Animal identification from remote camera images

Lewis Guignard, Noam Weinberger

June 6, 2016

1 Introduction

Currently, remote wildlife cameras capture images based on passive infrared sensing or similar simple motion detecting sensors. This may capture intended wildlife, but also captures many unintended activities such as wind, etc. False positive rates are very high, as well as true negatives. A task is relegated to the researcher to filter through large amounts of images with search requirements including species identification and count. This information is often used to track migration patterns and population dynamics. Using humans to perform these tasks is the current practice, and is extremely inefficient, in lieu of the millions of images in a modest sized biologist's dataset.

Thus we found that automating this process would provide value to the research community and yield a high impact work in computer vision. We used the Snapshot Serengeti [1] dataset, which consists of 3.2 million images of over 50 species taken over 5 years in the Serengeti ecosystem in subsaharan Africa. To date, the dataset has been classified using volunteers, and their website gives special thanks to over 28,000 participants who have assisted in the identifying process.

2 Related Work

2.1 Previous Work

The problem of identifying plants and animals from photos has been long-standing, and several methods have been implemented to attempt to address it. Some groups use geometric information known about the subject [2], while others use techniques taught in the course, such as SIFT, and sparse coding spatial pyramid to identify animals [3]. Neural Networks are a more recent contender in the space of animal identification, with inputs ranging from acoustic recordings of animals [4], to images themselves [5].

We especially note that another group published on using a deep convolutional neural network on this same dataset, during this course [5], with good results, but also a work in progress.

2.2 Key Contributions

Our work helps to ease the burden of massive human effort in the identification problem, while scaling state of the art computer vision techniques to modern dataset sizes. The neural network is still relatively young in its use cases, and so general experimentation of parameters across a variety of datasets is important, and this exploratory work of network architecture assists to that end.

3 Implementation Details

3.1 Summary of Implementation

We took a subset of the 3.2 million image dataset using only single animal images of two species, as well as images with no animals. We then manually created bounding boxes around animals in this subset. We trained using a Multi-Layer Perceptron (MLP) network and compared error of our network to human classification error rates in the Snapshot Serengeti datset in order to judge its performance.

3.2 In depth Implementation

3.2.1 Preprocessing (Munging)

Image metadata, including web URL's where the images reside, were provided by Snapshot Serengeti in .csv files. The python package 'pandas' was used to subset these metadata files for the images we were concerned with. Many photos were of animals far in the distance, occluded by objects, or only partially in the frame. Manually subsetting these out from our training set would prove both time consuming and unscalable to a system that learned over larger datasets, so we

chose to retain these images. Some other images were misclassified in the Serengeti dataset as having one animal when they had none or many, etc. We manually removed these from our subset of data.

Instead of the 50 animal types in the complete dataset, we chose to first train on two species: Zebra and Gazelle (Thomson's). We took 300 images of each species, and 900 images containing no animals from the dataset. We further subsetted to images having only 1 or 0 animals in them, as Snapshot Serengeti have metadata for the number of animals in each image. Bounding boxes were manually created around animals in our subset, using a Matlab script. Finally, images with animals were cropped to the bounding boxes around them, and all images were resized to the same dimensions, and were tested both reduced to grayscale, and in RGB.

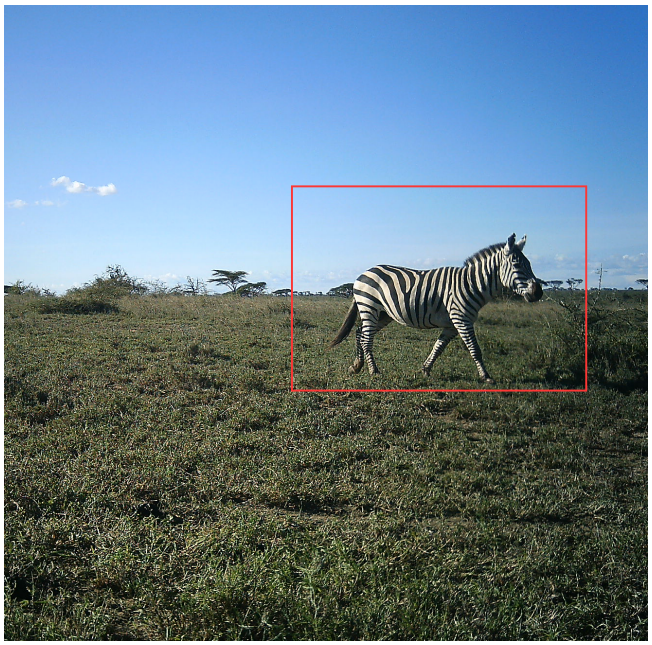

Figure 1: Sample image from dataset with bounding box

3.2.2 Parameters of the Network

We implemented a MLP network, using the python package 'lasagne', which is build on top of 'theanos.' The input layer to the network was the raw image information. In the case of RGB images, each color channel was concatenated to the others, and in grayscale, just the image matrix was provided. 20% dropout was applied to the input layer. The network hidden layers consisted of two fully connected layers, 50% dropout after each layer, and ReLU activation function to model non-linearities. The number of nodes in each dense layer were set to roughly three times the number of pixels in the image. The weights of the hidden dense layers were initialized using Glorot's scheme [6]. The output layer consisted of three nodes, one for each classification, and applied the softmax function to simulate a probability for each class. Error was measured using cross entropy loss, a standard method in neural networks used for multi-class classification. We used stochastic gradient descent with Nesterov momentum, with a learning rate of .01, and momentum of 0.9.

The network was run for 100 epochs, on input image sizes ranging from 20x20 to 300x300 pixels. Memory constraints limited the input image sizes, to a maximum of 50x50 pixels for training. The network was run on both a Intel i5 1.8 Ghz processor, and a GPU on Amazon Web Services (AWS) with 16 GB of memory. Computation time on the GPU was an order of magnitude faster than the CPU, but accuracy did not improve.

4 Experimental Results

Snapshot Serengeti claims human classsification accuracy of 96.1%, and this was our target to beat, in order for our tool to become viable. A MLP using 50x50 pixel RGB images on our dataset reported 92% accuracy. Other variants of input datasize and grayscale vs RGB yielded accuracy results in the range of 89-91%. A more detailed analysis of classification accuracy can be seen in Table 1.

	$Gazelle(label)\%$	$Zebra(label)\%$	$NoAnimal(label)\%$
Gazelle	80	12.7	7.3
Zebra	20.8	75	4.2
NoAnimal	0	.5	99.5

Table 1: Confusion matrix of images and labeling of images

The higher frequency of animal to animal misclassification suggests that the neural network is much more accurate at identifying the existence of an animal than classifying which animal is in the image.

Figure 2: Example of misclassified image

5 Conclusions

We have found that a MLP network on a small subset of the overall dataset performs nearly as well as humans (92% vs 96.6%). We anticipate higher accuracy with more images in the training set, but a drop in accuracy when including more classes to identify, leaving some ambiguity in the overall accuracy of this method when scaling to the complete Snapshot Serengeti dataset. Accuracy gains from 25x25 to 50x50 pixel images were marginal, in the 2-3% improvement range, however, including color data significantly improves accuracy. We conclude that for computational complexity, retaining color channels is preferrable to retaining pixel density, when the choice must be made.

Figure 2 shows an example of an image with a Gazelle in it misclassified as containing no animal. We speculate the model had difficulty in classifying images such as this for two main reasons: the similarity of the coloring of the Gazelle to its background, and the fact that the complete body of the Gazelle lies below the horizon. This may partially explain why the misclassification of Gazelle as no animal is nearly double that of the misclassification of Zebra as no animal, for Zebras rarely blend in with their background.

5.1 Future Work

For improvements to the model, we would explore using more preprocessing features as inputs to the system, such as SIFT. We could also input metadata of the images, such as time of day and knowledge of the sleeping distributions of different animals to increase accuracy.

We would expand the dataset to double the number of classes iteratively, watching the performance at each level, until reaching the entire 50 animal class breadth of the dataset.

We also explored the use of R-CNN's with the bounding boxes as input to perform animal detection, but could not get a working model running in the time constraints of the project. In the future, we would explore this path in more depth.

We further attempted adjusting a google TensorFlow CNN model pretrained on the ImageNet dataset, and adjusting the weights of the last layer to retrain on our own dataset, but again, ran into time constraints.

References

[1] Alexandra Swanson, Margaret Kosmala, Chris Lintott, Robert Simpson, Arfon Smith, and Craig Packer. Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific data*, 2, 2015.

[2] Neeraj Kumar, Peter N Belhumeur, Arijit Biswas, David W Jacobs, W John Kress, Ida C Lopez, and João VB Soares. Leafsnap: A computer vision system for automatic plant species identification. In *Computer Vision–ECCV 2012*, pages 502–516. Springer, 2012.

[3] Xiaoyuan Yu, Jiangping Wang, Roland Kays, Patrick A Jansen, Tianjiang Wang, and Thomas Huang. Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing*, 2013(1):1–10, 2013.

[4] E David Chesmore. Application of time domain signal coding and artificial neural networks to passive acoustical identification of animals. *Applied Acoustics*, 62(12):1359–1374, 2001.

[5] Alexander Gomez and Augusto Salazar. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *arXiv preprint arXiv:1603.06169*, 2016.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.