Online Continual Learning in Keyword Spotting for Low-Resource Devices via Pooling High-Order Temporal Statistics

Umberto Michieli, Pablo Peso Parada and Mete Ozay

Samsung Research UK

{u.michieli, p.parada, m.ozay}@samsung.com

Abstract

Keyword Spotting (KWS) models on embedded devices should adapt fast to new user-defined words without forgetting previous ones. Embedded devices have limited storage and computational resources, thus, they cannot save samples or update large models. We consider the setup of embedded online continual learning (EOCL), where KWS models with frozen backbone are trained to incrementally recognize new words from a non-repeated stream of samples, seen one at a time. To this end, we propose Temporal Aware Pooling (TAP) which constructs an enriched feature space computing high-order moments of speech features extracted by a pre-trained backbone. Our method, TAP-SLDA, updates a Gaussian model for each class on the enriched feature space to effectively use audio representations. In experimental analyses, TAP-SLDA outperforms competitors on several setups, backbones, and baselines, bringing a relative average gain of 11.3% on the GSC dataset.

Index Terms: Online Continual Learning, Keywords Spotting, Audio Classification, Embedded Devices.

1. Introduction

Recently, deep networks have arrived on user devices to support popular tasks as Virtual Assistant (VA) interaction. In this paper, we focus on Keyword Spotting (KWS), which detects keywords in audio streams [1, 2] and plays a key role, *e.g.*, to activate VAs with keywords [3] or provide commands [4]. Soon, users manifested the need for KWS models to recognize new user-defined words; however, despite the outstanding results of deep networks on standard benchmarks, they struggle to adapt.

Continual Learning (CL) emerged to adapt models to a target label space, unknown *a-priori*. Its main challenge is *forgetting* previous classes when learning new ones. CL gained wide interest to avoid accessing personal data, however, most methods demand large computational (*e.g.*, computing Fisher matrix [5]), or memory (*e.g.*, storing replay data [6]) resources, which are unavailable on resource-constrained devices [7, 8]. Standard CL does not allow for fast adaptation. Ideally, a data point should be used immediately without storing it. These problems are addressed by: (1) a single-epoch training, *i.e.*, Online CL (OCL) [9], to avoid storing samples of current task; (2) small batch sizes and (3) frozen backbone, *i.e.*, OCL for Embedded systems (EOCL) [10, 11], to comply with fast adaptation on low-resource devices. We focus on EOCL for KWS with no exemplar memory, where KWS models are partially updated with unitary batch size of data sampled from a non-repeated stream.

Non-online CL has been applied to KWS [12], combining data augmentation, replay and knowledge distillation; and ASR [13], where the term *online* is used with a different meaning than in the current literature [9], using 10 epochs. [14]
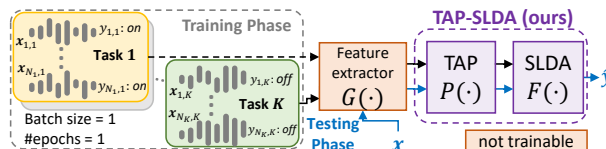

Figure 1: *Our TAP-SLDA approach proposed for EOCL.*

presents a progressive CL method for KWS where different subnetworks are trained for different sets of keywords and a shared memory is used to store learnt features and perform knowledge transfer. Unlike our setup, [14] assumes task index known at inference time to select a proper sub-network. [12] and [14] train all parameters with large batch sizes and pre-train the models on 50% keywords. Standard CL methods (*e.g.*, EWC [5] and SI [15]) are not applicable in our EOCL scenario because they assume: 1) training with large batch sizes; 2) storage of samples to perform many training epochs (not feasible on edge devices processing sensitive data); 3) updating whole feature extractors with many trainable parameters; 4) having one or more additional classification models for parameter regularization (*e.g.*, through knowledge distillation). They perform poorly when evaluated in our setup [9], hence they have been discarded.

Despite the progress on CL, EOCL for KWS remains unexplored in the literature, to the best of our knowledge. OCL methods have been recently developed in vision tasks [9], mainly based on memory [6, 16] and class prototypes [17, 10]. In this work, we propose an effective approach for EOCL KWS without memory. Our contributions are summarized as follows:

- We tackle the unsolved problem of EOCL for KWS for embedded devices with no storage and low computation power.
- Our method (TAP-SLDA) employs a new statistical pooling to extract enriched temporal information from speech features and a Gaussian-based classifier to model class representations on the enriched space with a shared covariance matrix, *i.e.*, Streaming Linear Discriminant Analysis (SLDA) [18].
- TAP-SLDA outperforms multiple OCL methods and pooling mechanisms in a wide range of experiments, showing 9.9% relative average gain compared to the best competitor.

2. Our Method

The proposed method (Fig. 1) comprises: i) a feature extractor $G(\cdot)$, ii) a pooling mechanism $P(\cdot)$, and iii) a classifier $F(\cdot)$. We incrementally train a model $F(P(G(\mathbf{x}_{n,k})))$ to generate predictions $\hat{y}_{n,k}$ via supervised OCL over K subsequent tasks, where $\mathbf{x}_{n,k}$ is the n-th ($n = [N_k]$) audio signal of the k-th task ($k = [K]$). $G(\cdot)$ is a pre-trained backbone (*e.g.*, Wav2Vec2 [19]) and $F(\cdot)$ is a classifier. Thus, we update $F(\cdot)$ in an embedded online fashion (*i.e.*, processing one example at a time

with no repetitions) using a fixed $G(\cdot)$. In Sect. 4, we consider the most challenging case of one single class per task.

To perform pooling $P(\cdot)$, we propose a Temporal Aware Pooling (TAP) which computes and concatenates the first R statistical moments from the output of $G(\cdot)$. Such moments characterize the distribution $\mathcal{G}$ of $g \triangleq G(\mathbf{x}_{n,k}) \in \mathbb{R}^{t \times d}$ where $t \times d$ is the temporal $\times$ feature size, and thus capture rich temporal statistics that improve keyword recognition. More formally,

$$P(g) = \left\| \left(\mu, E_{\mathcal{G}}\left[(g-\mu)^2\right]^{\frac{1}{2}}, \left\|_{r=3}^{R} E_{\mathcal{G}}\left[\frac{g-\mu}{E_{\mathcal{G}}\left[(g-\mu)^2\right]^{\frac{1}{2}}}\right]^r \right), \quad (1)$$

where $E_{\mathcal{G}}[\cdot]$ is the expectation on $\mathcal{G}$, μ the empirical mean, and $\|$ the concatenation. That is, $P(g) \in \mathbb{R}^{R \cdot d}$ concatenates the first R moments of g. In speaker verification: [20] shows that 3rd-4th moments alone are not useful; [21] uses $R=4$ for auxiliary tasks. In our case, we feed $R=5$ moments to the classifier.

Lastly, we develop a classifier $F(\cdot)$ via SLDA [18]. This classifier computes a running mean feature vector per class (*i.e.*, prototypes) and one covariance matrix shared across classes, that is updated online via [22]. During inference, SLDA assigns the label of the Gaussian model of a category to the input sample. Our TAP-SLDA extracts an enriched feature space with different temporal statistics, which encode information about evolution of phonetic sounds within each word [19]. Consequently, the covariance matrix shared among classes allows to identify common relationships between the temporal statistics among different words which helps to recognize new unseen words. We hypothesise that features of different classes have similar distribution of 1st moments, while higher moments capture the difference, as illustrated in Fig. 2 and quantified in Sec. 4.

OCL baselines to update $F(\cdot)$ are described next. **Fine-Tuning (FT)** updates a linear output layer using stochastic gradient descent and cross-entropy loss with no mechanisms to prevent forgetting. **Online Perceptron (PRCP)** keeps one weight vector for each class [10], initialized to the first sample of the class. After that, when the model misclassifies a new sample, the class weight vector and the weight vector of the misclassified class with the highest score are updated. During inference, the label is assigned by taking the $\arg\max$ over the dot product between weights and input vector. **Nearest Class Mean (NCM)** stores a class-wise running mean feature vector [6, 17, 10, 23]. At inference on a new sample, NCM assigns the label of the nearest centroid using Euclidean distance. **Online CBCL** extends NCM with multiple class centroids [24]. At inference, CBCL searches for the weighted nearest neighbor, where class weights are inversely proportional to the number of samples seen so far for that class. We use the default parameters suggested in [25]. **Streaming One-vs-Rest (SOvR)** measures how close a new input is to a class mean vector while also considering its distance to examples from other classes [10], which is reminiscent of SVMs. **SQDA** extends SLDA estimating one covariance matrix for each class (*i.e.*, Quadratic). The drawbacks are increased memory consumption and need for many samples per class to estimate reliable covariance matrices [26]. **Streaming Gaussian Naïve Bayes (SNB)** estimates a running variance vector per class [27] (*i.e.*, diagonal covariance matrices assuming independent features). It requires significantly less memory than SQDA. **Online iCaRL** stores a class-balanced memory buffer [6, 28, 18] and randomly replaces an example from the most represented class with a new sample when the buffer is full. During training, it randomly draws examples from the buffer, combines them with the new sample and updates a single linear output layer. While effective, it stores sensitive samples.
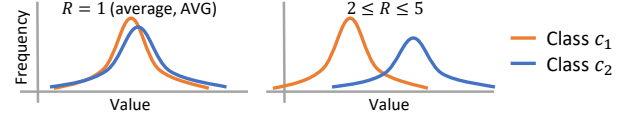

Figure 2: *Distribution of statistical moments of $\mathcal{G}$ for $c_1 \neq c_2$.*

Pooling baselines compared against our TAP are described next: **AVG** is average pooling [29]; **MAX** is max pooling [30]; **MIX** takes a linear combination of AVG and MAX [31]; **STOCHASTIC** selects activations based on probabilities and further calculates output based on multinomial distribution [32], inspired by dropout; **RAP** concatenates top-k% features to discover information spread over the time [33]; **AVGMAX** concatenates AVG and MAX pooling [34]; **TSTP** concatenates first and second order moments [35]; **TSDP** considers only the second moment [35]; **L_p-norm** computes average of p-th order non-central moment [36], while we employ high order central moments; and **iSQRT-COV** computes second-order covariance pooling via iterative matrix square root normalization [37].

3. Implementation Details

Datasets. We consider two datasets. Each task includes only keywords of the same class (*i.e.*, samples are sorted by label). *GSC v2* [38] comprises $\sim$100k utterances and 35 different English words. The number of elements per word is unbalanced ranging from 1.5k to 4k. *MSWC* [39] contains 23.4M spoken words and over 340k different words in 50 languages. We picked the 5 most represented languages (*i.e.*, *en*, *fr*, *ca*, and *rw*) and we created 3 micro-sets for each language with different number of words $N = \{25, 50, 100\}$. Each micro-set includes the N most frequent words, randomly selecting 1k samples for each word. *Dev* and *test* sets are also filtered for each case to include only the N words present in the *train* set [40].

Unless otherwise stated, we consider a class-IID setup, where all samples are ordered by class and shuffled within each class. **Models.** We use 6 recent speech recognition backbones as $G(\cdot)$: Wav2Vec2-B/L [19], HuBERT-B/-L/-XL [41] and Emformer-B [42]. All models are pre-trained [43] on LibriSpeech [44]. **Metrics.** We evaluate plasticity and forgetting of EOCL methods [9] on test sets, at the end of training. We compute accuracy (Acc %, $\uparrow$) to assess the final model performance on all classes. We compute the relative gain of Acc_2 compared to Acc_1 with respect to the available room of improvement by $(\text{Acc}_2 - \text{Acc}_1)/(100 - \text{Acc}_1)$. We measure: (1) stability by backward transfer (BwT, $\uparrow$), which tracks the influence that learning a new task has on the preceding tasks' performance; (2) forgetting (Forg, $\downarrow$), which averages the difference of class-wise Acc achieved at the last step and the best class-wise Acc achieved previously; (3) plasticity (Pla, $\uparrow$) as the average Acc achieved on each task evaluated right after learning that task [14]. Memory usage is discussed directly in Sec. 4.

4. Experimental Results

All results are averaged over 5 class orderings, although standard deviation is not always shown for readability. Offline bounds are not shown since $F(\cdot)$ is different for each approach. **TAP-SLDA vs. OCL methods** is shown in Tab. 1. Some methods (FT and PRCP) largely overfit to the last seen data and cannot mitigate forgetting (low Acc/BwT and high Forg/Pla). SNB and SOvR improve on HB-B and XL thanks to the centroid-based inference mechanism (Pla dropped significantly). NCM

Table 1: *Results of 10 OCL methods and 6 backbones on the GSC dataset. Each entry is averaged over 5 distinct class ordering.*

	W2V-B				W2V-L				Emf-B				HB-B				HB-L				HB-XL			
	Acc	BwT	Forg	Pla	Acc	BwT	Forg	Pla	Acc	BwT	Forg	Pla	Acc	BwT	Forg	Pla	Acc	BwT	Forg	Pla	Acc	BwT	Forg	Pla
FT	$2.5_{\pm1.0}$	1.2	98.5	**100**	$2.5_{\pm1.0}$	1.3	98.5	**100**	$2.7_{\pm0.8}$	1.5	97.6	**99.8**	$2.5_{\pm1.0}$	1.2	98.5	**100**	$2.5_{\pm1.0}$	1.2	98.5	**100**	$2.5_{\pm1.0}$	1.2	98.5	**100**
PRCP [10]	$2.5_{\pm1.0}$	1.2	98.4	**100**	$2.5_{\pm1.0}$	1.3	98.4	**100**	$2.9_{\pm0.7}$	2.2	95.8	97.5	$2.5_{\pm1.0}$	1.4	98.4	**100**	$2.5_{\pm1.0}$	1.2	98.4	**100**	$2.6_{\pm1.0}$	1.3	98.4	**100**
SNB [27]	$3.7_{\pm0.0}$	6.5	38.6	15.7	$9.4_{\pm2.4}$	9.2	36.1	16.6	$7.2_{\pm0.0}$	13.1	24.8	22.8	$77.4_{\pm0.0}$	80.9	18.1	84.2	$6.5_{\pm0.0}$	12.0	37.1	23.7	$57.9_{\pm0.0}$	60.3	23.8	65.2
SOvR [10]	$1.8_{\pm0.0}$	6.2	39.6	14.5	$1.8_{\pm0.0}$	6.2	31.6	15.8	$4.9_{\pm0.0}$	7.3	24.4	14.5	$19.1_{\pm0.0}$	29.8	41.2	42.5	$14.0_{\pm0.0}$	21.8	34.8	32.9	$15.7_{\pm0.0}$	22.6	35.3	33.8
NCM [23]	$67.5_{\pm6.0}$	74.0	13.2	77.3	$69.5_{\pm7.0}$	76.0	12.1	80.0	$8.8_{\pm0.0}$	14.1	23.6	22.0	$83.9_{\pm0.0}$	86.0	8.2	89.1	$46.7_{\pm0.0}$	55.4	22.8	62.4	$62.5_{\pm0.0}$	66.7	17.3	72.2
SLDA [18]	$82.4_{\pm0.1}$	84.2	8.0	87.0	$81.6_{\pm0.1}$	83.8	8.3	86.8	$23.2_{\pm0.0}$	32.9	23.7	41.9	$94.2_{\pm0.0}$	94.9	3.8	96.2	$85.5_{\pm0.0}$	88.2	8.3	90.8	$93.3_{\pm0.0}$	94.1	5.1	95.4
SQDA [26]	$80.6_{\pm2.5}$	78.2	5.7	81.2	$80.5_{\pm2.4}$	76.9	**5.1**	80.6	$24.3_{\pm0.7}$	21.9	**17.3**	31.5	$90.0_{\pm3.4}$	87.8	**2.8**	90.0	$67.4_{\pm4.4}$	59.4	**4.4**	64.8	$83.0_{\pm2.1}$	73.9	**0.0**	76.8
TAP-SLDA (ours)	$\mathbf{89.9}_{\pm0.0}$	**91.8**	5.6	93.7	$\mathbf{90.0}_{\pm0.0}$	**91.7**	5.4	93.4	$\mathbf{50.8}_{\pm0.3}$	**58.8**	20.3	65.8	$\mathbf{95.7}_{\pm0.0}$	**96.0**	3.0	96.9	$\mathbf{90.8}_{\pm0.0}$	**91.8**	6.1	93.6	$\mathbf{95.5}_{\pm0.0}$	**95.8**	3.4	96.6
iCaRL [6]	$76.9_{\pm1.0}$	79.1	14.7	83.6	$73.6_{\pm1.8}$	78.0	17.4	83.5	$18.2_{\pm0.3}$	26.9	28.8	44.7	$93.7_{\pm0.1}$	94.6	4.2	96.7	$78.5_{\pm0.5}$	83.1	12.8	85.1	$92.9_{\pm0.3}$	93.8	5.3	95.5
Avg	47.5	49.6	33.5	73.0	48.1	50.1	32.5	73.7	15.1	19.3	38.0	46.2	64.3	65.9	28.6	88.5	44.1	46.9	34.6	71.6	56.8	57.7	30.4	80.8

and CBCL achieve the same results due to the default parameter configuration of CBCL suggested in [24]. They improve results on all backbones (medium Acc/BwT/Forg and high Pla), except for the memory-efficient Emf-B. SLDA significantly improves final performance thanks to the shared covariance matrix among classes (high Acc/BwT/Pla and low Forg). SQDA achieves lower results than SLDA: it can preserve previous tasks (lowest Forg, *i.e.*, good stability), however, it cannot adapt well to the new tasks (Pla lower than SLDA, *i.e.*, worse plasticity). Storing one covariance matrix per each class makes SQDA not only storage demanding, but also ineffective if data is scarce and reliable statistics cannot be extracted for each class, as highlighted by high standard deviation. Hence, we selected SLDA as the best method, and we applied our TAP pooling on top of it (*i.e.*, TAP-SLDA). TAP-SLDA shows strong Acc gains on every backbone, improving on average by relative 37.8% compared to SLDA (76.7% vs. 85.5%). On Emf-B, the relative gain is 35.0% compared to the runner-up SQDA, and 25.9% compared to the baseline SLDA. Looking at BwT, Pla and Forg metrics, we observe that our TAP-SLDA achieves the best trade-off between plasticity in learning new tasks and forgetting past ones. The key element in TAP-SLDA is the computation of the high-order feature space that provide useful temporal characteristics to the subsequent centroids- and covariance- based modeling. That is, providing richer, temporal-aware statistics of input waveform is beneficial for EOCL KWS, where models must adapt fast exploiting all the information and relate it to previous knowledge.

Finally, we use a simple replay method, iCaRL, with 1k buffer size. iCARL is competitive, however: 1) it doubles the batch size as one new sample is coupled with a replay one, 2) it stores sensitive data, thus raising privacy/storage concerns, 3) our method surpasses it by large margin (85.5% vs. 72.3%).

TAP improves every OCL method. To show benefits of TAP to EOCL KWS, we include it on top of each OCL method in Tab. 2. Comparing the results against Tab. 1 emerges that TAP improves OCL methods almost every time (7 exceptions out of 54 cases). Sometime, we observe a large gain up to ∼60%. This is the consequence of using high-order statistics. As preluded in Fig. 2, we compute the mean Wasserstein distance between distributions of features of pair-wise classes to quantify this effect. The distance of the 1st moments (0.09 ± 0.02) is considerably lower than the mean distance of the 2nd to the 5th moments (0.22 ± 0.07) indicating that classes are more entangled in the feature space of the 1st moment (low distance) than of higher moments. Thus, high moments improve the classification score.

TAP vs. other pooling schemes is evaluated in Tab. 3 for the best performing OCL method (SLDA). STOCH and L2/L3 pooling underperform AVG. RAP outperforms AVG and is

Table 2: *Acc of OCL methods employing our TAP on GSC. Improvement with respect to original methods within brackets.*

TAP+	W2V-B	W2V-L	Emf-B	HB-B	HB-L	HB-XL	Avg
FT	5.4	6.1	2.9	2.7	2.7	2.7	3.8 (+1.2)
PRCP	3.5	4.6	3.0	2.8	2.8	2.9	3.3 (+0.7)
SNB	3.9	7.1	9.3	84.1	6.9	59.9	28.5 (+2.3)
SOvR	51.3	60.9	5.8	54.3	14.9	49.6	39.5 (+29.9)
NCM	78.2	79.8	12.1	87.2	44.5	84.9	64.5 (+8.0)
CBCL	75.9	77.3	12.0	88.7	48.0	86.1	64.7 (+8.2)
SLDA	**89.9**	**90.0**	**50.8**	**95.7**	**90.8**	**95.5**	**85.5** (+8.8)
SQDA	85.5	84.0	48.7	88.8	67.1	82.7	76.1 (+5.1)
iCaRL	82.9	85.7	31.0	90.9	76.9	90.8	76.4 (+4.1)
Avg	52.9	55.1	19.5	66.1	39.4	61.7	

Table 3: *Acc of SLDA under different pooling strategies on GSC (best in **bold**, runner-up underlined).*

	W2V-B	W2V-L	Emf-B	HB-B	HB-L	HB-XL	Avg
AVG [29]	82.4	81.6	23.2	94.2	85.5	93.3	76.7
MAX [30]	87.7	88.3	34.9	94.8	87.2	94.1	81.2
MIX (50%) [31]	87.8	87.3	31.1	94.7	87.3	94.0	80.4
STOCH [32]	80.9	77.6	24.6	85.3	64.5	77.9	68.4
L2 [36]	79.0	79.7	17.4	92.9	74.4	89.7	72.2
L3 [36]	79.3	81.2	15.9	92.4	70.4	89.1	71.4
RAP (10%) [33]	86.5	86.9	36.3	94.8	87.5	93.8	81.0
AVGMAX [34]	<u>89.1</u>	<u>89.6</u>	44.8	<u>95.2</u>	<u>89.1</u>	<u>94.7</u>	<u>83.8</u>
iSQRT-COV [37]	80.3	80.3	**55.1**	92.4	83.8	90.3	80.4
TSDP [35]	83.9	83.6	32.4	94.4	84.9	93.9	78.9
TSTP [35]	87.4	87.6	39.1	95.1	88.0	94.5	82.0
TAP (ours)	**90.0**	**90.0**	<u>50.8</u>	**95.7**	**90.8**	**95.5**	**85.5**
Avg	84.5	84.5	33.8	93.5	82.8	91.7	

comparable to MAX. AVGMAX is highly effective compared to AVG and MAX used alone. To extract more useful temporal clues, covariance (iSQRT-COV) or variance (TSDP and TSTP) of features are used. iSQRT-COV is especially useful on Emf-B: overall, it outperforms AVG but not MAX. Using only standard deviation (TSDP) is less useful than using it in conjunction with AVG (*i.e.*, TSTP). TSTP shows robust gains compared to AVG in every scenario by preservation of temporal statistics after pooling. Our TAP moves from similar ideas and encodes further higher-order statistics in the pooled output. TAP is the best performing method on every architecture except for Emf-B where it ranks second. On average, it improves by relative 37.8% compared to AVG, by 10.5% compared to AVGMAX and by 19.4% compared to TSTP. We confirm the superiority of

Table 4: *Acc and increase of parameters (%) over the backbone (Δ_p). Metrics averaged over the 6 networks on GSC. We use TAP with variable R (R=1 is AVG). TAP has minimal footprint.*

TAP+	$R=1$		$R=2$		$R=3$		$R=4$		$R=5$		$R=6$	
	Acc	Δ_p	Acc	Δ_p	Acc	Δ_p	Acc	Δ_p	Acc	Δ_p	Acc	Δ_p
FT	1.6	0.01	1.5	0.01	1.5	0.02	1.5	0.02	**3.8**	0.03	2.6	0.04
NCM	52.1	0.01	62.8	0.01	63.5	0.02	64.0	0.02	**64.5**	0.03	56.7	0.04
SLDA	76.7	0.10	82.0	0.10	84.4	0.11	85.2	0.12	**85.5**	0.12	84.9	0.13

Table 5: *Acc w/o and w/ pre-training evaluated on GSC.*

	FT		TAP-FT		NCM		TAP-NCM		SLDA		TAP-SLDA	
	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
W2V-B	2.5	2.5	2.5	5.4	13.0	67.5	20.2	78.2	42.3	82.4	47.2	89.9
W2V-L	2.5	2.5	2.5	6.1	14.0	69.5	21.7	79.8	43.2	81.6	48.1	90.0

Table 6: *Acc in IID setup on GSC. Absolute improvement compared to class-IID setup is reported within brackets.*

	FT	TAP-FT	NCM	TAP-NCM	SLDA	TAP-SLDA
W2V-B	44.6 (42.1)	78.4 (73.0)	67.5 (0)	78.2 (0)	82.5 (0.1)	90.0 (0)
W2V-L	51.7 (49.2)	81.1 (75.0)	69.5 (0)	79.8 (0)	81.6 (0)	90.0 (0)
Emf-B	13.9 (11.9)	10.4 (8.9)	8.8 (0)	12.1 (0)	23.2 (0)	50.8 (0)
HB-B	86.2 (84.7)	88.1 (86.6)	83.9 (0)	87.2 (0)	94.2 (0)	95.7 (0)
HB-L	52.6 (51.1)	67.1 (65.6)	46.7 (0)	44.5 (0)	85.5 (0)	90.8 (0)
HB-XL	86.8 (85.3)	88.1 (86.6)	62.5 (0)	84.9 (0)	93.3 (0)	95.5 (0)
Avg	56.0	68.9	52.1	64.5	76.7	85.5

Table 7: *Acc of pooling methods which increase pooled feature size. Evaluation on the class-IID setup of GSC using SLDA as CL method. Δ_{fs}: average feature size increment multiplier.*

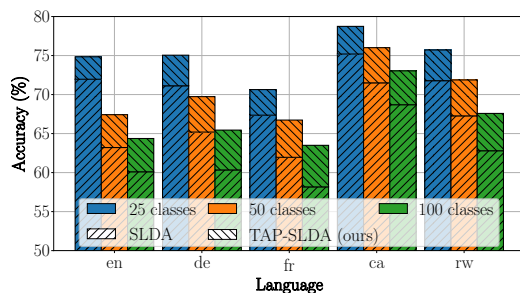|	W2V-B	W2V-L	Emf-B	HB-B	HB-L	HB-XL	**Avg**	Δ_{fs}
AVG	82.4	81.6	23.2	94.2	85.5	93.3	76.7	1
MAX	87.7	88.3	34.9	94.8	87.2	94.1	81.2	1
RAP 5%	85.7	85.8	35.1	94.5	86.8	93.8	80.3	26.8
RAP 10%	86.5	86.9	36.3	94.8	87.5	93.8	81.0	53.5
RAP 20%	85.7	85.9	36.3	94.5	86.8	93.8	80.5	107
MAXW$_2$	87.7	88.3	34.9	94.8	87.2	94.1	81.2	5
MAXW$_5$	85.8	86.5	34.9	94.7	87.1	93.9	80.5	11
MAXW$_{10}$	85.6	86.1	35.3	94.7	86.7	93.7	80.3	21
FLAT	85.1	86.2	24.6	94.3	85.7	93.5	78.2	535
TAP (R=2)	87.4	87.6	39.1	95.1	88.0	94.5	82.0	2
TAP (R=3)	89.3	89.2	47.3	95.5	89.8	95.4	84.4	3
TAP (R=4)	90.0	90.0	49.7	95.6	90.4	**95.5**	85.2	4
TAP (R=5)	90.0	90.0	**50.8**	95.7	**90.8**	95.5	**85.5**	5
TAP (R=6)	**90.1**	**90.2**	47.8	95.7	89.9	95.5	84.9	6

Figure 3: *Acc of HB-B on MSWC micro-splits. Acc of FT averaged over class sets are respectively: 1.6, 1.5, 1.6, 1.4, 1.7.*

our TAP for EOCL KWS, which is achieved by extracting rich temporal dynamics from the single iteration over the input data.
Is TAP adding large overhead? Tab. 4 studies Acc and relative % increase of parameters (Δ_p) compared to the pre-trained backbone when using TAP. We observe that: 1) $R = 5$ always brings the highest Acc for all the 3 methods; and 2) Δ_p of TAP-SLDA (0.12% for the highest Acc) shows only minimal increase compared to that of SLDA (0.10%). On average, TAP increases training (inference) time by just 2.1% (2.6%).
Randomly initialized weights. Pre-training helps speech tasks but requires abundant data and computing power. Even in case of no pre-training or fine-tuning, using TAP alone still improves the accuracy thanks to the richer temporal statistics extracted from the scarce input data as we prove in Tab. 5.
Performance on IID case (no class ordering). Tab. 6 analyzes an IID setup where samples are seen at random (*i.e.*, with no class ordering) by the CL model. On one hand, FT can greatly improve the Acc (as well as other models with high Forg on the class-IID setup of Tab. 1) because the same class is presented to the learner at multiple stages during training, thus mitigating forgetting. On the other hand, we observe that methods relying on prototypes and covariance for inference are robust and not influenced by the non-stationarity of data in this setup, thus achieving results similar to the IID setup. Importantly, TAP is still effective even when no consideration is taken: TAP-FT on IID data improves Acc by relative 29.3% over FT on average.
Is larger feature space all we need? We investigate whether increasing the size of the pooled output is enough to achieve the performance gains observed so far. We report results in Tab. 7 in terms of Acc and average feature size increment multiplier (Δ_{fs}). AVG and MAX are the baselines with no increase in size (*i.e.*, $\Delta_{fs} = 1$). RAP extracts the top-k% features: the large increase of feature size does not reflect into Acc gains. Motivated

by the success of MAX and aiming at bringing temporal awareness into pooling, we considered a $2 \cdot l$ window around MAX (*i.e.*, MAXW$_l$). MAXW$_l$ can only match the results of MAX although showing higher feature size. On the extreme, FLAT flattens all features into a long 1D vector (*i.e.*, no pooling). Despite drastic increase of feature size, these schemes cannot capture temporal dynamics of features. On the other hand, our method improves Acc thanks to the richer temporal statistics extracted from input features while maintaining practically unchanged the total number of parameters as shown in Tab. 4
Personalization to other languages. Fig. 3 shows the best performing HB-B architecture (pre-trained on unlabeled English data only) adapted online to recognize words from a different language. Acc is reduced compared to GSC, due to the harder MSWC benchmark; however, higher results can be achieved by multilingual pre-training [45]. Nonetheless, TAP-SLDA improves over the baseline on every language and class set (mean relative gain of 12.8%, and of 12.6% in the hardest case with 100 classes), therefore being extremely effective when domain of use changes, as it often happens for deployed KWS systems.

5. Conclusion

We proposed the first EOCL KWS system where only one sample is presented at a time on one single training epoch with a frozen backbone. We introduced a novel pooling mechanism (TAP) based on the first 5 moments of features to extract richer temporal-aware statistics. TAP provided the best results across different setups. In particular, our TAP-SLDA achieves 95.7% for EOCL KWS on GSSC, setting a new SOTA for the task.

6. References

[1] R. Rose and D. Paul, "A hidden Markov model based keyword recognition system," in *ICASSP*, 1990, pp. 129–132 vol.1.

[2] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *ICASSP*, 2014.

[3] I. López-Espejo, Z.-H. Tan, J. Hansen, and J. Jensen, "Deep spoken keyword spotting: An overview," *IEEE Access*, 2021.

[4] R. Tang and J. Lin, "Deep Residual Learning for Small-Foot-print Keyword Spotting," in *ICASSP*, 2018, pp. 5484–5488.

[5] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *PNAS*, vol. 114, 2017.

[6] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *CVPR*, 2017.

[7] L. Pellegrini, V. Lomonaco, G. Graffieti, and D. Maltoni, "Continual learning at the edge: Real-time training on smartphone devices," *arXiv:2105.13127*, 2021.

[8] G. Petit, A. Popescu, H. Schindler, D. Picard, and B. Delezoide, "Fetril: Feature translation for exemplar-free class-incremental learning," in *WACV*, 2023, pp. 3911–3920.

[9] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, 2022.

[10] T. L. Hayes and C. Kanan, "Online Continual Learning for Embedded Devices," *CoLLAs-2022*, 2022.

[11] G. Borghi, G. Graffieti, and D. Maltoni, "On the challenges to learn from natural data streams," *arXiv:2301.03495*, 2023.

[12] Y. Xiao, N. Hou, and E. S. Chng, "Rainbow Keywords: Efficient Incremental Learning for Online Spoken Keyword Spotting," in *Interspeech*, 2022, pp. 3764–3768.

[13] M. Yang, I. Lane, and S. Watanabe, "Online continual learning of end-to-end speech recognition models," *Interspeech*, 2022.

[14] Y. Huang, N. Hou, and N. F. Chen, "Progressive continual learning for spoken keyword spotting," in *ICASSP*, 2022.

[15] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *ICML*. PMLR, 2017.

[16] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *ICLR*, 2019.

[17] Z. Mai, R. Li, H. Kim, and S. Sanner, "Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning," in *CVPRW*, 2021.

[18] T. L. Hayes and C. Kanan, "Lifelong machine learning with deep streaming linear discriminant analysis," in *CVPRW*, 2020.

[19] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," in *NeurIPS*, vol. 33, 2020, pp. 12 449–12 460.

[20] M. Rouvier, P.-M. Bousquet, and J. Duret, "Study on the temporal pooling used in deep neural networks for speaker verification," in *EUSIPCO*. IEEE, 2021, pp. 501–505.

[21] L. You, W. Guo, L. Dai, and J. Du, "Multi-Task Learning with High-Order Statistics for X-vector based Text-Independent Speaker Verification," in *Interspeech*, 2019.

[22] S. Dasgupta and D. Hsu, "On-line estimation with the multivariate Gaussian distribution," in *ICCLT*. Springer, 2007, pp. 278–292.

[23] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *TPAMI*, vol. 35, no. 11, pp. 2624–2637, 2013.

[24] A. Ayub and A. R. Wagner, "Cognitively-inspired model for incremental learning using a few examples," in *CVPRW*, 2020, pp. 222–223.

[25] ——, "F-SIOL-310: A Robotic Dataset and Benchmark for Few-Shot Incremental Object Learning," in *ICRA*. IEEE, 2021.

[26] C. Anagnostopoulos, D. K. Tasoulis, N. M. Adams, N. G. Pavlidis, and D. J. Hand, "Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification," *SADM*, vol. 5, no. 2, pp. 139–166, 2012.

[27] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[28] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *ECCV*, 2018, pp. 233–248.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *IEEE*, 1998.

[30] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature NN*, vol. 2, no. 11, pp. 1019–1025, 1999.

[31] Q. Zhou, Z. Qu, and C. Cao, "Mixed pooling and richer attention feature fusion for crack detection," *PRL*, vol. 145, pp. 96–102, 2021.

[32] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *ICLR*, 2013.

[33] S. Bera and V. K. Shrivastava, "Effect of pooling strategy on convolutional neural network for classification of hyperspectral remote sensing images," *IET IP*, vol. 14, no. 3, pp. 480–486, 2020.

[34] J. Monteiro, M. J. Alam, and T. Falk, "On the performance of time-pooling strategies for end-to-end spoken language identification," in *LREC*, 2020, pp. 3566–3572.

[35] S. Wang, Y. Yang, Y. Qian, and K. Yu, "Revisiting the statistics pooling layer in deep speaker embedding learning," in *ISCSLP*. IEEE, 2021, pp. 1–5.

[36] J. Feng, B. Ni, Q. Tian, and S. Yan, "Geometric l_p-norm feature pooling for image classification," in *CVPR*, 2011.

[37] P. Li, J. Xie, Q. Wang, and Z. Gao, "Towards faster training of global covariance pooling networks by iterative matrix square root normalization," in *CVPR*, 2018, pp. 947–955.

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

[39] M. Mazumder, S. Chitlangia, C. Banbury, Y. Kang, J. M. Ciro, K. Achorn, D. Galvez, M. Sabini, P. Mattson, D. Kanter *et al.*, "Multilingual Spoken Words Corpus," in *NeurIPS*, 2021.

[40] Splits are available at: https://github.com/umbertomichieli/TAP-SLDA.

[41] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *TASLP*, vol. 29, pp. 3451–3460, 2021.

[42] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *ICASSP*, 2021, pp. 6783–6787.

[43] "Torchaudio models," https://pytorch.org/audio/stable/pipelines. html, Accessed: 2023-03-03.

[44] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP*. IEEE, 2015, pp. 5206–5210.

[45] C. Wang, Y. Wu, Y. Qian, K. Kumatani, S. Liu, F. Wei, M. Zeng, and X. Huang, "Unispeech: Unified speech representation learning with labeled and unlabeled data," in *ICML*, 2021, pp. 10 937–10 947.