CS230

Final Report for CS230 - Spring 2020

Automatic Species Classification in Camera Trap Images for New Locations

A Computer Vision Project

Emma Dolan `edolan5`
Samantha Koire `skoire`
Gen Woods `genwoods`

Abstract

Camera trap images offer a large amount of data to conservation biologists, including the well-being and survival rate of animals over time, the behavioral and activity patterns of animals, and animal migration patterns. However, manually labeling each image is a laborious task that limits scientists' ability to spend their working hours actually drawing conclusions from the data. In this paper, we utilize data from cameras in 441 locations to attempt to classify 675 different types of animals automatically, even if the animal has not been seen before at that specific location. By running the camera trap images through a pre-trained model to detect animals and passing the focused selection to a CNN using EfficientNet, transfer learning via ImageNet, early stopping, and 26 epochs, we were able to correctly label animals from a new camera location 48.5% of the time.

1 The Problem and Its Challenges

From iWildCam: "Conservation biologists invest a huge amount of time reviewing camera trap images, and – even worse – a huge fraction of that time is spent reviewing images they aren't interested in. This primarily includes empty images, but for many projects, images of people and vehicles are also "noise", or at least need to be handled separately from animals. Machine learning can accelerate this process, letting biologists spend their time on the images that matter."

As a means to measure biodiversity, camera traps (cameras in the wild with infrared triggers) are becoming increasingly common around the world. We will utilize data from cameras in 441 locations to attempt to classify wildlife automatically, even if the animal has not been seen before at that specific location.

How do we train models that perform well on new (unseen during training) camera trap locations? Classifying animals in new locations poses a challenge because the set of species seen in each camera overlap but are not identical. Technical challenges that make image detection difficult include poorly illuminated images, motion blur, small ROI, obstruction, differing weather conditions, camera malfunctions, temporal changes and non-animal variability. A small sample of the training data can be seen below.

CS230: Deep Learning, Spring 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

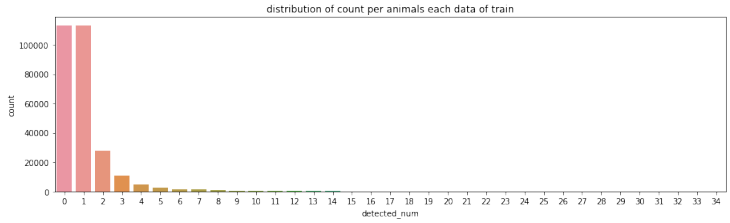
```
fig = plt.figure(figsize=(25, 16))
for i,im_path in enumerate(train_jpeg[16:32]):
    ax = fig.add_subplot(4, 4, i+1, xticks=[], yticks=[])
    im = Image.open(im_path)
    im = im.resize((480,270))
    plt.imshow(im)
```

This project is an annual competition (with slight variation - last year was focused solely on the American West). Most of last year's models used DenseNet, MobileNet, or EfficientNet (which is what we used in our model).

2 Dataset and Features

iWildCam 2020 has an ongoing competition on Kaggle to try to crowdsource a solution to this difficult problem. The WCS training set contains 217,959 images from 441 locations, and the WCS test set contains 62,894 images from 111 locations. These 552 locations are spread across the globe. There are 675 different types of animals that have been labeled. We also explored using supplemental training data from the iNaturalist 2017, iNaturalist 2018 and iNaturalist 2019 competition datasets and a Landsat-8 multispectral imagery dataset for each camera location.

The images have been run through a pre-trained model to detect animals, people, and vehicles in camera trap images so that the classification process can be broken into two steps: 1) find the object(s) of the picture 2) identify it(/them). The vast majority of pictures have 0 or 1 animals in them, but a fair number have up to 4. At first we believed it to be likely that herd animals would be photographed together and that this could be learned by the model, but we ultimately found that we obtained the best results when training on pictures with 0 and 1 animals. Further improvements could potentially be made by eliminating the reflections of animals in water.



We are also able to analyze the images by timestamp. If certain animals have hibernation schedules or migration patterns or are nocturnal/diurnal, that pattern may be learned by the neural net. This could be helpful since we cannot rely too much on the camera locations themselves since the test set contains photos only from locations that are not represented in the training set.

3 Literature Review

Since this course is our first exposure to this topic, we spent a fair amount of time learning more about convolutional neural networks and how the hyperparameters interact with each other. Additionally, we researched how others have dealt with data distribution mismatch problems without having any access to the test set to redistribute data and delved deeper into transfer learning and ImageNet.

In the context of this specific problem, we read all the **information provided** by the competition organizers on Kaggle and all the public discussions on the page regarding preliminary best approaches to take. **This research paper** helped us to better understand the data and how to prepare a model for camera trap imaging. It uses a "randomly initialized Inception-v3 with input size 299 299, [and] was trained using only camera trap images." It utilizes the rmsprop optimizer and assigned a learning rate of 0.0045. **This research paper** from Cornell entitled "Generalizing from a Few Examples: A Survey on Few-Shot Learning" and **this GitHub repository**, named "One-Shot-Learning-with-Siamese-Networks" helped us better understand ways to approach mitigating our data mismatch problem. **This research paper**, published in 2018 by the winners of the 2017 iNaturalist competiton and entitled "Large Scale Fine-Grained Categorization and Domain-Specific Transfer Learning", and **This research paper**, published in 2016 by Cornell and entitled "What makes ImageNet good for transfer learning?", highlighted the importance of transfer learning in this project and were the basis of our inclusion of ImageNet. Finally, **This article** entitled "A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks" broadened our understanding of the role of early stopping.

Additionally, linked in the relevant sources are an additional research paper and article that have helped us to think about the best way to approach this challenge. In future sections, our research into the best architecture to use is discussed in the context of our development iterations.

4 Methods and Results

All of our models were trained using EfficientNet from Keras, which is a CNN, and we have been using a Jupyter Notebook run through AWS to train our model.

- Our Milestone 1 model only had one hidden layer, and unfortunately our accuracy was not very good for this milestone–it was able to identify the correct species 22% of the time. We were relatively happy with the training and validation accuracy, which was closer to 70% accuracy, but that likely meant that we were over-fitting to the training data.

- Our Milestone 2 model achieved a 38.2% test accuracy, which was a large improvement. At the same time, our training accuracy held steady around 70%.

- Our Final model was our best model yet, as one would hope, and reached a 48.5% test accuracy and a 72.5% training accuracy. We experimented with a number of methods to achieve this result:

Since this project requires us to identify species from a camera that is not present in the training data, we implemented transfer learning from ImageNet.

Additionally, we chose to use average pooling over max pooling because it encourages the network to identify features in the image more completely. In the words of Zhou et al. "This is because, when doing the average of a map, the value can be maximized by finding all discriminative parts of an object as all low activations reduce the output of the particular map." As such, average pooling should outperform max pooling in large scale and complication classification problems due its superior ability to localize diverse object features.

We ran our model with 6 to 50 epochs. To compare performance across the different numbers of epochs, we ensured that the number of epochs was the only variable being changed. Performance on both the training, validation, and test sets increased as the number of epochs increased until the 26th epoch and then declined again, so we chose 26 epochs for the parameter. Additionally, we implemented early stopping so that we can make sure that the model will stop if the accuracy starts decreasing so as not to over-fit to the training set. We also implemented checkpoints so that if the model stopped running we would not lose all the progress it made.

Additionally, we tried using both EfficientNetB3 and EfficientNetB7. It seemed as though using the latter was too labor-intensive and the program would crash, so we had to significantly reduce the batch size in order for it to run. We also increased the ratio of training-validation samples upon which the model was trained on, because as discussed in class you can often use a 98/1/1 split.

5 Reflection on Results

While we have seen great improvement in the performance of our model (we started with approximately 20% accuracy and now have about 48.5% accuracy), we didn't have the results that we had hoped for at the beginning of the quarter. The top-scoring team on Kaggle ended up with just over 90% accuracy, so we had hoped to get much closer to that than we did in the end. As was discussed in our last meeting with Advay, we had hoped to implement one or few shot learning, as that would mean that we could train our model on only a few images for each species. We were also hopeful that implementing this would improve our results and get us to over 50% accuracy. Unfortunately, we were unable to get a functioning version of one shot learning, so we had to take it out of the final version of our model. This was in large part due to current events, which forced us to scale back some of the expectations for the final version of the model. Even though we are pleased to have seen quite a significant improvement in the accuracy of our model, we do wish that had been able to submit something that both implemented one shot learning and had a >50% accuracy.

6 Contributions

We were all very new to deep learning, so generally we did most things together as opposed to delegating different tasks to different group members. Some of the things we struggled most with were both setting up and using AWS (as well as remembering to stop the instance when not in use!), finding and applying a model to fit our use case, as well as then making effective and meaningful changes to that model.

We believe it was very useful to have a team of people who came from technical but diverse backgrounds (1 MSE undergrad, 1 MSE undergrad and CS coterm, and one Symbolic Systems undergrad). Bringing different backgrounds to the table helped in that where one person had a gap in knowledge the others could help to fill it. That being said, because we were all beginners at deep learning, it meant both that one person didn't step in and take over the project and also that we as a group had similar expectations for what we would be able to accomplish with the project.

7 Additional Sources

Tabak, MA, Norouzzadeh, MS, Wolfson, DW, et al. Machine learning to classify animal species in camera trap images: Applications in ecology. Methods Ecol Evol. 2019; 10: 585– 590. https://doi.org/10.1111/2041-210X.13120. **Link**.

How to Classify Photos of Dogs and Cats (with 97 percent accuracy) by Jason Brownlee on May 17, 2019. **Link**.

Implementation of EfficientNet model. Graphs of accuracy. **Github Link**