Using Deep Learning to Classify Animals in the Wild

Sai Nadendla[1], Joseph Granados[2], Dr. Gurman Gill[2]
[1]Casa Grande High School, Petaluma, [2]Department of Computer Science, Sonoma State University, Rohnert Park

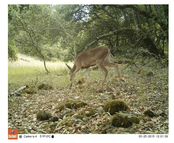
Introduction

Machine learning is branch of computer science in which the machine is able to learn to perform certain tasks without being explicitly programmed. This technology gives computers the ability to classify various images after being trained on a large dataset of images. Here we investigate the viability of applying deep learning, a machine learning technique, to a relatively small dataset of wild life images.

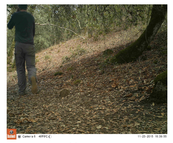
Dataset

The dataset we used for the experiments is composed of roughly 1600 images taken by trail cameras on SSU Preserves. The trail cameras take a series of three images every time they detect motion. The dataset contains eight classes of images which are pictured below. All the images below were correctly classified and underneath the class name is the confidence the model had in its prediction.

Bobcat
Confidence Score = 0.99799



Deer
Confidence Score = 0.99972



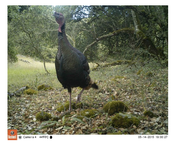
Human
Confidence Score = 0.99932

Nothing
Confidence Score = 0.89992

Possum
Confidence Score = 0.97996

Skunk
Confidence Score = 0.98984

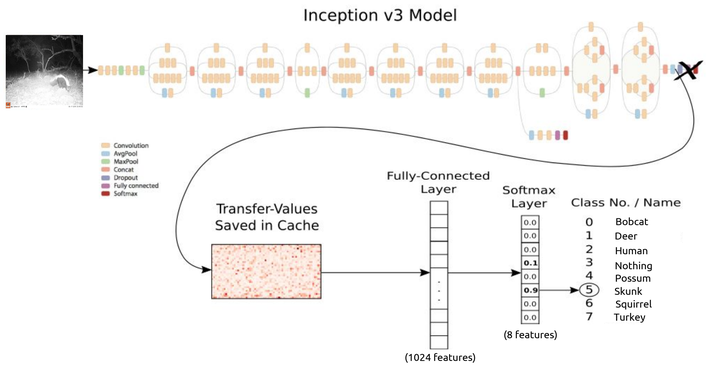
Squirrel
Confidence Score = 0.86793



Turkey
Confidence Score = 0.99993

Deep Learning

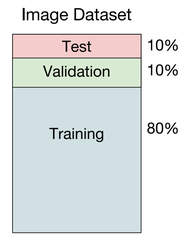
The machine learning technique we applied here is known as a convolutional neural network, a type of neural network architecture inspired by biological image processing systems. These networks consist of very long chains of operators with many parameters which are tuned as the network undergoes a process known as "learning". During this process, the network is able to learn to distinguish between different classes of images given millions of training images. Typically training an entire convolutional neural network from scratch can take weeks, and it requires immense computational power. To get around this we used a network that was already trained to recognize certain images and modified the final layers of the network to recognize images from our dataset; this is known as transfer learning. The network we retrained is known as the Inception v3 model which is pictured below. In order to complete the retraining process, we used an example script provided by TensorFlow, an open source machine learning software library developed by Google, and adapted it for our use case.



Experiments

Baseline

To begin our experimentation we ran the retraining script with our image dataset to calculate a baseline classification accuracy. The script provided by TensorFlow splits our image dataset into three subsets (test, validation, training); the percentage of images within each subset is shown to the right. The training subset is used to retrain the network for our image data, the validation subset is used to check the progress of training and ensure that network is learning properly, and the test subset is used after the training is complete to assess the model's performance on images it has never seen. The images are placed randomly into each subset. During the training process, the model takes random batches of 20 images at a time from the training subset and tunes its parameters depending on the images it was unable to classify in the batch; it repeats this process 4000 times.



Grouping Based on Timestamp

The baseline performance of the model is not necessarily a reliable measure of the model's true accuracy due to manner in which our image dataset was created. For our dataset, every time a trail camera detected motion it would snap a series of three images over a capture period of three seconds. In many cases, these three images would be very similar in nature especially if the animal did not move too much in the capture period (shown below). This would lead to a greater performance from our model due to fact that it is being potentially tested on images very similar to those it trained on. To counteract this issue we changed the retraining script to group images that were taken within two seconds of each other and place the entire group into a subset; this way images taken in the same capture event would be within the same subset.

Image 1

Image 2

Image 3

K-fold Validation

The performance of the model even after grouping images based on timestamp may still not be completely reliable; this is due to fact that images within the test set may coincidentally be images that the model is very good at classifying. To counteract this problem we divide the dataset into five subsets(folds) and run the model five times. Each time the model runs a new subset is assigned as the test data while the rest is assigned as training data. Once the five runs are completed, an average fold accuracy will be calculated. This eliminates any bias that may occur due to having certain images in the test set and provides an accurate measure of our model's performance.

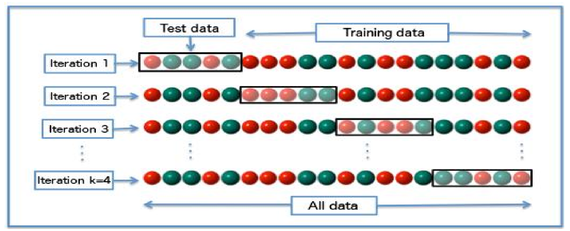
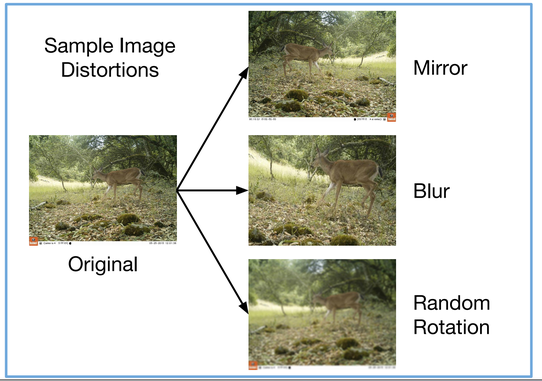


Image Augmentation

In order to improve the model's performance, we decided to employ image augmentation, a technique that applies various distortions to an image. For this experiment we decided to mirror, randomly rotate(-10° to 10°), and blur every image in the dataset. A sample image and its distortions are shown to the right. The reason image augmentation is effective in some cases is due to it greatly enlarging the image dataset and training the model with images that represent real world variations. For example, a blurry image could represent a similar image taken in the fog. This should theoretically help the model classify images taken in a variety of conditions. The model is still using k-fold validation and the distorted images are grouped with their original image counterparts.



Results

The following results are the model's accuracy on the test subset.

Baseline

Model's Accuracy: 93.9%

Grouping Based on Timestamp

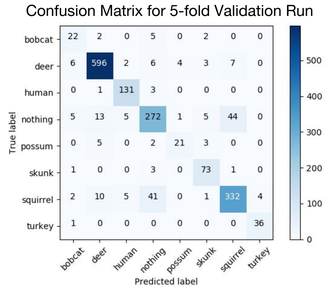
Model's Accuracy: 91.3%

5-fold Validation

Model's Average Fold Accuracy: 88.41%

Image Augmentation

Mirror: Model's Average Fold Accuracy: 88.36%
Blur: Model's Average Fold Accuracy: 88.38%
Rotate: Model's Average Fold Accuracy: 88.72%



Result Analysis

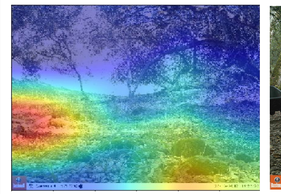
Below are three correctly classified images that illustrate the model's capabilities in classifying difficult images. The confidence the model had in its prediction is listed underneath the class name.

Bobcat
Confidence Score = 0.61231

Deer
Confidence Score = 0.95618

Turkey
Confidence Score = 0.64954

Below are three incorrectly classified images that illustrate areas in which the model struggles. The image on the left had a localization map applied to it which highlights the important regions for predicting the class of an image. In the region highlighted as most important, there is a rock which the model likely confused for a squirrel. The middle image has a lot of motion blur and the human is almost out of frame. The right image barely has the deer in frame with only a part of the deer's ear being visible.



Actual Class: Nothing
Predicted Class: Squirrel
Confidence Score = 0.52377

Actual Class: Human
Predicted Class: Squirrel
Confidence Score = 0.61954

Actual Class: Deer
Predicted Class: Nothing
Confidence Score = 0.96952

Conclusion and Future Research

Applying current machine learning techniques to a relatively small dataset has proven to be fairly effective, but it is still very far from reaching the classification precision of a human. As technology progresses, the same experiments could be conducted in the future with an updated convolutional network. Our classification accuracy would likely only increase as models grow more and more sophisticated. The experiments could also be conducted again in the future with a larger dataset, as the trail cameras capture even more images. This would probably also lead to greater performance from our model.