ISYE 6740 - Summer 2023
Final Report

Team Member Names: Chloe Pomeroy

Project Title: Exploring the Effectiveness of Facial Recognition in Matching Lost Pets

Problem Statement

I recently joined NextDoor to engage more with my local community, and noticed that a significant proportion of posts revolve around lost pets. Individuals frequently share pictures of their missing pets, hoping for any leads, while others who spot stray animals share pictures with accompanying location details, aiming to reunite them with their owners. I see many people making these kinds of posts, but the success rate seems to be rather low. The sheer volume of these posts makes it easy for pet owners to overlook some of them, and individuals who come across a lost pet are less likely to notice the corresponding owner's post if they are not actively seeking it.

To address this problem, I investigated the efficacy of different image recognition models in matching lost pets, and made observations whether certain species of pets are more identifiable with this approach. The end goal is to create a model that can analyze a pet picture and provide the top N closest matches from the database. Since the goal is to look for similar images, I believe that I could leverage facial recognition models for this purpose. By testing different facial recognition methods, I hope to create an accurate model that could help people find their lost pets more easily.

Data Source

For this project, I'm going to need to perform both object detection (finding where the object is in the image) and recognition (getting a similarity index between objects). The Common Objects in Context (COCO) dataset[1] is a famous dataset with bounding boxes for object detection and segmentation with many animal images. However, the annotations are only the species name (cat, dog, etc.), making it difficult to use for the recognition phase since the annotations don't have any indication of similarity. To obtain the data I could either use COCO's API or use a subset uploaded by others. For example, I've found this subset of COCO that contains 4000 cat images[2] that would be useful for my project.

The Oxford-IIIT Pet Dataset[3] contains 37 categories of dog/cat breeds with approximately 200 images per class, and could be useful for the recognition piece. Although the focus of this dataset lies in distinct breeds, it provides a suitable starting point for assessing similarity that can later lead to identifying individual animals.

In practice, many people will have pets that aren't purebred and therefore they won't neatly fit into the breed categories. I supplemented the Oxford-IIIT Pet Dataset with my own images of pets from friends and family, which I cropped and annotated manually. To further enhance the data in the future, I could scrape pet adoption websites that contain multiple images of each animal, such as the Toronto Humane Society[4].

Methodology

In this project, I want to compare 3 different aspects for effectiveness: the species of the animal, the algorithm used, and the angle of the picture.

Figure 1: Sample images after running through YOLOv8

Figure 2: Sample training images of 4 different classes for Eigenfaces and Fisher Faces, cropped by hand

I first used YOLOv8[5] out of the box to add bounding boxes and classifications to my images. If YOLOv8 could not find a dog or cat in the image, the image was discarded from the dataset. Then, I cropped the images to the bounding box with the correct classification (either cat or dog), because some images had other items classified such as chairs or couches. Finally, I resized them all using padding to match the largest image, so that all images would have uniform sizes.

I initially hoped to use the YOLOv8 object detection model to crop the images that I could use with Eigenfaces, FisherFaces, and FaceNet, but after attempting to train an Eigenface model with my cropped images from YOLOv8, I didn't get very promising results at all. I realized that these facial recognition models work much better with square images that contain only the face, and I wasn't able to find pre-existing models that detected only the face of animals and could provide me a bounding box like YOLOv8. There are many of these that work for human facial detection, but after attempting to use Hiroki Taniai's well-known FaceNet model that is pretrained on celebrity faces[8], no faces were being detected in my pet images. In the end, I wasn't able to find a pre-existing object detection model to do this for me, so I manually cropped the images. I created a dataset of 44 images, 11 images each from 4 different classes: 2 different cats (one was my cat and the other was the Abyssinian from the Oxford Pet dataset) and 2 different dogs (the yorkshire terrier and Staffordshire Bull Terrier from the Oxford pet dataset).

Eigenfaces

I trained three models using the Eigenfaces algorithm, one with all 4 classes, one with just the 2 cat classes, and one with just the 2 dog classes.

To perform the Eigenface algorith, for each category, each image is transformed into a vector, then

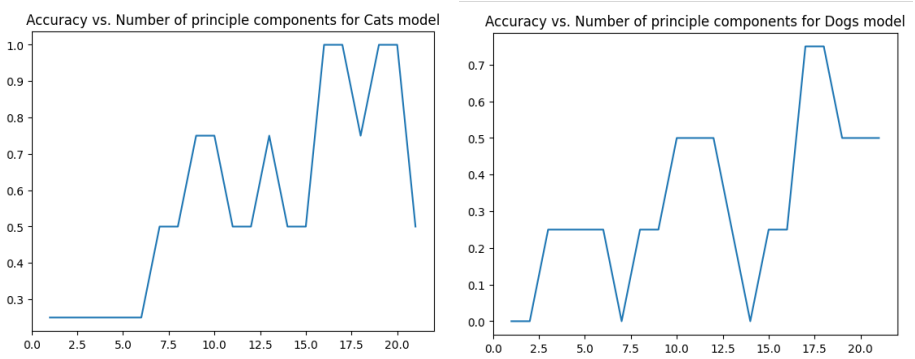

Figure 3: Accuracy vs. Number of principle components for both the Cats and Dogs Eigenface models

Training Set	Misclassification Rate
Cats	0%
Dogs	25%
Combined	67%

Figure 4: Misclassification rates for the different Eigenface models

the average image for the category is calculated and subtracted from each vector. These vectors are then put together into a matrix A, the covariance matrix is found, and then eigendecomposition is performed on the covariance matrix. I did this using sklearn's PCA and SVC functions. The optimal number of eigenfaces needed is a hyperparameter that I've tuned experimentally.

Based on the above figure, I chose 17 principle components for both models. Since my dataset is so small, it could be that I'm capturing a lot of random variation of my specific training/testing images and that my model is overfit. Ideally, I would create a much larger dataset to train this model with to get a more accurate and certain result.

To test the model on new data, the test vector is projected onto each category and assigned to whichever category gives the smallest projection residual. We can also determine a tolerance level for the residual such that if the smallest projection residual is greater than the tolerance level, no category is assigned.

As evidenced by Figure 4, the models trained to specific species perform much better than the combined model.

Fisherfaces

For the Fisherfaces algorithm, I used the same training dataset as with the Eigenfaces algorithm, with the FisherFaceRecognizer algorithm from Open CV[7].

The Fisherface algorithm calculates the average image for each class and subtracts the average from

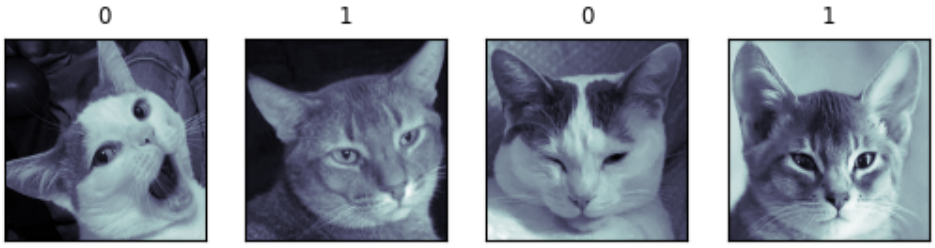

Figure 5: Testing results for the cats Eigenface model

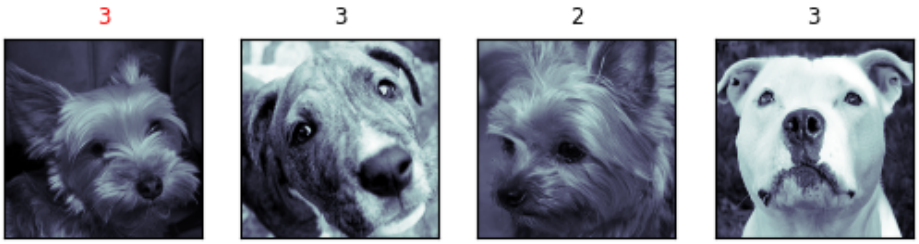

Figure 6: Testing results for the dogs Eigenface model, where a red number indicates misclassification

Figure 7: Testing results for the combined Eigenface model, where a red number indicates misclassification

Training Set	Misclassification Rate
Cats	20%
Dogs	40%
Combined	56%

Figure 8: Misclassification rate for the Fisherface algorithms

Figure 9: Testing results for the cats Fisherface model, where a red number indicates misclassification

each vector, as with eigenfaces. Then, the within class differences are estimated with the within-class scatter matrix, and the between class differences are estimated using the between-class scatter matrix. Combining these two matrices into the scatter matrix, eigendecomposition is then performed on the scatter matrix and choose the top K eigenvectors.

I didn't achieve a high rate of success with the Fisherface algorithm, and it performed overall worse than Eigenfaces.

YOLO

I had initially wanted to train a FaceNet algorithm, but I decided instead to leverage the YOLOv8 algorithm and train it with the breed classes from the Oxford Pet Dataset to see if I could achieve good results without having to crop the image to show just the face. To train the FaceNet algorithm, I would've needed cropped images and my manual dataset is just not large enough to achieve good results with this algorithm. In the future I hope to spend time building out the dataset so that I can test FaceNet's effectiveness.

Using YOLO for image classification means it's performing two tasks. First the algorithm is performing object detection to find the object in the image, by dividing the image into a grid and giving each square in the grid a probability that it contains an object, and using that to create the bounding boxes. At the same time, the algorithm is predicting the probability of each class being present in each bounding box. Afterwards, bounding boxes with a low likelihood are discarded. YOLO (you only look once) is different than other Neural Networks because it performs these tasks at the same time and is much faster.

I split the Oxford pet dataset into train, test, and validation datasets, and set up the directory as required to train YOLOv8, with individual folders for each class. After training with 5 epochs, looking only at the top predicted category, the model had an accuracy of 82.7% and a misclassification rate of

Figure 10: Testing results for the dogs Fisherface model, where a red number indicates misclassification

Figure 11: Testing results for the combined Fisherface model, where a red number indicates misclassification

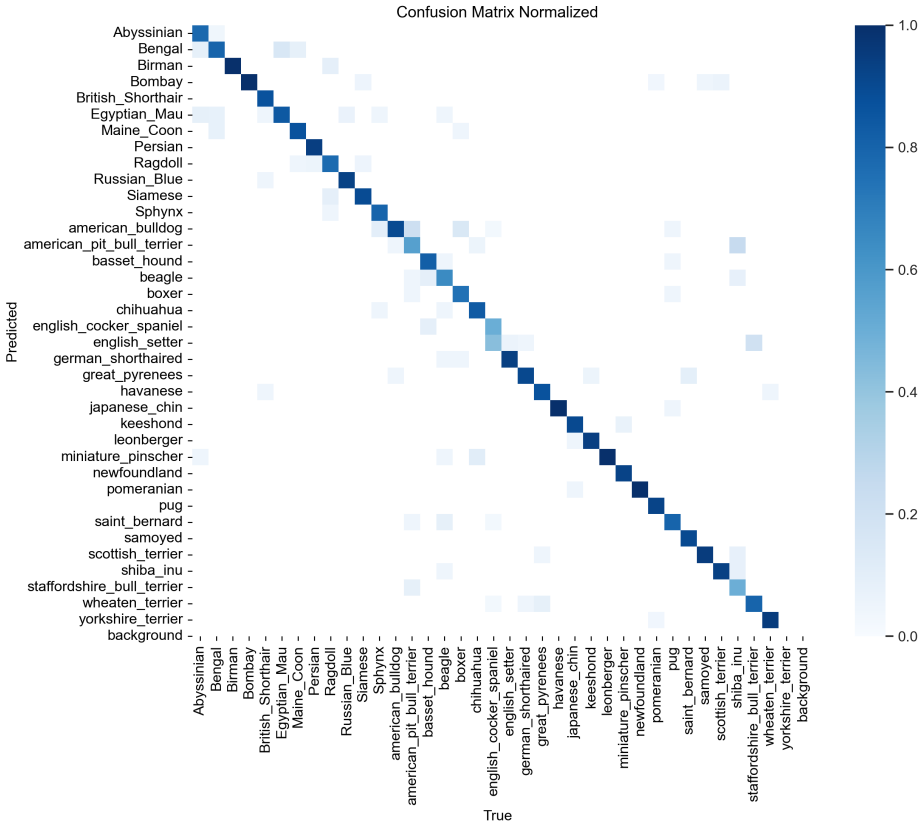

Figure 12: Confusion Matrix for the YOLOv8 Model on the training dataset

Figure 13: YOLO predictions for my cat, who doesn't fit neatly into any breed category

17.2%, much better than either of the previous models. When looking at the top 5 predicted categories, the model was able to predict the correct category in the top 5 97.7% of the time, which is great for my use case since I've established that it's much more important to have the correct prediction in the top N results, rather than trying to prevent any misclassification at all.

This model still needs work for the use case however, because it has trouble with animals that are mixed breeds and don't fit perfectly into the categories. I tried running the prediction on 10 different images of my cat for example, who doesn't have an exact breed. I was expecting the results to be similar from image to image, so that I would still be able to identify my cat based on the breeds that were predicted, but the predictions were wildly different, as shown in Figure 13.

Evaluation and Final Results

For my project, my primary concern was ensuring that at least one of the first N results obtained is correct (N being small, between 3 and 5), rather than eliminating the occurrence of incorrect results altogether. I think the YOLOv8 model that I trained could work really well for the use case I'm exploring, because when a user uploads an image of their lost pet, the algorithm could make a prediction about which breed class the pet belongs to. Then, the application could present users the images of found pets that had the most similar classification (based on the model prediction) to the lost pet.

Based on the results for Eigenfaces and Fisherfaces, it's clear that combined models perform much worse than models that are trained for a specific species, so for future work I intend to focus on more specific species-focused models. I also noticed that the cat models performed marginally better than the dog models. My hypothesis is that this is because of the ear shape: cats almost always have triangle-shaped ears, but the dog classes I had chosen had a wider variety of ear-shapes even within the same class. Determining how big of a role ear shape plays could be a future area of study.

Eigenfaces performed better than Fisherfaces overall, but neither provided the results I had hoped for. Since these models are best suited for head-on face images where facial features remain in the same part of the image, I don't think these models are very suitable to this use case. The training images I used could have had some pre-processing improvements, most notably rotating them so they are all at the exact same angle could be very helpful and could increase accuracy. Ultimately though, for the case of taking pictures of stray or runaway pets in the wild, these models are not suited because it would be difficult in that environment to get clear, head-on images of the pets faces. I've discovered through this research that although I did find that the angle of the images matters in the Eigenface and Fisherface algorithms, I require an algorithm that is more robust to changes in the image angle.

Algorithm	Training Set	Misclassification Rate
Eigenfaces	Cats	0.0%
Eigenfaces	Dogs	25.0%
Eigenfaces	Combined	66.7%
FisherFaces	Cats	20.0%
FisherFaces	Dogs	40.0%
FisherFaces	Combined	55.6%
YOLOv8	Combined	17.2%

Figure 14: Results table for all trained models

YOLO performed the best by far, but didn't perform well when tested with images of pets that didn't fit neatly into the breed categories. To tackle this issue and imrpove the accuracy of the model, I could add more breeds to the dataset, or create categories based on other characteristics. For example, cats are often referred to in terms of their markings (calico, tortoiseshell, etc.). It might also be worthwhile to try a model that performs clustering rather than classification, so that the model will only output similar images without the need for an exact label. One other option I could try is a Siamese Network[9], which is a Neural Network that's made up of two identical sub-networks and is used to compute similarity between images.

I had intended to identify the optimal value of N (the number of images a user should be shown to maximize the likelihood that they will find a match for their lost pet), but I believe this will require further study. This is a very important aspect for this model to be used in the practical application I'm looking at, but more work still needs to be done on the model itself before I can reach that point in my research. In the future, by trying different values of N, I can identify the optimal threshold that balances two crucial factors. The chosen value should not be too large, as it may overwhelm users and prevent them from seeing all the results. At the same time, the value should yield the highest probability of returning a correct match.

References

[1] COCO Dataset. (2017). Retrieved from https://cocodataset.org/#home

[2] Aquino, B. (2021). Coco 2017 Cats [Dataset]. Roboflow Universe. Retrieved from https://universe.roboflow.com/breno-aquino/coco-2017-cats/dataset/1

[3] Oxford Visual Geometry Group. (2012). The Oxford-IIIT Pet Dataset. Retrieved from https://www.robots.ox.ac.uk/~vgg/data/pets/

[4] Toronto Humane Society. (n.d.). Retrieved from https://www.torontohumanesociety.com

[5] Explore YOLOv8. (2023). Retrieved from https://yolov8.com/

[6] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. CoRR, abs/1503.03832. Retrieved from http://arxiv.org/abs/1503.03832

[7] OpenCV. (n.d.). FisherFaceRecognizer Class - OpenCV Documentation. Retrieved from https://docs.opencv.org/3.4/d2/de9/classcv_1_1face_1_1FisherFaceRecognizer.html

[8] Nyoki-mtl. (n.d.). keras-facenet. GitHub. Retrieved from https://github.com/nyoki-mtl/keras-facenet

[9] Zubizarreta, I., Gurrutxaga, I., Pertusa, S., Otaegui, A., Garay-Vitoria, N., Astigarraga, A. (2020). Using a Convolutional Siamese Network for Image-Based Plant Species Identification with Small Datasets. Sensors (Basel, Switzerland), 20(7), 1938. https://doi.org/10.3390/s20071938