Toward in-context bioacoustic sound event detection

Benjamin Hoffman
Earth Species Project
benjamin@earthspecies.org

David Robinson
Earth Species Project
david@earthspecies.org

***Abstract*—We introduce an in-context learning approach to bioacoustic sound event detection. Our approach consists of a large pre-trained transformer model which, when prompted with a small amount of labeled audio, directly predicts detection labels on unlabeled audio. To train our model, we constructed a large audio database, which we used to generate acoustic scenes with temporally fine-grained detection labels. On the validation set for the 2024 DCASE Few-shot bioacoustic event detection challenge, our best-performing submission achieves an average F1 score of 0.584, improving on the challenge baseline by 0.063.**

1. Introduction

Few-shot learning is a promising approach for processing bioacoustic data for scientific applications [1]. In natural language processing, few-shot learning has been formulated as in-context learning [2], [3], where a large language model performs few-shot learning tasks without fine-tuning. This is accomplished by presenting the model with labeled support data, as well as unlabeled query data, in a single text prompt. In-context learning has been adopted in other domains, such as image segmentation [4] and text-to-speech synthesis [5].

In our submission[1] to Task 5 of the 2024 DCASE Challenge ("Few-shot bioacoustic event detection"), we developed an in-context learning framework for bioacoustic sound event detection. We trained a large transformer-based model to directly predict detection labels on unlabeled query audio, without any fine-tuning. The input prompt to this model is the labeled support audio, as well as the unlabeled query audio (Figure 1A). Unlike prototypical networks [6], which have been used in previous iterations of the DCASE Challenge, this framework allows the model to attend to support and query audio simultaneously, and does not rely on any metric learning.

To develop our model, we constructed a large database of background audio and bioacoustic sound events from publicly available sources. We used this database to generate acoustic scenes, together with temporally fine-grained detection labels, on-the-fly. By doing so, we gain access to the scale of data required to train a large transformer model (Figure 1B). Additionally, we can generate data that reflects a domain shift between support and query data [1], for instance, by changing the background audio between a support example and its associated query example.

1. Model weights and few-shot inference API will be posted at https://github.com/earthspecies/fewshot.

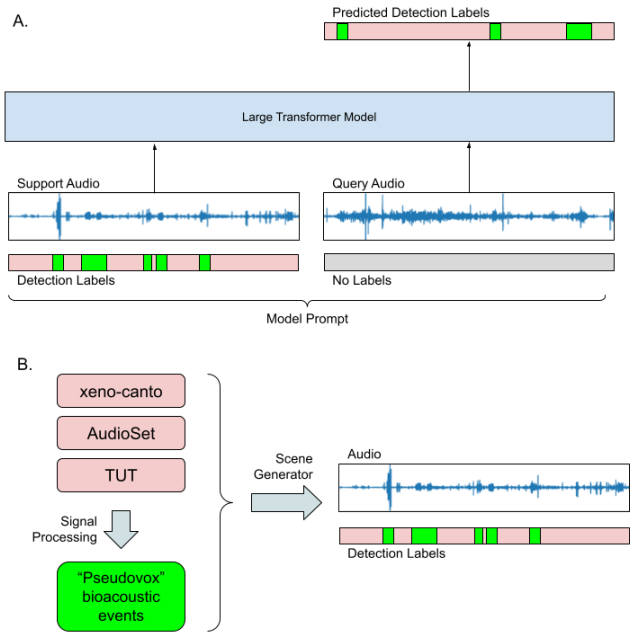

Figure 1. Summary of our system. A: In-context learning approach. B: Data generation procedure.

On the 2024 validation set, our best performing submission achieves a F1 score of 0.584, which is an improvement of 0.063 over the challenge baseline system. Overall, our work introduces two approaches to few-shot bioacoustic sound event detection: in-context learning and on-the-fly training data generation. Future work could focus on refining our model architecture and training settings, as well as broadening our approach to data generation to reflect a wider set of detection problems in the training data.

2. Data Generation

To create our training data, we gather *background* audio from publicly available sources, and from this audio extract short sub-clips called pseudo-vocalizations or *pseudovox*.

To generate data, we add pseudovox into randomly chosen background audio clips. We use the start- and end-times of these pseudovox as detection labels for the generated data. Throughout, we apply data augmentations to increase the variety of sounds we produce.

2.1. Data Pre-processing

We constructed a large database of audio drawn from xeno-canto [7], AudioSet [8], and TUT acoustic scenes [9] (Table 1). In xeno-canto, each recording is tagged with at most one focal species; we considered species with at least 25 recordings (4487 species), and randomly sampled 10 recordings from each. We also sampled 1000 recordings tagged as *Soundscape*. From AudioSet, we used the balanced training and evaluation datasets. From TUT, we used the entire development dataset. All recordings were trimmed to a maximum duration of 60 seconds. We included all of these recordings in our set of background audio clips.

For each recording, we applied a sequence of signal-processing steps to produce a set of short events called *pseudovox* that could plausibly resemble animal vocalizations. We resampled each recording to 22.5 kHz, and then used BirdMixit [10] to separate the recording into four stems.[2] Next, for each stem, we computed an amplitude envelope, and extracted contiguous segments where the amplitude envelope exceeded 25% of the maximum amplitude of the raw recording. Finally, we applied BirdNet [11] to each of these segments. We set aside all segments where BirdNet detected nothing, a non-biological sound source (e.g. *Engine* or *Fireworks*), or a human-generated sound source. The remaining segments became our set of pseudovox. In subsequent steps, we used the BirdNet prediction for each pseudovox as its *pseudo-label* (Table 1, right). In the end, we obtained most pseudovox from xeno-canto, but did obtain some from AudioSet and TUT.

Our audio database relied on freely available bird-related data (xeno-canto) and pre-trained machine learning models (BirdMixit, and BirdNet). Our assumption was that birds make a wide variety of sounds, many of which resemble those made by other species. Future work could extend data collection methods to other taxa.

2.2. Data Generation Procedure

Using our audio database, we develop a method for generating acoustic scenes with fine-grained detection labels, on-the-fly during model training. We focus on generating the widest variety of these scenes as possible, making our approach loosely related to domain randomization [12] that is used in other simulated-to-real data transfer problems.

To generate a scene, we do the following:

1) Sample two background audio clips, which form the support background audio and query background audio. With probability 0.5, these two audio clips are the same. These are looped to 40 seconds and 10 seconds, respectively.
2) Sample λ from $\{.04, .08, .17, .33\}$ calls/second. Sample $n_s \sim \text{Poisson}(40\lambda)$, $n_q \sim \text{Poisson}(10\lambda)$.
3) Sample a pseudo-label l uniformly at random. Sample a background "origin" audio clip a that gave rise to ≥ 4 pseudovox of pseudo-label l.
4) Sample with replacement n_q pseudovox with pseudo-label l that came from a. Sample n_q start times from $\text{Unif}(0, 10)$.
5) Sample with replacement n_s pseudovox with pseudo-label l that came from a, that were not sampled in step 4. Sample n_s start times from $\text{Unif}(0, 40)$.
6) Sample $SNR_{dB} \sim \text{Unif}(-5, 2)$. Sample r from $\{0.5, 1, 2\}$
7) Resample the n_q pseudovox by a factor of r. Add them to the query background audio at the chosen start times, with the chosen signal-to-noise ratio SNR_{db}. Do the same for the n_s support pseudovox.
8) Record the start and end times of these pseudovox. These are the events labeled POS in our data. Other times are labeled NEG.
9) Repeat steps 2-7 to add additional distractor events; do not record their start and end times.

We also sample from different alterations of this generation process, which we call *scenarios*. The scenarios we used are summarized in Table 2.

3. Transformer model

We propose a transformer architecture predicting on the query audio while prompted by the support and query. Our architecture consists of an *audio encoder*, a *context encoder*, and an *aggregation module* (Figure 2).

3.1. Audio Encoder

Given a support and query audio pair, we independently pass support and query audio in 10-second windows through the audio encoder, producing frame-level embeddings (Figure 2A). The audio encoder is initialized from ATST-FRAME [13], which was pretrained on AudioSet using self-supervision. The input to the audio encoder is mono audio resampled to 16KHz, and the output audio embedding is a 9216-dimension frame-level embedding at 25 Hz. We chose to window the data because of the quadratic complexity of transformer models, and we chose a 10-second window because it matched the pre-training setup of ATST-FRAME.

3.2. Context Encoder

The context encoder (Figure 2B) is a second transformer, which processes label-enriched support in the same sequence as the query in order to tailor embeddings to the few-shot support prompt.

2. BirdMixit is included in Google Perch, which is allowed under DCASE challenge rules.

Data Source	# Files Sampled	Total Duration (hours)	# Pseudovox Sampled	Pseudovox Duration (hours)	# Pseudolabel types
xeno-canto	45870	470	778940	72.5	3263
AudioSet	31361	86	66882	6.7	1964
TUT	4680	13	2641	0.2	575

TABLE 1. SUMMARY OF AUDIO DATABASE USED TO GENERATE ACOUSTIC SCENES WITH DETECTION LABELS.

Scenario Name	Effect on generation process
Basic	None; described in main text.
Low SNR	In step 5, $SNR_{dB} \sim \text{Unif}(-10, -5)$. Background audio sampled only from TUT.
Fine-grained SNR	Pseudovox for POS and distractor events are drawn from the same origin clip and have the same pseudo-label. However, distractor events are lower SNR than POS events.
Disjunction across pseudo-labels	Pseudovox for POS events are drawn from two pseudo-labels, from two different origin clips
Disjunction within pseudo-labels	Pseudovox for POS events still represent one pseudo-label, but are drawn from two different origin clips.
Generalization within pseudo-labels	Pseudovox for POS events in query audio still represent one pseudo-label, but are drawn from different origin clip than the support POS pseudovox. Query pseudovox are required to be similar duration to support pseudovox.
Fine-grained general	Pseudovox for distractor events are required to be similar duration to the pseudovox for POS events.

TABLE 2. "SCENARIO" VARIANTS OF DATA GENERATION PROCEDURE.

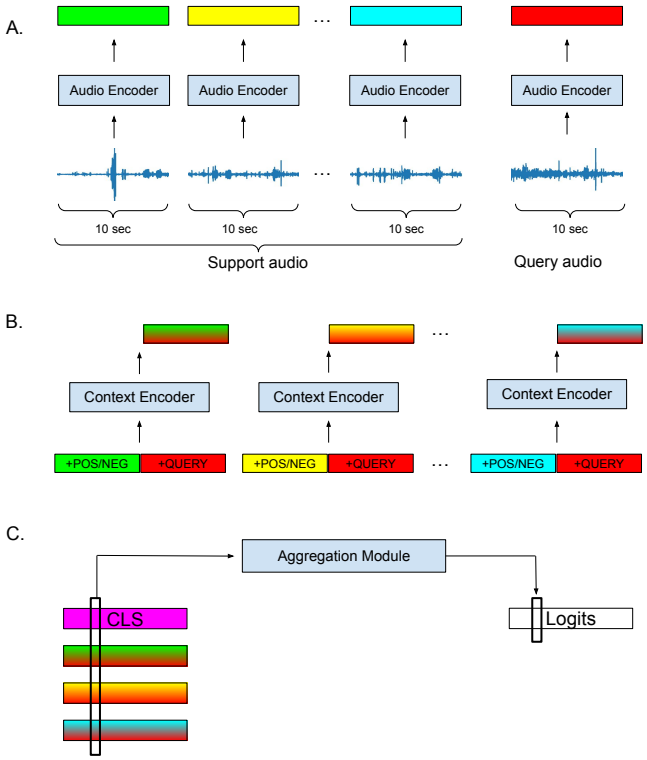

Figure 2. Summary of how our model makes predictions for one 10-second window of query audio. A: 10-second audio windows are passed through audio encoder B: Encoded labels are added to support and query embeddings, which are concatenated and passed to the context encoder C: Context embeddings are passed into the aggregation module, which makes a final detection prediction for the query window.

We one-hot encode labels for each frame; possible labels are POS for events of interest, NEG for times where there are no events of interest, and QUERY for audio where the model should make a prediction. These are passed through a linear layer and added directly to the audio embeddings. Then, for each (10-second support window, 10-second query window) pair, the enriched support and query audio embeddings are concatenated and fed into the context encoder.

The context encoder has an architecture matching BERT-small [14], but with SwiGLU activation [15] and rotary positional encoding [16]. The input to the context encoder is of shape $[9216, 500]$, corresponding to 10 seconds of support audio at 25 Hz, and 10 seconds of query audio at 25 Hz. We retain only the output corresponding to the 10 seconds of query audio, which is of shape $[512, 250]$.

3.3. Aggregation Module

We initially divided our audio into 10-second windows, and therefore we need to combine information from each support window and make a final prediction. To do so, we introduce an aggregation module, which is an additional transformer (Figure 2C). For a fixed query window Q, consider the pairs $(S_0, Q), \ldots, (S_{K-1}, Q)$, where each S_k is a distinct support window. The context encoder produces K-many frame-level context embeddings $c_0, \ldots, c_{K-1}$ associated with the query window Q, each of shape $[512, 250]$. The aggregation module makes a final prediction of the detections for Q based on these c_k. For each frame $0 \leq i < 250$, the aggregation module takes as input the set of 512-dimension feature vectors $\{c_0[:, i], \ldots, c_{K-1}[:, i]\}$, as well as a learned CLS token. We retain only the output associated with the CLS token. This is passed through a final linear layer to produce a one-dimensional vector, interpreted as a logit for the detection probability of frame i.

The aggregation model's architecture matches BERT-tiny with SwiGLU activation and no positional encoding. We exclude positional encoding hoping to improve length-extrapolation at inference [17] and because the aggregated tokens correspond to the same frame $c_k[:, i]$ and lack an inherent order beyond the original ordering of the S_k. By including this aggregation module, we also avoid post-hoc

aggregation techniques such as those used in [4]. The entire model, including the aggregation module, is trained end-to-end on our generated data.

3.4. Training

We train the model to predict the binary frame labels of the query on the generated dataset. We train for 10^6 steps with focal loss [18] and the AdamW optimizer [19] with weight decay of .01 for non-bias parameters. We use a cosine-annealing learning rate schedule with 10^5 steps warmup, a peak learning rate of 5×10^{-4}, and a batch size of 64. Some of our submissions were selected from mid-training checkpoints based on performance on the validation set. The audio encoder was frozen for the entire duration of training; unfreezing the audio encoder during training may improve our system's performance. Loss measured on our generated training data had plateaued by the end of training, indicating that further training with these settings would not improve performance further.

3.5. Few-shot learning

Few-shot learning is performed by presenting the trained model with the prompt of annotated support audio, together with unlabeled query audio. While no fine-tuning is required, the results can be influenced by the format of the prompt, as well as post-processing strategies.

3.5.1. Prompt Engineering. We sub-sample from the provided support audio to avoid extremely long prompt durations. To do so, we divide the support audio into chunks of duration D seconds, and then discard chunks without positive events until we have M chunks remaining, or until we cannot discard any more chunks. In our submission, we use $D = 8$ and $M = 5$; Initial experiments indicated that model outputs are not sensitive to small changes in these parameters.

After sub-sampling the support audio, we create 10-second support and query windows, using a 50% overlap for each. For each 10-second query window, we put all support windows, together with this query window, as a single prompt for the model. We do this for each query window, retaining the middle 50% of the model outputs for each query window, in order to avoid boundary artifacts.

3.5.2. Post-Processing. The output of our model is a series of logits, corresponding to detection probabilities at 25 Hz. To arrive at start and end time predictions, we accept as positive detections all frames with detection logit above a threshold $\tau \in \mathbb{R}$. We fill in gaps between detections, if the gap is less than $\min\{0.5 \cdot v_{min}, 1\}$ seconds, and then discard detected events of duration less than $\min\{0.5 \cdot v_{min}, 0.5\}$ seconds. Here v_{min} is the minimum duration of an event provided in the support set.

The threshold τ for detections can be adapted to each few-shot task (i.e., each full audio file in the DCASE validation set). To determine this threshold, we set a prior distribution $p_{thresh}(\tau) = N(0, 1)$ on the logit threshold τ. Next, we estimate the distribution of event durations p_{dur} in the entire audio file as $\hat{p}_{dur}(d) = N(\mu, \sigma)$, where μ and σ are, respectively, the sample mean and corrected standard deviation of the durations of events in the support set. For a fixed threshold τ', we obtain a set of detected events, whose durations form a set $\mathrm{durs}(\tau')$. We choose our final value of τ as the logit value that maximises the average likelihood of detection durations, weighted by p_{thresh}:

$$\tau = \mathrm{argmax}_{\tau'} \left(\frac{p_{thresh}(\tau')}{|\,\mathrm{durs}(\tau')|} \sum_{d \in \mathrm{durs}(\tau')} \hat{p}_{dur}(d) \right).$$

We found that this adaptive thresholding sometimes improved our results by a small margin (Table 3) over a fixed threshold of $\tau = 0$.

#	HB	ME	RD	PB	PB24	PW	All	All, $\tau = 0$
1	.64	.67	.51	.35	.51	.70	**.533**	.515
2	.65	.63	.50	.46	.58	.82	**.584**	.533
3	.66	.59	.47	.43	.54	.81	**.559**	.554
4	.61	.77	.48	.22	.40	.40	.412	**.444**

TABLE 3. SUBMISSION RESULTS (F1 SCORE) ON 2024 DCASE TASK 5 VALIDATION SET. RIGHTMOST COLUMN IS WITHOUT ADAPTIVE THRESHOLDING. SUBMISSION AND PER-DATASET SCORES (HB-PW) IN EACH ROW CORRESPOND TO THE THRESHOLDING METHOD IN BOLD.

4. DCASE Results and Conclusion

The evaluation metrics for our submitted models are provided in Table 3. Overall, our best system makes a small improvement to the challenge baseline. However, our submission introduced two new approaches for this task: in-context learning, and on-the-fly training data generation. We treat our submission as a proof-of-concept for this general approach. We expect that further work in this direction would yield further improvements.

Acknowledgments

The authors would like to thank the competition organizers for their efforts, as well as Maddie Cusimano, Masato Hagiwara, Sara Keen, Jen-Yu Liu, Marius Miron, and Gregory Yauney for valuable discussions and feedback during the course of the project.

References

[1] J. Liang, I. Nolasco, B. Ghani, H. Phan, E. Benetos, and D. Stowell, "Mind the domain gap: a systematic analysis on bioacoustic sound event detection," 2024.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui, "A survey on in-context learning," *arXiv preprint arXiv:2301.00234*, 2022.

[4] X. Wang, X. Zhang, Y. Cao, W. Wang, C. Shen, and T. Huang, "Seggpt: Towards segmenting everything in context," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1130–1140, 2023.

[5] S. Chen, S. Liu, L. Zhou, Y. Liu, X. Tan, J. Li, S. Zhao, Y. Qian, and F. Wei, "Vall-e 2: Neural codec language models are human parity zero-shot text to speech synthesizers," 2024.

[6] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[7] W.-P. Vellinga and R. Planqué, "The xeno-canto collection and its relation to sound recognition and classification.," in *CLEF (Working Notes)*, 2015.

[8] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 776–780, IEEE, 2017.

[9] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 1128–1132, IEEE, 2016.

[10] T. Denton, S. Wisdom, and J. R. Hershey, "Improving bird classification with unsupervised sound separation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 636–640, IEEE, 2022.

[11] S. Kahl, C. M. Wood, M. Eibl, and H. Klinck, "Birdnet: A deep learning solution for avian diversity monitoring," *Ecological Informatics*, vol. 61, p. 101236, 2021.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30, IEEE, 2017.

[13] X. Li, N. Shao, and X. Li, "Self-supervised audio teacher-student transformer for both clip-level and frame-level tasks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[14] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.

[15] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[16] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[17] A. Kazemnejad, I. Padhi, K. Natesan Ramamurthy, P. Das, and S. Reddy, "The impact of positional encoding on length generalization in transformers," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.