Interspeech 2022
18-22 September 2022, Incheon, Korea

Personalized Keyword Spotting through Multi-task Learning

Seunghan Yang[1], *Byeonggeun Kim*[1], *Inseop Chung*[1,2,*], *Simyung Chang*[1]

[1]Qualcomm AI Research†, Qualcomm Korea YH, Seoul, Republic of Korea
[2]Seoul National University, Seoul, Republic of Korea
{seunghan, kbungkun, ichung, simychan}@qti.qualcomm.com

Abstract

Keyword spotting (KWS) plays an essential role in enabling speech-based user interaction on smart devices, and conventional KWS (C-KWS) approaches have concentrated on detecting user-agnostic pre-defined keywords. However, in practice, most user interactions come from target users enrolled in the device which motivates to construct personalized keyword spotting. We design two personalized KWS tasks; (1) Target user Biased KWS (TB-KWS) and (2) Target user Only KWS (TO-KWS). To solve the tasks, we propose personalized keyword spotting through multi-task learning (PK-MTL) that consists of multi-task learning and task-adaptation. First, we introduce applying multi-task learning on keyword spotting and speaker verification to leverage user information to the keyword spotting system. Next, we design task-specific scoring functions to adapt to the personalized KWS tasks thoroughly. We evaluate our framework on conventional and personalized scenarios, and the results show that PK-MTL can dramatically reduce the false alarm rate, especially in various practical scenarios.

Index Terms: keyword spotting, personalization, speaker verification, multi-task learning

1. Introduction

An always-on lightweight keyword spotting system has been exploited to wake up smart audio devices. When the system detects the keyword, the following audio stream can be uploaded to speech recognition systems [1, 2, 3, 4, 5, 6]. This process can reduce power consumption while maintaining a high recall rate and low false alarm rate. Conventional keyword spotting (C-KWS) [2, 7] aims to detect a small set of pre-defined speech signals such as wake-up words, *e.g.*, "Alexa" and "OK Google," and has been applied to the always-on keyword spotting system.

However, this system is not personalized, *i.e.*, it only focuses on pre-defined keywords and does not consider the users. To adapt the system to the users, query-by-example keyword spotting [8, 9, 10] has been proposed to allow the users to enroll their own keywords, but these approaches only adapt to new keywords, not explicitly considering the user identity. In practice, most user interactions come from the *target* users, and hence the model requires to be biased on the target users. Moreover, since the aforementioned systems ignore user information, they cannot prevent detecting general negatives of background sounds containing target keywords or other keywords having similar pronunciation to the target, *e.g.*, streaming audios from TV, online meetings, and conversations. It could lead to undesirable power consumption by unnecessarily activating the recognition systems. To address these problems, we introduce more practical, personalized KWS tasks, considering the target users enrolled in the device.

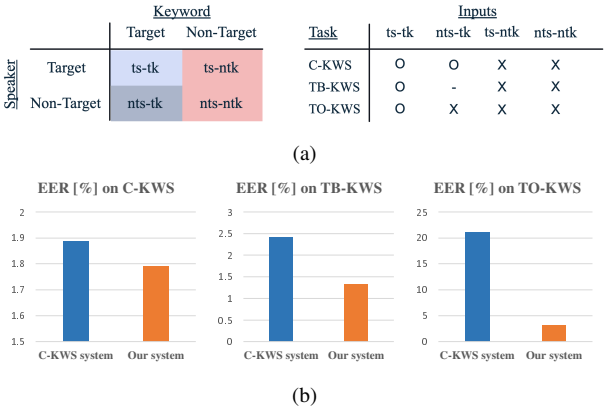


Figure 1: *(a): Inputs of keyword spotting systems can be categorized into four cases: ts-tk, nts-tk, ts-ntk, nts-ntk. TB- and TO-KWS consider nts-tk as neutral and negative, respectively. (b): The conventional C-KWS system [7] shows low equal error rate (EER) on C-KWS, but it downgrades on two personalized tasks. Our system has a capability to conduct TB- and TO-KWS, while performing comparable result on C-KWS.*

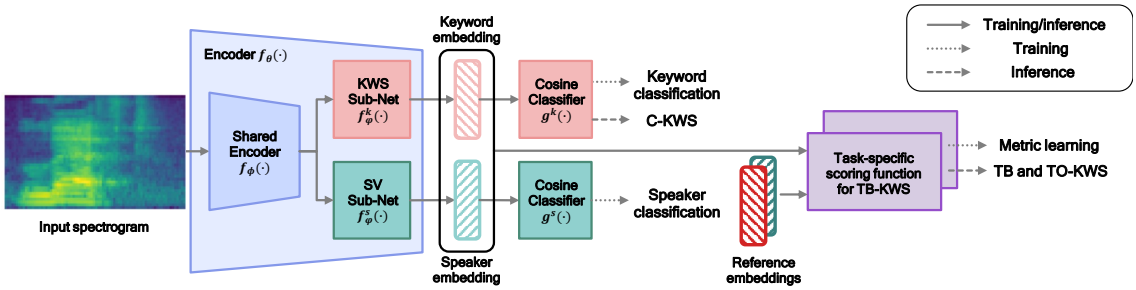
To describe personalized tasks, we categorize input utterances of the KWS system into four when a target speaker and a target keyword are given as Fig. 1a left. The four categories are *ts-tk* (utterances from the target speaker and the target keyword), *nts-tk* (utterances from a non-target speaker and the target keyword), *ts-ntk*, and *nts-ntk*. The vague one is *nts-tk*, which is the target keyword to detect but from a non-target user. We design two personalized keyword spotting tasks considering *nts-tk* differently as Fig. 1a right: Target user Biased KWS (TB-KWS) and Target user Only KWS (TO-KWS) to better focus on the target users. TB-KWS requires the model to be biased to the target users and does not explicitly consider *nts-tk*. Moreover, some applications in devices prefer to be activated by only the target users, hence TO-KWS only detects *ts-tk* and not *nts-tk*. We evaluate the C-KWS system [7] on multiple KWS tasks in Fig. 1b, and it shows performance degradation on our proposed personalized tasks since it does not consider user information.

We propose personalized keyword spotting through multi-task learning (PK-MTL) that can be selectively utilized on all three tasks, C-KWS, TB-KWS, and TO-KWS. Our PK-MTL is a two-stage system consisting of multi-task learning and task-adaptation. First, we propose to apply multi-task learning on keyword spotting and speaker verification to leverage speaker information to the keyword spotting system. With a slight increase in the number of parameters, our keyword spotting system learns both keyword and speaker characteristics, and we can get both representations in a computation-efficient

† Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. *Author completed the research in part during an internship at Qualcomm Technologies, Inc.

Copyright © 2022 ISCA

10.21437/Interspeech.2022-947



(a) Multi-task learning (b) Task-adaptation

Figure 2: ***Personalized keyword spotting through multi-task learning*** *consists of two parts. (a) A multi-task learning part: the shared representation of keyword and speaker classification is encoded in low-level layers, and sub-networks learn the characteristics of each task in high-level features. (b) A task-adaptation part: initial representations are learned by the multi-task loss function, and these representations are adapted to each task through task-specific scoring functions.*

way. Next, we introduce task-specific scoring functions to adapt learned representations to TB- and TO-KWS tasks. Since the purpose of the two tasks is different (see Fig. 1a), a function that combines the information from keyword and speaker representations suitable to each task is needed. We propose two modules, Score Combination Module (SCM) and Task Representation Module (TRM), as task-specific scoring functions. SCM is an optimization-free approach that directly combines initial representations from the multi-task learning part. The second approach is to construct a trainable Task Representation Module (TRM) that is designed to take the two representations and learn new task-specific representations for each task. With the aid of these modules, PK-MTL can be adapted to each task.

We evaluate on Google Speech Commands [11]. Upon the multiple keywords spotting backbones [2, 7, 12], our framework significantly boosts the performance of TB- and TO-KWS while performing comparable results of C-KWS, with a slight increase in the number of parameters and computation. We also test PK-MTL in a practical setting, where negative samples continuously come from TV or other sources. Specifically, we exploit WSJ-SI200 [13] and Librispeech [14] as negative samples, and the results show that our system dramatically reduces the false alarm rate with the aid of task-specific scoring functions.

2. Method

As shown in Fig. 2, the overall architecture of our system comprises two parts: multi-task learning on keyword spotting (KWS) and speaker verification (SV) and task-specific scoring functions to adapt to personalized KWS tasks.

Notations. The training data, $\mathcal{D}_{\text{train}}$, consists of labeled samples, $\{(x_i, y_i)\}_{i=1}^{|\mathcal{D}_{\text{train}}|}$, where x_i is an input audio feature, and $y_i = (y_i^k, y_i^s)$. y_i^k and y_i^s are the corresponding keyword and speaker labels, respectively. To leverage speaker characteristics to the keyword spotting system, we design a multi-task learning architecture that comprises a shared encoder, $f_\phi(\cdot)$, sub-networks for keyword spotting and speaker verification, $f^k_\varphi(\cdot)$ and $f^s_\varphi(\cdot)$, and classifiers, $g^k(\cdot)$ and $g^s(\cdot)$.

Decision process. Given a pre-defined target keyword, conventional keyword spotting system obtains a score $\psi_{i,\text{ref}}$ of a test sample x_i belonging to the given target keyword using the trained keyword classifier [2, 3]. The system accepts x_i as the positive sample when $\psi_{i,\text{ref}} > \delta$, where δ is a threshold; otherwise x_i is rejected as the negative sample. In this paper, we additionally utilize an enrolled utterance of the target user x_{ref} following personalized systems [15, 16] and leverage the similarity of x_i and x_{ref} into the score $\psi_{i,\text{ref}}$.

2.1. Multi-task learning for personalized keyword spotting

For a given backbone network, $f_\theta(\cdot)$, where $\theta = \{\phi, \varphi\}$, we adopt hard-parameter sharing [17, 18] for low-level layers, *i.e.*, the shared encoder, $f_\phi(\cdot)$. The shared encoder can learn complementary information from both KWS and SV, and the shared design is more efficient in terms of memory and computation than using separate task designs [19, 20]. Since features for KWS and SV are adversarial in high-level concepts [21], *i.e.*, keyword features are speaker-agnostic and vice versa, sub-networks, $f^k_\varphi(\cdot)$ and $f^s_\varphi(\cdot)$, are added to learn the characteristics of each task. Then, we get keyword and speaker features, $z_i^k = f^k_\varphi(f_\phi(x_i))$ and $z_i^s = f^s_\varphi(f_\phi(x_i))$. On top of that, we apply cosine similarity based classifiers [22, 23] as follows:

$$g^k(z_i^k) = \text{softmax}(w \cdot \text{sim}(z_i^k, W^k) + b), \tag{1}$$

where W^k is the learnable weight for keyword classification, sim represents cosine similarity, $\text{sim}(a, b) = a \cdot b/(||a||\,||b||)$, and w and b indicate scale and bias, respectively (for simplicity of notation, we omit the superscript k). We define a keyword classification loss by minimizing the negative log probability of the true class:

$$L_k = \sum_i -y_i^k \log g^k(z_i^k). \tag{2}$$

We get L_s for SV in the same manner and combine two task-specific loss functions as below:

$$L_{\text{mtl}} = L_k + \lambda L_s, \tag{3}$$

where λ indicates the importance of speaker information. Different from the goal of previous multi-task learning on KWS and SV [24, 25] that focuses on boosting each task, our framework focuses more on leveraging user information for better-personalized KWS. After training the network with the loss in Eq. 3, we use learned representations for TB- and TO-KWS.

2.2. Task-specific scoring functions for TB- and TO-KWS

Score Combination Module (SCM). The first approach is to obtain two scores of keyword and speaker independently and combine them directly into a task-specific score for the decision process in each personalized task. Given the enroll utterance x_{ref} of a target user y^s_{ref}, keyword and speaker scores of the test input sample x_i for the user y_t^s are calculated by:

$$\psi^k_{i,\text{ref}} = \text{sim}(z_i^k, W^k_{\text{ref}}) \;\text{ and }\; \psi^s_{i,\text{ref}} = \text{sim}(z_i^s, f^s_\varphi(f_\phi(x_{\text{ref}})),$$

where $\psi^k_{i,\text{ref}}$ is the cosine similarity score between the keyword embedding z_i^k and the trained keyword classifier weight W^k_{ref}

that can be regarded as the most representative embedding of the pre-defined target keyword. $\psi^s_{i,\text{ref}}$ is the score between the speaker embedding of the input z^s_i and the embedding of the enrolled target user $f^s_\varphi(f_\phi(x_{\text{ref}}))$. Note that target keywords are pre-defined, but speakers are not overlapped between training and test. Therefore, we use the reference embedding, z^s_{ref}, of the target user at the test time. We define $\text{SCM}(\cdot,\cdot;\alpha)$ that combines two scores, ψ^k and ψ^s, into new score for each task, ψ^{tb} and ψ^{to}. SCM can be any combination functions, but we use a simple linear combination function, $\alpha \cdot \psi^k + (1-\alpha)\cdot\psi^s$. The objective of keyword spotting is to minimize the false rejection rate (FRR) at the given false alarm rate (FAR). Here, FAR is the percentage of negative samples being incorrectly accepted, while FRR is that of positive samples being incorrectly rejected. To achieve this goal, we obtain the parameters of SCM at the target FAR c (%) as follows:

$$\alpha^* = \arg\min_{\alpha} \text{FRR}(\text{SCM}(\psi^k, \psi^s; \alpha)), \text{ s.t. FAR} = c. \quad (4)$$

Task Representation Module (TRM). The second approach is to add a trainable neural network, $\text{TRM}_{\text{tb}}(\cdot,\cdot)$ and $\text{TRM}_{\text{to}}(\cdot,\cdot)$, whose inputs are keyword and speaker embeddings, and an output is a task-specific embedding. We apply metric learning loss with our proposed batch construction to train this module to form the discriminative embedding for TB- and TO-KWS. We define positive and negative samples based on an anchor sample mimicking the test case of each task as described in Fig. 1a. Then, we apply an angular prototypical loss function [16] to make the positive samples closer and the negative samples apart from the anchor sample. First, we measure the similarity between query samples and prototypes as follows:

$$\psi^{\text{tb}}_{i,j} = \text{sim}(\text{TRM}_{\text{tb}}(z^k_i, z^s_i), \text{TRM}_{\text{tb}}(p^k_j, p^s_j)), \quad (5)$$

where prototypes p^k_j and p^s_j are reference embeddings of the corresponding keyword and speaker of j-th sample, here we use learned classifiers' weights W^k_j and W^s_j as prototypes in training while the embedding of the enrolled speaker is used for p^s_j in test. Then, we define a task-specific loss function as follows:

$$L_{\text{tb}} = -\frac{1}{N}\sum_{i=1}^{N} \frac{\exp(w\cdot\psi^{tb}_{i,i} + b)}{\sum_{j=1}^{N}\exp(w\cdot\psi^{tb}_{i,j} + b)}. \quad (6)$$

We apply the same task-specific loss function for TO-KWS, but negative samples are selected differently, *i.e.*, *nts-tk* is selected for negative samples. TRM can extract discriminative features for each task by minimizing Eq. 6. We denote our mutli-task learning architecture with TRM as PK-MTL.

Inference time. Given the test input x_i, we use the score from keyword embeddings, $\psi^k_{i,\text{ref}}$, for C-KWS and task-specific scores from SCM or TRM, $\psi^{tb}_{i,\text{ref}}$ and $\psi^{to}_{i,\text{ref}}$, for personalized tasks, as illustrated in Fig. 2.

3. Experiments

3.1. Experimental Setup

Datasets. We evaluate conventional and personalized KWS systems on Google Speech Commands v1 [11]. The dataset contains 64,727 utterances of total 30 words from 1,881 speakers. We follow the conventional 12 class classification setting [11], which consists of ten classes of "Yes," "No," "Up," "Down," "Left," "Right," "On," "Off," "Stop," and "Go" with two additional classes "Unknown" and "Silence," which indicate remaining twenty words and no speech, respectively. We divide the dataset into training, validation, and test in the same way as [3, 2, 7, 11, 26]. For testing on TB- and TO-KWS, we make sample-to-sample pairs consisting of positive and negative pairs. We randomly select an anchor sample and choose *ts-tk*, *nts-tk*, *ts-ntk*, and *nts-ntk* samples of the anchor. To reduce the performance variation, we obtained 10 test splits where each test split contains 16,000 pairs. We report the average performance from 10 test splits. Note that "Silence" and "Unknown" classes can be selected for non-target keywords but not for target keywords. Moreover, we make 10 test splits to validate speaker verification performance, which contain 160,000 speaker pairs that are randomly sampled from the test set.

We also evaluate the models trained on Google Speech Commands in a practical scenario, where speeches continuously come from news or conversations containing words that are general negatives for the system. We exploit the WSJ-SI200 [13] and Librispeech [14] datasets as the negatives for the above scenario. Negative samples from WSJ-SI200 are segmented from the whole audio stream into the one-second-long following [9]. For Librispeech, we segment the entire audio stream into one-second-long in the public noisy test set. We make a test pair where one sample is from general negatives and the other is the target keyword with a randomly selected speaker from Google Speech Commands, and we conduct this process for all samples by pairing with 10 target keywords.

Implementation details. We exploit three keyword spotting architectures as the backbone network, BC-ResNet [7], Res15 [2] and DS-ResNet [12], for our PK-MTL. For BC-ResNet, we use input features of 40-dimensional log Mel spectrograms with 30 ms window length and 10 ms frameshift and apply data augmentations following [7]. For the other two backbones, we follow [2, 12] to add noise and random shift to each segment. Then, we extract 40 dimensional Melfrequency cepstrum coefficient features and use them as inputs. We design the shared encoder using the backbones except the last two conv layers for BC-ResNet and the last conv block for Res15 and DS-ResNet. Sub-networks follow the shared encoder, which consist of the remaining layers of each backbone and an additional fully connected layer. We follow the official training strategy [2, 7, 12] of each baseline network, *e.g.*, a learning rate and a mini-batch size. We set λ to 0.1 in Eq. 3. For the baselines, *i.e.*, BC-ResNet, Res15, and DS-ResNet, we add an additional fully connected layer before the classifier and use the cosine classifier, $g^k(\cdot)$. This modification improves the performance with additional parameters and computational cost.

We define Score Combination Module (SCM) as a linear combination function. We adopt the attention modules [27] for Task Representation Module (TRM). TRM extracts the concatenated normalized keyword and speaker embeddings attended by the attention weights that are obtained by two fully connected layers whose intermediate feature size is 2. TRM is trained for 50 epochs using the same training strategy of the backbones.

Evaluation metric. We use false alarm rate (FAR), false rejection rate (FRR), and equal error rate (EER) at which FAR and FRR are the same. Also, Top-1 test accuracy is used for evaluating multiple keywords classification (C-KWS).

3.2. Ablation studies on Google Speech Commands

Effectiveness of multi-task learning. We compare three methods: 1) Vanilla is the conventional KWS model, 2) Vanilla (+SV) uses an additional single task model for SV, and 3) Naive MTL indicates our MTL framework trained by Eq. 3. In Table 1, the results of EER on SV and C-KWS demonstrate that the keyword spotting system can learn speaker representations without performance degradation on KWS with a marginal increase in the number of parameters and computation compared

Table 1: *Comparison of Top-1 test accuracy (%), EER (%), and FRR (%) at FAR* 1% *and FAR* 10% *of SV, C-KWS, TB-KWS, and TO-KWS on Google Speech Commands v1. Reported numbers are mean (std) over five trials.*

		SV	C-KWS		TB-KWS			TO-KWS				
Method	Backbone	EER	Acc.	EER	FRR at 1%	FRR at 10%	EER	FRR at 1%	FRR at 10%	EER	#Mult	#Param
Vanilla	BC-ResNet-3 [7]	-	97.57 (0.04)	**1.92 (0.15)**	2.85 (0.09)	1.89 (0.20)	2.47 (0.03)	96.50 (0.07)	64.32 (0.26)	21.17 (0.08)	16.7M	63.5k
Vanilla (+SV)	BC-ResNet-3 [7]	**3.32 (0.10)**	97.57 (0.04)	**1.92 (0.15)**	2.85 (0.09)	1.89 (0.20)	2.47 (0.03)	96.50 (0.07)	64.32 (0.26)	21.17 (0.08)	22.5M	82.4k
Naive MTL	BC-ResNet-3 [7]	3.36 (0.13)	**97.68 (0.18)**	1.98 (0.07)	2.82 (0.10)	1.71 (0.32)	2.44 (0.06)	96.44 (0.09)	64.26 (0.24)	21.12 (0.09)	17.5M	80.2k
PK-MTL w/ SCM-M	BC-ResNet-3 [7]	3.36 (0.13)	**97.68 (0.18)**	1.98 (0.07)	2.99 (0.17)	0.90 (0.09)	2.12 (0.19)	6.28 (0.27)	2.73 (0.13)	3.89 (0.06)	17.5M	80.2k
PK-MTL w/ SCM-GS	BC-ResNet-3 [7]	3.36 (0.13)	**97.68 (0.18)**	1.98 (0.07)	2.75 (0.14)	1.08 (0.52)	2.40 (0.05)	**6.24 (0.42)**	1.97 (0.70)	3.76 (0.17)	17.5M	80.2k
PK-MTL	BC-ResNet-3 [7]	3.36 (0.13)	**97.68 (0.18)**	1.98 (0.07)	**2.58 (0.10)**	**0.75 (0.25)**	**2.02 (0.22)**	6.56 (0.20)	**1.52 (0.18)**	**3.37 (0.15)**	17.5M	82.0k
Vanilla	Res15 [2]	-	95.82 (0.21)	2.38 (0.10)	4.17 (0.23)	2.23 (0.17)	3.03 (0.05)	96.36 (0.28)	63.83 (0.82)	21.25 (0.08)	966.3M	241.1k
PK-MTL	Res15 [2]	4.43 (0.13)	96.30 (0.19)	2.25 (0.13)	3.30 (0.33)	0.92 (0.08)	2.14 (0.19)	9.35 (0.46)	2.56 (0.21)	4.44 (0.18)	1,040.6M	262.3k
Vanilla	DS-ResNet18 [12]	-	96.83 (0.21)	1.85 (0.20)	3.10 (0.17)	1.33 (0.17)	2.41 (0.17)	96.11 (0.13)	63.76 (0.30)	21.18 (0.07)	305.6M	79.9k
PK-MTL	DS-ResNet18 [12]	3.29 (0.07)	96.85 (0.32)	1.91 (0.21)	2.46 (0.12)	0.81 (0.15)	1.61 (0.06)	6.62 (0.37)	1.90 (0.18)	3.68 (0.24)	326.1M	89.9k
Vanilla	BC-ResNet-8 [7]	-	98.01 (0.16)	1.89 (0.17)	2.69 (0.11)	0.77 (0.41)	2.43 (0.04)	96.30 (0.32)	63.20 (0.64)	21.20 (0.09)	91.9M	386.9k
PK-MTL	BC-ResNet-8 [7]	2.58 (0.10)	97.87 (0.12)	1.79 (0.27)	1.72 (0.10)	0.47 (0.05)	1.33 (0.04)	5.01 (0.17)	1.41 (0.28)	3.22 (0.27)	96.6M	501.7k

Table 2: *FAR (%) at FRR* 1% *and* 5% *on WSJ and Librispeech datasets. Reported numbers are mean (std) over five trials.*

		GSC+WSJ		GSC+Librispeech	
Method	Backbone	FAR at 1%	FAR at 5%	FAR at 1%	FAR at 5%
Vanilla	Res15 [2]	41.49 (2.37)	1.31 (1.87)	33.77 (2.34)	0.51 (0.08)
Vanilla	DS-ResNet18 [12]	47.44 (5.76)	0.25 (0.39)	37.46 (9.05)	0.48 (0.06)
Vanilla	BC-ResNet-3 [7]	46.61 (7.45)	0.00 (0.01)	27.59 (3.23)	0.19 (0.02)
Vanilla	BC-ResNet-8 [7]	45.00 (4.98)	0.00 (0.00)	21.83 (2.79)	0.12 (0.02)
PK-MTL (TB)	BC-ResNet-3 [7]	5.58 (4.20)	0.00 (0.00)	2.69 (1.98)	0.02 (0.01)
PK-MTL (TO)	BC-ResNet-3 [7]	**0.15 (0.11)**	**0.00 (0.00)**	**0.15 (0.07)**	**0.00 (0.00)**
PK-MTL (TB)	BC-ResNet-8 [7]	1.32 (1.55)	0.00 (0.01)	0.99 (0.36)	0.01 (0.00)
PK-MTL (TO)	BC-ResNet-8 [7]	**0.20 (0.30)**	**0.00 (0.01)**	**0.15 (0.06)**	**0.01 (0.00)**

to Vanilla. Vanilla (+SV) also leverages speaker information using single-task SV model, but it requires a higher computational cost due to separately performing each task.

Impact of task-specific scoring functions. To analyze the impact of scoring functions, we apply SCM and TRM to Naive MTL whose backbone is BC-ResNet-3. With a combination function whose α is manually chosen as 0.5, SCM-M, it cannot consistently improve TB-KWS even though speaker information is exploited to the task. We can improve the performance at the target FAR, especially 1%, by finding α through grid-search minimizing Eq. 4 at the target FAR on the validation set, and PK-MTL w/ SCM-GS largely reduces FRR at FAR 1% of TB- and TO-KWS. However, it has limited performance improvements because their representations cannot learn task-specific characteristics explicitly. Therefore, we utilize TRM so that PK-MTL fully adapts keyword and speaker representations to TB- and TO-KWS, and it helps reducing FRR and EER significantly.

PK-MTL on various baselines. Our framework can be applied to any C-KWS architectures, hence we use three backbones, Res15, DS-ResNet18, BC-ResNet-8. In Table 1, even though the performance of SV and the influence of SV to KWS induced by multi-task learning are different depending on the structure and size of the backbones, PK-MTL improves the performance consistently on all personalized tasks with the aid of the speaker representations and the task-specific scoring function, TRM.

3.3. Experiments on the realistic scenario

The KWS system trained on Google Speech Commands selects a threshold based on the target FRR of positive samples, *i.e.*, *ts-tk*. With the threshold, we evaluate KWS systems on WSJ and Librispeech datasets to validate whether they can reject general negatives. All vanilla methods cannot reject general negatives that contain the target keywords, *e.g.*, "Yes" and "On," or similar keywords because speaker information is ignored for detecting keywords. It makes KWS systems induce high FAR at the operating points of FRR 1% as shown in Table 2, and it can lead to increased power consumption. However, in our systems, general negatives are easily rejected by considering speaker representations since the speaker characteristics of general negatives

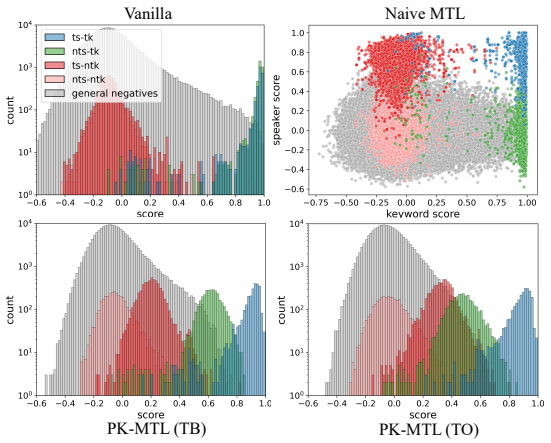


Figure 3: *The histograms (log scale) for Vanilla and PK-MTL show the distribution of scores from keyword and task-specific embeddings, respectively. The scatter plot for Naive MTL shows the distribution of keyword (x-axis) and speaker (y-axis) scores.*

are different from the enrolled users. Especially, PK-MTL (TO) can effectively reject general negatives even at the low operating point, FAR 1%, because it learns to reject *nts-tk*.

We analyze the score distribution of KWS systems as shown in Fig. 3. We plot the scores of *ts-tk*, *nts-tk*, *ts-ntk*, *nts-ntk* pairs from Google Speech Commands and pairs of target keywords and general negatives from Librispeech. In Vanilla, the scores of target keywords and non-target keywords are well split regardless of speaker identities, but a lot of general negatives also result in high scores. We plot both keyword and speaker scores through Naive MTL. As expected, general negatives have low speaker scores even though their keywords are similar or the same as the target. The figure for PK-MTL (TB) shows that the score of *ts-tk* is higher than those of *nts-tk*, which indicates the model is biased toward the target users. Moreover, *ts-tk* is well separated from other samples in PK-MTL (TO), which objective is to accept the target user only. In PK-MTL, the scores of general negatives are apart from those of positive samples, thus we can reject them through the scores of TB or TO.

4. Conclusions

In this paper, we propose to leverage speaker information into the keyword spotting system to tackle personalized keyword spotting tasks. Through proposed multi-task learning and task adaptation, our system can adapt to personalized tasks. Extensive experiments show that leveraging speaker information reduces false alarm and rejection rates significantly in personalized and realistic keyword spotting scenarios.

5. References

[1] A. Gruenstein, R. Alvarez, C. Thornton, and M. Ghodrat, "A cascade architecture for keyword spotting on mobile devices," *arXiv preprint arXiv:1712.03603*, 2017.

[2] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[3] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," in *Proceedings of the INTERSPEECH*, 2019.

[4] S. Mittermaier, L. Kürzinger, B. Waschneck, and G. Rigoll, "Small-footprint keyword spotting on raw audio data with sinc-convolutions," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[5] P. Zhang and X. Zhang, "Deep template matching for small-footprint and configurable keyword spotting." in *Proceedings of the INTERSPEECH*, 2020.

[6] B. Zhang, W. Li, Q. Li, W. Zhuang, X. Chu, and Y. Wang, "Autokws: Keyword spotting with differentiable architecture search," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[7] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted Residual Learning for Efficient Keyword Spotting," in *Proceedings of the INTERSPEECH*, 2021.

[8] Y. Yuan, C.-C. Leung, L. Xie, H. Chen, B. Ma, and H. Li, "Learning acoustic word embeddings with temporal context for query-by-example speech search," in *Proceedings of the INTERSPEECH*, 2018.

[9] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang, "Query-by-example on-device keyword spotting," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.

[10] J. Huang, W. Gharbieh, H. S. Shim, and E. Kim, "Query-by-example keyword spotting system using multi-head attention and soft-triple loss," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[11] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *CoRR*, vol. abs/1804.03209, 2018.

[12] M. Xu and X.-L. Zhang, "Depthwise separable convolutional resnet with squeeze-and-excitation blocks for small-footprint keyword spotting," in *Proceedings of the INTERSPEECH*, 2020.

[13] D. B. Paul and J. M. Baker, "The design for the wall street journal-based CSR corpus," in *ICSLP*. ISCA, 1992.

[14] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[15] D. Belli, D. Das, B. Major, and F. Porikli, "A personalized benchmark for face anti-spoofing," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, 2022.

[16] J. S. Chung, J. Huh, S. Mun, M. Lee, H. S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, "In defence of metric learning for speaker recognition," in *Proceedings of the INTERSPEECH*, 2020.

[17] R. Caruana, "Multitask learning: A knowledge-based source of inductive bias," in *International Conference on Machine Learning (ICML)*, 1993.

[18] J. Baxter, "A bayesian/information theoretic model of learning to learn via multiple task sampling," *Machine learning*, vol. 28, no. 1, pp. 7–39, 1997.

[19] M. Kanakis, D. Bruggemann, S. Saha, S. Georgoulis, A. Obukhov, and L. V. Gool, "Reparameterizing convolutions for incremental multi-task learning without task interference," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

[20] G. Sun, T. Probst, D. P. Paudel, N. Popović, M. Kanakis, J. Patel, D. Dai, and L. Van Gool, "Task switching network for multi-task learning," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[21] S. Yun, J. Cho, J. Eum, W. Chang, and K. Hwang, "An end-to-end text-independent speaker verification framework with a keyword adversarial network," in *Proceedings of the INTERSPEECH*, 2019.

[22] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] J. Liu, L. Song, and Y. Qin, "Prototype rectification for few-shot learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

[24] R. Kumar, V. Yeruva, and S. Ganapathy, "On convolutional lstm modeling for joint wake-word detection and text dependent speaker verification." in *Proceedings of the INTERSPEECH*, 2018.

[25] M. Jung, Y. Jung, J. Goo, and H. Kim, "Multi-task network for noise-robust keyword spotting and speaker verification using ctc-based soft vad and global query attention," in *Proceedings of the INTERSPEECH*, 2020.

[26] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," in *Proceedings of the INTERSPEECH*, 2020.

[27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.