DUET: A Tuning-Free Device-Cloud Collaborative Parameters Generation Framework for Efficient Device Model Generalization

Zheqi Lv*
Zhejiang University
China
zheqilv@zju.edu.cn

Wenqiao Zhang*
National University of Singapore
Singapore
wenqiao@nus.edu.sg

Shengyu Zhang
Zhejiang University
China
sy_zhang@zju.edu.cn

Kun Kuang†
Zhejiang University
Key Laboratory for Corneal Diseases Research of Zhejiang Province
China
kunkuang@zju.edu.cn

Feng Wang
Alibaba Group
China
windpls@gmail.com

Yongwei Wang
Shanghai Institute for Advanced Study of Zhejiang University
China
yongwei.wang@ntu.edu.sg

Zhengyu Chen
Zhejiang University
China
chenzhengyu@zju.edu.cn

Tao Shen
Zhejiang University
China
tao.shen@zju.edu.cn

Hongxia Yang
Alibaba Group
China
firewater1984@gmail.com

Beng Chin Ooi
National University of Singapore
Singapore
ooibc@comp.nus.edu.sg

Fei Wu†
Zhejiang University
Shanghai Institute for Advanced Study of Zhejiang University
Shanghai AI Laboratory
China
wufei@zju.edu.cn

ABSTRACT

Device Model Generalization (DMG) is a practical yet under-investigated research topic for on-device machine learning applications. It aims to improve the generalization ability of pre-trained models when deployed on resource-constrained devices, such as improving the performance of pre-trained cloud models on smart mobiles. While quite a lot of works have investigated the *data distribution shift* across clouds and devices, most of them focus on model fine-tuning on personalized data for individual devices to facilitate DMG. Despite their promising, these approaches require on-device re-training, which is practically infeasible due to the overfitting problem and high time delay when performing gradient calculation on real-time data. In this paper, we argue that the computational cost brought by fine-tuning can be rather unnecessary. We consequently present a novel perspective to improving DMG without increasing computational cost, *i.e.*, device-specific parameter generation which directly maps data distribution to parameters. Specifically, we propose an efficient **D**evice-clo**U**d collaborative paramet**E**rs gener**a**t**I**on framework (**DUET**). DUET is deployed on a powerful cloud server that only requires the low cost of forwarding propagation and low time delay of data transmission between the device and the cloud. By doing so, DUET can rehearse the device-specific model weight realizations conditioned on the personalized real-time data for an individual device. Importantly, our DUET elegantly connects the cloud and device as a "duet" collaboration, frees the DMG from fine-tuning, and enables a faster and more accurate DMG paradigm. We conduct an extensive experimental study of DUET on three public datasets, and the experimental results confirm our framework's effectiveness and generalisability for different DMG tasks.

CCS CONCEPTS

• **Information systems** → **Mobile information processing systems**; **Personalization**; • **Human-centered computing** → **Mobile computing**.

KEYWORDS

Device Model Generalization, Device-Cloud Collaboration, On-Device Machine Learning, Parameters Generation

ACM Reference Format:
Zheqi Lv, Wenqiao Zhang, Shengyu Zhang, Kun Kuang, Feng Wang, Yongwei Wang, Zhengyu Chen, Tao Shen, Hongxia Yang, Beng Chin Ooi, and Fei Wu. 2023. DUET: A Tuning-Free Device-Cloud Collaborative Parameters Generation Framework for Efficient Device Model Generalization. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3543507.3583451

*These authors contributed equally to this research.
†Corresponding authors.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583451

1 INTRODUCTION

The high performance of Deep Neural Networks (DNNs) [9, 26] is tempered by the huge parameter size of intricate design patterns and the demand for high computational costs. This situation greatly hinders the application of intelligent services in mobile phones and Internet of Things (IoT) devices since their hardware resources are tightly constrained by the form factor, battery, and heat dissipation. Therefore, on-device machine learning (DML) that goes beyond DNNs by exploiting lightweight neural networks (LNNs) for task-specific learning and inference on devices is gaining traction, such as MobileNets [11, 12, 24], DIN [41], SASRec [15], GRU4Rec [10]. Along with the rapid development of cloud computing, the predominant DML paradigm is not simply learning on the device. It often collaboratively connects with a powerful cloud server with perceivably limitless resources. As shown in Figure 1(a), the cloud trains a global primary model conditioned on rich data collected from different devices, and then the local device performs inference using the trained model. Some researches [22, 30, 31] have achieved great success on image classification tasks and recommender systems, demonstrating the potential value of the device-cloud collaboration scheme for DML.

Unfortunately, due to the heterogeneity of data distributions across devices, a primary cloud model trained on the data aggregated from all the devices might not generalize well and is typically less personalized to each particular device[4, 16, 18, 32, 33, 35–37]. For instance, user data originating from devices of different geographical locales is potentially heterogeneous, leading to performance degradation for personalized DML [20, 21, 30]. In other words, distribution shifts between the cloud and different devices require personalization of the cloud model before deployment on the device. To alleviate such *data distribution shift* problem, Device Model Generalization (DMG) is needed to improve the generalization ability of a pre-trained model on a specific device. In this varying distribution setting, a commonly known DMG technique, as illustrated in Figure 1(b), fine-tunes the pre-trained cloud model based on the current device data to mitigate the distribution shift issue, yielding an improvement of DMG for personalized learning[1].

Despite their promising, fine-tuning based approaches may not be the true savior to resolve the DMG problems, due to the two key challenges summarized as follows. (i) **Overfitting Issue and Annotation Demand.** The distribution of the real-time data input to the device is dynamic, and sometimes it may change drastically. For example, in the product recommendation task shown in Figure 1, the user behavior sequence consists of dozens of recently clicked items by the users, which may exhibit different preferences in a short time snippet (t: 🏀 $\rightarrow t+1$: 🥪 $\rightarrow t+2$: 🖊). In order to achieve personalization, fine-tuning based DMG methods require re-training the model conditioned on those very limited samples with rapidly changing distributions on the device. Such a learning paradigm may cause overfitting and performance degradation. To make the matters worse, in vision tasks, the data on the device is generally not annotated. To use the fine-tuning based methods, it is necessary to perform real-time annotation on the real-time data generated on the device. However, such frequent annotation is time-consuming and human-labor intensive to acquire for a device, which may not even be feasible in real-life applications. (ii) **High Time-Resource Cost.** In addition to overfitting and extra annotation, on-device fine-tuning is time-consuming as it incurs numerous calculations on the gradients to update the model parameters, which is undesirable when the device applications typically have the real-time requirement constraint. It also consumes substantial amounts of device computing resources, thus leading to power consumption problem of smart devices. Therefore, these on-device training methods are not suitable for real-time DML in resource-constrained devices. Summing up, the premise of better DMG is to mitigate the aforementioned issues. The limitations require us to revisit the design of DMG solution for device-cloud collaboration.

In this paper, we propose a novel framework for DML, called **D**evice-clo**U**d collaborative paramet**E**rs genera**T**ion framework, (**DUET**) to address the aforementioned limitations. The core idea of our approach for DMG is to learn a device-specific model weight generator that dynamically adjusts from personalized data to solve a learning task. As shown in Figure 1(c), our framework comprises the following parts: (1) **Universal Meta Network** (UMN) first collects data from all the devices over a time period, labels them on the cloud server and then trains a primary model with these labeled data. For DMG, we divide the trained model into static layers and dynamic layers. The parameters of the static layers (backbone) are fixed, while the parameters of the dynamic layers (classifier) are dynamically generated based on device-specific real-time data in inference. (2) The **Personalized Parameters Generator** (PPG) leverages the *HyperNetworks* [7] that efficiently share parameters across devices to generate separate classifier parameters for every device. Each device, with its unique real-time samples, passes as input to the designed PPG on the cloud to produce its personalized classifier weights. Then, the cloud will deliver the dynamic layer parameters to the device and enable real-time on-device inference with the personalized model. It is worth noting that the PPG is parallelly trained with UMN based on global data and only requires feedforward computation of local real-time data. The small time delay is due to the transmission of few data between the device and cloud, which produces a real-time DMG scheme. (3) We also propose a **Stable Weight Adapter** (SWA) which aims to generate stable parameters for dynamic layers by multiple PPGs due to the observation that one single PPG suffers from the performance oscillation problem. SWA measures the correlations between individualized trained PPGs, based on which a self-corrected adapter adaptively predict the optimal parameters of the primary model on the device. From the perspective of machine learning, the utilization of the correlation and similarity among related learning methods can be regarded as a form of inductive transfer. It can introduce the *inductive bias* [2] to make the combined learning method prefer the correct hypothesis, thereby improving the performance.

In summary, as illustrated in Figure 1(d), compared to the fine-tuning approaches that incur *high calculation cost and high annotation demand*, our proposed DUET entails *zero calculation cost and zero annotation demand*. Our proposed DUET is arguably more pragmatic and suitable for real-time DMG. In this work, we make the following four key contributions:

- To the best of our knowledge, we are the first to incorporate the model parameter generation into device-cloud collaboration without expensive fine-tuning in on-Device Machine Learning.

[1]This paper studies the DMG problem in which primary models on the cloud and device have the same architecture but different parameters.

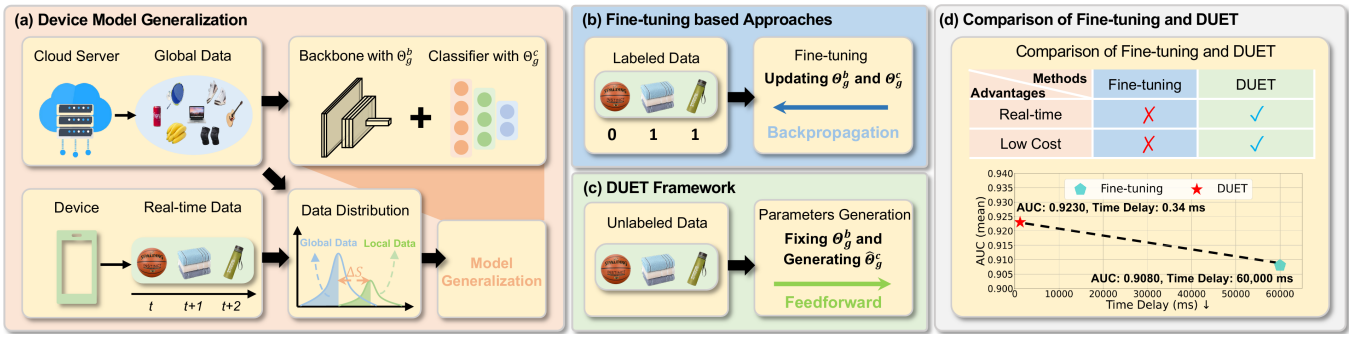

Figure 1: (a) describes the device model generalization in device-cloud collaboration, ΔS indicates the data distribution shift of global and local data. (b) and (c) are overviews of fine-tuning based approaches and our DUET, respectively. Ω_b and Ω_c respectively denoted the parameters of backbone and classifier. (d) is the comparison of fine-tuning and DUET (`Time Delay: 0.34ms (DUET)` $\ll$ `60,000ms (Fine-tuning))`, `AUC: 0.9230 (DUET)` $>$ `0.9080 (Fine-tuning))`.

- We propose the personalized parameter generator which directly maps the device-specific data to model parameters for fast model personalization.
- We design a stable weight adapter to reduce the performance oscillation of the dynamic model and further boost generalization across heterogeneous devices.
- We conduct extensive experiments with various baselines on real-world benchmark datasets. The results demonstrate the consistent superiority and generalizability of DUET.

2 RELATED WORK

Lightweight Neural Network. The performance of traditional neural networks for different research tasks [5, 23, 38, 39] is already impressive. However, when the model is deployed on a device, the device's storage space and computing power have to be considered. Therefore, many lightweight CNN models [8, 11–13, 19, 24, 27, 40] have been proposed in recent years. SqueezeNet [13] reduces the number of parameters by extensively using fire modules with 1×1 convolutions. MobileNetV1 [12] decomposes traditional convolution kernels into depth-wise convolution kernels and point convolution kernels. MobileNetV2 [24] introduces inverted residuals and linear bottlenecks. MobileNetV3 [11] builds the network based on AutoML, manually fine-tunes the optimization to obtain the best network structure, and improves the performance and efficiency of the activation function. ShuffleNetV1 [40] uses channel shuffle to enhance information exchange between channel groups. ShuffleNetV2 [19] introduces channel split to improve inference speed. EfficientNet [27] uses a neural network architecture (NAS) with a hybrid scaling method. GhostNet [8] applies a linear transformation layer with fewer parameters to generate ghost feature maps. These models achieve good performance with small parameters and FLOPs, but the number of parameters still limits the performance and generalization ability.

HyperNetwork. HyperNetwork [1, 3, 6, 7, 25, 28, 29, 34] is a neural network that generates its parameters for another neural network. When HyperNetwork was first proposed by Ha et al. [7], it achieved model compression by reducing the number of parameters the model needs to train. Subsequently, the research on HyperNetwork gradually increased. Oscar et al. [3] studied parameter initialization for HyperNetwork. At the same time, the research of HyperNetwork is applied to various tasks, such as continual learning [28], graph [34], meta-learning [29], federated learning [25], Etc. HyperNetwork-related research has mainly focused on generating different network parameters from different data inputs in the past two years. For example, HyperStyle [1] and HyperInverter [6] both use HyperNetwork to generate different decoder parameters for different images, thereby improving the quality of the reconstructed images. In our work, we adapt HyperNetwork to the device-cloud system with unique challenges arising from the problem setup.

3 METHODOLOGY

This section describes our proposed **D**evice-clo**U**d collabora**T**ive paramet**E**rs enera**T**ion framework (**DUET**) for device model generalization. We shall present each module and its training strategy.

3.1 Problem Formulation

In the problem of device model generalization (DMG) in the device-cloud collaboration system, we have access to a set of devices $\mathcal{D} = \{d^{(i)}\}_{i=1}^{\mathcal{N}_d}$, each device with its personal i.i.d history samples $\mathcal{S}_{H^{(i)}} = \{x_{H^{(i)}}^{(j)}, y_{H^{(i)}}^{(j)}\}_{j=1}^{\mathcal{N}_{H^{(i)}}}$ and real-time samples $\mathcal{S}_{R^{(i)}} = \{x_{R^{(i)}}^{(j)}\}_{j=1}^{\mathcal{N}_{R^{(i)}}}$ in current session, where $\mathcal{N}_d$, $\mathcal{N}_{H^{(i)}}$ and $\mathcal{N}_{R^{(i)}}$ represent the number of devices, history data and real-time data, respectively. The goal of DMG is to generalize a trained global cloud model $\mathcal{M}_g(\cdot;\Theta_g)$ learned from $\{\mathcal{S}_{H^{(i)}}\}_{i=1}^{\mathcal{N}_d}$ to each specific local device model $\mathcal{M}_{d^{(i)}}(\cdot;\Theta_{d^{(i)}})$ conditioned on real-time samples $\mathcal{S}_{R^{(i)}}$, where Θ_g and $\Theta_{d^{(i)}}$ respectively denote the learned parameters for the global and local models.

$$\underbrace{\textbf{DUET}}_{\text{DMG Model}} : \underbrace{\mathcal{M}_g(\{\mathcal{S}_{H^{(i)}}\}_{i=1}^{\mathcal{N}_d};\Theta_g)}_{\text{Global Cloud Model}} \rightarrow \underbrace{\mathcal{M}_{d^{(i)}}(\mathcal{S}_{R^{(i)}};\Theta_{d^{(i)}})}_{\text{Local Device Model}}. \quad (1)$$

Figure 2 illustrates the overview of our DUET framework which consists of three modules to improve the generalization ability of the trained models on the device: (a) *Universal Meta Network* (UMN) aims to learn a global benchmark model based on the global data (in Sec. 3.2); (b) *Personalized Parameters Generator* (PPG) that generates the network parameters for local device-specific model (in Sec. 3.3); (c) *Stable Weight Adapter* (SWA) presents the personalized parameter optimization strategy for robust DMG learning (in Sec. 3.3.3).

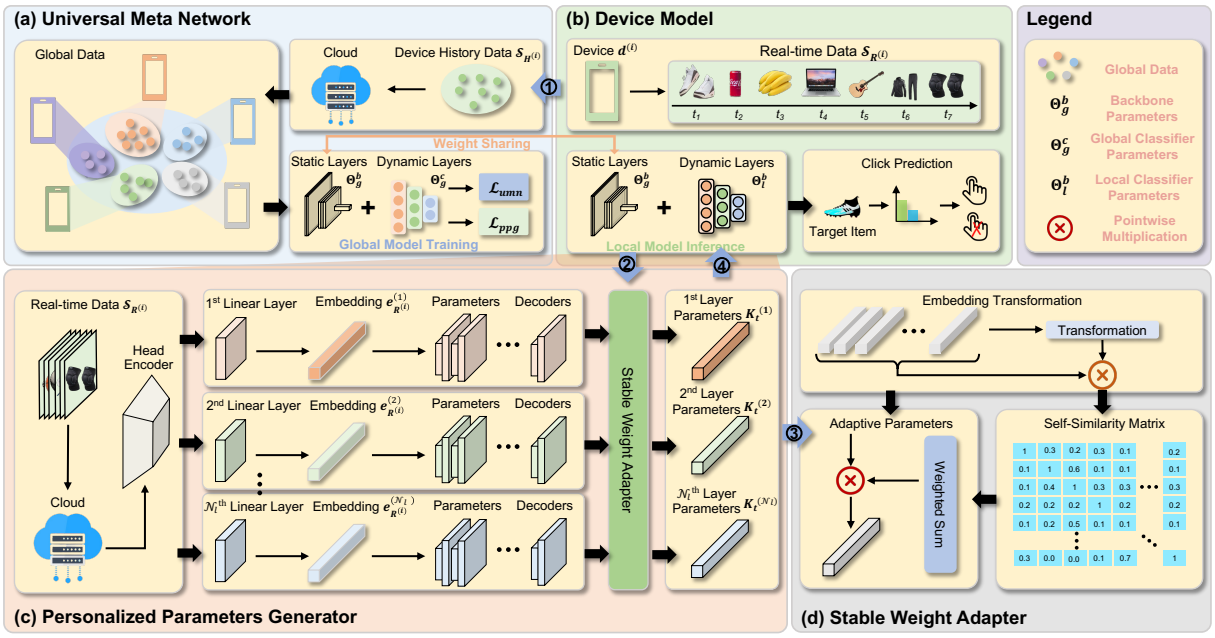

Figure 2: Overview of the proposed DUET. The UMN is trained on the cloud that contains a backbone with parameters Θ_b and a classifier with parameters Θ_c. The PPG is deployed on the cloud, which generates and delivers the personalized parameters Θ_l^c of dynamic layers for the device classifier based on the distribution of the real-time samples uploaded from the device. The SWA aims to reduce the performance oscillation of single PPG, accelerating the convergence and improving the prediction stability.

3.2 Universal Meta Network.

In **Universal Meta Network** (UMN) (Figure 2(a)), we train a primary model with a backbone and a classifier for the global cloud model development. Given a set of devices $\mathcal{D} = \{d^{(i)}\}_{i=1}^{\mathcal{N}_d}$ and their corresponding history data $\mathcal{S}_{H^{(i)}} = \{x_{H^{(i)}}^{(j)}, y_{H^{(i)}}^{(j)}\}_{j=1}^{\mathcal{N}_{H^{(i)}}}$, the goal of the proposed UMN can thus be formulated as the following optimization problem:

$$\min_{\Theta_g^b, \Theta_g^c} \mathcal{L}_{umn} = \sum_{i=1}^{\mathcal{N}_d} \sum_{j=1}^{\mathcal{N}_{R^{(i)}}} D_{ce}(y_{H^{(i)}}^{(j)}, \Omega(x_{H^{(i)}}^{(j)}; \Theta_g^b); \Theta_g^c)), \tag{2}$$

where $D_{ce}(;\Theta_g^b)$ denotes the cross-entropy between two probability distributions. $\Omega(x_{H^{(i)}}^{(j)};\Theta_g^b)$ is the backbone extracting features from sample $x_{H^{(i)}}^{(j)}$. Θ_g^b and Θ_g^c are the learnable parameters for the backbone and classifier, respectively.

In our DMG setting, we decouple the joint backbone and classifier training scheme as modeling the "static layers" and "dynamic layers" to achieve the personalized model generalization:

- **Static Layers.** The backbone with Θ_g^b learned from global data can accurately map the user's behavior into the feature space. We fixed the backbone as "static layers" to generate a generalized representation for any given input concerning the global data distribution.
- **Dynamic Layers.** Depending on the user's behavior, the personalized samples obtained from a specific device are input to our proposed PPG to learn personalized classifier weights Θ_l^c . The improvement of personalized generalization can be achieved by just adjusting the classifier.

3.3 Personalized Parameters Generator.

The Personalized Parameters generator (PPG) can generate the dynamic parameters for the personalized classifier with Θ_l^c conditioned on the real-time samples from a specific device, which aims to improve the generalization to different distributions of data. We shall start by introducing the definition of **HyperNetwork** [7] and then propose our DMG mechanisms of parameters generation.

3.3.1 Retrospect of HyperNetwork. First, we will outline the procedure for using a *HyperNetwork* to output the weights of a feedforward convolutional network that performs the learning task. The *HyperNetwork* regards the parameters $K^{(n)}$ for n^{th} layer ($n \in \mathcal{N}_l$) of the CNN filter as a matrix of $\mathbb{R}^{C_{in}f_w \times C_{out}f_h}$, where $\mathcal{N}_l$ is the depth of the main network, the convolutional kernel contains $C_{in} \times C_{out}$ filters and each filter correspond the dimensions of $f_w \times f_h$. For n^{th} layer, the *HyperNetwork* is a two-layer MLP $g(\cdot;\Theta_p)$ with parameters Θ_p receives a layer embedding $z^{(n)} \in \mathbb{R}^{C_z}$ ($C_z \ll C_{in}f_w \times C_{out}f_h$) as input to and predicts $K^{(n)}$, which can be regard as the a matrix factorization scheme as follows:

$$K^{(n)} = g(z^{(n)};\Theta_p), \forall n = 1, \cdots, \mathcal{N}_l. \tag{3}$$

In the *training procedure*, $z^{(1)} \sim z^{(\mathcal{N}_l)}$ and $g(\cdot)$ are randomly initialized. As in a regular neural network, the network learns the mapping relationship between samples x to y. Notably, the gradients are returned to $z^{(n)}$ and $g(\cdot)$ instead of $K^{(n)}$, which saves more space and computing power than storing parameters in $K^{(1)} \sim K^{(\mathcal{N}_l)}$. In the *inference procedure*, $K^{(n)}$ is generated in blocks. The base convolution kernels $K^{'(n)}$ of all convolutional layers to be generated are set as convolution kernels with a dimension of $C_{in}^{'} f_w^{'} \times C^{'}{}_{out} f^{'}{}_h$. All convolutional layers to be generated need to conform that, (1)

$f_w^{'}$ and f'_h are required to be equal to f_w and f_h, respectively. (2) C_{in} and C_{out} are integer multiples of $C_{in}^{'}$ and C'_{out}, respectively. Each base convolution kernel is generated by a base latent vector $z^{'(n)} \in \mathbb{R}^{C_z^{'}}$. The modular generation is below:

$$K^{(n)} = \begin{pmatrix} K_{1,1}^{'(n)} & \cdots & K_{1,j}^{'(n)} \\ \vdots & \ddots & \vdots \\ K_{i,1}^{'(n)} & \cdots & K_{i,j}^{'(n)} \end{pmatrix} = g \begin{pmatrix} z_{1,1}^{'(n)} & \cdots & z_{1,j}^{'(n)} \\ \vdots & \ddots & \vdots \\ z_{i,1}^{'(n)} & \cdots & z_{i,j}^{'(n)} \end{pmatrix} \quad (4)$$
$$\triangleq g(z^{(n)}), i = \frac{C_{in}}{C_{in}^{'}}, j = \frac{C_{out}}{C_{out}^{'}}, \forall n = 1, \cdots, \mathcal{N}_l.$$

After retrospecting characteristics of *HyperNetwork*, which seems naturally suitable for learning a diverse set of personalized models. As *HyperNetwork* dynamically generates target networks conditioned on the input embeddings, *i.e.*, the "Dynamic Layers" of the primary model can be modeled by *HyperNetwork*. However, our observation (in Appendix) indicates that directly utilizing the *HyperNetwork* may not satisfactorily resolve the DMG problem for two key reasons:

- **Weak Correlation.** The original *HyperNetwork* uses a random latent vector z to initialize the model that lacks the strong correlation between parameters generation and a specific device, which may yield a performance decay.
- **Unstable Prediction.** The empirical experiments indicate that the performance of *HyperNetwork* is intuitively unstable during training and inference, mainly because a single *HyperNetwork* is hard to measure the parameters.

To this end, we carefully design the Personalized Parameters Generator (PPG) and Stable Weight Adapter (SWA) to deal with the above limitations.

3.3.2 Device-specific Parameters Generation. Considering the *Weak Correlation* between *HyperNetwork* and the random latent vector z, as shown in Figure 2(b), we propose to model the "Dynamic Layers" of the primary model by replacing the z with specific samples from devices in inference. Further, to satisfy the architecture consistency of pre-trained classifier and "Dynamic Layers", we develop a hierarchical *HyperNetworks* to generate its parameters.

In device inference, we use the real-time samples $\mathcal{S}_{R^{(i)}} = \{x_{R^{(i)}}^{(j)}\}_{j=1}^{\mathcal{N}_{R^{(i)}}}$ in each session to generate the model parameters.

To generate the parameters for n^{th} layer of "Dynamic Layers" in the primary model, we develop a layer encoder to represent the n^{th} layer parameters as an embedding $\boldsymbol{e}_{R^{(i)}}^{(n)}$. To model relationships of different layers, instead of constructing the one-to-one encoder-layer correspondence, the $\boldsymbol{e}_{R^{(i)}}^{(n)}$ share one encoder neck but use different linear layers to change the real-time data features.

$$\boldsymbol{e}_{R^{(i)}}^{(n)} = L_{\text{layer}}^{(n)}(E_{\text{share}}(\mathcal{S}_{R^{(i)}})), \forall n = 1, \cdots, \mathcal{N}_l, \quad (5)$$

where $E_{\text{share}}(\cdot)$ represents the shared encoder neck. $L_{\text{layer}}^{(n)}(\cdot)$ is a linear layer used to adjust the output of $E_{\text{share}}(\cdot)$ to the n^{th} dynamic layer features.

We treat it as a matrix $K^{(n)} \in \mathbb{R}^{N_{in} \times N_{out}}$, where N_{in} and N_{out} represent the number of input neurons and output neurons of the n^{th} FCL, respectively. Then we use the generator $g(\cdot)$ to convert the real-time data features into parameters of the primary models by $K_{R^{(i)}}^{(n)} = g^{(n)}(\boldsymbol{e}_{R^{(i)}}^{(n)})$. Specifically, we input $\boldsymbol{e}_{R^{(i)}}^{(n)}$ into the following two MLP layers to generate parameters according to the consistent structure of "Dynamic Layers" of the primary model.

$$\begin{aligned} \boldsymbol{w}_{R^{(i)}}^{(n)} &= (W_1 \boldsymbol{e}_{R^{(i)}}^{(n)} + B_1) W_2 + B_2, \\ K_{R^{(i)}}^{(n)} &= \boldsymbol{w}_{R^{(i)}}^{(n)} + \boldsymbol{b}_{R^{(i)}}^{(n)}, \end{aligned} \quad (6)$$

where weights of the two MLP layers are denoted by W_1 and W_2, respectively. B_1 and B_2 indicate the biases.

In cloud training, all layers of the PPG are optimized together with the static layers of the primary model that are conditioned on the global history data $\mathcal{S}_{H^{(i)}} = \{x_{H^{(i)}}^{(j)}, y_{H^{(i)}}^{(j)}\}_{j=1}^{\mathcal{N}_{H^{(i)}}}$, instead of optimizing the static layers of the primary model first and then optimizing the PPG. The PPG loss function $\mathcal{L}_{ppg}$ is defined as follows:

$$\min_{\Theta_g^b, \Theta_p} \mathcal{L}_{ppg} = \sum_{i=1}^{\mathcal{N}_d} \sum_{j=1}^{\mathcal{N}_{R^{(i)}}} \gamma^t D_{ce}(y_{H^{(i)}}^{(j)}, \Omega(x_{H^{(i)}}^{(j)}; \Theta_g^b); g(\boldsymbol{e}_{R^{(i)}}^{(n)}; \Theta_p))), \quad (7)$$

where γ is a hyperparameter used to adjust the training. When the γ is closer to 1, the PPG considers all samples in each session as equally important. Otherwise, PPG pays more attention to the earlier samples in each session.

Here we use a special group-wise convolution in PPG, so that the entire framework can generate parameters for the primary models during training and inference in parallel. This greatly improves PPG training and inference efficiency and makes it easier to deploy in real environments.

3.3.3 Stable Weight Adapter. As described in Sec. 3.3.1, directly using *HyperNetwork* often produces large oscillations during learning, and thus yields *Unstable Prediction*. To resolve this issue, we propose the Stable Weight Adapter (SWA) module to improve the prediction stability.

First, we develop $\mathcal{N}_p$ PPGs to generate a set of parameters rather than a single generator are trained in the same way, denoted by $W_1^{'}$. Then splicing multiple $W_1^{'}$ into a matrix, denoted as $W_1^{'} = \{W_{1,1}^{'}, W_{1,2}^{'}, ..., W_{1,m}^{'}\}$. $W_{i,j}^{'}$ denoted the i-th MLP layer of the j-th generator. Then we can get the similarity between $W_{1,i}^{'}$ and $W_{1,j}^{'}$, thus a self-similarity matrix S of dimension $m \times m$ is obtained by $S = W_1^{'} * (W_1^{'})^T$.

Summing S by row, we can get the weight vector $\boldsymbol{p}^{'} = \{p_1^{'}, p_2^{'}, ..., p_m^{'}\}$ with dimension $m \times 1$. Among them, p_i can be regarded as the importance of $W_{1,i}^{'}$ in the multiple generators. We also set temperature to adjust the final weight vector $\boldsymbol{p}$,

$$p_i = \text{Softmax}(\frac{p_i^{'}/\tau}{\sum_{j=1}^{m} p_j^{'}}) \quad (8)$$

Then we can calculate the final W_1 and W_2 like,

$$W_1 = \sum_{i=1}^{m} p_i * W_{1,i}^{'} \quad (9)$$

Finally, we use Eq. (9) to get W_1 and W_2, and get the model parameters after replacing W_1 and W_2 in Eq. (6).

4 EXPERIMENTS

We conduct a range of sequential recommendation and facial expression recognition experiments on three public datasets to demonstrate the effectiveness of the proposed DUET framework.

Table 1: Performance comparison of the proposed method and baselines on sequential recommendation datasets (`Movielens-1M` and `Movielens-100k`). LR^* indicates that the DMG models require the label information of real-time data from device. ↑ and ↓ respectively indicate a larger and smaller score has better performance. Acronym notations of baselines can be found in Sec. 4.1.2. We color each row as the best, second best, and third best.

Baselines	DMG Methods	LR^*	Movielens-1M Dataset				Time Delay	Movielens-100k Dataset				Time Delay
			AUC(mean) ↑	AUC(std) ↓	Param.	FLOPs		AUC(mean) ↑	AUC(std) ↓	Param.	FLOPs	
DIN [41]	-		0.9077	0.0006	896.83K	1.82M	0	0.8348	0.0045	271.83K	0.60M	0
	Fine-tuning	✓	0.9080	0.0006			≥60,000ms	0.8429	0.0045			≥60,000ms
	DUET (w/o SWA)		0.9233	0.0008			≥0.34ms	0.8560	0.0030			≥0.15ms
	DUET		0.9230	0.0007				0.8581	0.0055			
SASRec [15]	-		0.9280	0.0007	888.63K	1.99M	0	0.8721	0.0026	263.63K	0.77M	0
	Fine-tuning	✓	0.9279	0.0006			≥60,000ms	0.8719	0.0027			≥60,000ms
	DUET (w/o SWA)		0.9313	0.0007			≥0.34ms	0.8721	0.0027			≥0.15ms
	DUET	✓	0.9326	0.0003				0.8723	0.0009			
GRU4Rec [10]	-		0.9279	0.0016	886.62K	1.94M	0	0.8723	0.0017	261.62K	0.72M	0
	Fine-tuning	✓	0.9286	0.0014			≥60,000ms	0.8711	0.0019			≥60,000ms
	DUET (w/o SWA)		0.9311	0.0014			≥0.34ms	0.8751	0.0055			≥0.15ms
	DUET		0.9311	0.0004				0.8755	0.0017			

Table 2: Performance comparison of the proposed method and baselines on facial expression recognition datasets (`CK+`).

Baselines	DMG Methods	CK+ Dataset				Time Delay
		AUC(mean) ↑	AUC(std) ↓	Param.	FLOPs	
MobileNetV3 -Large [11]	-	76.97	2.76	2.69M	0.27B	0
	DUET (w/o SWA)	79.49	2.84			≥51ms
	DUET	82.45	2.67			
MobileNetV3 -Small [11]	-	67.88	2.15	1.24M	0.06B	0
	DUET (w/o SWA)	71.82	2.91			≥31ms
	DUET	75.15	1.41			

4.1 Experimental Setup

4.1.1 Datasets. **Sequential Recommendation.** We evaluate DUET on `Movielens-1M` and `Movielens-100k` [2], two widely used public benchmarks in the recommendation tasks. Following conventional practice, all user-item pairs in the dataset are treated as positive samples. In the training and test sets, the user-item pairs that do not exist in the dataset are sampled at 1:4 and 1:100, respectively, as negative samples [10, 15, 41]. **Facial Expression Recognition.** We evaluate our method on `CK+` [3] [14, 17]. `CK+` is a facial expression recognition dataset containing 593 videos of 123 people, of which 327 videos are annotated with expressions. The labels contain seven basic emotions: anger, contempt, disgust, fear, happiness, sadness, and surprise. To simulate a real device-cloud collaborative environment, we treat each video as a session and keep each video's first and last three frames. Since each video records a person from no expression to an exaggerated expression consistent with the label, we set the label of the first three frames as natural and the label of the last three frames as the emotion label of the video.

4.1.2 Baselines. In the sequential recommendation task, DIN [41], SASRec [15], and GRU4Rec [10], three of the most widely used methods in the academia and industry, are chosen as the baselines. In the facial expression recognition task, we choose MobileNetV3 [11] as the baseline, which is one of the most popular lightweight networks.

4.1.3 Implementation Details. **Training Procedure**. When training DUET, in the sequential recommendation task, we input the most recent click sequence as the real-time samples to PPG to get the dynamic layers' parameters. Then we update the obtained parameters to the primary model's dynamic layers and model the mapping relationship between samples and labels in this session. In the Facial Expression Recognition task, the first frame of each video is set to a real-time sample. After updating the parameters similar to the above process, we model the mapping relationship between samples and labels in this video. Gradients are passed back to UMN's static layers and PPG.

Inference Procedure. The inference process of baselines is performed on the device. Fine-tuning based DMG methods need to fine-tune the base model with real-time data and then make inferences on the device. In the inference process of DUET, the device first uploads real-time samples to the cloud. PPG then generates the parameters of dynamic layers in UMN according to the distribution information of real-time data samples and sends them to the device. The updated parameters of dynamic layers will be sent to devices, and make inferences together with static layers. The device only requests model parameters at the beginning of each session/video. In the actual deployment, the recommendation task regards opening the APP and refreshing the page as the beginning of a session. The vision task's content at a fixed time interval is regarded as a video. Table 5 in the Appendix shows the hyperparameters and training schedules of DUET on the three datasets.

4.2 Experimental Results.

Results of Sequential Recommendation. Table 1 summarizes the quantitative results of our framework and other DMG methods on `Movielens-1M` and `Movielens-100k` dataset. From this table, we have the following findings: (1) Almost all DMG models can improve the baseline's performance (AUC (std)) across the two datasets, which demonstrates the application value of model generalization on the device. (2) The effect of model fine-tuning is insignificant, and we observe performance degradation in some cases,

[2] http://grouplens.org/datasets/movielens/
[3] https://www.jeffcohn.net/Resources/

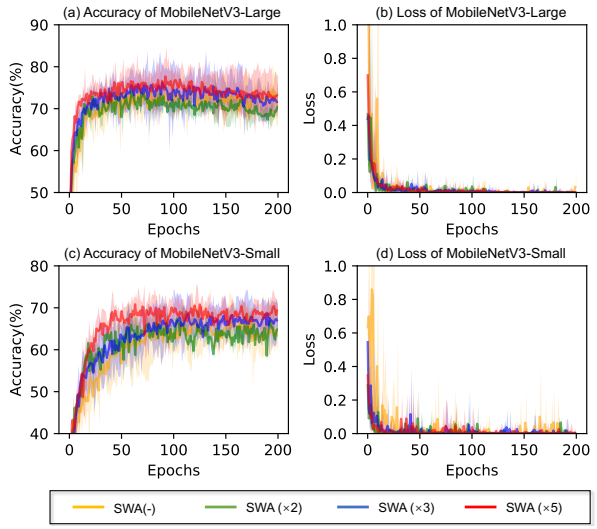

Figure 3: Effects of the number of SWAs in training.

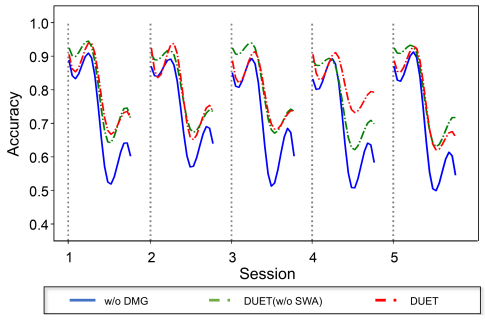

Figure 4: Performance comparison on each session.

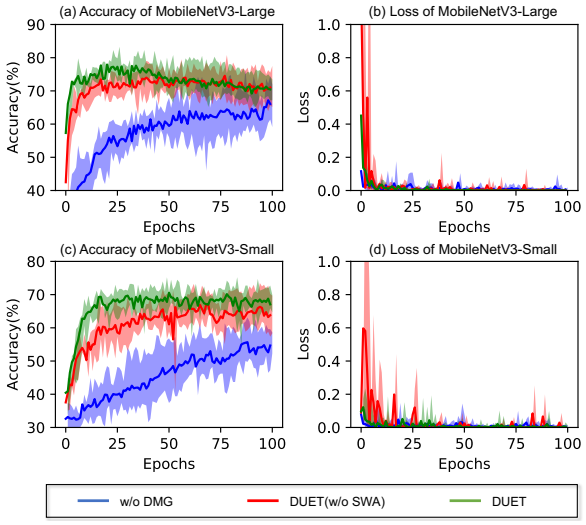

Figure 5: Training visualization of DUET and baselines.

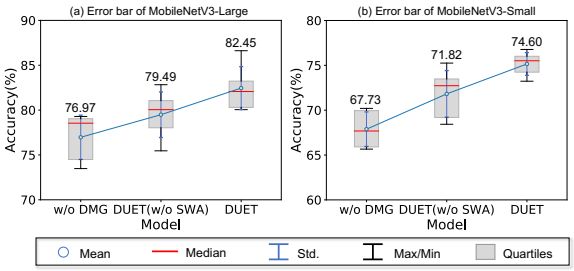

Figure 6: Detailed performance comparison of DUET and baselines.

e.g., fine-tuning for baseline SASRec (Row. 6). This phenomenon is reasonable as the fine-tuning model may encounter the over-fitting issue when trained on limited real-time data. In addition, it also causes a high-time delay, which is impractical for applications on the device. (3) DUET and its variant DUET (w/o SWA) both outperform fine-tuning-based DMG model by a large margin, *e.g.*, **DUET (w/o SWA) improves 0.0153 AUC (mean) (Row. 3) and DUET improves 0.0150 AUC (mean) (Row. 4)** using DIN baseline, respectively. Notably, it enables a real-time recommendation with an extremely low time delay. (4) DUET (w/o SWA) and DUET produce a similar performance in all datasets. However, Table 1 suggests that DUET consistently maintains a stable performance across all the baselines, *i.e.*, the AUC (std) of DUET is smaller than DUET (w/o SWA), *e.g.*, Row. 7 *vs* Row. 8. These results indicate that the SWA can bring improved robustness for the DMG task.

We also present the Param. (size of parameters) and FLOPs (floating-point operations per second) in Table 1. Intuitively, models with low FLOPs and parameters are easier to deploy on devices. Note that the Parameters and FLOPs shown in the table are the primary models that need to be deployed on the device. Since PPG is deployed on the cloud, the parameters and FLOPs of the primary model under DUET framework are the same as the baselines.

Results of Facial Expression Recognition. Table 2 reports the performance comparison between our model and the adopted baselines on the facial expression recognition task. The conclusions are generally consistent with the experiments on the recommendation task. The main differences are: (1) Since the annotation of real-time on-device samples is not routinely available, the fine-tuning experiments cannot be performed. (2) Compared with DUET (w/o SWA), the performance improvement of DUET on vision tasks is more significant than the recommendation task. Our intuition is that the real-time data distribution gap between real-time data in the computer vision task is smaller than the recommendation task, where the SWA module can further boost the parameters generation-based DMG framework.

Summing up, the results presented above confirm the superiority of the proposed tuning-free device-cloud collaborative parameters generation framework, which exhibits a faster and more accurate DMG paradigm simultaneously.

4.3 In-Depth Analysis.

We conducted the additional experiments on the CK+ dataset to verify the strength of the proposed DUET.

Table 3: Time delay caused by device-cloud communication, ⇑ and ⇓ represent upload (device → cloud) and download (cloud → device), respectively. Data transmission || embedding transmission indicates the different upload settings.

Datasets	Models	Size	4G: 5MB/s	4G: 15MB/s	5G :50MB/s	5G: 100MB/s
CK+	MobileNetV3-Large	⇑: 0.14MB || 0.25KB ⇓: 5.31MB	⇑: 0.03s || 0.05ms ⇓: 1.06s	⇑: 0.01s || 0.017ms ⇓: 0.35s	⇑: 0.003s || 0.005ms ⇓: 0.10s	⇑: 0.001s || 0.003ms ⇓: 0.05s
	MobileNetV3-Small	⇑: 0.14MB || 0.25KB ⇓: 3.06MB	⇑: 0.03s || 0.05ms ⇓: 0.61s	⇑: 0.01s || 0.017ms ⇓: 0.20s	⇑: 0.003s || 0.005ms ⇓: 0.06s	⇑: 0.001s || 0.003ms ⇓: 0.03s
Movielens-1M	DIN	⇑: 25.63KB || 8.06KB ⇓: 8.06KB	⇑: 5.13ms || 0.05ms ⇓: 1.60ms	⇑: 1.71ms || 0.017ms ⇓: 0.53ms	⇑: 0.51ms || 0.005ms ⇓: 0.16ms	⇑: 0.26ms || 0.003ms ⇓: 0.08ms
	SASRec					
	GRU4Rec					
Movielens-100k	DIN	⇑: 7.32KB || 0.25KB ⇓: 8.06KB	⇑: 1.46ms || 0.05ms ⇓: 1.60ms	⇑: 0.49ms || 0.017ms ⇓: 0.53ms	⇑: 0.15ms || 0.005ms ⇓: 0.16ms	⇑: 0.07ms || 0.003ms ⇓: 0.08ms
	SASRec					
	GRU4Rec					

Table 4: Effect of the number of SWAs on performance. The best results are highlighted in bold.

Baselines	DMG Methods	$\mathcal{N}_p$	Accuracy(%)	Std(%)
MobileNetV3-Large	-	-	76.97	2.76
	DUET	-	79.49	2.84
		2	80.40	4.26
		3	81.88	2.81
		5	**82.45**	**2.67**
MobileNetV3-Small	-	-	67.88	2.15
	DUET	-	71.82	2.91
		2	73.11	3.80
		3	73.74	2.19
		5	**75.15**	**1.41**

Detail Performance Analysis. To further study the effectiveness of DUET, we visualize the accuracy and loss in training and inference in Figure 5, the corresponding mean value, median value, standard derivation, maximum/minimum, and quartiles of real-time inference are shown in Figure 6. All experiments are repeated five times. As shown in Figure 5, the loss of DUET and DUET (w/o SWA) decreases faster than baseline, demonstrating its superior convergence speed in training. However, this figure also indicates that the DUET (w/o SWA) brings a huge performance improvement but alone with the prediction oscillation problem. In addition, DUET with the SWA kindly solved this problem, resulting in a stable prediction in training and inference, which verifies the effectiveness of the proposed SWA module. More analysis about the stability is shown in sec. 4.3. In Figure 6, the dark-colored line is the Mean value of the data, and the light-colored area is the fluctuation range, that is, the Maximum and Minimum values. For more clarity, the curves in the figure are obtained by 1:25 sampling, *i.e.*, one epoch in the figure represents the actual 25 epochs. It also shows that DUET achieves consistent superiority in terms of all of the metrics.

Generalizability. Figure 4 depicts the performance change of different architectures over distribution shifts. Specifically, during two adjacent sessions (*e.g.*, 1 & 2), we simulate the distribution shifts by selecting heterogeneous data samples. At the beginning of each session, we update the base models with DUET and DUET (w/o SWA). MobileNetV3-Large (w/o DMG) refers to the base model trained on the cloud with all device data without any DMG methods. According to the results, we observe that DUET shows consistent performance improvement over the baseline against distribution shifts, which demonstrate the high generalization capability of the proposed DUET.

Prediction Stability of Different $\mathcal{N}_p$. To build insights of stability on the SWA module, we perform the ablation study that sets different numbers $\mathcal{N}_p$ of SWAs. Our experiments were repeated five times to observe its effect on accuracy and loss. On the one hand, as shown in Table 4, the best performance is achieved with $\mathcal{N}_p$=5, both in accuracy and standard deviation. When the $\mathcal{N}_p$ is smaller, the standard deviation higher - especially the DUET (w/o SWA) - produces a higher performance fluctuation than baseline (w/o DMG), which indicates that the SWA module can improve the stability in prediction. On the other hand, Figure 3 systematically presents the explicit benefits of the $\mathcal{N}_p$ increase conditioned on two baselines. The figure shows that as the number of SWAs increases, the accuracy curve and loss curve are more stable, which also effectively accelerates the convergence speed and improves the performance in training. These results empirically verified the robustness of the SWA module, which provides a reliable solution that guarantees prediction stability.

Time Delay Analysis Table 3 shows the size and time delay of *uploading* real-time samples and *downloading* dynamic layer model parameters on the device. Across all settings, this table suggests that our method produces a low time delay from 0.08s ∼ 1.06s, which seems acceptable for real-time applications. It is noteworthy that if we upload the embedding of real-time samples, it can generate a faster DMG and protect the user privacy of these samples. The observation above and analysis verify the effectiveness of DUET in implementing the real-time requirement for applications, thereby rendering the practicability for on-device learning.

5 CONCLUSION

In this paper, we propose the DUET for efficient device model generalization by generating adaptive device model parameters from the cloud without on-device training. Our method effectively learns a mapping function from real-time samples to device model parameters, which yields a low-time delay and better device-specific personalization. Extensive experiments conducted on Movielens-1M, Movielens-100k and CK+ show that DUET outperforms fine-tuning methods by a large margin in terms of accuracy and real-time performance, which validates the potential value in practical applications.

ACKNOWLEDGMENTS

This work was supported in part by National Key Research and Development Program of China (2022YFC3340900), National Natural Science Foundation of China (U20A20387, No. 62006207, No. 62037001), Project by Shanghai AI Laboratory (P22KS00111), Program of Zhejiang Province Science and Technology (2022C01044), the StarryNight Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-0010), Fundamental Research Funds for the Central Universities (226-2022-00142, 226-2022-00051), the National Research Foundation, Singapore under its Emerging Areas Research Projects (EARP) Funding Initiative.

REFERENCES

[1] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. 2022. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18511–18521.

[2] Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of artificial intelligence research* 12 (2000), 149–198.

[3] Oscar Chang, Lampros Flokas, and Hod Lipson. 2020. Principled Weight Initialization for Hypernetworks. In *8th International Conference on Learning Representations, ICLR 2020*.

[4] Zhengyu Chen and Donglin Wang. 2021. Multi-Initialization Meta-Learning with Domain Adaptation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1390–1394.

[5] Zhengyu Chen, Teng Xiao, and Kun Kuang. 2022. BA-GNN: On Learning Bias-Aware Graph Neural Network. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 3012–3024.

[6] Tan M Dinh, Anh Tuan Tran, Rang Nguyen, and Binh-Son Hua. 2022. Hyperinverter: Improving stylegan inversion via hypernetwork. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11389–11398.

[7] David Ha, Andrew M. Dai, and Quoc V. Le. 2017. HyperNetworks. In *5th International Conference on Learning Representations, ICLR 2017*.

[8] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. 2020. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1580–1589.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *International Conference on Learning Representations 2016* (2016).

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1314–1324.

[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861 (2017). arXiv:1704.04861 http://arxiv.org/abs/1704.04861

[13] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[14] Takeo Kanade, Jeffrey F Cohn, and Yingli Tian. 2000. Comprehensive database for facial expression analysis. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*. IEEE, 46–53.

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[16] Mengze Li, Ming Kong, Kun Kuang, Qiang Zhu, and Fei Wu. 2020. Multi-task attribute-fusion model for fine-grained image recognition. In *Optoelectronic Imaging and Multimedia Technology VII*, Vol. 11550. SPIE, 114–123.

[17] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. 2010. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*. IEEE, 94–101.

[18] Zheqi Lv, Feng Wang, Shengyu Zhang, Kun Kuang, Hongxia Yang, and Fei Wu. 2022. Personalizing Intervened Network for Long-tailed Sequential User Behavior Modeling. *arXiv preprint arXiv:2208.09130* (2022).

[19] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*. 116–131.

[20] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. 2021. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems* 34 (2021), 15434–15447.

[21] Jed Mills, Jia Hu, and Geyong Min. 2021. Multi-task federated learning for personalised deep neural networks in edge computing. *IEEE Transactions on Parallel and Distributed Systems* 33, 3 (2021), 630–641.

[22] Xufeng Qian, Yue Xu, Fuyu Lv, Shengyu Zhang, Ziwen Jiang, Qingwen Liu, Xiaoyi Zeng, Tat-Seng Chua, and Fei Wu. 2022. Intelligent Request Strategy Design in Recommender System. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 3772–3782.

[23] Fang-Yu Qin, Zhe-Qi Lv, Dan-Ni Wang, Bo Hu, and Chao Wu. 2020. Health status prediction for the elderly based on machine learning. *Archives of gerontology and geriatrics* 90 (2020), 104121.

[24] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[25] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*. PMLR, 9489–9502.

[26] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015*, Yoshua Bengio and Yann LeCun (Eds.).

[27] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[28] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. 2020. Continual learning with hypernetworks. In *8th International Conference on Learning Representations, ICLR 2020*.

[29] Zhou Xian, Shamit Lal, Hsiao-Yu Tung, Emmanouil Antonios Platanios, and Katerina Fragkiadaki. 2021. HyperDynamics: Meta-Learning Object and Agent Dynamics with Hypernetworks. In *9th International Conference on Learning Representations, ICLR 2021*.

[30] Yikai Yan, Chaoyue Niu, Renjie Gu, Fan Wu, Shaojie Tang, Lifeng Hua, Chengfei Lyu, and Guihai Chen. 2022. On-Device Learning for Model Personalization with Large-Scale Cloud-Coordinated Domain Adaption. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. 2180–2190.

[31] Jiangchao Yao, Feng Wang, Xichen Ding, Shaohu Chen, Bo Han, Jingren Zhou, and Hongxia Yang. 2022. Device-cloud Collaborative Recommendation via Meta Controller. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. 4353–4362.

[32] Junkun Yuan, Xu Ma, Defang Chen, Kun Kuang, Fei Wu, and Lanfen Lin. 2022. Domain-specific bias filtering for single labeled domain generalization. *International Journal of Computer Vision* (2022), 1–20.

[33] Junkun Yuan, Xu Ma, Defang Chen, Kun Kuang, Fei Wu, and Lanfen Lin. 2022. Label-Efficient Domain Generalization via Collaborative Exploration and Generalization. In *Proceedings of the 30th ACM International Conference on Multimedia*. 2361–2370.

[34] Chris Zhang, Mengye Ren, and Raquel Urtasun. 2019. Graph HyperNetworks for Neural Architecture Search. In *7th International Conference on Learning Representations, ICLR 2019*.

[35] Fengda Zhang, Kun Kuang, Yuxuan Liu, Long Chen, Jiaxun Lu, Fei Wu, Chao Wu, Jun Xiao, et al. [n. d.]. Towards Multi-level Fairness and Robustness on Federated Learning. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*.

[36] Fengda Zhang, Kun Kuang, Yuxuan Liu, Long Chen, Chao Wu, Fei Wu, Jiaxun Lu, Yunfeng Shao, and Jun Xiao. 2021. Unified group fairness on federated learning. *arXiv preprint arXiv:2111.04986* (2021).

[37] Shengyu Zhang, Xusheng Feng, Wenyan Fan, Wenjing Fang, Fuli Feng, Wei Ji, Shuo Li, Wang Li, Shanshan Zhao, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2023. Video Audio Domain Generalization via Confounder Disentanglement. In *AAAI*.

[38] Wenqiao Zhang, Siliang Tang, Yanpeng Cao, Shiliang Pu, Fei Wu, and Yueting Zhuang. 2019. Frame augmented alternating attention network for video question answering. *IEEE Transactions on Multimedia* 22, 4 (2019), 1032–1041.

[39] Wenqiao Zhang, Lei Zhu, James Hallinan, Shengyu Zhang, Andrew Makmur, Qingpeng Cai, and Beng Chin Ooi. 2022. Boostmis: Boosting medical image semi-supervised learning with adaptive pseudo labeling and informative active annotation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20666–20676.

[40] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6848–6856.

[41] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.