iWildCam 2020 Trail Camera Animal Classification

Guanyu Zhang[1], Brent Bouslog[1], Joseph Dieciedue[1], and Roger Terrazas[1]

[1]University of Texas at Austin

EE 461P DATA SCIENCE PRINCIPLES

Joydeep Ghosh

2020-05-10

Author Note

Project website: EE 461P 2020 Spring Project

Abstract

This term project for EE 461 Data Science Principles presents our approach to the iWildCam 2020–FGVC7 Kaggle Competition. Camera traps (wild cams) enable the automatic collection of large volumes of image data that biologists use to monitor biodiversity and estimate animal population densities. Our primary objective is to build an image classification model from the provided dataset and then validate its performance on the test dataset.

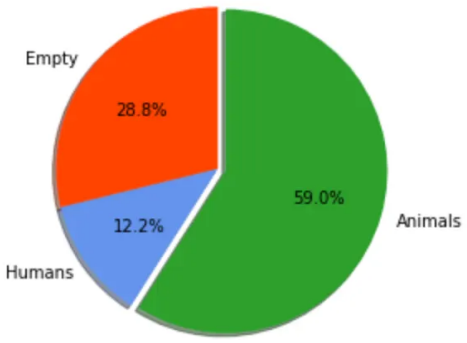
iWildCam 2020 Trail Camera Animal Classification

Introduction and Motivation

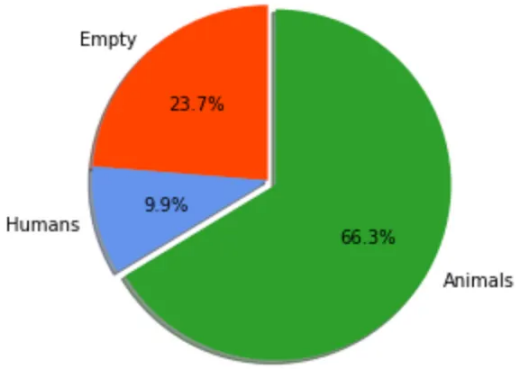
Across the globe, Camera Traps (or Wild Cams) produce massive collections of images that biologists and nature enthusiasts use to monitor wildlife diversity and population densities. Recently there has been a push by the general scientific community to use image classification techniques to streamline the collection of wildlife data. However, one of the biggest issues with wildlife image classification is that while one model may work well in a specific camera location, it will most likely not perform well in another. Kaggle has subsequently hosted the iWildCam competition since 2018 to entice data scientists to produce a solution to the wildlife classification problem. The following article will describe the approach a group of engineering students from The University of Texas at Austin took in order to achieve an accuracy score of 65.7% on the iWildCam 2020 competition test data set.

Exploratory Data Analysis

The most important step when starting a data science project is taking a look at the data. The dataset that we trained our models on consists of 280,852 images taken by 552 Camera Traps in locations spread all over the globe. The majority of these images, roughly 78%, are in the training set. Furthermore, the training set has cameras in 441 different locations and the test set has cameras in 111 different locations, meaning that there are no shared camera traps between the training and test sets. This location discrepancy is reflected by the distribution of the classes in each set. Since the cameras in each of the sets are in different locations, each set does not capture all of the classes. Therefore our model has to be able to classify animals that it never sees within the training set. The figures below show the class distribution differences between the training and test sets.



(a) *The distribution of categories in the training data*



(b) *The distribution of categories in the test data*

Figure 1

Comparison of class distributions between the training and test sets.

Here we can see that the percentage of images classified as empty and human differ in the training and test set. The animals part of the pie graph symbolizes the remaining 265 classes of species. This is also our first glimpse at how imbalanced the dataset we are working with truly is. Images classified as empty, meaning some moving foliage or falling tree branch triggered the camera trap, account for over 27% of all images in the dataset. Humans also trigger camera traps a lot considering they account for 11% of all images in the dataset. However, the class imbalance does not stop at these two most popular classes. We followed a notebook by Aleksandra Deis to get a better look at the data (Deis, 2020). Below are some graphs which illustrate the severity of the imbalance. Classes with the highest occurrence have thousands of images in the dataset while several classes have less than 10 images. Some classes even have only one image in the entire dataset. Our model will have to overcome these class imbalances as this type of a dataset distribution is expected when working with trail cameras. Many images that camera traps take aren't useful to a scientist trying to monitor biodiversity and population density. A good model should be able to identify and sift out the useless images while correctly identifying animals, even ones that it has encountered a small number of times.

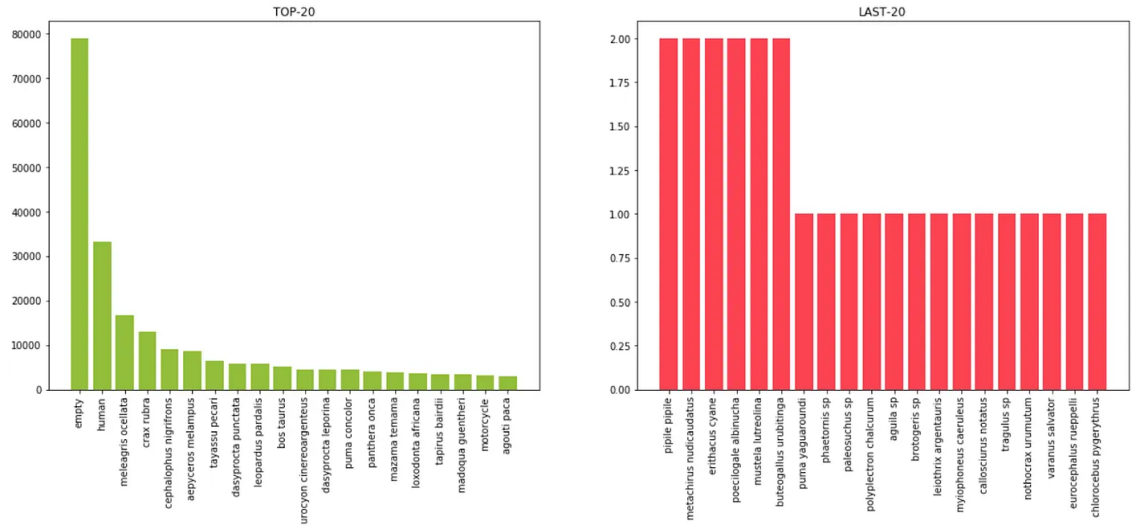
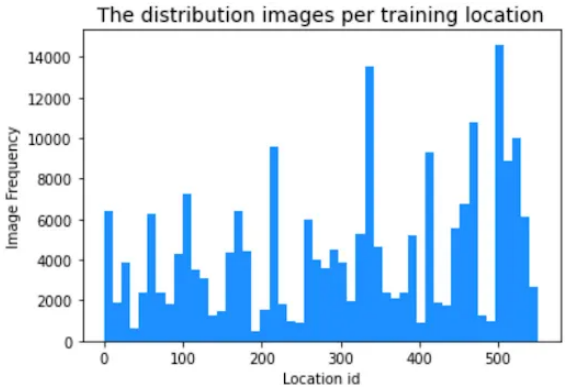


Figure 2

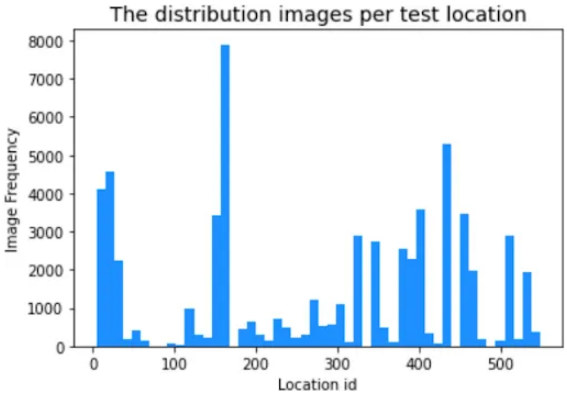
Class distribution within the iWildCam image dataset.

There is also a discrepancy in the number of images that come from each of the camera trap locations. Some of the cameras that contribute to the dataset are responsible for thousands of images while others contribute very little. There are a few reasons that could explain this imbalance such as foliage density, animal density, and camera age. The graphs below show this disparity. This location imbalance, in addition to the fact that training and test locations are not shared, create some problems for classification accuracy for our models. However, this is a challenge that we must overcome as the goal of this competition is to create a model that will work as new locations and new cameras are added.

Lastly, let's take a look at the number of animals found in each image. The vast majority of the images contain fewer than two animals. Nearly every image has less than five animals, but some have up to 80, such as the image below. In images like this, the animals in the image are of the same class as it is usually a herd of some species. In other images with multiple animals, most of the time they all belong to a single species. Nevertheless, there will be some cases when a camera trap is triggered and more than one class is present in the image. However, in the dataset that we were given images are only associated with a single class, so we would not be able to correctly classify both classes in the image.



(a) *The distribution images per training location*



(b) *The distribution images per test location*

Figure 3

Location distribution within the iWildCam image dataset.

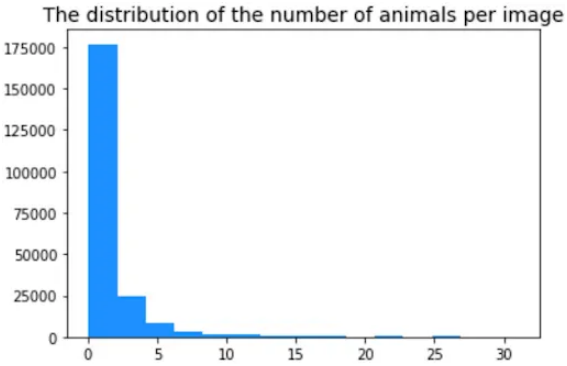


Figure 4

Location distribution within the iWildCam image dataset.

Variation in the Data

Trail camera images come with many issues that are usually not prevalent in other image classification tasks. There are many technical issues that arise with trail cameras and there are many environmental factors that affect the quality of images that are taken (Aodha et al., 2020). Below are some examples of data variation in wild cams that we had to overcome within our data preprocessing and model development.

Figure 5

Examples of poor lighting, motion blur, and occlusion respectively in trail camera images.

Figure 6

Example of temporal changes in trail camera images (Same location, different times of year).

Data Preprocessing

Contrast Limited Adaptive Histogram Equalization (CLAHE)

A large issue with our dataset, and any trail camera dataset, is poor quality images with poor contrast due to the fact that many images are taken at night. To address this issue we decided to apply Contrast Limited Adaptive Histogram Equalization to each image as suggested by a Kaggle notebook authored by Chanran Kim (Kim, 2020). Below is a picture showing how the CLAHE method works on one of the pictures from our dataset. The left image is the original image and the green box shows the animal (sun bear) location in this image. In this original image the animal is hard to distinguish from the background because of underexposure. One method to address this issue is a method called Histogram Equalization and the result is shown in the middle image. After applying this algorithm to

the original image with a histogram that has an uneven distribution, the new picture's histogram will be stretched evenly such that each pixel value has some corresponding points. Even after Histogram Equalization there is still some misinformation. In this example, Histogram Equalization caused overexposure to happen. The CLAHE algorithm, which is a modified version of Histogram Equalization, splits the original image into small parts and in each part applies separate histogram equalizations. Bilinear interpolation is applied to smooth the borders between the split parts of the image. The right image shows the final result of applying the CLAHE algorithm. The local contrast of the image is enhanced and we can see more information about the outline of the animal (Open Source Computer Vision, 2015).

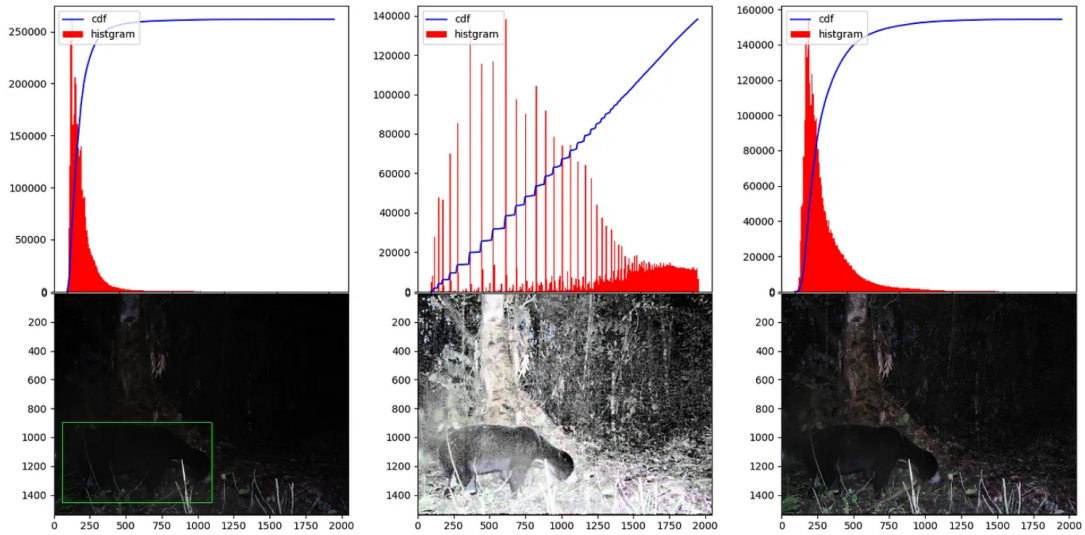


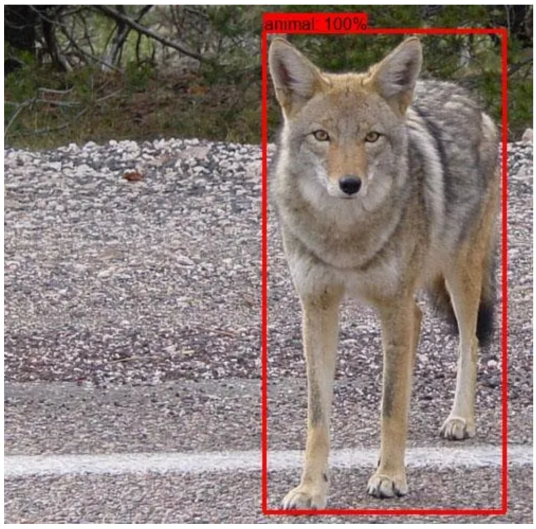
Figure 7

Example of Histogram Equalization vs. CLAHE on an image from the iWildCam image dataset.

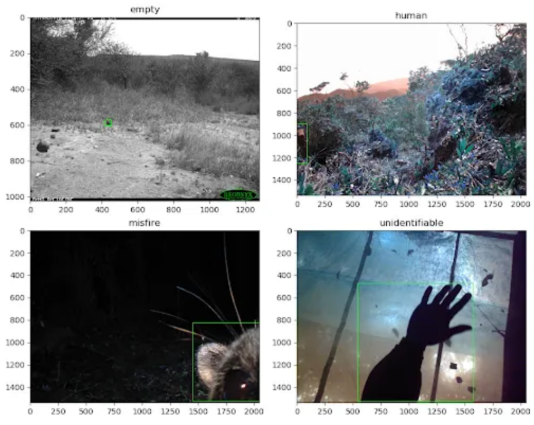
MegaDetector Image Cropping

The goal of this competition is to classify animal pictures taken from new locations rather than just those from the training set. Therefore we would like to focus on just the animals and not the background. Considering that some pictures contain too much information

about the background, we want to extract the animal information from these images. The competition provided every competitor with a baseline animal detection model developed by Microsoft which they called MegaDetector. Not only does this model classify each image as either an image containing nothing, an image containing an animal, or an image containing a human, but it also creates a bounding box with a confidence value around each detected animal or human (Morris et al., 2019). Instead of having to run the model on all the images, we were given a file containing classification, confidence, and bounding box information for each image as was given by the MegaDetector model. We used the bounding box information to crop the original images and we only cropped the image if the confidence of the box was greater than 50%. After this step, the new images should theoretically contain only animal information and should be easier to classify.



(a) *Example output from the MegaDetector model on an image of a coyote.*



(b) *Examples of output from the MegaDetector model on various non-animal classes from the iWildCam image dataset.*

Figure 8

MegaDetector output examples.

Data Augmentation

Another issue with the dataset was that many classes had fewer than five images and some even had as few as only one. Therefore, to avoid overfitting for these smaller classes we decided to apply various data augmentations to each image before each epoch of training. For example, we applied random horizontal flips, shifts, and rotations to give our model the illusion that we had more variation in our data than we actually did. This allowed us to gain more diversity in our training dataset without having to go out and collect more data. Moreover, this helped us to prevent overfitting to sequences of images that had the same animal in the exact same position and environment. However, the downside to data augmentation is that it is computationally expensive and since we had to call it during the creation of every batch during training, it increased our overall training time. After all of these data preprocessing methods were applied, we downsized each image to 200 pixels by 200 pixels to decrease the overall runtime and make the size of the dataset more manageable.

Model Description

For our model we decided that we would implement transfer learning on a pre-trained convolutional neural network. The iWildCam dataset is huge and in order to train a deep convolutional neural network that would perform even remotely well, we would need more gpu resources than we had available at our disposal. Therefore, to save time and resources we decided to train a smaller neural network on the output embeddings of a pre-trained convolutional neural network which had been trained on a vast image dataset such as ImageNet. ImageNet is a vast image database containing millions of images with hundreds of different classes, many of which being various species of animals, so we decided that models trained on that dataset could be reasonably applied to our problem.

As for the architecture for the pre-trained convolutional neural network, we settled on the ResNet50 architecture (He et al., 2015). Winner of the 2015 ImageNet challenge, the ResNet50 architecture was one of the first architectures to efficiently solve the problem of

vanishing gradients in deep networks. This allowed for much more successful training of extremely deep networks than had ever been accomplished before. They accomplished this with the addition of “skip-connections” between layers (Dwivedi, 2019).

Figure 9

Skip Connection image from DeepLearning.AI.

The idea behind skip connections is that they allow for an alternate path for data as well as gradients to flow through the model. Therefore, instead of flowing through the convolutional layers, the data and gradients could bypass them entirely through the skip connections. The problem with deep networks normally is that with each subsequent layer the gradient with respect to the error function becomes vanishingly small, effectively preventing the weights from updating and the model from improving. Skip connections mitigate this problem by allowing an alternate path for the gradients to flow so they vanish less quickly in extremely deep networks. Moreover, theoretically they allow for the creation of an identity function of the lower layer. This would mean that the higher layer should perform at least as well as the lower layer because if it didn’t the model would just be better off bypassing the layer entirely with the skip connection [7].

To implement this transfer learning, we followed an article for implementing transfer learning with the RestNet50 architecture in Pytorch (Fotache, 2018). We added two fully connected layers with ReLu activation as well as one output layer with softmax activation to serve as our class predictions. Note that the output layer has only 216 nodes as the

training set only had 216 classes. The other 49 classes exist solely in the test dataset and we figured there would be no way to classify them if they were not in the training set without collecting more data. We left the ResNet50 weights pretrained on the ImageNet dataset and only trained our new three layers on the iWildCam dataset. Due to the sheer size of our dataset we only needed to train for 3 epochs before we saw our validation loss stop decreasing.

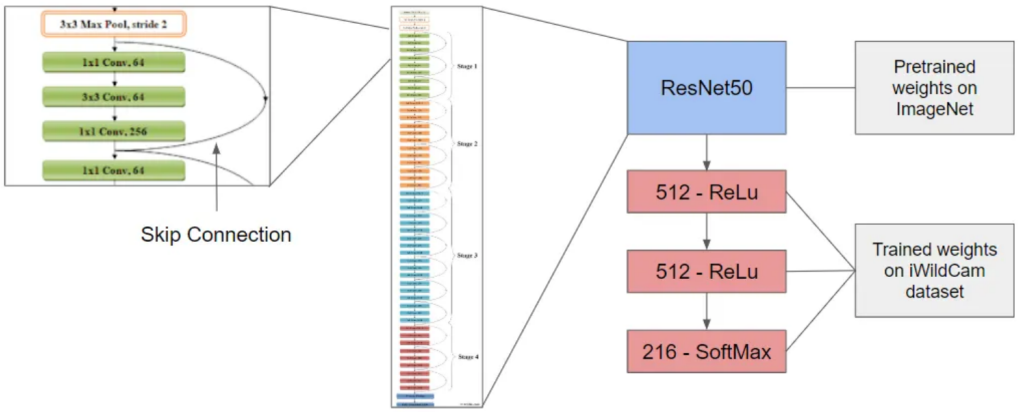


Figure 10

Transfer learning convolutional neural network model diagram.

With this baseline model, we got about 60% accuracy on the test Kaggle dataset. This was reasonably good, but we knew we could increase our model's performance with some metadata about where and when the images were taken. Along with the images we had data about what time of day and what month images were taken. We thought that this would be important due to the activity patterns that animals tend to follow which coincide with the time of day or the seasons. Moreover, we were given Landsat-8 images for all the locations for which the trail cameras were positioned. Landsat-8 is a satellite that takes images from space with many bands of spectral information such as visible light, infrared waves, and thermal information. We took those images for each location and averaged all of the pixel values of the 9 bands of information which we deemed most important (Cole, 2020). We figured this could give us insight into the type of habitats for each location and

therefore narrow down which animals could possibly be in the image. Lastly, we had information about how many images were taken for a given sequence of images. We figured that this might be important as larger animals tend to walk more slowly than smaller animals and therefore stay in frame longer, producing more images for each time they were spotted on the trail camera. We trained a random forest classifier on this new dataset with the outputs of the convolutional neural network and the metadata as features and were able to achieve an accuracy of 65.7% on the test Kaggle dataset.

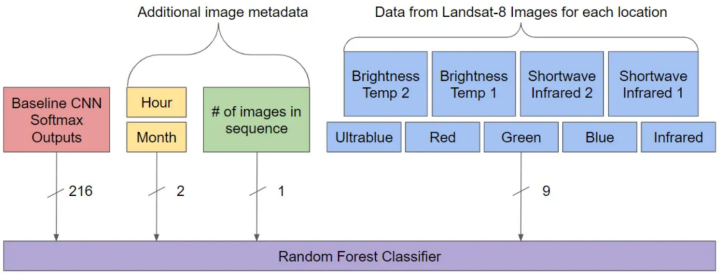


Figure 11

Stacked random forest classifier using baseline CNN outputs and metadata as features.

Results and Analysis

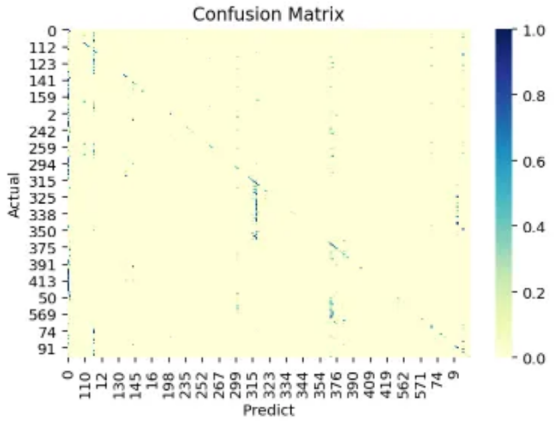


Figure 12

Confusion matrix for our final classification model.

Overall our model received 65.7% accuracy on the test dataset, a 4.4% increase from the baseline benchmark set by the competition. This means that our model performs reasonably well on locations that were not seen within the training dataset. Therefore we can say that we accomplished the goal of the competition which was to create a model that generalizes well to new locations. Above you can see the confusion matrix of our model on the training dataset. You can see a general diagonal line signifying that most classes were predicted correctly at a decent rate. However, you can also see a few vertical lines, representing classes for which many other different classes were mispredicted as. A few of these classes were class 3, 96, 115, 317, 372, and 374 representing the central american agouti, impala, gunther's dik-dik, black-fronted duiker, ocellated turkey, and great curassow respectively.

Figure 13

Images of a central american agouti (top left), impala (top middle), gunther's dik-dik (top right), black-fronted duiker (bottom left), ocellated turkey (bottom middle), and great curassow (bottom right) respectively.

If one were to naively look at these images, one might think that many images were mispredicted as these animals simply due to the non-distinct features that fit the profile of many different animals. This might be the case, but it must also be noted that these classes have the 8th, 6th, 18th, 5th, 3rd, and 4th most images respectively out of all the classes in the training dataset. Therefore our model is certainly suffering from some issues that arise from significantly unbalanced classes as we described earlier.

On the contrary, our model was very good at distinguishing a few different classes without mispredicting other classes as that class. Specifically, one of these classes was class 404 or the misfire class. Now this is extremely important because along with class 0, or the nothing class which our model also predicts very accurately, these images represent images that biologists who might want to use this model would want to filter out with a high accuracy. The only important images to them are images that contain information about biodiversity in the form of different species so being able to filter out these images is extremely important.

Figure 14

Example of a misfire image from the iWildCam image dataset.

Not only was our model good at predicting images that did not contain any animals, but there were a few species that were predicted with very high accuracy without causing many other species to be mispredicted as them either. A few of these classes were class 2, 71, 91, 110, 112, and 139 representing the white-lipped peccary, cow, giant pouched rat, water buffalo, giraffe, and great argus respectively.

Looking at these images it is not hard to recognize that each of these species has some very recognizable features. Both the peccary and cow images have multiple individuals, the pouched rat has a very long tail, the water buffalo has very distinguishable horns, the giraffe has both spots and a very large neck, and the great argus has very large and colorful feathers. Therefore our convolutional neural network probably picked up on some of those features which allowed our model to predict these images with such a high accuracy. However, it must also be noted that some of our metadata features were very important

Figure 15

Images of a white-lipped peccary (top left), cow (top middle), giant pouched rat (top right), water buffalo (bottom left), giraffe (bottom middle), and great argus (bottom right) respectively.

when it came to classifying images with our final random forest model.

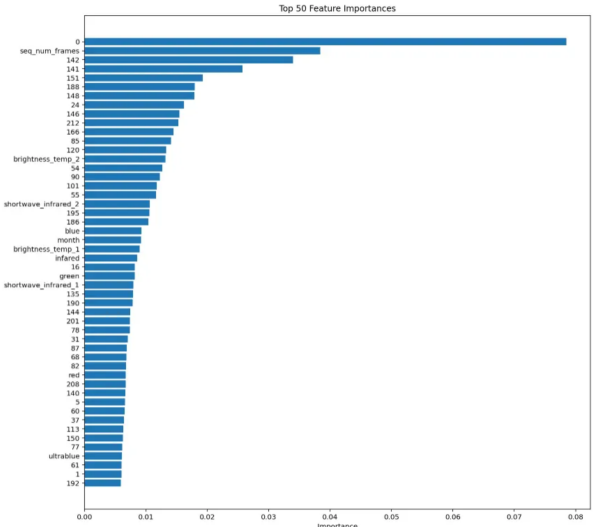


Figure 16

Top 50 feature importances for final stacked random forest classifier using CNN outputs and metadata as features.

Surprisingly, the number of frames in a sequence was the second highest feature in terms of importance. This could explain why we were able to predict classes such as the water buffalo, cow, and giraffe class with such high accuracy because they are more likely to stay

in frame longer due to their tendency to walk slower. Similarly, the time of year was important for determining species, likely due to the tendency of animals to migrate to different locations throughout the year. Moreover, the location metadata from the Landsat-8 images was also very important. Theoretically it could have allowed us to distinguish normally indistinguishable animals in terms of looks. For example, the mountain lion and the lion are relatively indistinguishable in terms of looks. However, the mountain lion tends to live in more mountainous and forested areas while the lion tends to live in grassier savanas. Therefore, just from information about the location one could determine which species the image came from. We can see this from the results of our model because lions were mispredicted as mountain lions and vice versa 0% of the time.

Figure 17

Mountain Lion (Left) and Lion (Right) images from the iWildCam image dataset.

Conclusion

In our project, we implemented a classification model using a transfer learning method stacked with a model using additional metadata as features. However, there are still some problems we would have liked to address and features we would have liked to implement had we had more time. Even after our data preprocessing, some images from the training set still had low quality. While this is expected from trail cameras which generally take lower quality images than most cameras, we would have liked to research more methods for improving image quality such as removing blur from images. In addition, we would have loved to include images from the previous iWildCam competitions to help us deal with

classes with very few images. Our limiting factor throughout the entire project was the sheer size of the data. As the training set for the 2020 competition was already 108 GB, which we compressed down to about 8 GB after resizing the images, we decided to refrain from trying to deal with even more data even though we knew it would improve our model. Had we had more time we would have put more effort into hosting our data and models on a cloud platform such as Amazon Web Services. With that in place we would be able to host our data in a storage bucket rather than our local machines and would have been able to include data from previous competitions, which in turn would most likely have improved our models. One feature that we wish we could have implemented was some sort of anomaly detection or a way to classify images as species that had not yet been seen by the model. This would be important for determining whether a new species had been discovered and therefore would be something that many biologists would be interested in. In conclusion, although there are things we can still improve, in terms of creating a trail camera animal classifier that performs well on new locations, we can safely say that we accomplished this goal. Our hope is that models like ours can be used by biologists around the world to help monitor biodiversity as we enter an age where wildlife is becoming increasingly threatened by society.

References

Aodha, O., Beery, S., & Cole, E. (2020, March). Iwildcam competition 2020 [Retrieved April 9, 2020, from GitHub]. https://github.com/visipedia/iwildcam_comp/blob/master/readme.md

Cole, E. (2020, March). How to parse the remote sensing data [Retrieved May 4, 2020, from Kaggle]. https://www.kaggle.com/c/iwildcam-2020-fgvc7/discussion/135642

Deis, A. (2020, March). Iwildcam eda [Retrieved April 17, 2020, from Kaggle]. https://www.kaggle.com/aleksandradeis/iwildcam-eda

Dwivedi, P. (2019, January). Understanding and coding a resnet in keras [Retrieved April 25, 2020, from Towards Data Science]. https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

Fotache, C. (2018, November). How to train an image classifier in pytorch and use it to perform basic inference on single images [Retrieved April 20, 2020, from Towards Data Science]. https://towardsdatascience.com/how-to-train-an-image-classifier-in-pytorch-and-use-it-to-perform-basic-inference-on-single-images-99465a1e9bf5

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*. https://arxiv.org/abs/1512.03385

Kim, C. (2020, March). Image pre-processing for iwild 2020 [Retrieved April 15, 2020, from Kaggle]. https://www.kaggle.com/seriousran/image-pre-processing-for-iwild-2020

Morris, D., Herenu, D., & Yang, S. (2019, March). Cameratraps overview [Retrieved April 16, 2020, from GitHub]. https://github.com/microsoft/CameraTraps/commits/master/README.md

Open Source Computer Vision. (2015, December). Histograms — 2: Histogram equalization [Retrieved April 15, 2020, from OpenCV]. https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html

Zhang, G. (2020). *Project code repository for iwildcam classification* [Retrieved May 10, 2020, from GitLab]. https://gitlab.com/guanyuzhang/ee461p_2020spring

Code Availability

Our implementation is publicly available (Zhang, 2020).