Deep learning tools for ecological image analysis: an example using TensorFlow and Optical Character Recognition.

To monitor biodiversity, ecologists and conservation managers need high quality temporal and spatial data on animal presence, movement, and behavior. The high costs, complex logistics, and significant expertise of human observation studies limit the ability to gather sufficient data collection for analysis. Increasingly, ecologists are turning towards greater automation to facilitate biodiversity monitoring. The combination of decreased costs of image capture and growing image analysis tools makes automated biodiversity monitoring closer than ever before.

A major obstacle to the growth of image-based ecological analysis is the efficiency of scoring and extracting information from images. Annotating images with metadata, such as time, date and location, and species identity requires significant time investment. The emerging field of computer vision can decrease the time for image annotation, increase consistency among annotators, and engage less experienced observers in biodiversity monitoring. Computer vision is a field of image-based computer science that uses image pixels to mimic human perception based on image characteristics, shape and sequence.

While computer vision has made incredible strides in a variety of fields, its growth in ecology has been slowed by a lack of access to high level algorithms. Computer vision articles are often short, terse, and filled with jargon. Source code is rarely made available, given the potential lucrative opportunities for new algorithms. While there have tremendous advances in artificial intelligence capability, ecologists have largely been left out of this revolution. The challenge lies in adapting existing tools to meet the time and budget limitations of ecologists.

The recent unveiling of the Google Cloud Machine learning platform could be a quantum leap in access to machine learning tools. Released in December 2016, google cloudML gives users access to a pre-trained image classification model, ‘Inception’. Inception is a convolutional neural network (CNN) which uses image features to classify objects into known categories. Described colloquially as ‘deep learning’ CNNs do x,y,z by x,and y. For a full treatment of the technical details of CNN see X,Y,Z.

While deep learning for ecology is itself not unprecedented, it is the availability of highly sophisticated tools for average users which holds such promise. Rather than building a model from scratch, user can ‘retrain’ Inception by teaching it new image classes, while maintaining the underlying architecture. This greatly reduces the time and expertise needed to implement image analysis solutions. The previous bottleneck was in each researcher developing his own solution independently. While the number of classes for state of the art solution is increasing (e.g), these models are unlikely to contain all labels that might benefit ecologists. For example, Inception may be able to differentiate a bear from a bird, but it may not have been taught the taxonomic resolution needed to contribute to ecological inventories. Through transfer learning…we have access to new tools.

A sample problem.

In 2013, my colleagues and I began placing time-lapse cameras in the montane cloud forests to monitor hummingbird-plant interactions. Our goal was to replace traditional human observation with image-based tools to increase the extent of spatial and temporal sampling. Ecological networks, and plant-pollinator networks in particular, are chronically undersampled, due to the rarity of many interacting partners, and the temporal pattern of plant phenology. By using cameras which turned on at 6am, turned off at 6pm, and took 1 photo per second, we can refine our understanding of the ecology of tropical communities and the co-evolution of plant-pollinator interactions. At our site at the Maquipucuna Ecolodge (), we currently have twenty cameras generating nearly 100 GB of video data per month. The videos are screened using computer vision software to find important candidate motion events, which reduces the total number of reviewed frames by >90%. These candidate frames are then scored by a human to record species identity, the time, date, and behavior of individuals. While the camera ID is encoded into the video, the timestamp of each frame is manually extracted by humans reviewing frames. Manually review of frames still is a large bottleneck in generating datasets, and limits the scalability of our approach to multiple geographic sites.



Figure 1. A sample image of a female violet-tailed sylph (aglaocercus colestris) visiting a Guzmania jaramolloi. At the bottom of each image is a camera ID, date and time stamp.

Our aim is test the performance of Google CloudML in extracting timestamp metadata from a set of 500 scored photos (Fig 1). This represents the first step in moving towards automated image analysis, and will provide a simple test case for the effectiveness and costs of this new service. Machine learning for optical character recognition (OCR) is a well-tested field of research and was a good choice for a first test of this new service.

Comparison to existing solutions

As a standard of comparison, I compared the performance of a re-trained Inception model on Google CloudML with tesseract, a highly regarded open-source tool for OCR. Tesseract is itself a machine learning tool developed at google, and is designed for transcribing text from visual images.

Discussion

Acknowledge.

The author acknowledges no conflict of interest and has no connection to Google Inc or its parent company.

Compared to tesseract OCR  
Compare to google cloud vision api.

Notes:

Tensorflow intro: <https://www.tensorflow.org/versions/r0.11/tutorials/mnist/beginners/>

https://www.ruk.si/notes/machine\_learning/gcml

Step 1. Train locally.

Step 2. Inspect logs on tensorboard - > to see the logs, docker needs to the port forwarding to be set correctly. Perhaps

docker run -it -p 8080:8080 bw4sz/cloudml

Step 3. Try on one worker

Submit job. Wait in Queue.