INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

A meta-learning approach for user-defined spoken term classification with varying classes and examples

Yangbin Chen[1], *Tom Ko*[2*], *Jianping Wang*[3]

[1] Department of Information Engineering, The Chinese University of Hong Kong, China
[2] Department of Computer Science and Engineering,
Southern University of Science and Technology, China
[3] Department of Computer Science, City University of Hong Kong, China
{dongyiwu92,tomkocse}@gmail.com, jianwang@cityu.edu.hk

Abstract

Recently we formulated a user-defined spoken term classification task as a few-shot learning task and tackled the task using Model-Agnostic Meta-Learning (MAML) algorithm. Our results show that the meta-learning approach performs much better than conventional supervised learning and transfer learning in the task, especially with limited training data. In this paper, we extend our work by addressing a more practical problem in the user-defined scenario where users can define any number of spoken terms and provide any number of enrollment audio examples for each spoken term. From the perspective of few-shot learning, this is an N-way, K-shot problem with varying N and K. In our work, we relax the values of N and K of each meta-task during training instead of assigning fixed values to them, which differs from what most meta-learning algorithms do. We adopt a metric-based meta-learning algorithm named Prototypical Networks (ProtoNet) as it avoids exhaustive fine-tuning when N varies. Furthermore, we use the Max-Mahalanobis Center (MMC) loss as an effective regularizer to address the problem of ProtoNet under the condition of varying K. Experiments on the Google Speech Commands dataset demonstrate that our proposed method outperforms the conventional N-way, K-shot setting in most testing tasks.

Index Terms: spoken term classification, few-shot learning, meta-learning

1. Introduction

Spoken term classification aims at identifying spoken terms, which is applied in voice-based Human-Computer Interaction (HCI) products, such as Apple Siri, Google Assistant, and Amazon Alexa. However, most spoken terms are predefined and known in advance. A user-defined spoken term classification system, on the contrary, allows users to enroll new spoken terms by recording only a few audio examples, which is more flexible.

We face two challenges in developing a user-defined spoken term classification system. The first challenge is that we do not know how many spoken terms users would like to define. It leads to an open-set classification problem, in which newly defined spoken terms can be the same as or very different from existing ones in the training data pool. The second challenge is that we do not know how many audio examples of the new spoken terms users would like to record. The number cannot be significant in practical applications, which results in limited enrolled data of the new spoken terms.

* corresponding author

Researchers formulate the user-defined spoken term classification task as a few-shot learning task and tackle it by meta-learning approaches [1, 2]. They define an N-way, K-shot task, in which N is the number of new spoken terms and K is the number of labeled examples per spoken term. In the training stage, plenty of N-way, K-shot meta-tasks are sampled from the source data. A model initializer [1] or an embedding function [2] is learned by training across the meta-tasks. In the testing stage, the initializer or embedding function is evaluated by another group of N-way, K-shot tasks sampled from the target data. Specifically, in each meta-task, the initializer is fine-tuned on labeled examples (a.k.a. support set) and evaluated on query examples (a.k.a. query set). The embedding function extracts features from examples, and the query examples are classified by distances between them and class centers of the support set.

The meta-learning approaches achieve competitive performance with limited enrolled data from the new spoken terms. However, to the best of our knowledge, most few-shot learning tasks are set as N-way, K-shot, where the values of N and K depend on the predefined testing tasks. Nevertheless, there are disadvantages to this kind of setting. There is no guarantee that the same settings of N and K in the training meta-tasks as the testing tasks will perform the best. Moreover, a practical user-defined spoken term classification system should allow users themselves to decide how many new spoken terms to define and how many enrollment audio examples to record, which means that N and K can vary among different users.

This paper proposes a series of strategies for user-defined spoken term classification with varying classes and examples. We relax th N and K values during training and replace MAML with ProtoNet to avoid exhaustive fine-tuning when N varies in testing tasks. Moreover, as ProtoNet considers the mean rather than the variance of examples in the support set, there may be a shortcoming when ProtoNet meets varying K values. For example, given $class_A$ with large K and variance and $class_B$ with small K and variance, a $class_A$'s query example near the boundary between $class_A$ and $class_B$ may be misclassified to $class_B$ whose center is closer to the example. To solve this problem, we add the Max-Mahalanobis Center (MMC) loss as a regularizer to move different classes far apart in feature space.

Contributions of this paper can be summarized as follows:

- We investigate the effect of N in the training stage and find that a large or varying N is helpful in the prototypical networks.
- We investigate the effect of K in the training stage and find that a similar K to the testing tasks improves the performance.

- Our method of training with a significant N and a varying K, together with our proposed regularizer, outperforms training with fixed N and K in most testing tasks, especially when the testing tasks are with varying N and K.

In the following sections, we review a series of related works in Section 2. In Section 3, we introduce our proposed methodology. In Section 4, we present details of our experiment. Conclusions are drawn in Section 5.

2. Related work

A few-shot learning problem is a machine learning problem that learns with limited labeled data of target tasks by incorporating source data, which has a different distribution from the target data [3]. Meta-learning, which aims to learn the learning algorithm itself to adapt to any new task quickly, has become popular in solving few-shot learning problems. Model-Agnostic Meta-Learning (MAML) [4] and Prototypical Networks (ProtoNet) [5] are two representative meta-learning approaches. Both train a backbone model across various meta-tasks, each consisting of a support set and a query set. MAML learns a model initializer that can quickly generalize to new tasks with only a few labeled data and fine-tuning steps, while ProtoNet learns a non-linear mapping from the input space to the embedding space with a predefined distance metric.

Meta-learning approaches have been widely applied in speech tasks, such as speech recognition [6, 7, 8], speaker verification [9, 10, 11], and acoustic event classification [12, 13, 14]. These tasks face challenges in adaptations from high-resource tasks to low-resource tasks. Most conventional research works in spoken term classification focus on developing deep neural networks and training the networks on large-scale annotated data [15, 16, 17]. With the development of personalized virtual assistants, user-defined spoken term classification becomes popular, which can be tackled by meta-learning approaches. [1] defined an N+M-way, K-shot task, where N is the number of new spoken terms and M is the number of fixed spoken terms and developed an extended MAML algorithm. [2] leveraged ProtoNet to learn task-variant representations for tackling the few-shot spoken intent classification task. Our work distinguishes from these works as all of them limit the number of classes and examples.

3. Methodology

3.1. Problem definition

Assume that a training set D_{source} containing predefined classes $\{C_1, C_2, ..., C_T\}$ and a test set D_{target} containing new classes $\{C_{T+1}, C_{T+2}, ..., C_{T+L}\}$ are given. Each predefined class consists of labeled examples $\{(\boldsymbol{x}_{i,j}, y_i)|_{i=1}^{T}\}$, and each new class consists of both labeled examples $\{(\boldsymbol{x}_{i,u}, y_i)|_{i=T+1}^{T+L}\}$ and unlabeled examples $\{\boldsymbol{x}_{i,v}|_{i=T+1}^{T+L}\}$. Our motivation is to train a model using D_{source}, such that the model can make correct predictions to the unlabeled examples in D_{target} with the help of only a few labeled examples.

3.2. Prototypical networks with varying N and K

ProtoNet is a metric-based meta-learning approach, which assumes that examples from the same class will cluster around a single point (i.e., prototype representation). It aims to learn a non-linear mapping from an input space to an embedding space, so that classification can be performed by calculating the distances between the test examples and the prototype representation of each class in the embedding space.

Specifically, in few-shot learning tasks with varying N and K, each target task contains a random number of classes, and each class has a random number of labeled examples. In the training stage, as Figure 1 illustrates, a mini-batch of meta-tasks is sampled from D_{source} in each training step. Every single meta-task T_τ is built with two sets – a support set $\mathcal{S}_\tau$ and a query set $\mathcal{Q}_\tau$. $\mathcal{S}_\tau$ contains examples $\{(\boldsymbol{x}_{i,k}, y_i)|i = 1, 2, ...N_\tau, k = 1, 2, ..., K_{\tau,i}\}$ and $\mathcal{Q}_\tau$ contains examples $\{(\boldsymbol{x}_{i,h}, y_i)|i = 1, 2, ..., N_\tau, h = 1, 2, ..., H\}$. N_τ is the number of classes sampled from D_{source}, $K_{\tau,i}$ is the size of class C_i in the support set of meta-task $\mathcal{T}_\tau$, and H is the size of each class in the query set. In our work, N_τ and $K_{\tau,i}$ are flexible for a meta-task T_τ and a class C_i within the meta-task, respectively, and H is a manually defined fixed value.

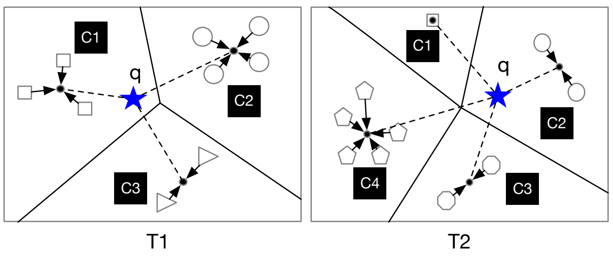

Figure 1: *A framework of Prototypical Networks with varying N and K. C_i denotes a class from the support set. Each class is represented by a prototype embedding. q denotes an example from the query set.*

The first step of ProtoNet is to calculate N_τ mean vectors as prototype representations in the support set:

$$\boldsymbol{c}_i = \frac{1}{K_{\tau,i}} \sum_{k=1}^{K_{\tau,i}} f_\theta(\boldsymbol{x}_{i,k}) \tag{1}$$

where f_θ is the embedding function.

Then we calculate the squared Euclidean distances between the examples in the query set and the N_τ prototype representations of the support set:

$$d(f_\theta(\boldsymbol{x}_{i,h}), \boldsymbol{c}_{i'}) = ||(f_\theta(\boldsymbol{x}_{i,h}) - \boldsymbol{c}_{i'}||_2^2 \tag{2}$$

A probability distribution of a query example over the N_τ classes in the support set is a softmax output based on the distances:

$$p_\theta(y_i|\boldsymbol{x}_{i,h}) = \frac{exp(-d(f_\theta(\boldsymbol{x}_{i,h}), \boldsymbol{c}_i))}{\sum_{i'} exp(-d(f_\theta(\boldsymbol{x}_{i,h}), \boldsymbol{c}_{i'}))} \tag{3}$$

The negative log-probability based classification loss for each meta-task is depicted as follows:

$$L_\tau = \frac{1}{N_\tau} \sum_{i=1}^{N_\tau} \frac{1}{H} \sum_{h=1}^{H} -logp_\theta(y_i|\boldsymbol{x}_{i,h}) \tag{4}$$

3.3. To address the shortcoming of ProtoNet under varying K

Classes containing a varying number of examples in the support set have different mean and variance. Metric-based meta-learning algorithms consider the mean rather than the variance.

As Figure 2 (a) illustrates, according to the data distribution, the query example q should obviously belong to C_1. However, according to the distances between the query example and the prototype representations, it will be classified to C_3. The reason is that the clusters of various classes are too close in the embedding space. Under this condition, small-shot classes can easily misclassify large-shot ones.

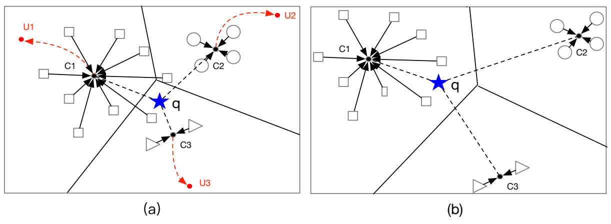

Figure 2: *A meta-task with varying K. C_i denotes a class from the support set and U_i denotes an ideal center we try to push C_i to.*

In this paper, we propose a regularizer to force the classes to move far apart from each other in the embedding space, as Figure 2 shows, which is inspired by Max-Mahalanobis center (MMC) loss [18]. We predefine a number of 'ideal centers' $\{\boldsymbol{u}_i|_{i=1}^{T}\}$ for each class in $\mathcal{D}_{source}$, which have the same dimensions as prototype representations and far apart from each other in the embedding space. Then we define the regularizer as follows:

$$L_{\tau}^{reg} = \frac{1}{2} log \frac{\sum_i K_{\tau,i} ||\boldsymbol{c}_i - \boldsymbol{u}_i||_2^2}{\sum_i K_{\tau,i}} \tag{5}$$

which aims to pull the prototype representations close to the 'ideal centers'. The total loss within each meta-task is a weighted combination of the classification loss and the regularization loss:

$$L_{\tau}^{total} = L_{\tau} + \lambda L_{\tau}^{reg} \tag{6}$$

The backbone model is trained across the meta-tasks in order to learn robust embeddings, which can maximize the between-class distances and minimize the within-class distances.

4. Experiments

4.1. Dataset

Our experiment is conducted on the Google Speech Commands dataset (v0.02) [19], which collects 105,829 1-second audio clips from 35 keywords. We follow the work of user-defined digits recognition in [1], splitting the dataset into four parts – source keywords, target keywords, unknown, and silence. As Table 1 shows, the source data has twenty source keywords, and the target data has ten target keywords. The unknown and silence are two fixed classes, which appear in both the source and target data as prior knowledge. In our experiment, the target task is $N_\tau + 2$-way, $K_{\tau,i}$-shot, where we sample the number of user-defined spoken terms $N_\tau \sim U[1, 10]$ independently for each task and the number of labeled examples $K_{\tau,i} \sim U[1, 10]$ independently for each spoken term by a uniform distribution. The number of unlabeled examples in the query set is 100 in each class. We evaluate the learned model on 10,000 sampled target tasks. As for the training meta-tasks, we sample $N_\tau \sim U[1, 20]$ independently, according to the size of source data.

Table 1: *Data split for user-defined digits recognition. The unknown class is formed by 5 keywords: 'bed', 'dog', 'happy', 'marvin', and 'wow'.*

Source data	Target data
'yes','no','up','down', 'left','right','on','off', 'stop','go', 'cat', 'bird', 'forward', 'backward', 'follow', 'learn', 'sheila', 'tree', 'visual', 'house' **unknown**, **silence**	'zero', 'one','two','three', 'four', 'five','six','seven', 'eight','nine', **unknown**, **silence**

4.2. Implementation details

For each audio clip sampled at 16kHz, we extract 40-dimensional MFCC features with a frame length of 30ms and a frame step of 10ms. A Convolutional Neural Network (CNN) is used as the backbone model to learn the embeddings. It contains four convolutional blocks, each comprising a 3×3 convolution kernel and 64 filters, followed by ReLU and batch normalization [20]. The embedding size of the flattened layer is 576. The coefficient λ in Equation (6) is 1. In the training stage, we take Adam [21] as the optimizer. All models are trained on a single Nvidia GeForce RTX 2080 Ti GPU using PyTorch[1]. Our code is available at https://github.com/Codelegant92/STC-ProtoNet.

4.3. Effect of the training N value

To analyze the effect of the number of classes in meta-tasks during training, we experiment on an $N + 2$-way, 5-shot setting. In the training stage, N is set to 1, 2, 3, 5, 10, 20, and N_τ (i.e., a random integer between 1 and 20). In the testing stage, N is set to 5, 10, and N_τ (i.e., a random integer between 1 and 10). We add three baselines to compare. The first is conventional supervised learning (Super.V.), which can access the target data only. The second is transfer learning (Transf.L.), which pre-trains a 22-class classifier using the source data and fine-tunes the classifier using the target data. The difference between transfer learning and 20+2-way training in our work is that the backbone model is trained across mini-batches of examples in transfer learning. However, in ProtoNet, the backbone model is trained across mini-batches of meta-tasks. The third baseline is MAML. Table 2 lists the results. The results of three baselines on N_τ+2-way, 5-shot are not listed because they need to rebuild and reinitialize the last fully-connected layer of the backbone model in every testing task, which is time and space consuming.

Overall speaking, ProtoNet performs much better than conventional supervised learning and transfer learning approaches. To analyze the N value effect, we can see that keeping the same N in the training meta-tasks and the testing meta-tasks does not mean the best performance. However, a more significant value or a random setting of N in the training stage achieves slightly better performance. The reason is that a more difficult meta-task enhances representation learning.

4.4. Effect of the training K value

We also investigate the effect of the number of examples in each class. We experiment on 20+2-way, K-shot meta-tasks for training, and 10+2-way, K-shot tasks for testing. In the

[1]https://pytorch.org

Table 2: *Accuracy with 95% confidence intervals of experiments on N+2-way, 5-shot classification tasks.*

	Testing		
Training	**5+2-way**	**10+2-way**	**N_τ+2-way**
Superv.L.	27.52 ± 0.27	24.83 ± 0.38	-
Transf.L.	62.67 ± 0.38	54.43 ± 0.47	-
MAML	67.57 ± 0.91	63.22 ± 0.71	-
1+2-way	62.73 ± 0.12	52.32 ± 0.05	63.14 ± 0.21
2+2-way	74.33 ± 0.10	65.21 ± 0.05	54.42 ± 0.25
3+2-way	75.32 ± 0.10	66.38 ± 0.04	75.09 ± 0.16
5+2-way	76.38 ± 0.10	67.84 ± 0.04	76.47 ± 0.16
10+2-way	76.30 ± 0.09	67.92 ± 0.04	76.39 ± 0.15
15+2-way	76.28 ± 0.09	67.55 ± 0.04	76.23 ± 0.16
20+2-way	76.86 ± 0.09	$\mathbf{68.44 \pm 0.04}$	76.78 ± 0.15
N_τ+2-way	$\mathbf{76.90 \pm 0.09}$	68.13 ± 0.04	$\mathbf{76.82 \pm 0.16}$

training stage, K is set to 1, 6, 10, and $K_{\tau,i}$ (i.e., a random integer between 1 and 10). In the testing stage, K is set from 1 to 10. Figure 3 illustrates the accuracy change with the increase of testing shot under different training conditions. The colored curves represent five training conditions.

From the green, yellow, and blue curves, we find that a comparable K value between the training and testing tasks improves the performance. To compare these three conditions, we can see that when the testing shot's value is no more than 5, 1-shot training performs the best while 10-shot training performs the worst. When the testing shot's value is more than 5, 10-shot training performs the best while 1-shot training performs the worst. From the purple and red curves, we find that both curves lie above almost all the other curves, which means varying K-shot training achieves better and more stable performance, whether the testing shot's value is small or large. We also find that adding the MMC loss-based regularizer improves the performance.

Moreover, in Table 3, we show the testing results on $K_{\tau,i}$-shot under several training conditions. We find that training with $K_{\tau,i}$-shot performs much better than training with a fixed shot, and adding the regularizer achieves another 1.58 percent's improvement. The reason is that a random value of K, together with the MMC loss maximize the inter-to-intra class variance.

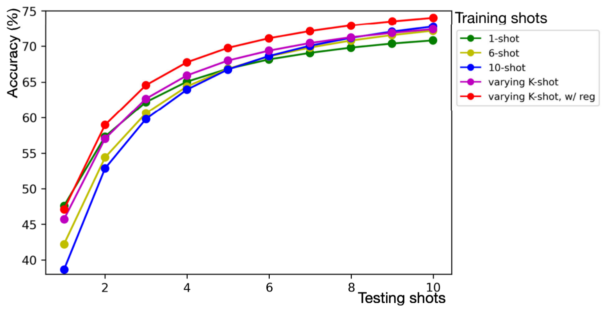

Figure 3: *Experiments on 20+2-way, K-shot tasks for training and 10+2-way, K-shot tasks for testing.*

4.5. Testing on tasks with varying classes and examples

We do an evaluation of our proposed method in target tasks with varying classes and examples. According to our findings in Section 4.3 and Section 4.4, a large or varying N value, a varying K value, or an MMC loss-based regularizer will boost the performance. We combine these factors all in this experiment. The results are listed in Table 4. We can see from the table that our method, which uses 20+2-way, $K_{\tau,i}$-shot meta-tasks and the regularizer performs much better than most fixed settings of the training way and training shot.

Table 3: *Accuracy with 95% confidence intervals of experiments on 10+2-way, $K_{\tau,i}$-shot tasks for testing.*

	Testing
Training	**$K_{\tau,i}$-shot**
1-shot	66.32 ± 0.05
2-shot	65.94 ± 0.06
4-shot	66.20 ± 0.06
6-shot	64.96 ± 0.07
8-shot	64.11 ± 0.07
10-shot	64.60 ± 0.07
$K_{\tau,i}$-shot	$\mathbf{67.29 \pm 0.06}$
$K_{\tau,i}$-shot (w/reg)	$\mathbf{68.87 \pm 0.06}$

Table 4: *Accuracy with 95% confidence intervals of experiments on N_τ-way, $K_{\tau,i}$-shot tasks for testing.*

	Testing
Training	**N_τ-way, $K_{\tau,i}$-shot**
1+2-way, 1-shot	75.02 ± 0.17
1+2-way, 5-shot	74.92 ± 0.17
5+2-way, 1-shot	75.02 ± 0.17
5+2-way, 5-shot	74.92 ± 0.17
10+2-way, 1-shot	75.23 ± 0.17
10+2-way, 5-shot	74.56 ± 0.17
20+2-way, 1-shot	74.90 ± 0.17
20+2-way, 5-shot	74.95 ± 0.17
20+2-way, 10-shot	72.88 ± 0.17
20+2-way, $K_{\tau,i}$-shot	$\mathbf{75.77 \pm 0.16}$
20+2-way, $K_{\tau,i}$-shot (w/ reg)	$\mathbf{77.21 \pm 0.16}$

5. Conclusion

This paper raises a practical user-defined spoken term classification task allowing users to define any number of spoken terms and enroll any number of audio examples for each spoken term. We formulate it as a few-shot learning task with varying N-way and varying K-shot and tackle it using Prototypical Networks. We first investigate the effect of training way and training shot in an N-way, K-shot task, which shows that a significant or varying N and a varying K can improve the performance. Then we define an Max-Mahalanobis Center (MMC) loss-based regularizer, which forces different classes' prototype representations to move far apart from each other in the embedding space. We propose a strategy that samples meta-tasks with a significant N and a varying K in the training stage and adds the MMC loss-based regularizer for back-propagation. Experiments on the Google Speech Commands dataset demonstrate that our proposed method achieves significant improvement in target tasks with varying classes and examples.

6. Acknowledgements

The authors would like to thank Ziquan Liu from City University of Hong Kong for his useful discussions and feedback.

7. References

[1] Y. Chen, T. Ko, L. Shang, X. Chen, X. Jiang, and Q. Li, "An investigation of few-shot learning in spoken term classification," in *INTERSPEECH*, 2020, pp. 2582–2586.

[2] A. Mittal, S. Bharadwaj, S. Khare, S. Chemmengath, K. Sankaranarayanan, and B. Kingsbury, "Representation based meta-learning for few-shot spoken intent recognition," in *INTERSPEECH*, 2020, pp. 4283–4287.

[3] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[4] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017, pp. 1126–1135.

[5] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4077–4087.

[6] J.-Y. Hsu, Y.-J. Chen, and H.-y. Lee, "Meta learning for end-to-end low-resource speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7844–7848.

[7] S. Indurthi, H. Han, N. K. Lakumarapu, B. Lee, I. Chung, S. Kim, and C. Kim, "End-end speech-to-text translation with modality agnostic meta-learning," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7904–7908.

[8] G. I. Winata, S. Cahyawijaya, Z. Liu, Z. Lin, A. Madotto, P. Xu, and P. Fung, "Learning fast adaptation on cross-accented speech recognition," in *INTERSPEECH*, 2020, pp. 1276–1280.

[9] T. Ko, Y. Chen, and Q. Li, "Prototypical networks for small footprint text-independent speaker verification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6804–6808.

[10] R. Li, J.-Y. Jiang, X. Wu, H. Mao, C.-C. Hsieh, and W. Wang, "Bridging mixture density networks with meta-learning for automatic speaker identification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3522–3526.

[11] O. Klejch, J. Fainberg, P. Bell, and S. Renals, "Speaker adaptive training using model agnostic meta-learning," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 881–888.

[12] B. Shi, M. Sun, K. C. Puvvada, C.-C. Kao, S. Matsoukas, and C. Wang, "Few-shot acoustic event detection via meta learning," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 76–80.

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

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

[15] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.

[16] J. Bae and D.-S. Kim, "End-to-end speech command recognition with capsule network." in *INTERSPEECH*, 2018, pp. 776–780.

[17] S. Ö. Arık, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," in *INTERSPEECH*, 2017, pp. 1606–1610.

[18] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, and J. Zhu, "Rethinking softmax cross-entropy loss for adversarial robustness," in *International Conference on Learning Representations (ICLR)*, 2019.

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

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 448–456.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015, pp. 1–13.