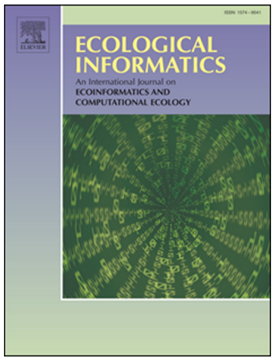


On the role of audio frontends in bird species recognition

Houtan Ghaffari, Paul Devos

PII: S1574-9541(24)00115-8

DOI: https://doi.org/10.1016/j.ecoinf.2024.102573

Reference: ECOINF 102573

To appear in: *Ecological Informatics*

Received date: 4 October 2023

Revised date: 18 March 2024

Accepted date: 22 March 2024

On the Role of Audio Frontends in Bird Species Recognition

Houtan Ghaffari *[1] and Paul Devos[1]

[1]*Department of Information Technology, WAVES Research Group, Ghent University, Belgium*

March 18, 2024

* Corresponding author: houtan.ghaffari@ugent.be

Abstract

Automatic acoustic monitoring of bird populations and their diversity is in demand for conservation planning. This requirement and recent advances in deep learning have inspired sophisticated species recognizers. However, there are still open challenges in creating reliable monitoring systems of natural habitats. One of many open questions is whether predominantly used audio features like mel-filterbanks are appropriate for such analysis since their design follows human's perception of the sound, making them susceptible to discarding fine details from other animals' vocalization. Although research shows that different audio features work better for particular tasks and datasets, it is hard to attribute all advantages to input features since the experimental setups vary. A general solution is to design a learnable audio frontend to extract task-relevant features from raw waveform since it contains all the information in other audio features. The current paper thoroughly analyzes the role of such frontends in bird species recognition, which helped to evaluate the adequacy of traditional time-frequency representations (static frontends) in capturing the relevant information from bird vocalization. In particular, this work shows that the main performance gain in learnable audio frontends comes from the normalization and compression operations rather than the data-driven frequency selectivity and functional form of filters. We observed no significant discrepancy between the frequency bands of the learned and static frontends for bird vocalization. Although the performance of learnable frontends was much higher, we will show that adequate normalization and compression enhance the accuracy of traditional frontends by more than 16% to achieve comparable results for bird species recognition. Ablation studies of the frontends under different configurations and detailed analysis of noise robustness provide evidence for the conclusions, validate the use of mel-filterbanks and similar features in prior works, and provide guidelines for designing future species recognizers. The code is available at `https://github.com/houtan-ghaffari/bird-frontends`.

***Keywords*—** Bioacoustics; Audio frontend; Bird sound recognition; Deep learning

1 Introduction

The loss of biodiversity will disrupt vital ecosystem processes and services, such as pollination, seed dispersal, and natural pest control, which lead to undesirable ecological and societal consequences [1]. Frankly, it is concerning how much anthropogenic activities, such as changing land use patterns and climate change, have negatively impacted the diversity of life due to habitat loss and have reduced the environmental capacity for sustaining other lifeforms [2], [3]. Consequently, trustworthy and continual surveys of the wild populations' density are crucial for conservation planning and future ecological studies [4]. Fortunately, a large subset of animals are identifiable by their unique sounds. This property led to the development of bioacoustics and ecoacoustics as non-invasive and scalable means to monitor environmental activities via sound [5], [6]. In particular, birds are highly active vocal animals, and their populations and diversity are noticeable indicators of ecosystem health [7]. Thus, they can provide valuable information for conservation planning [8], [9].

Although it is onerous to expand habitat and population monitoring systems by deploying human experts, there are large volumes of bird sound recordings across the globe due to automatic field recorders and public science projects [10]–[12]. Consequently, the main challenge has shifted from data gathering to developing reliable and automated analysis tools since it is infeasible to manually and continually process these massive and noisy datasets [13], [14]. In the following, the terms audio representation and feature are interchangeable when talking about neural networks.

Feature engineering and designing an appropriate classifier used to be separate problems. Bioacoustics researchers have put tremendous effort into creating reliable species classifiers by designing proper acoustic features and combining them with various machine learning models. Although this is far from being representative of the background works, the traditional models include template matching [15]–[17], hidden Markov models [18], [19], support vector machines [20], and random forests [21], each being leveraged with different hand-crafted features across the literature. It made it difficult to compare the models since experimental setups differed in many aspects, including the feature design and datasets. Regardless, the field is moving fast, and many traditional methods are outdated compared to current tools for analyzing massive datasets. They are not flexible enough to address current open challenges unless used purposefully on a narrow task.

Recent years have seen impressive progress in bird species recognition and related tasks [8], [14]. It was foreseeable that deep learning would emerge as the most successful method for large-scale bioacoustics analysis [8], [9], [14], [22]–[24]. The main advantage of neural networks is to avoid many brittle and difficult choices in designing manual features for the underlying task. For example, Xie et al. [25] compared three types of features for bird species classification consisting of hand-crafted acoustic features, including features from Mel Frequency Cepstral Coefficient (MFCC), visual features extracted from the Constant-Q Transform (CQT) time-frequency representation, and training a convolutional network directly on the CQT. They also tested the fusion of classifiers trained on these features and found a slight performance improvement. However, training a neural network directly on the CQT was, by far, the best choice among the features. The other successful contemporary works have explored various types of neural networks as species recognizers by leveraging Short-Time Fourier-Transform (henceforth STFT or spectrogram) [22] and its relatives, such as mel-filterbanks applied on STFT (henceforth mel-spectrogram) [26] and MFCC [24] as input feature.

Nevertheless, there is still no solid guideline on how to develop neural networks for non-human animal sounds. In particular, it is ambiguous what type of audio feature works best as the input for bird's sound modeling since none outperforms others consistently [14]. Also, many incorporated time-frequency representations are designed based on human psychoacoustics instead of the target animals (e.g., mel-filterbanks and MFCC) [27], [28]. This could be problematic since there are substantial variations in the frequency content of different species' vocalization. Thus, these traditional features might miss important fine details in other animals, especially in high frequencies [14], [28]–[30]. Although many of these features give satisfactory results when combined with a deep neural network, one should also focus on the interpretability and robustness of the methods to validate their soundness and improve their generalization across the datasets and species. In this paper, we evaluate the adequacy of the traditional time-frequency representations in capturing the relevant information from bird vocalization.

It seems to require an exhaustive knowledge of each species' frequency niche to create optimal individualized features. Yet, the most celebrated capability of deep neural networks is automatic and task(s)-relevant feature extraction while doing the classification task(s) jointly, which has led to unprecedented achievements [31]. Hence, one could ask whether to consider a time-frequency representation or the waveform as raw input to neural networks to capture all the relevant information from bird vocalization. Although the former is the prevalent approach, data processing inequality indicates that post-processing can not increase a signal's information [32]. Therefore, the waveform is a promising alternative to traditional features derived from it when leveraging neural networks. This union of neural networks and waveform provides a flexible family of models capable of self-adjusting to the characteristics of the target species. In particular, it is an attractive paradigm when targeting many species without adequate prior knowledge about their frequency niches.

Nonetheless, the accuracy of models trained on the waveform is usually lower or comparable to the time-frequency features [33]. Regarding the resources, using waveform is considerably costlier during the training and slower for inference. A successful method requires expertise in signal processing and careful architecture design, usually through specialized modules with physically informed and constrained feature extractors, known today as audio frontends [33]–[38]. An audio frontend can be plugged into any neural network classifier to extract features from waveforms. We do not attempt to survey the audio frontend literature here. Hence, we explain a few relevant works to provide a general insight into the line of progress.

Sainath et al. [34] was an early work to show that waveform can match the log-mel-spectrogram in performance on speech processing, where the authors used a 1d convolution layer as the frontend and initialized its parameters (a.k.a kernels) by Gammatone filters [39]. Similarly, Zeghidour et al. [35] proposed the learnable time-domain filterbanks where the convolution kernels were initialized by Gabor filters [40]. However, they also used learnable pooling and pre-emphasis layers. Some works suggested going beyond informed initialization by constraining the convolution kernels to be physically interpretable. In particular, Ravanelli and Bengio [36] showed that using learnable band-pass sinc filters as the convolution layer kernels benefits speech processing tasks; their frontend is known as SincNet. Please notice that the difference with previously mentioned works was that SincNet kernels had only two learnable parameters to learn the band-pass sinc filters and were not imitating them only at initialization. Later, Noé et al. [37] modified the SincNet using Gabor filters for their optimal trade-off between time and frequency resolution [40]. Recently, Zeghidour et al. [38] proposed a fully-learnable frontend called LEAF. It also leverages Gabor filters and the best practices from prior works. LEAF was not limited to speech processing and showed comparable or better results across diverse audio processing tasks and datasets.

At the time of writing and to our knowledge, the literature on the topic is relatively sparse in bioacoustics. There is a bird species recognition task among the experiments of [33] and [38]. However, they lack sufficient detail for bioacoustics analysis. Bravo Sanchez et al. [30] have tried the original SincNet [36] for bioacoustics with limited investigation. Therefore, we provide an exhaustive analysis of modern audio frontends to examine their efficacy compared to traditional time-frequency features for bird species recognition. The ablation study will demystify some questions around these models and eventually lead to practical findings that others can adapt in their statistical modeling. In particular, a series of detailed experiments will show that,

i. the bulk of the improvement in the audio frontend comes from normalization operations rather than data-driven frequency selectivity,
ii. normalization makes the conventional features comparable to learnable frontends,
iii. compression and normalization together make the models robust against unseen noise,
iv. the learnable frontend prefers log-linearly spaced frequency bands for bird vocalization, although it is reluctant to change if initialized by either mel-filterbanks or linearly spaced STFT,
v. the functional form of the frontend filters only has noticeable significance in the absence of normalization and smoothing modules.

The paper is structured as follows: section 2 will provide information about the datasets, experiments, and background materials for understanding the audio frontends. Section 3 is about conducting the experiments and reporting the results. Section 4 discusses the results further and provides practical guidelines. The conclusion is in section 5.

2 Methodology

2.1 Data Acquisition

The recordings of ten prevalent bird species in four neighboring countries of Western Europe (Belgium, The Netherlands, France, and Germany) were downloaded from the Xeno-Canto public repository [12]. The prevalent species are Cettia Cetti (Cetti’s Warbler), Erithacus Rubecula (European Robin), Fringilla Coelebs (Common Chaffinch), Luscinia Megarhynchos (Common Nightingale), Parus Major (Great Tit), Phylloscopus Collybita (Common Chiffchaff), Sylvia Atricapilla (Eurasian Blackcap), Troglodytes Troglodytes (Eurasian Wren), Turdus Merula (Common Blackbird), and Turdus Philomelos (Song Thrush).

These recordings were filtered using the accompanying metadata to keep the high-quality ones (quality ’A’ and ’B’). This dataset was split into 70-10-20% train-validation-test sets while stratifying by species labels to preserve the distribution of classes in each set. The recordings longer than five minutes were removed from the test set. As explained in the following section, only the first five seconds of the validation set recordings were used. Table 1 shows the number of recordings and their duration. The project repository provides details of the dataset preparation.

Table 1: Number of the recording files and their total duration in hours (h).

Species	train files (duration)	validation files (duration)	test files (duration)
Cettia Cetti	306 (2.51 h)	43 (0.06 h)	87 (0.46 h)
Erithacus Rubecula	1032 (21.69 h)	147 (0.20 h)	249 (5.14 h)
Fringilla Coelebs	861 (15.97 h)	122 (0.17 h)	222 (3.90 h)
Luscinia Megarhynchos	553 (18.98 h)	79 (0.11 h)	144 (3.55 h)
Parus Major	1158 (18.23 h)	165 (0.23 h)	315 (5.08 h)
Phylloscopus Collybita	997 (13.00 h)	142 (0.20 h)	274 (5.26 h)
Sylvia Atricapilla	995 (26.13 h)	142 (0.20 h)	279 (5.83 h)
Troglodytes Troglodytes	677 (13.73 h)	96 (0.13 h)	185 (3.14 h)
Turdus Merula	1193 (47.58 h)	170 (0.23 h)	298 (7.73 h)
Turdus Philomelos	797 (29.74 h)	113 (0.14 h)	154 (4.53 h)
Total	8569 (207.56 h)	1219 (1.67 h)	2207 (44.62 h)

2.2 Cropping Strategy for Long Recordings

Xeno-Canto only offers single-class and weakly labeled recordings. It means the accompanying metadata of each recording indicates the presence of one foreground species vocalization (single-class) without providing any temporal information about when it happened (weakly labeled). The weak labels are problematic since the target vocalization may occupy a small portion of its recording. Also, the vocalization and recording vary in duration drastically per example. The single-class label brings difficulty since the recording may contain potentially overlapping and unannotated geophony, biophony, and anthropophony sound events. These unannotated events may be present or absent from the closed set of the target classes (ten bird species in our case). All of these can reduce the model’s quality. Thus, this work relies on the best practices to use this dataset.

A good evaluation strategy without temporal information is to use the whole recording. It requires windowing the lengthy recordings due to limited machine memory. Hence, each test recording was split into five-second chunks with one second of overlap. Then, the model predicted all five-second excerpts and averaged their outputs to indicate the recording-level decision. Although suitable for the one-time run on the test set, this strategy is computationally too expensive. Thus, two different cropping methods were used for the training and validation recordings to facilitate the speed of experimentation.

Cropping without temporal information will inevitably result in some incorrect input-output pairs. However, the labeled species of the high-quality recordings from Xeno-Canto generally are the dominant sound event of their recordings. The early experiments showed that random cropping outperforms selecting a fixed-position window from the beginning of recordings, probably due to creating a more diverse training set with enough fidelity. Therefore, a two-second excerpt was randomly extracted from each training recording per epoch.

The above randomness is undesired for the evaluation. Thus, the first five seconds of the validation recordings were used to monitor the models during training. It works since the labeled species are usually present during the first few seconds of the recordings from Xeno-Canto. The preliminary experiments confirmed that this strategy significantly reduces the computation time while producing valid results compared to the full-length evaluation used for the test set.

2.3 Data Normalization

A common practice in data analysis is normalizing the input to a suitable and expected numerical range. Input normalization is rarely discussed in adequate detail since it is assumed to have an insignificant impact on the final model. However, this work shows that normalization is the main factor in closing the gap between static and learnable audio frontends. The experiments leveraged two mainstream normalization schemes, including min-max normalization,

$$\tilde{x} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

and standardization,

$$\tilde{x} = \frac{x - \overline{x}}{\overline{s}} \tag{2}$$

where $\overline{x}$ and $\overline{s}$ are the empirical mean and standard deviation, respectively. The normalization was applied per data point (not batch) to the frontend's output before feeding it to the classifier.

2.4 Data Augmentation

Data augmentation refers to any label (semantic) preserving data transformation. Such techniques have proven valuable in enhancing the generalization capability of neural networks as they artificially enrich the dataset size and diversity. Similar to the normalization scheme, it contributes to inconsistencies among the published results. This work uses four popular data augmentation for the audio waveform, including changing audio speed, pitch shift, amplitude scaling (a.k.a gain modulation), and adding Gaussian noise. The ablation study examines the models in the presence and absence of data augmentation. The project repository provides additional implementation details.

Figure 1: The high-level schematic of the audio frontends computation blocks, where the red boxes are learnable and the gray ones are static.

2.5 Audio Frontends

This work considers two learnable audio frontends, LEAF [38] and SincNet [36], and two static (non-learnable) ones, spectrogram and mel-spectrogram. For brevity, (mel-)spectrogram refers to both spectrogram and mel-spectrogram. This section explains the original form of the frontends. However, the ablation study modifies them with learnable and static components for deeper examination. Although many learnable audio frontends have been proposed, we found LEAF and SincNet the most widely adopted ones that do what we require here. Moreover, the literature shows that these two learnable frontends adequately represent what such models are capable of since they are the accumulation of the prior works' successful modifications.

The (mel-)spectrogram combined with log compression results in a static audio frontend. At a high level, a learnable frontend has similar components with additional flexibility to adapt to the dataset's characteristics. The usual design has a convolution layer with specialized kernels to capture distinct frequency contents, an activation function, a pooling to downsample, and a compression layer, as depicted in Figure 1.

LEAF increased the flexibility of the frontend by making the pooling and compression layers learnable. It substitutes the commonly used mean or max pooling by low-pass filtering using

Gaussian kernels with learnable spreads for each frequency band. The downsampling happens by strided convolution during low-pass filtering, where the stride size is equivalent to the hop size in STFT. Furthermore, it leverages learnable PCEN [41] for both compression and normalization.

PCEN [41] is an adaptive method that replaces static functions such as log or root compression. It is formulated as,

$$PCEN(t,f) = \left(\frac{E(t,f)}{(\epsilon + M(t,f))^{\alpha}} + \delta \right)^{r} - \delta^{r} \tag{3}$$

where t and f are time and frequency indices, $E(t,f)$ is the energy at position (t,f), ϵ is a small constant to avoid division by zero, $\alpha \in [0,1]$ controls the gain normalization strength, and δ and r control the dynamic range compression. Furthermore, $M(t,f)$ is an exponential moving average of $E(t,f)$,

$$M(t,f) = (1-s)M(t-1,f) + sE(t,f) \tag{4}$$

where $s \in (0,1)$ is a smoothing coefficient. In the LEAF implementation [38], δ, α, r, and s are learnable parameters for each frequency band.

LEAF's authors [38] showed that learnable Gaussian pooling and PCEN improve the SincNet [36], which they called SincNet+. Moreover, PCEN is useful for pattern recognition in far-field noisy recordings [42], which is a desirable property for many bioacoustics tasks. Hence, to make the comparison fair and follow the best practices to our knowledge, this work uses the SincNet+.

The main difference between the LEAF and SincNet+ is the functional form of their filters in the convolution layer. LEAF uses the Gabor filter [40], which is produced by multiplying a complex sinusoidal wave with a Gaussian window,

$$\phi_{Gabor}(t) = e^{i(2\pi\omega t)} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} \tag{5}$$

where ω is the frequency where the filter yields the greatest response and σ is the spread of the Gaussian window. Therefore, each filter requires only two learnable parameters for the center frequency and spread.

SincNet+ uses the band-pass sinc filter,

$$\phi_{sinc}(t) = 2 f_h sinc(2\pi f_h t) - 2 f_l sinc(2\pi f_l t) \tag{6}$$

where f_l and f_h are learnable parameters determining the lower and higher frequencies of the band-pass filter. The sinc function is defined as,

$$sinc(x) = \frac{sin(x)}{x} \tag{7}$$

Finally, to smooth the discontinuities of the sinc filters at the edges, they are multiplied by a Hamming window of length L, which is defined as,

$$w(t) = 0.54 - 0.46 \cdot cos(\frac{2\pi t}{L}) \tag{8}$$

2.6 Backend Neural Network

The complete model combines an audio frontend with a backend neural network classifier. This work uses an EfficientNet-B0 [43] from the official implementation of TorchVision [44] to keep the experiments consistent with prior works, reproducible, and accessible. Also, the backend leveraged the pre-trained weights from image classification since it improves the performance and stability of optimization.

2.7 Noise Dataset

Eight sound events consisting of wind, thunderstorm, rain, cricket, train, engine, helicopter, and airplane were picked from the dataset for Environmental Sound Classification (ESC-50) [45] as external natural and urban noise recordings. This dataset was only leveraged for adding noise to the test set in analyzing the robustness of frontends to unseen noise. The frequency profile of each noise class is depicted in Figure 2.

Figure 2: Frequency profile of the sound events in the noise dataset.

3 Experiments and Results

The recordings were resampled to 32 ksps (kilo samples per second) for efficient computation since the bird vocalizations are typically below 16 kHz. The spectrogram was generated by an fft size of 512 points (16 ms at 32 ksps) and a hop length of 320 points (10 ms). Dropping the first row of the spectrogram (DC component) resulted in 256 frequency bins with a resolution of 62.5 Hz. The mel-spectrogram uses the same setup with 80 mel-filterbanks. Accordingly, LEAF and

SincNet+ were initialized by 80 filters using the peak frequency and width of the mel-filterbanks [36], [38]. For consistency with the original works, each filter of the LEAF and SincNet+ had 801 points (25 ms). The stride of the convolution in Gaussian pooling was set to 320 points to be consistent with the STFT hop length.

Table 2: Test results (in %) of the frontends. The codes ’A’, ’S’, and ’M’ indicate data augmentation, standardization, and min-max normalization, respectively. Sinc refers to SincNet+.

Model	accuracy	top-3 accuracy	f1-score	precision	recall
Mel-log-A	76.44	89.62	77.06	83.64	76.69
Mel-log	77.84	88.94	77.21	83.65	76.28
Mel-pcen-S	91.80	98.01	91.75	92.39	91.35
Mel-log-M	92.25	96.96	91.77	92.67	91.32
Mel-log-S	92.34	97.69	92.03	92.68	91.68
Mel-log-M-A	92.48	97.37	92.05	93.29	91.32
Mel-log-S-A	92.43	97.55	92.18	93.30	91.50
Mel-pcen	92.52	98.19	92.29	92.53	92.19
Mel-pcen-M	92.57	98.23	92.36	92.96	92.12
Mel-pcen-S-A	92.84	98.32	92.76	93.28	92.43
Mel-pcen-A	92.80	98.19	92.89	93.76	92.32
Mel-pcen-M-A	93.79	98.28	93.62	94.06	93.38
Stft-log-A	76.39	87.27	76.50	82.51	75.71
Stft-log	77.53	89.44	76.64	79.64	76.17
Stft-log-S	89.85	96.01	89.18	89.51	89.43
Stft-pcen	90.30	97.24	90.73	92.01	90.08
Stft-log-M-A	91.93	97.24	91.39	92.61	90.65
Stft-log-M	92.34	97.37	91.82	92.59	91.39
Stft-pcen-M	92.21	97.92	92.13	92.57	91.95
Stft-log-S-A	92.57	97.01	92.28	93.12	91.74
Stft-pcen-S	92.89	97.92	92.87	93.46	92.57
Stft-pcen-A	93.34	98.10	93.25	93.80	92.95

Stft-pcen-M-A	93.48	98.23	93.37	94.23	92.77
Stft-pcen-S-A	93.84	98.46	93.46	94.32	92.88
Sinc-log-A	90.26	97.37	90.38	91.36	89.93
Sinc-log	90.80	97.24	90.70	91.41	90.50
Sinc-log-M	90.85	97.33	90.73	91.71	90.16
Sinc-pcen	91.44	97.69	91.36	91.82	91.22
Sinc-log-S-A	91.93	97.78	91.85	92.69	91.23
Sinc-log-M-A	92.07	97.73	92.07	92.73	91.71
Sinc-pcen-M	92.12	98.28	92.29	93.21	91.69
Sinc-log-S	92.80	98.23	92.51	93.60	91.76
Sinc-pcen-S-A	92.75	98.10	92.51	93.19	92.07
Sinc-pcen-A	93.61	98.64	93.63	93.97	93.40
Sinc-pcen-S	93.79	98.69	93.70	93.85	93.62
Sinc-pcen-M-A	94.47	98.82	94.57	94.81	94.39
Leaf-log-M	92.71	97.64	92.39	93.25	91.83
Leaf-pcen-S	92.84	98.23	92.63	93.45	92.06
Leaf-log	93.11	98.23	92.73	93.48	92.31
Leaf-pcen	93.20	98.01	92.86	93.60	92.37
Leaf-log-S	93.20	98.37	92.93	93.58	92.53
Leaf-pcen-M	93.66	98.60	93.36	93.89	93.04
Leaf-pcen-S-A	93.66	98.32	93.37	94.12	92.89
Leaf-log-A	93.75	98.46	93.66	94.27	93.24
Leaf-pcen-A	94.20	98.32	93.70	94.60	93.17
Leaf-log-S-A	94.11	98.28	93.81	94.07	93.63
Leaf-log-M-A	94.11	98.37	93.99	94.70	93.47
Leaf-pcen-M-A	94.20	98.41	94.11	94.71	93.67

Including the initial general setup, 12 distinct models were trained for each frontend by varying the normalization, compression, and data augmentation schemes. It resulted in a total of 48 models that will be assessed shortly to gain further insight into the machinery of audio frontends.

The models were trained for 100 epochs using Adam optimizer [46] with the default hyper-parameters and a cosine decay scheduler for the learning rate. The loss function was cross-entropy for multi-class single-label classification. The macro-averaged f1-score on the validation set aided in monitoring the model convergence. Macro averaging gives equal importance to all classes regardless of the number of data points in each one.

The experiments were conducted on a server with 64 Intel Xeon Gold 6226R CPUs, 192 GiB RAM, and one NVIDIA RTX A-6000 GPU with 48 GiB RAM. The training time varied based on the experiment, with the lowest for static frontends being around 1.5 hours and for LEAF around 5 hours. SincNet+ is slightly faster than LEAF since the latter has complex valued filters. The project used the following Python packages: PyTorch [47], TorchAudio [48], TorchVision [44], Scikit-learn [49], NumPy [50], Matplotlib [51], Pandas [52], and tqdm [53].

Figure 3: A summary from Table 2 shows that PCEN, normalization, and data augmentation are beneficial overall. For compression, 24 models used PCEN, and 24 used log. There are 16 models for each normalization scheme. Also, 24 models used data augmentation and 24 did not. The box covers the first to third quartile of the data. The whiskers extend up to 1.5 times this interquartile range. Circles are data points beyond the whiskers.

3.1 Main Results

Table 2 shows the test results. The largest performance gap between the learnable and traditional frontends stems from using PCEN instead of log compression. When combined with PCEN, the (mel-)spectrogram is on par with the learnable frontends. Noticeably, a proper normalization on top of the log compression produced similar results to the PCEN version of the models. It is hard to eyeball some of these effects, especially for the data augmentation. Therefore, a performance summary for the 48 models is depicted in Figure 3 to comprehend the importance of each component. Figure 4 shows the confusion matrices of the base and the best configuration for each frontend.

It is of negligible practicality to pick the best model based on very close numerical values and distribute credits. The rankings will slightly change from task to task depending on many factors, especially the dataset, fft size, and hop length. The impressive point here is the proximity of the results. It shows that all frontends were adequate for bird species recognition if given suitable configurations, which followed a logical consistency. The following sections provide a detailed analysis of the frontends with additional experiments to support the conclusions.

Figure 4: Confusion matrices of the base frontends and their best versions in Table 2. The code names are: Cettia Cetti (CC), Erithacus Rubecula (ER), Fringilla Coelebs (FC), Luscinia Megarhynchos (LM), Parus Major (PM), Phylloscopus Collybita (PC), Sylvia Atricapilla (SA), Troglodytes Troglodytes (TT), Turdus Merula (TM), and Turdus Philomelos (TP).

Figure 5: Filters have remained close to their initialization based on mel-filterbanks.

3.2 Ablation studies for Analyzing Audio Frontends

A learnable audio frontend should automatically adjust the frequency response of its filters to match the characteristics of a given dataset. However, Figure 5 shows that SincNet+ and LEAF remained close to their initial state after the training. It explains the proximity of the results in Table 2 since all frontends extracted similar features, as depicted in Figure 6.

The number of filters might be lower than required and the reason for their stickiness (Figure 5). If true, Table 2 shows that the learnable frontends could do better on this dataset and task using more filters. In contrast, it might be that the (mel-)spectrogram is close to what learnable frontends desire for this scenario. Then, the same results show that traditional frontends are performing as well as the learnable ones. Therefore, the model would not need to alter a well-initialized frontend to reduce the classification error. Section 3.2.1 examines these two hypotheses.

Modifying the filter’s functional form has been the primary source of novelty and improvement in the audio frontend literature. Yet, the results show a negligible gap between LEAF (Gabor filters) and SincNet+ (band-pass sinc filters). The extent of such claims has been questioned on conventional audio tasks and datasets [33]. Therefore, section 3.2.2 will probe the effect of the filter’s functional form in the context of bird species recognition. It contributes to comprehending the utility of these models for future work in bioacoustics.

Figure 6: Feature representation of frontends on a recording from European Robin, shown on top (photograph credit: `https://en.wikipedia.org/wiki/File:Erithacus_rubecula_with_cocke`

d_head.jpg). The features are similar, which explains the proximity of their performance in Table 2. No postprocessing was applied for visual purposes.

Table 3: Test results for LEAF-L and LEAF-B. See section 3.2.1 for details.

Model	accuracy	top-3 accuracy	f1-score	precision	recall
LEAF-L	93.97	98.55	93.53	94.48	92.82
LEAF-B	89.17	96.38	88.62	89.12	88.29

Figure 7: The filters' frequency response for (a) LEAF-L with 256 filters initialized from STFT, and (b) LEAF-B with 80 filters initialized linearly in an inappropriate region. LEAF-B changed significantly and formed an almost log-linear spacing after 1 kHz, similar to mel-filterbanks.

3.2.1 Impact of Frequency Resolution and Initialization on the Frontend

The frequency resolution of LEAF improved by initializing it using 256 linearly spaced filters. It imitates STFT from the main experiment to remove the imposed bias towards the log-linear scale. Call this larger model LEAF-L. Figure 7a shows that filters preserved their trend despite the additional flexibility. Hence, the stickiness was not due to the limited number of filters.

Another LEAF was initialized by 80 linearly spaced filters in low-frequency bands, which is inappropriate for analyzing bird vocalization since birds usually occupy the 1-16 kHz bands (mostly under 8 kHz). Call this badly initialized model LEAF-B. Figure 7b shows that the filters moved substantially this time and formed a log-linear spacing despite linear initialization.

Table 3 shows the performance of LEAF-L and LEAF-B, which are close to the results in Table 2. LEAF-B could improve, but the training stopped at 100 epochs for consistency across all experiments. LEAF-L and LEAF-B together reveal that the closeness of the results is due to the adequacy of the (mel-)spectrogram and not the suboptimal choice of the hyper-parameters in learnable frontends. Moreover, LEAF-B shows that a learnable frontend prefers log-linearly spaced filters like mel-filterbanks. The filters adapted to the frequency characteristics of the dataset when initialized inappropriately. Also, the occupied frequency bands were faithful to the content of our bird dataset, while the filters could reach 16 kHz. Overall evidence shows that STFT and mel-filterbanks put the learnable filters in a locally optimal spot for this dataset and task.

Table 4: Test results for LEAF-P and SincNet-P. See section 3.2.2 for details.

Model	accuracy	top-3 accuracy	f1-score	precision	recall
LEAF-P	82.78	93.34	82.03	84.73	81.79
SincNet-P	88.31	96.47	87.41	88.59	86.93

Figure 8: Summary of test results at decreasing levels of SNR for analyzing the natural noise tolerance. The ’*’ indicates the original result from Table 2. Each subfigure resulted from grouping the models having the attribute written on top within the 48 models.

3.2.2 Utility of the Filters Functional Form

Looking at the main result in Table 2, one notices a large gap between the static and the learnable frontends when using log compression without normalization. The following experiment will examine if the functional form of the filters has contributed to this performance gap.

Some details require an explanation. First, the learnable frontends had filters of length 801 (25 ms), but the (mel-)spectrogram used an fft window of length 512 (16 ms). Second, the learnable filters convolve with the waveform without stride (hop). The downsampling happens in the Gaussian pooling by a convolution with a stride of 320 points (10 ms). This stride is the same as the hop length for the (mel-)spectrogram. However, Gaussian pooling has a smoothing effect, unlike hopping. Thus, adjusting these details is necessary for a fair examination.

This experiment used both learnable frontends with 80 filters of length 512 (16 ms) that were initialized by mel-filterbanks. Gaussian pooling was removed, and downsampling happened in the filtering convolution layer by a stride of 320 (10 ms). Then, the outputs were log-compressed without normalization. These plain frontends are called LEAF-P and SincNet-P, and Table 4 shows their test results.

Both models significantly outperformed the (mel-)spectrogram. Although not shown here, the filters remained close to initialization. Thus, the previous conclusions about the adequacy of the (mel)-spectrogram still hold. However, it shows that the benefit of an audio frontend is not confined to the data-driven frequency band selection because models with virtually the same frequency bands are performing drastically differently. Therefore, the functional form of the filters has a noticeable impact, but other frontends can compensate for it by using additional operations like normalization.

Figure 9: Same as Figure 8 but using urban noise sources.

3.3 Noise Tolerance

Large neural networks can handle minor variations in the input caused by different compression schemes [34]. Thus, changing the range and scale of the inputs is unlikely to make a noticeable difference. Yet, it is readily evident in Table 2 that combining the compression with normalization is significantly better regardless of the frontend, whether through PCEN or conventional methods.

Lostanlen et al. [42] showed that PCEN whitens the data by Gaussianizing the magnitudes and decorrelating the frequency bands. Thus, it alleviates stationary background noise. The following analyses confirm this result for species recognition from outdoor recordings. Furthermore, it shows that other combinations of compression and normalization will also suppress noise.

Testing all 48 models on a noisy test set illustrates this property at various signal-to-noise ratios (SNR). The noise dataset is described in section 2.7, and the models did not see this dataset during the training. The summary results are shown in Figure 8 for the natural noise and Figure 9 for the urban noise. PCEN and other combinations of compression with normalization alleviate both types of interferences with bird vocalization signals.

4 Discussion

The current trend in computational bioacoustics is moving toward cutting-edge deep learning methods to enhance the analysis toolchains [8], [9], [14], [22]–[24], [28], [30], [54]. However, the machine learning field is moving fast [55], which leads to a surge of novel computation-oriented works for bioacoustics. This rapid development inevitably causes some contradictory results since the datasets, their quality, and experimental setups vary drastically among bioacoustics researchers. Also, deep learning models' stochastic and obscure nature contributes significantly to the discrepancy among the results [56]–[58]. Hence, practitioners make many design choices based on fuzzy evidence. Furthermore, proposed models and their specific hyper-parameters do not always behave as nicely as reported outside their benchmark datasets.

In this work, we aimed to reduce the ambiguity around one such design choice, selecting the proper audio feature to train the neural networks for bird species classification. Prior work showed concern about using features like mel-filterbanks and suggested investigating the waveform to extract species-specific information [14]. Our experiments showed that the (mel-)spectrogram is apt for capturing the informative patterns in bird vocalization to distinguish the common bird species. We demonstrated that many discrepancies in results stem from seemingly trivial technical details in data pre-processing and training setup. We now summarize our findings to aid future works in bird species recognition.

No free lunch for the frontends: Audio frontend literature has consistently reported the superiority of the proposed methods to traditional time-frequency features [34], [36], [38]. However, Schlüter et al. [33] proposed a faster version of LEAF, called EfficientLEAF, and showed that neither one outperforms mel-filterbanks consistently. Our work confirmed this in bird species recognition with an in-depth analysis of the working mechanism of such frontends. Additionally, the exhaustive experiments in section 3.1 showed why and when the results differed significantly. The discrepancy was not due to the (mel-)spectrogram missing the relevant frequency bands or their resolution since the learnable frontends preferred almost identical representation. It was mainly due to the more sophisticated pipeline of normalization, smoothing, and compression operations in learnable frontends. Also, the results showed that PCEN and normalization methods consistently improve the (mel-)spectrogram to produce comparable results to modern frontends.

We think picking the best model based on very close results is not informative. Although the learnable frontends were negligibly better in performance, they were much worse in computation time. Also, if we had not used the additional normalization with learnable frontends and had omitted this detail while using it as default with the (mel-)spectrogram, we could report that the learnable frontends are negligibly inferior. Moreover, we did not experiment with fft and hop sizes to keep the number of experiments manageable. These two hyper-parameters significantly impact the performance of the (mel-)spectrogram. Although the kernel and hop sizes of the learnable frontends were fixed, they had a slight advantage since the spreads of the Gaussian pooling filters for downsampling were learnable.

The main difference between many proposed audio frontends in the literature is the type of filter and sometimes the adjustments to compression and normalization operations. Some works slightly deviate from this narrative, such as convolutional restricted Boltzmann machine [59] and learnable wavelet transform [60]–[62]. However, as far as the task involves extracting joint temporal and spectral information from a signal without prior knowledge about the nature of the data and the task, the (mel-)spectrogram seems sufficient. One of the exceptions is learning the appropriate frequency bands if the sound events of interest occupy widely spread-out frequency bands. However, we did not observe a benefit from this property for bird vocalization in our ten chosen species.

Reporting a large gap between such models requires thorough investigation since the extracted features hardly differ in the amount of information they carry for most practical purposes. We should look for discrepancies in task-specific data pre-processing that could be structurally embedded in a learnable frontend rather than reporting the overall superiority of one frontend. For example, a wavelet transform might help a specific task and dataset by discarding the noise [62]. However, specialization in a narrow domain should logically come at the cost of less generalization. It does not mean one audio frontend is always superior across all datasets and tasks unless it solves a flaw shared by other frontends. Also, the result heavily depends on the backend classifier and the types of features it prefers due to its inductive biases. Thus, other frontends might achieve comparable results by modifying them to compensate for their disadvantages.

Efficacy of normalization: The remarkable improvement of the (mel-)spectrogram and learnable frontends from a simple normalization was partly unexpected since input normalization usually results in different parameter values but similar performance. We remind the readers that the normalization was applied on the backend's input, which is the frontend's output. Maybe a comparable accuracy is attainable without normalization but requires an optimal training regime, extensive hyper-parameter tuning, and potentially modifying the backend classifier. Therefore, input normalization at least made the learning much easier since all models used the same backend and optimization setting. Another reason for performance gain is that the PCEN and normalization schemes strengthened the models against unseen natural and urban noise (section 3.3). It is a desirable property since many bioacoustics tasks use outdoor noisy recordings.

A peculiar observation was that combining PCEN with other normalization methods enhanced the models further (Table 2). Notice that PCEN already has a normalizing effect due to adaptive gain control besides its root compression (see section 2.5). However, PCEN is a local operator that processes the channels independently and sequentially. In contrast, other methods like min-max normalization and standardization use the input's global statistics. Therefore, we think these two types of normalization had distinct and complementary effects. We should note that PCEN is a computationally expensive operation. Thus, log compression combined with a global normalization might be more suitable depending on the requirements. Also, see [33] for a faster alternative to PCEN that leverages temporal median subtraction and batch normalization.

Role of the filter's functional form: The functional form of the filters impacted the performance in complicated ways. In Table 2, smoothing by Gaussian pooling was the main distinguishing component between the learnable and static frontends that used log compression without normalization. Yet, the learnable ones outperformed the (mel-)spectrogram significantly despite capturing similar frequency bands (f1-scores: Mel-log 77.21, Stft-log 76.64, Sinc-log 90.70, and Leaf-log 92.73). Additionally, section 3.2.2 showed that without PCEN, normalization, and Gaussian pooling, Gabor and sinc filters still outperform the log-(mel-)spectrogram, although their performance declined (f1-scores: SincNet-P 87.41, LEAF-P 82.03). We conjecture that the filters have a smoothing effect that results in subtle denoising compared to STFT since the performance gap shrank in the presence of explicit normalization and PCEN. Regardless, this shows that some inherent properties of the filters can affect the results even when they capture similar frequency bands. Meanwhile, we also saw that other frontends can compensate for these by additional operations such as PCEN and normalization.

Another observation from section 3.2.2 was that sinc filters significantly outperformed Gabor filters (Table 4) while the prior work showed the opposite [38]. However, the best models of each frontend in the main experiments were comparable (Table 2). Hence, what is best depends on the particular task, dataset, and interaction of all the frontend components.

Data augmentation: Although the data augmentation was less effective than anticipated, we only augmented the waveform to reduce the backend's randomness and keep the experiments manageable. However, the backend could benefit from time-frequency augmentation techniques [63]. Thus, we encourage the practitioners to try both types of data augmentation for potential gain. For example, adding Gaussian noise could be more effective if applied on the backend's input since the frontend could reduce it by smoothing. Also, we suggest focusing more on sensible transformations that one can readily trust in their general benefits. Showing that a particular audio classifier benefits from extreme image data augmentation is suspicious since they do not respect the physical meaning of time-frequency representations. For example, birds do not sing flipped or rotated, but slight frequency translation might be helpful since aging impacts voice pitch [64]. Nanni et al. [65] provide a comprehensive study on audio augmentation techniques for natural sounds (cat and bird). It can serve as a guideline for choosing the baseline data augmentation methods in practice and for future investigation.

Future of audio frontends in bioacoustics: We suggest future frontend works in bioacoustics analyze their model’s cons and pros in more detail to show if their superior frontend addresses a flaw shared by others while having their best properties. It helps to identify when and why such learnable representations are necessary for acceptable results. Furthermore, revealing the mutual shortcomings aids future works to create better frontends and tests for their analysis. Some interesting investigations for future work are improving denoising [62] and leveraging source separation [66] in audio frontends (e.g., analyzing songs from dawn chorus).

In our humble opinion, making a component learnable because it is merely amenable to differentiable parametrization is not always an improvement. It is better to use knowledge-driven models instead of data-driven ones if the performance heavily depends on initialization based on already known and well-behaved parameters while deteriorating significantly by diverging from these initial values. We suggest always providing thorough analysis, even for seemingly trivial details, to reduce the epistemic uncertainty for practitioners and future research.

Clarifying the terminology: Riad et al. [67] proposed a 2d Gabor convolutional frontend to capture data-specific temporal and spectral modulations from the mel-spectrogram. They drew parallels from the neuroscience of the auditory system and showed that the filters learn meaningful parameters and representations. Similarly, Ren et al. [68] proposed an equivalent frontend for underwater acoustics using the spectrogram as input. These types of frontends require an already computed time-frequency representation. Therefore, it is better to consider them as part of the classifier or an intermediate component since they are one step further than the audio frontends operating directly on the waveform.

Notice that our work studies the adequacy of hand-designed and learnable time-frequency representations in capturing sufficient information to discern bird species. We do not make any claim about what type of model is the best to leverage the time-frequency information for species classification. The complete pipeline is beyond the choice of the input features and includes many other open challenges. However, it would be interesting to combine the frontends [33], [36], [38] and intermediate modules [67], [68] in future work and analyze their combined impact. For example, it is impossible to visualize the spectro-temporal development of all bird species songs using one time-frequency resolution (syllables might attach due to short spectral and temporal gaps). Hence, it seems beneficial to use an audio frontend with limited but learnable frequency bands on waveforms with high sampling rates and leverage one intermediate module of 2d Gabor filters in the first convolution layer of the backend to capture the species-specific spectro-temporal patterns.

Choosing the features in practice: The learnable filterbanks are very useful if prior knowledge about the dataset implies flexible frequency band selection. However, given the current audio frontend technology, the (mel-)spectrogram is almost as suitable as the waveform for bird species recognition. Moreover, we showed that frontends learn similar features (sections 3.2 and 3.2.1). If the development and inference time are of concern, the (mel-)spectrogram is the safest default choice unless the potential marginal improvement of a learnable frontend is necessary. We suggest using the (mel-)spectrogram and trying a few resolutions to pick the best STFT hyper-parameters experimentally (on a validation set). The fft and hop sizes usually impact the results significantly and might even remove the little performance gap between the static and learnable filterbanks for species classification. We also encourage using PCEN (see [33] for a faster alternative) and a global normalization method for bird species recognition. The min-max normalization is a good default.

5 Conclusion

A detailed ablation study of the learnable and static audio frontends showed little benefit from data-driven frequency selectivity for bird vocalization. Nonetheless, the functional form of the learnable filters impacted the performance despite the homogeneity of frequency channels across the frontends. However, adequate normalization and compression operations reduced the performance gap between the frontends. In particular, PCEN, min-max normalization, and standardization made the models resilient against unseen environmental noise and consistently made the (mel-)spectrogram comparable to modern audio frontends.

An in-depth explanation was provided for each experiment, followed by a thorough discussion of all the results to summarize the observations and practical findings. This work concludes that the (mel-)spectrogram combined with PCEN and a global normalization method is on par with learnable audio frontends that operate on the waveform. The findings may not apply to all animals and scenarios but should be valid for typical bird species recognition tasks.

Audio frontends that use waveforms might significantly increase computation time and latency. Therefore, a marginal improvement compared to using the (mel-)spectrogram should be necessary for the task to be considered a proper trade-off. Regardless, adapting the time-frequency representation to the dataset's characteristics and using physically informed filters are intriguing ideas for the discussed reasons. The topic deserves further research to build efficient learnable frontends for bioacoustics and identify the circumstances where they are most appropriate.

Declaration of Competing Interest

None.

Acknowledgements

This research was funded by the Ghent University Special Research Fund (grant BOF/STA/202102/005) and published with the support of the University Foundation of Belgium. The authors also would like to thank the editor and the reviewers for their valuable comments, which considerably improved this work.

References

[1] Ç. H. Şekercioğlu, G. C. Daily, and P. R. Ehrlich, "Ecosystem consequences of bird declines," *Proceedings of the National Academy of Sciences*, vol. 101, no. 52, pp. 18 042–18 047, 2004.

[2] K. J. Gaston, T. M. Blackburn, and K. K. Goldewijk, "Habitat conversion and global avian biodiversity loss," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, no. 1521, pp. 1293–1300, 2003.

[3] C. Parmesan and G. Yohe, "A globally coherent fingerprint of climate change impacts across natural systems," *nature*, vol. 421, no. 6918, pp. 37–42, 2003.

[4] T. A. Marques, L. Thomas, S. W. Martin, *et al.*, "Estimating animal population density using passive acoustics," *Biological reviews*, vol. 88, no. 2, pp. 287–309, 2013.

[5] T. Ganchev, *Computational bioacoustics: Biodiversity monitoring and assessment*. Walter de Gruyter GmbH & Co KG, 2017, vol. 4.

[6] J. Sueur and A. Farina, "Ecoacoustics: The ecological investigation and interpretation of environmental sound," *Biosemiotics*, vol. 8, pp. 493–502, 2015.

[7] R. D. Gregory and A. van Strien, "Wild Bird Indicators: Using Composite Population Trends of Birds as Measures of Environmental Health," *Ornithological Science*, vol. 9, no.

1, pp. 3–22, 2010. DOI: 10.2326/osj.9.3. [Online]. Available: https://doi.org/10.2326/osj.9.3.

[8] J. Xie, Y. Zhong, J. Zhang, S. Liu, C. Ding, and A. Triantafyllopoulos, "A review of automatic recognition technology for bird vocalizations in the deep learning era," *Ecological Informatics*, p. 101 927, 2022.

[9] D. Stowell, M. D. Wood, H. Pamu la, Y. Stylianou, and H. Glotin, "Automatic acoustic detection of birds through deep learning: The first bird audio detection challenge," *Methods in Ecology and Evolution*, vol. 10, no. 3, pp. 368–380, 2019.

[10] A. Balmford, R. E. Green, and M. Jenkins, "Measuring the changing state of nature," *Trends in Ecology & Evolution*, vol. 18, no. 7, pp. 326–330, 2003.

[11] T. S. Brandes, "Automated sound recording and analysis techniques for bird surveys and conservation," *Bird Conservation International*, vol. 18, no. S1, S163–S173, 2008.

[12] *Xeno-canto public repository of bird recordings*, https://xeno-canto.org/, Accessed: 2023-08-23.

[13] K. A. Swiston and D. J. Mennill, "Comparison of manual and automated methods for identifying target sounds in audio recordings of pileated, pale-billed, and putative ivory-billed woodpeckers," *Journal of Field Ornithology*, vol. 80, no. 1, pp. 42–50, 2009.

[14] D. Stowell, "Computational bioacoustics with deep learning: A review and roadmap," *PeerJ*, vol. 10, e13152, 2022.

[15] Z. Chen and R. C. Maher, "Semi-automatic classification of bird vocalizations using spectral peak tracks," *The Journal of the Acoustical Society of America*, vol. 120, no. 5, pp. 2974–2984, 2006.

[16] I. Potamitis, "Automatic classification of a taxon-rich community recorded in the wild," *PloS one*, vol. 9, no. 5, e96936, 2014.

[17] M. Ehnes and J. Foote, "Comparison of autonomous and manual recording methods for discrimination of individually distinctive ovenbird songs," *Bioacoustics*, vol. 24, no. 2, pp. 111–121, 2015.

[18] T. S. Brandes, "Feature vector selection and use with hidden markov models to identify frequency-modulated bioacoustic signals amidst noise," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1173–1180, 2008.

[19] M. B. Trawicki, "Multispecies discrimination of whales (cetaceans) using hidden markov models (hmms)," *Ecological Informatics*, vol. 61, p. 101 223, 2021, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2021.101223. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954121000145.

[20] T. Andreassen, A. Surlykke, and J. Hallam, "Semi-automatic long-term acoustic surveying: A case study with bats," *Ecological Informatics*, vol. 21, pp. 13–24, 2014.

[21] D. Stowell and M. D. Plumbley, "Audio-only bird classification using unsupervised feature learning," in *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014*, 2014, pp. 673–684.

[22] S. Kahl, C. M. Wood, M. Eibl, and H. Klinck, "Birdnet: A deep learning solution for avian diversity monitoring," *Ecological Informatics*, vol. 61, p. 101 236, 2021.

[23] S. Zsebok, M. F. Nagy-Egri, G. G. Barnaföldi, *et al.*, "Automatic bird song and syllable segmentation with an open-source deep-learning object detection method–a case study in the collared flycatcher," *Ornis Hungarica*, vol. 27, no. 2, pp. 59–66, 2019.

[24] Q. Tang, L. Xu, B. Zheng, and C. He, "Transound: Hyper-head attention transformer for birds sound recognition," *Ecological Informatics*, vol. 75, p. 102 001, 2023, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2023.102001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954123000304.

[25] J. Xie and M. Zhu, "Handcrafted features and late fusion with deep learning for bird sound classification," *Ecological Informatics*, vol. 52, pp. 74–81, 2019, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2019.05.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954118302991.

[26] X. Zhang, A. Chen, G. Zhou, Z. Zhang, X. Huang, and X. Qiang, "Spectrogram-frame linear network and continuous frame sequence for bird sound classification," *Ecological Informatics*, vol. 54, p. 101 009, 2019, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2019.101009. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S1574954119303206.

[27] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[28] V. Morfi, R. F. Lachlan, and D. Stowell, “Deep perceptual embeddings for unlabelled animal sound events,” *The Journal of the Acoustical Society of America*, vol. 150, no. 1, pp. 2–11, 2021.

[29] J. Sinnott, M. Sachs, and R. D. Hienz, “Aspects of frequency discrimination in passerine birds and pigeons.,” *Journal of Comparative and Physiological Psychology*, vol. 94, no. 3, p. 401, 1980.

[30] F. J. Bravo Sanchez, M. R. Hossain, N. B. English, and S. T. Moore, “Bioacoustic classification of avian calls from raw sound waveforms with an open-source deep learning architecture,” *Scientific Reports*, vol. 11, no. 1, pp. 1–12, 2021.

[31] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] N. J. Beaudry and R. Renner, “An intuitive proof of the data processing inequality,” *arXiv preprint arXiv:1107.0740*, 2011.

[33] J. Schlüter and G. Gutenbrunner, “Efficientleaf: A faster learnable audio frontend of questionable use,” in *2022 30th European Signal Processing Conference (EUSIPCO)*, IEEE, 2022, pp. 205–208.

[34] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, “Learning the speech front-end with raw waveform CLDNNs,” in *Proc. Interspeech 2015*, 2015, pp. 1–5. DOI: 10.21437/Interspeech.2015-1.

[35] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, “Learning filterbanks from raw speech for phone recognition,” in *2018 IEEE international conference on acoustics, speech and signal Processing (ICASSP)*, IEEE, 2018, pp. 5509–5513.

[36] M. Ravanelli and Y. Bengio, *Speaker recognition from raw waveform with sincnet*, 2019. arXiv: 1808.00158 [eess.AS].

[37] P.-G. Noé, T. Parcollet, and M. Morchid, “Cgcnn: Complex gabor convolutional neural network on raw speech,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7724–7728. DOI: 10.1109/ICASSP40776.2020.9054220.

[38] N. Zeghidour, O. Teboul, F. d. C. Quitry, and M. Tagliasacchi, “Leaf: A learnable frontend for audio classification,” *arXiv preprint arXiv:2101.08596*, 2021.

[39] R. Schluter, I. Bezrukov, H. Wagner, and H. Ney, “Gammatone features and feature combination for large vocabulary speech recognition,” in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07*, vol. 4, 2007, pp. IV-649-IV–652. DOI: 10.1109/ICASSP.2007.366996.

[40] D. Gabor, “Theory of communication. part 1: The analysis of information,” *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[41] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, “Trainable frontend for robust and far-field keyword spotting,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 5670–5674.

[42] V. Lostanlen, J. Salamon, M. Cartwright, *et al.*, “Per-channel energy normalization: Why and how,” *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 39–43, 2018.

[43] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[44] T. maintainers and contributors, *Torchvision: Pytorch’s computer vision library*, https://github.com/pytorch/vision, 2016.

[45] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, Brisbane, Australia: ACM Press, Oct. 13, 2015, pp. 1015–1018, ISBN: 978-1-4503-3459-4. DOI: 10.1145/2733373.2806390. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390.

[46] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980[cs.LG].

[47] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[48] Y.-Y. Yang, M. Hira, Z. Ni, *et al.*, "Torchaudio: Building blocks for audio and speech processing," *arXiv preprint arXiv:2110.15018*, 2021.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[50] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[51] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[52] T. pandas development team, *Pandas-dev/pandas: Pandas*, version v1.5.3, Jan. 2023. DOI: 10.5281/zenodo.7549438. [Online]. Available: https://doi.org/10.5281/zenodo.7549438.

[53] C. da Costa-Luis, S. K. Larroque, K. Altendorf, *et al.*, *tqdm: A fast, Extensible Progress Bar for Python and CLI*, version v4.66.1, Aug. 2023. DOI: 10.5281/zenodo.8233425. [Online]. Available: https://doi.org/10.5281/zenodo.8233425.

[54] G. Gupta, M. Kshirsagar, M. Zhong, S. Gholami, and J. L. Ferres, "Comparing recurrent convolutional neural networks for large scale bird species classification," *Scientific reports*, vol. 11, no. 1, p. 17 085, 2021.

[55] X. Tang, X. Li, Y. Ding, M. Song, and Y. Bu, "The pace of artificial intelligence innovations: Speed, talent, and trial-and-error," *Journal of Informetrics*, vol. 14, no. 4, p. 101 094, 2020, ISSN: 1751-1577. DOI: https://doi.org/10.1016/j.joi.2020.101094. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1751157720301991.

[56] R. Dror, S. Shlomov, and R. Reichart, "Deep dominance-how to properly compare deep neural models," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2773–2785.

[57] R. Saleem, B. Yuan, F. Kurugollu, A. Anjum, and L. Liu, “Explaining deep neural networks: A survey on the global interpretation methods,” *Neurocomputing*, vol. 513, pp. 165–180, 2022, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.09.129. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222012218.

[58] J. Gawlikowski, C. R. N. Tassi, M. Ali, *et al.*, “A survey of uncertainty in deep neural networks,” *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1513–1589, 2023.

[59] H. B. Sailor, D. M. Agrawal, and H. A. Patil, “Unsupervised Filterbank Learning Using Convolutional Restricted Boltzmann Machine for Environmental Sound Classification,” in *Proc. Interspeech 2017*, 2017, pp. 3107–3111. DOI: 10.21437/Interspeech.2017-831.

[60] W. Ha, C. Singh, F. Lanusse, S. Upadhyayula, and B. Yu, “Adaptive wavelet distillation from neural networks through interpretations,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 669–20 682, 2021.

[61] G. Michau, G. Frusque, and O. Fink, “Fully learnable deep wavelet transform for unsupervised monitoring of high-frequency time series,” *Proceedings of the National Academy of Sciences*, vol. 119, no. 8, e2106598119, 2022.

[62] G. Frusque and O. Fink, “Learnable wavelet packet transform for data-adapted spectrograms,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 3119–3123.

[63] D. S. Park, W. Chan, Y. Zhang, *et al.*, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” in *Proc. Interspeech 2019*, 2019, pp. 2613–2617. DOI: 10.21437/Interspeech.2019-2680.

[64] M. L. Berg, S. C. Beebe, J. Komdeur, *et al.*, “Senescence of song revealed by a long-term study of the seychelles warbler (acrocephalus sechellensis),” *Scientific reports*, vol. 10, no. 1, p. 20 479, 2020.

[65] L. Nanni, G. Maguolo, and M. Paci, “Data augmentation approaches for improving animal audio classification,” *Ecological Informatics*, vol. 57, p. 101 084, 2020, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2020.101084. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S1574954120300340.

[66] Y. Luo and N. Mesgarani, “Tasnet: Time-domain audio separation network for real-time, single-channel speech separation,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 696–700.

[67] R. Riad, J. Karadayi, A.-C. Bachoud-Lévi, and E. Dupoux, “Learning spectro-temporal representations of complex sounds with parameterized neural networks,” *The Journal of the Acoustical Society of America*, vol. 150, no. 1, pp. 353–366, 2021.

[68] J. Ren, Y. Xie, X. Zhang, and J. Xu, “Ualf: A learnable front-end for intelligent underwater acoustic classification system,” *Ocean Engineering*, vol. 264, p. 112 394, 2022, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2022.112394. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801822016833.

Declaration of interests

☒The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Highlights

- Learnable audio frontends impact species classifiers significantly.
- Learnable filters are log-linearly spaced for bird vocalization like mel-filterbank.
- PCEN and normalization make mel-spectrogram and learnable filters comparable.
- Compression and normalization make the models robust against environmental noise.
- Filter parametric form is significant without normalization and smoothing modules.

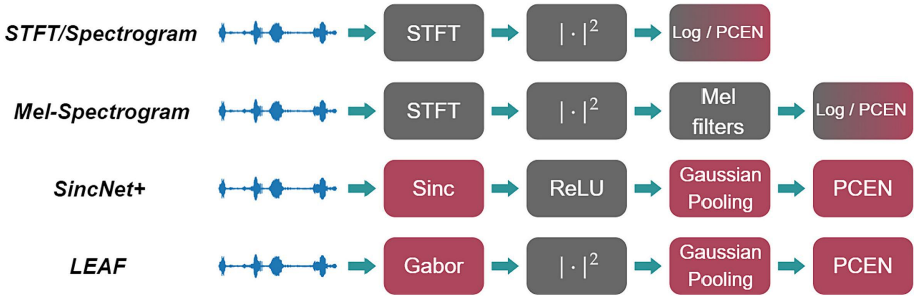

STFT/Spectrogram
STFT
$|\cdot|^2$
Log / PCEN
Mel-Spectrogram
STFT
$|\cdot|^2$
Mel filters
Log / PCEN
SincNet+
Sinc
ReLU
Gaussian Pooling
PCEN
LEAF
Gabor
$|\cdot|^2$
Gaussian Pooling
PCEN

Figure 1

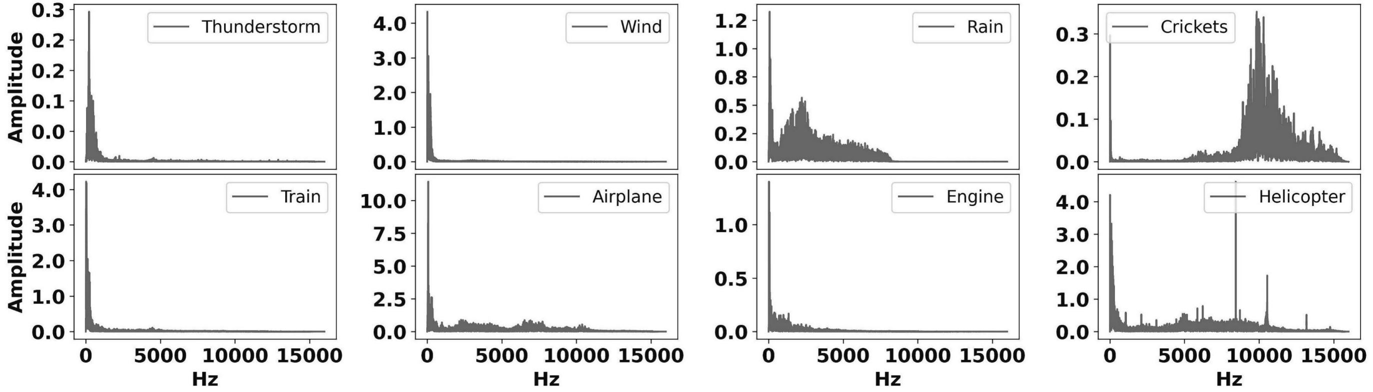
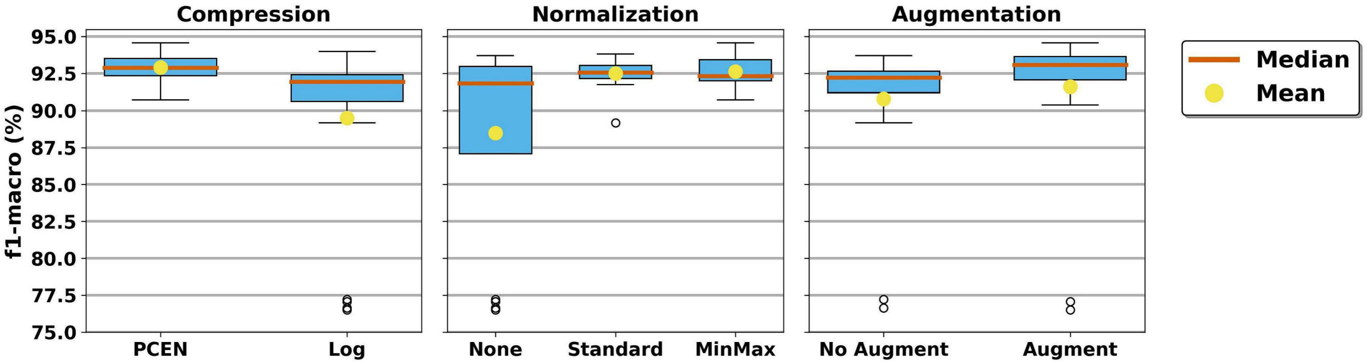
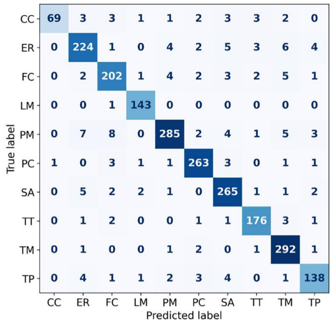

Thunderstorm
Wind
Rain
Crickets
Train
Airplane
Engine
Helicopter
Amplitude
Hz
0.3
0.2
0.1
0.0
4.0
3.0
2.0
1.0
1.2
0.8
0.5
10.0
7.5
5.0
2.5
0
5000
10000
15000

Figure 2

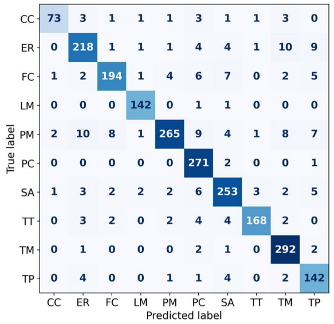


Compression
Normalization
Augmentation
f1-macro (%)
95.0
92.5
90.0
87.5
85.0
82.5
80.0
77.5
75.0
PCEN
Log
None
Standard
MinMax
No Augment
Augment
Median
Mean

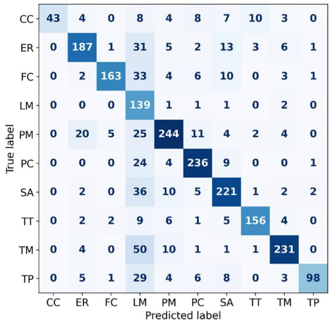
Figure 3



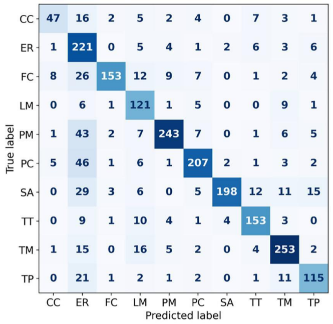
(a) LEAF-PCEN



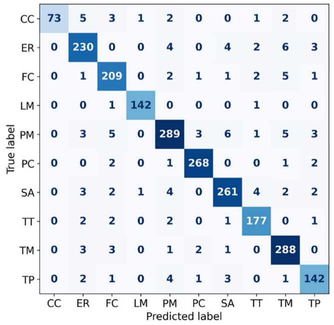
(b) Sinc-PCEN



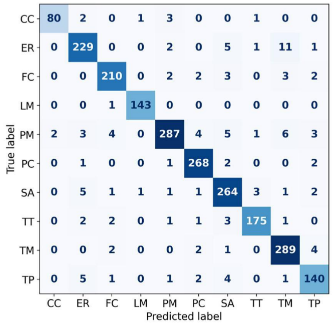
(c) Mel-Log



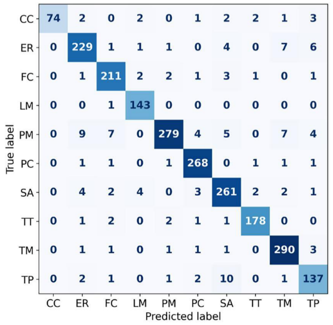
(d) STFT-Log



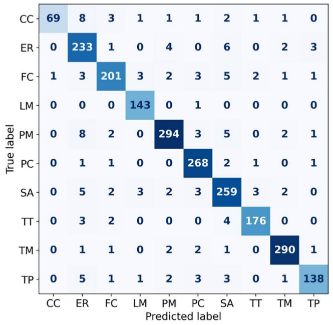
(e) LEAF-PCEN-M-A



(f) Sinc-PCEN-M-A

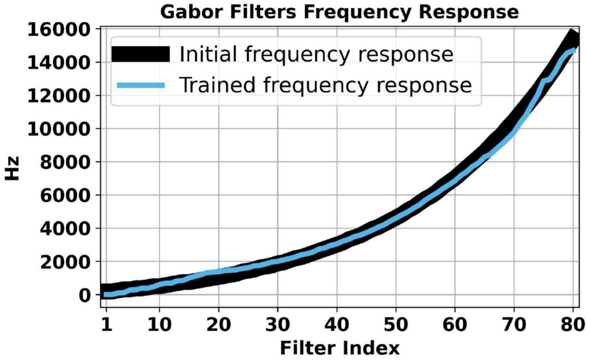


(g) Mel-PCEN-M-A

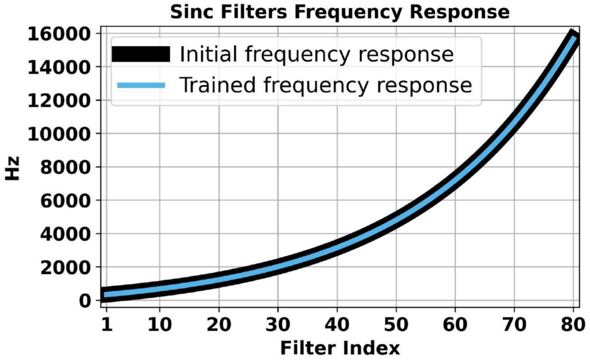


(h) STFT-PCEN-S-A

Figure 4

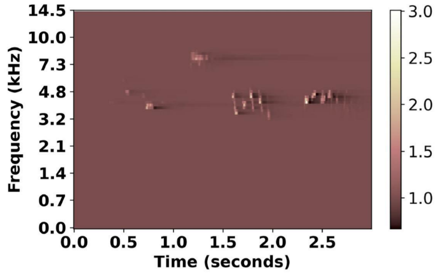


(a) LEAF filters.

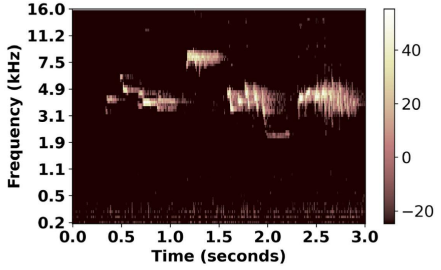


(b) SincNet+ filters.

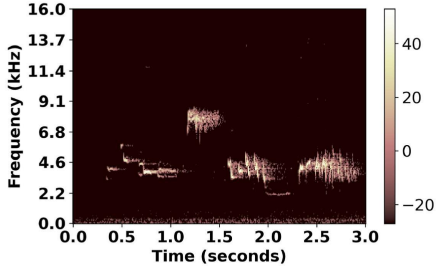
Figure 5



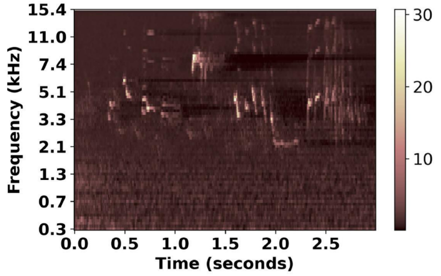
(a) LEAF-PCEN



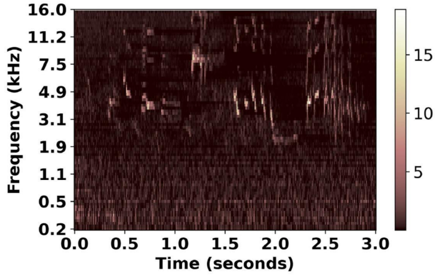
(b) Mel-Log



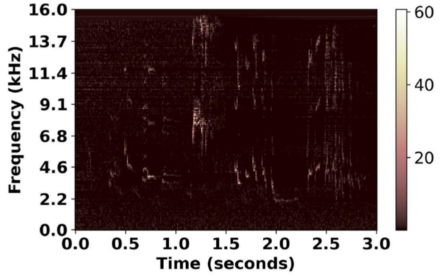
(c) STFT-Log



(d) Sinc-PCEN

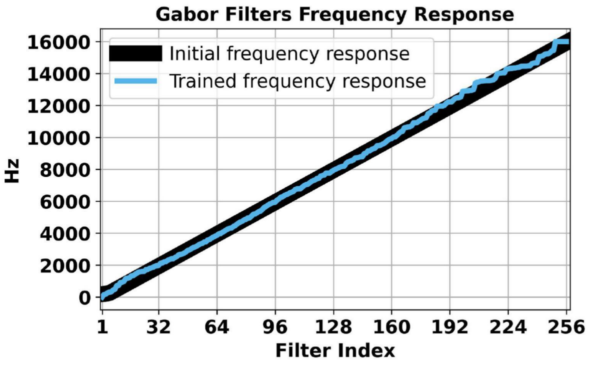


(e) Mel-PCEN

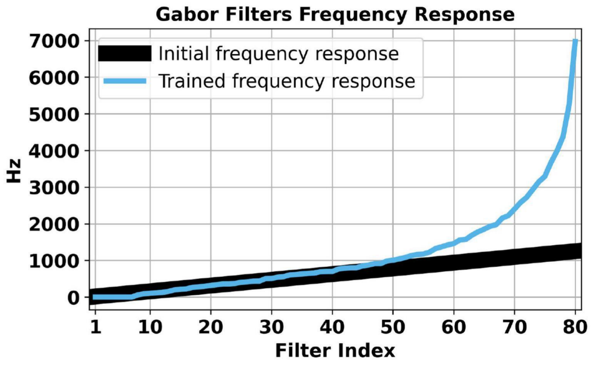


(f) STFT-PCEN

Figure 6



(a) LEAF-L



(b) LEAF-B

Figure 7

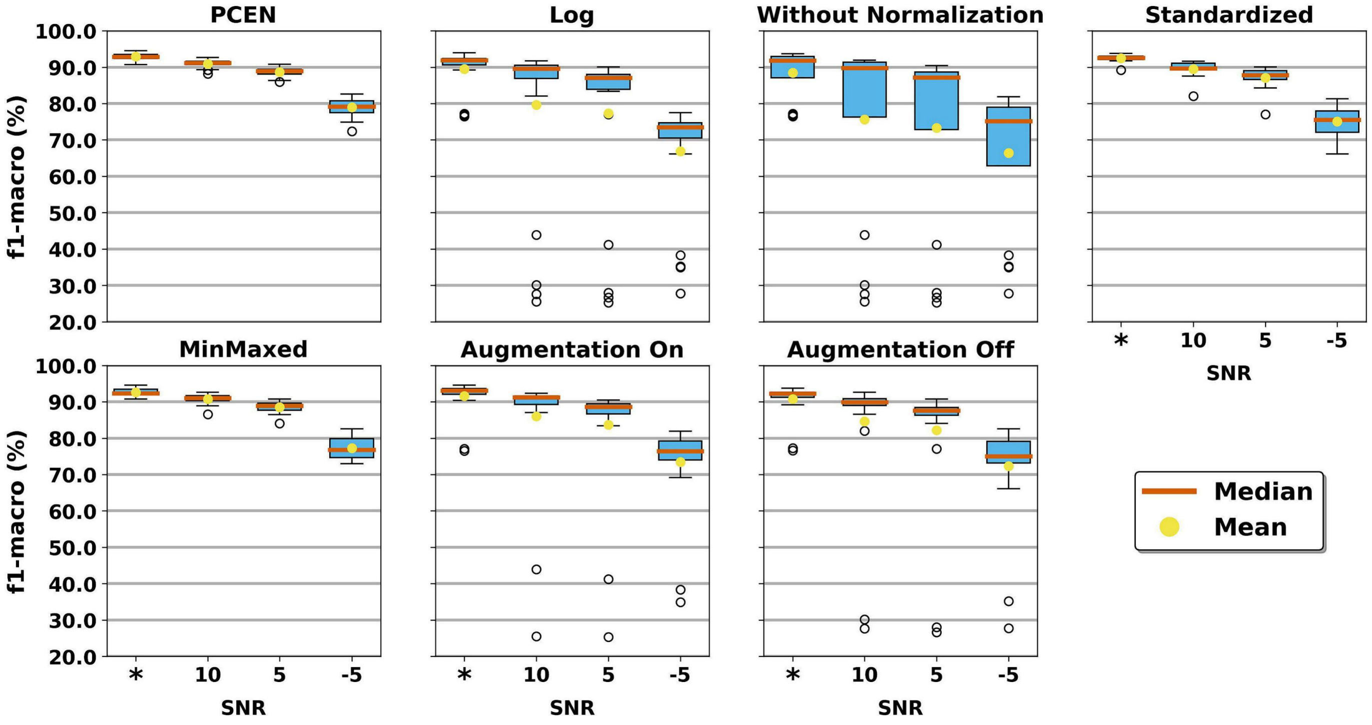

PCEN
Log
Without Normalization
Standardized
MinMaxed
Augmentation On
Augmentation Off
f1-macro (%)
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
*
10
5
-5
SNR
Median
Mean

Figure 8

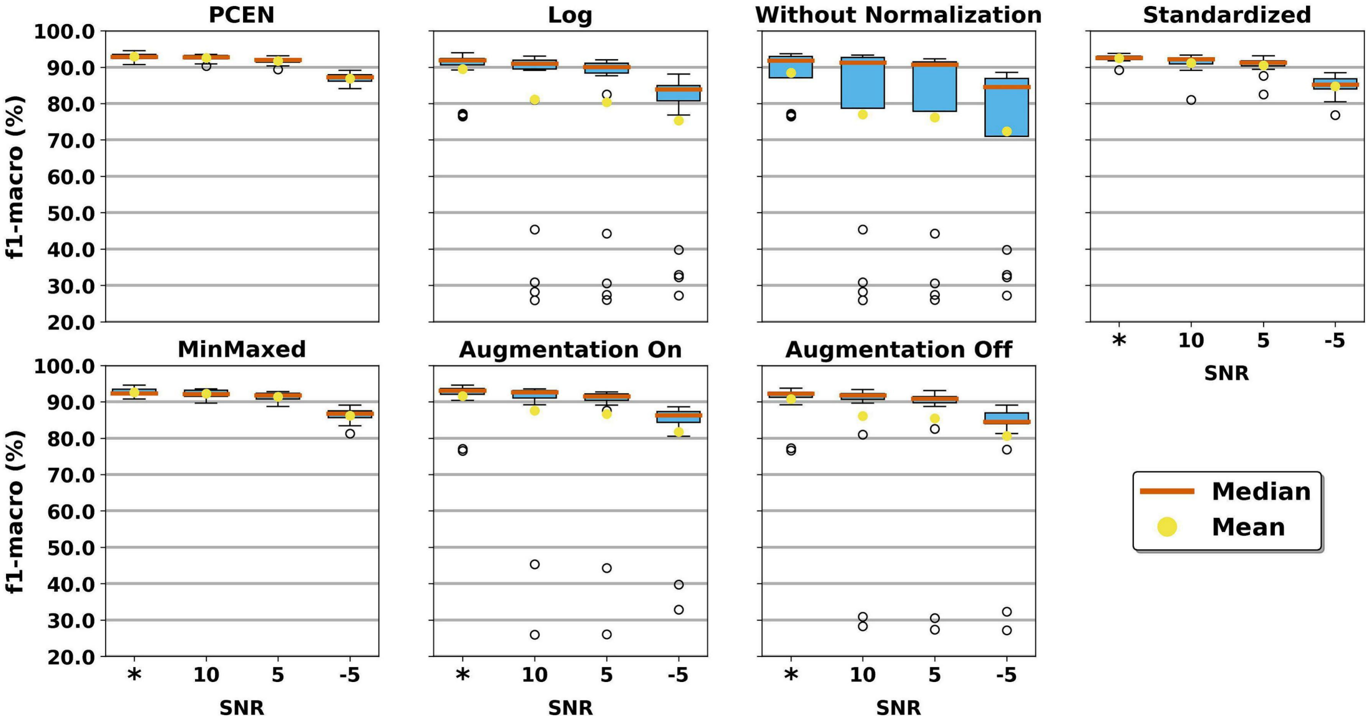

PCEN
Log
Without Normalization
Standardized
MinMaxed
Augmentation On
Augmentation Off
f1-macro (%)
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
*
10
5
-5
SNR
Median
Mean

Figure 9