Exploring On-Device Learning Using Few Shots for Audio Classification

Jagmohan Chauhan
University of Southampton
*University of Cambridge**

Young D. Kwon
University of Cambridge

Cecilia Mascolo
University of Cambridge

***Abstract*—Few shot learning (FSL) improves the generalization of neural network classifiers to unseen classes and tasks using small annotated samples of data. Recently, there have been attempts to apply few shot learning in the audio domain for various applications. However, the focus has been mainly on accuracy. Here, we take a holistic view and investigate various system aspects such as latency, storage and memory requirements of few shot learning methods in addition to improving the accuracy with very deep learning models for the tasks of audio classification. To this end, we not only compare the performance of different few shot learning methods but also, for the first time, design an end-to-end framework for smartphones and wearables which can run such methods completely on-device. Our results indicate the need to collect large datasets with more classes as we show much higher gains can be obtained with very deep learning models on big datasets. Surprisingly, metric-based methods such as ProtoTypical Networks can be realized practically on-device and quantization helps further (50%) in reducing the resource requirements, while having no impact on accuracy for the audio classification tasks.**

***Index Terms*—Few Shot Learning, Acoustic Event Classification, Keyword Spotting, On-Device Learning, Performance**

I. Introduction

Few shot learning is rapidly emerging as a powerful technique to create more generalizable AI agents which can adapt to new and unseen tasks using a limited set of annotated data. This resembles the human ability to learn novel concepts from few examples. Typically, these methods are composed of an embedding model that maps the input domain into a feature space and a base learner that maps the feature space to task variables. The objective is to learn an embedding model such that the base learner generalizes well across tasks. Gradient-based methods [1], [2] use gradient descent to adapt the embedding model parameters (e.g., all layers of a deep network) given training examples. Metric based methods [3], [4] learn a distance-based prediction rule over the embeddings.

Few shot learning has achieved good results in the fields of computer vision [1], [3], [5]–[8] and natural language processing [9], [10]. At the same time, the number of audio based applications ranging from understanding the environment through sounds in daily applications, such as surveillance, smart cities, and industry [11], [12] to conversational agents and smart speaking assistants [13] has soared. The potential of using few shot learning in audio have been investigated in some studies [14]–[17]. Chou et al. [14] proposed attention mechanisms which can be combined with metric based few shot learning methods for detecting acoustic events. The closest work to ours by Pons et. al. [17] compared transfer learning approaches with Prototypical Networks for acoustic event classification (AEC) and audio scene classification tasks. They tried from one to 100 samples and found both the approaches to obtain comparable performance.

Despite the aforementioned efforts, there still remain a few significant limitations in existing work which merits further research. Firstly, these works analyze the performance on datasets with small number of classes e.g., UrbanSounds [12] and TUT [11]. Also, they focus only on accuracy measures and forego other measures: latency and memory requirements of the few shot learning methods. To this end, our work systematically studies the applicability of few shot learning regimes on two popular audio sensing tasks in the context of accuracy, latency, storage and memory requirements: acoustic event classification (AEC) and keyword spotting (KWS) with a focus on on-device learning. We used ESC [18] for AEC and Google Speech Command [19] dataset for KWS. We also developed a framework for Android which can perform metric based few shot learning methods (adaptation phase) completely on-device by solving memory issues. This marks an important step in enabling on-device learning with few shots and personalization of deep learning models for audio classification tasks further ensuring data privacy.

Our results reveal that metric-based methods are more generalizable and scalable than gradient based methods for audio classification tasks (similar to [20]). Also, the generalizability degrades with increasing number of classes. But, in stark contrast to the previous works, we show that more complex deep learning models using metric-based learning methods can achieve significant gains (up to 9%) in accuracy on bigger datasets. Finally, we were able to execute ProtoTypical Network on two platforms: a smartphone and a smartwatch indicating that few shot learning can be practically realized on-device for the first time. The adaptation latency to unseen classes with 5-shots is small: 5.5–20 seconds on the smartphone. While the latency ranges between 34.5–190 seconds on a smartwatch.

*Author did the work when he was a Postdoc at the University of Cambridge.

II. Methodology

Methods: Our end goal is to obtain a model that can be trained using few shot learning algorithms to perform reasonably accurate classification of new audio sounds (acoustic events and hot keywords) provided a few labeled examples (1–5). A few shot learning setup comprises of three phases: meta training, meta validation, and meta testing and each phase have their own dataset (set) and disjoint classes. Meta training consists of multiple training epochs. In each epoch, a training episode is formed by randomly selecting N classes from the training set. Then, for each selected class, K and Q number of disjointed samples are selected to build a support and query set respectively. The training objective is to minimize prediction loss of the samples in Q conditioned on K. Essentially, K are the shots (labeled samples). While Q are the samples for which label is to be predicted. Therefore, in each training episode, the model is learning to solve N-way K-shot classification task. Each task contains N * (K+Q) samples. Each episode can have many classification tasks (meta batch size).

After each training epoch, meta validation phase is executed on the meta validation set. This is done for hyper-parameter tuning and model selection. The model selection criteria can be based on validation losses (minimum) or accuracy (maximum). The validation process remains similar to meta training. By training on a large collection of episodes, each consisting of a different set of N classes, and validation on unseen classes, the model learns how to learn from limited labeled data and obtain a generalizable discriminative ability. Once a few-shot model is trained and validated, it is applied to classify unseen sound events and hot keywords (Q) at meta testing phase, given a few labeled examples (K) of the target sounds. We assess four few shot learning methods: Prototypical Networks [3], MetaOptNet [4], MAML [1], and ANIL [2]. The first two are metric-based learning, while the last two are gradient based learning approaches.

Different few shot learning methods differ in how they make predictions conditioned on support set. ProtoTypical Networks [3] and MetaOptNet [4] depends on distance-based rules to learn and classify. Provided a support set of N × K labeled samples, a metric based method tries to learn embedding (where points cluster around a single prototype representation) for each class provided their support set. The learning is done by performing a nonlinear mapping of the input into an embedding space utilizing a deep neural network to get the embeddings. The mean of embeddings of a class support set is considered its (class) prototype. At the prediction time, distance is measured between the query embedding and the class mean (prototype) derived from support embeddings to perform the classification. The distance can be measured using a fixed or learned metric function. In ProtoTypical Networks squared Euclidean distance is used. We choose ProtoTypical Networks as it is the most popular metric learning method. Similar to ProtoTypical Networks, MetaOptNet learns a classifier in embedding (feature) space. But, instead of nearest neighbour it utilizes a linear SVM classifier to get the embeddings. Although computationally more expensive than ProtoTypical Networks, MetaOptNet currently obtain (state of the art) performances on image classification.

Model Agnostic Meta Learning (MAML) [1] is an initialization based few shot learning method, where initial model parameters (base model) are adapted using a few updates of gradient descent (adaptation steps) with the help of a support set. This is done by calculating the loss i.e. gradient computation between support and query sets. Note that when the query set samples are predicted by the adapted model in the meta-training phase, the loss of the query set is used to update the base model, and not the adapted model. This process is also named as outer loop update. Whereas, the process of adaptation is called inner loop update. The learning rates for both inner and outer loop updates can be tuned. It is the base model (generated from outer loop process) which is generalizable and is used at meta testing phase. MAML require the computation of second order gradients which are computationally expensive. So, we used first-order approximations of MAML to fasten the process. Almost No Inner Loop (ANIL) [2] is a significant simplification of MAML. It removes the inner loop updates for all but the head (final layer) of a neural network during meta training and meta testing phase as opposed to MAML and performs similar to MAML on standard few-shot classification image datasets. The success and recency of ANIL while providing computational benefits over MAML compelled us to use it.

Datasets: For the acoustic event classification task, we chose ESC-50 dataset [18] as it has 50 classes which allows us to perform n way classification at different levels. This dataset has 2000 audio samples with 40 samples per class which are categorized in different categories: animals, natural soundscapes and water sounds, human and non-speech sounds, interior/domestic sounds, and exterior/urban noises. Each sample is 5-second-long sampled at 44.1 kHz and mono channel. To create the data required for our neural network models, we followed standard practices [17] and created an input of size 128 * 216, where 128 represents the log-mel spectrogram and 216 represents the number of frames (STFT parameters: window size=hop size=1024).

For the KWS task we chose Speech Commands dataset [19]. It contains 105,829 samples of 35 hot key words (classes). Each sample is sampled at 16 kHz with mono channel. We discarded samples more than one second in duration to end up using 95,394 samples. Similar to ESC-50, we created an input of size 128 * 126, where 128 represents the log-mel spectrogram and 126 represents the number of frames (STFT parameters: window size=512, hop size=128).

Experimental Setup: For ESC-50, data from 32, 6 and 12 classes form the meta training, meta validation, and meta test set respectively. Note that the classes are non-overlapping for the three sets. For KWS, the meta training, meta validation, and meta test set composed of 20, 6, and 9 classes respectively. We use a 4-layered convolutional neural network (CNN). Each layer is composed of 3 * 3 convolution, ReLU, Batch Norm and 3 * 3 max-pooling layers. We also tried a ResNet-12 model as specified by Lee et al. [4].

We experimented with 1-shot and 5-shot setting [1], [4]. To test the scalability of the few shot learning methods, we set the number of train and test ways to be 2, 4, and 6. We also tried cases where number of train ways can be higher than test ways such as 4 (train) and 2 (test). The query set is composed of 15 samples for testing and 5 for training per way. We randomly sample 5000 training tasks, 1000 validation tasks and 600 test tasks for ESC-50. Due to the large size of KWS dataset, we sampled 10000, 2000 and 600 tasks for training, validation and testing respectively. Different learning rates of 0.01, 0.001 and 0.0001 were used for base learning. Meta learning rate was kept the same as base learning rate. Note that the settings for validation setup are kept same as the test setup. We used default values for specific hyperparameters related to MetaOptNet such as max iterations for SVM (3), dropout rate for residual layers (0.1), and dropblock size (5) as mentioned in the original paper [4]. For MAML, we varied the number of adaptation steps as 1, 3, and 5. For ProtoTypical Networks and MetaOptNet embedding sizes of 1600 and 16000 were inspected.

Meta training was performed for 40 epochs with 100 steps per epoch. At each step a meta batch size of 8 (i.e., 8 tasks) were used to do training. At the end of each epoch, validation was done by calculating validation loss for 100 randomly chosen tasks from the validation tasks set. If the validation loss does not decrease for 10 subsequent epochs, then the training is terminated (early stopping). The neural network model is saved otherwise. For evaluation, we measure accuracy on 600 test tasks with the best performing model selected from the validation phase. Each experiment was repeated three times.

III. Results

A. Generalization and Scalability

Accuracy of the four few shot learning methods (best results obtained with five shots) is shown in Figure 1. The classification ability of all the methods decreases with increasing number of ways: n as more new classes have to be classified. The accuracy varied between 91% and 73% and 90% and 78% for ESC and KWS dataset, respectively. These performances are reasonable as the number of samples required to learn the new classes were only five and shows that few shot learning is effective at learning new classes for audio classification tasks. Our results also show that n should not be fixed (at five), as done in previous work to assess the scalability of few shot learning methods with increasing number of classes during the testing phase.

Simple metric based few shot learning methods consistently achieved higher accuracy than more complex gradient based methods and scale better with increasing n. Gradient based methods such as MAML and ANIL performed poorly with higher n. This is due to the numerical stability issue that prevents these two methods from converging [8]. This also resulted in a very high variance between the runs for these two methods. We observed almost negligible variance with metric based few shot learning methods.

B. Impact of Architecture

In this section, we explore if deeper neural network architecture can help increase the accuracy. As metric-based methods perform better than gradient based approaches we only analyze ProtoTypical Network and MetaOptNet in this section. The results are shown in Table I for KWS. In all the cases, the accuracy improved (3.5% to 9%) and, importantly, this improvement gets higher with the number of ways n. This result is in contrast to all the previous works [14], [17] which found that deeper neural networks have no advantage over shallower nets for audio classification tasks in the context of few shot learning. This is mainly because previous works experimented with smaller datasets such as ESC and TUT [11]. While in our case, KWS has nearly 100,000 data points.

TABLE I: Average Accuracy of metric-based few shot learning methods with changing neural network architectures on KWS dataset. CNN-4 represents a CNN network with four layers. ResNet-12 represents a Residual Network with 12 layers.

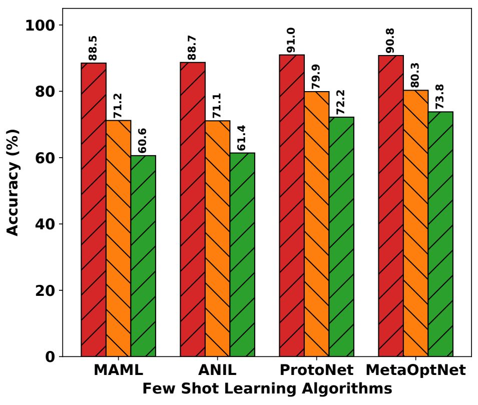
Architecture / FSL Method	**Two**	**Four**	**Six**
CNN-4 / ProtoTypical Network	89.5	82.3	77.9
CNN-4 / MetaOptNet	89.3	80.5	75.62
ResNet-12 / ProtoTypical Network	92.73	88.1	**85.4**
ResNet-12 / MetaOptNet	**94.8**	**89.1**	83.75

C. Costs

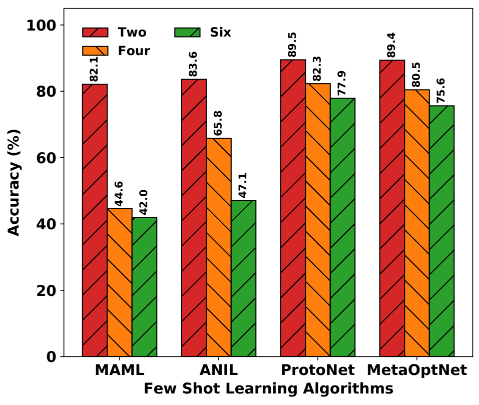
Previous studies have focused solely on the accuracy of the few shot learning methods. Departing from this, we tried to measure other costs associated with these methods such as storage and computational latency. In terms of storage, we find that few shot learning methods are independent of the model size (neural network parameters such as weights and biases). The storage required to store CNN4 and ResNet-12 based models is 486 KB and 50.2 MB, respectively. This is a small requirement given that modern devices have a large storage. The size of the models is calculated using Pytorch.

To assess the algorithmic complexity of each method, we measure their latency during testing phase on a GPU based machine: NVIDIA Quadro RTX 8000. Note that 600 tasks were evaluated. Latency is measured by the time it takes to load, execute and get the classification results from the deep learning model for audio classification task. MAML is the most expensive method. The latency varied between 18–120 and 12–84 seconds for ESC and KWS dataset, respectively. The latency increases with more adaptation steps: (1–5) and the number of ways, n. The smaller latency for KWS is attributed to the smaller input size compared to ESC. ANIL's execution time varied between 18–46 and 12–24 seconds for the two datasets. However, unlike MAML the adaptation steps did not impact the latency due to the "Almost No Inner Loop" nature of ANIL which reduces the computation.

ProtoTypical Networks (PN) is the fastest method as it took only 10–30, 8–22 and 50–130 seconds for ESC, KWS–CNN4 and KWS–ResNet respectively with increasing number of ways. While MetaOptNet was 2–5x slower than PN for ESC and KWS on CNN4 architecture. At increased embedding

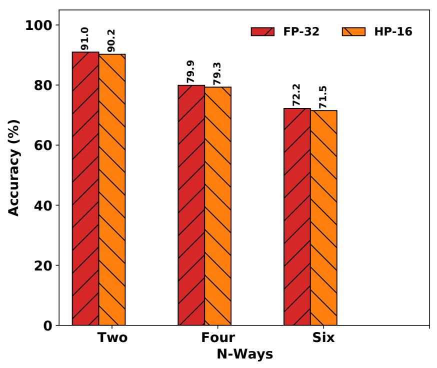


(a) ESC

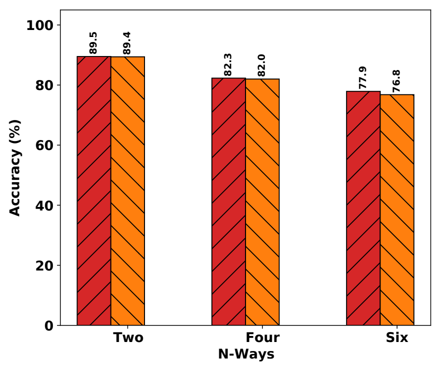


(b) KWS

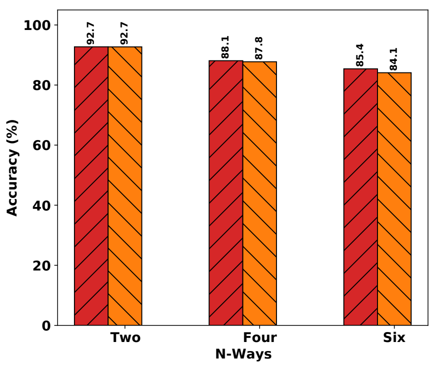
Fig. 1: Average Accuracy for different few shot learning algorithms with varying n ways (2, 4, and 6) for the two datasets on CNN-4.



(a) ESC



(b) KWS – CNN4



(c) KWS – ResNet-12

Fig. 2: Performance of ProtoTypical Network on 16-bit quantized network. FP and HP is Full and Half Precision, respectively.

size of 16000 the latency of MetaOptNet was 20–50% slower than PN for KWS on ResNet architecture. The slower nature of MetaOptNet is due to the use of linear SVM. Overall, ProtoTypical Networks is the least expensive method followed by ANIL, MetaOptNet and MAML and is hence evaluated in the next section for its feasibility for on-device learning.

D. On-device Learning

Motivated by the fact that there is a fuelling demand to adapt and learn on-device deep learning [21], [22], we explored if and how few shot learning methods can run locally on the device. To this end, we created a mobile application in Java using PyTorch Mobile. TorchScript is used to interface between Java and the deep learning code. The app size is 150 MB which mainly consists of PyTorch mobile library. We measured the latency (load, execute and get classification results) for the audio classification task in testing phase for ProtoTypical Networks, due to its high accuracy and efficiency. This is akin to a scenario where a pretrained model (trained with a few shot learning method) can be used to identify new classes in an unseen environment (adaptation phase) fully on the device and we term it as on-device learning in this paper.

On-device learning was performed on two Android devices: a smartphone (OnePlus 7 Pro) having 12 GB of RAM with octa core processors and a smartwatch (TicWatch Pro 3) having 1 GB of RAM and a Wear 4100 processor. We faced challenges with the watch: we could not run ProtoTypical Networks due to small RAM if we try to load and execute forward pass on the neural network model with all the training samples (shots) simultaneously. To solve this challenge, we executed a forward pass on each sample (shots and queries) sequentially and stored the learned embeddings in the memory. Finally, the distance between the query and the shots was used to do the classification.

Table II shows the feasibility of running ProtoTypical Networks on device with six ways (extreme case). The latency ranges between 5.5–20 seconds with CPU utilization of 30–50% and a reasonable memory usage on the smartphones. We observed similar memory usage and CPU utilization but 3x smaller latency with two ways. The latency and resource consumption are higher on smartwatch with latency varying

TABLE II: On-device learning Performance for ProtoTypical Networks for 6 ways, 5 shots and 1 task.

Device	Dataset/Architecture	Latency (sec)	Peak Memory (MB)	CPU Utilization
Smartphone	ESC /CNN-4	8.5	90	30%
	KWS / CNN-4	5.5	67	30%
	KWS / ResNet-12	20	160	50%
Smartwatch	ESC / CNN-4	60	70	40–70%
	KWS / CNN-4	34.5	52	30–70%
	KWS / ResNet-12	190	130	40–90%

between 34–190 seconds. However, two distinct patterns are observed. Unlike, on the smartphone where the CPU utilization was constant throughout the process, it keeps fluctuating on the smartwatch. The lower utilization is attributed to the occurrences when garbage collection is triggered to release the objects, which could pause the app for some time. This also results in lower peak memory usage. However, the total memory usage remains high due to higher latency.

Impact of Quantization: We quantified the impact of quantization on the performance of audio classification tasks with ProtoTypical Networks. Figure 2 shows that 16-bit quantization has negligible impact on the accuracy for both ESC and KWS datasets. We also observed a 50% decrease in the model sizes and 40–60% improvement in the latency during testing phase. We are the first to evaluate the impact of quantization on audio classification tasks for few shot learning.

IV. Conclusion and Discussion

Few shot learning provides an excellent way to learn unseen classes using a few samples and its applicability in audio domain have been explored. However, there still remain a few limitations in existing work which merits further work. To this end we investigate the few shot learning methods for two audio classification tasks: acoustic event and keyword. Through a large number of experiments, we show that these methods can adapt to new classes well and simple metric-based methods are more efficient and scalable than complex gradient based methods. We also showed that complex neural network architecture should be explored in the audio domain when doing few shot learning. However, this can be only achieved with much bigger datasets having a large number of classes. We also overcome challenges to enable few shot learning on-device for unseen classes in the adaptation phase which will ensure complete privacy of the users data.

Acknowledgments

This work is supported by ERC through Project 833296 (EAR), and Nokia Bell Labs.

References

[1] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning*, 2017, pp. 1126–1135.

[2] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, “Rapid learning or feature reuse? towards understanding the effectiveness of maml,” 2020.

[3] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4080–4090.

[4] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-learning with differentiable convex optimization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[5] Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai, “Few-shot object detection with attention-rpn and multi-relation detector,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4013–4022.

[6] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, “Improved few-shot visual classification,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 493–14 502.

[7] S. Gidaris and N. Komodakis, “Dynamic few-shot visual learning without forgetting,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4367–4375.

[8] A. Antoniou, H. Edwards, and A. Storkey, “How to train your maml,” in *ICLR*, 2019.

[9] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 3637–3645.

[10] S. Srivastava, I. Labutov, and T. Mitchell, “Zero-shot learning of classifiers from natural language quantification,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 306–316.

[11] 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)*. IEEE, 2016, pp. 1128–1132.

[12] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, “Detection and classification of acoustic scenes and events,” *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[13] P. Swarup, R. Maas, S. Garimella, S. H. Mallidi, and B. Hoffmeister, “Improving asr confidence scores for alexa using acoustic and hypothesis embeddings.” in *INTERSPEECH*, 2019, pp. 2175–2179.

[14] S.-Y. Chou, K.-H. Cheng, J.-S. R. Jang, and Y.-H. Yang, “Learning to match transient sound events using attentional similarity for few-shot sound recognition,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 26–30.

[15] Y. Wang, J. Salamon, N. J. Bryan, and J. P. Bello, “Few-shot sound event detection,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 81–85.

[16] S. Zhang, Y. Qin, K. Sun, and Y. Lin, “Few-shot audio classification with attentional graph neural networks.” in *INTERSPEECH*, 2019, pp. 3649–3653.

[17] J. Pons, J. Serrà, and X. Serra, “Training neural audio classifiers with few data,” in *ICASSP 2019*. IEEE, 2019, pp. 16–20.

[18] K. J. Piczak, “Esc: Dataset for environmental sound classification,” in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

[19] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *arXiv preprint arXiv:1804.03209*, 2018.

[20] B. Shi, M. Sun, K. C. Puvvada, C.-C. Kao, S. Matsoukas, and C. Wang, “Few-shot acoustic event detection via meta learning,” in *ICASSP 2020*. IEEE, 2020, pp. 76–80.

[21] J. Chauhan, Y. D. Kwon, P. Hui, and C. Mascolo, “Contauth: Continual learning framework for behavioral-based user authentication,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, pp. 1–23, 2020.

[22] H. Cai, C. Gan, L. Zhu, and S. Han, “Tinytl: Reduce memory, not parameters for efficient on-device learning,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.