Training Neural Networks to Classify Bird Calls

Pete Gray[*]
ptgray@gmail.com
ptgray@my.yorku.ca
217653247
CSML1020 - Machine Learning at Scale - York University School of Continuing Studies
Toronto, Ontario, Canada

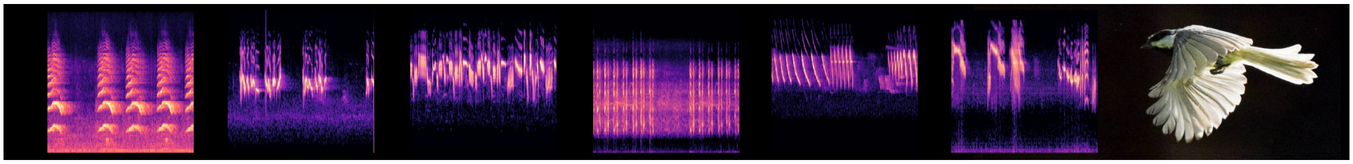


Figure 1: An assortment of spectrograms and a Bird. Photo by Pete Gray

ABSTRACT

Neural Networks can be trained to recognize bird calls. In this project, a number of approaches to this task are taken. Accurate classification of numerous species of birds requires computationally intense training of deep algorithms. Approaches to achieving computational intensity are explored in this project as well.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Machine learning approaches**; **Neural networks**; **Machine learning algorithms**; *Artificial intelligence*; *Philosophical/theoretical foundations of artificial intelligence*; Cognitive science;

KEYWORDS

machine learning, convolutional neural networks, training at scale, feature extraction, audio classification

ACM Reference Format:
Pete Gray. 2020. Training Neural Networks to Classify Bird Calls. In *Proceedings of Machine Learning at Scale (CSML1020).* ACM, New York, NY, USA, 4 pages. https://doi.org/na

1 INTRODUCTION

There are a number of commercially available apps that can recognize bird song in the field. Training a neural network to recognize audio signals is a common task in the field of machine learning. Data from a 2013 Kaggle Competiton [2] are used in this project, however the machine learning techniques used here are perhaps more modern, but certainly quite different.

Cloud-based GPUs are used to train models that perform accurate classification for the full range of bird species represented in the data. This cloud-based training is compared thoughtfully, if not accurately, to similar training on local computers.

2 OVERVIEW

Several approaches to audio classification are explored. Two approaches are explored in depth, with one of those approaches being fleshed out into a deep exploration of hyperparameter tuning, data augmentation, and achieving high model performance by leveraging the processing power of cloud-based computing platforms.

Because this is a project for an academic course, and because the definition of the assignment strongly suggests that the journey of exploration is more important than the quantifiable utility of the results, an unusually disproportionate amount of time and thought will be put into abstract intuitions that would fall into the "put yourself in the math's shoes" category.

2.1 Commercial Implementations

The use of trained neural networks to identify bird calls is common. A number of companies and organziations have produced apps that allow a user to sample audio in the field and perform inference on that audio in order to identify the species of birds that are audible in the sample.

- Merlin Bird ID by Cornell Lab [4]
- Song Sleuth [6]
- Chirp-O-Matic [1]

One possible commercial venue for the results of this project would be to produce a simple bird recognition tool that can be embedded on a web page. Rather than pay to install an app, users could visit the page, be exposed to an advertisment, and have their local bird species identified free of charge. Should this product be developed, it will be called Cheapcheep.

[*] Pete prepared this report, working individually, for the final project in CSML1020 Machine Learning at Scale.

CSML1020, York University School of Continuing Studies,
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/na

2.2 Interaction with Human Learning

One insight into machine learning that these classification tools provide is an illustration of the power of labeling data. Humans learn to recognize bird calls by hearing them in the field. Without knowledge of which bird they are hearing, the ability to identify the species by name will not develop. Traditionally, humans have relied on other humans to tell them what type of bird they are hearing - this becomes a "label" which can be used to infer the species of a bird in the future. Bird call recognition apps can perform this data labeling function, helping humans become instant experts in bird calls, without having to pick the brains of knowledgable humans to get there.

3 APPROACHES

Several approaches were considered. Two approaches were explored. All approaches had two things in common - a feature extraction phase before model training, and a deep neural network employed at some point during the process.

3.1 Spectrograms and a ConvNet

In this approach, feature selection is performed with Librosa, and training and inference are done with a convolutional neural net (ConvNet) [7] [3]

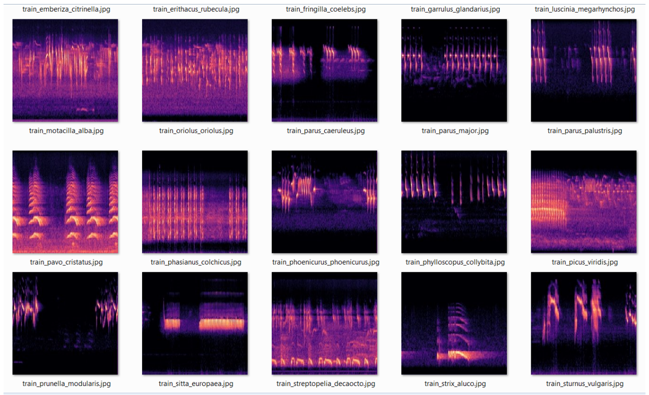

train_emberiza_citrinella.jpg
train_erithacus_rubecula.jpg
train_fringilla_coelebs.jpg
train_garrulus_glandarius.jpg
train_luscinia_megarhynchos.jpg
train_motacilla_alba.jpg
train_oriolus_oriolus.jpg
train_parus_caeruleus.jpg
train_parus_major.jpg
train_parus_palustris.jpg
train_pavo_cristatus.jpg
train_phasianus_colchicus.jpg
train_phoenicurus_phoenicurus.jpg
train_phylloscopus_collybita.jpg
train_picus_viridis.jpg
train_prunella_modularis.jpg
train_sitta_europaea.jpg
train_streptopelia_decaocto.jpg
train_strix_aluco.jpg
train_sturnus_vulgaris.jpg

Figure 2: A selection of MFCC spectrograms generated from audio of bird calls by Librosa.

These spectrograms represent the intensity of different sound frequencies plotted through the duration of the audio samples. It can be clearly seen that different species of birds produce calls that result in visually distinct spectrograms. This separation of frequency intensity provides the features, extracted, that enable the model to train and infer effectively.

Using these spectrograms as inputs,a convolutional neural network can be trained to perfrom image classification.

For training of models at scale, many samples are taken from the audio files and converted to spectrograms. Using "sliding windows" as one would with image data, many samples can be generated up and down the time index of the audio file.

3.2 Feature Extraction with a Pre-trained VGG-19

This approach is much more mysterious than the first, and serves best as a technological curiousity and a point of reference in discussions about explainability.

A VGG-19 model with pre-trained ImageNet weights is used to extract features from the raw audio. [5] To conform to the VGG-19's input shape, a 224x244 array is created, with one axis being time, and the other being the intensity of the audio signal. The output of the model is flattened, resulting in a long vector with 25,088 elements, most of which are zeroes.

What this vector represents is anyone's guess. The VGG-19's idea of features. As the neural network was trained using labeled photographs, one wonders how it must be interpreting these spectrograms. That it's notions get squished down into a linear feature vector makes it even more mysterious.

In this approach, the model that is used for inference is a support vector classifier. (SVC). It is worth noting that the feature vectors emitted by the VGG-19 bear a strong resemblance to the feature vectors emitted by TF-IDF (term frequency, inverse document frequency). which are often used with an SVC in the field of natural language processing (NLP).

3.3 Comparison of Approaches

The main differences between these two approaches fell into 3 areas:

- Power and ease of implementation
- Explainability
- The possibility of playing with models in the cloud

3.4 Power and Ease of Implementation

The VGG-19 approach was very simple to set into motion. Audio samples are fed into the neural net for inference, one at a time, so it takes a while but does not demand exceptional computaional resources. The training of the SVC happens very quickly and achieves very high validation accuracy (>80%) on first running.

Using the spectrograms with the untrained network requires model configuration just to start training. It is very easy to overwhelm the computational resources by trying to train with too much data. Most early attemps resulted in very low validation accuracy (<50%).

3.5 Explainability

While convolutional neural networks themselves aren't exactly the acme of explainability, the mystery surrounding the training features emitted by the pre-trained VGG-19 adds a whole new dimension to that.

It is most interesting that despite its mysterious nature, this approach is able to yield more accurate results so easily. This speaks to the power of neural networks, while highlighting their capacity to be inexplicable.

3.6 The Possibility of Playing with Models in the Cloud

While compelling in its power, the approach that uses a VGG-19 for feature extraction doesn't provide the opportunity to play with deep models. Without some understanding of the feature vectors it emits, it seems perplexing to imagine re-training the model for improved performance.

The approach that uses spectrograms, on the other hand, provides an opportunity to monkey around with every aspect of raw untrained neural networks.

This approach is therefore chosen for scaling up and playing with models in the cloud.

4 SCALING UP

The ability to train models to recognize bird calls is limited by the processing power of ordinary computers. By using cloud-based GPUs, both volume of data and number of training cycles can be dramatically increased, allowing meaningful models to be trained in very short periods of time.

4.1 Preparing to Scale Up

Programming for data preparation and model training was prepared on a small scale offline before implemention in the cloud. The implementation that was successfully trained and played with in the cloud worked well and was ready for rapid changes to be made to how it worked.

4.2 Increasing Training Data

The offline implementation only considered 15 species of birds. It trained slowly with 4 spectrogram samples for each. After setting it up on a cloud-based GPU, it was easily able to train on 10 samples from each of the 35 species in the dataset.

4.3 Tuning the Models

With much more data and the potential for deeper training cycles, hyperparameter tuning provided a variety of ways to improve the training of the models.

It was also observed, with some delight, that reasonasble training could occur over a very short time period, allowing the programmer to develop more of a "groove" in which he can bounce ideas quickly off the code to see what sticks.

4.4 Batch Size, Step Size and Number of Epochs

These could be increased in the cloud environment, There seems to be a sweet spot, where a balance of the three yields the best results over a given training time.

4.5 Learning Rate, Decay, and Momentum

With different configurations of batch and step size, adjusting the learning rate slightly from the default value was found to be helpful. It was also seen to be helpful to customize the decay rate of the learning rate to harmonize with the base rate and the number of epochs.

Momentum values were also tweaked experimentally, and while an effect on training was observed, it was difficult to see that effect as being helpful.

4.6 Different Optimizers

Models were compiled successfully with two different optimizers - RMSprop and Adam. Adam generally required a much higher learning rate than RMSprop. SGD (stochastic gradient descent) was also tried, but results were most of the way back to a dummy classifier.

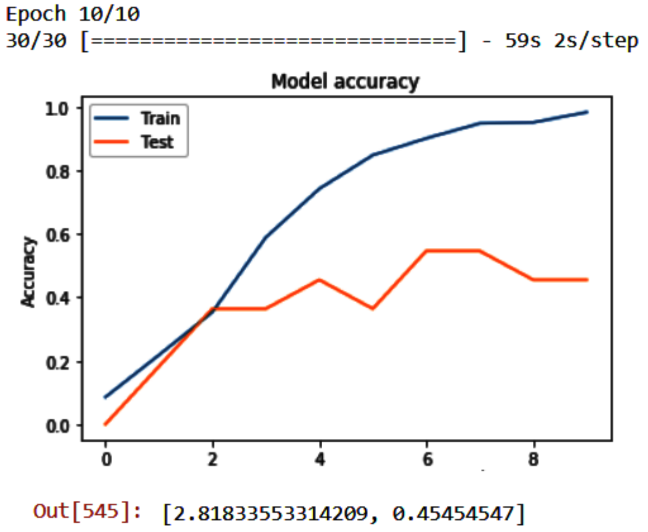
5 COMPARING CLOUD GPU PERFORMANCE

Many things are different, between training on the laptop and training in the cloud. Amount of training data, quantity and arrangement of training cycles, and many hyperparament settings.

To get a feel for the difference in performance between a laptop and a cloud GPU, we will isolate and compare a small set of factors:

- Number of classes
- Validation accuracy of model
- Training time

The following two figures show a stark contrast. In the first, 15 classes are trained on a laptop (8-core i5), to 45% accuracy, in 10 minutes. The second shows a model training to classify 35 species, on a cloud-based GPU, to 93% accuracy, in 1 minute.



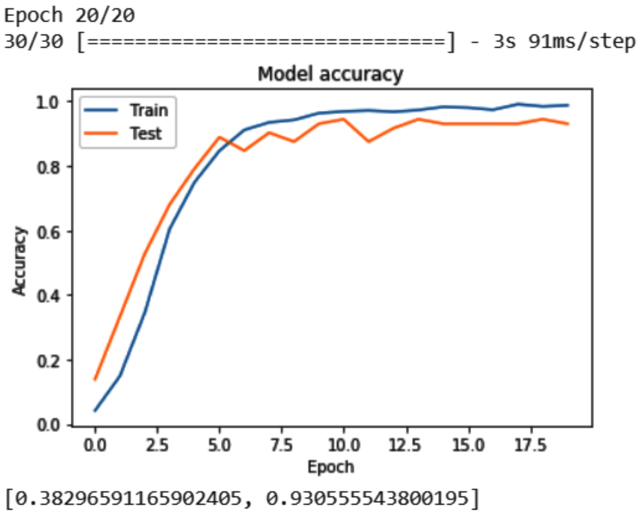
```
Epoch 10/10
30/30 [==============================] - 59s 2s/step
```

```
Out[545]: [2.81833553314209, 0.45454547]
```

Figure 3: Training on an 8-core i5 laptop, 45% validation accuracy, 10 minutes.

The dramatic increase in performance can largely be attributed to the much greater volume of training data. With smaller training sets, the models being trained on the laptop were topping out at very low numbers, and further training did not change this.

The quicker processing on the cloud GPU allowed the models to train much more quickly, even though they were processing much more data. Based on this example, accounting for both volume of data processed and time to return results, the difference appears to be well above two orders of magnitude. It truly made the difference



```
Epoch 20/20
30/30 [==============================] - 3s 91ms/step
```

```
[0.38296591165902405, 0.930555543800195]
```

Figure 4: Training on Google Cloud GPU, 93% validation accuracy, 1 minute.

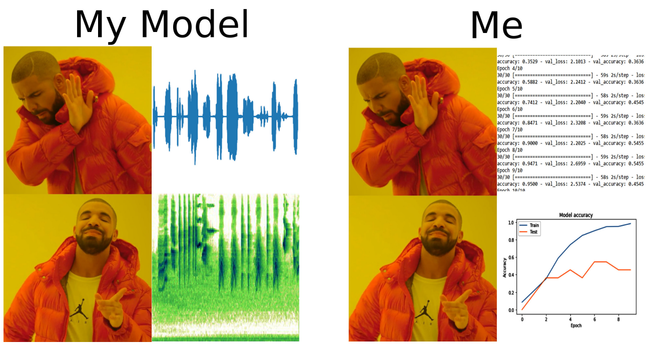
between training inadequate models slowly, and training reasonably competent models quickly.

6 GRATUITOUS WHIMSICAL INTUITION

From the "put yourself in the math's shoes" file:

As work on this project unfolded, an interesting parallel became apparent. The machine learning algorithms have a difficult time learning from raw, unprocessed sequence data. We therefore do feature extraction before training a model. Machine learning programmers can make better, easier inferences about how their models are training by looking at a graph of training data, rather than the raw sequence of numbers that appears during training.

This meme was created to distill the parallel into a simple idea:



```
accuracy: 0.3529 - val_loss: 2.1013 - val_accuracy: 0.3636
Epoch 4/10
30/30 [==============================] - 59s 2s/step - los
accuracy: 0.5882 - val_loss: 2.2412 - val_accuracy: 0.3636
Epoch 5/10
30/30 [==============================] - 58s 2s/step - los
accuracy: 0.7412 - val_loss: 2.2040 - val_accuracy: 0.4545
Epoch 6/10
30/30 [==============================] - 59s 2s/step - los
accuracy: 0.8471 - val_loss: 2.3208 - val_accuracy: 0.3636
Epoch 7/10
30/30 [==============================] - 58s 2s/step - los
accuracy: 0.9000 - val_loss: 2.2025 - val_accuracy: 0.5455
Epoch 8/10
30/30 [==============================] - 59s 2s/step - los
accuracy: 0.9471 - val_loss: 2.6959 - val_accuracy: 0.5455
Epoch 9/10
30/30 [==============================] - 58s 2s/step - los
accuracy: 0.9500 - val_loss: 2.5374 - val_accuracy: 0.4545
Epoch 10/10
```

Figure 5: Empathizing with ML models on raw data and feature extraction.

Making graphs is like feature extraction for humans.

7 SUMMARY

Two approaches to audio classification with deep learning were explored. One used MFCC spectrographs as input for a convolutional neural net of the type used for image recognition. The other approach used flattened output from a pre-trained VGG-19 to extract features for an SVC to use for classification. The spectrograms and convnet approach was fleshed out in the cloud, where it was possible to quickly train a variety of models to above 90% validation accuracy.

8 FUTURE WORK

The models used for classification in one appoach were very similar to the models used for feature extraction in the other approach. Combining the two into a single neural net that can do the feature extraction and the classification would be interesting. It would also facilitate the development of Cheapcheep by only requiring a single piece of technology to be deployed, that can recognize bird calls all on its own.

REFERENCES

[1] Chirp-O-Matic. 2020. Chirp-O-Matic. http://www.chirpomatic.com/
[2] Kaggle. 2013. The ICML 2013 Bird Challenge. https://www.kaggle.com/c/the-icml-2013-bird-challenge
[3] Kaggle. 2018. Beginner's Guide to Audio Data. https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-audio-data
[4] Cornell Labs. 2020. Merlin Bird ID by Cornell Lab. https://merlin.allaboutbirds.org/
[5] Asad Mahmood. 2019. Audio Classification with Pre-trained VGG-19 (Keras). https://towardsdatascience.com/audio-classification-with-pre-trained-vgg-19-keras-bca55c2a0efe
[6] Song Sleuth. 2020. Song Sleuth. https://www.songsleuth.com/#/
[7] Adrian Yijie Xu. 2019. Urban Sound Classification using Convolutional Neural Networks with Keras: Theory and Implementation. https://medium.com/gradientcrescent/urban-sound-classification-using-convolutional-neural-networks-with-keras-theory-and-486e92785df4