Variational On-the-Fly Personalization

Jangho Kim [*1,2] **Jun-Tae Lee** [*1] **Simyung Chang** [1] **Nojun Kwak** [2]

Abstract

With the development of deep learning (DL) technologies, the demand for DL-based services on personal devices, such as mobile phones, also increases rapidly. In this paper, we propose a novel personalization method, *Variational On-the-Fly Personalization*. Compared to the conventional personalization methods that require additional fine-tuning with personal data, the proposed method only requires forwarding a handful of personal data on-the-fly. Assuming even a single personal data can convey the characteristics of a target person, we develop the variational hyper-personalizer to capture the weight distribution of layers that fits the target person. In the testing phase, the hyper-personalizer estimates the model's weights on-the-fly based on personality by forwarding only a small amount of (even a single) personal enrollment data. Hence, the proposed method can perform the personalization without any training software platform and additional cost in the edge device. In experiments, we show our approach can effectively generate reliable personalized models via forwarding (not back-propagating) a handful of samples.

1. Introduction

In recent years, most of the deep learning researches have paid attention to developing universal models with sophisticated architectures using a large-scale dataset covering an entire target domain. However, in edge devices, such as mobile phones and IoT sensors, deep models are required to process (learn or infer) a *personal* domain where data

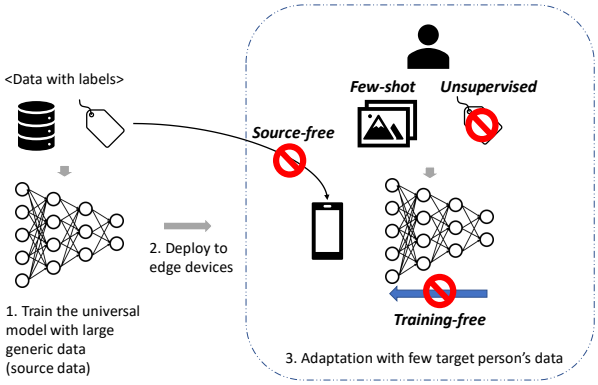


Figure 1: Our personalization scenario on edge devices with practically crucial constraints (few-shot target, unsupervised, source-free, and training-free).

are generated in a specific environment (Wang et al., 2019b; Chen & Ran, 2019). For example, data could be always collected by a specific user, then the device may never confront other personal domains. In this case, using a universal model is inefficient, and also its performance may degrade in some personal domains. Hence, to successfully deploy deep learning algorithms in edge devices, it is significant to specialize a model to the data distribution of the personal domain, i.e, *personality*, in question.

Despite the importance of personalization, there has been little progress due to the following practical constraints of edge devices, which are depicted in Fig. 1. First, it is hard to access the source data (generic data) used for training (*source-free*). Second, only a few user-specific target data (personal data) can be available (*few-shot*). Third, for usability, personal data are preferable to be unlabeled since a service requiring user's annotation drastically reduces usability (*unsupervised*). Also, on-device training with user's data suffers from some non-algorithmic constraints (*training-free*). For example, collecting personal data to train the models may cause privacy concerns. And model training requires an on-device training platform and much more hardware resources (memory and computational units) than inference.

Various existing works have tried to qualify a model to specified domains. Nevertheless, they do not completely meet the aforementioned constraints, and are not directly applicable to our personalization task. Fallah et al. (2020) localized multiple models to different personal domains, federating to

[*]Equal contribution [1]Qualcomm AI Research, an initiative of Qualcomm Technologies, Inc. Jangho Kim completed the research in part during an internship at Qualcomm Technologies, Inc. [2]Seoul National University, South Korea. Correspondence to: Jangho Kim <kjh91@snu.ac.kr>, Jun-Tae Lee <juntlee@qti.qualcomm.com>, Simyung Chang <simychan@qti.qualcomm.com>, Nojun Kwak <nojunk@snu.ac.kr>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

produce a universal one. However, in the localized learning, a plenty of annotated personal data are required. Although Motiian et al. (2017) and Luo et al. (2017) adapted a model to a target domain with a few data, they need source domain data as well as the labels of the few target data. While unsupervised domain adaptation was devised by Long et al. (2016), enough target and source domain data are required. Recent source-free domain adaptation techniques (Yang et al., 2021; Kundu et al., 2020) adapted a model without any source domain data and label of target domain data, but they still use abundant target domain data. Commonly, those existing methods are not training-free. Therefore, it is still challenging to flexibly personalize a model to a given personal domain using only a small-sized personal data.

To attain the personalization satisfying the constraints shown in Fig. 1, we introduce a novel on-the-fly personalization paradigm. Specifically, given a personal data, we compute the weights of layers specialized to its personality on-the-fly via forwarding only a few personal data. To this end, we propose a *variational on-the-fly personalization* (VoP) method. The key of our method lies in a small detachable module, the *variational hyper-personalizer* which is trained to produce an approximated posterior distribution of weights of a layer based on the personality. Assuming the data in a personal domain share the same personality, we set the target prior distribution as the averaged distribution, called *prototype*, of the personal data in the variational inference of the hyper-personalizer. In the testing phase, by forwarding a small amount of testing data with the same personality, called enrollment data, the hyper-personalizer estimates the posterior distribution of weights in the layer according to the personality represented by the enrollment data. Then, a personalized layer is generated from the estimated posterior distribution. Since the hyper-personalizer is detached from the model after generating the personalized weights of the layer, we can obtain a personalized model without increasing computational cost.

To show the effectiveness of the proposed variational on-the-fly personalization, we first apply it to two tasks strongly relevant to personalization: keyword spotting and open-set speaker verification. Then, we extend our VoP to few-shot classification task, as well. For each task, we show that our VoP successfully increases the performance of a baseline method on the considered public benchmark dataset.

The contribution of our work is summarized as follows:

- To the best of our knowledge, we propose the first on-the-fly personalization method to specialize a model to a given personality.
- We formulate the personalization via forwarding as a variational inference problem, and solve it by considering the prototype distribution as the target posterior distribution of the personal layer's weights.
- By forwarding a few personal data, the variational hyper-personalizer captures the weight distribution of layers, and produces the personalized layer weights.
- We analyze the effectiveness of the proposed method on various tasks such as keyword spotting, open-set speech verification, and few-shot classification tasks.

2. Related work

Neural networks for edge devices Many on-device deep learning services have been introduced, then personalization of deep learning algorithms gets important.

Keyword spotting (KWS) recognizes a specific keyword to wake up a device. As KWS models are frequently deployed and used in edge devices, diverse methods have been developed for efficient KWS in the devices, such as lightening model sizes (Tang & Lin, 2017; Zhang et al., 2017; Tang & Lin, 2018), or reducing computation (Choi et al., 2019; Li et al., 2020). Although KSW deals with an enrolled user's voice rather than anonymous ones, research on personalization has not been actively studied.

Speaker verification (SV) also requires personalization for accurate verification for a target user. To obtain robust target speaker's embedding, metric learning-based methods are widely addressed (Wang et al., 2019a; Chung et al., 2020; Ko et al., 2020). However, while they aim to make a better embedder in the training phase, our VoP can personalize the model by forwarding a handful of personal data on-the-fly.

Recently, to address the degeneration issue of federated learning for non-IID local data, personalized federated learning was devised. In this line of research, several works decentralized the model-agnostic meta-learning (Jiang et al., 2019; Fallah et al., 2020; Deng et al., 2020). Other approaches mixed the global and personal (local) models (Deng et al., 2020; Hanzely & Richtárik, 2020). While these methods are akin to our VoP in that the global and local layers are used separately, unlike ours, they inevitably require training models on personal data.

Existing personalization methods are mostly tailored to specific tasks, and also require task-specific training for target personality. Whereas, as a versatile algorithm for diverse tasks, the proposed VoP uses no extra training on devices. This enables a lightweight personalized neural network platform for edge devices that may not support training.

Few-shot embedding adaptation To overcome the need for massive training data and generalize the model with a few limited examples, lots of approaches have been attempted in many research fields. In the field of few-shot domain adaptation (Tzeng et al., 2015; Motiian et al., 2017; Luo et al., 2017), a model trained on a source domain is adopted to a heterogeneous target domain with a few labeled target data. However, those methods highly rely on prior information from a large-scale source data.

In few-shot classification (Snell et al., 2017; Iakovleva et al., 2020; Ye et al., 2020; Gordon et al., 2018; Das et al., 2022), a model meta-learns how to build embedding with a few examples in source classes with sufficient labeled data, and then the model is applied to tasks sampled from unseen target classes, where each task consists of a few labeled support set and a query set. Recently, Gordon et al. (2018) exploited a Bayesian framework to relieve the model uncertainty by insufficient support examples, where the weight of the classification layer is generated for each task in the testing phase. To further tailor the Bayesian framework to this task, Iakovleva et al. (2020) regularized the classifier generator based on the relationship between support and query sets. While these Bayesian approaches have relevance with our VoP, they only focus on the weight generation of the final linear layer based on query set for the few-shot classification task. Contrarily, we supposed that the inherent personality can be conveyed by even a single personal sample for generic use in various personalization tasks with scarce personal data. Also, we designed VoP to personalize both convolutional and linear layers in various depths.

Algorithm 1 Variational On-the-Fly Personalization (VoP)

$P \leftarrow$ The number of iterations
Given $D = \{(s_i, y_i, r_i)\}_{i=1}^{N}$ where $y_i \in [C]$, $r_i \in [R]$
$\psi, \theta \leftarrow$ Initialization
[Training]
for Iter = 1 ,..., P **do**
  D^M=$\{(s_i, y_i, r_i)\}_{i=1}^{M} \leftarrow$ Minibatch of M samples (drawn from D), where $y_i \in \mathcal{C}^M \subset [C]$, $r_i \in \mathcal{R}^M \subset [R]$
  $\{x_i\}_{i=1}^{M} \leftarrow$ Extract features from $\{s_i\}_{i=1}^{M}$, using the encoding module f_ψ
  $\{\mu_i, \Sigma_i\}_{i=1}^{M} \leftarrow$ Outputs from q_θ
  $\{\bar{\mu}_k\}_{k=1}^{|R^M|}, \{\bar{\Sigma}_k\}_{k=1}^{|R^M|} \leftarrow$ Obtain proto-means and proto-variances using (9)
  $\{\omega_i\}_{i=1}^{M} \leftarrow$ Sample sample-specific weights using $p(\epsilon)$ based on $\{\mu_i, \Sigma_i\}_{i=1}^{M}$
  Update ψ, θ by minimizing $\widehat{\mathcal{L}}_{VoP}$ in (12)
end for
[Testing] to generate the k-th personality model
$D^E = \{(s_i, y_i, r_i = k)\}_{i=1}^{E}$: Few enrollment samples for the k-th personality
Obtain $\{\bar{\mu}_k\}$ using f_ψ, q_θ and D^E
$\{\bar{\omega}_k\} \leftarrow$ Set $\bar{\mu}_k$ as the personal weights, which maximize the likelihood for the personal samples
Abolish q_θ.

3. Method

In this section, we first introduce the framework of the proposed VoP. Note that VoP is generally applicable to personalize deep networks, via only forwarding a small amount of samples on-the-fly. Next, we describe the variational inference process of the hyper-personalizer which produces the personalized weight for a corresponding layer. Hypermodules (Ha et al., 2017; Ballé et al., 2018; Minnen et al., 2018) have been widely adopted in diverse tasks, but our variational hyper-personalizer is specially designed for personalization. Lastly, we analyze the convergence of VoP.

3.1. Overall Framework

Fig. 2 depicts the framework of the proposed VoP. Our network consists of an encoding module f_ψ and a detachable hyper-personalizer q_θ. Then, the network is learned to solve a classification problem considering different personalities of input samples. To this end, we suppose a dataset $D = \{(s_i, y_i, r_i)\}_{i=1}^{N}$ with multiple personalities. A sample s_i is annotated by a class label y_i and a personality label r_i where $y_i \in [C]$, $r_i \in [R]$ ($[n]$ is a positive integer set $\{1, \ldots, n\}$)[1].

In the training stage, the encoding module, which is shared across the personalities of inputs, first extracts features of input samples. Then, the encoded features are forwarded to the hyper-personalizer. For each personality, the hyper-personalizer performs the variational inference where the true posterior of the sample-specific weights of the corresponding layer is approximated. Then, the sample-specific layer is generated from the approximated posterior. Finally, the sample-specific layer yields the personalized feature (or prediction if it is the last layer).

Note that, in the testing phase, the hyper-personalizer is used only once for a few enrollment samples, in order to generate the weights of the personalized layer for the personality corresponding to the enrollment samples on-the-fly without any back-propagation.

[1]For example, for a keyword spotting task, each keyword corresponds to a class, while each utterer corresponds to a personality.

3.2. Variational On-the-Fly Personalization

The goal of the hyper-personalizer is to find a proper weight distribution shared across the samples with the same personality. For this purpose, for a personality k, we estimate the sample-specific true posterior from the encoded sample feature x_i^k ($= f_\psi(s_i)$ where $r_i = k$), because it is difficult to directly find the true posterior from raw input samples. However, in general, the true posterior of the weight ω, $p(\omega|x_i^k, y_i^k)$, is intractable (Gal, 2016).

Accordingly, in order to approximate the true posterior $p(\omega|x_i^k, y_i^k)$, we design a variational distribution $q_\theta(\omega|x_i^k)$ parameterized by θ. Then, we minimize the Kullback–Leibler (KL) divergence between the two distributions:

$$
\begin{aligned}
KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k, y_i^k)) &= \int q_\theta(\omega|x_i^k) \ln \frac{q_\theta(\omega|x_i^k)}{p(\omega|x_i^k, y_i^k)} d\omega \\
&= \int q_\theta(\omega|x_i^k) \ln \left(\frac{q_\theta(\omega|x_i^k)}{p(\omega|x_i^k)} \frac{p(y_i^k|x_i^k)}{p(y_i^k|x_i^k, \omega)} \right) d\omega \\
&= KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k)) \\
&\quad - \mathbb{E}_{\omega \sim q_\theta(\omega|x_i^k)}[\ln p(y_i^k|x_i^k, \omega)] + \ln p(y_i^k|x_i^k).
\end{aligned}
\tag{1}
$$

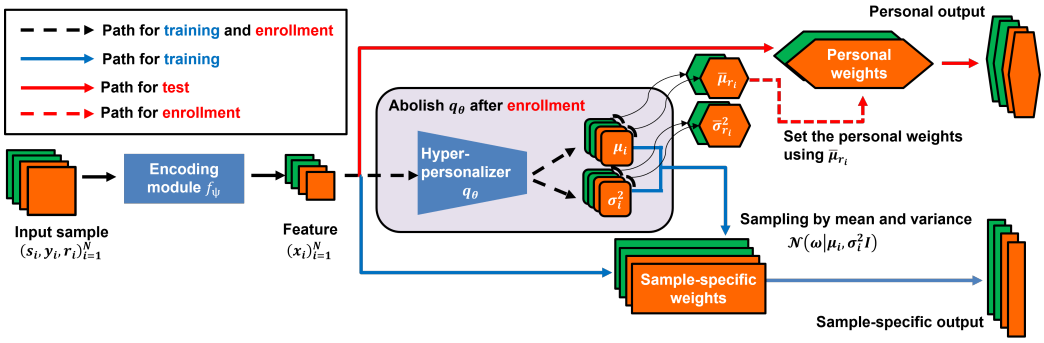


Figure 2: The overall process of the proposed VoP. Colors in input samples represent personalities for each input sample. In the training phase, VoP trains the encoding module and hyper-personalizer to estimate sample-specific weights via black dashed and blue bold arrows. At testing phase, for each personality, VoP generates personal weights by forwarding a few enrollment samples via black and red dashed arrows, once. After generating the personal weights (model), VoP abolishes the hyper-personalizer, and then only utilizes the encoding module and the personal weights. More details are in Sec. 3.

In (1), the minimization of the KL divergence is equivalent to maximizing the evidence lower bound (ELBO):

$$\mathbb{E}_{\omega\sim q_\theta(\omega|x_i^k)}[\ln p(y_i^k|x_i^k,\omega)] - KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k)) \leq \ln p(y_i^k|x_i^k). \tag{2}$$

This approximation process is known as *variational inference* (VI) (Hoffman et al., 2013). In other words, we approximate the true posterior distribution by minimizing the following objective function (negative ELBO):

$$\mathcal{L}_{VI}(\theta,(x_i,y_i,k)) = -\mathbb{E}_{\omega\sim q_\theta(\omega|x_i^k)}[\ln p(y_i^k|x_i^k,\omega)] + KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k)). \tag{3}$$

In the minimization of the objective function, the first term induces $q_\theta(\omega|x_i^k)$ to predict the correct output and the second KL term encourages the variational distribution to resemble the sample-specific prior $p(\omega|x_i^k)$.

Note that the first RHS term requires estimation by sampling ω from the variational distribution $q_\theta(\omega|x_i^k)$. For the sampling, we assume the true posterior $p(\omega|x_i^k, y_i^k)$ and the sample-specific prior $p(\omega|x_i^k)$ to be uncorrelated multivariate Gaussian. Then, the hyper-personalizer calculates parameters for the variational distribution $q_\theta(\omega|x_i^k)$ following the multivariate Gaussian, $q_\theta(\omega|x_i^k) = \mathcal{N}(\omega|\mu_i, \Sigma_i)$, with $\Sigma_i = \text{diag}(\sigma_i^2)$. Here, μ_i and σ_i^2 are outputs of the hyper-personalizer which is a multi-layer perceptron (MLP) with a single hidden layer ($\mu_i, \sigma_i^2 \in \mathbb{R}^Z$ where Z is the dimension of ω).

After calculating μ_i, σ_i^2, we apply the reparameterization trick for a pathwise derivative estimator (Kingma & Welling, 2013; Gal, 2016). To reparameterize a random variable ω following $q_\theta(\omega|x_i^k)$, we exploit a noise variable ϵ following $p(\epsilon)$. Then, ω can be represented by a non-random differentiable function $g(\epsilon, x_i^k; \theta)$:

$$\omega = g(\epsilon, x_i^k; \theta) = \mu_i + \sigma_i \odot \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, I), \tag{4}$$

where $\odot$ denotes elementwise multiplication. From this, we can rewrite (3) w.r.t. $p(\epsilon)$ (for convenience, $KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k))$ is referred to as KL):

$$\begin{aligned}\mathcal{L}_{VI}(\theta,(x_i,y_i,k)) &= -\int q_\theta(\omega|x_i^k)\ln p(y_i^k|x_i^k,\omega)d\omega + KL \\ &= -\int p(\epsilon)\ln p(y_i^k|x_i^k, g(\epsilon,x_i^k;\theta))d\epsilon + KL\end{aligned} \tag{5}$$

whose first term can be approximated with Monte Carlo (MC) estimator as follows:

$$\begin{aligned}&-\int p(\epsilon)\ln p(y_i^k|x_i^k, g(\epsilon,x_i^k;\theta))d\epsilon \\ &\simeq -\frac{1}{L}\sum_{l=1}^{L}\ln p(y_i^k|x_i^k, g(\epsilon_l,x_i^k;\theta)),\end{aligned} \tag{6}$$

where ϵ_l is independently sampled from $p(\epsilon)$. We can train θ with the MC estimator:

$$\mathcal{L}_{MC}(\theta,(x_i,y_i,k)) = -\frac{1}{L}\sum_{l=1}^{L}\ln p(y_i^k|x_i^k, g(\epsilon_l,x_i^k;\theta)) + KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k)). \tag{7}$$

To compute the KL divergence in (7), we assume that the sample-specific prior of x_i follows the distribution of its personality $r_i = k$:

$$p(\omega|x_i^k) \simeq p(\omega|k). \tag{8}$$

Then, we define the prototype distribution for the personality by the mean and variance of a proto-set $\mathcal{S}_k$. This proto-set, which is pre-defined for each k, contains the mean and variance for each of a few samples belonging to the kth

personality. Hence, the proto-mean $\bar{\mu}_k$ and proto-variance $\bar{\Sigma}_k$ are obtained by

$$\bar{\mu}_k = \frac{1}{|\mathcal{S}_k|} \sum_{\mu_j \in \mathcal{S}_k} \mu_j, \qquad \bar{\Sigma}_k = \frac{1}{|\mathcal{S}_k|} \sum_{\Sigma_j \in \mathcal{S}_k} \Sigma_j \quad (9)$$

where μ_j and Σ_j are the means and variances from the hyper-personalizer for the jth sample in $\mathcal{S}_k$, respectively.

For brevity, we assume that Σ and $\bar{\Sigma}$ are diagonal, i.e, $\Sigma = \text{diag}(\sigma^2)$ and $\bar{\Sigma} = \text{diag}(\bar{\sigma}^2)$. Hence, the KL divergence term of (7) can be calculated analytically (See Appendix A):

$$KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k)) \simeq \frac{1}{2}\sum_{z=1}^{Z}(\ln \bar{\sigma}^2_{(k,z)} - \ln \sigma^2_{(i,z)} - 1 + \frac{\sigma^2_{(i,z)}}{\bar{\sigma}^2_{(k,z)}} + \frac{(\mu_{(i,z)} - \bar{\mu}_{(k,z)})^2}{\bar{\sigma}^2_{(k,z)}}) \quad (10)$$

where we use $p(\omega|x_i^k) \simeq p(\omega|k) = \mathcal{N}(\omega|\bar{\mu}_k, \text{diag}(\bar{\sigma}_k^2))$ and $z \in [Z]$ is the index of multivariate Gaussian dimension.

Based on the sample-specific MC estimator in (7), the overall loss function for all N samples is defined as follows:

$$\mathcal{L}_{VoP} = \frac{1}{N}\sum_{i=1}^{N}(-\frac{1}{L}\sum_{l=1}^{L}\ln p(y_i^{r_i}|x_i^{r_i}, g(\epsilon_l, x_i^{r_i};\theta)) + \alpha KL(q_\theta(\omega|x_i^{r_i})||p(\omega|x_i^{r_i}))). \quad (11)$$

Here, α is a scale hyper-parameter to balance between the negative log likelihood and the KL divergence. We can construct an unbiased stochastic estimator $\widehat{\mathcal{L}}_{VoP}$ to (11) by data sub-sampling, i.e., minibatch:

$$\widehat{\mathcal{L}}_{VoP} = \frac{1}{M}\sum_{i=1}^{M}(-\frac{1}{L}\sum_{l=1}^{L}\ln p(y_i^{r_i}|x_i^{r_i}, g(\epsilon_l, x_i^{r_i};\theta)) + \alpha KL(q_\theta(\omega|x_i^{r_i})||p(\omega|x_i^{r_i}))). \quad (12)$$

where M is the minibatch size. Here, we set L to 1 following (Kingma & Welling, 2013). VoP can train both the encoding module f_ψ and the hyper-personalizer q_θ with this VoP loss function (12). At test time, for each personality, the hyper-personalizer inferences only a few enrollment samples D^E to generate personalized weights ($\bar{\omega}$) on the fly. During this forwarding, we generate personalized weights based on the proto-mean, which can maximize the likelihood of personal samples. Note that, after obtaining the personalized weights ($\bar{\omega}$) for every personality, the hyper-personalizer is discarded. Thus, only the encoding module and the personalized layers are left. The overall process is described in Algorithm 1.

3.3. Analysis on VoP convergence

One may wonder if the sample mean and variance diverge as the iteration goes on because the proto-mean and proto-variance in (9) are not fixed during training. Fig. 3 left

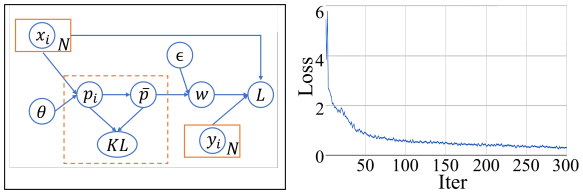


Figure 3: Convergence of VoP: Computation graph of VoP (left) and loss curve on the KWS task (right).

is the computation graph of our VoP in training where $p_i \triangleq [\mu_i, \sigma_i^2]^T$. The weight-loss relation of a network with nonlinearity is generally nonconvex, and the convergence proof of SGD is not straightforward and is an active research area. However, when the loss function is convex w.r.t. an intermediate variable (e.g. softmax output), we expect a better convergence behavior.

Theorem 3.1. *In VoP, $\theta \to KL$ and $w \to L$ are nonconvex, but $p_i \to KL$ (see dashed box in Fig. 3 left) is convex when $\bar{\sigma}^2 \leq 2\sigma_i^2$:*

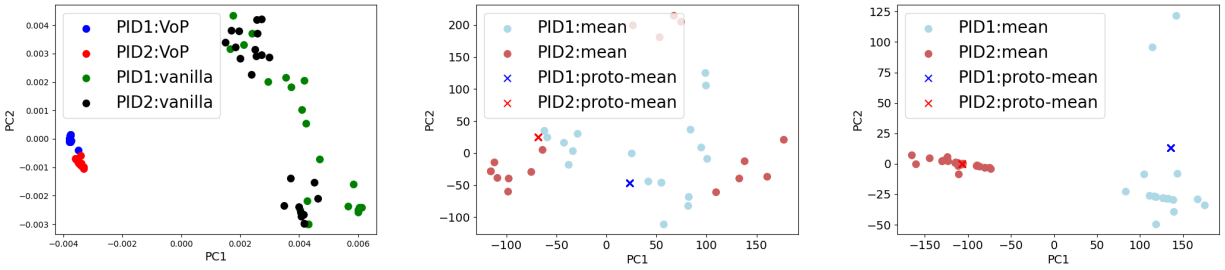
Proof. From (10), $KL(\mu_i)$ is convex because of the last quadratic term and $\mu_i - \bar{\mu} = \mu_i - \frac{1}{N}\sum_j \mu_j = \frac{N-1}{N}\mu_i - X$ where X denotes other terms independent of μ_i. Likewise, if we take $\frac{\partial^2 KL}{(\partial \sigma_i^2)^2}$, it is nonnegative when $\bar{\sigma}^2 \leq 2\sigma_i^2$. Thus, $KL(\sigma_i^2)$ is convex in this condition. If KL is convex w.r.t $p_i = [\mu_i, \sigma_i^2]^T$, it can be quadratically approximated near the optimal solution $p_i = \bar{p}$ such that $KL \simeq \frac{1}{2}\sum_i (p_i - \bar{p})^T \Lambda (p_i - \bar{p})$, where Λ is a positive definite matrix.

Let $p_i' = \Lambda^{\frac{1}{2}} p_i$ and taking the gradient descent step on p_i', it becomes $p_i'^{+} = p_i' - \eta \nabla p_i' = (1-\beta)p_i' + \beta \bar{p}'$ where $\beta = \eta \frac{N-1}{N}$. Computing $p_i'^{+} - \bar{p}'$, it becomes $p_i'^{+} - \bar{p}' = (1-\beta)(p_i' - \bar{p}')$, which means that p_i' (thus p_i) moves toward the mean $\bar{p}'(\bar{p})$ as iterations go on. By aggregating all p_i's, we can show that $\bar{p}^{+} = \frac{1}{N}\sum_i p_i^{+} = \bar{p}$ and it is fixed under gradient descent. Therefore, we can prove the convergence of p_i when only KL term is used.

In reality, as w is updated to minimize L, $\bar{p}$ is not fixed and moves according to w during training. However, as KL enforces contraction towards $\bar{p}$, p_i does not diverge and we can interpret KL as a regularizer (See Fig. 4). We also show the converging loss curve for the KWS task in Fig. 3 right.

4. Experiments

To verify the proposed VoP, we apply it to three tasks: keyword spotting, speaker verification and few-shot classification. For each task, we apply VoP employing baseline networks where VoP and the baseline model share the same architecture except the hyper-personalizer consisting of MLPs. Note that, in testing, as we discard the hyper-personalizer after generating personalized layers,the VoP-learned baselines have the same capacity as the original ones. We provide further experiments and analysis in Appendix C, D, E.



(a) Means from two models with two PIDs (b) Means from the vanilla with two PIDs (c) Means from VoP with two PIDs

Figure 4: Visualizations by PCA on outputs of the hyper-personalizers of VoP and vanilla models under two identical personal identifications (PIDs), i.e, personality labels. (a) shows the means (μ_i's) from hyper-personalizers of the two models in the same PCA space. To see the personalization in closer views, (b) and (c) display the means (μ_i's) and the proto-means ($\bar{\mu}$) from the hyper-personalizers of the vanilla and VoP models, respectively.

Table 1: Relative variances of the vanilla and VoP models depending on PCA axes for each PIDs from Figure 4a.

Method	PID1 - PC1	PID1 - PC2	PID2 - PC1	PID2 - PC2
Vanilla	148.20	933.84	90.01	1140.58
VoP	1.00	1.53	1.13	2.05

4.1. Keyword Spotting

We apply the proposed VoP on Keyword spotting (KWS) which is a classification task to detect pre-defined keywords. Due to the diversity of the texture of voices, it is crucial to personalize the KWS algorithm to individual speakers. Therefore, in this task, personality refers to an individual speaker. We employ the 'cnn-one-stride1' (Tang et al., 2018) as the baseline network. Using our VoP, we personalize the last two fully connected layers (remaining layers are the encoding module). In training, following the most basic setting (the optimizer and total training epochs) of (Tang & Lin, 2017), we use a larger minibatch size (512) to compute the proto-means and the proto-variances of different personalities, together. Also, for the stable learning of VoP, we experimentally set both learning rate and α in (12) as 0.0005, and use no learning rate decay. In testing, we forward the enrollment set of 5 samples per each personality.

Dataset We use Qualcomm Keyword Speech (Kim et al., 2019) dataset where wav files are recorded with 16 kHz with mono channel in 16 bits. It consists of 4,270 utterances spoken by 42-50 individuals which are annotated by six classes: four English keyword classes ('Hi Galaxy,' 'Hey Snapdragon,' 'Hi Lumina,' and 'Hey Android') and two non-keyword classes ('silence' and 'unknown.') We address both closed-set and open-set[2] tasks. In the closed-set setting, including 42 speakers, the dataset is divided to 80% training and 20% testing sets. In the open-set setting, the dataset consists of 37 speakers for training and five ones for testing. Note that the five speakers cannot be seen on training.

[2]Testing speakers (personalities) are not seen during training.

Analysis on Efficacy of VoP As aforementioned, our VoP loss consists of the negative log likelihood and the KL divergence. While the negative log likelihood is a cross-entropy loss for the classification task (task loss), the KL divergence helps to learn the personality. Therefore, to explore the effectiveness of the proposed VoP in personalization, we compare two models: VoP model trained with our VoP loss, and vanilla model with only task-specific cross-entropy loss. Note that both models have the same architecture including the encoding modules and hyper-personalizer. We train those two models on Qualcomm Keyword Speech dataset.

For intuitive comparison, we visualize the means of the hyper-personalizer, using the principal component analysis (PCA). Firstly, we compare the vanilla and VoP models in terms of the means computed by their hyper-personalizer for two different PIDs (PID1 and PID2). To compare the VoP and vanilla models according to the dispersion of means, we re-scale each mean by $L1$ normalization and compute a covariance matrix from the sample-wise concatenated normalized means. In Fig. 4a, we show the means of several testing samples from the two models for each PID, regardless of the class labels. The blue and red circles represent the means from VoP model for PIDs 1 and 2, respectively. Similarly, the green and black circles are those for the vanilla models. We see that the means from our VoP model are more concentrated compared to the means from the vanilla model. In Table 1, for the PIDs, we also provide the variances of the normalized means on each PCA axes (PC1 and PC2). We set the variance of VoP on PC1 as 1 and report the relative variances for clear understanding. Compared to the vanilla model, all the variances of the VoP model are even and very small. Thus, the KL divergence in (10) pushes each output of hyper-personalizer to mimic the distribution of the sample-specific prior following the distribution of the corresponding personality. This sample-wise regulation can give representativeness of personality to each sample, which is very important in the on-the-fly personalization.

Table 2: Keyword spotting accuracy(%) on Qualcomm Keyword Speech dataset.

Method	Closed-set	Open-set
Baseline	87.46 ± 1.68	74.45 ± 0.77
Baseline w/ Dropout	81.77 ± 1.75	77.35 ± 1.90
Baseline w/ samovar (2fc)	17.25 ± 1.42	24.35 ± 1.84
Baseline w/ samovar (1fc)	87.47 ± 1.27	81.43 ± 1.02
VoP	92.80 ± 1.40	83.60 ± 0.84

Table 3: Ablation studies on Qualcomm Keyword Speech dataset under closed-set setting.

Method	No. hidden units	α	Accuracy(%)
VoP with one hyper-personalizer	32	5e-4	42.35 ± 1.34
VoP	32	5e-4	92.80 ± 1.40
VoP (α = 5e-5)	32	5e-5	88.10 ± 0.84
VoP (α = 0; Vanilla)	32	0	68.40 ± 2.12
VoP with 64 hidden units	64	5e-4	85.71 ± 0.70

Finally, we verify the effect of the VoP loss on the personalization of the hyper-perosnlizer for multiple PIDs. For this purpose, we visualize the means, outputs of hyper-perosnlizers, of two different PIDs, for the VoP and vanilla models in Figs. 4b and 4c, respectively. The light red and light blue circles represent means from the hyper-perosnlizer respective to PID regardless of four classes. The red and blue crosses denote the proto-mean of PID1 and PID2, respectively. In vanilla case depicted in Fig. 4b, outputs are scattered sporadically and there is little correlation among the means within the same PID. Interestingly, in Fig. 4c, which is the case of the VoP model, means are well clustered near the proto-mean according to each PID. Note that the KL divergence of the VoP loss enforce the mean of a sample to follow the distribution of the corresponding personality. Namely, the VoP loss has no explicit design, such as contrastive loss, to separate the samples with different PIDs. Nevertheless, the KL divergence successfully groups the samples depending on their personalities.

From this analysis, the KL divergence loss considers estimating the reliable variational distribution to a sample-specific prior. By minimizing cross-entropy and KL divergence together by the VoP loss, each personalized weight distribution can be unique during optimized to both the target task and the personality. Also, sample-wise generated model can achieve representativeness corresponding to personality with sample-wise KL regularization.

Result We compare our VoP with multiple variants of the baseline model. As in (Tang et al., 2018), we first include the dropout regularization on the baseline (Baseline w/ Dropout). Also, we applied samovar (Iakovleva et al., 2020), a Bayesian weight generation technique, to the baseline (Baseline w/ samovar). Since samovar generates layer's weights based on the set-to-set relationship between support and query sets for few-shot classification task, we apply samovar splitting a minibatch into two sets (considering 30% for each personality as their support set) for weight generation. For the proposed VoP, we personalize two last fully-connected (fc) layers.

Table 4: Adequacy for no. the enrollment samples.

No. samples	5	4	3	2	1
Accuracy(%)	92.8±1.4	92.3±1.8	93.3±1.0	92.7±1.6	92.1±1.7

In Table 2, we report the accuracy scores. In the open-set setting, the dropout regularization is slightly effective. But, it results in a degraded performance on the closed-set setting. Whereas, the proposed VoP significantly improves the baseline in both settings by 4.99% and 9.15%. As samovar is designed for the last classification layer, we first apply it on the last fc layer of the baseline. Also, for fair comparison with our VoP, we also apply samovar on the last two fc layers. In the open-set setting, samovar decently personalizes the last fc layer, but is less effective on the closed-set setting. Also, it fails to personalize more than the final classification layer, yielding severe performance drop in both settings. Note that, our VoP outperforms all the variants of the baseline at least 4.98% and 2.23% on closed- and open-set settings, respectively.

This is because hyper-personalizer of VoP is well-designed for giving the representativeness to each generated model from the single sample. Namely, the hyper-personalizer is beneficial for such few enrollment samples. Hence, our VoP is effective to the KWS task which needs personalization.

Ablation studies To analyze the efficacy of VoP, we provide ablation studies for several important factors: the number of the hyper-personalizer, the number of hidden units per hyper-personalizer, and α. In Table 3, we report the accuracy scores of five variants where the best performed one is bold-faced. Notice that we personalize the last fully-connected layer when using one hyper-personalizer, and the last two ones for two hyper-personalizers. When using less number of the hyper-personalizer, the accuracy score is degraded by 50.10% due to weak personalization. Next, as qualitatively studied in Sec. 4.1, on using the vanilla VoP (α = 0), the personalization is unreliable compared to α = 5e-4. Similar tendency is shown for α = 5e-5. Furthermore, when doubling the number of hidden units, VoP suffers from performance degradation, as well. From these ablation studies, we see that those factors are important for successful personalization via forwarding in VoP. We also show the relationship between the test performance and the size of enrollment sample in Table 4. Until reducing the number of enrollment samples from 5 to 2, the performance is maintained. And, at using just one sample for enrollment, there is only 0.6% performance degradation. This means that VoP gives representativeness to the independent sample by regularizing every generated model from a single sample with the KL loss on training.

Table 5: Open set speaker verification equal error rates (lower is better) on VoxCeleb1 test set.

Method	Objective	Equal error rates(%)	
		Fast ResNet-34	VGG-M-40
Baseline	Angular Prototypical	5.46±0.02	8.13±0.24
VoP		5.24±0.08	7.91±0.14

Table 6: Few-shot classification accuracy on *mini*ImageNet. The baseline is re-implemented following the literature.

Method	Backbone	5-way accuracy(%)	
		1-shot	5-shot
Baseline	ConvNet-64	49.42 ± 0.73	67.39 ± 0.62
VoP		51.43 ± 0.54	68.44 ± 0.64

4.2. Speaker Verification

Speaker verification (SV) is a task to dichotomize whether an utterance belongs to a specific speaker based on the enrolled utterances. To apply the proposed VoP to this task, we employ the Fast ResNet-34 (Chung et al., 2020) and the VGG-M-40 (Nagrani et al., 2017) as baseline networks. Here, considering each speaker as a personality, we personalize the last fully connected layer of each baseline. Note that, in the benchmark VoxCeleb1 (Nagrani et al., 2017) dataset, all testing speakers are not pre-defined in the training set. Hence, both baselines are designed to solve metric learning problem in their literature. To apply VoP, we take the angular prototypical loss, commonly used in the baselines, as our task loss. Then, we use these two samples to calculate each proto-mean and proto-variance. For each 200-sized minibatch, we randomly select one person for generating personal weights from proto-mean. As VoP computes the metric loss in the same space, it helps to generate metric learning space for a given proto-set. α is set as 1e-7.

Dataset VoxCeleb1 contains 153,516 utterances for 1,251 celebrities, which are extracted from YouTube videos. This data consists of 55% of the male speakers and 45% of the female speakers, which is gender balanced. The speakers cover a different accents, ethnicities and ages.

Result Following the typical evaluation protocol (Chung et al., 2020), we sample 10 four-second temporal crops at regular interval from each test sample, and compute similarities with all possible pairs (10 × 10). Table 5 shows the equal error rates on the VoxCeleb1 test set. VoP generates personalized weights with the 10 crops from a target speaker. We see that the proposed VoP show better performance than both baselines due to personalized metric feature space.

4.3. Few-Shot Classification

We also apply our VoP on few-shot classification task on the *mini*ImageNet (Vinyals et al., 2016). We employ the Prototypical Net (Snell et al., 2017) (Proto-Net) as the baseline, where the embedding architecture is composed of four convolutional blocks. Each block is comprised of 64 3 × 3 convolutions, batch normalization layer, a ReLU nonlinearity and a 2 × 2 max-pooling layer. When applied to the images of *mini*ImageNet, this architecture results in a 1,600-dimensional output space. In this task, we consider the personality of each episode. Namely, the hyper-personalizer is trained to produce the weight distribution of each episode from the support set. Specifically, we personalize the convolution filter of the last block. Since the target categories keep changed in each episode, the baseline doesn't include the classification layer. Hence, we only add the KL divergence loss on their basic loss where α is empirically set to 5e-4. In testing, the personalized embedding architecture can produce embedded features specialized to the personality of the given episode via forwarding the support set. Except for the loss function, we follow the experimental setting of the baseline *e.g.* learning rate, optimizer, episode configuration.

Dataset The *mini*Imagenet dataset consists of 60,000 color images of size 84 × 84 with 100 classes where each class includes 600 images. We followed the split introduced by (Vinyals et al., 2016): 64, 16, and 20 classes for training, validation and testing, respectively. The 16 validation classes is used for monitoring generalization performance.

Result Following the standard setting adopted by the Proto-Net, we conducted 5 way 1-shot and 5-shot classification tasks. We train using 30-way episodes for 1-shot classification and 20-way episodes for 5-shot classification. We match train shot to test shot and each class contains 15 query points per episode. Table C shows the accuracy scores for 1-shot 5-way and 5-shot 5-way where each is averaged over 600 test episodes and are reported with 95% confidence intervals. Even though using the same architecture and computational cost with the baseline, the proposed VoP yields higher accuracy scores for both tasks by 2.01% and 1.05%, each. Accordingly, we see that the proposed method is effective to personalize the embedding space by forwarding merely a few training examples.

5. Conclusions

We proposed Variational On-the-Fly Personalization (VoP), a novel personalization method that can produce a personalized network via forwarding a small amount of personal data on-the-fly. The proposed VoP can effectively estimate the weight distribution suitable for an individual without additional training using a large amount of personal data. Through extensive experiments on the three important tasks including the open-set problem, we showed that VoP successfully generates an accurately personalized model without increasing the computational cost. Also, as additional training is not required, it can be easily applied to various scenarios on edge devices e.g. mobile and IoT devices.

Acknowledgements

Nojun Kwak was supported by the National Research Foundation of Korea (NRF) grant (2021R1A2C3006659) and IITP grant [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)], both funded by the Korea government (MSIT). The research that is the subject of this paper, and the paper itself, was solely funded by Qualcomm Technologies, Inc.

References

AntixK. https://github.com/AntixK/PyTorch-VAE, 2018.

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. In *The 35th International Conference on Machine Learning*. 2018.

Castorini. Honk: Cnns for keyword spotting. https://github.com/castorini/honk, 2017.

Chen, J. and Ran, X. Deep learning with edge computing: a review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

Cheng, Z., Sun, H., Takeuchi, M., and Katto, J. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Choi, S., Seo, S., Shin, B., Byun, H., Kersner, M., Kim, B., Kim, D., and Ha, S. Temporal convolution for real-time keyword spotting on mobile devices. In *Interspeech*, 2019.

Chung, J. S., Huh, J., Mun, S., Lee, M., Heo, H.-S., Choe, S., Ham, C., Jung, S., Lee, B.-J., and Han, I. In defence of metric learning for speaker recognition. In *Interspeech*, 2020.

ClovaAI. Clova baseline system for the voxceleb speaker recognition challenge 2020. https://github.com/clovaai/voxceleb_trainer, 2020.

Das, D., Yun, S., and Porikli, F. Confess: A framework for single source cross-domain few-shot learning. In *International Conference on Learning Representations*, 2022.

Deng, Y., Kamani, M. M., and Mahdavi, M. Adaptive personalized federated learning, 2020. Forthcoming.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning with theoretical guarantees: a model-agnostic meta-learning approach. In *The 34th Conference on Neural Information Processing Systems*. 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Gal, Y. Uncertainty in deep learning. *Cambridge*, 2016.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. E. Meta-learning probabilistic inference for prediction. In *The Sixth International Conference on Learning Representation*, 2018.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. In *The 34th International Conference on Machine Learning*. 2017.

Hanzely, F. and Richtárik, P. Federated learning of a mixture of global and local models, 2020. Forthcoming.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

Iakovleva, E., Verbeek, J., and Alahari, K. Meta-learning with shared amortized variational inference. In *The 37th International Conference on Machine Learning*, 2020.

Jiang, Y., Konečnỳ, J., Rush, K., and Kannan, S. Improving federated learning personalization via model agnostic meta learning, 2019. Forthcoming.

Jin, Q., Yang, L., and Liao, Z. Adabits: Neural network quantization with adaptive bit-widths. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Kim, B., Lee, M., Lee, J., Kim, Y., and Hwang, K. Query-by-example on-device keyword spotting. https://developer.qualcomm.com/project/keyword-speech-dataset, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2013. Forthcoming.

Ko, T., Chen, Y., and Li, Q. Prototypical networks for small footprint text-independent speaker verification. In *International Conference on Acoustics, Speech, & Signal Processing*, 2020.

Kundu, J. N., Venkat, N., Babu, R. V., et al. Universal source-free domain adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Lee, J., Cho, S., and Beack, S.-K. Context-adaptive entropy model for end-to-end optimized image compression. In *The 36th International Conference on Machine Learning*. 2018.

Li, X., Wei, X., and Qin, X. Small-footprint keyword spotting with multi-scale temporal convolution. In *Interspeech*, 2020.

Long, M., Zhu, H., Wang, J., and Jordan, M. I. Unsupervised domain adaptation with residual transfer networks. In *The 30th Conference on Neural Information Processing Systems*. 2016.

Luo, Z., Zou, Y., Hoffman, J., and Fei-Fei, L. Label efficient learning of transferable representations across domains and tasks. In *The 31th Conference on Neural Information Processing Systems*. 2017.

Minnen, D., Ballé, J., and Toderici, G. D. Joint autoregressive and hierarchical priors for learned image compression. *Advances in Neural Information Processing Systems*, 2018.

Motiian, S., Jones, Q., Iranmanesh, S. M., and Doretto, G. Few-shot adversarial domain adaptation. In *The 31th Conference on Neural Information Processing Systems*. 2017.

Nagrani, A., Chung, J. S., and Zisserman, A. Voxceleb: a large-scale ppeaker identification dataset. In *Interspeech*, 2017.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *The 5th International Conference on Learning Representations*, 2016.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *The 31th Conference on Neural Information Processing Systems*. 2017.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: relation network for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018.

Tang, R. and Lin, J. Honk: a pytorch reimplementation of convolutional neural networks for keyword spotting, 2017. Forthcoming.

Tang, R. and Lin, J. Deep residual learning for small-footprint keyword spotting. In *International Conference on Acoustics, Speech, & Signal Processing*, 2018.

Tang, R., Wang, W., Tu, Z., and Lin, J. An experimental analysis of the power consumption of convolutional neural networks for keyword spotting. In *International Conference on Acoustics, Speech, & Signal Processing*, 2018.

Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. Simultaneous deep transfer across domains and tasks. In *IEEE/CVF International Conference on Computer Vision*. 2015.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *The 30th Conference on Neural Information Processing Systems*. 2016.

Wang, J., Wang, K.-C., Law, M. T., Rudzicz, F., and Brudno, M. Centroid-based deep metric learning for speaker recognition. In *International Conference on Acoustics, Speech, & Signal Processing*, 2019a.

Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., and Chen, M. In-edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, 2019b.

Yang, S., Wang, Y., van de Weijer, J., Herranz, L., and Jui, S. Generalized source-free domain adaptation. 2021.

Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. Few-shot learning via embedding adaptation with set-to-set functions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

Zhang, Y., Suda, N., Lai, L., and Chandra, V. Hello edge: keyword spotting on microcontrollers. *CoRR*, abs/1711.07128, 2017.

A. Proof for $KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k))$

In eq. (10), we analytically calculated the KL divergence with two assumptions:

- The sample-specific prior follows the distribution of the personality, i.e. $p(\omega|x_i^k) \simeq p(\omega|k)$.
- The variational distribution and the personality distribution are an uncorrelated Gaussian distribution.

Based on the assumptions, we prove eq. (10):

$$
\begin{aligned}
&KL(q_\theta(\omega|x_i^k)||p(\omega|x_i^k)) \simeq KL(\mathcal{N}(\omega|\mu_i,\Sigma_i)||\mathcal{N}(\omega|\bar{\mu}_k,\bar{\Sigma}_k)) \\
&\quad = \mathbb{E}_{\omega\sim q_\theta}[\ln q_\theta(\omega|x_i^k) - \ln p(\omega|x_i^k)] \\
&\quad = \mathbb{E}_{\omega\sim q_\theta}\left[\frac{1}{2}\ln\frac{|\bar{\Sigma}_k|}{|\Sigma_i|} - \frac{1}{2}(\omega-\mu_i)^T\Sigma_i^{-1}(\omega-\mu_i) + \frac{1}{2}(\omega-\bar{\mu}_k)^T\bar{\Sigma}_k^{-1}(\omega-\bar{\mu}_k))\right] \\
&\quad = \frac{1}{2}(\ln\frac{|\bar{\Sigma}_k|}{|\Sigma_i|} - \mathbb{E}_{\omega\sim q_\theta}[\mathrm{tr}[(\omega-\mu_i)(\omega-\mu_i)^T\Sigma_i^{-1}]] + \mathbb{E}_{\omega\sim q_\theta}[(\omega-\bar{\mu}_k)^T\bar{\Sigma}_k^{-1}(\omega-\bar{\mu}_k)]) \\
&\quad = \frac{1}{2}(\ln\frac{|\bar{\Sigma}_k|}{|\Sigma_i|} - \mathrm{tr}[\Sigma_i\Sigma_i^{-1}] + \mathbb{E}_{\omega\sim q_\theta}[(\omega-\bar{\mu}_k)^T\bar{\Sigma}_k^{-1}(\omega-\bar{\mu}_k)]) \\
&\quad = \frac{1}{2}(\ln\frac{|\bar{\Sigma}_k|}{|\Sigma_i|} - Z + \mathrm{tr}\{\bar{\Sigma}_k^{-1}\Sigma_i\} + (\mu_i-\bar{\mu}_k)^T\bar{\Sigma}_k^{-1}(\mu_i-\bar{\mu}_k)) \\
&\quad = \frac{1}{2}(\ln\prod_{z=1}^{Z}\bar{\sigma}_{(k,z)}^2 - \ln\prod_{z=1}^{Z}\sigma_{(i,z)}^2 - Z + \sum_{z=1}^{Z}\frac{\sigma_{(i,z)}^2}{\bar{\sigma}_{(k,z)}^2} + \sum_{z=1}^{Z}\frac{(\mu_{(i,z)}-\bar{\mu}_{(k,z)})^2}{\bar{\sigma}_{(k,z)}^2}) \\
&\quad = \frac{1}{2}\sum_{z=1}^{Z}(\ln\bar{\sigma}_{(k,z)}^2 - \ln\sigma_{(i,z)}^2 - 1 + \frac{\sigma_{(i,z)}^2}{\bar{\sigma}_{(k,z)}^2} + \frac{(\mu_{(i,z)}-\bar{\mu}_{(k,z)})^2}{\bar{\sigma}_{(k,z)}^2})
\end{aligned}
$$

where $z \in [Z]$ is the index of multivariate Gaussian dimension.

B. Architecture of Hyper-Personalizer

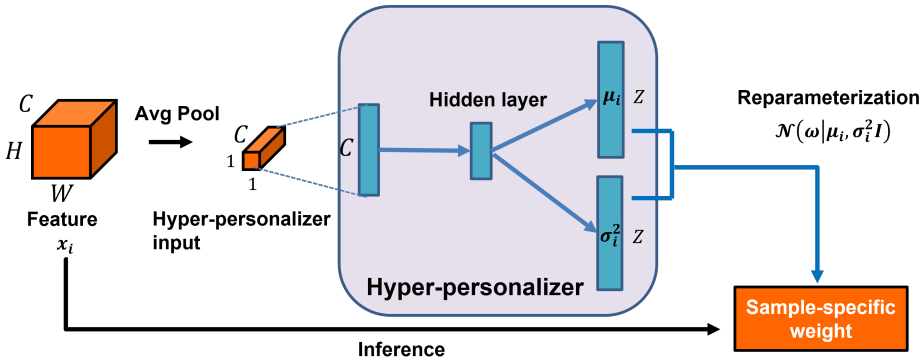


Figure 5: The overall architecture of the hyper-personalizer.

Fig. 5 depicts the detailed architecture of the hyper-personalizer in the proposed VoP. Similar to the encoder of VAE (AntixK, 2018), our hyper-personalizer is designed by a multi-layer perceptron with a hidden layer which is shared to two output layers.

For ith sample, taking the spatially average-pooled encoded feature as an input, the hyper-personalizer produces two outputs: μ_i and σ_i^2 (both are in $\mathbb{R}^Z$). The sizes of μ_i and σ_i^2 are equal to that of a target personalized layer (reparameterized sample-specific weight). Hence, the hyper-personalizer computes the mean and variance for each element of the sample-specific weight.

C. Comparison with Other Bayesian Approaches in Few-Shot Classification

Table 7: Comparison of the proposed VoP and another Bayesian approach, samovar (Iakovleva et al., 2020), in few-shot classification.

Method	Features	5-way Acc.(%)		ΔAcc. to baseline	
		1-shot	5-shot	1-shot	5-shot
Baseline (versa, (Gordon et al., 2018))	Conv-5	53.4 ± 1.9	67.4 ± 0.9	-	-
samovar		52.4 ± 0.3	69.8 ± 0.2	-1.0	1.6
Baseline (Proto-Net, (Snell et al., 2017))	Conv-4	49.4 ± 0.7	67.4 ± 0.6	-	-
VoP (Ours)		51.4 ± 0.5	68.4 ± 0.6	2.0	1.0

Table 8: Comparison of the proposed VoP and existing few-shot classification methods for different backbone features.

Method	Features	5-way Acc. (%)	
		1-shot	5-shot
Matching Nets (Vinyals et al., 2016)	Conv-4	46.6	60.0
Meta LSTM (Ravi & Larochelle, 2016)		43.4 ± 0.8	60.6 ± 0.7
MAML (Finn et al., 2017)		48.7 ± 1.8	63.1 ± 0.9
RelationNets (Sung et al., 2018)		50.4 ± 0.8	65.3 ± 0.7
Proto-Net (Snell et al., 2017)		49.4 ± 0.7	67.4 ± 0.6
VoP (Ours)		51.4 ± 0.5	68.4 ± 0.6
versa (Gordon et al., 2018)	Conv-5	53.4 ± 1.9	67.4 ± 0.9
samovar (Iakovleva et al., 2020)		52.4 ± 0.3	69.8 ± 0.2
VoP (Ours)		52.8 ± 0.2	70.3 ± 0.3

In Sec. 4.3, we extended the proposed VoP to the few-shot classification. In this field, versa, a Bayesian approach, has tried to generate the weight of the last linear classification layer, for given task (support and query sets). While our VoP is based on the sample-specific posterior approximation for personalization, it starts from the formulation of the posterior approximation of the entire support set. Namely, versa is consciously designed to this few-shot classification task. On the top of the ConvNet backbone, versa generates weight and bias vectors for each class in class independent manner. To this end, it manipulates the size of conv-4 block feature with another 1×1 convolution layer, so that their embedding backbone, called as Conv-5, is larger than ours. Although versa uses more computation and is fitted to this few-shot classification task, the proposed VoP, simple extension to this task, outperforms it by 1.0% on the 5-shot setting in Table 7. Also, as shown in Table 8, when using the same backbone (Conv-5), our VoP achieves higher accuracy than versa by a large margin 2.9% in the 5-shot setting.

Similar to versa, samovar also generates the task-specific weight of the classification layer. Based on the versa's formulation, samovar exploits the relationship between the support set and the union of support and query sets. In specific, the posterior for the union set is approximated, using the prior given by the support set. Hence, samovar is more deliberately formulated for this few-shot classification task.

Furthermore, samovar has several advantages in the experimental setting. First, samovar employs versa as the baseline which is better than our baseline (Proto-Net). versa inevitably requires another 1×1 convolution layer and generates the linear classification layer on top of the layer. Whereas, we personalize the Conv-4 block (the last layer of ConvNet backbone), itself. Hence, we doesn't need more computation. Second, samovar exploits task conditioning and auxiliary co-training schemes to further improve the performance while VoP doesn't.

Due to the different assumption in formulation, we can't direct apply VoP on the versa baseline. Also, the source code of samovar is not released, and it was impossible to reproduce samovar. Accordingly, it is difficult to compare our VoP and samovar on fair experimental setting of the baseline and w/o additional schemes. Therefore, keeping the difference in backbones, we first compare VoP and samovar in terms of the improvement to the corresponding baselines (ΔAcc.) in Table 7. In the 5-shot setting, samovar yields better ΔAcc. than ours by 0.6%, since it is based on the set-to-set relationship. Our VoP method is not designed to this few-shot classification task. Nevertheless, VoP is successfully extended to task in both 1- and 5-shot settings (better than the proto-net baseline by 2.0% and 1.0%, respectively). Especially, the proposed VoP achieves a larger ΔAcc. than samovar in challenging 1-shot setting (2.0% vs -1.0% in ΔAcc.). Next, in Table 8 we apply the same backbone of samovar (Conv-5) to our VoP. Here, VoP outperforms samovar for both settings.

Table 9: Comparison of VoP and BatchNorm Calibration (BC) in KWS and few-shot classification for accuracy (%).

Method	Keyword spotting	Few-shot classification	
		1-shot	5-shot
Baseline	87.46 ± 1.68	49.42 ± 0.73	67.39 ± 0.62
BatchNorm Calibration	90.70 ± 1.03	49.62 ± 0.38	67.19 ± 0.44
VoP	92.80 ± 1.40	51.43 ± 0.54	68.44 ± 0.64

Table 10: Comparison of VoP and the simple fine-tuning of the global model with five enrollment samples using different learning rates (lr) in KWS according to accuracy (%).

Method	Baseline	Baseline w/ Dropout
Pre-trained	87.46 ± 1.68	81.77 ± 1.75
Fine-tuning (lr=1e-3)	58.55 ± 0.92	59.90 ± 0.57
Fine-tuning (lr=1e-4)	72.95 ± 0.77	77.10 ± 0.98
Fine-tuning (lr=1e-5)	87.87 ± 0.09	78.95 ± 0.21
Fine-tuning (lr=1e-6)	87.80 ± 0.28	77.70 ± 0.84
VoP	92.80 ± 1.40	–

D. Comparison with Other Naive Techniques

On the KWS tasks with the closed-set setting, we compare the proposed VoP with two naive techniques, BatchNorm Calibration (BC) (Jin et al., 2020) and fine-tuning the global model with few enrollment samples (5), which are straightforwardly applicable to the personalization task. Since BC (Jin et al., 2020) inevitably requires the batch normalization layer, we apply it to a network with batch normalization layers. In specific, we pass the target data through the network, and then re-calculate the batch normalization parameters for their personalization. As demonstrated in Table 9, BC improves the baseline by 3.3%, but is worse than our VoP. This result means that the personalization of only the parameters of the batch normalization layers has a limitation since most of parameters are still fixed (not personalized). Contrarily, our VoP can effectively regularize the entire parameters of the target layers to represent discriminative characteristics among personalities.

Next, we compare the proposed VoP with the simple fine-tuning of the pre-trained global models varying the learning rates. In Table 10, the pre-trained (Baseline) is Baseline of Tables 2 and 9, and the pre-trained (Baseline w/ Dropout) is Baseline w/ Dropout of Table 2. We fine-tune each model with five enrollement samples for 10 times on different learning rates. In the case of Baseline w/ Dropout, the fine-tuning degrades the pre-trained global model for all the learning rates. Only when the dropout scheme is removed (Baseline), fine-tuning with such small learning rates can increase the performance of the pre-trained model, but the gaps are meagre (0.41% and 0.34%). On the other hand, the proposed VoP notably outperforms all the variants of the fine-tuning technique, and achieves significant performance improvement of the pre-trained baseline.

E. Computational Overhead

The overhead of the hyper-personalizer (HP) itself is large as in the conventional hyper-network approaches. But, after training, in VoP, this overhead happens during enrollment process only. As we abolish the HP after enrollment, computation cost of VoP is same with the baseline in testing. We measure the training overhead by (Mac / no. params.) in Table 11.

Table 11: Mac and parameters

Module	Baseline	1st HP	2nd HP
Mac	10M	88M	0.1M
Memory	0.95M	0.1M	0.05M

F. Privacy Issue

In recent years, deep learning-based applications and services have focus on data privacy and security issues. Our proposed Variational On-the-Fly Personalization (VoP) also handles the private data. However, VoP is free from the above issues because VoP generates the personal weights at the edge device with few forwardings. In other words, at the server, the global model and hyper-personalizer are trained with the given training dataset. After training, the personalization process is on-the-fly conducted on the client's edge device without training, and the server does not access this process. We design the personalization process with a small amount of the enrollment samples at the personal device which can protect privacy.

G. Implementation Details

For all experiments, we implement VoP using the Pytorch libraries with a single 1080ti GPU. For the keyword spotting and speaker verification tasks, we used the released official implementations (Castorini, 2017; ClovaAI, 2020) following a MIT License. In few-shot classification, we re-implemented the baseline (Snell et al., 2017) on the Pytorch libraries. All experiments are conducted three times with a GeForce GTX 1080 Ti GPU and we report the mean and standard deviation.