Prototypical Few-Shot Learning with HuBERT

Ilan Bacry
ENS Paris-Saclay
Paris, France
ilan.bacry20@gmail.com

Théo Basseras
ENS Paris-Saclay
Paris, France
ilan.bacry20@gmail.com

Abstract

Deep learning approaches for audio classification often rely on large labeled datasets and expensive supervised fine-tuning. In this work, we explore few-shot learning as a lightweight alternative for adapting speech foundation models to downstream audio tasks. Using HuBERT as a frozen feature extractor, we evaluate prototype-based few-shot methods across a diverse set of real-world and synthetic audio classification datasets. Our results highlight the trade-offs between adaptation cost and predictive performance and show that few-shot learning can provide an effective solution in low-resource settings.

1 Introduction

Self-supervised learning has become a key paradigm for learning transferable representations from large amounts of unlabeled data. In natural language processing, models such as BERT [2] have laid the foundations of what are now known as foundation models.

This paradigm has been extended to speech with HuBERT [4], which learns high-level speech representations without transcriptions and supports adaptation to a wide range of audio tasks.

Adapting foundation models typically relies on supervised fine-tuning, which can be costly in terms of labeled data and computation. In this work, we investigate few-shot learning as a lightweight alternative and focus on prototype-based methods following Prototypical Networks [7]. We conduct an experimental comparison between few-shot learning used for inference only, few shot learning used for episodic training and supervised fine-tuning across multiple audio classification tasks.

This report is organized as follows. We first introduce HuBERT as a speech foundation model and review the few-shot learning framework based on Prototypical Networks. We then describe the experimental protocol and present a comparative evaluation of three adaptation strategies: few-shot inference, episodic prototypical training and supervised fine-tuning. Finally, we discuss the results across multiple audio classification tasks and conclude with perspectives for future work. Our code is available at https://github.com/luckyman94/Few-shot-learning-with-HuBERT.

2 From BERT to HuBERT

2.1 BERT

BERT [2] is based on a stack of Transformer encoders trained with a masked prediction objective, where randomly masked tokens are predicted from their surrounding context. By leveraging both left and right contexts, this approach enables BERT to learn rich bidirectional representations that capture contextual and semantic information.

2.2 HuBERT

HuBERT [4] extends the masked prediction paradigm introduced by BERT to the speech domain. Instead of operating on discrete text tokens, HuBERT learns from raw audio by predicting masked latent representations derived from speech signals. This allows the model to acquire high-level and transferable speech representations without relying on transcriptions.

2.2.1 Architecture

As illustrated in Figure 1, HuBERT is composed of three main components. First, a **convolutional feature encoder** processes raw audio and extracts latent speech representations. These representations are then fed into a **Transformer encoder**, inspired by BERT, which models long-range temporal dependencies using a masked prediction objective. Finally, an **acoustic unit discovery system** produces discrete acoustic units through unsupervised clustering, which are used as prediction targets during training, enabling fully self-supervised learning without transcriptions.

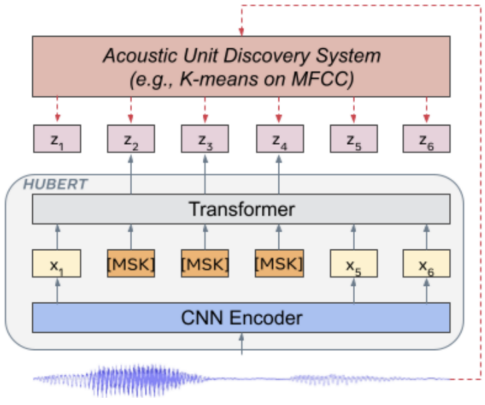

Figure 1: Overview of the HuBERT architecture.

3 Few-Shot Learning

3.1 Formulation

Few-shot learning aims to address classification problems in which only a limited number of labeled examples per class are available. In this setting, models must rapidly adapt to new tasks given minimal supervision.

In this work, we consider the standard k-shot, n-way episodic learning setting.

Definition We define an *episode* in the context of k-shot, n-way learning, where n denotes the number of distinct classes involved in the task. Each episode is composed of two disjoint subsets: a *support set* and a *query set*.

- **Support set.** For each of the n classes, k labeled examples are randomly sampled, resulting in a total of kn support samples.
- **Query set.** For each class, mk additional examples are sampled, ensuring that these samples are distinct from those used in the support set.

The objective of few shot learning is to classify the query samples using only the information provided by the support set.

In the following, we describe the prototypical few shot learning [7] strategy adopted in this work to perform few-shot learning with HuBERT representations.

3.2 Prototypical Few Shot Learning

We again consider the k-shot, n-way setting. Each episode is processed in two main phases :

- **Support phase.** We construct a support set containing n classes with k samples per class:

  $$(s_1^1, \ldots, s_k^1, s_1^2, \ldots, s_k^n).$$

  For each class j, the k samples are projected into a latent space using the projected HuBERT-based embedding function:

  $$l_i^j = \text{HuBERT}_{\text{projected}}(s_i^j), \quad i = 1, \ldots, k.$$

  The prototype for class j is then computed as the mean of its embeddings:

  $$r_j = \frac{1}{k} \sum_{i=1}^{k} l_i^j.$$

  Each prototype r_j represents its corresponding class in the latent space. Classification is then performed by assigning each query sample to the class whose prototype is closest in the embedding space.

- **Query phase.** We construct a query set containing the same n classes, with km samples per class:

  $$(s_1^1, \ldots, s_{km}^1, s_1^2, \ldots, s_{km}^n).$$

  Each query sample s_i^j is projected into the latent space:

  $$l_i^j = \text{HuBERT}_{\text{projected}}(s_i^j).$$

  For each query embedding l_i^j, we compute its squared Euclidean distance to all class prototypes:

  $$d_i^j = \left[\|r_1 - l_i^j\|^2, \ldots, \|r_n - l_i^j\|^2\right].$$

  A softmax over the negative distances yields a probability distribution over classes:

  $$p_i^j(c) = \frac{\exp\left(-\|r_c - l_i^j\|^2\right)}{\sum_{h=1}^{n} \exp\left(-\|r_h - l_i^j\|^2\right)}, \quad c = 1, \ldots, n.$$

  The predicted class is given by:

  $$\hat{y}_i^j = \arg \max_{c \in \{1,\ldots,n\}} p_i^j(c).$$

  The total loss over an episode is:

  $$\text{total\_loss} = \sum_{j=1}^{n} \sum_{i=1}^{km} \text{loss}_i^j.$$

  This loss is used to update the parameters of the projection network (a simple neural network on top of the HuBERT), while the HuBERT encoder itself remains frozen.

- **Backpropagation**

  For a query sample belonging to class j, the gradients with respect to the prototypes illustrate the learning dynamics:

  $$\frac{\partial \text{loss}_i^j}{\partial r_j} = \text{softmax}_i^j[j] - 1,$$

  which corresponds to a negative gradient, encouraging the prototype of the correct class to move closer to the query embedding.

  For any incorrect class $k \neq j$:

  $$\frac{\partial \text{loss}_i^j}{\partial r_k} = \text{softmax}_i^j[k],$$

  which pushes the incorrect prototypes away from the query embedding.

  The gradient of the distance with respect to the query embedding is:

  $$\frac{\partial d_i^j[s]}{\partial l_i^j} = 2(r_s - l_i^j).$$

  These gradients propagate back through the projection network, allowing its parameters to be updated via gradient descent.

4 Experimental Setup

4.1 Datasets

Experiments are conducted on a diverse set of audio classification datasets to evaluate few-shot learning under varying conditions. We consider two main categories.

First, we use real-world audio datasets from publicly available sources, primarily Kaggle, covering standard audio classification tasks. Depending on the dataset, balanced sub-sampling is applied to control the number of classes and samples per class and ensure fair few-shot evaluation.

Second, we include synthetic audio datasets designed to analyze few-shot learning in controlled settings and isolate specific factors affecting model behavior.

The datasets used in our experiments are detailed below.

4.1.1 Real-World Audio Classification Datasets

Speech Commands [9] A keyword recognition dataset of short isolated spoken words, used as a simple baseline task.[1]

CREMA-D [1] An emotional speech dataset where actors utter predefined sentences with different emotions.[2]

TIMIT [3] A phoneme-level speech dataset requiring fine-grained acoustic discrimination.[3]

Cats vs. Dogs vs. Birds A multiclass audio dataset of animal sounds (cats, dogs, and birds).[4]

Snoring Detection A binary dataset composed of snoring and non-snoring audio recordings.[5]

UrbanSound8K [6] A ten-class dataset of everyday urban sounds.[6]

4.1.2 Synthetic Datasets

Synthetic Audio Noise A synthetic audio classification dataset composed of multiple classes of sinusoidal signals corrupted by additive Gaussian noise at a fixed signal-to-noise ratio. Each class is associated with a distinct base frequency, and the dataset is balanced with fixed-length audio samples.

Synthetic Audio Harmonics A synthetic audio classification dataset where each class corresponds to a signal composed of a fundamental frequency and a varying number of harmonic components with decreasing amplitudes.

A summary of the main characteristics of all datasets used in our experiments is provided in Table 5.

4.2 Protocol and Training Parameters

4.2.1 Few-shot learning

Few-shot experiments are conducted on the audio classification datasets described above. Classification is performed at the episode level by sampling a small support set and a query set from a subset of classes. Unless stated otherwise, the following protocol is shared across all few-shot settings:

- **Task type:** Binary and multi-class audio classification.
- **Few-shot regimes:** 1-shot, 5-shot and 10-shot.
- **Evaluation metric:** Classification accuracy.
- **Distance metric:** Euclidean distance in the embedding space.

Few-shot benchmarking is performed by repeatedly sampling a fixed number of episodes for each dataset and each k-shot setting, with $k \in \{1, 5, 10\}$. For each episode, the support set contains k labeled examples per class, while the query set is composed of samples drawn from the remaining data. The number of classes involved in each episode depends on the dataset under study.

Final performance is reported as the mean accuracy and standard deviation across all sampled episodes, providing a robust estimate of performance under low-resource conditions.

Few-shot inference In this setting, the HuBERT encoder is kept frozen and no additional training is performed. We use a query set of 20 samples. Class prototypes are computed directly as the mean of the support embeddings in the HuBERT representation space. Query samples are classified by nearest-prototype assignment using Euclidean distance.

After evaluation, confusion matrices and t-SNE [8] visualizations are computed to further analyze class-wise performance and error patterns.

Prototypical training We extend the few-shot inference setting by introducing *episodic training* following the Prototypical Networks framework. The HuBERT encoder remains frozen, while a projection head is trained to learn a task-specific metric space for prototype-based classification.

For this setting, each dataset is first split at the *class level* into a training set and a test set, ensuring that evaluation is performed on previously unseen classes.

We use a query set of 10 samples for all datasets except UrbanSound8K, where the query set is reduced to 5 samples. During training, few-shot episodes are sampled exclusively from the training split. For each episode, class prototypes are computed from the projected support embeddings, and the projection head is optimized by minimizing a

[1] https://www.kaggle.com/datasets/nikhilkushwaha2529/speech-commands
[2] https://www.kaggle.com/datasets/ejlok1/cremad
[3] https://www.kaggle.com/datasets/nltkdata/timitcorpus
[4] https://www.kaggle.com/datasets/warcoder/cats-vs-dogs-vs-birds-audio-classification
[5] https://www.kaggle.com/datasets/tareqkhanemu/snoring
[6] https://www.kaggle.com/datasets/chrisfilo/urbansound8k

distance-based cross-entropy loss over the query set. Model parameters are updated using batched gradient updates for improved training stability.

At evaluation time, the trained projection head is used to perform few-shot classification on unseen classes sampled from the test split, following the same episodic protocol as in the inference-only setting.

4.2.2 Supervised fine-tuning

For three datasets : *Cats vs. Dogs vs. Birds*, *Snoring Detection*, and *UrbanSound8K* supervised fine-tuning of the HuBERT foundation model is performed to establish a reference point for comparison with few-shot learning.

- **Data splits:** Stratified train/validation/test splits ($70\%/15\%/15\%$).
- **Batch size:** 16 or 8, depending on the dataset.
- **Classification head:** Linear head applied to mean-pooled HuBERT representations.
- **Training epochs:** Between 8 and 10 epochs.
- **Loss function:** Cross-entropy loss.

Due to the large size of the HuBERT model, fine-tuning is performed using parameter-efficient LoRA [5] adapters. The LoRA configuration is fixed across experiments: *rank* = 4, α = 8, *dropout* = 0.2.

5 Results and Discussion

5.1 Few shot learning

5.1.1 Few-shot inference

Dataset	1-shot	3-shot	10-shot
Speech Commands	55.7	81.1	86.4
CREMA-D	26.6	33.1	36.8
TIMIT	42.0	62.2	72.3
Cats vs. Dogs vs. Birds	72.8	92.1	95.4
Snoring Detection	55.0	62.8	66.8
UrbanSound8K	20.7	22.7	22.1
Synthetic Noise (Low)	99.4	99.7	99.9
Synthetic Noise (Mid)	96.9	98.3	98.4
Synthetic Noise (High)	91.1	95.5	96.2
Synthetic Harmonics (Low)	99.4	99.7	99.9
Synthetic Harmonics (Mid)	72.2	70.6	69.8
Synthetic Harmonics (High)	1.0	1.0	1.0

Table 1: Few-shot accuracy averaged over 100 episodes for each k-shot setting. Synthetic datasets are evaluated at three levels of difficulty (Low, Mid, High).

Additional qualitative analyses, including confusion matrices and t-SNE visualizations for selected datasets, are reported in Appendix B.

For synthetic datasets, *Low*, *Mid* and *High* correspond to increasing task difficulty. For Synthetic Noise, difficulty is controlled by the signal-to-noise ratio (30, 10 and 0 dB), while for Synthetic Harmonics it is controlled by the number of harmonic components (2, 5 and 10).

Table 1 reports few-shot accuracy across datasets and k-shot settings. Across all datasets, performance consistently improves as the number of labeled examples per class increases from 1-shot to 10-shot, highlighting the strong dependence of few-shot learning on available supervision.

In the 1-shot regime, simpler and more controlled datasets such as *Speech Commands* and *Cats vs. Dogs vs. Birds* achieve high accuracy, indicating that HuBERT representations are sufficiently discriminative for single-example classification. In contrast, more complex datasets such as *CREMA-D* and *UrbanSound8K* exhibit lower performance due to higher acoustic variability and less separable classes.

Increasing the number of support examples generally improves performance by yielding more reliable prototypes, although gains are limited on challenging datasets such as *UrbanSound8K*. On synthetic datasets, few-shot learning performs very well in low-difficulty settings and degrades progressively as task difficulty increases, either due to stronger noise or increased class overlap.

Overall, these results confirm that few-shot performance strongly depends on dataset complexity, with more challenging real-world datasets exhibiting lower accuracy.

5.1.2 Prototypical training

Dataset	1-shot	5-shot	10-shot
Speech Commands	87.0	93.1	93.7
CREMA-D	58.2	66.1	70.0
TIMIT	56.5	83.0	80.5
Cats vs. Dogs vs. Birds	100.0	100.0	100.0
Snoring Detection	100.0	100.0	100.0
UrbanSound8K	34.0	36.3	34.7
Synthetic Noise (Low)	98.9	98.9	98.9
Synthetic Noise (Mid)	95.3	97.5	97.2
Synthetic Noise (High)	93.6	99.4	99.0
Synthetic Harmonics (Low)	33.3	28.8	47.9
Synthetic Harmonics (Mid)	39.0	47.6	18.7
Synthetic Harmonics (High)	100.0	100.0	100.0

Table 2: Few-shot accuracy averaged over 200 episodes for each k-shot setting using episodic prototypical training.

Table 2 reports few-shot classification performance when episodic prototypical training is applied on top of frozen HuBERT representations.

Compared to the inference-only setting, prototypical training leads to a substantial performance improvement on most real-world datasets. This effect is particularly pronounced in the low-shot regime, where learning a task-specific metric space significantly improves class separation.

On datasets such as *Speech Commands*, *CREMA-D* and *TIMIT*, episodic training consistently improves accuracy across all k-shot settings, with especially large gains observed in the 1-shot regime. This highlights the benefit of adapting the embedding space to the target task when only a few labeled examples are available.

For simpler datasets such as *Cats vs. Dogs vs. Birds* and *Snoring Detection*, prototypical training rapidly reaches perfect performance, even in the 1-shot setting. This suggests that the learned metric space is sufficient to fully separate classes when intra-class variability is limited.

In contrast, performance gains remain more limited on challenging datasets such as *UrbanSound8K*. While episodic training improves accuracy compared to few-shot inference, high acoustic diversity and complex class boundaries still limit overall performance.

On synthetic datasets, prototypical training preserves strong robustness to additive noise across all difficulty levels. However, performance on the Synthetic Harmonics dataset varies with task difficulty, indicating that metric learning alone may not fully resolve fine-grained frequency ambiguities in highly structured signals.

Overall, these results demonstrate that episodic prototypical training provides a significant advantage over inference-only few-shot learning, particularly in low-data regimes, while still struggling on highly complex real-world audio classification tasks.

5.1.3 Comparison with Fine-Tuning

We now compare few-shot learning approaches with supervised fine-tuning of HuBERT on three datasets of increasing complexity: *Cats vs. Dogs vs. Birds*, *Snoring Detection*, and *UrbanSound8K*.

Dataset	1-shot	5-shot	10-shot	FT
Cats vs. Dogs vs. Birds	72.8	92.1	95.4	97.8
Snoring	55.0	62.8	66.8	95.3
UrbanSound8K	20.7	22.7	22.1	14.6

Table 3: Few-shot accuracy using episodic prototypical training ($k \in \{1, 5, 10\}$), averaged over 100 episodes. Fine-tuning accuracy is reported after supervised training.

Prototypical training vs Fine-Tuning Table 3 compares episodic prototypical training with supervised fine-tuning. On the simple *Cats vs. Dogs vs. Birds* task, prototypical training rapidly approaches fine-tuning performance as the number of support examples increases, indicating that a learned metric space is sufficient to separate classes when intra-class variability is limited.

On *Snoring Detection*, fine-tuning provides a substantial improvement over prototypical training, reflecting the benefit of task-specific adaptation in the presence of background noise and higher acoustic variability.

In contrast, on the more challenging *UrbanSound8K* dataset, prototypical training slightly outperforms fine-tuning. This suggests that adapting a large foundation model with limited labeled data can be difficult in highly diverse multi-class settings, whereas metric-based few-shot learning remains more robust.

Dataset	1-shot	5-shot	10-shot	FT
Cats vs. Dogs vs. Birds	72.8	92.1	95.4	97.8
Snoring	55.0	62.8	66.8	95.3
UrbanSound8K	20.7	22.7	22.1	14.6

Table 4: Comparison between few-shot inference (without training) and supervised fine-tuning. Few-shot inference accuracy is averaged over 100 episodes.

Few-shot inference vs Fine-Tuning Table 4 highlights the gap between inference-only few-shot learning and supervised fine-tuning. While fine-tuning consistently achieves the highest accuracy on simpler tasks such as *Cats vs. Dogs vs. Birds* and *Snoring Detection*, few-shot inference remains competitive despite relying on a frozen encoder and a very limited number of labeled examples.

On *UrbanSound8K*, few-shot inference slightly outperforms fine-tuning, suggesting that naive supervised adaptation of a large model may be ineffective when labeled data is scarce and class variability is high.

Overall, these results illustrate a clear trade-off between predictive accuracy and adaptation cost. While fine-tuning can achieve superior performance when sufficient labeled data is available, episodic prototypical training and even inference-only few-shot learning provide robust and computationally efficient alternatives in low-resource and class-generalization settings.

6 Conclusion

In this work, we investigated few-shot learning as a lightweight adaptation strategy for speech foundation models, using HuBERT as a frozen feature extractor. Through experiments on a diverse set of real-world and synthetic audio classification datasets, we showed that few-shot methods can achieve strong performance with very limited labeled data, particularly on simple and moderately complex tasks.

Our results highlight a clear dependence on dataset complexity. While accuracy generally improves with additional support examples, gains tend to saturate on simpler datasets and remain limited on more challenging ones. Comparisons with supervised fine-tuning further reveal a trade-off between predictive performance and computational cost, with few-shot learning offering a robust and efficient alternative in low-resource settings.

These findings open promising perspectives for future work, including hybrid approaches combining episodic metric learning with parameter-efficient fine-tuning, as well as extensions to other speech foundation models and downstream audio tasks.

A Datasets

Dataset	Classes	Samples
Speech Commands	7	1000
CREMA-D	6	7442
TIMIT	10	1000
Cats vs. Dogs vs. Birds	3	610
Snoring Detection	2	1000
UrbanSound8K	10	8732
Synthetic Audio Noise	8	800
Synthetic Audio Harmonics	8	800

Table 5: Summary of datasets used in the experiments.

B Few-Shot Learning

B.1 Read Datasets

B.1.1 Speech Commands

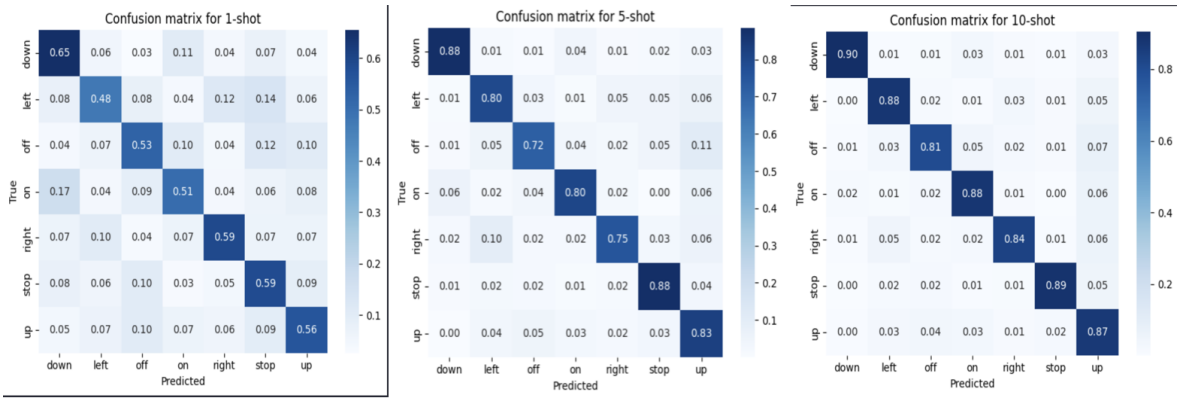

Figure 2: Confusion matrix for Speech Commands

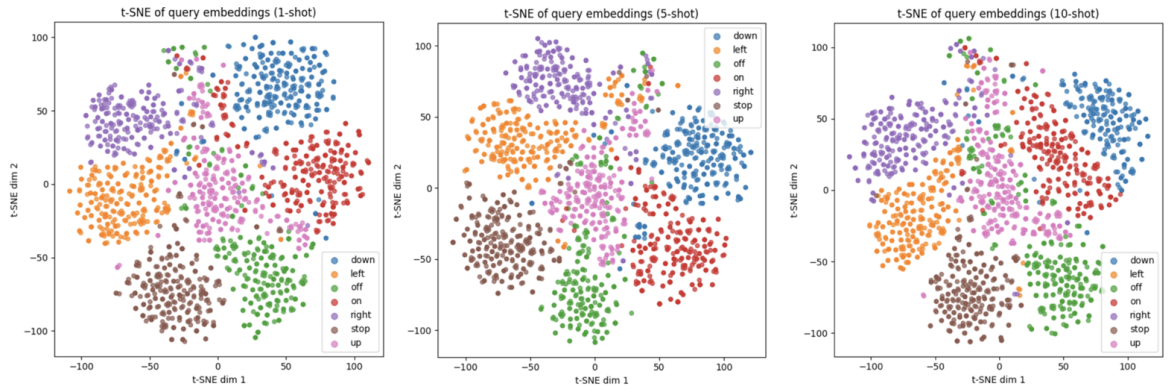

Figure 3: t-SNE for Speech Commands

B.1.2 CREMA-D

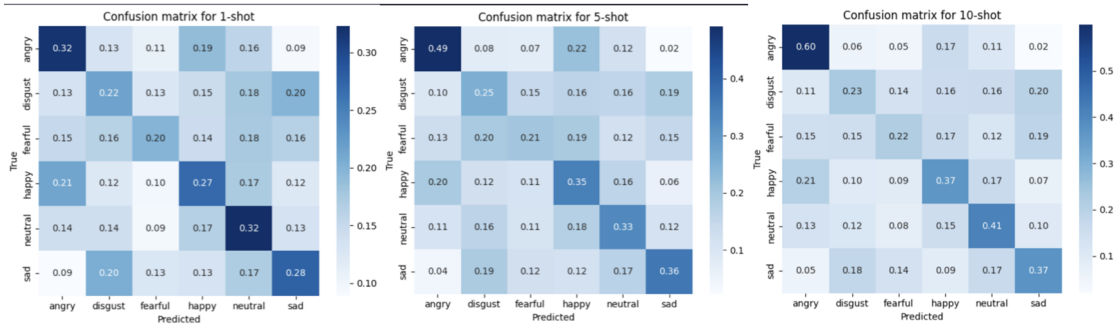

Figure 4: Confusion matrix for CREMA-D

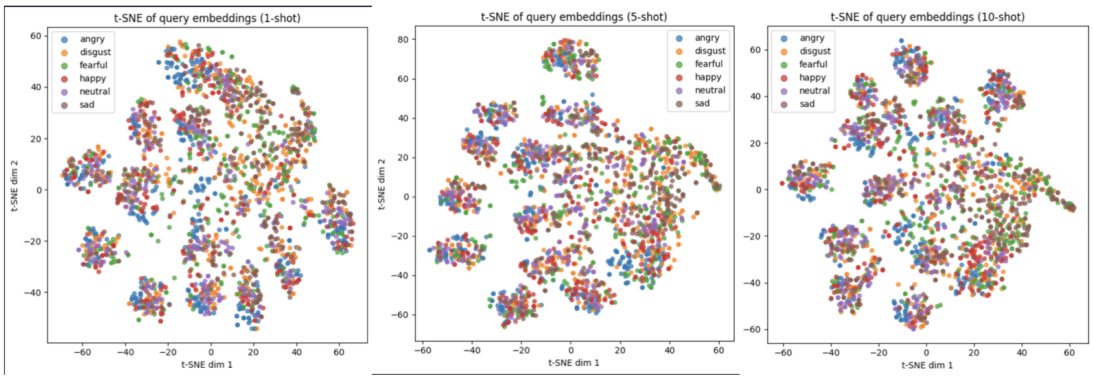

Figure 5: t-SNE for CREMA-D

B.1.3 TIMIT

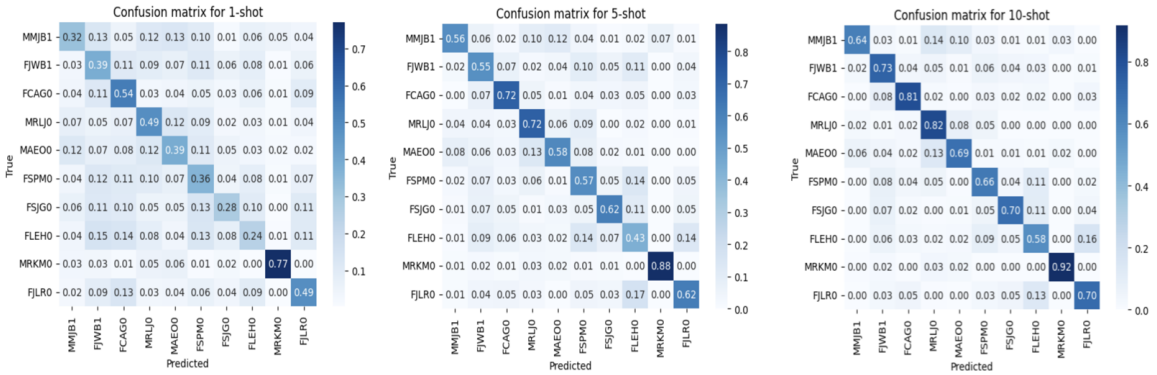

Figure 6: Confusion matrix for TIMIT

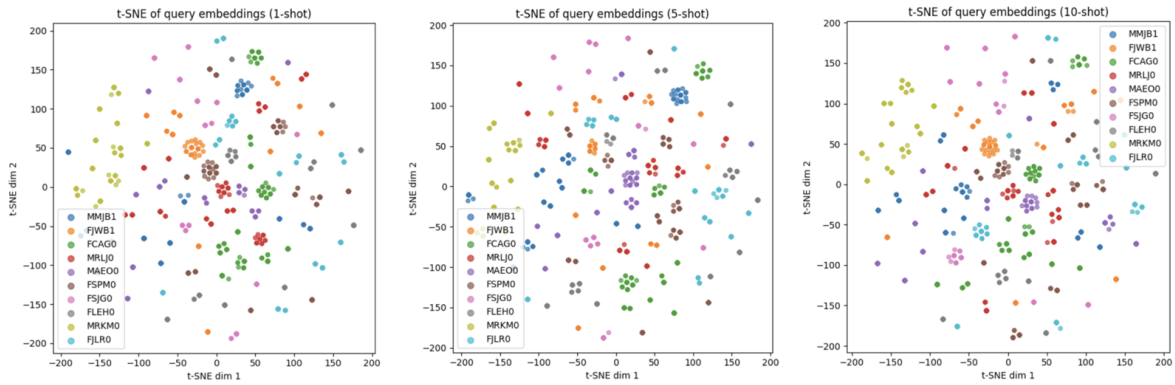

Figure 7: t-SNE for TIMIT

B.1.4 Cats vs. Dogs vs. Birds

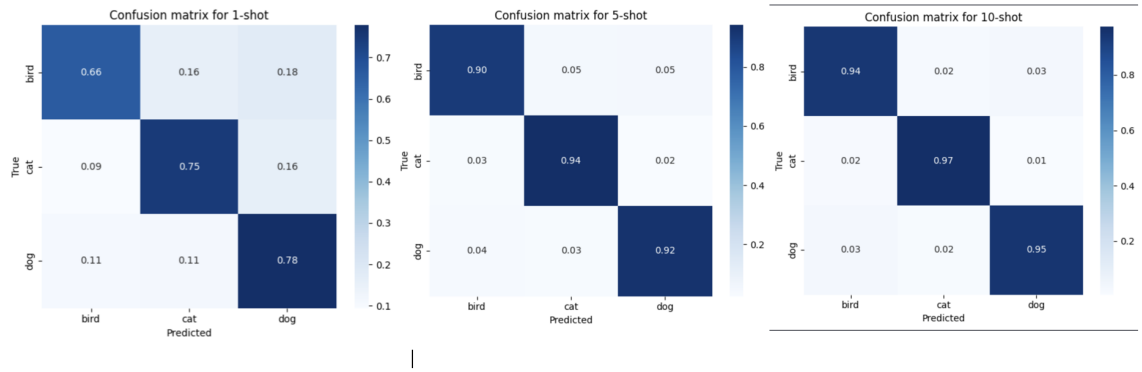

Figure 8: Confusion matrix for Cats vs. Dogs vs. Birds

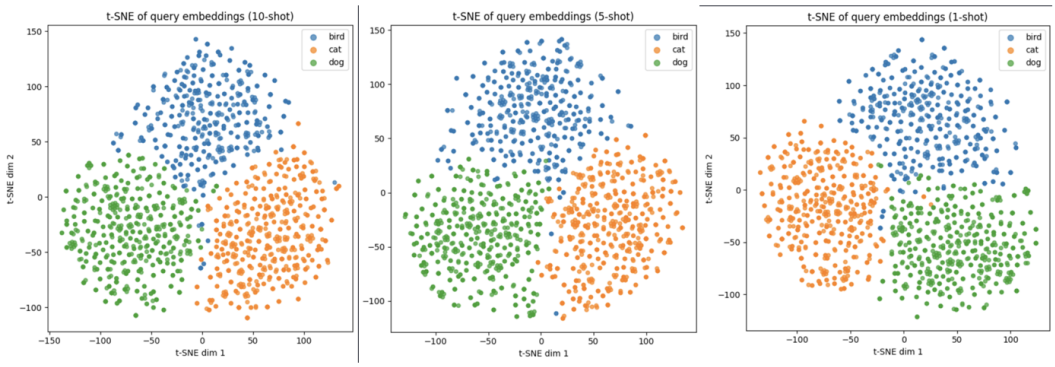

Figure 9: t-SNE for Cats vs. Dogs vs. Birds

B.1.5 Snoring Detection

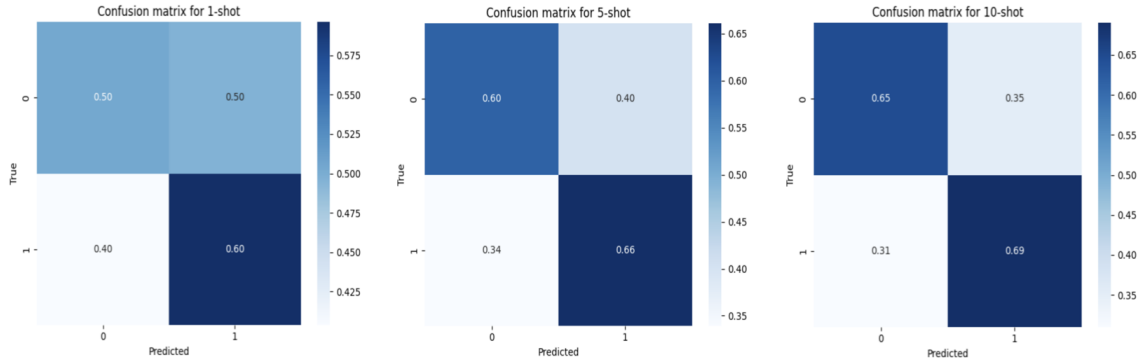

Figure 10: Confusion matrix for Snoring

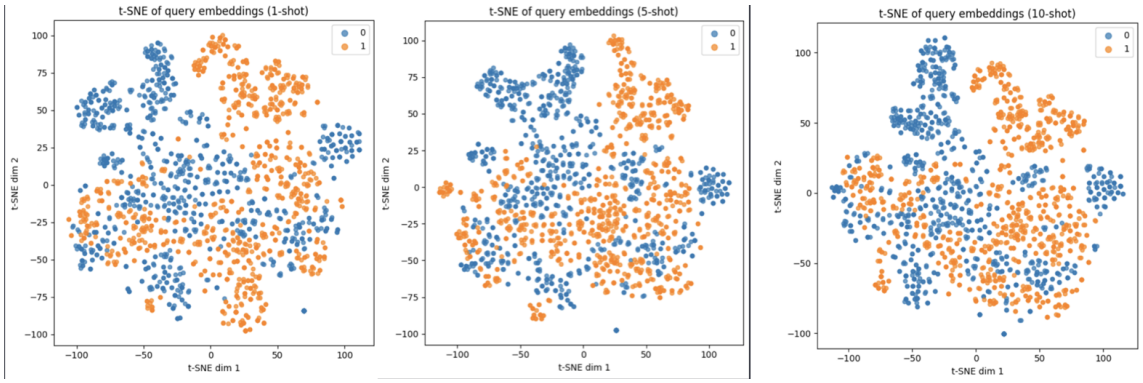

Figure 11: t-SNE for Snoring

B.1.6 UrbanSound8K

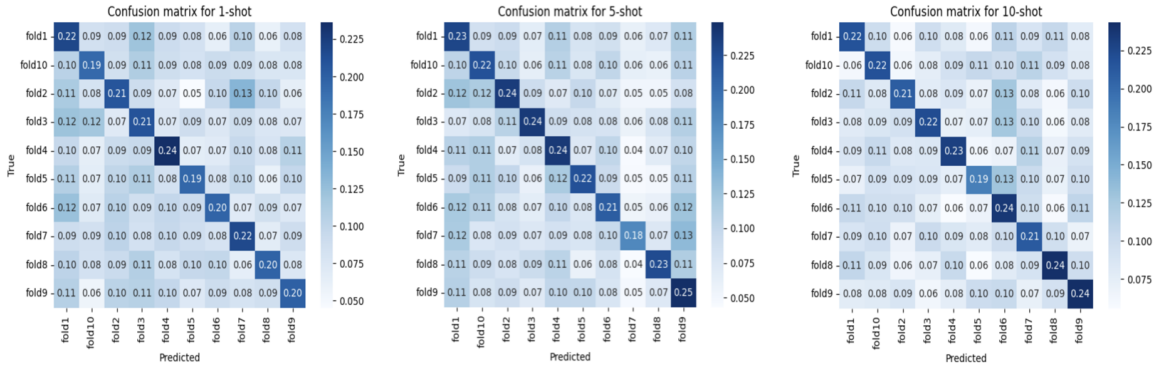

Figure 12: Confusion matrix for UrbanSound8K

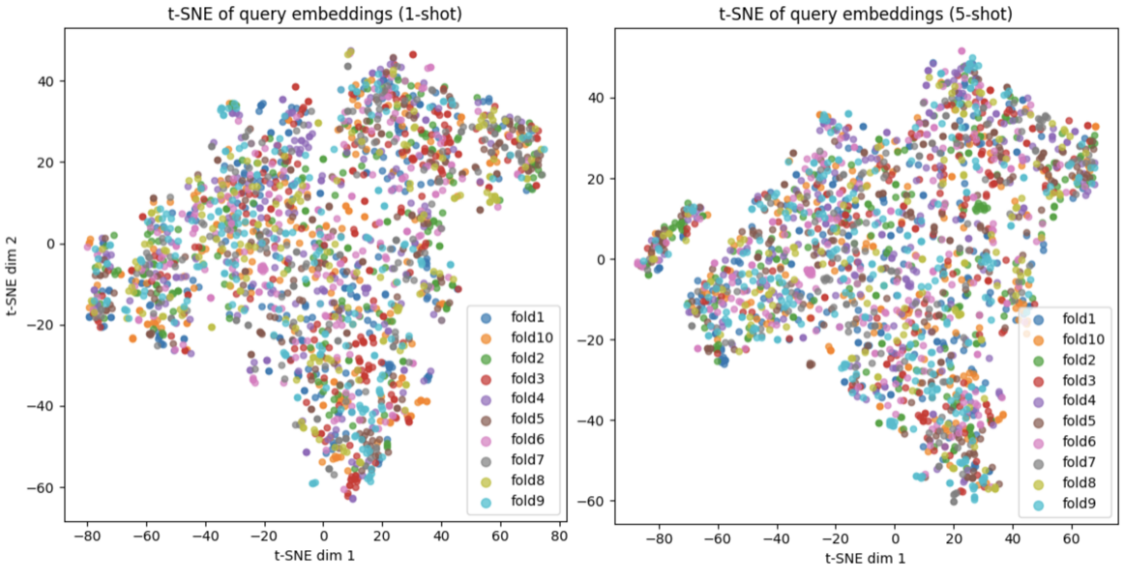

Figure 13: t-SNE for UrbanSound8K

B.2 Synthetic Datasets

B.2.1 Audio Noise Dataset (Low)

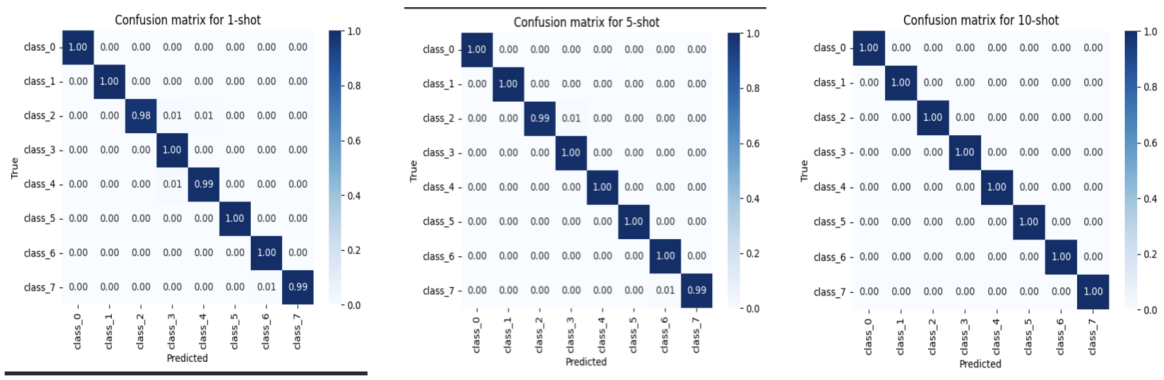

Figure 14: Confusion matrix for Synthetic Audio Noise (Low).

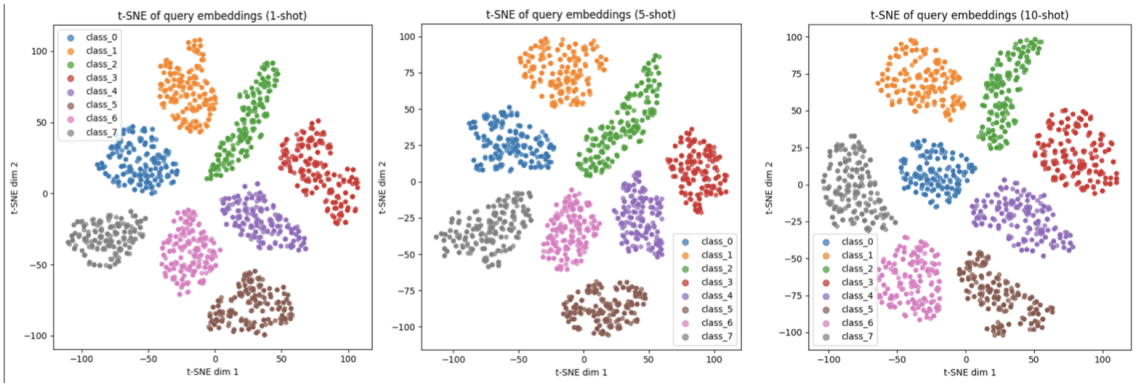

Figure 15: t-SNE visualization for Synthetic Audio Noise (Low).

B.2.2 Audio Noise Dataset (Mid)

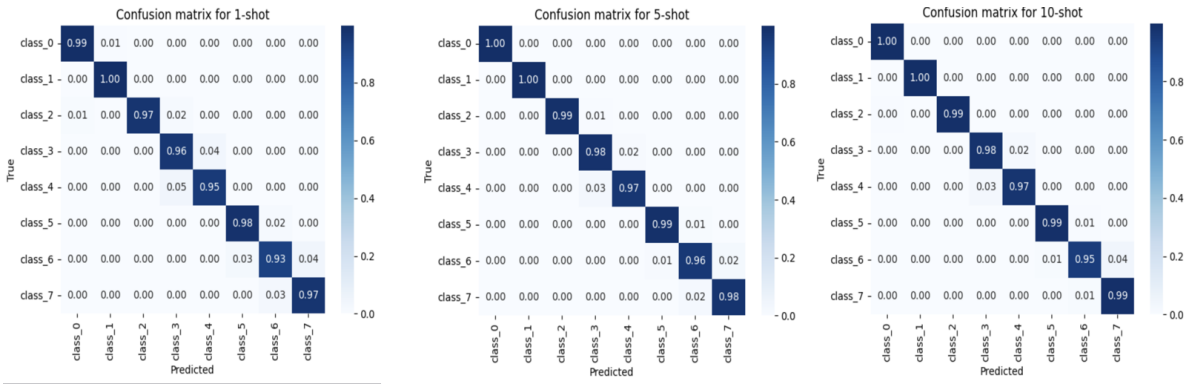

Figure 16: Confusion matrix for Synthetic Audio Noise (Mid).

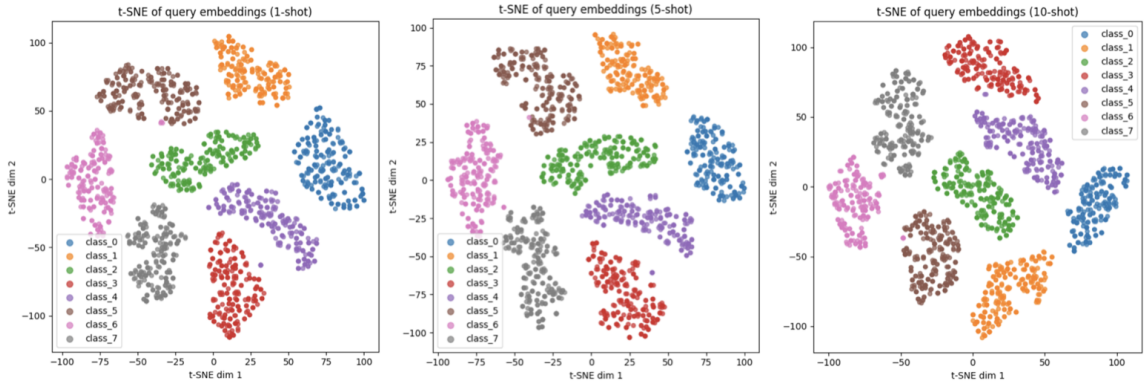

Figure 17: t-SNE visualization for Synthetic Audio Noise (Mid).

B.2.3 Audio Noise Dataset (High)

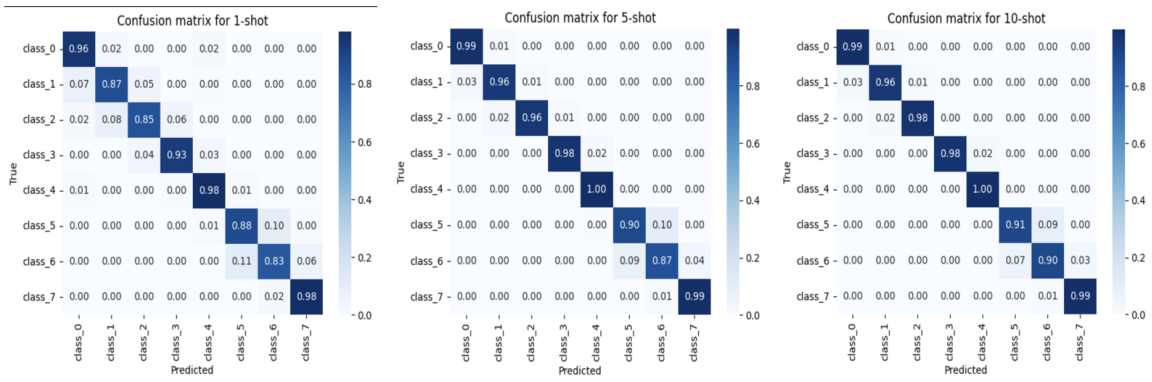

Figure 18: Confusion matrix for Synthetic Audio Noise (High).

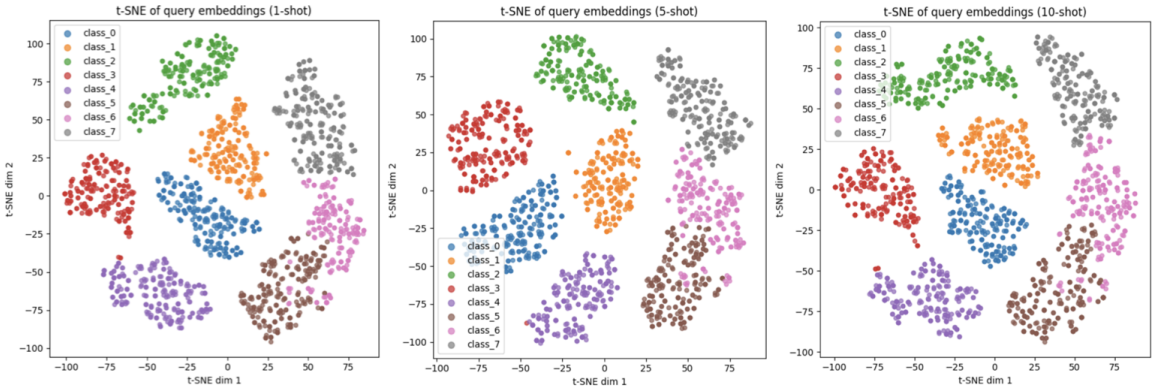

Figure 19: t-SNE visualization for Synthetic Audio Noise (High).

B.2.4 Audio Harmonics Dataset (Low)

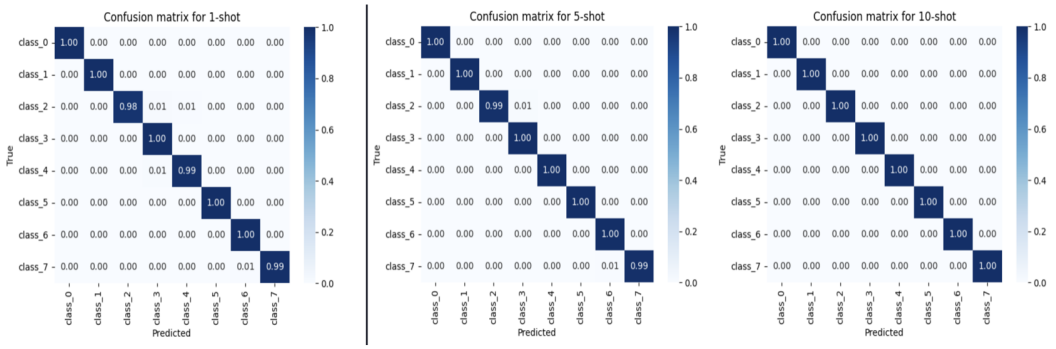

Figure 20: Confusion matrix for Synthetic Audio Harmonics (Low).

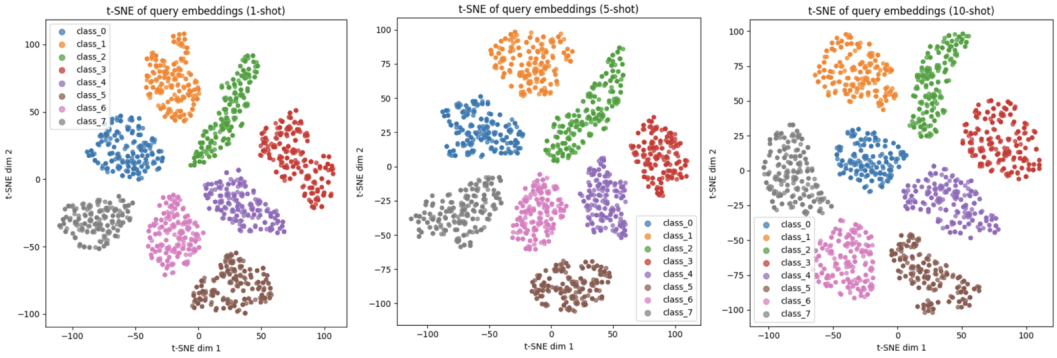

Figure 21: t-SNE visualization for Synthetic Audio Harmonics (Low).

B.2.5 Audio Harmonics Dataset (Mid)

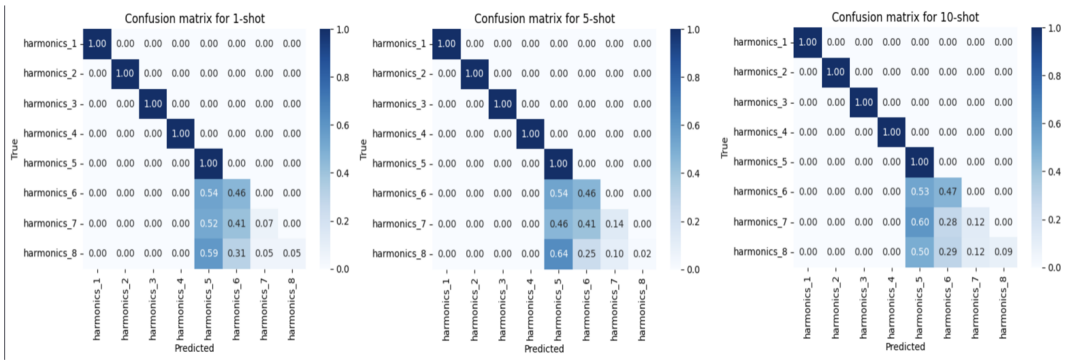

Figure 22: Confusion matrix for Synthetic Audio Harmonics (Mid).

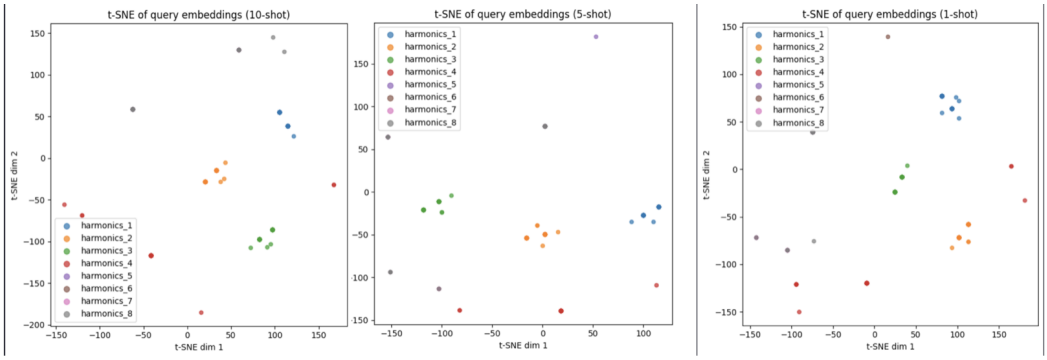

Figure 23: t-SNE visualization for Synthetic Audio Harmonics (Mid).

B.2.6 Audio Harmonics Dataset (High)

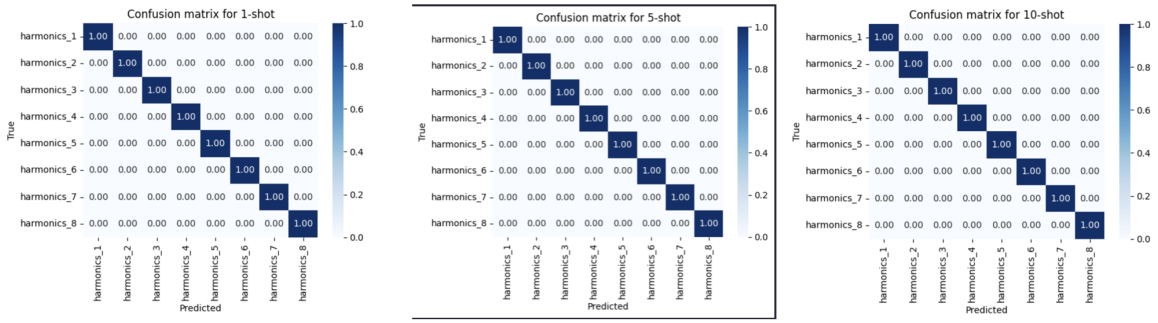

Figure 24: Confusion matrix for Synthetic Audio Harmonics (High).

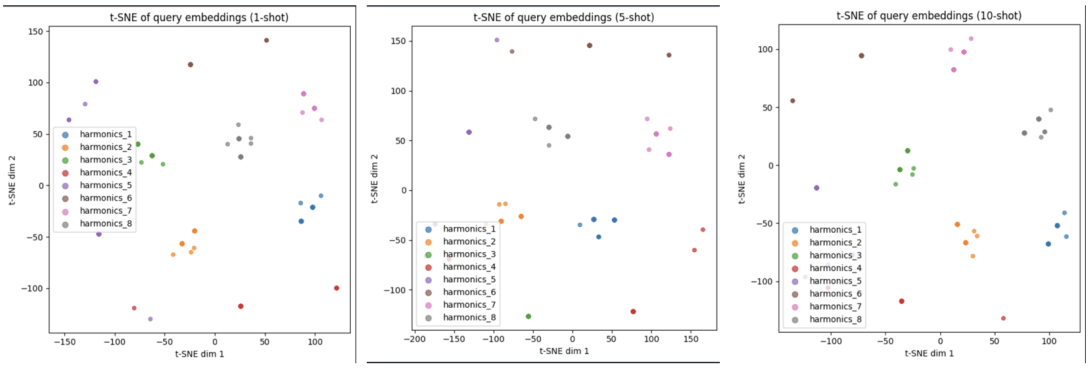

Figure 25: t-SNE visualization for Synthetic Audio Harmonics (High).

C Fine tuning

C.1 Real Datasets

C.1.1 Cats vs. Dogs vs. Birds

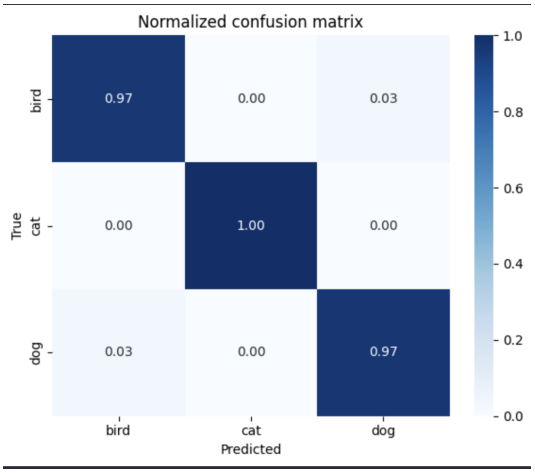

Figure 26: Confusion matrix for Cats vs. Dogs vs. Birds

C.1.2 Snoring Detection

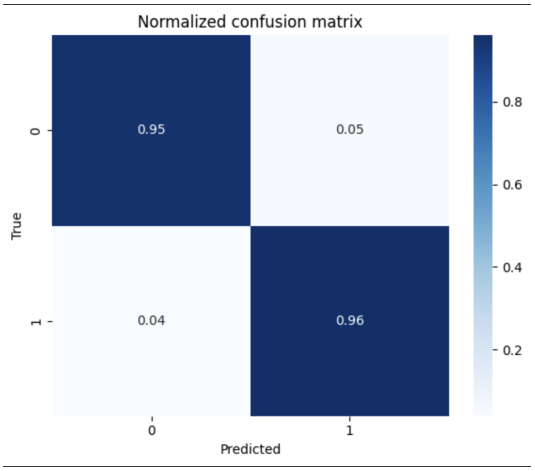

Figure 27: Confusion matrix for Snoring Detection

C.1.3 UrbanSound8K

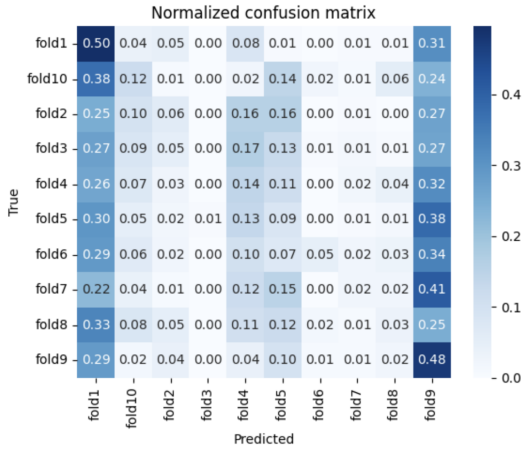

Figure 28: Confusion matrix for UrbanSound8K

References

[1] Houwei Cao, David G. Cooper, Michael K. Keutmann, et al. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE Transactions on Affective Computing*, 2014.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[3] John S. Garofolo, Lori F. Lamel, William M. Fisher, et al. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, 1993.

[4] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2021.

[5] Edward J. Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[6] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014.

[7] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[9] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.