CS230

Using CNNs to Recognize Bird Species by Song

Utuq Ablikim
Stanford University
utuq@stanford.edu

John Ngoi
Stanford University
johnngoi@stanford.edu

Patrick Liu
Stanford University
pliu1@stanford.edu

Abstract

The ability to identify bird species on a large scale can provide vital information about our ecosystems and biodiversity [1]. The quantity and composition of bird species can reflect regional ecological environment and indicate the health of changing ecosystems [2]. However, recognition of a large number of species based on song audio has proven to be challenging. In recent years, deep learning techniques such as convolutional neural networks (CNN) have been successfully applied in many arenas of science [3] and provided efficient methodologies, such as pattern recognition, object detection in images [3, 4], and classification of patterns in audios [5] and speech recognition. In this CS230 project, we utilized the bird song audio spectrogram data-set provided by the Biology department of Stanford and created a machine learning model that can identify four types of bird species with as high as 0.886 F1 dev score.

1 Introduction

Bird population, species diversity, migration, and geographical distribution are important factors for maintaining and protecting biodiversity, as well as monitoring of the health of our ecosystem [1, 2]. To correctly identify and classify bird species on a large scale is a crucial step for keeping track of bird species diversity. However, conventional methods for identification of individual bird species and classification of mass bird species have been proven to be time-consuming, and classification results often have limited accuracy. Recent computational advancements in machine learning and deep learning have been extremely efficient and accurate in classification of objects, animals, and plants from within large data sets.

For this CS230 project, our team collaborated with Kelley Elizabeth Langhans (researcher) and Andreas Paepcke (CS project lead) from Biology department of Stanford University to tackle the bird species identification problem using large bird song audio recordings files. The input data set is consisted of approximately 10,000 bird song grayscale spectrogram and from a total of 14 different bird species. As an initial step, our goal is to be able to identify 4 different species of birds from the dev data with reasonably high accuracy.

2 Prior Work

Machine learning approaches such as nearest neighbor matching [6] and decision trees [7] have been widely used in previous studies of bird species identification. The most extensive deep learning work is the BirdNet, a Cornell lab research work that used the artificial neural network to rank the most likely bird species from bird song recordings. BirdNet is capable of classifying 984 bird species and it has overall test accuracy of 91.5% [8].
In other studies, transfer learning from ResNet-50 CNN was employed to a smaller target data set

and achieved 79% average validation accuracy [9]. Due to the image file format of original data and additional resource needed for code development, other CNN networks were not validated against the test data of this project, see Next Steps.

3 Dataset and Features

The total amount of training data provided is 10,483 spectrogram files and the dev data consists of 1165 files. The top four species with the most training data available are:
Arremon aurantiirostris (ARRAUR-S): 1136 spectrogram files
Dysithamnus mentalis (DYSMEN-S): 1364 spectrogram files
Henicorhina leucosticta (HENLES-S): 2212 spectrogram files
Lophotriccus pileatus (LOPPIT): 1567 spectrogram files
Other bird species (OTHER): 4204 spectrogram files
Initial data is prepared and pre-processed by Andreas. Further processing and augmentation of the training data is planned for next step and will be updated in the final report. Furthermore, the collaborating team at Stanford is continually working on collecting more data and field recordings which we plan to use to augment our model's performance. Figure 1 shows an example of spectrogram that belongs to HENLES-S and Figure 2 shows a spectrogram from DYSMEN-S.

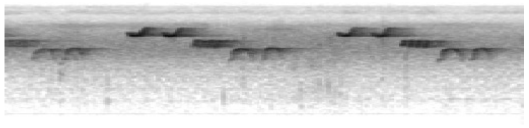
Figure 1: Sample image spectrogram from bird species: HENLES-S

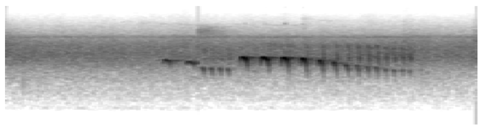
Figure 2: Sample image spectrogram from bird species: DYSMEN-S

4 Method and Model Architecture

Overall, our model loads in spectrogram data and outputs a length-5 vector predicting which of the 5 classes the spectrogram belongs to. As a multi-class classification task, we start by using cross-entropy loss as a training metric for our model, with an Adam optimizer. We chose F1 score as our evaluation metric on our model's performance because it better evaluates performance over imbalanced classes of data.

The input images are uniformly transformed to 512x128 pixels in size and normalized for training of the model. Since the goal is image classification, we use a CNN as our model architecture. Our baseline neural network consists of three alternating convolutional layers and max pooling layers, followed by three fully connected layers with dropout. Apart from those layers, batch normalization is applied after each convolutional layer. We found that the baseline achieved a maximum train set F1 score of 0.7 and dev set F1 score of 0.67, which is indicative of a high bias situation.

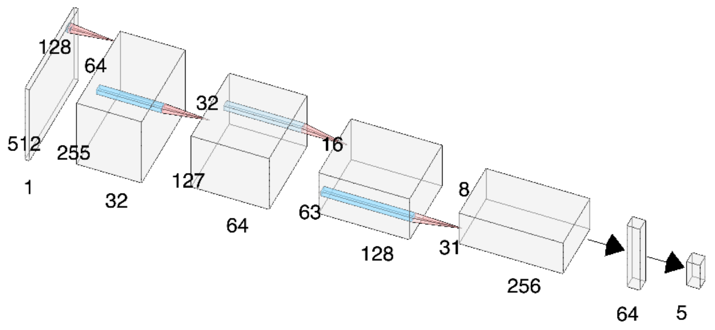

Figure 3: CNN v2

4.1 Expanded Convolutional Network

Since the baseline had a high bias problem, we increased model complexity by adding convolutional layers. Expanding to 4 and 5 convolutional layers resulted in a high variance problem, with very high training performance but mediocre dev performance. Thus, our later models work with 4 convolution layers since it is the lowest number that is complex enough to learn this problem.

The result of the expanded CNN model shown in Table 1 is presented in the Experiment section, Table 2. The training F1 score reached 0.99, indicating there is an overfit and dev F1 score reached 0.883, which is significantly higher than previous model.

#	Input			Output			Layer	Stride	Pad	Kernel	in	out	Parameters
1	512	128	1	255	64	32	Convolution	2	1	(3,5)	1	32	512
2	255	64	32	255	64	32	BatchNorm				32	32	64
3	255	64	32	127	32	64	Convolution	2	1	(3,5)	32	64	30784
4	127	32	64	127	32	64	BatchNorm				64	64	128
5	127	32	64	63	16	128	Convolution	2	1	(3,5)	64	128	123008
6	63	16	128	63	16	128	BatchNorm				128	128	256
7	63	16	128	31	8	256	Convolution	2	1	(3,5)	128	256	491776
8	31	8	256	31	8	256	BatchNorm				256	256	512
9	31	8	256	64	1	1	Fully Connected				63488	64	4063296
10	64	1	1	64	1	1	Dropout				64	64	0
11	64	1	1	5	1	1	Fully Connected				64	5	325
	Total												4710661

Table 1: CNN v2 Structure.

4.2 Data Augmentation

Data augmentation methods are frequently used to expand the dataset, which may prevent overfitting and achieve better dev results. Due to the nature of the sound spectrograms, the training images cannot be flipped, sheared or scaled without distorting the data beyond auditive recognition. Thus, during the training of the model, vertical and horizontal shifts were experimented as augmentation methods. Each training spectrogram image was shifted vertically, horizontally, or both by a small amount such that important features stayed within the bounds of the image. However, with each augmentation method, the F1 score/dev accuracy resulted in poorer test-time results than before.

4.3 Weighted Cross Entropy Loss

Since the training data is unbalanced for the classes of bird species, we assign class weights to balance performance across the species. The loss calculated with weighted average is the following:

$$loss = \frac{\Sigma_{i=1}^{N} weight(class[i]) loss(i, class[i])}{\Sigma_{i=1}^{N} weight(class[i])}.$$

Since the OTHER class has the most data points, we reduce the weight of our OTHER class from 1.0 to 0.4, which generally deincentivizes predicting the OTHER class; especially with simpler models, not weighting the data often led the model to predict the other class. Using our later models, we compared weighted and unweighted cross entropy loss and found that the performances are similar. The loss curves highly resemble each other, other than a period of slow improvement at the beginning of training with unweighted loss. As a result, we keep the weighted loss function to since it seems to speed up training with little impact on the final performance.

4.4 Max Pooling

It is common to have a max pool after the convolutional layer, however, in our case, adding max pooling hurt the dev performance. This may be due to the class imbalance and also the relatively small data sets, that caused the model to overfit on the training and perform poorly on the dev set. In our final architecture, the model no longer has any max pooling layers, since it performed better without max pooling layers.

4.5 Advantage of Noise

When setting the batch size to be much smaller or even 1, we can observe more noise in the loss and training performance. Given the goal here is to find a set of parameters for the model that performs best on the dev set, we took advantage of this noise and captured a set of parameters that performed best and saved it to disk. It really didn't matter if we ran 10,000 epoch training, if we found the best performance against the dev set at 200th epoch, that was the parameter combination for the model that mattered.
In a more classic sense, if we have training, dev and test sets, we would want to capture the best performing parameters for the test sets as the performance of the test set is more important than the dev set. Ultimately, it is most important how these parameters fair in real world applications.

5 Code

The project mainly consisted of data preparation script "data.py", training script "train.py", neural network structure module "cnn.py" and evaluation script "evaluation.py". To improve the team velocity, we made it easier for us to try various hyperparameters by adding support for hyperparameter arguments when running the training. For example, if we wanted to change hyperparameters for a training session, we could do the following from command line.

» python train.py batch_size=16 learning_rate=0.01num_epoch=3 dropout=0.2

In the final part of the project, we have expanded our command line arguments to 11.

» python train.py model_version=2 test_f1_score_threshold=0.80 augment_data=False cnn_activation=relu layers=4 conv_kernel_size=3,5 batch_size=4 learning_rate=0.0001 l2=0.01 num_epoch=200 dropout=0.3

We chose PyTorch as the machine learning framework for the project and the live system was built on AWS.

6 Experiments

The table below shows the experiments against various models and hyperparameters combinations, and the respective training and dev performance.

Experiment	Kernel size	Batch size	Learning rate	L2	Dropout	Train F1	Dev F1
1	(3,5)	32	0.0001	0.01	0.3	0.986	0.815
2	(3,5)	16	0.0001	0.01	0.3	0.995	0.837
3	(3,5)	8	0.0001	0.01	0.3	0.994	0.862
4	(3,5)	4	0.0001	0.01	0.3	0.971	**0.886**
5	(3,5)	2	0.0001	0.01	0.3	0.982	0.876
6	(3,5)	4	0.0001	0.01	0.4	0.983	0.877
7	(3,5)	4	0.0001	0.01	0.2	0.987	0.878
8	(3,3)	4	0.0001	0.01	0.3	0.991	0.873
9	(3,5)	4	0.0001	0.01	0.4	0.990	0.838
10	(3,5)	4	0.001	0.01	0.3	0.754	0.666

Table 2: Experiments.

7 Results

Across our experiments, we found that experiment 4 performed the best on the development set. Using 4 convolutional layers with 3 by 5 kernels, ReLU activations, learning rate of 0.0001, no data augmentation, L2 regularization constant of 0.01, and dropout of 0.3, this experiment reached a dev set F1 score of 0.886. The confusion matrix for the train and dev set from the best performed model are given in table 3 and 4.

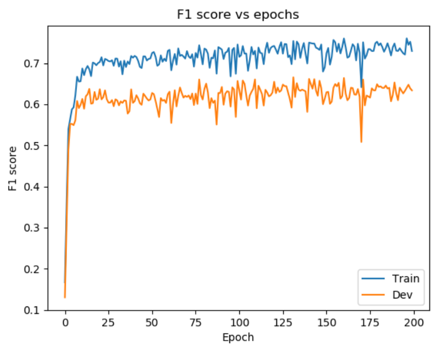

Figure 4: Experiment 10, batch size: 4, dropout: 0.3, kernel size: 3x5, learning rate: 0.001

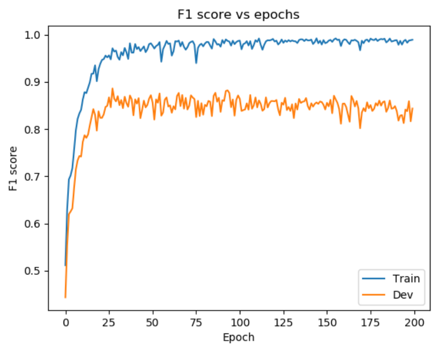

Figure 5: Experiment 4, batch size: 4, dropout: 0.3, kernel size: 3x5, learning rate: 0.0001

		Predicted					
		ARRAUR_S	DYSMEN_S	HENLES_S	LOPPIT	OTHER	Total
Actual	ARRAUR_S	1113	0	2	7	14	1136
	DYSMEN_S	1	1344	2	1	16	1364
	HENLES_S	0	1	2185	1	25	2212
	LOPPIT	0	1	1	1557	8	1567
	OTHER	65	95	45	21	3978	4204
	Total	1179	1441	2235	1587	4041	

Table 3: Experiment 4 predicted species vs. actual labels (train set)

		Predicted					
		ARRAUR_S	DYSMEN_S	HENLES_S	LOPPIT	OTHER	Total
Actual	ARRAUR_S	105	3	1	2	15	126
	DYSMEN_S	1	133	2	0	16	152
	HENLES_S	0	2	232	2	10	246
	LOPPIT	0	2	4	160	9	175
	OTHER	11	24	18	8	405	466
	Total	117	164	257	172	455	

Table 4: Experiment 4 predicted species vs. actual labels (dev set)

8 Challenges and Next Steps

One difficulty arises because the OTHER class consists of songs from several bird species which may be much more dissimilar from each other than from the other four classes. Throughout our development process, we consistently observed that our model performed relatively poorly on the OTHER class, based on the training and dev set confusion matrices, such as those in Tables 3 and 4. Going forward, we may dissolve the OTHER class into its constituent species.

Using audio spectrograms of bird songs for classification of bird species is a novel project topic and thus has not been studied widely. Hence, pretrained spectrogram models are not easily available. Although audio pattern recognition research work by Qiuqiang *et al.* [5] presents a pretrained neural network (PANNs) model, the audio recordings are general sounds from human speech and animal sounds and may not be directly applicable to bird songs. BirdNet, on the other hand, has compatibility issue with the existing training data since it uses bird song audio data. Further work is needed to either get access to the original audio data, or time to expand the existing code to implement greyscale spectrograms to test against BirdNet git reposition.

Other potential areas of development for the next step include: 1) exploring other loss functions and optimizers in an attempt to improve the performance are possible options. 2) reducing background noise in the training data. 3) obtaining more data, either through real-world fieldwork or through further experimentation with different methods of data augmentation. We would also like to develop an application that applies our model to predict bird species of audio recordings to test how our model generalizes to real-world examples.

Contributions

Utuq wrote the manuscript for project proposal, milestone as well as final report and polished the final versions together with teammates. He did literature research on BirdNet and other CNN applications on audio classification to help the writing and explore other CNN models to compare to our own. Participated the design and improvement of the CNN model through multiple discussions and initial testing with other teammates.

Patrick brings prior experience working with CNNs and deep learning projects in general. He provided key insight on error analysis and result interpretation, as well as choosing areas of exploration. He also contributed code for early CNN model architectures, model performance evaluation and graphical representation, data augmentation, and other aspects of the codebase. In addition, Patrick worked on reviewing the content and organization of the writing.

John took on the role of project lead. John worked on putting the team together, and reached out to various members in the Stanford community for project ideas including setting up the meetings. He contributed to setting up the standards and schedule for the team, and brought along his software engineering and product management experience for how to approach the problem and how to build the software. He designed and wrote the software with his team members, and ran experiments to collect logs and plots for the paper. He worked on the homegrown CNN model architecture, experimenting with hyperparameters and more complex models. He also helped divvy up tasks between team members and promoted iterative review of the work and writing.

References

[1] Kenneth V. Rosenberg, Adriaan M. Dokter1, Peter J. Blancher, John R. Sauer, Adam C. Smith, Paul A. Smith, Jessica C. Stanton, Laura Helft1, Michael Parr, Peter P. Marra, Decline of the North American avifauna, Science, Vol. 366, Issue 6461, pp. 120-124

[2] Johnston, A., Ausden, M., Dodd, A. M., Bradbury, R. B., Chamberlain, D. E., Jiguet, F., Thomas, C. D., Cook, A. S., Newson, S. E., Ockendon, N. et al. (2013) Observed and predicted effects of climate change on species abundance in protected areas. Nature Climate Change, 3, 1055–1061.

[3] Keiron O'Shea and Ryan Nash (2015) An Introduction to Convolutional Neural Networks.

[4] Schalkoff, R.J. (2007). Pattern Recognition. In Wiley Encyclopedia of Computer Science and Engineering, B.W. Wah (Ed.).

[5] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang and M. D. Plumbley, "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pp. 2880-2894, 2020

[6] Alexis Joly, Valentin Leveau, Julien Champ, Olivier Buisson (2015) Shared nearest neighbors match kernel for bird songs identification -LifeCLEF 2015 challenge. *CLEF: Conference and Labs of the Evaluation Forum*

[7] Lasseck, M. (2015) Improved automatic bird identification through decision tree based feature selection and bagging. In: Working notes of CLEF 2015 conference.

[8] Mehak Arif, Richard Hedley, Erin Bayne (2020) Testing the Accuracy of a birdNET, Automatic bird song Classifier. doi.org/10.7939/r3-6khb-kz18

[9] Dina B. Efremova, Mangalam Sankupellay, Dmitry A. Konovalov (2019) Data-Efficient Classification of Birdcall Through Convolutional Neural Networks Transfer Learning. arXiv:1909.07526

Project Code

Contact johnngoi@stanford.edu to get access to the code `https://github.com/johnnst/cs230-project`.