Combining Deep Learning Techniques for Improved Cat Face Recognition

Joseph Hersh and Ashwin Ramesh

May 8, 2023

1 Introduction

Cats, ubiquitous companions in human society since Neolithic times, now number over 200 million globally [1]. They have become integral members of many households, offering companionship and emotional support. However, the increasing population of domesticated cats has also led to a growing concern - missing cats [1]. In America alone, 15% of cat owners lose their pet at least once in a five-year period, and many of these cats are never reunited with their owners [3]. This loss not only causes emotional distress to the owners but also contributes to the burden on animal shelters and, in many cases, results in euthanasia [4].

Current methods to find lost pets, including neighborhood searches, owner-initiated trapping, and the use of identification tags, have not been significantly effective, particularly for cats [4]. Automated visual recognition, while a potentially convenient, highly available, and low-cost solution, has been notably overlooked in addressing this problem [3]. Previous studies have demonstrated the feasibility of image-based identification for pets and individual recognition of animals like dogs and livestock [1] [2] [3]. However, the application of these techniques to cat face recognition remains under-explored.

Drawing from techniques and methodologies used in other animal face recognition studies, our work aims to advance the field of cat face recognition. These techniques, including computer vision-based face recognition and non-invasive biometrics, have shown promise in animal identification, offering more humane and less time-consuming alternatives to current practices [2]. By exploring the application of these methods to cats, we aim to enhance the efficacy of locating missing cats and ultimately, contribute to improved recovery percentages.

The goal of this paper is to address the gaps in the literature by investigating the application of these techniques to cat face recognition. We hope that the insights gained from this study will lead to more effective searches for lost cats, thereby aiding in their reunion with their owners and reducing the number of unclaimed cats in shelters and municipal facilities [4].

2 Methods

2.1 Data Collection and Preparation

Figure 1: Dataset Overview

Our dataset consists of 15-30 second 1080p 30fps videos of 12 cats. These videos were converted into frames, and the cat faces were extracted and cropped using an OpenCV face cascade. We used an ImageDataGenerator from the Keras library to augment the images, incorporating various parameters such as rotation, width and height shift, brightness, shear, zoom, and channel shift to generate more robust training data.

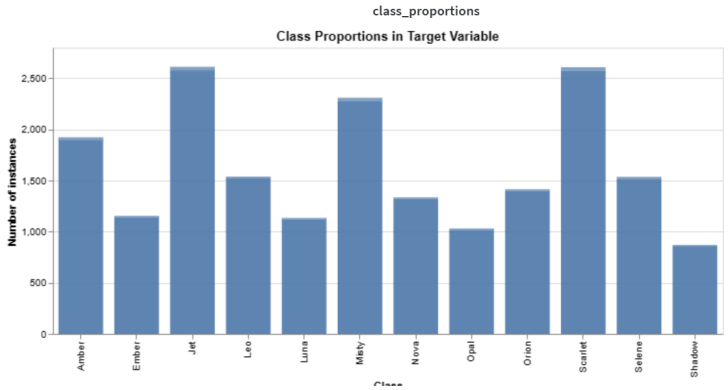

Figure 2: Training Instance Showing Class Proportions

We ensured that each class in the dataset has a minimum of 60 and a maximum of 150 images prior to augmentation. It randomly samples images from each class and splits the dataset into training and testing sets. As noted in [5],

since our data sets were built from frames extracted from video, the similarity between images in our training and validation sets might be too small. To address this, our code also includes a function to remove similar images based on a perceptual hashing (pHash) algorithm. The similarity threshold is set to 5 for the pHash comparison, but it can be adjusted as needed.

We built two models to compare different implementations. The first model was a Convolutional Neural Network (CNN) using Keras to classify the cat faces. The input to the model is the cropped cat face images. The model consists of several layers:

- Convolutional layers for feature extraction. We used ReLU (Rectified Linear Unit) as the activation function and applied L2 regularization to prevent overfitting.
- Batch normalization layers after each convolutional layer to stabilize the learning process and reduce the number of training epochs needed.
- Max pooling layers for downsampling the input along its spatial dimensions (height and width).
- Dropout layers after each max pooling layer to prevent overfitting by randomly setting a fraction of input units to 0 at each update during training time.
- A flatten layer to convert the 2D matrix data into a vector.
- Two fully connected (dense) layers. The first has ReLU activation and the second has softmax activation for outputting probabilities for each class.

The model was compiled using the Adam optimizer, categorical cross-entropy as the loss function, and accuracy as the performance metric.

The second model was a Support Vector Machine implementation based on [2]. The input for the SVM were the extracted features of each class's image.

To extract each image's features, we first used a pre-trained VGG Face model (specifically VGG16) which was loaded in via the Keras library (keras.applications.vgg16) using python 3.10. The model's weights were a preset from Imagenet. When initializing the model, the output was set to the second to last layer of the CNN (fc2). This represents the layer in which the last round of feature extraction occurs before the model starts attempting to classify the data. These features were then appended to a Numpy array along with the class labels on a separate column.

After the features were extracted, we used Scikitlearn's SVM package to train an SVM with a linear kernel. We added an additional parameter called probability and set it to be true. This would help us later down the line in finding the percent confidence for each model prediction.

2.2 Training

Both models were trained on a Google Colab Pro notebook with 12.7 GB of memory and a Tesla T4 GPU. Due to the high memory consumption during training, we encountered frequent crashes, which we managed by adjusting batch sizes and utilizing checkpoints to save the model after each epoch.

For model tracking and performance visualization, we used the Weights & Biases (wandb) tool. It provides useful insights and visualization tools for machine learning experiments. We set up a callback to log metrics and losses to the wandb dashboard for real-time monitoring and analysis.

3 Results

In this experiment, a total of 3845 cat face images were processed, all of which were cropped from videos using our model. The CNN model was trained over 15 epochs and the accuracy improved from 73.33% in the first epoch to 98.37% in the final epoch.

The model's performance in each epoch can be represented in the following graphs:

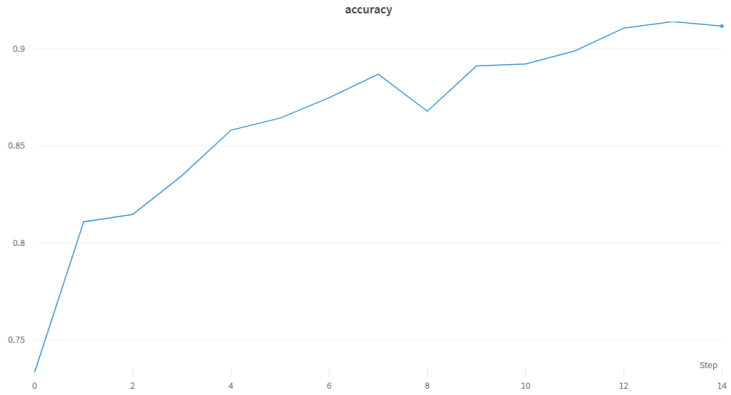

Figure 3: Training Accuracy over Epochs

During testing, the CNN model achieved an accuracy of 98.37% with a loss of 1.5701. The performance of the model in classifying each cat class in the test set is represented in the following confusion matrix:

The breakdown of each cat's individual accuracy, precision, recall, and support scores is as follows:

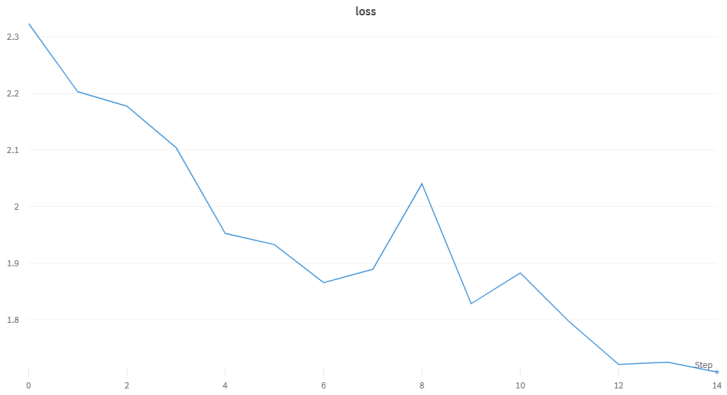

Figure 4: Training Loss over Epochs

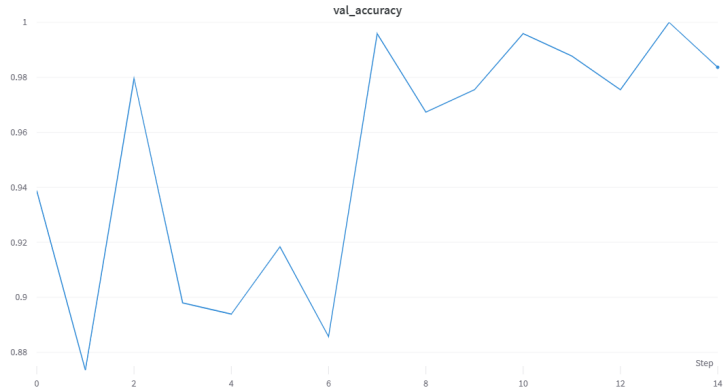

Figure 5: Validation Accuracy over Epochs

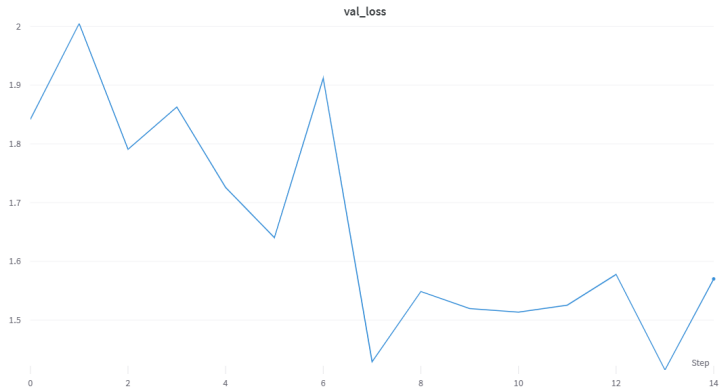

Figure 6: Validation Loss over Epochs

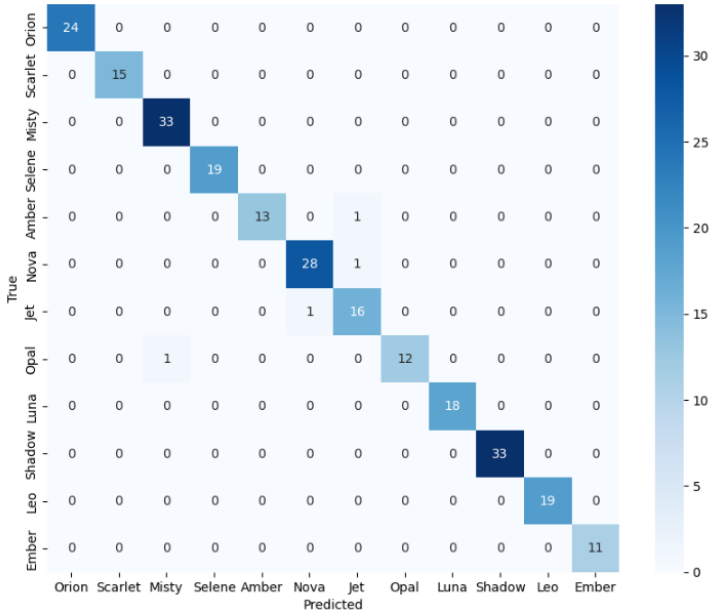

Figure 7: Confusion Matrix of Cat Classification

Class Label	Precision	Recall	F1-score	Support
Orion	1.00	1.00	1.00	24
Scarlet	1.00	1.00	1.00	15
Misty	0.97	1.00	0.99	33
Selene	1.00	1.00	1.00	19
Amber	1.00	0.93	0.96	14
Nova	0.97	0.97	0.97	29
Class Label	Precision	Recall	F1-score	Support
Jet	0.89	0.94	0.91	17
Opal	1.00	0.92	0.96	13
Luna	1.00	1.00	1.00	18
Shadow	1.00	1.00	1.00	33
Leo	1.00	1.00	1.00	19
Ember	1.00	1.00	1.00	11
Accuracy			0.98	245
Macro Avg	0.99	0.98	0.98	245
Weighted Avg	0.98	0.98	0.98	245

Table 1: Precision, Recall, F1, and Support Metrics for Each Cat Via CNN Model

For the SVM, during testing, the model achieved a 100% accuracy score. Multiple steps were taken in ensuring the model was properly predicting each cat and ensuring that the model wasn't over-fitted. First, we used Weights & Biases to plot the confusion matrix for the SVM. This would help us figure out if there were any wrongly classified instances per cat

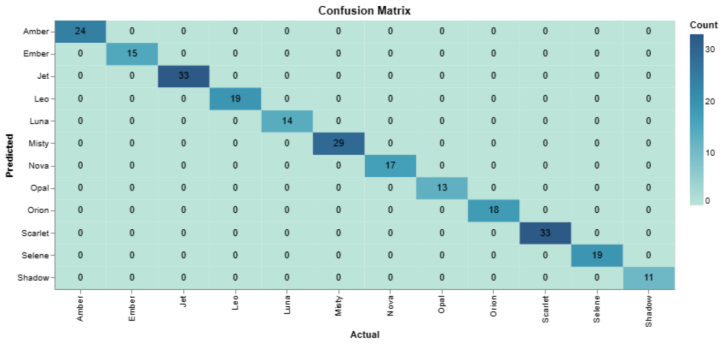

Figure 8: Confusion Matrix for Support Vector Machine Cat Classification

As evident by the figure, it predicted correctly every time. The next thing we looked at was the confidence rating for each classification the model made. For SVMs, the confidence rating is the probability that a given data point belongs to the predicted class. We used Scikitlearn's metrics package to find this probability for each prediction. We found multiple instances where the confidence was split between a few different classes. This indicates that the test and train data weren't the same and shows that there were data points that had a resemblance to multiple different classes.

The last test we ran was a cross-validation test, in order to be absolutely certain that the test and train datasets were not the cause. The accuracy was 99.97%.

Method	**Accuracy**	**F1 score**
CNN Based Model	98.37%	99.2%
SVM Based Model	100%	100%
SVM Based Model (with Cross-Validation)	99.97%	100%

Table 2: Accuracy and F1 score for two face-recognition models.

In summary, accuracy scores, precision, and F1 scores were all 100%. That only dipped by 0.03% when adding cross-validation. based on the tests done above, it seems unlikely that overfitting is the main source of the high accuracy rate. The VGG16 model appears to be very effective at extracting relevant features while the SVM classifier does an excellent job at classifying the cat faces based on these features. We believe that this classification task may have been trivial as a result of a sufficiently small set of classes compared to similar

research [1].

4 Conclusion

In this paper, we aimed to investigate the efficacy of using computer vision to identify cats in the medium of video. The study utilized a dataset of 1080p 30fps videos of 12 cats. Two models were built and trained, a Convolutional Neural Network (CNN) and a Support Vector Machine (SVM) implementation. The CNN model used ReLU activation, L2 regularization, batch normalization, max pooling, and dropout layers. The SVM implementation used pre-trained VGG-based CNN to extract features of each class's image, which were then classified using an SVM with a linear kernel. Both models showed promising results and were extremely cheap to build and train. This study has implications for the future of searching for lost pets, thereby aiding in their reunion with their owners and reducing the number of unclaimed cats in shelters and municipal facilities.

References

[1] 2018 1800316.(doi:) Authors,Tzu-Yuan Lin, Yan-Fu Kuo.: Cat face recognition using deep learning. ASABE Annual International Meeting (2018). https://doi.org/10.13031/aim.201800316

[2] Mark F. Hansen, Melvyn L. Smith, Lyndon N. Smith, Michael G. Salter, Emma M. Baxter, Marianne Farish, Bruce Grieve, Towards on-farm pig face recognition using convolutional neural networks, Computers in Industry, Volume 98, 2018, Pages 145-152, ISSN 0166-3615, https://doi.org/10.1016/j.compind.2018.02.016. (https://www.sciencedirect.com/science/article/pii/S0166361517304992)

[3] Moreira, T.P., Perez, M.L., Werneck, R. et al. Where is my puppy? Retrieving lost dogs by facial features. Multimed Tools Appl 76, 15325–15340 (2017). https://doi.org/10.1007/s11042-016-3824-1

[4] Huang L, Coradini M, Rand J, Morton J, Albrecht K, Wasson B, Robertson D. Search Methods Used to Locate Missing Cats and Locations Where Missing Cats Are Found. Animals. 2018; 8(1):5. https://doi.org/10.3390/ani8010005

[5] Wada, N., Shinya, M., &; Shiraishi, M. (2013). [short paper] pig face recognition using eigenspace method. ITE Transactions on Media Technology and Applications, 1(4), 328–332. https://doi.org/10.3169/mta.1.328