Local Species Identification with Deep Learning

Computer Vision
Stanford CS230 - Final Report

Grace Casarez
Computational & Mathematical Engineering
Stanford University
gcasarez@stanford.edu

Maya Frohna
Computational & Mathematical Engineering
Stanford University
mrf52@stanford.edu

1 Background

Animal tracking is a necessary task for wildlife conservation and public safety. Traditional wildlife tracking endeavors rely on camera trap data, which is footage recorded on hiking trails. However, the analysis of this data typically does not take into account the interface between wild and domestic animals, especially those that appear very similar to each other. As a result of climate change and deteriorating ecosystems, wild animals have increasingly ventured in spaces commonly occupied by people and domestic pets [1]. Tracking species that commonly interact with people and property could help researchers gain understanding about these dynamics and infer the health of the animal population. Our goal is to make an image classification system that focuses on commonly found wild and domestic species in California. This classification system could be useful in systems such as home security cameras. If implemented, observations of wild animals could serve both as a safety warning system to homeowners as well as a citizen science contribution to local wildlife tracking agencies.

Previously, computer vision methods have been used to attempt to detect and classify animals from trail camera footage. Attempts at object detection include MegaDetector, which performs robustly on various ecosystems [2]. In our study, we will use MegaDetector as a preprocessing step to isolate animal subjects. There have also been attempts at general species classification. In 2013, Yu et al. developed a species classifier on data from the Netherlands and Panama with an average of 82% accuracy [3]. In 2019, Tabak et al. trained a CNN with ResNet-18 architecture on over 3 million camera trap images, with a 90.4% accuracy and a 82% accuracy on out of sample data. [4]. However, there have been limitations in species classification work. As noted by Willi et al., model accuracy can depend on factors such as camera angle, species abundance, or location [5]. Since ecosystems can vary greatly and there are thousands of animal species, it is difficult to obtain high accuracy for a general species classifier. Additionally, trail footage can be poorly lit or blurry. There has been limited work in developing a model which can specifically distinguish between closely related species, so our aim is to develop a model that can accurately identify and classify similar species that are common in California. We will focus on classifying wild species (coyotes, bobcats, and foxes) and domestic species (cats and dogs) that can appear similar to each other on camera.

2 Dataset

Our dataset includes 23,692 camera trap images of local species of interest: cat, dog, bobcat, fox, and coyote. This dataset was collected by researchers at CalTech as a benchmark dataset for the computer vision task of species identification [6]. Since the footage is from trails near Pasadena, California, the landscape and animals accurately reflect a California ecosystem. To preprocess the data, we used the MegaDetector object detection model on the raw images to identify animals within the frame, then cropped the images using a bounding box, so that the final images only include the animals and not noise from the surrounding environment [2]. 1 shows an example of the preprocessing pipeline, in which the image is passed through MegaDetector and cropped. Some images in the dataset were incorrectly labeled, and were removed if the species was not detected in the image. The resulting images are square and have an image size of $(160, 160)$. 2 shows examples of images after processing.

Using camera trap data has many intrinsic challenges. For example, images may be poorly lit or blurry, especially for images taken at night. Since the footage is collected both during the day and night, some images are grayscale and some are RGB. Additionally, the animals in the image may be far away or too close to the camera, making identification difficult. Home security camera footage poses similar challenges, yet is not as easily accessible online, making camera trap data a useful candidate for a proxy. Another challenge of our dataset was that it was not evenly balanced among species. Some species, notably cat, dog, and fox, had limited observations. 9 shows the distributions of species after preprocessing. After detecting and cropping the images, we have 8,468 coyote images, 6,175 bobcat images, 3,324 dog images, 3,669 cat images, and 2,056 fox images.

Figure 1: Example of image preprocessing pipeline.

Figure 2: Examples from the dataset depicting a bobcat (left), coyote (middle), and dog (right).

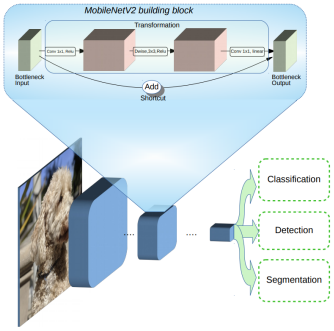
3 Baseline Model

For our baseline model, we are using the MobileNetV2 model [7]. The MobileNetV2 model has been pretrained on the ImageNet[8] dataset, a collection including over 14 million labeled images. Most previous work with camera trap species classification has been done with more computationally expensive models, such as ResNet-18 [5]. However, since we would like our model to eventually be able to be used in production in systems like home security cameras, we opted for a lower cost model. Notably, MobileNetV2 utilizes depth-separable convolution and bottleneck layers. The model replaces the full convolutional operator with a depthwise convolution followed by a 1×1 pointwise convolution, providing lightweight filtering and feature creation. Input and output bottleneck layers save information as the model progresses. 3 shows the basic breakdown of MobileNetV2's architecture. The model has 155 layers total and uses a kernel size of $k = 3$.

The baseline model achieves the results below, with an ending training accuracy of 82.0325%.

4 Methods

To approach our problem, we implement a two step detection and classification system. First, raw images are passed through the MegaDetector object detection model, which locates the animal within the image (if there is one) and surrounds it with a bounding box [2]. These images are then cropped to the region of the bounding box, so that only the animal remains in the frame. We utilize MegaDetector in its default settings as a preprocessing step of the system (see 1 for an example).



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 3: Basic diagram of the building block of MobileNetV2, and a table of the basic architecture of MobileNetV2, with expansion factor t, output channels c, number of times repeated n, and stride s. Table from [7]

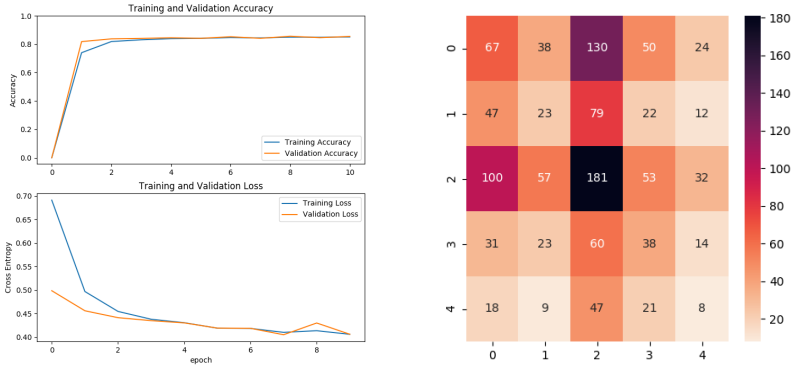


Figure 4: Baseline results: training and validation accuracy and loss, confusion matrix. 0=Bobcat, 1=Cat, 2=Coyote, 3=Dog, 4=Fox

After images are processed and cropped, we split the dataset into training, validation, and testing components. We split with a 90% training, 5% validation, 5% testing breakdown, which seemed like an appropriate choice given the size of our dataset. As a baseline result, we performed transfer learning from the MobileNetV2 model, training only the top two layers to adjust it to our problem. We trained on 10 epochs with a base learning rate of 0.001. We used a batch size of 32. For all training runs, we used a categorical crossentropy loss function with the Adam optimizer.

After getting a baseline result, we added fine-tuning training of the model. We froze the initial portion of the model, then trained the last 35 layers of the model, from the 120th layer onwards. In doing so, we hoped to better tailor the model to our problem. To explore MobileNetV2 even further, we performed various experiments with different choices of hyperparameters. In particular, we explore the influence of the number of epochs and the base learning rate, in order to observe MobileNetV2's performance on our dataset.

In addition to experimenting with features of the model, we also performed data tasks. Since dataset imbalance was a prominent limitation in our original dataset, we executed dataset balancing techniques. During training, we supplied class weights to the model, oversampling images from underrepresented species. We also balanced our test set according to the relative frequency of the species, since we would optimally like the model to perform well on each species of interest. Additionally, we performed data augmentation to effectively increase the size of our dataset, by randomly applying a horizontal flip on the images.

5 Analysis

We implemented 6 experiments in addition to the baseline model.

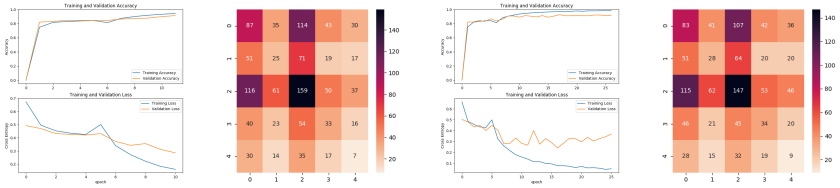


Figure 5: LEFT: **Experiment A**: baseline model with 5 fine-tuning epochs. RIGHT: **Experiment B**: baseline model with 20 fine-tuning epochs. For each set, the left shows training and validation accuracy and loss, and right shows confusion matrix. 0=Bobcat, 1=Cat, 2=Coyote, 3=Dog, 4=Fox

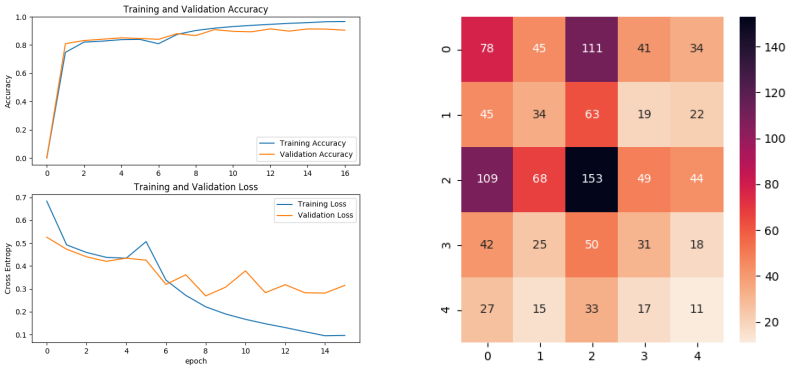


Figure 6: **Experiment C**: baseline model with 10 fine-tuning epochs and class balancing during training. For each set, left shows training and validation accuracy and loss, and right shows confusion matrix. 0=Bobcat, 1=Cat, 2=Coyote, 3=Dog, 4=Fox

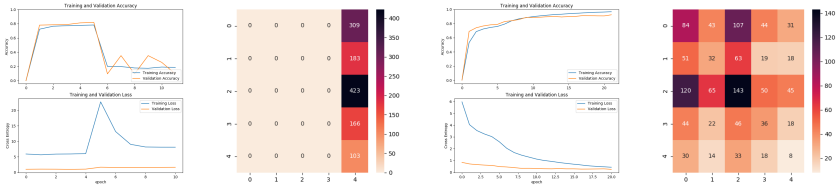


Figure 7: LEFT: **Experiment D**: baseline model with 5 fine-tuning epochs, base learning rate 0.01, and class balancing during training. RIGHT: **Experiment E**: baseline model with 15 fine-tuning epochs, base learning rate 0.0001, and class balancing during training. For each set, left shows training and validation accuracy and loss, and right shows confusion matrix. 0=Bobcat, 1=Cat, 2=Coyote, 3=Dog, 4=Fox

Through the numerous experiments run, we were able to see the effects of altering different parameters and hyperparameters. One of our best results came from the first experiment, where we simply added in a number of fine tuning epochs in order for the model to learn on our specific dataset. To try to improve results, we first increased the number of training epochs; however, rather than providing improvements, this seemed to just overfit our model as our training loss continued to decrease while validation loss stayed constant after about 10 epochs. // The next strategy tested was balancing. In experiment C, we implemented balancing over our datasets, given there was an unequal distribution of number of images for the species. This seemed to give a slight improvement to our previous

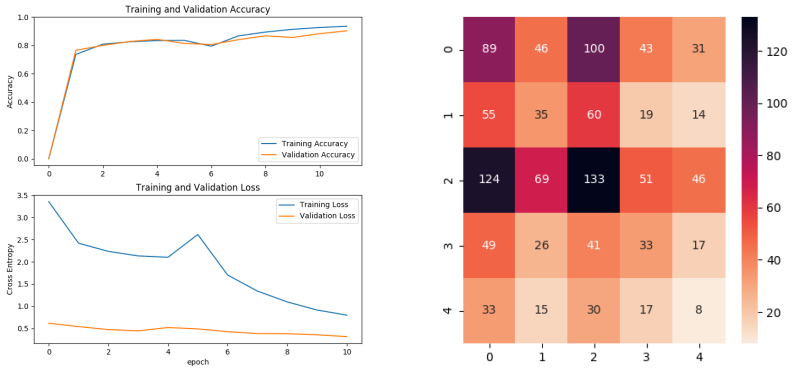


Figure 8: **Experiment F**: baseline model with 5 fine-tuning epochs, image size (224,224), and class balancing during training. For each set, left shows training and validation accuracy and loss, and right confusion matrix. 0=Bobcat, 1=Cat, 2=Coyote, 3=Dog, 4=Fox

models, so the following experiments all included balancing as well.
In experiments D and E, we adjusted the learning rate to try to better fit our model. Increasing the learning rate from 0.001 to 0.01 in experiment D seemed to drastically worsen our models performance, as it ultimately seemed to just predict a single species for every test example. When we decreased the learning rate from 0.001 to 0.0001 and trained on a slightly higher number of epochs, the loss and accuracy plots for the validation and training sets seemed to smooth out; however, the final confusion matrix showed a preference to 2 species - bobcat and coyote. In experiment F, we changed the input image size from (160,160) to (224, 224) and obtained a similar result. Despite attempts to balance the dataset by oversampling during training, it is likely that the model still could not learn well on species that have limited data, because it was not seeing a variety of examples as it was for species with more abundant data.

6 Conclusion

After trying multiple experiments, it appears that the model performed the best for the species with the most available camera trap data, the coyote. However, it struggled to classify species with limited amounts of data. Weighing classes according to their distributions in the dataset during training helped this problem a small amount, but did not fix it completely as the model continued to show preference towards the more common species. Future work on this problem should include obtaining more data on underrepresented species. For example, obtaining household security footage of domestic animals such as cats and dogs would bolster the small amounts of data for domestic animals on camera trap footage.

7 Contributions

Grace: executed preprocessing pipeline (object detection and cropping), organized code and scripts, collaborated on report and video

Maya: obtained and downloaded raw dataset, executed training runs on AWS, collaborated on report and video

References

[1] Erica von Essen, Jonathon Turnbull, Adam Searle, Finn Arne Jørgensen, Tim R. Hofmeester, and René van der Wal. Wildlife in the digital anthropocene: Examining human-animal relations through surveillance technologies. *Environment and Planning E: Nature and Space*, 0(0):25148486211061704, 0.

[2] Sara Beery, Dan Morris, and Siyu Yang. Efficient Pipeline for Camera Trap Image Review.

[3] Xiaoyuan Yu, Jiangping Wang, Roland Kays, Patrick A Jansen, Tianjiang Wang, and Thomas Huang. Automated identification of animal species in camera trap images. *J Image Video Proc*, 53, 2013.

[4] Michael A. Tabak, Mohammad S. Norouzzadeh, David W. Wolfson, Steven J. Sweeney, Kurt C. Vercauteren, Nathan P. Snow, Joseph M. Halseth, Paul A. Di Salvo, Jesse S. Lewis, Michael D. White, Ben Teton, James C. Beasley, Peter E. Schlichting, Raoul K. Boughton, Bethany Wight, Eric S. Newkirk, Jacob S. Ivan, Eric A. Odell, Ryan K. Brook, Paul M. Lukacs, Anna K. Moeller, Elizabeth G. Mandeville, Jeff Clune, and Ryan S. Miller. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, 10(4):585–590, 2019.

[5] Marco Willi, Ross T. Pitman, Anabelle W. Cardoso, Christina Locke, Alexandra Swanson, Amy Boyer, Marten Veldthuis, and Lucy Fortson. Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 10(1):80–91, 2019.

[6] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, volume 11220 of *Lecture Notes in Computer Science*, pages 472–489. Springer, 2018.

[7] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *In 2009 IEEE conference on computer vision and pattern recognition.*, pages 248–255, 2009.

8 Appendix

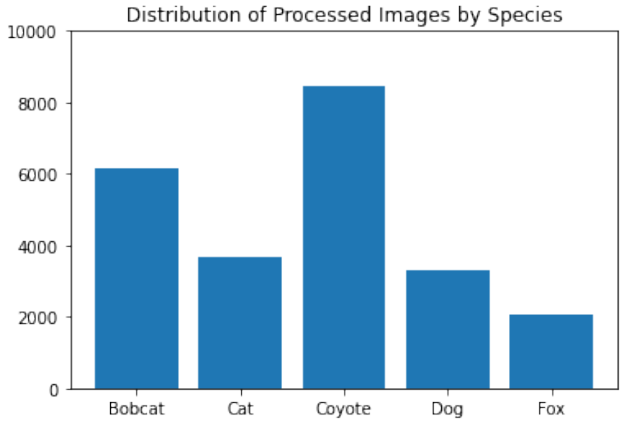


Figure 9: Distribution of species among dataset.