Interspeech 2024
1-5 September 2024, Kos, Greece

Few-Shot Keyword-Incremental Learning with Total Calibration

*Ilseok Kim, Ju-Seok Seong, Joon-Hyuk Chang**

Department of Electronic Engineering, Hanyang University, Seoul, Republic of Korea
{maybe911, as2835510, jchang}@hanyang.ac.kr

Abstract

Keyword spotting (KWS) models need to continuously recognize new keywords for user demand. However, two significant challenges exist in satisfying this requirement: catastrophic forgetting, where the model loses its ability to classify previously learned keywords, and insufficient data for new classes. To address these challenges, we propose a Few-shot keyword-Incremental Learning with total caLibration (FILL), a novel few-shot class-incremental learning (FSCIL) approach for KWS. FSCIL trains a model with sufficient data in an initial session, followed by incremental sessions where it learns new classes with limited data. FILL employs prototype calibration throughout total sessions to enhance class separation and mitigate misclassification. Notably, it utilizes manifold mixup in the initial session to generate new classes for prototype calibration. Experimental results on two KWS datasets demonstrate that FILL outperforms three baselines in terms of average accuracy.

Index Terms: keyword spotting, few-shot learning, incremental learning, few-shot class-incremental learning

1. Introduction

Keyword spotting (KWS) is a task that recognizes pre-defined keywords from an input audio [1–3]. Recent research in KWS has extended beyond recognizing pre-defined keywords to focus on user-defined keywords [4–6], enabling users to define and utilize their unique keywords. The introduction of user-defined keywords necessitates continuous model adaptation to new keywords added post-deployment. However, conventional KWS systems require inefficient retraining of the entire model to recognize new keywords. To address this problem, incremental learning (IL) has been introduced [7, 8]. IL is a learning approach that incrementally learns new classes in each session. The goal of IL is to learn the ability to recognize new classes while maintaining the ability learned in previous sessions. More recently, IL has been successful applications in KWS systems [9, 10]. However, it is an unrealistic assumption that sufficient data will given for additional classes. In reality, the data available for additional classes is inevitably limited [4, 11]. As a result, the few-shot class-incremental learning (FSCIL) [12–20] approach, which applies the few-shot setup to IL, has been proposed. Specifically, FSCIL involves training the model with a sufficient amount of data in the initial session (base session) and then, in incremental sessions, learning the recognition capability for additional classes using only N-way K-shot data.

In FSCIL, two primary issues are notably present [20].

*Corresponding author.

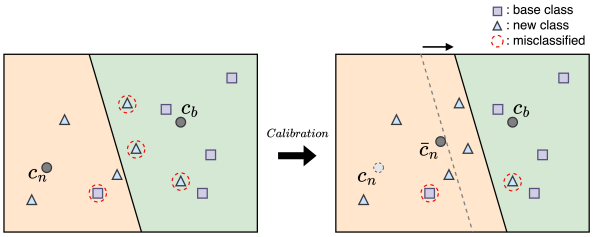


Figure 1: *Illustration for result of prototype calibration. It compares the embedding space before and after calibration. The prototype of the new class c_n is transformed to $\bar{c}_n$ through calibration, shifting the decision boundary closer to the base class prototype c_b, as shown on the right.*

First, during the incremental learning of new classes, there is the challenge of catastrophic forgetting [21], where the classification performance for previously learned classes deteriorates. Second, there is the issue of overfitting the model to the few-shot data used for learning new classes. A basic approach to address the challenges of FSCIL is to utilize the prototypical network [22]. In the base session, the feature extractor is trained using sufficient data, and prototypes are extracted for each base classes. Subsequently, the feature extractor is frozen in the incremental session, and prototypes for each class are extracted for classification. By fixing the feature extractor, this method helps prevent overfitting when learning new classes and alleviates the issue of catastrophic forgetting for existing classes. This highlights the strength of prototype-based methods as strong FSCIL baselines, as demonstrated by various studies [14–19]. For example, continually evolved classifier (CEC) [14] leverages a graph-based model to integrate individual classifiers learned in each session, facilitating continuous knowledge acquisition for new classes while retaining knowledge of previously learned classes. On the other hand, forward compatible training (FACT) [15] pre-allocates embedding space for future new classes in the embedding space during base session training by utilizing virtual classes. Also, training free calibration (TEEN) [16] employs prototype calibration to prevent misclassification of incremental session classes into base session classes by bringing the prototypes of incremental sessions closer to those of the base session, as shown in Figure 1.

Inspired by TEEN, we propose a new method that calibrates prototypes in both base and incremental sessions, improving overall accuracy compared to methods focusing solely on base or incremental sessions. Our experiments show that TEEN benefits incremental classes, but does not improve base classes accuracy. To mitigate catastrophic forgetting of the base session classes and facilitate efficient learning of new classes in incre-

10.21437/Interspeech.2024-1823

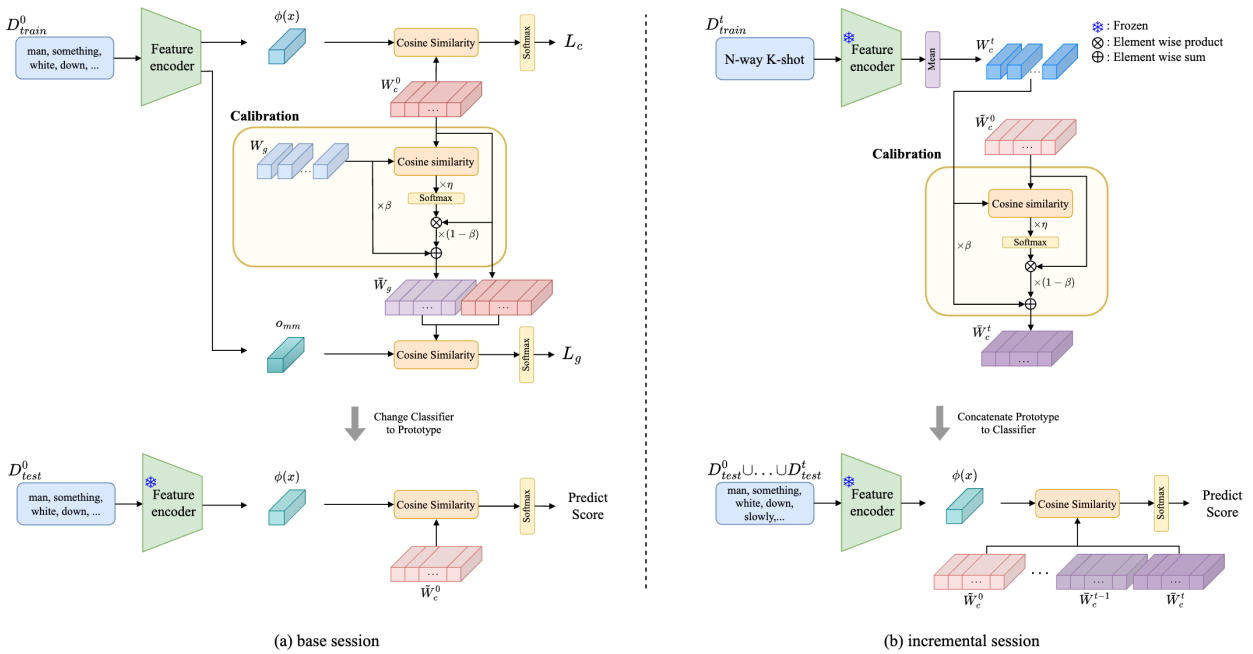


Figure 2: *Illustration of FILL training and evaluation phase. (a) The training uses true labels with embeddings for L_c and a manifold mixup for L_g. Evaluation relies on prototypes and cosine similarity after freezing the encoder. (b) Describes constructing a classifier in an incremental session. Bottom describes constructing a classifier in an incremental session.*

mental sessions, we perform prototype calibration on the generated class using manifold mixup in the base session. Additionally, we contribute by preparing several KWS datasets for FSCIL setup and conducting benchmark comparisons.

2. Preliminary

In the FSCIL scenario, suppose there are T sequential sessions consisting of a base session (session 0) and incremental sessions (excluding session 0). The dataset used for training in each session is represented by $\{\mathcal{D}^0_{train}, \mathcal{D}^1_{train}, ..., \mathcal{D}^{T-1}_{train}\}$, where $\mathcal{D}^t_{train} = \{(x_i, y_i)_{i=1}^{N_t}\}$ denotes the training dataset for session t. Each session t has a disjoint label space $\mathcal{C}_t$, meaning that for any two sessions t_1 and t_2, $\mathcal{C}_{t_1} \cap \mathcal{C}_{t_2} = \varnothing$. The base session training dataset, $\mathcal{D}^0_{train}$, contains a sufficient amount of data. Training datasets in incremental sessions are limited to reflect real-world scenarios; they use N-way K-shot data where each of the N classes has only K samples. The testing datasets for each session are represented by $\{\mathcal{D}^0_{test}, \mathcal{D}^1_{test}, ..., \mathcal{D}^{T-1}_{test}\}$. The testing dataset for each session consists of data with labels from the same label space as that session's training dataset. Furthermore, in incremental sessions, the testing datasets contain significantly more data than the training datasets to ensure a robust evaluation. At the end of each session, the model is evaluated using the union of its current testing dataset and all previous testing datasets. In other words, at the end of session t, the evaluation dataset is: $\mathcal{D}^0_{test} \cup ... \cup \mathcal{D}^{t-1}_{test} \cup \mathcal{D}^t_{test}$.

3. Method

Our proposed method, FILL, employs an architecture that consists of a feature extractor $\phi(\cdot)$ and a classifier $W = [w_1, w_2, ...] \in R^{d \times N}$, composed of representative weight vectors for each of the N classes (see Figure 2). The feature extractor transforms input data $x \in R^n$ into embedded representation $\phi(x) \in R^d$. FILL consists of training in the base session and the incremental session. The base classes are leveraged in the base session to jointly train the feature extractor and the classifier, establishing the foundational knowledge. During incremental sessions, to minimize catastrophic forgetting, the parameters of the feature extractor are frozen while the classifier is solely updated. The details are described below.

The base session training involves two loss components:

$$L = L_c + \alpha \cdot L_g, \tag{1}$$

where α is a scalar hyperparameter that controls the influence of the second loss component L_g, as shown in the top of Figure 2(a). The first loss component, denoted as L_c, is a simple classification loss that utilizes the cosine similarity between the classifier $W^0_c = [w_1, w_2, ..., w_{|\mathcal{C}_0|}] \in R^{d \times |\mathcal{C}_0|}$ and the embedding $\phi(x)$ extracted from the base class data x via the feature extractor:

$$L_c = \text{CrossEntropy}(S(\phi(x), W^0_c), y), \tag{2}$$

where $y \in \mathcal{C}_0$ is a label of the base class and $S(\cdot, \cdot)$ is the cosine similarity function; which is defined by:

$$S(\phi(x), W) = \frac{W^\top \phi(x)}{||W||\,||\phi(x)||}. \tag{3}$$

For the second loss component, L_g, we first calibrate the classifier weights of the generated classes, $W_g = [g_1, g_2, ..., g_{|\mathcal{C}_0|}] \in R^{d \times |\mathcal{C}_0|}$, with classifier weights of the base classes. Then, we leverage the manifold mixup approach in the base session to synthesize the generated class data, simulating calibration in the incremental session and facilitating the learning process. The calibrated weight $\bar{g}_n$ of the generated class weight g_n is as follows:

$$\bar{g}_n = \beta g_n + (1 - \beta)\Delta g_n, \tag{4}$$

where β is a hyperparameter that determines the strength of calibration. Here, Δg_n measures the cosine similarity between the classifier weights of the base session classes and the classifier weights of the n-th class in the generated classifier, and is calculated as:

$$\Delta g_n = \sum_{b=1}^{|\mathcal{C}_0|} r_{b,n} w_b, \tag{5}$$

$$r_{b,n} = \frac{e^{\eta S(w_b, g_n)}}{\sum_{i=1}^{|\mathcal{C}_0|} e^{\eta S(w_i, g_n)}}, \tag{6}$$

where a η is the scalar factor adjusting the weight distribution's sharpness. The calibration process results in the calibrated classifier $\bar{W}_g = [\bar{g}_1, \bar{g}_2, ..., \bar{g}_{|\mathcal{C}_0|}] \in R^{d \times |\mathcal{C}_0|}$. After the calibration of the generated classifier, the classification loss L_g is computed using the classifier $[W_c^0, \bar{W}_g] \in R^{d \times 2 \cdot |\mathcal{C}_0|}$ and the manifold mixup embedding, denoted by o_{mm}, which is derived from input data within the same batch but not belonging to the same class:

$$L_g = \text{CrossEntropy}(S(o_{mm}, [W_c^0, \bar{W}_g]), \hat{y}), \tag{7}$$

where $\hat{y}$ is a pseudo label for the generated class via $\text{argmax}_i \, g_i^\top o_{mm} + |\mathcal{C}_0|$ [15]. After all training has ended, the weights for each class in W_c^0 are replaced with prototypes calculated using the entire base training dataset corresponding to each class, i.e., $\tilde{W}_c^0 = [p_1, p_2, ..., p_{|\mathcal{C}_0|}] \in R^{d \times |\mathcal{C}_0|}$. The prototype vector for class b is calculated by taking the average of the embedded vector representations corresponding to class b:

$$p_b = \frac{1}{|C_b|} \sum_{i=1}^{|C_b|} \phi(x_i), \tag{8}$$

where C_b is the set of all data instances belonging to class b. At evaluation time, we extract embeddings from the frozen feature extractor and predict the class by calculating the cosine similarity with the base classifier.

During the incremental session, we use the feature encoder to extract embeddings from the N-way K-shot data and obtain prototypes for the N classes, as shown in the top of Figure 2(b). After that, we construct the classifier $W_c^t \in R^{d \times N}$ with the N prototypes. Then, W_c^t is calibrated according to the Eq. (4) formula using the base session classifier $\tilde{W}_c^0$. These extracted prototypes are then aligned with the base session classifier through a calibration process. The calibrated prototypes, $\bar{W}_c^t$, are adopted as the classifier for the incremental classes. Once $\bar{W}_c^t$ is obtained, it is incorporated into the classifier matrix, $[\tilde{W}_c^0, ..., \bar{W}_c^{t-1}, \bar{W}_c^t]$, and subsequently utilized in Eq. (3) for the evaluation process.

4. Experiments

4.1. Datasets

This study utilized two widely-used KWS datasets: Google Speech Commands v2 (GSCv2) [23] and the Multilingual Spoken Words Corpus (MSWC) [24]. GSCv2 contained 35 keywords and a total of 105,829 one-second utterances sampled at 16 kHz. The official train-test split was used for evaluation. The 20 keywords with the highest word counts were selected to ensure sufficient data for the base session. The remaining 15 keywords were allocated to the incremental session with a 3-way 5-shot setting. MSWC was a multilingual KWS dataset derived from Common Voice [25] using forced alignment. For this study, we selected the English portion's top 200 keywords (highest word counts). Each keyword had 4,000 utterances, split evenly between training and testing sets. The 100 keywords with the highest word counts were used in the base session (ensuring sufficient data), and the remaining 100 keywords were used for the incremental session with a 10-way 5-shot setting.

4.2. Experimental Setup

This study employed the BC-ResNet-8 [3] as the feature encoder for all methods due to its proven effectiveness in KWS tasks. The BC-ResNet-8 produced 256-dimensional output embeddings (for detailed architectural information, please refer to [3]). Baseline methods were reconfigured from their official implementations to align with KWS tasks, and experiments with the FILL method were conducted using the TEEN code as a foundation. In the CEC method, image data rotation (used in its original implementation) was replaced with SpecAugment [26] to maintain suitability for audio input.

For training, all utterances were standardized to 1-second lengths. GSCv2's audio data was used at its native 16 kHz sampling rate, while the MSWC dataset was downsampled to 16 kHz. To enhance model robustness, the noise was added to the training set of each dataset: GSCv2 utilized official noise [23], and the DEMAND dataset [27] served as noise for MSWC. A log mel spectrogram with 40 mel bins was used as the input feature. The base training session employed a batch size of 128 across 100 epochs, using the Adam optimizer with an initial learning rate of 0.005 adjusted via a cosine annealing schedule. Hyperparameters were configured as follows: Eq. (1) α = 1.0, Eq. (4) β = 0.5, and Eq. (6) η = 16.

4.3. Evaluation Metrics

We employed top-1 accuracy as the primary metric for evaluating the performance of each method across sessions. To provide a comprehensive evaluation, we utilized two key metrics: average accuracy (AA) and performance dropping rate (PD). AA, calculated as $\frac{1}{T}\sum_{t=1}^{T} A_t$ (where A_t represents top-1 accuracy for session t), provides an overall performance measure across all sessions. PD, defined as the difference between the base session accuracy and the final session accuracy $A_0 - A_T$, quantifies how much the model's performance declines over time. To ensure statistical significance, each incremental session was repeated ten times with different random N-way K-shot data, and the results presented are the average values of these repetitions.

5. Results

5.1. Experimental results

Table 1 presented a comprehensive performance comparison between FILL and four baseline methods on the MSWC dataset. Finetune, which simply retrained the entire model on new class data, experienced severe catastrophic forgetting. Its accuracy dropped significantly to 6.44% in the final session, resulting in a high PD of 71.65% and a low AA of 25.37%. These results demonstrated the limitations of this approach. CEC, despite starting with the lowest accuracy, maintained performance across sessions with a respectable 1.02% higher accuracy than FACT in the last session. While its PD was the second-best among the baselines, its low initial accuracy had to be considered when evaluating this improvement. FACT achieved the

Table 1: *Results of experiments on the MSWC dataset*

Method	Accuracy in each session (%) ↑											AA↑	PD↓
	0	1	2	3	4	5	6	7	8	9	10		
Finetune	79.60	55.46	39.24	26.20	22.02	16.39	10.19	9.07	7.59	6.91	6.44	25.37	71.65
CEC	76.13	73.82	71.49	69.87	68.56	67.18	65.66	64.29	64.03	63.10	62.49	67.87	13.64
FACT	79.26	**75.21**	72.44	70.39	68.84	67.20	65.42	63.84	63.36	62.23	61.47	68.15	17.79
TEEN	79.11	73.14	71.11	69.84	68.84	67.89	66.61	65.30	65.36	64.54	64.17	68.72	14.94
FILL	**79.94**	74.82	**72.83**	**71.59**	**70.76**	**69.96**	**68.68**	**67.45**	**67.44**	**66.66**	**66.35**	**70.59**	**13.59**

Table 2: *Results of experiments on GSCv2 dataset*

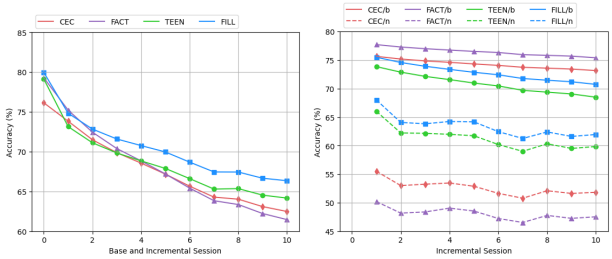
Method	Accuracy in each session (%) ↑						AA↑	PD↓
	0	1	2	3	4	5		
Finetune	97.80	81.06	60.60	56.53	48.24	38.66	63.82	59.14
CEC	97.40	95.14	92.63	90.38	87.61	84.53	91.28	12.87
FACT	**97.88**	95.17	92.42	89.45	86.25	83.24	90.74	14.64
TEEN	97.78	95.41	92.95	90.36	87.41	84.77	91.45	13.01
FILL	97.86	**95.56**	**93.17**	**91.01**	**88.16**	**85.20**	**91.83**	**12.66**

second-highest AA among the baselines but significantly declined as sessions progressed. It recorded the worst PD of 17.79% (excluding Finetune). TEEN outperformed other baselines after session 3, consistently demonstrating improved accuracy and achieving the highest AA of 68.72%. Our proposed method, FILL, outperformed all other methods in 10 of 11 sessions. It achieved the highest accuracy of 79.94% in the base session, demonstrating the effectiveness of its calibration technique in simulating the base session. Moreover, FILL exhibited a 2.18% performance increase over TEEN in the final session, showcasing its superior ability to preserve performance across incremental sessions. These results highlighted FILL's strength in addressing the challenges of keyword spotting in the FSCIL setup.

Table 2 presented a performance comparison between FILL and four baseline methods on the GSCv2 dataset. As expected, Finetune exhibited a significantly higher PD of 59.14%, showcasing this approach's limitations and highlighting the importance of FSCIL methods. CEC, despite starting with the lowest accuracy, performed better than FACT in the later sessions, suggesting better adaptability to new classes. However, its overall accuracy across sessions was lower compared to TEEN and FILL. FACT achieved the highest initial accuracy, demonstrating its strength in learning base classes. Yet, its performance declined significantly from session 5 onwards and fell behind FILL by 4.88% in the final session. This indicated challenges in effectively handling new classes. TEEN, though less accurate than FACT initially, demonstrated higher performance throughout the incremental sessions. This suggested that its prototype calibration technique aided in learning new classes even with lower base session accuracy. Our proposed method, FILL, achieved close to FACT's initial accuracy (within 0.02%) while exceeding all methods across the incremental sessions. With the highest AA of 91.83% and the lowest PD of 12.66%, FILL performed better in mitigating catastrophic forgetting while minimizing overfitting on new classes.

5.2. Ablation Study

We compared FSCIL methods by separating the accuracy of base classes from those learned incrementally. Figure 3(b) revealed the accuracy of each method on base and incremental



(a) *all classes* (b) *base and new classes*

Figure 3: *Accuracy per session on MSWC dataset. Figure (a) shows the accuracy for all classes, while Figure (b) presents the accuracy for the incremental session only. In Figure (b), /b denotes the base class, and /n denotes the new class.*

classes separately. CEC and FACT maintained high accuracy on base classes but performed poorly on new classes. This highlighted that conventional FSCIL methods prioritized the base session, potentially neglecting new class learning. TEEN exhibited the lowest accuracy for base classes throughout all sessions, but showed higher performance on new classes compared to conventional FSCIL methods. Our FILL model demonstrated superior overall performance compared to TEEN. While exhibiting slightly lower accuracy on base classes than CEC and FACT, it significantly outperformed them on new classes, aligning with the primary objective of FSCIL. Furthermore, FILL handled base classes more effectively than TEEN, suggesting further potential for improvement in this area.

6. Conclusion

In this paper, we adopted the FSCIL to the keyword spotting (KWS) domain to address the challenge of learning new keywords incrementally. Moreover, we proposed FILL, a novel FSCIL method that employs prototype calibration across sessions to facilitate learning new classes while mitigating the forgetting of previously acquired knowledge. Experimental results on the GSCv2 and MSWC datasets demonstrate that FILL outperformed existing FSCIL methods, achieving superior average accuracy in few-shot class-incremental learning scenarios.

7. Acknowledgement

This work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.RS-2023-00302424) and the National Supercomputing Center with supercomputing resources including technical support (No.TS-2024-RE-0034).

8. References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2014, pp. 4087–4091.

[2] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword transformer: A self-attention model for keyword spotting," in *Proc. INTERSPEECH*, 2021, pp. 4249–4253.

[3] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," in *Proc. INTERSPEECH*, 2021, pp. 4538–4542.

[4] S. Yang, B. Kim, I. Chung, and S. Chang, "Personalized keyword spotting through multi-task learning," in *Proc. INTERSPEECH*, 2022, pp. 1881–1885.

[5] W.-T. Kao *et al.*, "On the efficiency of integrating self-supervised learning and meta-learning for user-defined few-shot keyword spotting," in *Proc. IEEE Spoken Language Technology Workshop* (SLT), 2023, pp. 414–421.

[6] D. Lee, M. Kim, S. H. Mun, M. H. Han, and N. S. Kim, "Fully unsupervised training of few-shot keyword spotting," in *Proc. IEEE Spoken Language Technology Workshop* (SLT), 2023, pp. 266–272.

[7] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2017, pp. 2001–2010.

[8] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2019, pp. 831–839.

[9] Y. Huang, N. Hou, and N. F. Chen, "Progressive continual learning for spoken keyword spotting," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2022, pp. 7552–7556.

[10] Y. Xiao, N. Hou, and E. S. Chng, "Rainbow keywords: Efficient incremental learning for online spoken keyword spotting," in *Proc. INTERSPEECH*, 2022, pp. 3764–3768.

[11] A. Parnami and M. Lee, "Few-shot keyword spotting with prototypical networks," in *Proc. International Conference on Machine Learning Technologies* (ICMLT), 2022, pp. 277–283.

[12] X. Tao *et al.*, "Few-shot class-incremental learning," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2020, pp. 12 183–12 192.

[13] T. Ahmad *et al.*, "Few-shot class incremental learning leveraging self-supervised features," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2022, pp. 3900–3910.

[14] C. Zhang *et al.*, "Few-shot incremental learning with continually evolved classifiers," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2021, pp. 12 455–12 464.

[15] D.-W. Zhou *et al.*, "Forward compatible few-shot class-incremental learning," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR), 2022, pp. 9046–9056.

[16] Q.-W. Wang, D.-W. Zhou, Y.-K. Zhang, D.-C. Zhan, and H.-J. Ye, "Few-shot class-incremental learning via training-free prototype calibration," in *Proc. Advances in Neural Information Processing System* (NeurIPS), vol. 36, 2023.

[17] D.-W. Zhou *et al.*, "Few-shot class-incremental learning by sampling multi-phase tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (TPAMI), vol. 45, pp. 12 816–12 831, 2023.

[18] W. Xie, Y. Li, Q. He, W. Cao, and T. Virtanen, "Few-shot class-incremental audio classification using adaptively-refined prototypes," in *Proc. INTERSPEECH*, 2023, pp. 301–305.

[19] Y. Li, W. Cao, J. Li, W. Xie, and Q. He, "Few-shot class-incremental audio classification using stochastic classifier," in *Proc. INTERSPEECH*, 2023, pp. 4174–4178.

[20] S. Tian *et al.*, "A survey on few-shot class-incremental learning," *Neural Networks*, vol. 169, pp. 307–324, 2024.

[21] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, 1989, vol. 24, pp. 109–165.

[22] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Advances in Neural Information Processing System* (NeurIPS), vol. 30, 2017, pp. 4080–4090.

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

[24] M. Mazumder *et al.*, "Multilingual spoken words corpus," in *Proc. Advances in Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[25] R. Ardila *et al.*, "Common voice: A massively-multilingual speech corpus," in *Proc. Language Resources and Evaluation* (LREC), 2020, pp. 4211–4215.

[26] D. S. Park *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," *Proc. INTERSPEECH*, pp. 2613–2617, 2019.

[27] J. Thiemann, N. Ito, and E. Vincent, "The diverse environments multi-channel acoustic noise database (demand): A database of multichannel environmental noise recordings," in *Proc. Meetings on Acoustics*, vol. 19, 2013.