CS271P Sponsored Project

Daniel Swartz Qi Song Zhongyao Qian

December 2021

Abstract

Within the last decade, AI techniques have become particularly useful in assisting with criminal justice needs. One of the most popular AI techniques is facial recognition, which can help establish an individual's identity and trace their relationships through photos. In this project, a CNN model is trained with the Families in the Wild dataset to analyze the paired images of portraits. The resulting models can accurately classify the kinship between two blood-related persons.

1 Introduction

In a wide variety of fields, from genealogy to criminal investigations, it is crucial to determine how people are related to each other. Advancements in computer vision and Convolutional Neural Networks (CNNs) allow us to accelerate the development of creating and documenting relationships between individuals, solely based on images of their faces. Employing AI greatly reduces the intrusiveness of making familial ties during an investigation and the manhours required to infer these connections. We believe that knowing exactly how individuals in a family are related is important, so we implement and fine-tune a CNN model to determine the specific familial ties between portrait images of individuals within the same family.

The following paper is organized into sections. Section 2 covers the dataset and how it is used. Section 3 details how the baseline model and ablation studies are constructed and how training and validation are conducted. Section 4 contains the results and analysis regarding these findings. Lastly, Section 5 concludes the overall project.

The Jupyter Notebook used for this project is saved as "cs271p.ipynb" in the root directory of the Google Drive. drive.google.com/drive/folders/19X19TdBt0zFSAhtdIaFuPOIWLbvuG1A-?usp=sharing

2 Data

The dataset used is the Families in the Wild [1] image database, which is structured into three parts: training data, validation data, and test data. The training and validation data is structured in folders of families, numbering 1018

(a) Person one. (b) Person two.

Figure 1: Examples of the types of portraits found within the Families in the Wild dataset.

different groups. Each folder contains sub-folders for each family member, containing one or more cropped images of the face of each individual [Figure 1]. The photos for each member can vary in the age of that person. For example, there can be images from decades ago and from within the last 5 years for a given person. The family folders also contain a comma-separated value (CSV) list of the relationships between each member, each member's name, and their gender. Additionally, there are two separate CSV tables for the training and validation data that contain the information detailing the family number, the two member IDs being matched, what their relationship tie is, and how many sample images there are for each person. The relationships consist of eleven different class labels, bb (brother-brother), fd (father-daughter), fs (father-son), gfgd (grandfather-granddaughter), gfgs (grandfather-grandson), gmgd (grandmother-granddaughter), gmgs (grandmother-grandson), md (mother-daughter), ms (mother-son), ss (sister-sister), and sibs (brother-sister).

Upon further studying of the dataset, we realized that this set is imbalanced in terms of the types of relationship pairs. The percentages of each combination in the training data are illustrated in Figure 2, with the highest percentage being ms at 24.5% of the total relationships and the lowest being gmgd with 0.9%. Additionally, the validation data consists of a smaller dataset but is also imbalanced. The highest percentage in the validation pairs is fs at 18.7%, and the lowest percentages are gmgs and gmgd at 1.4% [Figure 3].

The images for both the training and validation data are preprocessed by loading them as a JPEG image, converting the image to grayscale, and resizing the image to 108 by 124 pixels. This function serves to standardize the images, as the original images come in different sizes and have different hues and lighting. Following the preprocessing, two new datasets, one for training and one for validation, are created. Both datasets consist of concatenated pairings between an image of the first person and second person that are to be paired, along with the label of their relationship. This creates a two-channel tensor with the shape of (128, 108, 124, 2). The batch size was selected to be 128. The order of the pairs is then shuffled to ensure that the model does not learn the order of the images and overfit to a certain pattern.

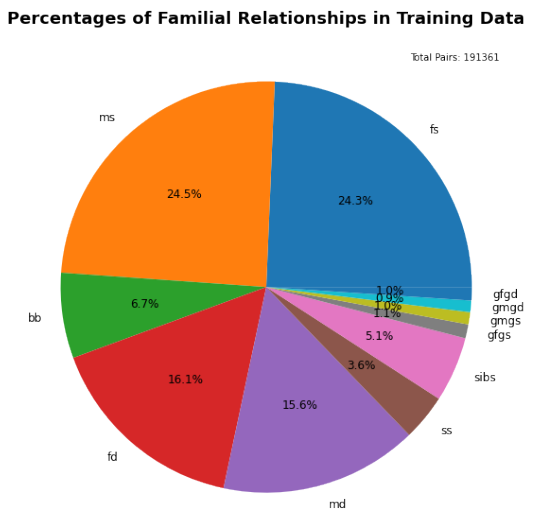


Figure 2: Chart of the percentages for each pair in the training data.

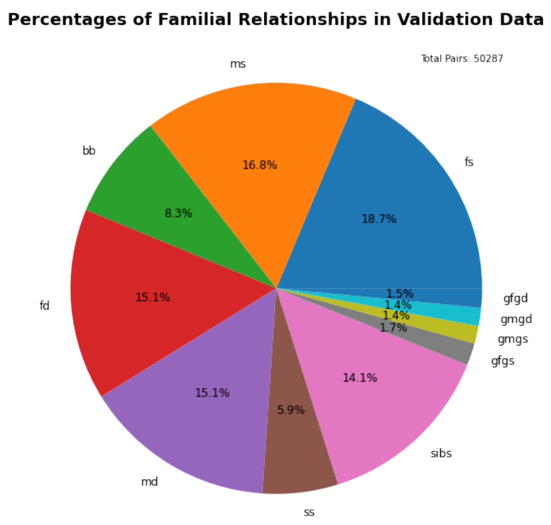


Figure 3: Chart of the percentages for each pair in the validation data.

3 Models

Baseline

For the baseline model, we decided on using a Convolutional Neural Network (ConvNet) with an EfficientNet architecture, specifically EfficientNetB0. The EfficientNet family boasts both higher accuracy and efficiency than most other ConvNets [2]. We perform transfer learning with a model pre-trained on the ImageNet dataset [3] to accelerate training and improve the model performance. The weights from the pre-trained model are applied to our model, but are not frozen and are subject to change during the training process. This is due to ImageNet having little connection to facial image classification.

The training procedure is outlined in the following steps. On the top layer, we implement global average pooling to average each feature map and feed it directly to the softmax layer [4]. In addition, a dropout is also added as a regularizer to prevent overfitting. The model is compiled using stochastic gradient descent with its learning rate optimized by the Adam algorithm [5] and sparse categorical cross entropy to calculate the loss. We train the model using up to 1000 epochs, but an early stop callback discontinues the training when validation accuracy doesn't reach its recorded maximum. In addition, a model checkpoint callback is used to save the best training progress.

Weighted

Our first ablation study attempts to correct the imbalanced dataset by implementing class weights. We calculate the weights for each class by dividing the total number of pairs by the number of pairs of that class, then divide that by the number of classes. The weights are applied to all of the classes, adding importance to the under-represented classes. The model is then trained using these new class weights, following the same procedure as the baseline model.

Oversampling

Another approach to address the imbalanced dataset was to oversample the minority classes. The oversampling process involves randomly sampling image pairs for each class to make the number of total pairs per class equal to the number of mother-son pairs at 46,969, so all of the other classes are then sampled to have that same number of pairs. This new dataset of classes of equal size is then used as the training data and the model is trained following the same procedure as the baseline model.

Augmentation

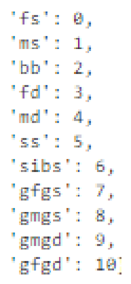
We attempt another ablation study through the augmentation of images within the oversampled training dataset. The data is randomly selected to be augmented and the changes include flipping images horizontally and slightly rotating them by $0.06(2\pi)$ radians. Applying this method serves the purpose of providing more data for the model to train on. We then train the augmented model using the same process as the baseline model.

Siamese Network

Lastly, we try an ablation study using a Siamese network. This architecture contains two embedded models with the same configuration for parameters and weights [6]. The oversampled dataset is preprocessed similarly to the previous models, but instead of taking two-channel tensors on one model, it takes one-channel tensors on each of the embedded models. The outputs from both of these embedded models are concatenated together, which then is further trained using a similar procedure to the baseline model.

4 Results

The Results section is structured by displaying plots of the accuracy and loss for both training and validation data as sub-figure (a) in the pair of figures. The accuracy is in decimal format, not percent. In sub-figure (b) the confusion matrix and corresponding heat map for the validation fitting are visualized. The confusion matrix displays the accuracy for each class of pairs, where each row represents instances of the actual classes, from 0 to 11 corresponding to the types of pairs possible [Figure 4]. Each column represents the instances of the predicted class, also from 0 to 11 with the same correspondence [Figure 4]. The diagonals, starting from top left to bottom right, represent the number of times a class was properly fit, as the column number equals the row number. When the row number does not equal the column number, that shows the number of times that incorrect fitting was made. The heat map below the matrix visualizes this confusion matrix, with the colors representing the percent over the total amount of pairs for that actual class.

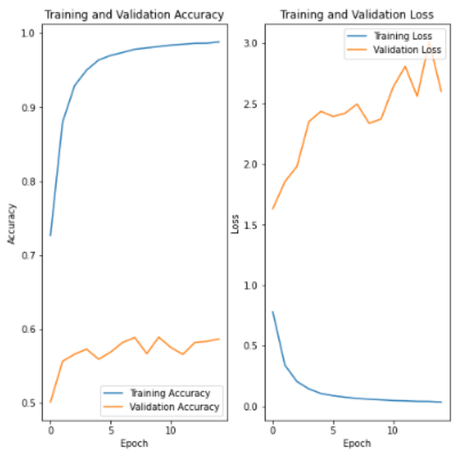


```
'fs': 0,
'ms': 1,
'bb': 2,
'fd': 3,
'md': 4,
'ss': 5,
'sibs': 6,
'gfgs': 7,
'gmgs': 8,
'gmgd': 9,
'gfgd': 10}
```

Figure 4: The classification names for each type of pair, along with the number that corresponds to the type of pair for labeling.

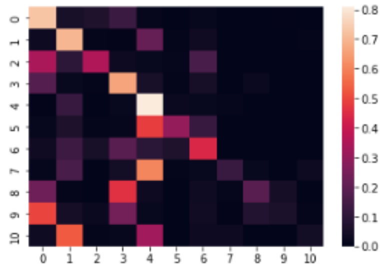
Baseline

The results of the Baseline model are shown in Figure 5. The trained model is saved as "baseline.h5" in the "/Model" directory of the Google Drive.



(a) Accuracy and Loss.

```
[[6800  493  616 1163   94   10  148    6   14   44   11]
 [ 230 5887   65   36 1802   65  290    9    0    2   55]
 [1419  397 1456  113   74   49  653    0    0    2    7]
 [1341  120   56 5015  390   61  381   12  177   16    6]
 [  30  902   15  137 6131  156  125   79    5    1    6]
 [  41  188   21   34 1438  902  357    9    1    0    0]
 [ 236  904  377 1321  647  452 3122    3    4    1    1]
 [  10  137    0   36  530    6   15  106   16    0   23]
 [ 160    6    2  322   18    1   22    8  129   32    1]
 [ 352   32   16  170   13    0   27    3   54   41    6]
 [  22  403    5   11  246    0   21   20    2    1   31]],
```



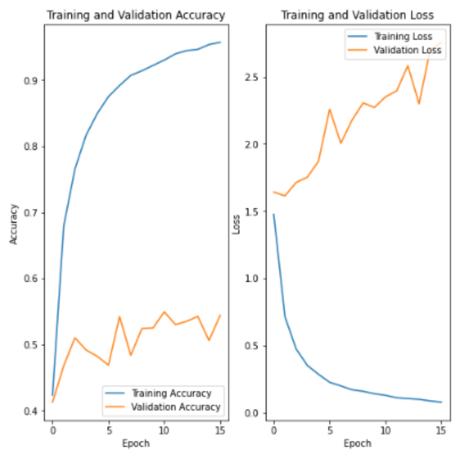
(b) Confusion matrix and heat map.

Figure 5: Results for the Baseline model.

For the Baseline model, the training accuracy steadily improved as Epochs progressed, starting at 73% and maximizing at about 99%, with loss decreasing from approximately 0.78 to about 0.03. Validation accuracy increased from 50% to around 59%, with loss also increasing nearly linearly as the epochs progressed, from 1.6 to 2.6. The confusion matrix and heat map show a trend, with the lower numbered classes (0-6) having better fit results than the higher number classes (7-11). The lower number classes correspond to first-degree relationship pairs, the parents and siblings, while the higher number classes represent the second-degree pairs, the grandparent to grandchild pairs.

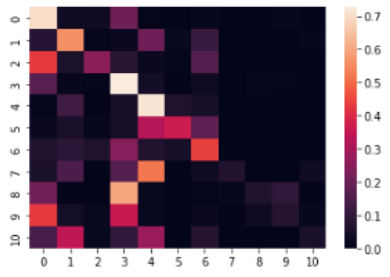
Weighted

The results of the Weighted model are shown in Figure 6. The trained model is saved as ”weighted.h5” in the ”/Model” directory of the Google Drive.



(a) Accuracy and Loss.

```
[[6571  254  281 1911   74   11  128    1    9  132   27]
 [ 621 4706   67  152 1741  128  932    9    4    7   74]
 [1785  216 1045  317   81   45  653    0    5    9   14]
 [1271  101   63 5509  271   28  194    2   80   50    6]
 [  83  916   29  334 5420  435  328   17    1    0   24]
 [  64  150   37  122  971 1094  539    1    7    0    6]
 [ 452  573  405 1736  453  308 3104    6   14    6   11]
 [  43  128    2  141  457    3   23   55    2    1   24]
 [ 149    2    3  420   11    0    4    9   40   62    1]
 [ 305   27   12  257   12    0   16    4   22   50    9]
 [ 107  261   12   73  214    1   52    2    0    2   38]]
```



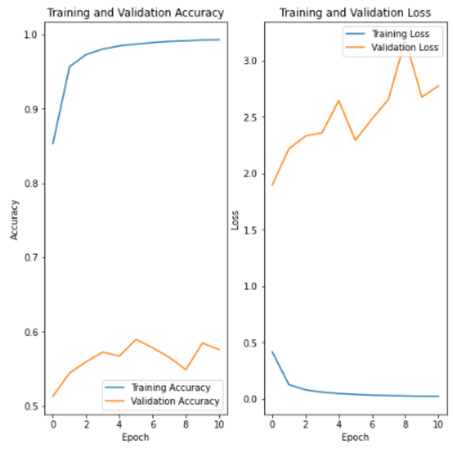
(b) Confusion matrix and heat map.

Figure 6: Results for the Weighted model.

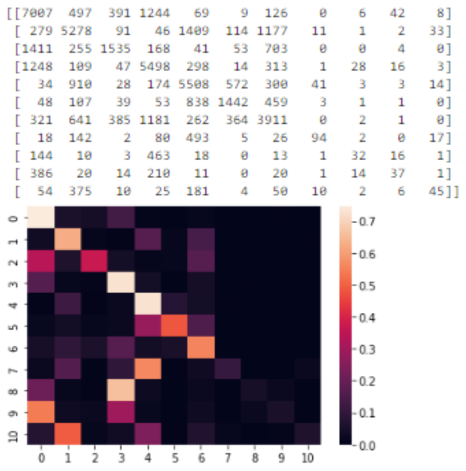
For the Weighted model, the training accuracy improved from about 42% to approximately 96%, with loss decreasing from 1.4 to 0.08. The validation accuracy peaked at 54%. The validation loss increased to a maximum of 2.8. The confusion matrix and heat map show a similar trend to that of the baseline, with the first-degree relationships having significantly better accuracy than the second-degree relationships.

Oversampling

The results of the Oversampling model are shown in Figure 7. The trained model is saved as ”oversample.h5” in the ”/Model” directory of the Google Drive.



(a) Accuracy and Loss.



```
[[7007  497  391 1244   69    9  126    0    6   42    8]
 [ 279 5278   91   46 1409  114 1177   11    1    2   33]
 [1411  255 1535  168   41   53  703    0    0    4    0]
 [1248  109   47 5498  298   14  313    1   28   16    3]
 [  34  910   28  174 5508  572  300   41    3    3   14]
 [  48  107   39   53  838 1442  459    3    1    1    0]
 [ 321  641  385 1181  262  364 3911    0    2    1    0]
 [  18  142    2   80  493    5   26   94    2    0   17]
 [ 144   10    3  463   18    0   13    1   32   16    1]
 [ 386   20   14  210   11    0   20    1   14   37    1]
 [  54  375   10   25  181    4   50   10    2    6   45]]
```

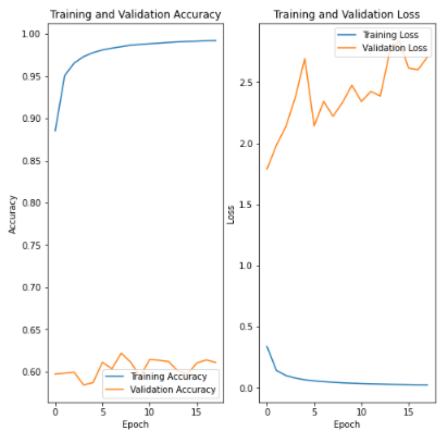
(b) Confusion matrix and heat map.

Figure 7: Results for the Oversampling model.

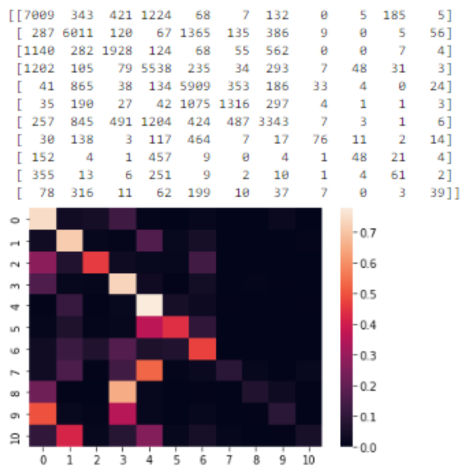
For the Oversampling model, the training accuracy improved from approximately 85% to about 99%. The training loss decreased from about 0.42 to 0.02. The validation accuracy increased from approximately 51% to 58.5%, while its loss increased from approximately 1.9 to 2.8. The confusion matrix and heat map show a similar trend to that of the weighted model, with the first-degree relationships having better accuracy than the second-degree relationships.

Augmentation

The results of the Augmentation model are shown in Figure 8. The trained model is saved as "augment_For_Sponsor.h5" in the "/Model" directory of the Google Drive.



(a) Accuracy and Loss.



```
[[7009  343  421 1224   68    7  132    0    5  185    5]
 [ 287 6011  120   67 1365  135  386    9    0    5   56]
 [1140  282 1928  124   68   55  562    0    0    7    4]
 [1202  105   79 5538  235   34  293    7   48   31    3]
 [  41  865   38  134 5909  353  186   33    4    0   24]
 [  35  190   27   42 1075 1316  297    4    1    1    3]
 [ 257  845  491 1204  424  487 3343    7    3    1    6]
 [  30  138    3  117  464    7   17   76   11    2   14]
 [ 152    4    1  457    9    0    4    1   48   21    4]
 [ 355   13    6  251    9    2   10    1    4   61    2]
 [  78  316   11   62  199   10   37    7    0    3   39]]
```

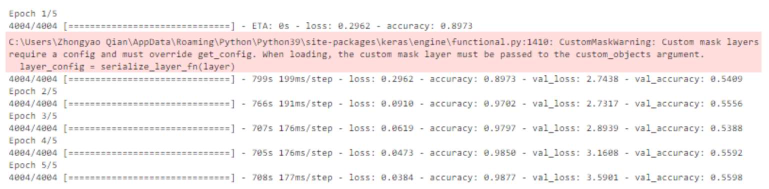
(b) Confusion matrix and heat map.

Figure 8: Results for the Augmentation model.

For the Augmentation model, the training accuracy increased from slightly below 80% to around 91%, while its loss decreased from approximately 0.5 to below 0.25. The validation accuracy peaked at about 61%, while the loss increased as the Epochs progressed, from about 1.6 to 2.1. The confusion matrix and heat map show the same trend as the previous models, with the first-degree relationships having better accuracy than the second-degree relationships.

Siamese Network

The results of the Siamese Network are shown in Figure 9. The trained model is saved as "siamese.h5" in the "/Model" directory of the Google Drive.

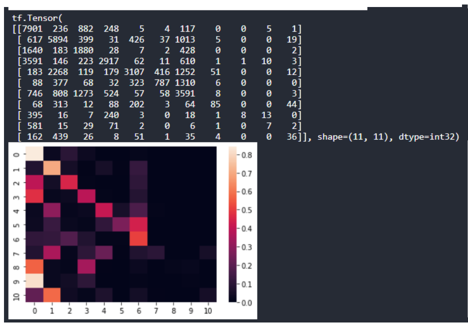


```
Epoch 1/5
4004/4004 [==============================] - ETA: 0s - loss: 0.2962 - accuracy: 0.8973
C:\Users\Zhongyao Qian\AppData\Roaming\Python\Python39\site-packages\keras\engine\functional.py:1410: CustomMaskWarning: Custom mask layers
require a config and must override get_config. When loading, the custom mask layer must be passed to the custom_objects argument.
  layer_config = serialize_layer_fn(layer)
4004/4004 [==============================] - 799s 199ms/step - loss: 0.2962 - accuracy: 0.8973 - val_loss: 2.7438 - val_accuracy: 0.5409
Epoch 2/5
4004/4004 [==============================] - 766s 191ms/step - loss: 0.0910 - accuracy: 0.9702 - val_loss: 2.7317 - val_accuracy: 0.5556
Epoch 3/5
4004/4004 [==============================] - 707s 176ms/step - loss: 0.0619 - accuracy: 0.9797 - val_loss: 2.8939 - val_accuracy: 0.5388
Epoch 4/5
4004/4004 [==============================] - 705s 176ms/step - loss: 0.0473 - accuracy: 0.9850 - val_loss: 3.1608 - val_accuracy: 0.5592
Epoch 5/5
4004/4004 [==============================] - 708s 177ms/step - loss: 0.0384 - accuracy: 0.9877 - val_loss: 3.5901 - val_accuracy: 0.5598
```

Figure 9: Results of incomplete Siamese model training.

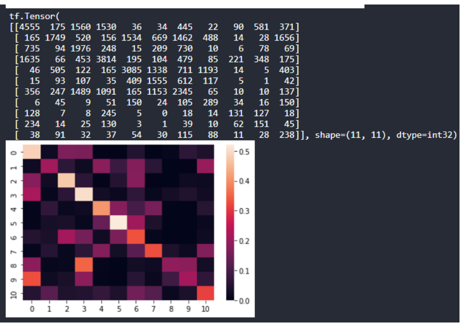
For the Siamese Network, the training process was taking significantly longer for each epoch, while the performance was not better than our previous model, so we discontinued after the fifth epoch.

One Epoch test results



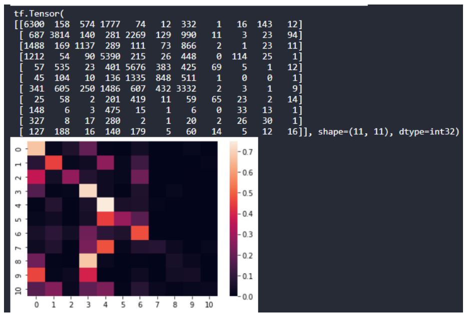
```
tf.Tensor(
[[7901  236  882  248    5    4  117    0    0    5    1]
 [ 617 5894  399   31  426   37 1013    5    0    0   19]
 [1640  183 1880   28    7    2  428    0    0    0    2]
 [3591  146  223 2917   62   11  610    1    1   10    3]
 [ 183 2268  119  179 3107  416 1252   51    0    0   12]
 [  88  377   68   32  323  787 1310    6    0    0    0]
 [ 746  808 1273  524   57   58 3591    8    0    0    3]
 [  68  313   12   88  202    3   64   85    0    0   44]
 [ 395   16    7  240    3    0   18    1    8   13    0]
 [ 581   15   29   71    2    0    6    1    0    7    2]
 [ 162  439   26    8   51    1   35    4    0    0   36]], shape=(11, 11), dtype=int32)
```

(a) Baseline confusion matrix and heat map.



```
tf.Tensor(
[[4555  175 1560 1530   36   34  445   22   90  581  371]
 [ 165 1749  520  156 1534  669 1462  488   14   28 1656]
 [ 735   94 1976  248   15  209  730   10    6   78   69]
 [1635   66  453 3814  195  104  479   85  221  348  175]
 [  46  505  122  165 3085 1338  711 1193   14    5  403]
 [  15   93  107   35  409 1555  612  117    5    1   42]
 [ 356  247 1489 1091  165 1153 2345   65   10   10  137]
 [   6   45    9   51  150   24  105  289   34   16  150]
 [ 128    7    8  245    5    0   18   14  131  127   18]
 [ 234   14   25  130    3    1   39   10   62  151   45]
 [  38   91   32   37   54   30  115   88   11   28  238]], shape=(11, 11), dtype=int32)
```

(b) Weighted confusion matrix and heat map.



```
tf.Tensor(
[[6300  158  574 1777   74   12  332    1   16  143   12]
 [ 687 3814  140  281 2269  129  990   11    3   23   94]
 [1488  169 1137  289  111   73  866    2    1   23   11]
 [1212   54   90 5390  215   26  448    0  114   25    1]
 [  57  535   23  401 5676  383  425   69    5    1   12]
 [  45  104   10  136 1335  848  511    1    0    0    1]
 [ 341  605  250 1486  607  432 3332    2    3    1    9]
 [  25   58    2  201  419   11   59   65   23    2   14]
 [ 148    6    3  475   15    1    6    0   33   13    1]
 [ 327    8   17  280    2    1   20    2   26   30    1]
 [ 127  188   16  140  179    5   60   14    5   12   16]], shape=(11, 11), dtype=int32)
```

(c) Oversampled confusion matrix and heat map.

Figure 10: Results from running only one epoch.

5 Conclusion

In this project, we were able to leverage, train, and finely tune a CNN model on the Families in the Wild dataset to determine the familial relationships between sets of two people. The imbalanced nature of the dataset prompted us to experiment with adjustments to the baseline model, using weighting and oversampling. In an attempt to improve the accuracy, we also implement augmentation to the oversampled baseline. In addition, we attempted a Siamese Network to compare its results with the previous models. The results of these models differ slightly, with Augmentation proving to be the most accurate with validation accuracy reaching 62.2%, followed by Oversampling at 59.0% accuracy, Baseline at 58.9%, and Weighted at 55.0%. We found that the Siamese model takes significantly longer to train on each epoch, due to the two models are being simultaneously trained, while producing similar results to the rest of the models. Out of the different methods, Augmentation returns the highest accuracy because it provides more randomness to the oversampled dataset.

For all of the models, we observed that as validation accuracy increased, so did its loss. We hypothesize that this is because the four second-generation classes aren't as recognizable to the model, due to the imbalanced data providing fewer instances of these relationships to train on. Even when using oversampling and weighting, the model's attempts to minimize the loss in training lead to overfitting on these last 4 classes, while still learning new things about the other classes. We developed this hypothesis based on tests done using only one epoch on each of these different models [Figure 10]. In the first epoch, the weighted results [Figure 10b] fit much better for the minority classes than either the baseline [Figure 10a] or oversampled [Figure 10c] models. However, the oversampled model does perform better than the baseline on the minority classes, while achieving the same accuracy. This reinforces the notion that as more training is conducted, the fit on the minority classes becomes worse.

A potential future approach to achieve better accuracy would be to use a Generative Adversarial Network [7]. This would be used instead of oversampling, as it generates more data for the minority groups. Another improvement we could envision would be to include metadata as a class, such as when the picture was taken or the age of the individual in the portrait, as this could assist the model with instances in the dataset of pictures of the same person from different decades.

References

[1] Joseph P. Robinson et al. *Families In the Wild A Large-Scale Kinship Recognition Image Database*. URL: https://web.northeastern.edu/smilelab/fiw/index.html. (accessed: 11/3/2021).

[2] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *International Conference on Machine Learning* (2019). DOI: https://arxiv.org/abs/1905.11946.

[3] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: (2009).

[4] Min Lin, Qiang Chen, and Shuicheng Yan. "Network In Network". In: (2014), p. 4. arXic: 1312.4400 (cs.NE).

[5] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2017). arXiv: 1412.6980 [cs.LG].

[6] Davide Chicco. "Siamese Neural Networks: An Overview". In: (2021). Ed. by Hugh Cartwright, pp. 73–94. DOI: 10.1007/978-1-0716-0826-5_3. URL: https://doi.org/10.1007/978-1-0716-0826-5_3.

[7] Antonia Creswell et al. "Generative Adversarial Networks: An Overview". In: *IEEE Signal Processing Magazine* 35.1 (Jan. 2018), pp. 53–65. ISSN: 1053-5888. DOI: 10.1109/msp.2017.2765202. URL: http://dx.doi.org/10.1109/MSP.2017.2765202.