Alexander Buchholz

CPE646

Final Project

Bird Classifier using Backpropagation through a Stochastic Neural Network

# Abstract

For the final project, a MATLAB script was created for classifying images of birds through a stochastic backpropagation neural network. A database of 600 bird images was supplied by The Ponce Group at Beckman Institute within the University of Illinois.[[1]](#footnote-1) Note that, since this project was started, the image sets were removed from their web location. This database contains 100 images for each of 6 different types of birds─ Egrets, Mandarins, Owls, Puffins, Toucans, and Wood Ducks. The images are all different sizes, and for the most part contain a single bird located in the center of the image with no borders (with some exceptions). For clarity, Figure 1 through Figure 6 show examples of the six different birds that the Bird Classifier attempts to classify.

The Bird Classifier script loops through Windows folders to store all image data into usable image sets, performs necessary pre-processing, and uses the images to train a neural network. The script then runs test images through the classifier and writes those images to different Windows folders based on their perceived class.

While the original plan was to use C++ to develop a bird classifier, it was decided part-way into the project that MATLAB would be a more convenient choice, because a healthy amount of backpropagation MATLAB code had already been written for homework 5 in CPE646. Additionally, MATLAB has better built-in support for matrix arithmetic, which was one of the more challenging parts of the project. The project was motivated mainly by educational purposes; while this specific neural network might not be used for any real-life applications, the techniques learned and applied here have numerous applications in both academia and industry.

** **

Figure 1 - Class 1, Egret Figure 2 - Class 2, Mandarin

 

Figure 3 - Class 3, Owl Figure 4 - Class 4, Puffin

 

Figure 5 - Class 5, Toucan Figure 6 - Class 6, Wood Duck

# Methodology

Given the material taught in CPE646, there were two approaches considered when first developing the bird classifier.

The first was to develop a feature extractor and kth nearest neighbor algorithm to compare distinct, manually features of each bird. For example, toucans tend to have very large, colorful beaks with black feathers, which are distinct from an egret’s pointy beak and white feathers─ by building a feature extractor for distinct bird-parts such as beaks, eyes, and torso-feathers, training data could be used to approximate a distribution function for each of these features. Test images could then be passed through those feature extractors and distribution functions to classify the bird by comparing measured features to the trained distribution of each feature.

The other approach considered was to develop a backpropagation neural network for classifying birds─ by using raw image data as the input matrix, the hidden layer would learn to detect features during training.

The neural network option was chosen over the kth nearest neighbor approach, because the kth nearest neighbor approach was overly reliant on the developer selecting intelligent features, while the neural network trains the feature selector through the learning process.

## Functional Description

There were two MATLAB files developed for the bird classifier. The first, ‘Initialize\_Data.m’, handles all of the image selection, preprocessing, and storage.

Images from the bird database are stored in two directories: ‘birds\_training’ and ‘birds\_testing’. The first 90 images of each class are divided into six subdirectories in ‘birds\_training’, while the remaining 10 images of each class are stored in the ‘birds\_testing’ directory. ‘Initialize\_Data.m’ loops through each subdirectory in ‘birds\_training’ and applies pre-processing functions to generate the input matrix ‘X’ for training. Input X, a vector containing 540 preprocessed images, is then passed to ‘Bird\_Classifier.m’, which contains the learning loop and classifier block of the tool.

Originally the input matrix was planned to be an array containing full-sized images, padded with zeros to make input sizes consistent at 900x900 (Figure 7). This created three problems, however:

1. Arithmetic and training with full-sized images takes a significant amount of time and memory, to the point that full-sized images would lock up the laptop running MATLAB, forcing a hard reboot.
2. Since all images contained similar borders, certain matrix calculations were generating value ‘NaN’, because all images contained a black border, which resulted in dividing by standard deviation elements equal to zero.
3. MATLAB does not have intrinsic support for 3D matrix multiplication, which was required in the learning loop for RGB images.



Figure 7 - Original Toucan, and Toucan Image Padded to 900x900

To solve these problems, ‘Initialize\_Data.m’ applies the following preprocessing functions to each input image:

1. All images are scaled down by a constant variable ‘SCALE’ to reduce memory requirements and to improve program speed.
2. Instead of padding each image with zeros, images are padded by replicating their border pixels, so each image has a different border the sum of all images have non-zero standard deviations for all elements.
3. Images were converted to grayscale before being sent through training and classification.

An example of the same processed image is shown in Figure 8. While this solved all the above problems, it came at the sacrifice of losing important features, reducing the effectiveness as a classifier.



Figure 8 – 90x90 Processed and Standardized Toucan Input Image (not to scale)

As part of the backpropagation algorithm, the following sigmoid activation function is inherited from homework 5:

with a=1.716,b=2/3

Once training is complete, the 10 pre-processed test images for each of the 6 classes are passed through the network, and copies of the original images are written to directory ‘birds\_classified’.

Additional information is contained within the source code comments.

# Results

Unfortunately, while the Bird Classifier is able process inputs and sort images into outputs, there is a high error rate. This error rate is attributed to loss of features during the heavy amount of pre-processing required for the classifier. Variable values were adjusted to try and improve the success rate, but unfortunately, too much adjustment lead to non-convergent neural networks.

Based on testing, the most consistent and effective values found for input size, Theta, Eta, and Nh were:

* Input size: 90x90 (Higher values take too long to process and cause MATLAB to crash)
* Theta = 100 (Lower values do not converge)
* Eta = 0.01 (Higher values cause the network to converge too quickly, lower values do not converge)
* Nh = 10 (Changing this did not significantly improve classification success)

With these values, training has been observed to complete itself between 30,000 and 160,000 iterations, with an estimated average of roughly 40,000 iterations for training.

Outputs of the neural network are provided within Appendix A, which show the selected classification for the 60 test images (10 of each class).

While the failure rate is high, this neural network provides a lot of insight into how this classifier might be improved further. As stated before, due to the way images were pre-processed, features were lost during pre-processing. The pre-processing stage can be improved by employing other techniques to eliminate the described problems:

1. Crop the image around the bird, instead of padding. Figure 9 demonstrates how small images, because of the standard input size, end up containing more padding data than relevant feature data. This would also remove instances where multiple birds appear in one image, such as the two-Toucan example provided in Figure 7 and Figure 8.
2. Use 3D matrix multiplication to process color. Color is an important feature that could help eliminate the grayscale Egret bias.
3. Add additional layers, and turn this from a 3-layer problem into a Deep learning problem. Some good options include a convolution layer for pre-processing, a tensor flow layer for transforms, and a dimension reduction layers afterwards.[[2]](#footnote-2)
4. General code optimization could improve run-time; many of the loops can be rewritten and contained within more optimized MATLAB functions.

 

Figure 9 - Problems with Padding of Smaller Images

# How to run code

To run the code, the ‘Bird\_Classifier’ folder needs to be placed in the following MATLAB directory: ‘C:\Users\USERNAME\Documents\MATLAB\Bird\_Classifier’

USERNAME should be the user that you are logged in and running MATLAB from. For me, this is ‘buchha2’. The ‘Initialize\_Data.m’ file will edit the file path to account for your account’s user name.

The ‘Bird\_Classifier’ folder contains all needed code and relevant subdirectories. Run ‘Bird\_Classifier.m’ from within that folder; wait for the code to run (this will usually take a couple of minutes) then check the ‘birds\_classified’ folder to see results. Note that images should be deleted from the ‘birds\_classified’ folder if you would like to check the result of running a different randomly generated classifier again.

Note that no extra toolboxes or files are needed other than what is described above─ the default MATLAB library is enough.

# Team

All code written for this project, and all text, images and data provided within this document, were provided by me without assistance from other teammates. Information on MATLAB function structures was pulled from the MathWorks help pages and forums.

# Appendix: Classified Image Sets

Bird1.jpg through bird10.jpg are Egrets, bird11 through bird20 are Mandarins, 21 through 30 are Owls, 31 through 40 are Puffins, 41 through 50 are Toucans, and 51 through 60 are Wood Ducks. Ideally, in a perfect classifier, each image set would contain all 10 images in of the appropriate classes test set, and no other images. While the classifier has a fairly low success rate, it is notably better than random classification─ all classes have a higher concentration of the appropriate test set.

## Egret Set

It is noted that the classifier is heavily biased towards classifying birds as Egrets. This is likely because all images are input as lossy grayscale images, and the white torso of the Egret is getting mixed with the similarly shaped bright feathers of other birds.



## Mandarin Set



## Owl Set

## Puffin Set



## Wood Duck Set

1. Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. [A Maximum Entropy Framework for Part-Based Texture and Object Recognition](http://www-cvr.ai.uiuc.edu/ponce_grp/publication/paper/iccv05.pdf). Proceedings of the IEEE International Conference on Computer Vision, Beijing, China, October 2005, vol. 1, pp. 832-838. <http://www-cvr.ai.uiuc.edu/ponce_grp/data/> [↑](#footnote-ref-1)
2. <https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4> [↑](#footnote-ref-2)