Motif Mining and Unsupervised Representation Learning for BirdCLEF 2022

Anthony **Miyaguchi**[1], Jiangyue **Yu**[1], Bryan **Cheungvivatpant**[1], Dakota **Dudley**[1] and Aniketh **Swain**[1]

[1]Georgia Institute of Technology, North Ave NW, Atlanta, GA 30332

Abstract

We build a classification model for the BirdCLEF 2022 challenge using unsupervised methods. We implement an unsupervised representation of the training dataset using a triplet loss on spectrogram representation of audio motifs. Our best model performs with a score of 0.48 on the public leaderboard.

Keywords

motif mining, matrix profile, unsupervised representation learning, embedding, CEUR-WS, BirdCLEF 2022

1. Introduction

The BirdCLEF 2022 challenge [1] [2] involves identifying species of native Hawaiian birds from soundscape recordings. The training dataset comprises 14.8 thousand recordings totaling over 190 hours from Xeno-canto [3] of varying length and quality. The task involves predicting the presence of 29 bird species in non-overlapping 5-second windows of 1-minute soundscape recordings. The recordings are Ogg Vorbis audio files encoded at a 32khz sample rate. The Xeno-canto training examples are labeled by species but not at 5-second intervals per the challenge's task. Therefore, they may contain both ambient noise and birdcalls from other species.

We focus our efforts on unsupervised methods to address the lack of concrete labels for the multi-label classification problem. First, we experiment with motif mining algorithms to identify windows of audio that contain birdcalls as an unsupervised process for generating labels. The motif mining process also provides metadata used to train downstream models. Additionally, instead of training a classification model directly from training examples, we choose to build an embedding that captures similarities between birdcalls across all training examples. We finally train classification models using the embedding model to reduce the dimensionality of the original data.

2. Motif Mining

We utilize a motif mining algorithm called SiMPle [4] to extract the segments of the audio clip that best represent common patterns in the clip. SiMPle provides two primary operations to compute summary data structures called the matrix profile and profile index: a self-join that computes the similarity within a track and a join that computes the similarity between two tracks. The procedure involves converting the raw audio track into a spectrogram to capture the frequency components of the audio over a sliding window in time. Then we apply SiMPle to compute the matrix profile and profile index, providing the distance to the nearest neighbor and the index of the nearest neighbor in the set of windows in the track. A motif is a window of an audio track with the lowest distance to all other windows in the join operation as given by the matrix profile. The indices of the matrix profile's minimum and maximum distance values are the motifs and discord, respectively.
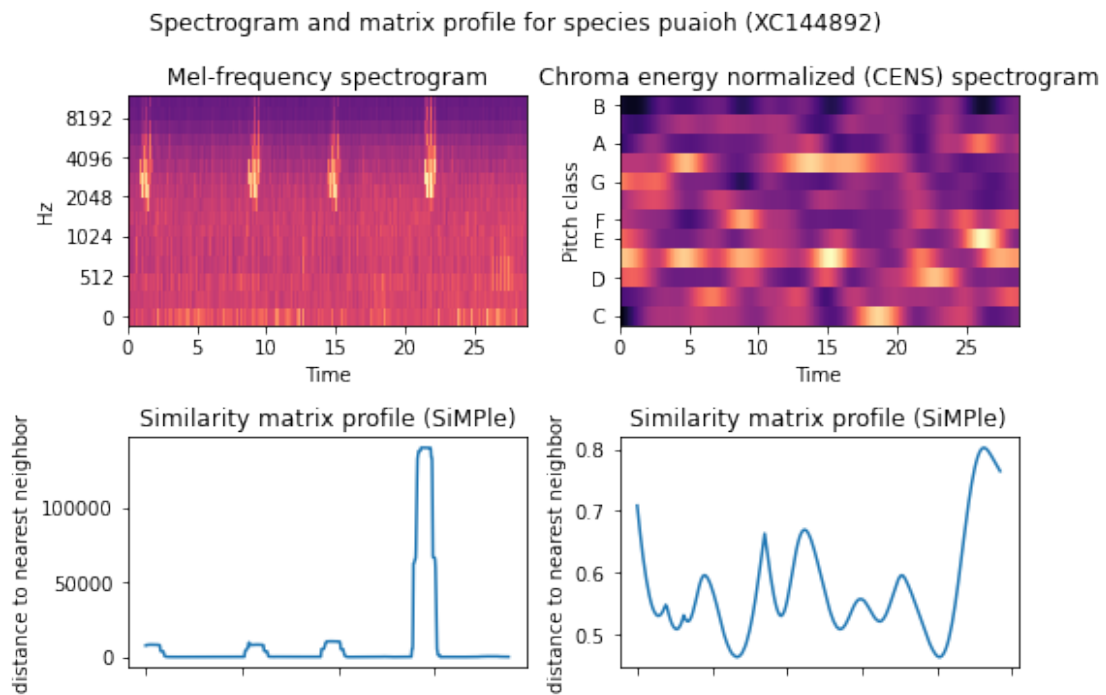

Figure 1: Spectrograms show frequency components of audio transformed via STFT. We apply SiMPLe to obtain a matrix profile that summarizes the distance to the nearest neighbor for all time-slices in the spectrogram. Spectrogram parameterization affects the quality of the matrix profile summary. The Mel-frequency spectogram renders log-scaled filter banks to approximate human hearing response, while the CENS spectrogram renders filter banks that map to a chromatic scale.

We hypothesize that the motifs and discords capture the audio clip's salient features, i.e., bird calls. Much like how SiMPle can identify the chorus of a song via the motif, it may be possible to extract birdcalls as a motif from the training examples. We transform the training dataset by extracting the matrix profile and profile index information from each track via a

self-join and using the 5-second motifs as a soft label for birdcalls in each species. We utilize the self-join index profile during the training procedure of the birdcall embedding. We also test the feasibility of using a matrix profile join as a feature in the classifier for the main BirdCLEF task by directly utilizing the resulting join against a random set of motifs extracted from the training dataset.

3. Birdcall Embedding

An embedding maps data from one space into a lower-dimensional space while maintaining relative distances between mapped data. We experiment with creating a birdcall embedding using a modified Tile2Vec model [5]. The Tile2Vec model utilizes a sampling procedure on spatially distributed data and a triplet loss to learn a lower-dimensional representation of Earth imagery tiles that retains semantic similarity between tiles. The triplet loss takes advantage of the triangle inequality by forming triplets between anchor, neighbor, and distant tiles. We borrow the loss for triplets (t_a, t_n, t_d) with a margin m where f_θ maps audio data to a d-dimensional vector of real numbers using a model with parameters θ.

$$L(t_a, t_n, t_d) = [||f_\theta(t_a) - f_\theta(t_n)||_2 - ||f_\theta(t_a) - f_\theta(t_d)||_2 + m]_+ \tag{1}$$

We hypothesize that we can learn an embedding by forming triplets using windows of audio with similar spectral qualities. We utilize the SiMPLe index of the audio spectrogram to determine the neighborhood of any given window of audio within a track. The Tile2Vec model is a modified ResNet-18 convolutional neural network – we make further modifications to generate spectrograms in the correct shape as the first layer in the model.

4. Experiments

4.1. Motif Mining Details

We begin by extracting the motifs and discords of the entire training dataset. However, this process is computationally intensive because it requires streaming the entire audio dataset, generating the spectrogram via STFT, and computing the matrix profile. Therefore, we pre-compute the matrix profile and index profile for offline processing. We use librosa [6] for audio loading and transformation and implement a NumPy implementation of the SiMPLe algorithm to facilitate usage in our processing pipeline. [1]

We use chroma energy normalized (CEN) and Mel-scaled spectrograms. We generate CEN spectrograms at ten samples per second using the default parameters in librosa, with a 50 sample window for motif mining purposes and the Mel-scaled spectrograms using an FFT window of 2048, a hop length of 80 samples, and 16 Mel bands with a 400 sample window.

As per figure 1, we note that the observed quality of a spectrogram and the resulting matrix profile varies depending on the algorithm and parameters used. We can clearly distinguish the chirps in the high-frequency ranges in the Mel spectrogram, with obvious discords occurring

---

[1]Implementation at github.com/acmiyaguchi/simple-fast-python

spectrogram format	transform time	shape	size	SiMPle window	self-join
Mel-scaled	131ms	(16, 7932)	126912	400	3.3s
CEN	287ms	(12, 292)	3504	50	8.85ms

Table 1
Statistics from processing track XC144892 which is 28.8 seconds long. Each operation is run 10 times. We note that our parameterization of the Mel-scaled spectrogram results in data that takes several orders of magnitude longer to process than the CEN spectrograms.

in the same time intervals. We are presented with a lower resolution spectrogram with CEN because it is aligned by chroma instead, which does not delineate between the bird call and the background noise as clearly.

We initially performed our motif mining using the CEN parameters, following closely with the SiMPLe cover song extraction experiment. We found that the lower resolution of the CEN spectrogram representation allowed us to process all training examples and store the primary representation on disk, as per table 1. Although noisy, the representation did extract bird calls as the primary motif, although lack of domain knowledge prevented us from quantifying the quality of these labels. As a result, some extracted motifs are background noise or human voices.

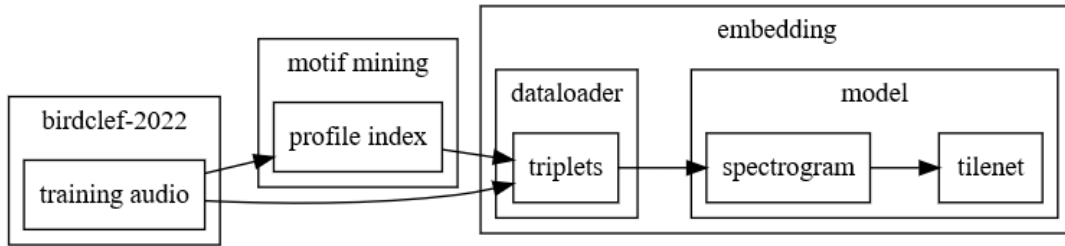## 4.2. Embedding Clustering Quality

Figure 2: Flow diagram of the constituent pieces of the birdcall embedding.

We reuse the open-sourced Tile2Vec model and add a Mel spectrogram layer via nnAudio [7]. We use an FFT window of 4096 with a calculated hop length matching the height of 128 Mel bands. We choose these parameters to mimic the 128x128 pixel images used to train the source model.

We built two separate data loaders for this task — the first data loader pre-computed triplets from the original Ogg Vorbis files into NumPy array files. We only use the extracted motifs during the data loading process. In addition, we attempt to adjust for the skewed species distribution by oversampling underrepresented species. A second data loader streamed all audio computing triplets using the pre-computed SiMPLe indices for each track. We break each track into windows of 5-seconds and assign the nearest neighbor using the profile index. Next, we create several queues, each containing window pairs from tracks of different species of birds.

The data loader pops pairs from each queue to form a mini-batch, after which triplets are formed by randomly assigning the third element from an element within the mini-batch. Finally, we augment the audio tracks once formed into triplets.

Algorithm 1 Sampling triplets from audio using precomputed SiMPle index

---

Require: D is the set of audio tracks as raw samples
Require: f_s is the sampling rate of the audio in Hz
Require: s is the size of the audio window in seconds
Require: I is the SiMPle index
Ensure: T is the set of triplets formed by all windows
Ensure: $|T| = |P|$
  Initialize pairs $P = \{\}$
  for $y \in D$ **do**
    $w \leftarrow \text{window}(y, f_s, s)$ $\triangleright\, w$ is an array of audio windows each of length $s \times f_s$
    for $i \leftarrow 1, \text{length}(w)$ **do**
      $P \leftarrow P \cup (y, w[i], w[I[i]])$ $\triangleright$ Use the SiMPle index to get the nearest neighbor
    end for
  end for
  $T \leftarrow \{(x_a, x_n, y_a) \,|\, \forall (x, x_a, x_n) \in P, \exists ((y, y_a, y_n) \in P,\, x \neq y\}$

The first approach is untenable for an online training procedure because it naively generated motif triplets using upwards of $3N$ disk reads. In addition, we noticed that running the data loading procedure online would cause underutilization between the CPU and GPU due to the loading bottleneck. By storing $5e5$ NumPy array triplets to disk, we could fully utilize the GPU at the cost of disk space, going from the original 6GB of audio data to 135GB for the pre-computed triplet data. The second approach required no additional memory because it took on an iterable approach to creating triplets. It has a different distributional semantic from the first since it includes all audio windows instead of just the track motifs. We also do not address class imbalances because online undersampling is difficult. We need further preprocessing to determine the total number and locations of birdcalls per class.

We apply a random gain of $[-20, 20]$ dB, Gaussian random noise between $[-5, 40]$ dB, and pitch shift between $[-4, 4]$ semitones using the audiomentations library [8]. We note that augmentation can become a bottleneck in the data loading process. Instead of applying it to the entire triplet, it suffices to apply it to just the motif pairs before triplet formation. We consider torch-audiomentations but had poor performance due to difficulties applying the correct device to torch tensors inside the data loader.

We validate the embedding learned on the iterable data loader by training a classifier on motifs transformed into the embedding space. We chose a subset of three species: `brnowl`, `skylar`, and `houfin`. These are common species with several hundred audio tracks available. We generate a training dataset comprised of the motif of each track and cross-validate a logistic regression model using model accuracy on species accuracy. We sample $k = 300$ motifs for ten models and report the mean and standard deviation of the accuracy on embedding models that vary in the output size.

We note a slight difference between accuracy in models of equivalent parameterization when
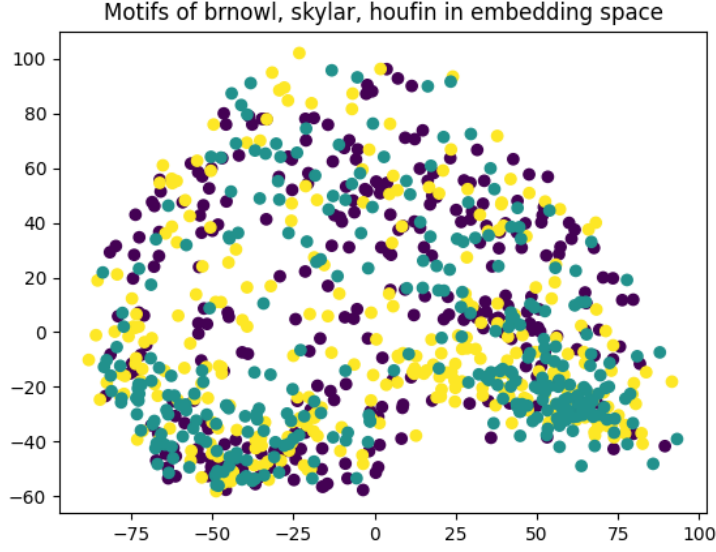
Figure 3: A scatter plot of a random set of motifs (n=300) drawn from three species of birds. The motifs are truncated, padded, and transformed into the embedding space. We plot the top two principle components found by PCA.

the output dimension of the model increases from 256 to 512 from 0.53 to 0.54.

dimension	accuracy (n=10)
128	0.528 (0.031)
256	0.529 (0.019)
512	0.544 (0.048)

Table 2
Accuracy table of a logistic regression model trained on the embedding using motif mined from three species of birds.

4.3. Classifier Performance

We train a classifier using the embedding model as a preprocessing step. The first classifier trains a gradient-boosted decision tree (GBDT) via LightGBM [9]. It uses a MultiOutputClassifer to create a model per task to suit the multi-label classification task using the default parameters of the LightGBM classifier. Next, we apply a similar set of augmentation to the embedding model, with a random gain of $[-20, 20]$ dB, pitch shift between $[-4, 4]$ semitones, time shift between $[-10, 10]\%$ of the window's length, and colored noise between $[-3, 30]$ dB and $[-2, 2]$ frequency power decay using the PyTorch-audiomentations library [10]. We also train a multi-layer perceptron (MLP) to perform multi-label classification directly using a binary cross-entropy loss. The loss function allows us to implement mixup [11] during data loading mini-batches as

an augmentation process.

We experimented with using matrix profile join data as a feature in our classification model. We draw 64 motif samples from a set of training motifs and run each audio window in the testing task through a matrix profile against the training sample. We use the concatenation of the matrix profile join's min, median, and max as an additional feature in the model. Unfortunately, we found that the additional CPU overhead to run SiMPle far outweighed any marginal performance benefit to the model and was omitted in subsequent model iterations.

classifier	macro F1-score
GBDT	0.0177
MLP	0.0151

Table 3
A comparison of the two classifiers' performance on a subset of the training audio data, using macro F1-score as metric. We limit the scored data to the first ten tracks of each species in the task.

We compute the macro F1-score in table 3 to compare our best GBDT model against our best MLP model. We note that the GBDT edges slightly over the MLP, but performance for both classifiers is poor. We often see a lack of positive predictions for a class, which leads to a score of 0. We also assume that all audio windows contain birdcalls to simplify the calculations, so these scores understate the actual performance of the models. However, we found that the general performance is still poor in the BirdCLEF task, hovering around 0.48 on the public leaderboard during the competition.

5. Discussion

In a small set of shorter tracks (30 seconds or less), we consistently found that the discord tends to be a birdcall instead of the motif. It might be more productive to standardize a mining procedure on shorter tracks instead of using the whole, variable-length tracks to take advantage of this observation. We can build a no-call detector using the matrix profile as a feature if we can consistently detect positions of bird calls through discords or motifs. In addition, we can achieve higher throughput on parallelization by reducing data skew in the distribution of training example lengths. Finally, both spectrogram transformations via nnAudio and the SiMPle algorithm can be written in PyTorch to increase the performance of the motif mining algorithm.

We found that the birdcall motif triplet embedding performed poorly for downstream prediction tasks. We hypothesize that the spectrogram parameters during motif mining cause the embedding to rest on a representation ill-suited for classification. One modification to the embedding triplet procedure would be to form pairs of anchors and neighbors from overlapping audio windows. These pairs would be similar due to their distance in time. As the experimental procedure before, triplets are formed by randomization in mini-batches. A sliding window triplet procedure would enable a comparison of the validity of the learned motif triplet embedding.

6. Conclusions

We implemented and experimented with the SiMPle motif mining algorithm, a modified Tile2Vec embedding model, and several multi-label classification models.[2] Our performance on the leaderboard was underwhelming, but we solved many engineering problems throughout the challenge with potential improvements for future BirdCLEF challenges.

Acknowledgments

References

[1] S. Kahl, A. Navine, T. Denton, H. Klinck, P. Hart, H. Glotin, H. Goëau, W.-P. Vellinga, R. Planqué, A. Joly, Overview of birdclef 2022: Endangered bird species recognition in soundscape recordings, Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum (2022).

[2] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, I. Bolon, H. Glotin, R. Planqué, W.-P. Vellinga, A. Navine, H. Klinck, T. Denton, I. Eggel, P. Bonnet, M. Šulc, H. Müller, Overview of lifeclef 2022: an evaluation of machine-learning based species identification and species distribution prediction, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2022.

[3] Xeno-canto, Xeno-canto: Sharing bird sounds from around the world, 2022. URL: https://xeno-canto.org.

[4] D. F. Silva, C.-C. M. Yeh, Y. Zhu, G. E. Batista, E. Keogh, Fast similarity matrix profile for music analysis and exploration, IEEE Transactions on Multimedia 21 (2018) 29–38.

[5] N. Jean, S. Wang, A. Samar, G. Azzari, D. Lobell, S. Ermon, Tile2vec: Unsupervised representation learning for spatially distributed data, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 3967–3974.

[6] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, O. Nieto, librosa: Audio and music signal analysis in python, in: Proceedings of the 14th python in science conference, volume 8, 2015.

[7] K. W. Cheuk, H. Anderson, K. Agres, D. Herremans, nnaudio: An on-the-fly gpu audio to spectrogram conversion toolbox using 1d convolutional neural networks, IEEE Access 8 (2020) 161981–162003. doi:10.1109/ACCESS.2020.3019084.

[8] I. Jordal, A. Tamazian, E. T. Chourdakis, C. Angonin, askskro, N. Karpov, O. Sarioglu, kvilouras, E. B. Çoban, F. Mirus, J.-Y. Lee, K. Choi, MarvinLvn, SolomidHero, T. Alumäe, iver56/audiomentations: v0.25.0, 2022. URL: https://doi.org/10.5281/zenodo.6594177. doi:10.5281/zenodo.6594177.

---

[2]Implementation at github.com/acmiyaguchi/birdclef-2022

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems 30 (2017) 3146–3154.

[10] I. Jordal, K. Nishi, H. BREDIN, F. Lata, H. C. Blum, P. Manuel, akash raj, K. Choi, P. Żelasko, amiasato, M. L. Quatra, E. Schmidbauer, FrenchKrab, asteroid-team/torch-audiomentations: v0.10.1, 2022. URL: https://doi.org/10.5281/zenodo.6381721. doi:10.5281/zenodo.6381721.

[11] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, arXiv preprint arXiv:1710.09412 (2017).