Project Report - Bird Classifier

Muhammad Hazim bin Hasnan, Muhammad bin Onn, Low Yuen Seng

Abstract

The main motivation of the project is to build a bird classifier, to be incorporated into an application that can be used by amateur and novice bird enthusiasts to identify and learn more about the birds they encounter in the real world. Our application aims to replace the traditional method of bird identification, which is to do cross checking using a bird manual or the internet. We apply transfer learning and finetune a ResNet50 model using a pre-curated dataset obtained from the data science website Kaggle. Note that this report discusses only the classification model building process; details about how such application is to be implemented are purposely left out and not included.

1 Dataset

The dataset is obtained from the data science website Kaggle. At the moment, the dataset can be found via the following link: *www.kaggle.com/gpiosenka/100-bird-species*. A summary of the dataset is given below:

Type of images	Number of images
training images	27503 (unbalanced)
validation images	1000
test images	1000

There are 200 classes in the dataset. The training set is not balanced, however each species has at least 100 training image files. All images are 224 x 224 colored images in *.jpg* format; they were gathered from the internet by species name. All duplicates were removed by the author. The author also cropped the images such that each bird occupies at least 50% of the pixels.

Figure 1: Example training image — Bald Eagle

2 Model

We apply transfer learning on a ResNet50 model. The last layer of the network is replaced with a custom 200-neuron layer, with the weights initialized using Xavier initialization. Other layer parameters are all standard multiclass classification components — we use the softmax function as the activation function, and multiclass cross entropy as the loss function. All layers are frozen, except for the newly replaced output layer. The resulting number of trainable parameters is about 400,000, which is about 1.67% of the total amount of the parameters of ResNet50.

3 Training

Since we are training on general purpose CPUs and not specialized equipments such as GPUs, we have to be careful about how train our model. To keep training time short, we do not apply image normalization and

augmentation. We considered performing grayscale conversion to increase computation speed, but decided not to do so because we thought color is an important feature when it comes to classifying birds.

We trained the model for two epochs, using a batch size of 10; the model trained for a total of 55 000 iterations, in other words, the weights of the model were updated a total of 55 000 times. Each epoch took about 1.5 hours to train on an Intel Core i5 processor (2018 Surface Pro 5).

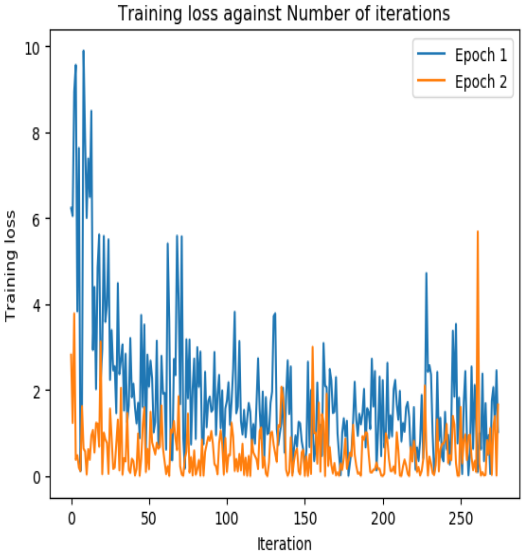

Figure 2: Training loss graph

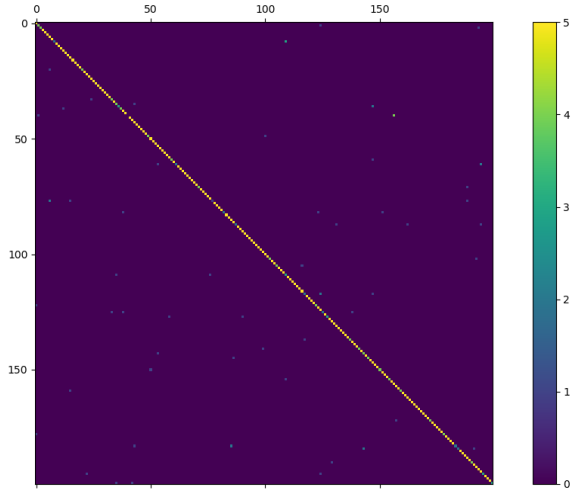

Figure 3: Test set confusion matrix

4 Evaluation

Here are the evaluation metrics of the model after training, evaluated on the test set:

Performance Measure	Pre-Finetune	Epoch 1	Epoch 2
Accuracy	0.0020	0.8630	0.9350
Precision	0.0047	0.9185	0.9511
Recall	0.0020	0.8630	0.9350
F1 score	0.0039	0.8715	0.9342

5 Possible Improvements

1. **Train on more epochs.** Jump in accuracy at the end of the second epoch of training is considerably high, suggesting that the model has not reached the saturation point yet. Further training is very likely to yield more increase in the model accuracy.

2. **Unfreeze more layers.** ResNet50 was trained on classifying general objects, which is somewhat dissimilar from classifying birds. Hence, the model performance is likely to improve if a few more layers are unfrozen during finetuning.

3. **Image normalization and transfomartion.** Image preprocessing is one general ML technique that can improve model performance.

4. **Hyperparameter tuning.** We can play around with the hyperparameters such as batch size, momentum, and number of epochs and see which one improves the model accuracy, using a validation set.

5. **Try different architectures.** The Free Lunch Theorem states that there is no "best" ML model, hence one could try out different model architectures and see if any of them work better at the task at hand than the current model.