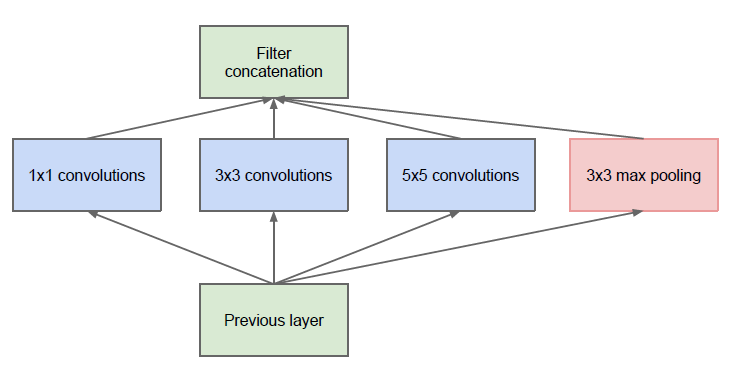
EfficientNet models are pre-trained on ImageNet and in general, they have higher accuracy and better efficiency than existing CNNs, despite fewer parameters and number of FLOPs. We decided to use EfficientNet-B4 as it is at the point where the gradient of the curve started to decrease in the figure below, indicating that the increase in accuracy was no longer proportional to the increase in parameters.

​​

*Figure 2.3.4.2 Model Size vs ImageNet accuracy*

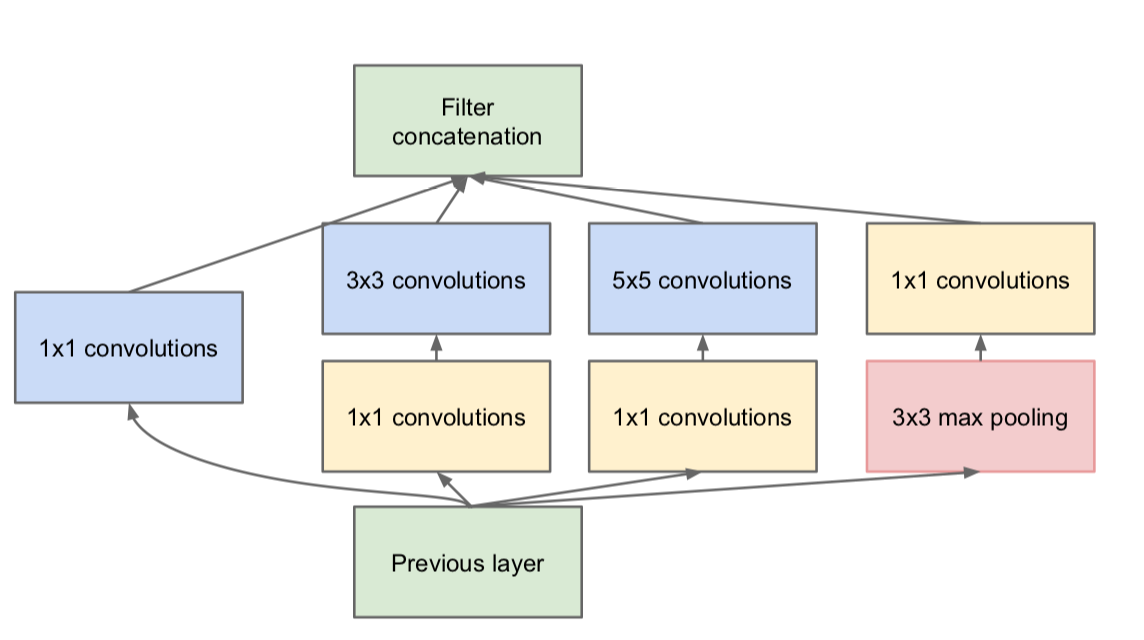
### 2.3.5 InceptionV3

Inception is a family of convolutional neural network architectures used for object classification and detection. The Inception network architecture consists of several inception modules of the following structure, with four operations in parallel, a 1x1 convolution layer, 3x3 convolution layer, 5x5 convolution layer and a max-pooling layer.



*Figure 2.3.5.1 Naive Inception-v1 module*

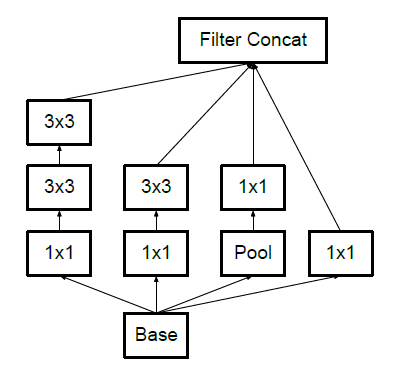
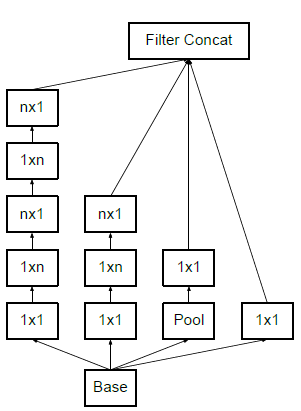
Since the 5×5 convolutional layer in the naive form is computationally expensive and time-consuming, the authors added a 1×1 convolutional layer for each convolutional layer, leading to reduced dimensions and faster computations.



*Figure 2.1.5.2 Inception-v1 module*

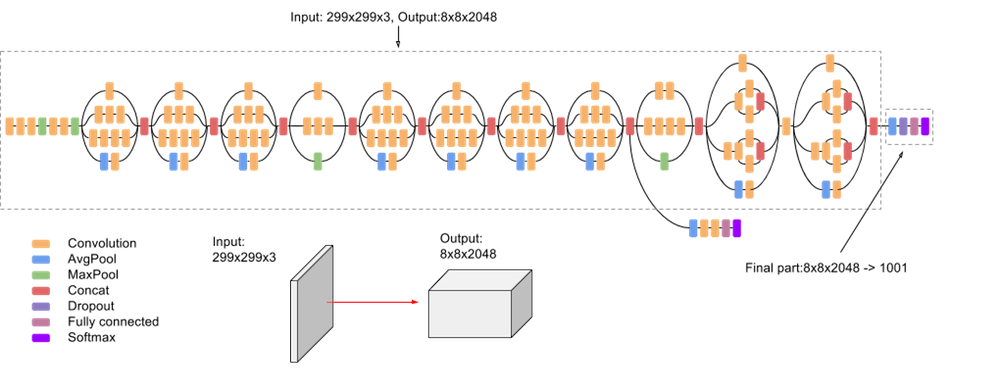
The InceptionV3 is an improved and optimised version of the InceptionV1 model, by using several techniques for optimising the network for better model adaptation.

In InceptionV3, the 5x5 convolutional layer was replaced by 2 3x3 convolutional layers. This factorization of larger convolutions into smaller convolutions helps reduce the parameters and computational costs. Furthermore, these 3x3 convolutions can be made more efficient using asymmetric convolutions, of the form n×1. Hence, combining these, the 3×3 convolutions were replaced with a 1×3 convolutional layer followed by a 3×1 convolution.

 ​​

| *Figure 2.3.5.3 Factorization into smaller convolutions* | *Figure 2.3.5.4 Asymmetric convolutions* |
| --- | --- |

At the end, an Auxiliary classifier was used to improve the convergence of deep neural networks. It is mainly used to address the vanishing gradient problem in deep networks. Hence, the auxiliary classifiers act as a regularizer in the InceptionV3 architecture. To reduce the grid size efficiently, the activation dimension of the network filters is expanded, and convolution and pooling is done in parallel before being concatenated.



*Figure 2.3.5.5 InceptionV3 model*

### 2.3.6 FaceNet

FaceNet is a state-of-the-art model developed by Google researchers in 2015 (Schroff et al., 2015). The network is trained such that the euclidean distance between embeddings directly corresponds to the similarity between two images. The greatest benefit of FaceNet for our case is that it enables a much more compact embedding representation of images compared to other deep convolutional networks. The resulting embedding is 128 dimensions compared to other networks such as ResNet, which yields >1000 dimensions.



*Figure 2.3.6.1. Brief model structure of FaceNet*

In short, triplets are generated consisting of 2 matching faces and one non matching face, and the triplet loss is used with the aim of separating the positive pair from the negative pair.



*Figure 2.3.6.2. Learning process where the triplet loss is used to minimise distance between positive pairs and maximise distance between negative pairs.*

Once the model is trained, the 128 dimensional vector (in euclidean space) can be used for clustering, KNN classification etc. In our case, we will use the vectors to learn a similarity function between pairs of faces.

## 2.4 Ensemble Models

Since our base models are obtained from various pretrained models, we expect that they are independent and would misclassify different kinds of training examples. Hence, we hypothesised that an ensemble classifier would provide better predictive performance compared to any individual model. To construct our ensemble models, we chose a cut off score of 0.7 and selected 5 base classifiers (see results section): SeNet, ResNet50, ResNet50-2, FaceNet and InceptionResNetV2 to construct our ensemble models. We will experiment with and evaluate a range of ensemble techniques, ranging from simple average to training a meta classifier using our best performing models.

### 2.4.1 Majority voting

The majority voting (hard voting) is the simplest ensemble technique whereby each base classifier votes 0/1 for class, and the mode of all individual votes will be assigned as the class label. This method yielded lower scores than expected (private score of 0.775 and public score of 0.77). This is perhaps attributed to the prediction probability being converted to discrete classes of 0s and 1s, which caused the evaluation score on kaggle to be low.

### 2.4.2 Simple Average

The simple average is an ensemble technique whereby the contributions of each base classifier is equal (regardless of individual accuracy). Compared to majority voting, this is a soft voting technique where the probability value is used in place of 0 or 1.

This ensemble technique yielded rather decent results, with a private score of 0.885 and public score of 0.872.

### 2.4.3 Weighted average

The weighted average ensemble technique is an ensemble machine learning technique that combines predictions from different base models, and the amount of contribution of each base model is proportional to its evaluation score. In general, base models with a higher score will have a greater impact on the final prediction. This way, we hope to develop a weighted function that produces better results than the best base model. For instance, in the case when the best base model misclassifies a certain instance, the results from the other base classifiers could hopefully correct the final outcome.

In our implementation, we adopt a soft voting classifier with the probability output of each model weighted by its normalised AUC score.

, where

Unsurprisingly, the weighted average gave us the best performance out of all models so far, with a private score of 0.886 and public score of 0.874. This is slightly better than the results from the simple average ensemble.

### 2.4.4 Stacking

### 

The stacking ensemble method involves training a meta classifier to learn to make predictions based on the predictions of individual models. In our meta learner, we simply concatenated the predictions of the 5 models and introduced a dense layer and a softmax layer to allow the meta classifier to learn relationships between the 5 predictions.

The meta classifier achieved a private score of 0.880 and a public score of 0.863.

#### 

## 2.5 Parameters

Adam optimization algorithm was used. Optimization algorithms first calculate the gradient, and move along the direction to the minima of the function. The learning rate, the step size that is incremented in the search space, was set at 0.00001. A callback function was used to reduce learning rate on plateau, and reduce the learning rate once the accuracy stopped improving. The patience was set to 20 epochs, after which the learning rate would be reduced if there was no improvement. For some of the models, to save time, early stopping was introduced to stop training once the loss has stopped improving.

# 3. Results

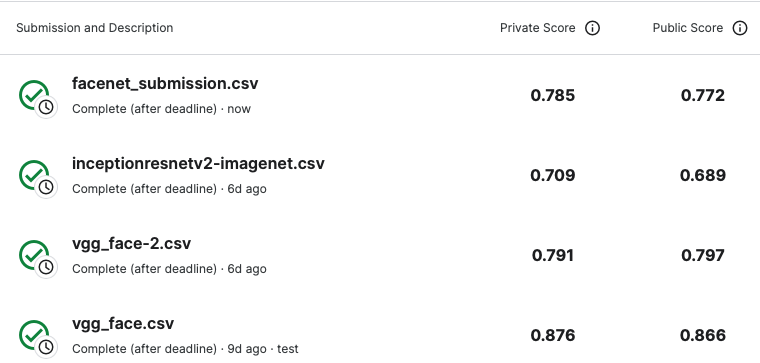
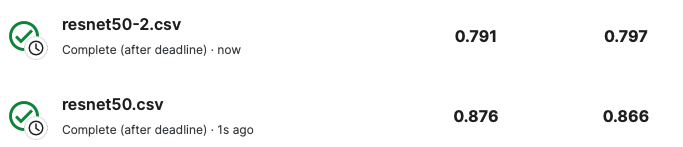
## 3.1 Summary of Kaggle leaderboard performance:

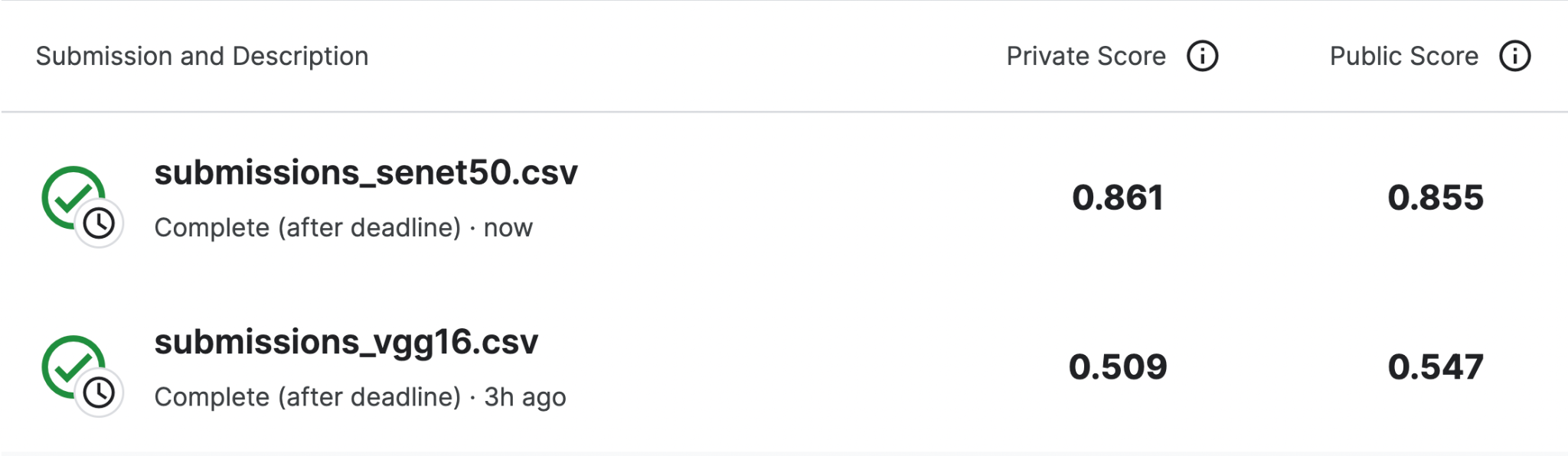
Legend:

|  | Top 20% |
| --- | --- |
|  | Top 25% |
|  | Top 30% |

| **Dataset Pretrained on** | **Model Submitted** | **Private Score** | **Public Score** | **Rank** |
| --- | --- | --- | --- | --- |
| VGGFace | **ResNet-50** | 0.876 | 0.866 | 139 (private)  (Top 25%) |
| 154 (public)  (Top 30%) |
| SENet-50 | 0.861 | 0.855 | 209 (private) |
| 210 (public) |
| ResNet50-2 | 0.791 | 0.797 | 291 (private) |
| 284 (public) |
| VGG-16 | 0.509 | 0.547 | 488 (private) |
| 472 (public) |
|  |  |  |  |  |
| ImageNet | InceptionResNetV2 | 0.709 | 0.689 | 410 (private) |
| 415 (public) |
| InceptionV3 | 0.633 | 0.638 | 440 (private) |
| 434 (public) |
| EfficientNet-B4 | 0.489 | 0.507 | 522 (private) |
| 494 (public) |
|  |  |  |  |  |
| Labelled Faces in the wild (LFW) | FaceNet | 0.785 | 0.772 | 294 (private) |
| 306 (public) |
|  |  |  |  |  |
| – | Ensemble  (Majority Voting) | 0.775 | 0.77 | 304 (private) |
| 306 (public) |
| **Ensemble**  **(Simple Average)** | 0.885 | 0.872 | 115 (private)  (Top 20%) |
| 133 (public)  (Top 25%) |
| **Ensemble (Weighted Average)** | 0.886 | 0.874 | 112 (private)  (Top 20%) |
| 132 (public)  (Top 25%) |
| **Ensemble (Stacking)** | 0.880 | 0.863 | 127 (private)  (Top 25%) |
| 167 (public)  (Top 30%) |

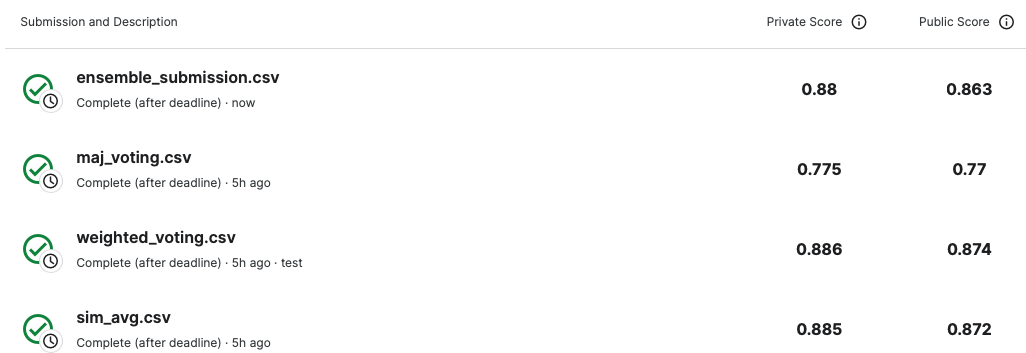
## 3.2 Screenshots from Kaggle







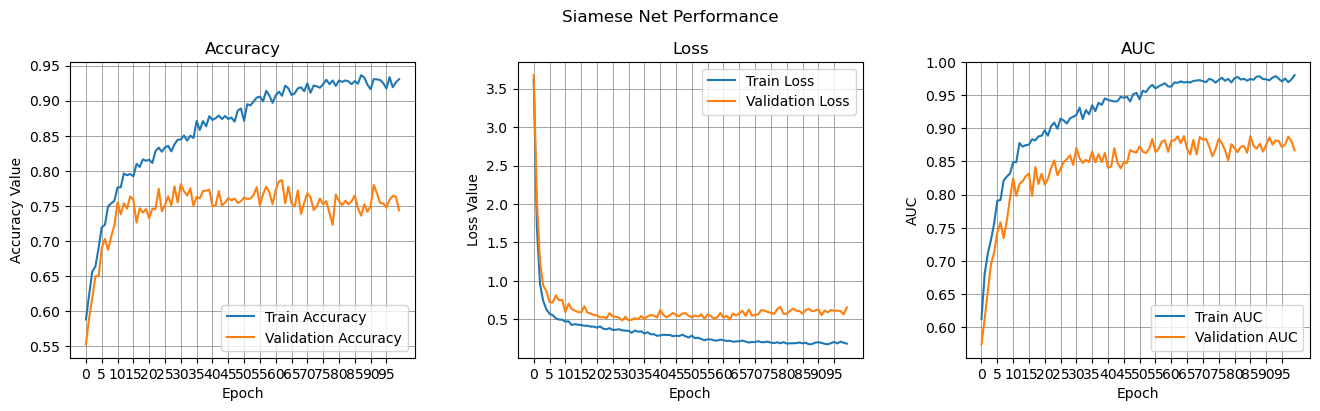




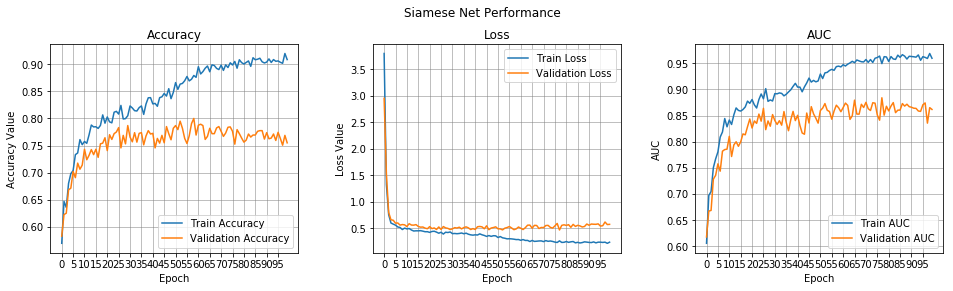
# 4. Conclusion

## 4.1 Discussion

Overall, ResNet, SENet, InceptionResNetV2 and FaceNet performed well on the Siamese Network with this dataset.



*Figure 4.1.1 ResNet50*



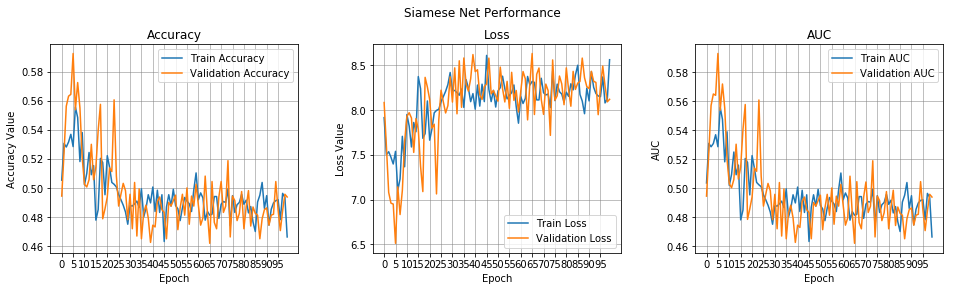
*Figure 4.1.2 SENet50*

However, there seems to have been some overfitting occurring in the ResNet50 and SENet50 models even though dropout layers and regularizations were already introduced to mitigate significant overfitting. From the plots, the training accuracy was higher than validation accuracy, and training loss was lower than the validation loss. Perhaps this overfitting could be due to the relatively small dataset and large network sizes of around 25 million parameters, leading to the model learning weights that fit too well to the data points but are not generalizable to the test dataset.

Even though FaceNet was trained on the LFW dataset which is relatively small compared to the VGGFace and ImageNet datasets, it may have performed well since the architecture is similar to a Siamese Network, taking in a few image inputs and calculating a loss function between them.

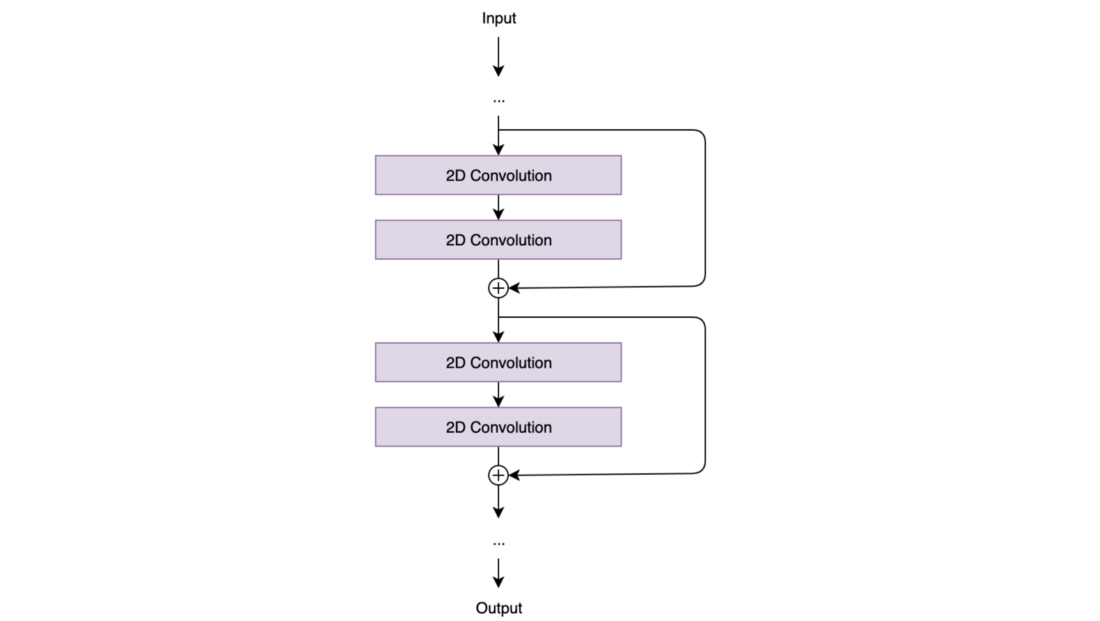
Models that were pre-trained on ImageNet did not do as well, likely since the dataset is more varied and used for object detection compared to VggFace2 and LFW datasets which only contain faces. Pretrained models on face datasets may be more likely to learn mappings that are unique to facial features, that could be useful for transfer learning in kinship recognition.

However, even though VGG-16 was also pre-trained on VGGFace, it did not perform well in this task.



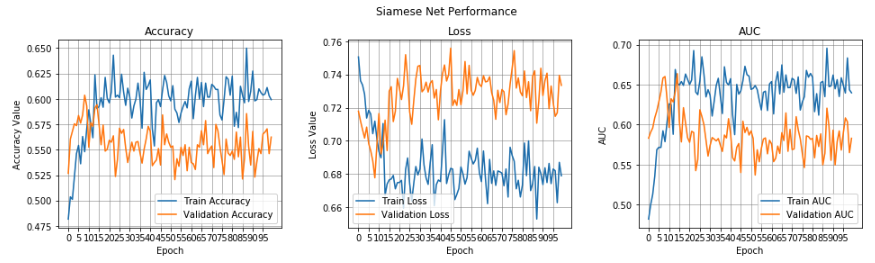
*Figure 4.1.3 VGG-16*

From the plots, in fact the accuracy actually decreases and the loss increases while fluctuating with the number of epochs. This observation could be explained by the fact that VGG-16 is susceptible to vanishing gradients. In deep networks, vanishing gradients could have a significant impact on performance. During back propagation, the weights will not be updated as the derivative vanishes, hence the network cannot learn and the performance decreases. However, ResNet is able to prevent vanishing gradients due to the skip connections that it has in its architecture as shown in Figure 4.1.4. These skip connections allow backpropagation to occur without the problem of vanishing gradients occurring, which could also be a reason why ResNet-50 performed better.



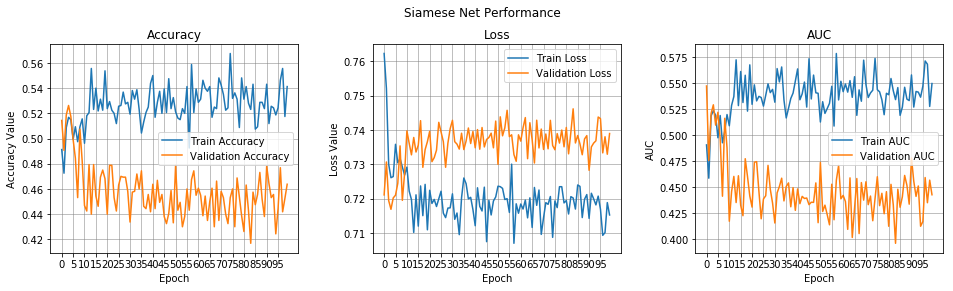
*Figure 4.1.4 Architecture of ResNet showing skip connections*

Similarly, the InceptionResNet-v2 model builds upon the Inception architecture by incorporating residual connections from the ResNet framework, allowing for gains in accuracy by increasing its layers, while not being affected by the high training error in deep models. This could explain why InceptionResNet-v2 outperformed InceptionV3 and EfficientNet-B4 even though all 3 were pretrained on the same dataset.



*Figure 4.1.5 InceptionV3*

Interestingly, EfficientNet-B4 had a private score of 0.489, and validation accuracy around 0.46, which is worse than the probability of randomly assigning 0 and 1 labels (random probability = 0.5). The training and validation accuracies are diverging, suggesting the model was adjusting to noise, and valuing the "wrong" features.



*Figure 4.1.6 EfficientNet-B4*

## 4.2 Reflection and Future Improvements

As our models were pre-trained on different datasets such as VGGFace, ImageNet and Labelled Faces in the Wild, the models would have differed in terms of their ability to distinguish between human faces. Specifically, we observed that models pre-trained on VGGFace and Labelled Faces in the Wild did better than those that were pre-trained on ImageNet. As these datasets have millions of images, we could not ensure that all models were pretrained on the same dataset, but a future improvement would be to pretrain all models on the same dataset (perhaps on a combination of both VGGFace and Labelled Faces in the Wild datasets, or even another dataset that has sufficient facial image data), and compare the performance again. Additionally, pretraining the same model on different datasets or even combining the features together could generate more independent models that could be used in ensemble model.

Apart from pretraining on different datasets, from the competition discussions page, it was suggested that different combinations in combining the 2 subnetworks (e.g. x1+x2 and x1-x2 or sqrt(x1) + sqrt(x2) and sqrt(x1) - sqrt(x2)) could be explored to generate the loss function in the Siamese network that would still result in an independent model to be used in the ensemble model. Perhaps with more good models, the ensemble accuracy could increase.

Perhaps we could also introduce data augmentation into preprocessing, such that minor alterations (transformations / rotations / minor changes in colour / introducing noise) are made to each image before training. While an increase in training instances would increase total computation time, it might be able to improve the ability of our models to generalise to a wide range of input images. It might also reduce overfitting of the models to an extent.

Another improvement would be to add more layers to the pre-trained model since feature recognition is a rather complex task and requires subtle features to be learnt by the model, and also further tune the hyperparameters such as the learning rate, number of epochs and type of optimiser used, to determine the optimal hyperparameters for our model.

## 4.3 Contributions

| **Tasks** | **Done by** |
| --- | --- |
| Model implementation and execution | Joelle, Grace, Anyi |
| Report | Joelle, Grace, Anyi |
| Video | Joelle, Grace, Anyi |

# 5. References

Ahonen, T., Hadid, A., & Pietikainen, M. (2006). Face description with local binary patterns: application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* *28*(12), 2037–2041.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFACE2: A dataset for recognising faces across pose and age. *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. https://doi.org/10.1109/fg.2018.00020

Dehghan, A., Ortiz, E. G., Villegas, R., & Shah, M. (2014). Who do I look like? determining parent-offspring resemblance via gated autoencoders. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2014.227

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-Excitation Networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

J. Lu, X. Zhou, Y. -P. Tan, Y. Shang and J. Zhou, "Neighborhood Repulsed Metric Learning for Kinship Verification," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 2, pp. 331-345, Feb. 2014, doi: 10.1109/TPAMI.2013.134.

Mingxing Tan, & Quoc V. Le. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. International Conference on Machine Learning, 6105–6114.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2015.7298682

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

Zhang, K., Huang, Y., Song, C., Wu, H., & Wang, L. (2015). Kinship verification with deep convolutional neural networks. *Procedings of the British Machine Vision Conference 2015*. https://doi.org/10.5244/c.29.148