CAI 4104/6108: Machine Learning Engineering

Project Report: Sound Classification

Nooshin Yousefzadeh
(Point of Contact)
nooshinyousefzad@ufl.edu

Hoang Ngo
hoang.ngo@ufl.edu

Tre' R. Jeter
t.jeter@ufl.edu

August 18, 2024

1 Introduction

Sound classification in machine learning involves automatically categorizing audio samples into different predefined classes based on their content or characteristics. Over the years, various techniques such as Fourier Transform, Short-time Fourier Transform (STFT), and spectrograms have been pivotal in analyzing the frequency content of audio signals. Spectrograms, for instance, provide a visual representation of a signal's amplitude variation over time at different frequencies, while Melspectrograms offer a conversion of spectrograms to the mel scale, capturing both temporal and frequency content.

These techniques serve as essential tools for training models in sound classification tasks, particularly when analyzing frequency content and temporal dynamics. Chromagrams, on the other hand, provide insights into the harmonic and tonal characteristics of audio signals, making them valuable for tasks involving tonal and harmonic content analysis.

In the realm of machine learning, sound classification using Melspectrograms finds applications in speech recognition, music genre classification, environmental sound monitoring, and more. However, one area that particularly stands out is the classification of bird sounds.

Birds produce a diverse array of songs and calls, each serving specific purposes such as mate attraction, warning signals, and territorial demarcation. Identifying bird species from audio clips of their calls is a challenging yet essential task. In this project, we focus on the problem of bird sound classification, aiming to accurately identify bird species from audio clip samples of their calls using Melspectrograms.

To benchmark our approach, we compare several baseline models, including Naive Bayes, Naive Bayes Nearest Neighbors (NBNN), Naive Bayes SVM (NBSVM), and a fully connected (FC) model. Additionally, we investigate the effectiveness of a proposed Convolutional Neural Network (CNN), which leverages the power of deep learning to capture intricate patterns in bird sounds for classification. Through this comparative analysis, we aim to expose the strengths and weaknesses of each approach, ultimately striving for more accurate and efficient bird sound classification methodologies.

2 Approach: Dataset(s) & Pipeline(s)

In this section, first, we define the learning task. Then, we describe the data processing. Finally, we propose a CNN-based model to handle our classification task.

2.1 Learning task

Bird species play an important role in the ecosystem. By monitoring the appearance and movement of different bird species, we can make prediction about climate change, natural disasters or habitat modification. However, in the wild environment, it is more difficult to actually see birds than hearing their sounds. Thus, by properly identifying bird songs, we can monitor the behaviour and movement of different bird species in the wild environment.

In this project, we aim to classify bird songs corresponding to different bird species. Specifically, the inputs are bird song snippets collected from the British Birdsong dataset. The expected outputs are bird species corresponding to the given bird songs. In the upcoming sections, we will discuss our extraction process where we identify important features from the raw inputs and further discuss our proposed model to classify bird songs.

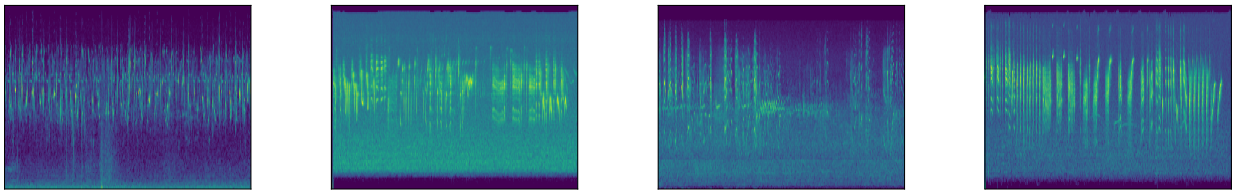

Figure 1: Visualizing sample melspectrograms

2.2 Data pre-processing

Here, we first describe the raw input used in our project. We use the British Birdsong dataset on Kaggle, originally compiled and shared by Dan Stowell [1]. This dataset comprises 264 audio files of bird songs from 85 bird species native to Britain, albeit sourced from diverse locations. Because the lengths of audio files are different, we split audio files into snippets with equal length. In total, we obtain 18386 snippets. Figure 3 illustrates the distribution of those snippets. We can observe that the number of snippets for each class is quite imbalance.

Next, we interpret snippets into sequences of melspectrograms which is a visual representation of the spectrum of a sound signal as it varies over time. The generation of melspectrograms is followed by some steps of preprocessing including merging data from multiple sources, data cleaning, removing missing values, normalization, feature extraction and outlier removal. Figure 1 is a visualization of a few sample melspectrograms that will be used as input representations to our image classifier. From raw melspectrograms, we continue to extract more meaningful features for our task including chromagrams and spectral centroids. For each snippet, we extract its spectral centroid with 13 continuous features and chromagram with 156 continuous features. As a result, each snippet is presented as a datapoint with 169 continuous features. In total, we obtain 18386 data points which are used to train our proposed model.

2.3 Proposed method

Here, first, we discuss how the CNN model (i.e., the convolutional layer) is beneficial for our task. Then, we present the architecture and setting of our proposed CNN-based model.

As mentioned in the previous part, for each snippet in an audio, we extract the normalized energy for each chroma bin at each frame. We observe that features in neighboring chroma bins and frames can have internal relationships. Thus, convolutional layers are good choices for capturing those relationships. In addition, CNN can be useful in mitigating the effect of noise from audio files to learnt features.

```
CNN(
  (model): Sequential(
    (0): Conv2d(1, 16, kernel_size=(1, 1), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(16, 32, kernel_size=(1, 1), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Flatten(start_dim=1, end_dim=-1)
    (10): Linear(in_features=5776, out_features=512, bias=True)
    (11): ReLU()
    (12): Dropout(p=0.2, inplace=False)
    (13): Linear(in_features=512, out_features=85, bias=True)
  )
)
```

Figure 2: The proposed CNN-based model's architecture

The detail of our proposed CNN-based architecture is summarized in Figure 2. In general, we have three convolutional layers with 16, 32 and 16 channels respectively. Each convolutional layer is followed by a batch-norm layer for normalizing the output features and a ReLU layer for performing the non-linearity transform and mitigating the gradient vanishing problem. After the series of convolutional layers, we flatten the resulting feature vector with 5776 dimensions and put it into a fully connected layer with 512 hidden dimensions. In the end, we forward the hidden feature vector into another fully connected layer to get the final feature vector with 85 dimensions corresponding to 85 different bird classes. This feature vector will be interpreted into the classification probability for bird classes using a softmax layer.

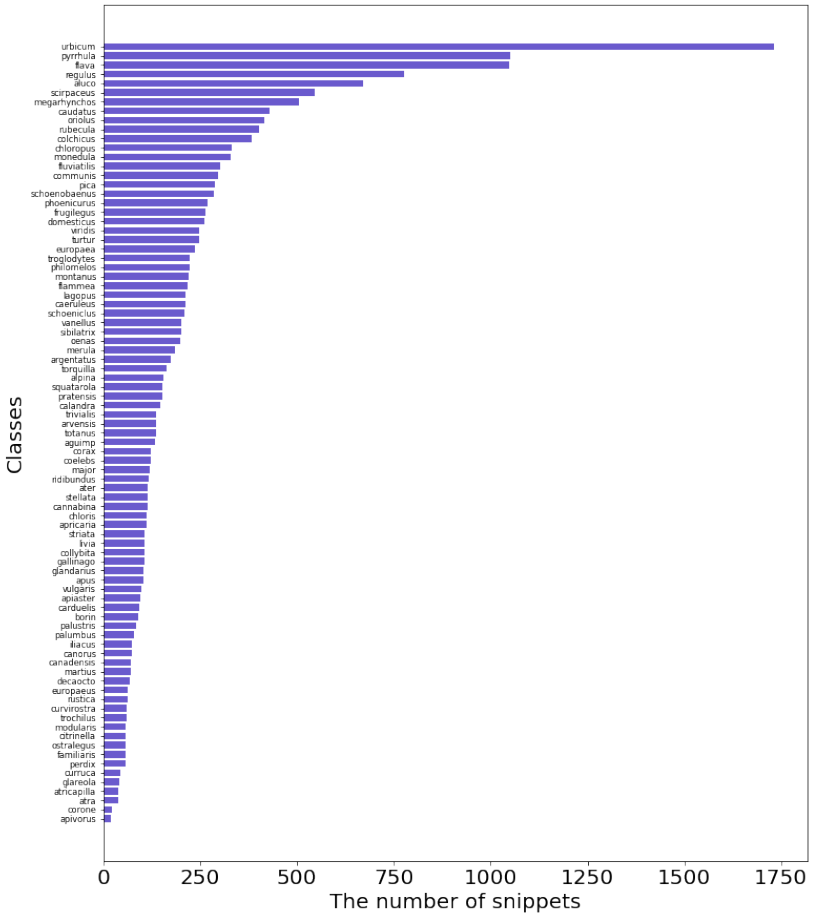

Figure 3: The distribution of datapoints corresponding to 85 classes

Furthermore, in order to avoid over-fitting, we include a dropout layer in our model. For each training round, a random number of neurons in our model (i.e., 0.2%) is deactivated. As a result, our trained model can be more robust because each neuron is forced to learn more important features on its own rather than relying on other neurons' information.

3 Evaluation Methodology

3.1 Baselines

We compare our proposed CNN to a simple fully connected (FC) model. The Beyond this, we also compare against a Naive Bayes classifier along with two variants: Naive Bayes Nearest Neighbors (NBNN) and Naive Bayes Support Vector Machine (NBSVM). Naive Bayes as a stand-alone model has proven to be more than capable of classifying text-based data. Combined with k-Nearest Neighbors and SVMs, Naive Bayes has also been proven to classify music with respectable accuracy [2]. Thus, we aimed to apply this same methodology to bird songs to observe the classification abilities compared to a more powerful CNN-based model.

3.2 Metrics

For a fair evaluation of our proposed CNN model and baselines, we report their classification accuracy and F1-scores. Accuracy measures a model's ability to correctly classify instances across all classes. It quantifies the proportion of correct predictions made by a model compared to all predictions. In terms of classification, a high accuracy score

usually signifies that the model effectively captures the underlying patterns in the data, resulting in a significant portion of correct predictions. On the other hand, a low accuracy score could suggest that the model misclassifies a large number of instances, indicating potential deficiencies in its learning process or data representation. Accuracy can be seen in Equation 1, where TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

F1-score combines precision and recall to give a holistic assessment of a classification model's efficacy. By calculating the harmonic mean of precision and recall, the F1-score offers balanced insight into both aspects of the model's performance. A high F1-score signifies high precision and high recall. Meaning, a high F1-score reflects the model's capability to achieve both accurate predictions and comprehensive coverage of relevant instances. On the other hand, a low F1-score suggests potentially low precision or recall. F1-score can be seen in Equation 2. Precision and recall can be seen in Equations 3 and 4.

$$F1 - score = \frac{2 * (precision * recall)}{precision + recall} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

3.3 Data splitting

In total, we have 18386 data points. We use 66% for training, 17% for validation and 17% for testing.

4 Results

In this section, we first illustrate the training performance of two deep learning models including our proposed CNN and the FC model (baseline 1). Then, we compare the inference performance of our proposed CNN to the FC model (baseline 1) and the Naive Bayes classifier (baseline 2) with three different variants: Naive Bayes, NBNN and NBSVM.

4.1 Training performance

First, we describe the training settings for the CNN and the FC. Both models use cross entropy as the loss function and use the Adam optimizer for optimizing the loss. After optimizing the hyperparameters by grid search, for the CNN, we set the learning rate as $3e-5$, and the weight decay as $1e-3$. On the other hand, for the FC, we set the learning rate as $3e-4$ and the weight decay as $1e-4$. The number of training epochs is 240.

Figures 4 and 5 show the learning curve of the CNN and the FC model respectively. In both cases, the validation accuracy is not below the training accuracy, indicating that our trained models does not suffer from overfitting. In addition, we observe that for the FC model, the validation accuracy is even higher than the train accuracy. From our opinions, there are two reasons. Firstly, we employ dropout only during training. Thus, during the validation phase, the model makes predictions with the full set of neurons, resulting in higher accuracy. Secondly, the validation dataset is relatively small. As a result, the model can generalize well to the validation set even before overfitting to the training set occurs.

4.2 Inference performance

Table 1 shows that our proposed CNN and baselines are all able to classify bird songs with very high confidence. This is evidenced by each model reporting above 90% testing accuracy and F1-scores, respectively. However, we do see an interesting occurrence where the simpler FC model outperforms our proposed CNN model. The FC model considers the aggregation of all features whereas the CNN model considers the aggregation of neighboring features. Knowing this, the CNN may lose important information about features which could explain its lower performance compared to the FC model. Otherwise, the CNN outperforms the rest of our baselines with respect to accuracy and is comparable with respect to F1-score.

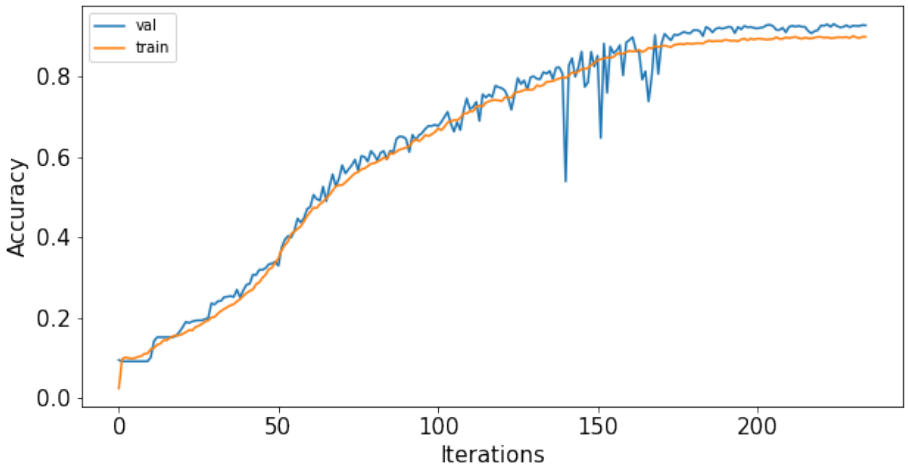

Figure 4: The learning curve of our proposed CNN

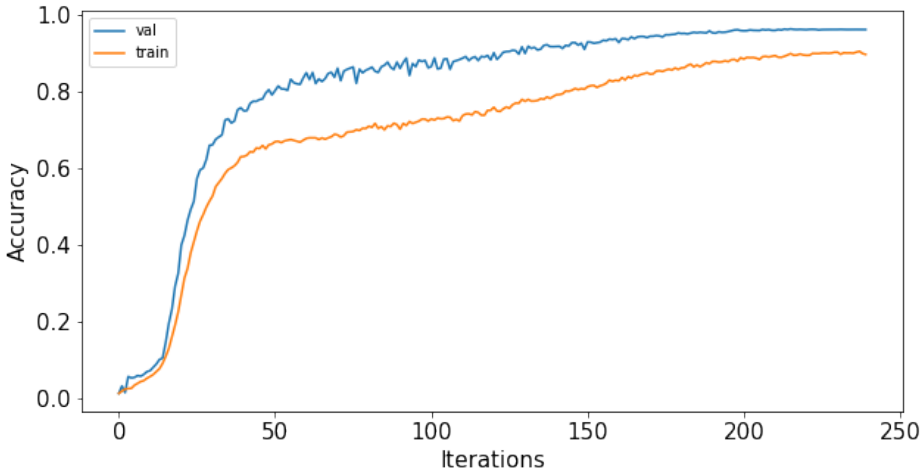

Figure 5: The learning curve of the FC model

	Accuracy	F1-Score
Naive Bayes	92.99	93.82
NBNN	93.76	94.57
NBSVM	92.71	93.77
FC Model	**96.27**	**96.06**
CNN (ours)	94.00	93.19

Table 1: Accuracy and F1-Score comparison of our proposed CNN model and baselines.

5 Conclusions

In this work, we study the bird song classification problem. Specifically, we propose a CNN-based model and implement two baselines including the fully connected model and the Naive Bayes model. From the experimental results, we can conclude that 1) the dropout is efficient in avoiding overfitting, 2) deep learning models (CNN and FC) outperforms traditional machine learning model (Naive Bayes) and 3) the FC performs better than the CNN in this task.

From our perspective, it is reasonable for the FC model to outperform the CNN model. This is because the number of input features in our task is not large, allowing the FC model to effectively extract all relationships among features. However, as the number of features increases, the CNN may become more advantageous because it can focus on important relationships while the FC model may struggle with the computational burden of considering all relationships. Our future plan is to increase the number of features extracted from snippets by raising the

sampling rate. Then, we will re-train both models and observe their performance to validate our hypothesis about the CNN's benefit.

References

[1] Xeno Canto Dan Stowell. xeno-canto. https://archive.org/details/xccoverbl_2014, 2014.

[2] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. Learning naive bayes classifiers for music classification and retrieval. In *2010 20th international conference on pattern recognition*, pages 4589–4592. IEEE, 2010.