Pet Cat Face Verification and Identification

Adam Klein
CS230 Fall 2019
Stanford University
adamdk@stanford.edu

Abstract

We investigate using deep convolutional neural networks to detect, verify and identify individual domestic cat faces in digital images. We describe a moderately sized data set of cat images sourced from internet pet adoption profiles. We report on training YOLOv3 to detect cat faces and EfficientNet with regularized triplet loss to generate 64-dimensional embeddings for verification and identification. We obtain state-of-the-art accuracy of around 95% on the verification task and a less impressive but still strong 81% on the rank-5 identification task using an open-set protocol with cats not seen during training. In contrast to prior approaches, face landmark tagging and pose alignment are not used.

1 Introduction

Deep learning approaches have yielded impressive achievements in human face recognition and verification [1]. There has been far less research into applying deep learning to individual animal recognition. Existing literature on household pet identification primarily concerns individual dog identification [2], [3].

There are approximately 76.8 million dogs and 58.4 million cats living as companion animals in households in the United States alone [4]. Many of these pets are microchipped by their owners so that the pets may be identified in case they run away. This procedure costs around forty-five US dollars, which includes registration in a database [5]. In contrast, a digital image-based pet registry would be less costly and more convenient to operate, thereby lowering the barrier to registering a pet.

Our work offers the following contributions. We describe a novel data set of over 130,000 individual images of over 23,500 distinct cats sourced from online pet adoption profiles. We report on training YOLOv3 [6] to detect and extract cat faces from source images, forgoing manual feature selection and image alignment, in contrast to prior approaches which preprocess images to align facial features and pose. Finally, we report on training EfficientNet [7] for verification and recognition tasks, using a regularized triplet loss function and hard triplets that are mined in an online fashion.

2 Related Work

Meugot et al [3] recently reported that a ResNet-like network architecture trained using the triplet loss from FaceNet [8] was able to verify dog faces with an accuracy of 92% and identify dogs with 74% rank-1 and 96% rank-5 accuracy on open-set data. The authors used a small data set of 3148 images of 485 dogs of which 298 images of 48 dogs were held out for testing. The authors hand-labeled the eyes and muzzle within each dog image in order to programmatically align faces during image preprocessing. As of December 2019, the authors have grown their data set to 8600 images, but have seen their verification accuracy decrease to 86% on account of having more animals of the same breed in the data [9]. It stands to reason the identification accuracy is likely also lower.

Lin and Kuo [10] focus on individual cat identification, and train a CNN to perform cat facial feature detection (eyes, nose, forehead, mouth), but then rely on traditional machine learning techniques (PCA and SVMs) to perform identification. They also use a small data set of 1500 images of 150

CS230: Deep Learning, Fall 2019 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

cats. It is not clear from the paper whether their reported identification accuracy of 94.1% excludes individuals or images seen during training.

3 Data Set

We sourced images from petfinder.com [11] by using their developer API to crawl cat adoption listings within a 500-mile radius of seven major US Cities: New York, Chicago, Miami, Los Angeles, Dallas, St. Louis, and San Francisco. We scanned approximately 570,000 distinct cat profiles. Of these, we selected the approximately 24,000 profiles that had at least five profile pictures. (Each profile had at most six images.) We downloaded the original-size images for these cats. We then used an MD5 hash of each image file to discard duplicates, eliminating redundant profiles or pictures. We were left with around 23,657 distinct cats with at least five pictures. (We say "around" and "approximately" because we cannot be sure there are no duplicate profiles or images that evade our heuristic checks.) Images are highly varied in quality and information content as can be seen in Figure 1. Profile pictures are known to include multiple cats, odd poses, an incorrect cat, and occluded or poorly captured subjects.

Figure 1: Quality problems, from left to right: (1) profiled cat's face is hidden; (2) profiled cat is not pictured; (3) profiled cat is second from top; (4) profiled cat is not showing face.

4 Face Detection

To detect and extract cat faces, we initially tried two pre-trained cat-face detectors: one utilizing a Haar Cascade classifier [12], and one a FHOG-based SVM approach [13]. Simple inspection revealed both detectors suffered from a significant false negative detection (high miss) rate on our data.

We found success in using transfer learning to train an implementation of YOLOv3 [6] for the task. We selected 660 images at random and used VoTT [14] to draw bounding boxes around cat faces as in [15]. We used 594 images for training and the rest for validation. We initialized the network with the official Darknet weights pre-trained on COCO [16]. We first trained the final three layers for a single class detection task for 50 epochs using an Adam optimizer with learning rate 10^{-3}, and then unfroze all layers and trained for another 50 epochs with learning rate 10^{-4}, using a learning rate reduction factor of 0.1 on validation loss plateau with patience of 3. The trained model demonstrates 100% mAP on on the validation set with a 49% confidence threshold [17]. As can be seen in Figure 2, the model is empirically adept at detecting cat faces across arbitrary poses, although lower confidence levels show some puzzling false positives.

Figure 2: YOLOv3 Cat Face Detection. Note false positive in far right picture: remote control identified as cat with 25% confidence.

5 Data Preparation

We ran our face detector with a 49% confidence threshold over the original images, skipping those that have either zero or multiple hits (the latter will yield invalid faces unless the input image is a collage of a single cat). Cats having less than five images remaining were then excluded from the

data. We resized each image to $max(height, width) = 256$ in 3 color channels using a Lanczos filter, maintaining the aspect ratio, and padding the border with black pixels if necessary. These images were then partitioned into 86867 training images of 15949 cats, 4803 validation images of 886 cats, and 4794 test images of 887 cats. The data include both easier and more challenging material (Figures 3 and 4 respectively show this for the validation data). There is also at least one image of a dog face and one of a human face in the validation data.

Figure 3: Example images for cat 12

Figure 4: From left to right: first six cats are 8203; seventh is 16107; eighth is 16134

6 Methodology

For the verification and identification tasks, we began by attempting to train from scratch the ResNet-like CNN architecture in Meugot et. al [3], and then moved on to a transfer learning approach using Inception-ResNet-V2 [18] and finally to EfficientNetB2 [19], the last two of which were pre-trained on ImageNet [20]. We connect the final global average pooling layer to a fully-connected layer with 64 outputs with identity activation which are then L^2-normalized. This outputs a 64-dimensional embedding given an 256x256x3 input image. We utilize a triplet loss objective function [8] with a "global orthogonal regularization" term per Zhang et al [21]:

$$
\begin{aligned}
\ell_{final} &= \ell_{triplet} + \alpha \ell_{gor} \\
\ell_{triplet} &= \max\left(0, \epsilon - \left\| f(\mathbf{x}_i) - f(\mathbf{x}_i^-) \right\|_2^2 + \left\| f(\mathbf{x}_i) - f(\mathbf{x}_i^+) \right\|_2^2 \right) \\
\ell_{gor} &= M_1^2 + \max(0, M_2 - \frac{1}{d}) \\
M_1 &= \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x_i})^T f(\mathbf{x_i}^-) \\
M_2 &= \frac{1}{N} \sum_{i=1}^{N} \left(f(\mathbf{x_i})^T f(\mathbf{x_i}^-) \right)^2
\end{aligned}
$$

The regularization term penalizes embeddings of randomly sampled non-matching pairs when they are far away from orthogonal, leading to more spread out feature descriptors. Without this term, during training, embeddings quickly collapsed to a single vector, setting the loss to be equal to the margin ϵ. We chose the values of the hyper-parameters ϵ, α and d to be 0.4, 1.1 and 64 respectively. Other values tried were $\alpha \in \{0.9, 1.0\}$ and $\epsilon \in \{0.1, 0.2, 0.3\}$

We trained the EfficientNetB2 network for 150 epochs in five stages. A single epoch consists of iterating over all classes (unique cats) in the training set, which are shuffled prior to training. Each iteration consists of randomly generating (anchor $\mathbf{x}_i$, positive $\mathbf{x}_i^+$, negative $\mathbf{x}_i^-$) triplets from $batchsize/4$ classes in an online fashion. The first stage is training only the final fully-connected layer for 10 epochs with a batch size of 768 triplets. The second stage is training layers from the final block (block seven) for 30 epochs with a batch size 612. The third, from block six for 30 epochs with batch size 384. The fourth consists of training all layers for 35 epochs with batch size 48. And fifth, for 45 epochs with batches sampled from classes clustered every five epochs according to the euclidean distances of the class centers, forcing the batches to contain similar classes [22]. The "batch hard" online triplet mining strategy as proposed in Hermans et al [23] is used; the implementation was

provided by Moindrot [24]. We used an Adam optimizer with default parameters and learning rate of 10^-3 for the first 70 epochs, followed by an exponentially decaying learning rate to 10^{-7} over the next 80 epochs. Note the sharp drop in accuracy at epoch 105 corresponding to switching to clustered class training.

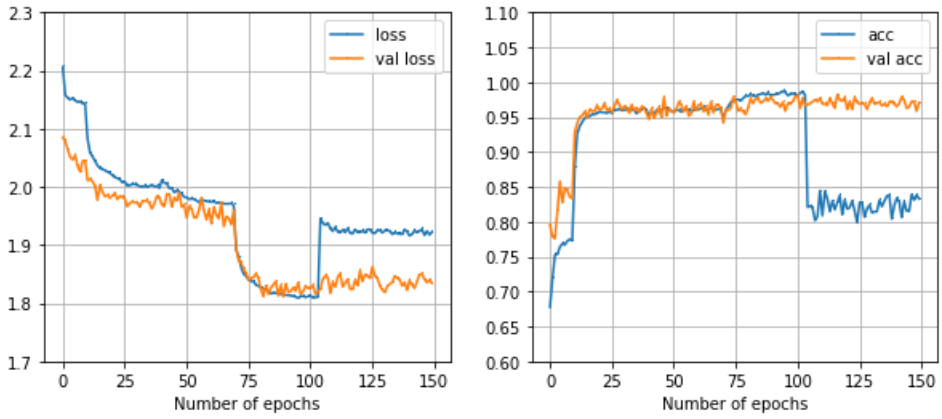
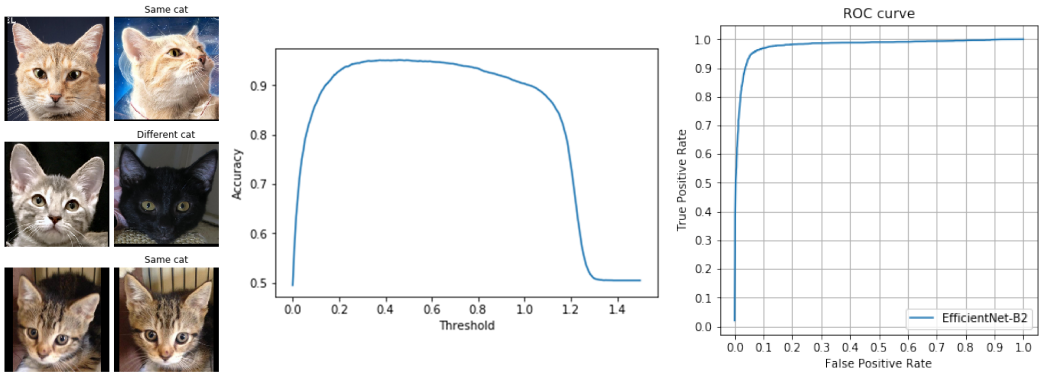


Figure 5: Training EfficientNetB2

7 Cat Face Verification

10,000 pairs of validation cat face images were generated with equal likelihood to be either same or different cats, as in Figure 6a. An optimal L_2^2 distance threshold of 0.455 was chosen to differentiate cats according to their embedding vectors, leading to a validation accuracy of 95% (Figure 6b). We plot the ROC curve in Figure 6c.



(a) Example cat pairs (b) Embedding distance threshold vs accuracy (c) ROC Curve

We then used 10,000 pairs of cats generated in the same way from the test data to perform the verification task. The resulting confusion matrix (Figure 7) reveals an accuracy of around 94.6% on data containing no previously observed cat classes. This number is conservative; Appendix A contains examples of both false positive and negative mistakes on the test data, which by inspection can be seen to include “mistakes" where the network is actually correct and the “ground truth" is mislabeled.

	Same	Different
Accept	47.31%	2.46%
Reject	2.93%	47.30%

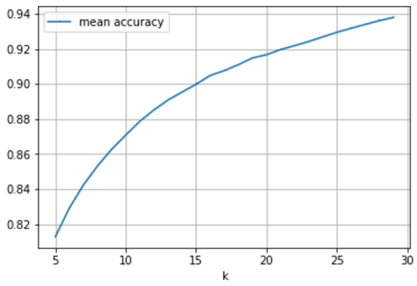
Figure 7: Confusion matrix with false positive (Different/Accept) and false negative (Same/Reject) error rates

8 Cat Face Identification

In the identification task, we use some test data to establish a database of cat identities and then try to associate the remaining test data with the correct classes. We follow the methodology in Meugot et al [3]: in each trial, we randomly split each cat class c containing N_c embeddings into M embeddings to be used in the database, and $N_c - M$ to be used for lookup using k nearest neighbors search (for rank-k identification). We ran 500 trials and report the mean, min, and max accuracy in Figure 8a. We report results for $M = 1$ and $k = 1$ which represents a one-shot recognition task; and then results for $k = M + 1$ for $M = 2, 3, 4$. We also graph the effect of k on accuracy for $M = 4$ with $k \geq 5$ in Figure 8b. We consider these numbers quite strong given the known quality problems in the data set.

		Identification Accuracy		
M	k	Mean	Min	Max
1	1	47.45%	45.18%	49.42%
2	3	70.78%	68.97%	72.52%
3	4	77.19%	75.15%	79.28%
4	5	81.32%	78.89%	83.95%

(a) 500 trials of rank-k identification using M embeddings per class



(b) Effect of k on accuracy

9 Interpretability

We see which areas of the test images draw the attention of the network using class activate maps. We list several examples in Appendix C in which the network appears to pay particular attention to the forehead and eye areas of the cat face images. Interestingly, the ears only seem to play a particular role in one of the ten example images reviewed.

Additionally, k-means clustering over the cat face embeddings of the test data is used to determine which images the network considers similar. We have chosen eight clusters to show in appendix B, from which the reader may see that, subjectively speaking, the network has learned quite well which cats appear alike.

10 Conclusion

We have presented an end-to-end deep learning approach for pet cat verification and identification using face imagery with minimal preprocessing. While there are few prior results against which to measure ours, we are encouraged that even with minimal quality control on the input data, we achieve state-of-the-art accuracy on the verification task and a strong showing on the identification task. In our judgment, the remaining hurdles are not in the chosen architecture or training methodology, but in the quality of the input data. Almost surely, we would achieve better results with curated data which could, for example, be used to maintain a high quality pet registry database.

Bibliography

[1] Mei Wang and Weihong Deng. Deep Face Recognition: A Survey. *arXiv e-prints*, page arXiv:1804.06655, Apr 2018.

[2] Thierry Pinheiro Moreira, Mauricio Lisboa Perez, Rafael de Oliveira Werneck, and Eduardo Valle. Where Is My Puppy? Retrieving Lost Dogs by Facial Features. *arXiv e-prints*, page arXiv:1510.02781, Oct 2015.

[3] Guillaume Mougeot, Dewei Li, and Shuai Jia. A deep learning approach for dog face verification and recognition. In Abhaya C. Nayak and Alok Sharma, editors, *PRICAI 2019: Trends in Artificial Intelligence*, pages 418–430, Cham, 2019. Springer International Publishing.

[4] U.S. Pet Ownership Statistics: Companion Animals. https://www.avma.org/KB/Resources/Statistics/Pages/Market-research-statistics-US-pet-ownership.aspx? [Online; accessed 09-October-2019].

[5] Pet Microchip FAQ. https://www.petfinder.com/dogs/lost-and-found-dogs/microchip-faqs/. [Online; accessed 09-October-2019].

[6] A Keras implementation of YOLOv3 (Tensorflow backend). https://github.com/qqwweee/keras-yolo3/. [Online; accessed 04-November-2019].

[7] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[8] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv e-prints*, page arXiv:1503.03832, Mar 2015.

[9] DogFaceNet. https://github.com/GuillaumeMougeot/DogFacenet. [Online; accessed 03-December-2019].

[10] Tzu-Yuan Lin and Yan-Fu Kuo. Cat face recognition using deep learning. In *2018 ASABE Annual International Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2018.

[11] Petfinder Developer API. https://www.petfinder.com/developers/. [Online; accessed 04-November-2019].

[12] Cat Face Detection using OpenCV. https://blogs.oracle.com/meena/cat-face-detection-using-opencv. [Online; accessed 09-October-2019].

[13] Cat facial detection and landmark recognition in Python. https://github.com/marando/pycatfd. [Online; accessed 10-October-2019].

[14] Visual Object Tagging Tool. https://github.com/microsoft/VoTT. [Online; accessed 04-November-2019].

[15] How to train your own YOLOv3 Detector From Scratch. https://blog.insightdatascience.com/how-to-train-your-own-olov3-detector-from-scratch-224d10e55de2. [Online; accessed 5-December-2019].

[16] YOLO: Real-Time Object Detection. https://pjreddie.com/darknet/yolo/. [Online; accessed 04-November-2019].

[17] mean Average Precision. https://github.com/Cartucho/mAP. [Online; accessed 04-November-2019].

[18] Inception-Resnet-V2. https://github.com/tensorflow/models/blob/master/research/slim/nets/inception_resnet_v2.py. [Online; accessed 4-December-2019].

[19] EfficientNet Keras. https://github.com/qubvel/efficientnet. [Online; accessed 4-December-2019].

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[21] Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. Learning spread-out local feature descriptors. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[22] Zuheng Ming, Joseph Chazalon, Muhammad Muzzamil Luqman, Muriel Visani, and Jean-Christophe Burie. Simple triplet loss based on intra/inter-class metric learning for face verification. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1656–1664. IEEE, 2017.

[23] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[24] Triplet Loss and Online Triplet Mining in TensorFlow. https://omoindrot.github.io/triplet-loss. [Online; accessed 4-December-2019].

[25] Raghavendra Kotikalapudi and contributors. keras-vis. https://github.com/raghakot/keras-vis, 2017.

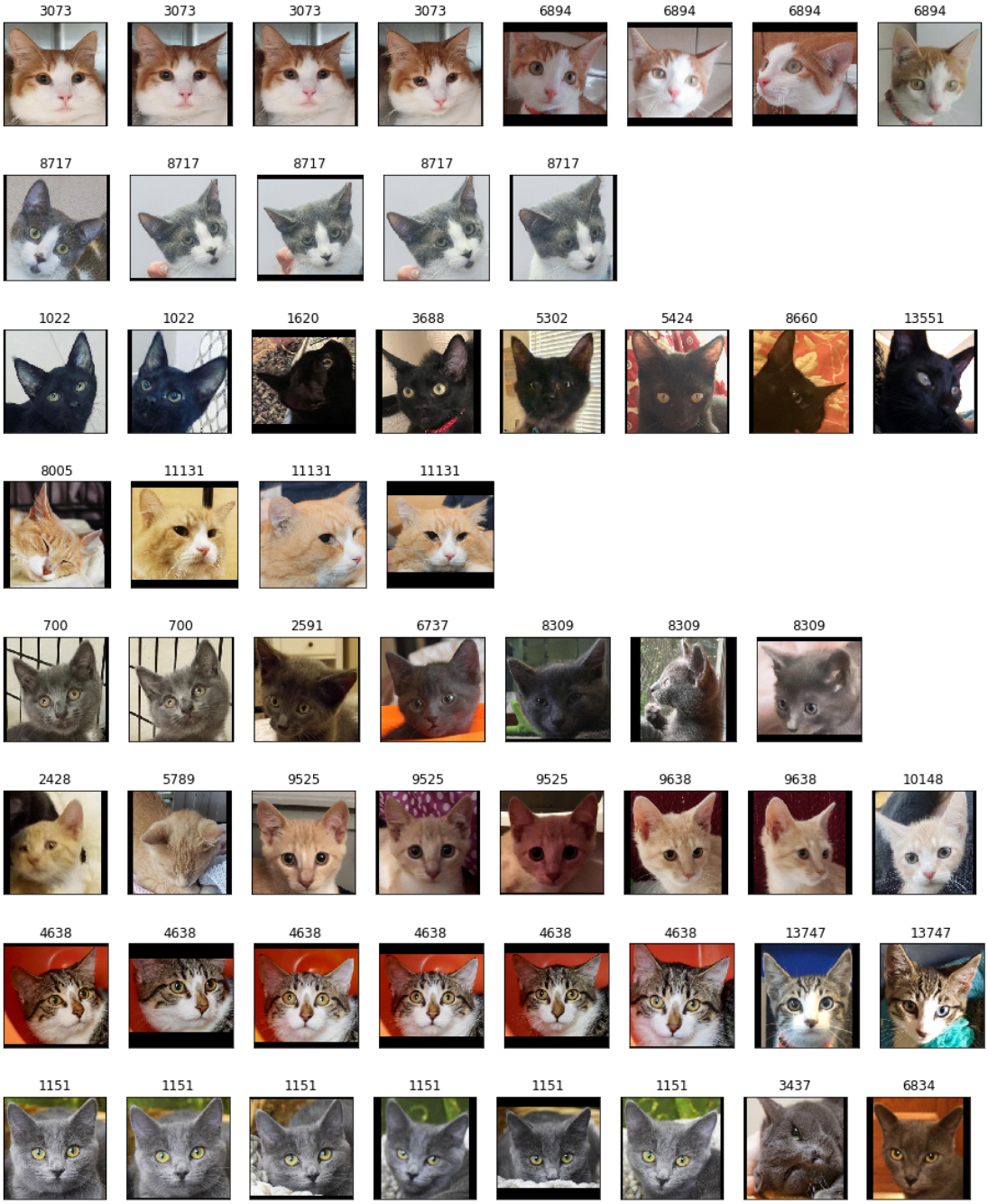
Appendices

A Verification Mistakes

Figure 9: Left pairs are incorrectly accepted as same; right pairs rejected as different. Images are titled with the true cat label.

B Face Clustering

We use k-means clustering on the test embeddings with k=887, the number of unique cats, showing a maximum of 8 images for 10 clusters. This demonstrates which cat images the network considers similar. Images are titled according to cat label.



C Class Activation Maps

We visualize what the CNN is paying attention to with class activate maps of several examples, using the last convolutional layer and the final global average pooling layer. The implementation is provided by keras-vis [25].