Monitoring the Entire Forest

Introduction

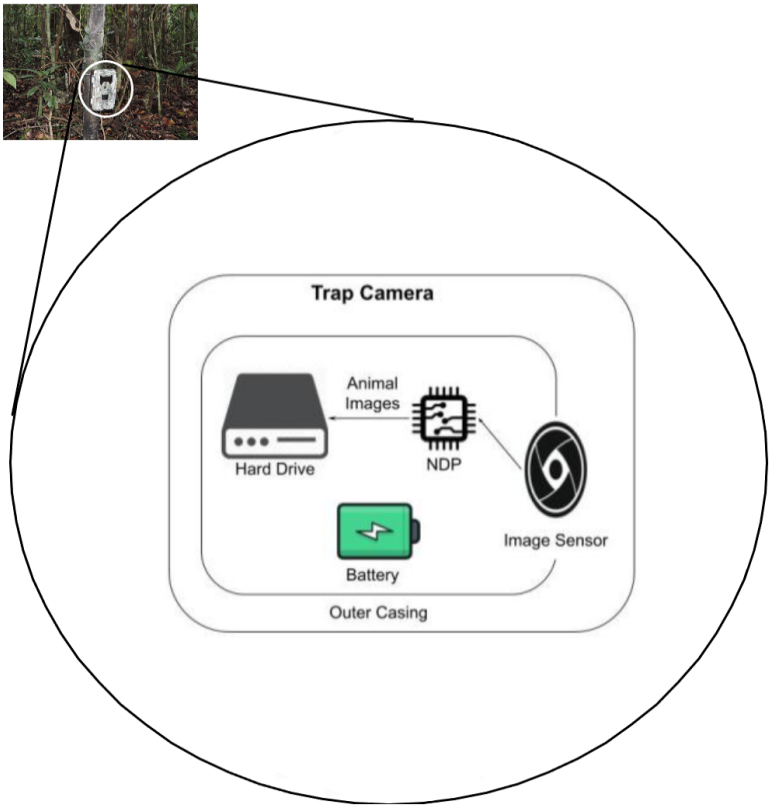
Earth observation data covers 100 percent of the earth's surface on a daily basis, but the resolutions available from near earth orbit are not sufficient to see wildlife smaller than most trees. Since animals are the leading indicators of ecosystem health, it is important to develop more complete measures of animal abundance on the ground. Despite the importance of abundance measures, they are rarely carried out continuously or at any significant scale. Collecting and cleaning data is an expensive and time consuming process carried out by a small number of people (e.g., ecologists) that are experts for certain ecosystems. Labor intensive observation processes thus do not cover adequate land area due to a lack of qualified people and funding for them to continuously monitor habitat at scale. However, with an inexpensive and ubiquitous camera connected to a neural network in the field, it becomes possible to systematize and scale visual observation data to cover entire forests at a low hardware and maintenance (i.e., labor) cost.

This blog post introduces the work of our undergraduate capstone project at Oregon State University in biodiversity monitoring for an entire forest using Syntiant's Neural Decision Processors, which are low cost and energy efficient neural accelerator chips. We trained a series of models for detecting animals as they move in front of inexpensive camera traps outfitted with a commodity camera sensor, a coin cell battery and a small storage medium. Combined, the parts cost less than $20 and can be built and deployed en masse within a forest, then collected at the end of a year of biodiversity monitoring. Without incorporating computer vision into the camera trap, the cost for buying and servicing devices becomes prohibitive. The hardware activates at every movement and records images at the expense of power, storage, and manual servicing of the trap. A "smarter" trap would conditionally activate and store images only when animals are present.

Our capstone project covers the first step of bringing species monitoring to whole-forest scales: preparing a dataset and training the neural network model that will be loaded onto a physical device. In the following sections we introduce our modeling work at more depth and present the elements of our project that are instructive to others looking to build edge neural network computer vision solutions.

While our proof-of-concept camera trap software has not been evaluated with a hardware prototype, all our data and modeling decisions are scoped to actual hardware

that can be produced at scale and deployed to real forests. The core components of our proposed trap camera include an image sensor, a battery, storage device, and the NDP which has our stored model for making decisions on animal versus non animal images.



Data

A neural network model is only as good as the data used to train it, so the first step in this project was to find data that could be used for training. In most cases the only way to get production-worthy performance from a neural network is to have data that is very closely related to the task being performed. Therefore, we needed to find a dataset of camera trap images collected from real deployments of camera traps. Thankfully, the LILA BC dataset (Labeled Information Library of Alexandria: Biology and Conservation) contains around 3.3 million North American camera trap images (NACTI). See table one below for counts of pictures per label. In total, the dataset is about 2.6 Terabytes large with each image having a resolution of 2048x1536 or 3.1 MegaPixels. Since we were creating a model that was strictly an “animal” or “not animal” picture classifier, we aggregated all animals into a single class. The pictures come from 5 locations across

North America: California, Colorado, South Carolina, Texas, and Saskatchewan, Canada. The number of pictures from each location are listed below in Table 2. One limitation of our dataset is that it includes far more pictures of animals than non-animals. This is because the NACTI team decided to remove many of the empty, non-animal photos before publishing the data set. In the absence of animals within frame, most images are the same between time steps so storing all empty images requires an unnecessary amount of data storage when training.

Label	Number of Images			
Domestic Cow	2,019,009		Red Fox	1,723
Empty	469,356		Unidentified Chipmunk	1,533
Red Deer	183,794		Virginia Opossum	1,527
Wild Boar	139,439		American Marten	1,433
Unidentified Deer	96,115		Black Tailed Jack Rabbit	1,150
Mule Deer	85,395		Unidentified Corvus	1,040
Unidentified Bird	67,070		Domestic Dog	750
Raccoon	37,875		Unidentified Pack Rat	715
California Ground Squirrel	33,866		North American River Otter	557
American Black Bear	28,770		North American	546

			Porcupine	
Eastern Gray Squirrel	27,167		Wolf	474
Vehicle	26,015		Yellow Bellied Marmot	309
Bobcat	25,434		Unidentified Accipitrid	208
Elk	22,143		European Badger	157
Coyote	20,968		Horse	133
Cougar	14,756		Unidentified Deer Mouse	128
Snowshoe Hare	13,868		Gray Jay	77
Striped Skunk	11,485		Long-Tailed Weasel	36
Gray Fox	10,225		Ermine	29
Moose	9,960		Dusky Grouse	8
Nine-Banded Armadillo	9,856		Dark Eyed Junko	5
Unidentified Rabbit	4,487		Unidentified Mouse	4
American Red Squirrel	4,029		House Wren	4
Wild Turkey	3,643		Stellar's Jay	3
Donkey	2,665		Unidentified Pocket Gopher	1
California Quail	2,275			
Total (Animal)	2,886,844			
Total (Not Animal)	495,371			
Total	3,382,215			

***Table should be a small box with scroll on the blog page since it's so long**
Figure 1: Number of images per label in the LILA BC North American Camera Trap dataset.

location	count

Archbold, FL	1,802,622
Lebec, California	780,549
(Blank)	469,356
San Juan Mntns, Colorado	329,688

Figure 2: Photo counts at each location.

Preparing the Data

Having a pre-labeled dataset saved a lot of time that would normally be spent going through the data and recording what each picture contains, but one of the initial obstacles with a dataset of this size was downloading and transferring it to the server for training took time due to limited bandwidth. Prior to running the dataset through the training script, they needed to be converted to a format that can be more efficiently processed during training. If you start with the unedited image files on every training epoch, then the training process will be very slow and you will not have time to iteratively improve. TensorFlow Records, or TFRecords for short, are a standard format supported in the training process (see this article for more details on the format). . TFRecords give the user the ability to convert their dataset into binary file format, which vastly improves read and write time, something that can hinder performance when working with a large data set such as the one used in this project. Another one of the advantages is that the dataset can easily be loaded into main memory in batches when the entire dataset is too big to fit into main memory all at once.

The dataset was converted to TFRecords using the included Python script(generate_tfrecord.py in GitHub repo). During this step the photos were also sized down to further reduce the necessary storage space on the server and main memory during training runs. Resizing the photos down to a resolution of 120x160 and removing photo resizing from the training script reduced the training time for about 1 million photos from over 2 days to under 2 hours.

Neural Architecture Selection

Convolutional neural networks are the preeminent network type for computer vision problems. Convolutional kernels leverage spatial properties of input data to produce some of the best task performances possible. Convolutional neural networks come in many different forms and the specific convolutional architecture selected for a project is

generally determined through an empirical process of trial and error. However, the most important driver of task performance is not the architecture you run, but the data that is collected and how you process it during training. Thus it is important to select an architecture that is adequate for experimenting with data, then upon developing the strongest performing data preparation techniques for a fixed architecture you can begin iterating on the model architecture. Consequently, we focused our experimental efforts on a simple and fast to train architecture that ships in Syntiant's Training Development Kit (TDK) as shown in Figure 2.

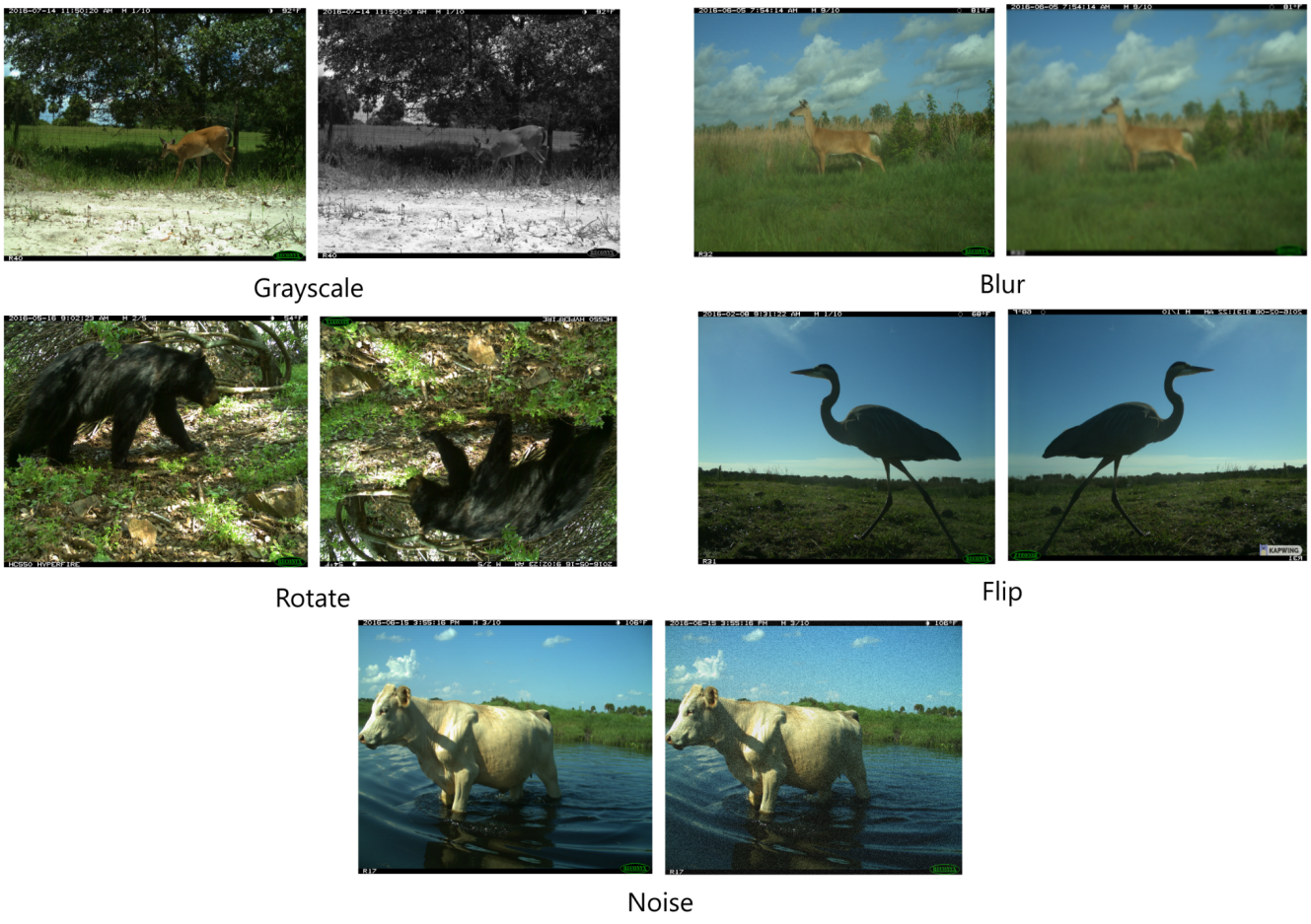
Deep Convolutional Neural Net

Layer Type	Output Shape	Param #
Input Layer	(1,120,160)	0
2-dimensional convolution	(32,60,80)	320
2-dimensional convolution	(32,30,40)	9248
2-dimensional convolution	(16,15,20)	4624
2-dimensional convolution	(16,15,20)	2320
2-dimensional convolution	(16,8,10)	2320
2-dimensional convolution	(16,4,5)	2320
Flatten	(320)	0
Dense	(256)	82176
Dropout	(256)	0
Dense	(256)	65792
Dropout	(256)	0
Dense	(2)	514
Softmax	(2)	0

Total parameters: 169,634
Trainable parameters: 169,634
Non-trainable parameters: 0

Working with the Training Script

A training script given in Syntiant's TDK, which determines if there is a person in a photograph, served as the foundation for the training script used in this project. This script originally utilized a built-in TensorFlow dataset and capability to read in our TFRecords dataset needed to be added. To train an effective model, the training set was created using pictures from Texas, Colorado and California. The validation and testing set was made using the pictures from Florida. Not mixing the locations ensured that similar locations or backgrounds were not in both the testing set ,validation set and the training set. The images were resized prior to training, but during training, augmentations were applied to the pictures randomly. These augmentations include rotating, flipping, blurring, and adding noise to the images. Examples of each can be seen in figure 3. All images are converted to grayscale prior to training so the neural network can be utilized for black and white cameras or images that can be converted to black and white.



Testing the Model

The ideal environment to test the effectiveness of our model for use in biodiversity monitoring would be on the testing bench placed in a local forest environment. Due to the time constraints of the project, this was not feasible. To compensate for this, 2 subsets of testing data was created: One large set that consists of animal and non animal images from a held-out set of the NACTI dataset and a smaller set with sequences of images showing animals moving into and out of frame (see figure 4 for a subset of that sequence).

The confusion matrix for the test set can be seen below. Overall, we have fairly good test loss and accuracy.

	Pred Neg	Pred Pos
True Neg	11,954	527
True Pos	6,137	33,597

Test loss: 0.2139023244380951
Test accuracy: 0.9039999842643738

Quantized Test loss: 0.1766078621149063
Quantized Test accuracy: 0.9160000085830688

When using our time-lapse sequence to test, we evaluated the performance of our model using the confusion matrix, accuracy, FAR (False Activation Rate), and FRR (False Reject Rate). FAR is defined as the number of times in a day that the neural network activates for an image that does not contain an animal. FRR is defined as the number of animal sightings that are not detected by the neural network. Overall, our results show good identification of animals that are clear in frame and don't blend in as much with their surroundings. Our one instance of a false reject was a deer moving across the frame that was camouflaged by the surrounding grass. Our model also falsely identifies images with big differences in frame (for example, a defined green tree against a light blue background) as having animals in them.
False activation rate over 24 hours: 17 times = .71 times an hour
False reject rate over 24 hours: 1 times = .04 times an hour

Confusion matrix for the time sequence test set.

	Pred Neg	Pred Pos
True Neg	61	17
True Pos	7	114

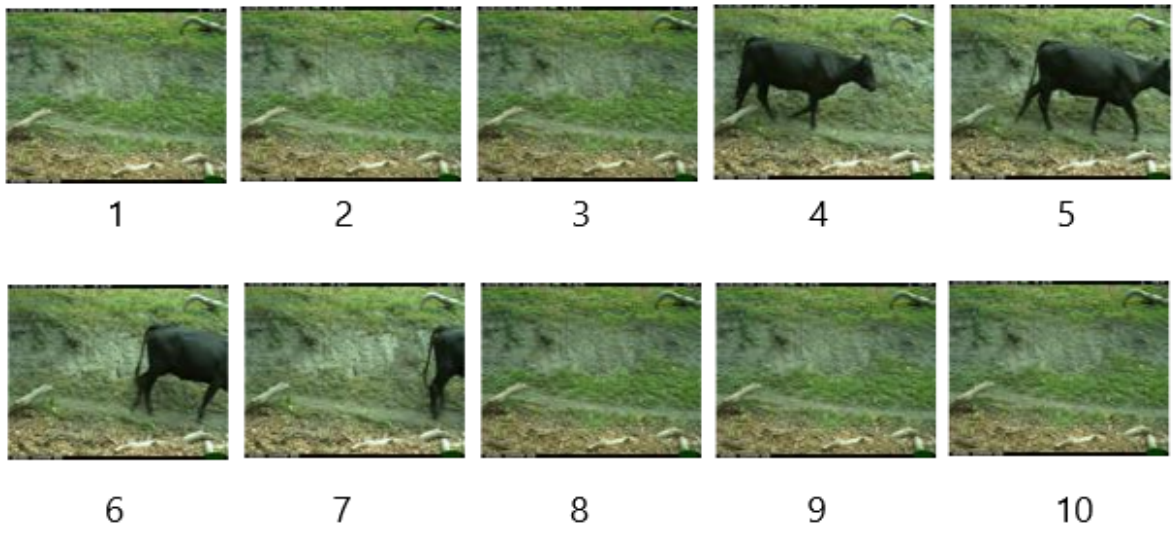
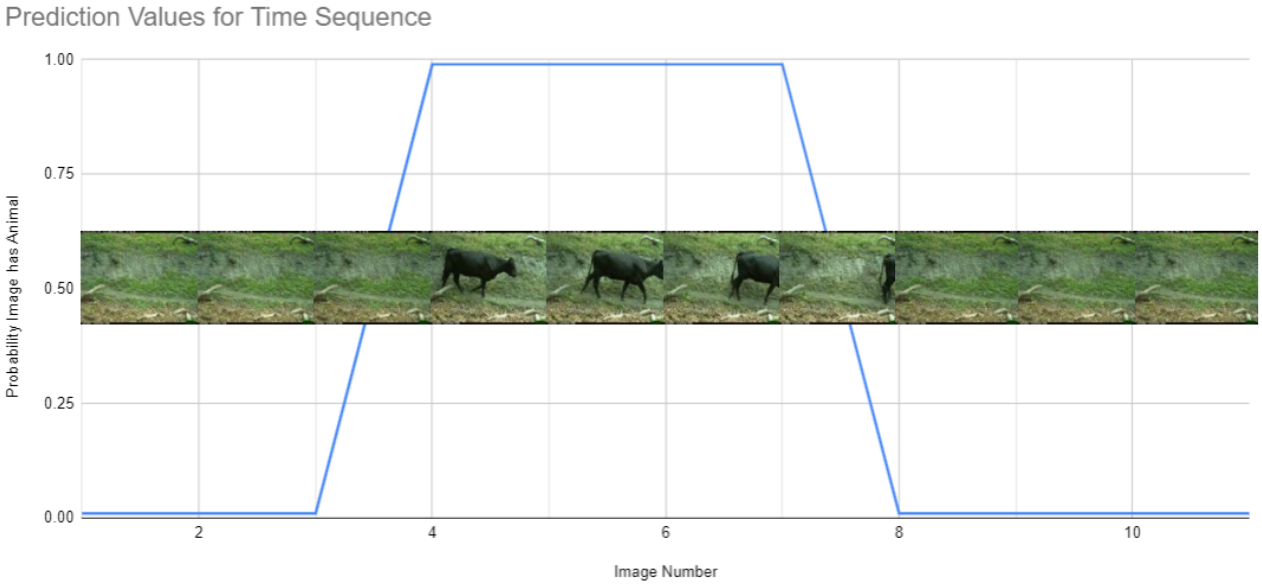


Figure 4: One of the time sequences in the testing set.



Results of time sequence test set

Conclusions and Next Steps

Prior to this project, none of the members of our group had any prior knowledge of neural networks or how to train an effective model. This hindered progress, but we are satisfied with the progress we were able to make over the course of this project. Using data from our test set, we see 87% precision and 94% recall. To put this data into perspective, if a trap camera were out in the wild taking pictures of animals over 24 hours, we would take approximately 17 images that did not contain animals and miss 1 instance of an animal moving across a frame which saves energy usage on the camera's battery.

Next steps for this project would be executing the plan to set up a simple testing bench that would enable the model to be tested in a forest environment. Using the performance from the testing benches would provide better insight into how the model could be improved. This may mean expanding the dataset to include more animals and environments.

Sources:

Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., VerCauteren, K. C., Snow, N. P., ... & Teton, B. (2018). Machine learning to classify animal species in camera trap images: applications in ecology. Methods in Ecology and Evolution.

https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network

https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564