Bird Identification using the iNaturalist Dataset

Springboard | Capstone 2 Milestone Report #2

By: Lauren Broussard

Business Problem

There are millions of different types of wildlife in the world. Can we use image classification to distinguish some of them from others? iNaturalist, a site that allows users to add images of wildlife they observe, hosts an annual Kaggle competition that encourages participants to help improve image classification for their repository of images. Successful image classification in the natural world has many benefits -- for instance, it can help conservationists and zoos to identify species in the wild without disturbing their habitat, or can be used to identify potential endangered species that are being traded. Paired with geolocation data, image classification of wildlife can also display where certain species are typically found in the world in order to help chart migration patterns or the existence of invasive species.

Sample images from iNaturalist image files.

This project uses deep learning, and specifically convolutional neural networks (CNN) to read images from 1010 different species and 6 different wildlife types. While the original Kaggle competition sought to classify one species from another - for instance, a picture of a pigeon versus a picture of an eagle, this project seeks to look at images from a more high level to classify a specific type wildlife type -- birds. We are interested in whether or not we can train a model to detect whether or not there is a bird in the photo.

Potential Client(s):

Zoos, Wildlife Conservatories: It would be useful to be able to quickly classify images of the animals they work with and/or come across. It might also be useful for wildlife protection agencies who are scanning online databases for a particular type of bird that might be being sold, kept, or traded.

Wildlife Researchers: Being able to classify birds may also help to track migratory patterns of birds, or to identify when there are certain birds in areas they would not be expected to be.

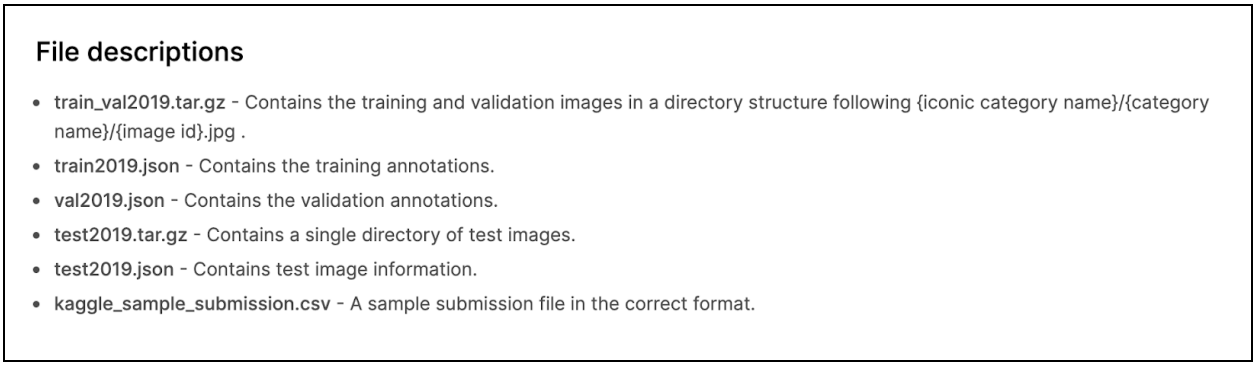
Other Clients: It could be useful for social media managers, who can use the image classification for quick tagging of their posts, or for sites to more quickly label an image for a screen reader for additional accessibility.

Data Collection and Wrangling

Dataset and Images

Data and images were downloaded from the iNaturalist competition page on the Kaggle website. The iNaturalist site allows users to upload their own images of wildlife they see. According to the iNaturalist overview on Kaggle, the images in this case were collected and validated by multiple users from the iNaturalist site. The images were part of the Fine-Grained Visualization Categorization workshop, FGVC6, and includes fewer species that may be harder to classify - as it may be an image of a green frog against other green leaves.

The original dataset included **268,243 images with annotations,** including the following files and descriptions, as described by the competition documentation:



File descriptions

- **train_val2019.tar.gz** - Contains the training and validation images in a directory structure following {iconic category name}/{category name}/{image id}.jpg .
- **train2019.json** - Contains the training annotations.
- **val2019.json** - Contains the validation annotations.
- **test2019.tar.gz** - Contains a single directory of test images.
- **test2019.json** - Contains test image information.
- **kaggle_sample_submission.csv** - A sample submission file in the correct format.

Source: Kaggle iNaturalist 2019 FGVC6 Competition/Data

For the purposes of this project, all data was initially downloaded, but the following files were primarily used to approach the classification problem:

- **train2019.json, val2019.json (57MB, 816.46KB)**: includes annotation information about each of the images, including the path to the image file, pixel height and width information, and encoded species information - i.e. kingdom, phylum, class, etc.

- **train_val2019 (79GB)**: contained in a larger train_val2019.tar.gz file, this folder includes all of the wildlife image data, stored as a jpeg file, and separated in folders by high level category (Plants, Amphibians, Birds, etc.) then species which is stored as a numeric code.

Data Wrangling

Data was imported and prepared in Python primarily using pandas. Though not always the case, this data came in largely standardized, so did not require as much cleaning as in previous projects. However, the following steps were taken to import and prepare the data for modeling.

*Loading Data***.** Annotations from both the *train2019.json* and *val2020.json* files had the following keys: 'info', 'images', 'licenses', 'annotations', 'categories'. For each of these two files, two DataFrames were created using data in 'images' and 'annotations', and the rest of the keys in the json file were ignored. For the *images* dataframe, the following fields were included: file name, image height, image width, as well as the image id. The *annotations* DataFrame included: image id and category_id, which corresponded to a code denoting the species of the image. The DataFrames were then merged, to create one master file of all data.

*Adding/Removing Columns***.** To be more descriptive, the column 'category_id' was changed to 'species_id'. Additionally, since we were interested in labeling Birds, we pulled the 'iconic category' name from the subfolders in the 'file_name' field. This created field was named "wildlife_type."

```
df.head()
```

image_id	file_name	height	width	species_id	wildlife_type
0	train_val2019/Plants/400/d1322d13ccd856eb4236c...	800	600	400	Plants
1	train_val2019/Plants/570/15edbc1e2ef000d8ace48...	533	800	570	Plants
2	train_val2019/Reptiles/167/c87a32e8927cbf4f06d...	600	800	167	Reptiles
3	train_val2019/Birds/254/9fcdd1d37e96d8fd94dfdc...	533	800	254	Birds
4	train_val2019/Plants/739/ffa06f951e99de9d220ae...	600	800	739	Plants

*Missing Data or Duplicates***.** There were no duplicated rows, or missing data in the annotation files.

Numerical Encoding. Two numerical columns were created - one binary column called 'is_bird,' that gives a 1 if the 'wildlife_type' is Birds, and 0 if it is not, The other column was named "wildlife_num" which mapped a numerical value for each wildlife type as follows: Amphibians: 1, Birds: 2, Fungi: 3, Insects: 4, Plants: 5, Reptiles: 6.

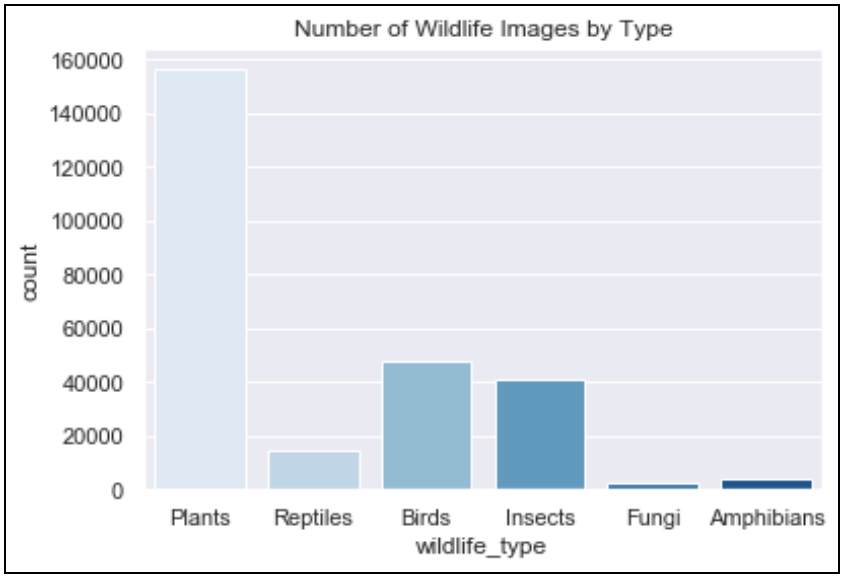
image_id	file_name	height	width	species_id	wildlife_type	type_num	is_bird
0	train_val2019/Plants/400/d1322d13ccd856eb4236c...	800	600	400	Plants	5	0
1	train_val2019/Plants/570/15edbc1e2ef000d8ace48...	533	800	570	Plants	5	0
2	train_val2019/Reptiles/167/c87a32e8927cbf4f06d...	600	800	167	Reptiles	6	0
3	train_val2019/Birds/254/9fcdd1d37e96d8fd94dfdc...	533	800	254	Birds	2	1
4	train_val2019/Plants/739/ffa06f951e99de9d220ae...	600	800	739	Plants	5	0

Exploratory Data Analysis

With the data prepared, we can look at features of the dataset and images. Overall, there are 265,213 images in our dataset, 1010 different species, and 6 different wildlife types.

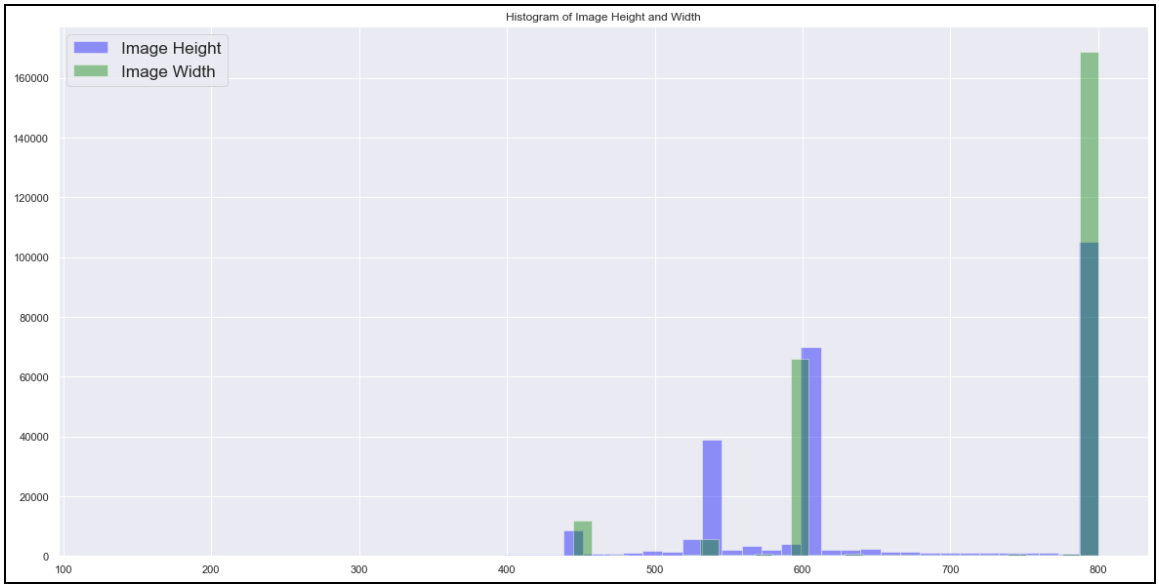
Wildlife Types & Species

The vast majority of images in the dataset (nearly 160,000) are of Plants, followed by Birds, Insects, then Reptiles and Amphibians, and finally Fungi. The column species_id is a value denoting the species of the image. **There are 47867 images of birds in 126 different species.** There is a class imbalance in our dataset, which we will remedy by taking equal samples of birds vs not birds later in the project.



Pixel Height and Width

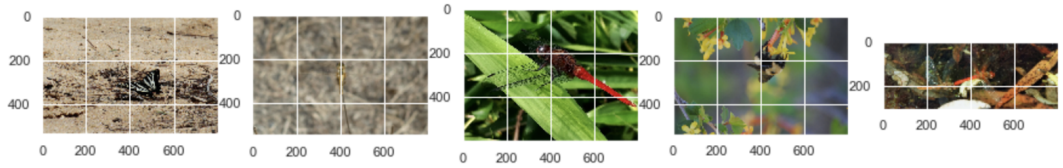
Each image has a maximum height and width of 800 pixels. The average height is approximately 663 pixels, while the average width is 720. Images in the dataset tend to be wider (horizontal) than they are taller (vertical).



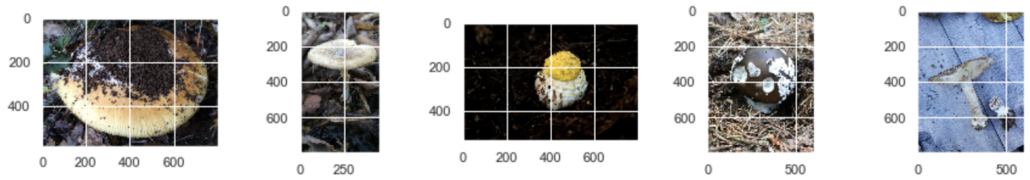
Sample Images (All Categories)

Below are sample images showing five jpeg images of each of the six categories. Many images are obscured in some way by other things, for instance, the amphibians are in water or are partly camouflaged in plants (another one of our categories). In other cases, the animal is being held in a person's hand. Being able to train a model to distinguish between the two will be useful.

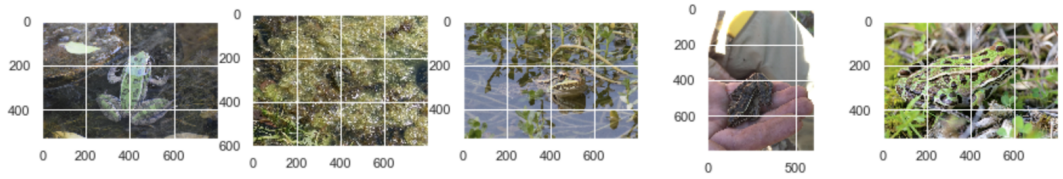
Insects



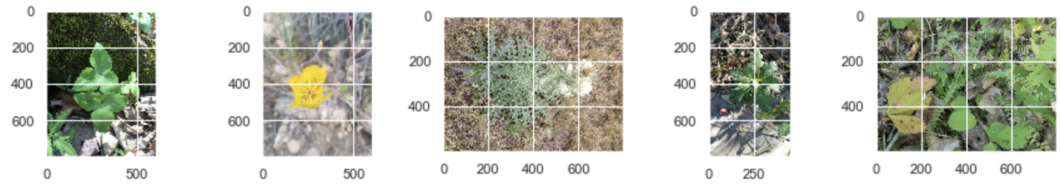
Fungi



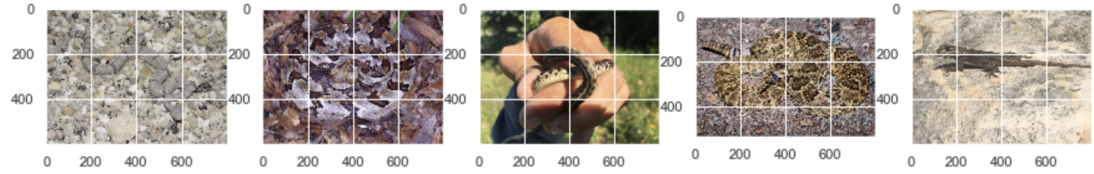
Amphibians



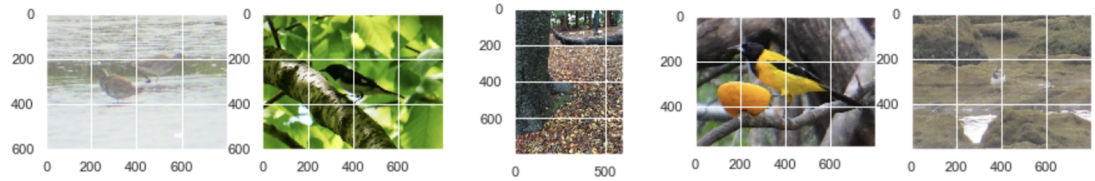
Plants



Reptiles

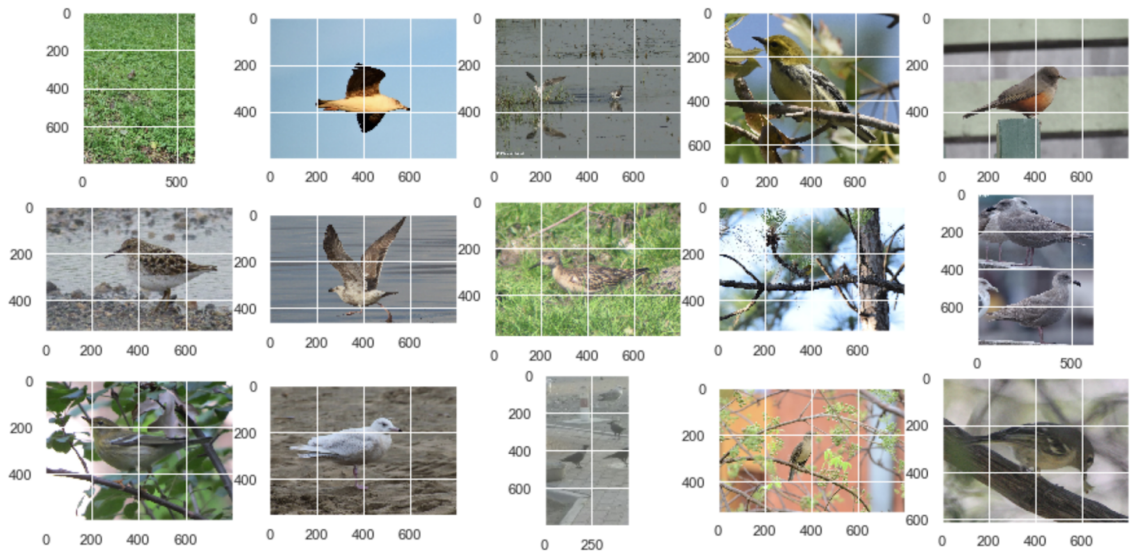


Birds



Sample Images (Birds)

Finally, since our target is to classify birds, let's look at a few more images of them in our dataset. The birds are in different environments - in grass, in the air, in trees - and in different positions -- mid-flight or perched.



In-Depth Analysis & Modeling

About Neural Networks

This project uses deep learning, specifically neural networks for image classification. The first few layers in the network look at things like the orientation of lines or simple texture, while layers further down look at more complex features.

Pre-Processing

Since the computing power on my personal laptop was limited, I ran the project end to end with a smaller amount of the image data to create the model, and gradually increased the number of images over time. Further to this end, I pulled a random subset of images to train, validate, and test with. Finally, to balance out the dataset, an equal number of images showing birds and images not showing birds were selected.

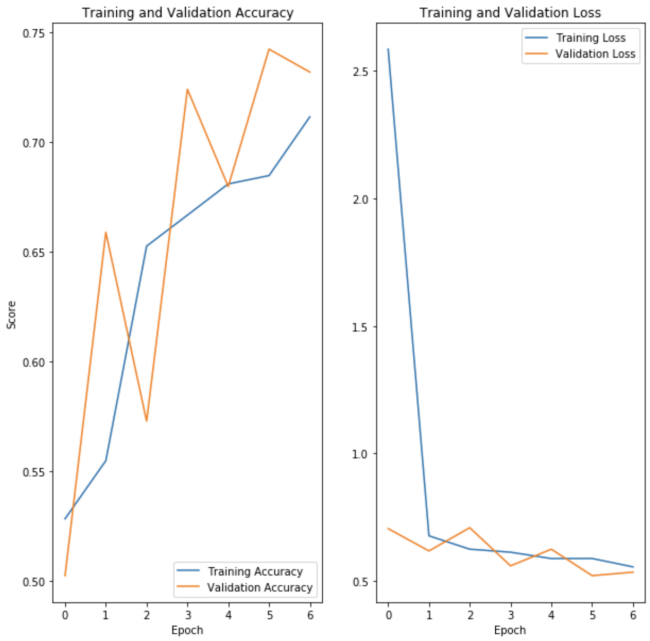
Architecture

Prior to running the models, constants were set to use in the model, including a set height and width to resize images, batch sizes, and number of epochs.

```
# Model configuration
img_dir = "../data/raw/" # where images are stored
img_width, img_height = 128,128 # resize images to account for smallest img size
batch_size = 16 # to account for small training size, can increase to 32,64,etc
no_epochs = 10
no_classes = 2 # choices of bird or not bird
patience = 3 # for Early Stopping callback
```

Initial Model - Fully Connected Network

To get a baseline of performance, a simple neural network was run. It was a fully connected network using Dense layers -- meaning every layer (pixels) is connected to all the units in the previous layer. This included five hidden layers and a flatten layer. Our first layer included 32 nodes, all hidden layers used ReLU activation (which applies a weight of 0 or 1 depending on if the corresponding input node is positive or negative), and the output layer used sigmoid activation, which is appropriate for a binary classification. I began by trying a softmax activation in the final layer, but this created static predictions (approximately 50% accuracy). Switching to a sigmoid activation, the accuracy for both training and validation data increased to 74% over 7 epochs.



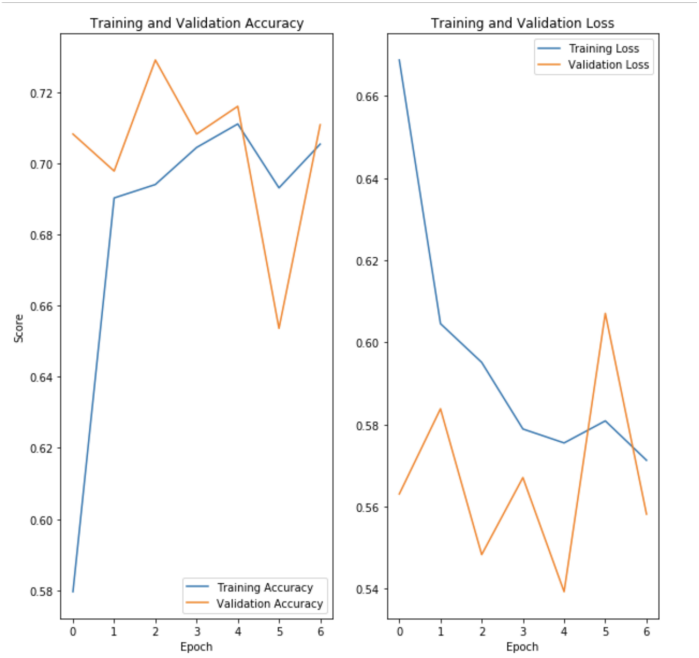

```
model1 = Sequential()

model1.add(Dense(32,activation = 'relu', input_shape = (img_width,img_height,3)))
model1.add(Dense(32, activation = 'relu'))
model1.add(Dense(32, activation = 'relu'))

model1.add(Flatten())
model1.add(Dense(32, activation = 'relu'))
model1.add(Dense(100, activation = 'relu'))
model1.add(Dense(1, activation = 'sigmoid'))
```

Subsequent Model - Convolutional Neural Network (CNN)

The second model used a series of convolutional neural networks.

It also included the following additional layers:

- **Max Pooling:** The max pooling layer can help when there are a large number of parameters, as is the case in this model. It summarizes groups of pixels based on their max value.
- **Dropout Layer:** The dropout layer randomly sets inputs to 0 at a rate of 0.6 at each step during training, which according to the documentation can help prevent overfitting.



The model achieved a validation accuracy of 74% as well. This model, despite its many layers does not seem to perform significantly better than the original, simple network. However, neither model achieves results over 80%. Throughout modeling, the following variables were tweaked for better performance: number of epochs was set to 7 using EarlyStopping, batch size was raised from 16 to 64, and optimizer was switched from ‘rmsprop’ to ‘adam.’

Prediction

Predictions were made using the second model. The following classification report gives results from our predictions.

Classification Report				
	precision	recall	f1-score	support
Not Bird	0.49	1.00	0.66	236
Bird	0.00	0.00	0.00	244
accuracy			0.49	480
macro avg	0.25	0.50	0.33	480
weighted avg	0.24	0.49	0.32	480

Resources

iNaturalist: https://www.inaturalist.org/

iNat2019 Starter Keras Code, Kaggle: https://www.kaggle.com/ateplyuk/inat2019-starter-keras-efficientnet

Kaggle Competition Information: https://www.kaggle.com/c/inaturalist-2019-fgvc6/data

LA County Threatened and Endangered Species:
https://data.lacounty.gov/Sustainability/LA-County-Threatened-and-Endangered-Species-2018-/7uw4-g37f

Keras.io Blog: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

DataCamp - Image Processing in Keras:
https://campus.datacamp.com/courses/image-processing-with-keras-in-python