Face Detection, Classification, and Recognition from UAV Footage

Anthony Vasquez
Johns Hopkins University
Whiting School of Engineering
Baltimor, MD
avasque1@jh.edu

***Abstract*—**

Unmanned aerial vehicles (UAVs) are usually controlled by a ground pilot but can also be flown autonomously. UAVs are usually equipped with a streaming device such as a camera, from which humans and events can be captured from a bird's eye-view. One application of UAV footage includes detection of humans of interest (HOI). This might include missing or lost persons, criminals, or UAV owners. In this paper, current literature and methods are explored, and a face detection and recognition pipeline are created and compared to the work done by DroneFace.

Keywords—UAV, detection, recognition, HOI

I. INTRODUCTION

UAVs are increasingly less expensive to purchase and have many off-the-shelf models with optional high-grade camera upgrades. Furthermore, flight hardware, batteries, and long-range controller signals, allow for the use of UAVs at long range distances. UAVs are agile, directable, and able to get into hard-to-reach spaces. In short, UAVs are effective tools for face recognition tasks on human populations.

After doing preliminary research, it was discovered that there is no shortage of interest in this topic. The face can be thought of as similar to a visual fingerprint that humans and computer vision algorithms alike use to recognize one another. The use-cases for face detection and recognition are countless, but one example is finding a missing elderly person (Hsu, 2015). Attempts of face recognition using UAVs have been around for more than a decade, and many open-source and licensed implementations have been created. However, there are many challenges that have yet to be solved without expensive hardware or labor-intensive data curation. Furthermore, many failure modes exist, such as large angle and long distance of camera location relative to target. Camera quality, framerate, weather, and other factors potential lead to failure modes. This work attempts to take an open-source dataset, label images, and use open-source architectures to create a cheap off-the-shelf multi-stage face recognition model.

II. RELATED WORK

A. *Face Detection*

The authors of the FaceDrone experiment investigated Face++ (Andreas Rossler, 2019), ReKognition (Amazon, 2024), Haar Cascades face detection implementation (OpenCV, 2024), and Local Binary Patterns (LBPs) (Rosebrock, 2021). For the task of face detection, their findings showed that Face++ and ReKognition performed poorly in comparison to the open-source Haar methods. These are important findings, because it shows that for high resolution low altitude drone footage open-source methods exist that significantly outperforms pretrained models that require a license to use.

Method	# of faces	TPR	FPR
Face^{++}	20	0.14	0.05
ReKognition	37	0.27	0.13
Haar (default)	14,777	0.71	0.93
Haar (alt)	1,700	0.77	0.37
Haar (alt2)	2,545	0.78	0.57
Haar (alt tree)	510	0.37	0.002
LBP	2,964	0.63	0.70

*The methods are asked to detect 1,364 target faces from 620 pictures.

Figure 1. Face Detection Performance Results of DroneFace (Hsu, 2015)

B. *Face Classification*

Face classification is not explored in the FaceDrone study, however, it is an interesting topic that will also be explored in this work, if nothing else than to satisfy curiosity. The definition of face classification as used here is a classification model that classifies one person from another. In this example, there are 11 different people, and the classifier is trained to correctly classify each of the people separately. This is not the case with face recognition as it is defined below. No literature

search is done for this study; however, a simple CNN is experimented with.

C. Face Recognition

FaceDrone used Face++ and ReKognition to train face recognition models. These models performed much better TPR and FPR than was the case for the face detection task. It shows that there is a market for these licensed APIs, and they perform well for face recognition. However, the performance is highly dependent on angle of depression, or the angle created between the UAV and HOI. These models are stable for most depression angles when the UAV is 4 meters from target. Unfortunately, these scores suffer as the angle of depression increases, especially for Face++.

Three models each for Face++ and ReKognition named J, P, and B are trained and tested with TPR as the performance metric. With 95% confidence, as a default match level, the trained models are tested to acquire the recognition score (TPR). A match is where the model matches the detected face with the correct profile. Results shows that ReKognition far exceeds the recognition score above the default match level then does Face++. The results from that study are shown below (Figure 2).

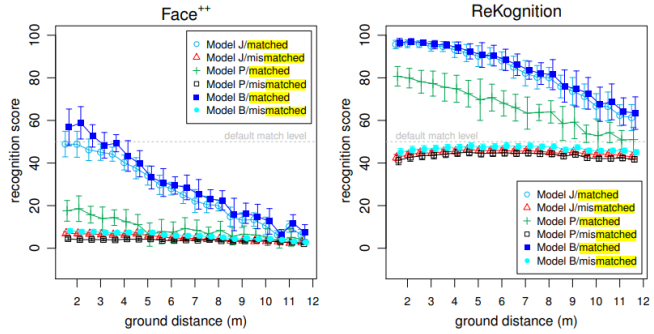

Figure 2. Matched and Mismatched Performance of Face++ and ReKognition models. (Hsu, 2015)

III. Face Detection Experiment

As previously mentioned, the dataset and models used are all open-source. In the case of YOLOLv8, this can be used for free, as long as monetary gain from the model is not needed. The raw data involves label generation and preprocessing. The preprocessing used include augmentations that are mentioned in more detail in the results section. The architectures are tuned using a basic policy that can be explored in more depth for future work.

A. Face Detection Dataset

The dataset is from DroneFace (Chen, 2015). It contains 620 GoPro (GoPro, n.d.) large-scale 24-bit RGB images of resolution 3680x2760 with a 170-degree field-of-view (FOV). Each of the large-scale frames include two to three HOIs on the center of the horizontal axis. However, there are times when several more HOIs are within the frame at different locations, because the data was collected on a university campus, so students and professors external to the experiment are seen walking through the campus. The intended HOI population include 11 subjects altogether. The footage is from 2-17 meters away from the HOIs with footage recorded at distances of 0.5m intervals. The UAV hovers either at 1.5 (eye-level), 3.0, 4.0, or 5.0 meters high.

Figure 3. Large-Scale UAV Frame

Additionally, there are 1364 24-bit RGB chipped images of 7 males and 4 females that are all HOIs within the large-scale raw images. Chipped images are images extracted from the bounding box coordinates of larger images. These chips range in resolution between 23x31 to 384x384 pixels depending on camera to HOI distance and angle. These images are all extracted from the large-scale images.

Figure 4. Example of Chipped Profile from Large-scale Images

There are an additional seven frontal images of the HOIs with white backgrounds with resolution 1174x1430.

Figure 5. Example of Ideal Profile Image

There are also 30 frontals, right, and left profile pictures of each HOI of resolution 696x696. However, since these images were not taken in a controlled environment, the background colors are not uniform.

Figure 6. Example of Left Profile Image

B. Face Detection Preprocessing

The dataset included chipped images of all HOIs, however, it does not provide bounding box information of the location of the detection within the large-scale image. The dataset is small enough to manually label, since it only has 620 large-scale images. For that reason, these images were manually labelled and will be used for match/unmatched data. Yolo_Label (Kwon, 2023) was used to generate the labels. Each large-scale image had an associated text file with the same name as the image, and each object was listed (one object per line) in the format: class, upper left x, upper left y, lower right x, lower right y.

C. Face Detection Architecture

As mentioned by other researchers including FaceDrone, a face can only be recognized if it can first be detected. For that reason, a well-known and generally high performing object detector was used for face recognition, namely You-Only-Look-Once Version 8 (Ultralytics YOLOv8, 2024). This version of YOLOv8 is different than that of Joseph Redmon's implementation (Redmon J. , 2016), which uses darknet53 as a CNN backbone to the object detection algorithm.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 7. Darknet53 Backbone which YOLOv8 is derived from (Redmon J. F., 2018)

Unfortunately, YOLOv8 is not open-source and the full implementation, including the backbone is unknown by its users, but it is based on the YOLOv3 implementation. It is also more of a framework than YOLOv3, as it can be used for classification, segmentation, or object detection. Though it boasts of higher benchmark performance, in terms of speed and mean-average-precision (mAP) on common computer vision datasets, Ultralytics has yet to publish a paper on the subject. What makes YOLOv8 attractive is the ease of use and speed of development. Also, as long as monetary gain is not a goal of the use-case, YOLOv8 is a viable option.

D. Face Detection Performance Evaluation

TPR (1) is the metric used to compare performance of DroneFace, so the same metric is used here for all tasks. Face detection performance is expected to be comparable with LBP or ReKognition, because YOLOv8 is a reliable architecture capable of performing most object detection tasks well compared to other similar architectures.

Equation 1. The metric used for face detection, classification, and recognition is the true positive rate.

$$TPR = \frac{TP}{TP+FN}$$

E. Face Detector Train and Test Results

The training and validation sets are made of 620 large-scale manually labelled images, with a 10% split for validation. Because the dataset is so small, the dataset was only split two ways for training and validation, as opposed to training, validation, and testing. This methodology should be revisited pending procuration of new data.

Most of the truth labels are located along the center of the horizontal axis, but some faces also occur around the image boundaries. This is due to bystander campus traversal. Altogether, there are about 1400 face instances that vary in width and height, about 0.02 to 0.10 normalized units in comparison to the large-scale images. Normalization here refers to the spatial dimensions of the bounding box with respect to the size of the input image. The top right quadrant of Figure 8 shows the relative sizes of the bounding box to the image size. The lower quadrants show meta-data about the detections.

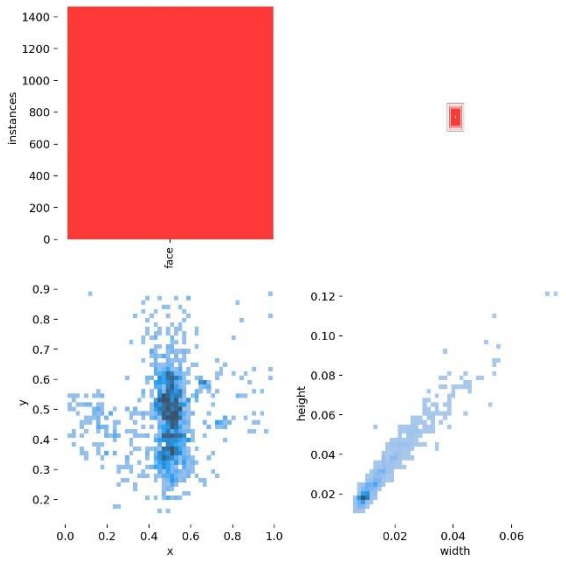

Figure 8. Basic Train Set Statistics

Moderate image augmentations were used to keep the model from overfitting the data. The augmentations used were horizontal flip, 30 percent rotations, RGB to HSV transfer, saturation, contrast, scaling, translation, copy-paste, and mosaic. An example of what that looks like can be seen in Figure 9.

Figure 9. Augmentations Used During Training

The model trained for 65 epochs before converging to an F1 score of about 0.91 at a confidence of 0.50.

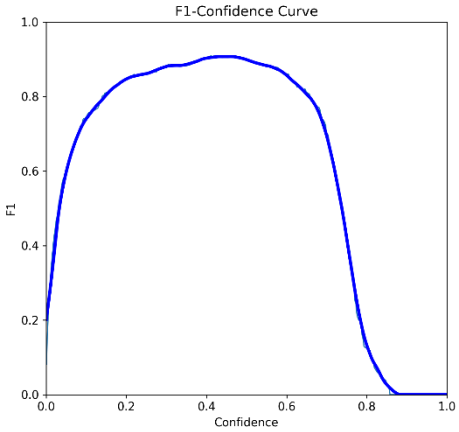

Figure 10. F1 Confidence Curve Obtained from the Validation Set

Other performance metrics were also gathered from training and generally show performance convergence, including train box loss, train class loss, train and test precision, train and test recall, and train and test mAP scores.

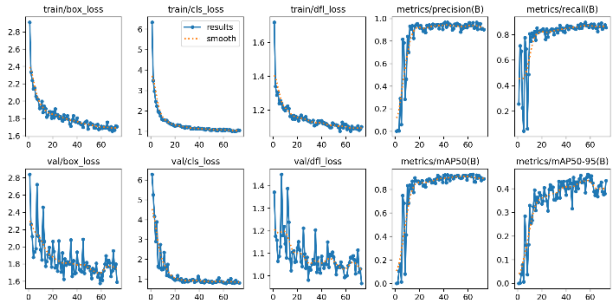

Figure 11. Common Performance Metrics

TPR was captured for the validation set and scored 0.96 at a 0% confidence and 0.92 on 0.5% confidence as shown by the Recall curve.

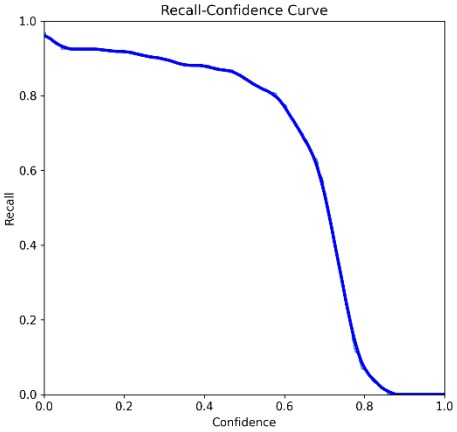

Figure 12. Recall vs. Confidence Curve on Validation Set

These results show that a face detector trained on a small amount of data can be implemented with YOLOv8. It also shows that this method performed better than the best LBP method used by the DroneFace authors. It should be noted that

this network was trained on the same dataset, but their actual train/test split is unknown. For that reason, direct score comparisons cannot be done. However, 14% is a big improvement and a solid start for face recognition

IV. Face Classification Experiment

The aim of the classification experiment was to understand if drone footage could be used for face recognition using a small edge-computer worthy CNN architecture. This experiment serves two purposes. The first purpose is to know if classification is a good approach to this problem. The second is to compare with a similar network trained using contrastive learning and the Siamese Network paradigm. For the classification method, this might not be an effective way of face recognition, because each POI would need to be included in the dataset for a trained model to be effective. The use-case would be to detect a small known population of people for which ample labeled data are provided.

A. Face Classification Dataset

The cropped images included with the DnHFaces dataset are used to train a classifier on the eleven participants. These snapshots are like image chips output by the object detector and can be seen in Figure 4. As mentioned above, the object detector was only trained to detect a face within an image. However, the cropped images were labelled by a specific person ID, making an identification classifier possible. In this case, each of the eleven people was labelled as a letter (a-k). Each class had 125 images, with a total dataset of about 1375 images.

B. Face Classification Preprocessing

The transforms used to augment data features for training include RandomResizedCrop, RandomAdjustSharpness, RandomAutoContrast, RandomInvert, GaussianBlur, RandomHorizontalFlip, and Normalization. For validation and testing, only Resize, Center Crop, and Normalization were used. All the mentioned augmentations are widely used among computer vision experts and are available using Torchvision (torchvision, 2017).

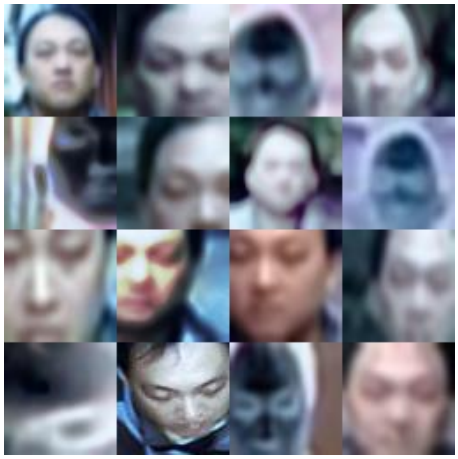

Figure 13. Torchvision training augmentation shown on half of a training batch.

The augmentations used for face classification were also used for face recognition. Further studies may include the effects of augmentations on model generalizability. At this point it is difficult to know if these augmentations limit the ability of the Siamese network to learn meaningful representations of the data. What is known however, is that these transformations proved to be an exceptional combination on this dataset for the classification task. Results are shown in the Classification results section.

C. Face Classification Architecture

A very small CNN network with about 95K trainable parameters was coded in python using Pytorch for the task of face classification. The architecture is called SimpleCNN, and an implementation was made available for open source at github.com (Vasquez, 2024). This is the same architecture used for the Siamese network mentioned in a later section of this paper (Face Recognition Architecture).

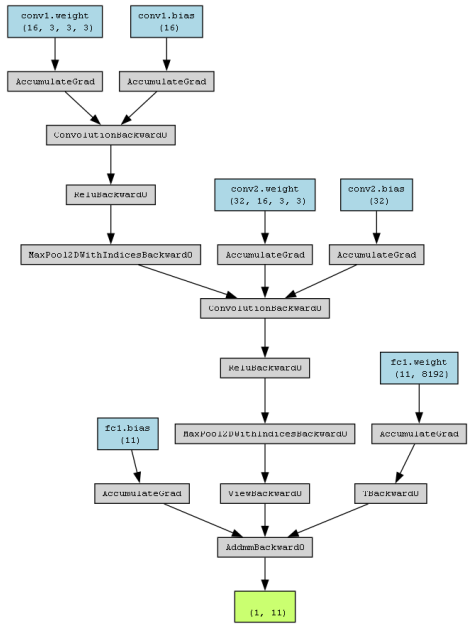

Figure 14. The SimpleSiam face classifier was kept small to not overfit the small dataset.

D. Face Classification Evaluation of Performance

Equation 1 is used to evaluate the performance of the face classifier.

E. Face Classification Train and Test Results

The network was trained for 300 epochs with a learning rate scheduler with Adam optimizer. The optimizer was initialized at 0.0001 learning rate, and the scheduler stepped every 30 epochs gradually reducing the learning until training was complete. A 1-d gaussian filter was applied to the training curves and shows a smooth and decreasing loss curve.

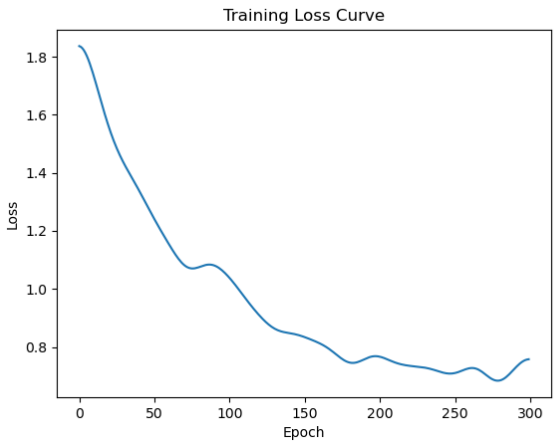

Figure 15. The network trained for 300 epochs before converging to about cross-entropy-loss of 0.4.

The results showed that the SimpleCNN network generalized better than expected. The validation metrics were recorded every two epochs and showed smooth training curves. Figure 16 shows that both precision and recall curves increase to nearly a perfect score of 1.0, while validation loss approaches zero.

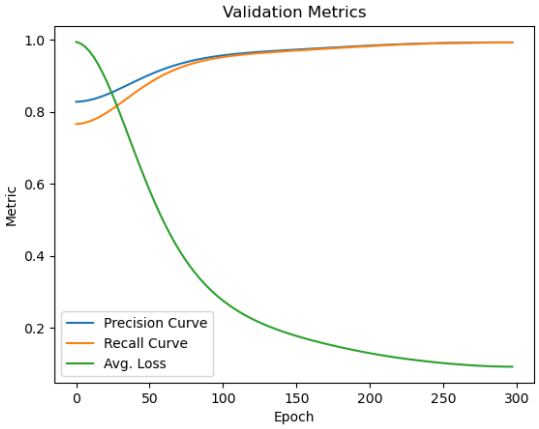

Figure 16. Precision, TPR (Orange), and cross-entropy-loss were recorded during training to produce the above plot.

A confusion matrix was also used on the test set to understand the model's ability to generalize to new data (Figure 17).

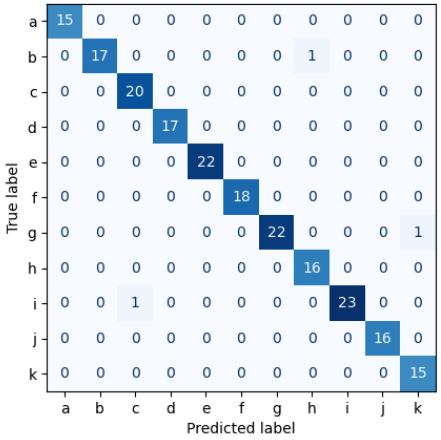

Figure 17. Confusion matrix showing the generalization performance on new data.

The y-axis of the confusion matrix shows true labels with each identity represented as a letter. The x-axis shows the predicted labels. There were only three incorrect classifications on the test set, which affirms the conclusion that face classification is effective at correctly classifying faces for this dataset.

V. Face Recognition Experiment

A. Face Recognition

For the Face recognition task, a Siamese network was used, where two images are forward passed through a trained model, and a distance metric is used on the output to compare similarity.

B. *Face Recognition Dataset*

The recognition dataset was the same dataset used for the classification task. However, the Siamese network training paradigm makes pairing of the dataset necessary. For example, one person was paired with other images of the same person at different angles and distances. These types of pairings are known as genuine pairs (Figure 18).

Figure 18. Example of genuine pair images.

A separate complimentary dataset is also created of people that are not the same person. These pairings are known as imposter pairs (Figure 19).

Figure 19. Example of imposter pair images.

The train and test split involved the segregation of genuine images, with each sample consisting eight genuine examples. Likewise, the imposter pairs contained eight imposter pairs each. The total number of such examples was 35000 for training and 350 for testing.

C. *Face Recognition Architecture*

Previously, a VGG16 network was going to be used as a feature extractor, however, given the size of the dataset, this network was determined to be over-parameterized which could lead to an overfit model.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 20. VGG-16 Architecture (Zisserrman, 2015)

The reason that VGG16 was first considered, was that pretrained checkpoint weights trained on ImageNet could be leveraged for model fine-tuning (Pytorch, 2017). Using pretrained weights can make model training shorter and potentially yield better training results. Unfortunately, the extremely small dataset made pretrained model usage null. To prevent overfitting, a much smaller CNN architecture with approximately 95K trainable parameters was used for both classification and recognition. These small custom CNN architectures are called SimpleCNN and SimpleSiam respectively (Vasquez, 2024). For simplicity's sake, the two networks are identical, except for a sigmoid activation at the head of SimpleSiam. Though the two networks are nearly identical, the way that they are used differs significantly, since two forward passes are combined into a single forward pass function prior to updating the weights.

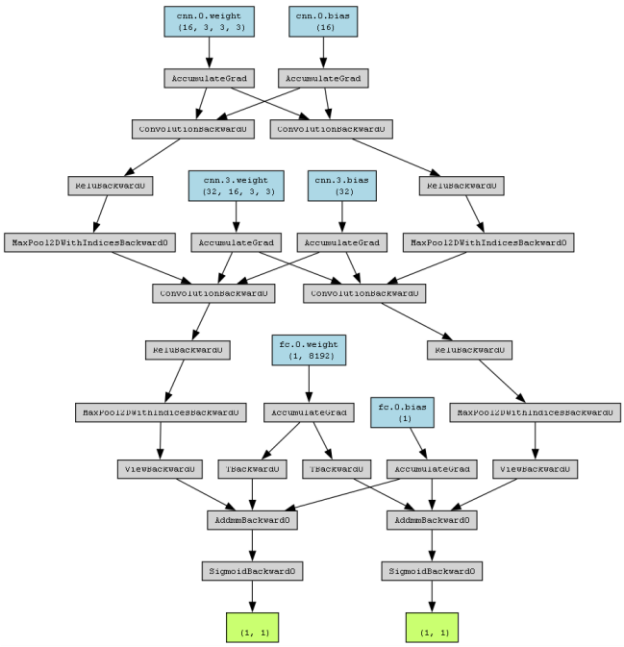

Figure 21. The graph of SimpleSiam is shown here using the pytorchviz software created in 2021 (Zagoruyko, 2024).

Notice that SimpleSiam has two heads, one for each input image. The idea is that the difference between the feature space representation of the two input images should be different. If the difference between one input image and the other is large beyond a predetermined threshold, then in theory, the match is considered an imposter. On the other hand, if the distance is small, then the match is considered genuine. For this case, the two input images are of either the same person, or of two different people, and the metric used to determine the difference is the Euclidean distance.

Equation 2. Euclidean distance metric used for determining sample similarity.

$$D_w = \sqrt{(x_2 - x_1)^2 + (y_2 - y1_1)^2}$$

The distance here refers to the distance between the network output representations of the two input images. There are many other distance metrics that can be used for this problem, however the only one used in this study is the Euclidean distance. The loss criterion used for optimization guidance is called Contrastive Loss.

Equation 3. The Contrastive Loss criterion is shown below. Notice that the input requires two samples (feature space representations of each image in this case), and one truth label.

$$L(x_1, x_2, y) = \frac{1}{2}(1 - y){D_w}^2 + \frac{1}{2} y * max(0, m - D_w)^2$$

This loss function requires two input feature maps and a truth label (y). A label of "0" means that the pair are genuine, whereas a "1" indicates an imposter pair. The margin "m" is a preselected value of 0.6, which is the best performing margin. Other margins were experimented with, such as, 0.1, 0.2, 0.4, 0.5, 1.0, and 2.0, but are not shown here. As Equation 3 suggests, a genuine pair (y=0) gives a loss function of $1/2D_w$, encouraging optimization to small distance. On the other hand, imposter pairs (y=1) equate to a loss function of $1/2\max(0, m - D_w)^2$, effectively encouraging optimization to large distances.

D. Face Recognition Performance evaluation

For face recognition, TPR (Equation 1) will be used for matched and unmatched results. Unlike face detection, face recognition may not perform as well, because the model might be archaic in comparison to the Face++ and ReKognition models which were built with face recognition in mind.

E. Face Recognition Train and Test Results

Training the Siamese network is slower and took 5+ hours to complete 100 epochs. This is because, as Figure 21 suggests, the network shares the same weights, but both genuine and imposter pairs are forwarded through the network and backpropagation calculations are also doubled.

During training, the contrastive loss curve was monitored and showed a decreasing curve. However, the loss converged after only about 40 epochs. It is hard to say why, but one theory is that the augmentations were a poor fit for training. Another theory is that the network was not complex enough to learn deeper features. To better understand why, more conclusive methods would need to be experimented with.

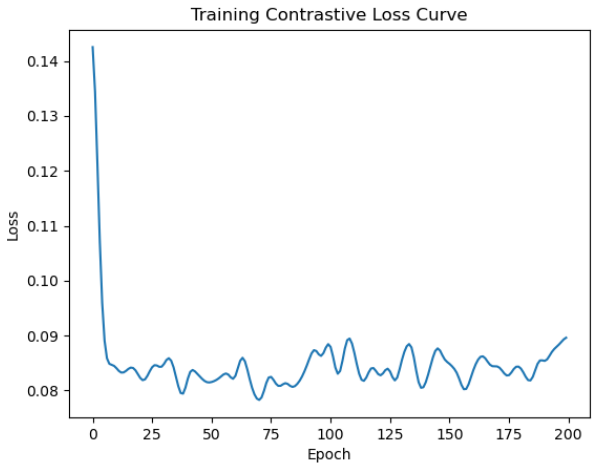

Figure 22. The contrastive loss curve is shown to converge early in training and fails to continue learning.

Just like the classification results, precision, TPR and validation loss were also recorded during training. Figure 23 shows that validation convergence for Precision, TPR, and contrastive loss occur at about 75 epochs. These results further

affirm the possibility that architecture complexity is not sufficient, since it fails to continue learning, and no divergence in validation loss occurs.

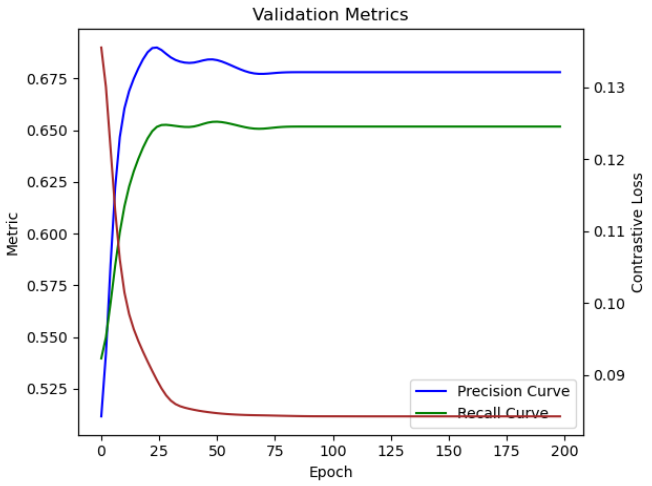

Figure 23. Validation metrics were recorded during training and show that the performance values all converge relatively early on, at just 75 epochs.

An unimpressive confusion matrix of the test set is shown below (Figure 24). Unlike the performance of the object detector, and classifier, the TPR on the test set is only about 0.65, showing that improvements are needed for face recognition.

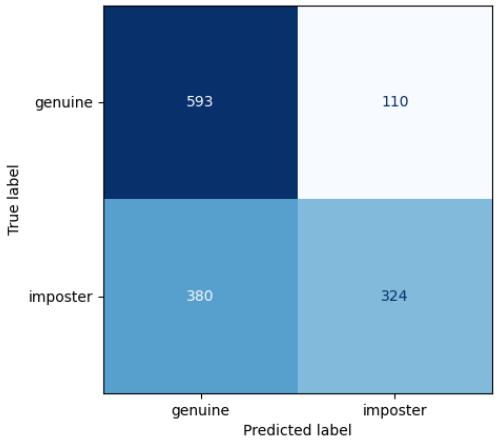

Figure 24. The test set confusion matrix is shown above and proves that the architecture used for this task is suboptimal.

The Siamese network outputs feature maps, which are used to calculate the Euclidean distance. As explained earlier, the perfect result is that inputs with smaller distances are likely genuine, and those with larger distances are likely imposters. Figure 24 was created by setting a pass-distance-threshold, so if the distance between samples was below the threshold, then the prediction was correct, else the prediction was incorrect. One drawback of this approach is that this threshold is another hyperparameter that needs tuning.

VI. Conclusion

Three paradigms were explored in this work, namely face detection, classification, and recognition. The face detection method used here scored a 14% increase in TPR compared to the top model of the 2018 DroneFace study of Face++, ReKognition, Haar, and LBP, on their custom dataset (Figure 1). Furthermore, the Siamese network also scored a better average TPR than Face++ for the task of recognition. As described earlier, Face++ generally scored poorly and was significantly outperformed by ReKognition. Unfortunately, the Siamese network scored an average of 20+ percent lower TPR than ReKognition (Figure 2). However, the Siamese network architecture used here was shown to be under parameterized for this task, and making small changes to the network would likely yield good results. Further work should be done to analyze the Siamese performance by using a better fit architecture. Finally, as a bonus, face classification was also explored. This task was more of a curiosity, but the methods used yielded promising results. It showed that a small high performance CNN architecture could be trained on the drone DroneFace dataset.

References

[1] Adam Pazke, E. a. (2017). Automatic Differentiation in Pytorch.

[2] Amazon. (2024, April 10). *What is Amazon Rekignition*. Retrieved from aws: https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html

[3] Andreas Rossler, E. a. (2019, August 2019). FaceForensics++: Learning to Detect Manipulated Facial Images. Munich, Germany.

[4] Chen, H.-J. H.-T. (2015). *DroneFace*. Retrieved from DroneFace: https://hjhsu.github.io/DroneFace/

[5] GoPro. (n.d.). *GoPro*. Retrieved from https://gopro.com/en/us/

[6] Hsu, H.-J. a.-T. (2015, May). Face Recoognition on Drones: Issues and Limitations. Taipei, Taiwan.

[7] Kwon, Y. (2023). *Yolo_Label*. Retrieved from github.com: https://github.com/developer0hye/Yolo_Label

[8] OpenCV. (2024, April 9). *Face Detection using Haar Cascades*. Retrieved from OpenCV.org: https://docs.opencv.org/4.x/d2/d99/tutorial_js_face_detection.html

[9] Pytorch. (2017). *VGG16*. Retrieved from pytorch.org: https://pytorch.org/vision/main/models/generated/torchvision.models.vgg16.html

[10] Redmon, J. (2016). *darknet*. Retrieved from pjreddie.com: https://pjreddie.com/darknet/

[11] Redmon, J. F. (2018, April 8). YOLOv3: An Incremental Improvement. Washington.

[12] Rosebrock, A. (2021, May 3). *Face Recognition with Local Binary Patterns (LBPs) and OpenCV*. Retrieved from pyimagesearch: https://pyimagesearch.com/2021/05/03/face-recognition-with-local-binary-patterns-lbps-and-opencv/

[13] torchvision. (2017). *torchvision*. Retrieved from pytorch.org: https://pytorch.org/vision/stable/index.html

[14] Ultralytics YOLOv8. (2024). Retrieved from ultralytics.com: https://github.com/ultralytics/ultralytics

[15] Vasquez, A. (2024, May 2). *face_recognition*. Retrieved from git: https://github.com/vanthony715/face_recognition/tree/master

[16] Zagoruyko, S. (2024, 5 2). *pytorchviz*. Retrieved from git: https://github.com/szagoruyko/pytorchviz

[17] Zisserrman, K. S. (2015, April 10). Very Deep Convolutional Networks For Large-Scale Image Recognition. Oxford, United Kingdom.