RobustDA

Lightweight Robust Domain Adaptation for Evolving Data at Edge

Guo, Xinyu; Zuo, Xiaojiang; Han, Rui; Ouyang, Junyan; Xie, Jing; Harold Liu, Chi; Zhang, Qinglong; Guo, Ying; Chen, Jing; Chen, Lydia Y.

DOI
10.1109/JETCAS.2024.3478359

Publication date
2024

Document Version
Final published version

Published in
IEEE Journal on Emerging and Selected Topics in Circuits and Systems

RobustDA: Lightweight Robust Domain Adaptation for Evolving Data at Edge

Xinyu Guo, Xiaojiang Zuo, Rui Han, Junyan Ouyang, Jing Xie, Chi Harold Liu, *Senior Member, IEEE*, Qinglong Zhang, Ying Guo, Jing Chen, and Lydia Y. Chen, *Senior Member, IEEE*

***Abstract*— AI applications powered by deep learning models are increasingly run natively at edge. A deployed model not only encounters continuously evolving input distributions (domains) but also faces adversarial attacks from third-party. This necessitates adapting the model to shifting domains to maintain high natural accuracy, while avoiding degrading the model's robust accuracy. However, existing domain adaptation and adversarial attack preventation techniques often have conflicting optimization objectives and they rely on time-consuming training process. This paper presents RobustDA, an on-device lightweight approach that co-optimizes natural and robust accuracies in model retraining. It uses a set of low-rank adapters to retain all learned domains' knowledge with small overheads. In each model retraining, RobustDA constructs an adapter to separate domain-related and robust-related model parameters to avoid their conflicts in updating. Based on the retained knowledge, it quickly generates adversarial examples with high-quality pseudo-labels and uses them to accelerate the retraining process. We demonstrate that, comparing against 14 state-of-the-art DA techniques under 7 prevalent adversarial attacks on edge devices, the proposed co-optimization approach improves natural and robust accuracies by 6.34% and 11.41% simultaneously. Under the same accuracy, RobustDA also speeds up the retraining process by 4.09x.**

***Index Terms*— Robustness, deep learning, edge computing, domain adaptation, adversarial attacks.**

I. Introduction

THE fast development of Deep Neural Networks (DNNs) has lead to a number of successful vision applications at edge [1]. Typically, a DNN model is pre-trained with massive labeled data in the cloud before being deployed at edge. The dynamic nature of edge environment means the model often encounters continuously evolving input distribution [2], known as *domain shifts*) [2], and needs **domain adaptation** (DA) [3], [4] to mitigate its performance depredation. At the same time, the deployed model may face third-party *adversarial attacks* [5], which use crafted adversarial examples to mislead DNN models into making false predictions. For such an issue, **adversarial training** [6] is applied to enhance the adversarial robustness of DNNs. In this context, successfully deploying DNN models at edge requires addressing both domain shifts and adversarial attacks through efficient on-device model retraining.

A. Example Scenario

Fig. 1 (a) illustrates an edge-side digit recognition scenario with three parts. (1) *Offline pre-training in the cloud*. Using the (*source domain*) data, the adversarial pre-training in the cloud boosts the WideResNet-16-2 [7] model's **natural accuracy** [8] (i.e., the model accuracy in inference) and **robust accuracy** [8] (i.e., the accuracy under adversarial attacks) to 97.59% and 88.46%, respectively. (2) *Online DA at edge*. When deployed on a NVIDIA Jetson AGX Xavier device (a commodity edge device), the model continuously encounters 30 new target domains, thus requiring retraining to mitigate natural accuracy loss for each domain. The retraining process must be done using limited resources and unlabeled samples on the device. (3) *Retraining for adversarial attacks*. For each target domain, the model also needs adversarial retraining to maintain its robust accuracy. For example, leveraging the low robustness in the target domain 1, attackers can trick the system into wrongly identifying a digit "2" as "7". It is exceedingly challenging to simultaneously conduct both types of model retraining on resource-constrained edge devices.

1) Dilemma of Natural Accuracy and Robust Accuracy in Model Retraining: Existing online DA methods [9], [10], [11] focus on optimizing natural accuracy by learning generic features. On the other hand, adversarial training methods such as Unsupervised Adversarial Training (UAT) [12], [13] aim at enhancing resistance to adversarial attacks by emphasizing specific robust feature extraction. This means two training methods may have different optimization directions in loss space, as illustrated in Fig. 1 (b). Hence directly combing a UAT method into online DA may lead to natural accuracy

Received 21 June 2024; revised 21 August 2024; accepted 2 October 2024. Date of publication 11 October 2024; date of current version 13 December 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3301500; in part by the National Natural Science Foundation of China under Grant 62272046, Grant 62132019, Grant 61872337, and Grant U21A20519; and in part by the Open Project of Ministry of Education's Key Laboratory of Computing Power Network and Information Security under Grant 2023PY002. This article was recommended by Guest Editor P.-Y. Chen. *(Xinyu Guo and Xiaojiang Zuo contributed equally to this work.) (Corresponding author: Rui Han.)*

Xinyu Guo, Xiaojiang Zuo, Rui Han, Junyan Ouyang, Jing Xie, Chi Harold Liu, and Qinglong Zhang are with Beijing Institute of Technology, Beijing 100081, China (e-mail: 3220221043@bit.edu.cn; 3120195517@bit.edu.cn; hanrui@bit.edu.cn; 3220225183@bit.edu.cn; 1120182942@bit.edu.cn; chiliu@bit.edu.cn; 3120235198@bit.edu.cn).

Ying Guo and Jing Chen are with the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250316, China (e-mail: guoying@sdas.org; jingchen94@163.com).

Lydia Y. Chen is with the Department of Computer Science, Delft University of Technology (TU Delft), 2628 CD Delft, The Netherlands (e-mail: lydiaychen@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2024.3478359.

Digital Object Identifier 10.1109/JETCAS.2024.3478359

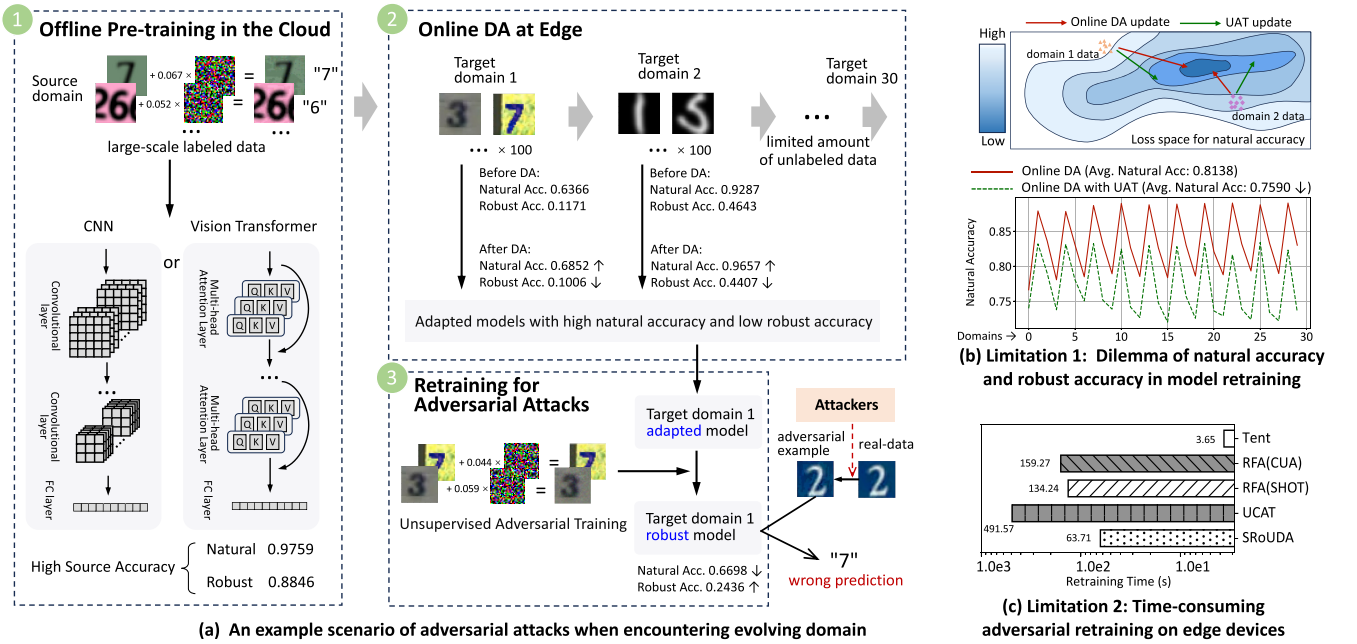

Fig. 1. Example scenario and limitations. Figure 1(a) shows the necessity of performing both online DA and retraining to defend against evolving adversarial attacks for a pre-trained DNN model at edge. However, existing methods have two limitations in addressing this problem. First, they struggle to balance natural and robust accuracy during retraining (Figure 1(b)). Second, adversarial retraining on edge devices is time-consuming (Figure 1(c)).

degradation (e.g., dropping from 81.38% to 75.90%), and vice versa. The **first challenge**, therefore, is *how to design a co-optimization method that simultaneously maintains high natural and robust accuracies in retraining.*

2) Time-Consuming Adversarial Retraining on Edge Devices: A DA often has short retraining window (e.g. 10 minutes) in order to maintain consistently high natural accuracy. However, current UAT methods usually need a large number of iterations to generate high-quality adversarial samples in adversarial retraining, thus incurring much longer training time than the retraining window. For example, Fig. 1 (c) shows that UAT's retraining duration can be 17.45 to 134.68 times that of the online DA method Tent [9]. Hence the **second challenge** is *how to develop an efficient on-device retraining method to quickly enhance model robustness.*

In this paper, we introduce RobustDA, a lightweight DNN retraining approach to co-optimize natural and robust accuracies on edge devices. The key idea of RobustDA is constructing a set of **low-rank adapter** for evolving domains, where each adapter uses compressed model parameters to store a target domain's learned knowledge. In each model retraining, RobustDA constructs an adapter to separate domain-specific parameters and robustness-related parameters to avoid their potential conflicts in model updating. With retrained adapters, our approach can also quickly generate and select high-quality pseudo-labels for adversarial samples and use them to accelerate the model retraining process. In particular, the contributions of this paper are as follows:

- We propose a comprehensive yet cost-effective co-optimization framework that enables non-conflict online DA and adversarial retraining to simultaneously improve natural and robust accuracies.
- We design low-rank adapters that eliminate the need of computationally expensive generation of adversarial samples, yet can dynamically and efficiently adapt to evolving domains to speed up model retraining process.
- We fully implement RobustDA using PyTorch and compare it against 14 state-of-the-art DA techniques under popular adversarial attacks. Experiment results on heterogeneous edge devices show: (i) RobustDA increases robust accuracy by 11.41% while also improving natural accuracy by 6.34%; (ii) with similar accuracies, RobustDA speeds up the retraining time by 4.09x.

The rest of this paper is organized as follows. In Section II, we introduce the background and related work of this work. Section III illustrates the methodology of our proposed RobustDA. Section IV presents the evaluation settings and results. Finally, Section V concludes our work.

II. Background and Related Work

A. Adversarial Robustness

1) Adversarial Attack: Recent research has shown that DNNs can be susceptible to misbehavior when presented with adversarial samples. This possibility was first identified by Szegedy et al. in [5], and it has led to extensive discussion in machine learning field. Adversarial samples are very similar to the original samples and can deceive DNNs. For example, an adversarial sample is a picture that can be classified by human eyes but can mislead the image classification model, resulting in incorrect classification and serious security threats. Adversarial attacks can be categorized into white-box and black-box attacks. In white-box attacks, attackers can access all information about the model, including the internal structure, parameters, training process, inputs, and outputs. In contrast, black-box attacks only allow to access the input

and output interfaces of the targeted model, and they can be further divided into transfer-based attacks and query-based attacks.

2) Adversarial Defense:

a) Definition of adversarial robustness: Consider a standard classification task where samples $x \in \mathbb{R}^d$ and labels y follow the data distribution $\mathbb{D}$. The loss function is $\mathcal{L}(x, y; \theta)$, where θ represents the parameters of model. The typical objective of optimization is to identify the optimal model parameters that minimizes the risk $\mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\mathcal{L}(x, y; \theta)\right]$, which is usually obtained by solving an empirical risk minimization (ERM) problem on training data. Furthermore, the optimization objective of adversarial training turns the issue into a "min-max" problem by incorporating an attack model within empirical risk minimization:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\max_{\delta\in\mathbb{S}} \mathcal{L}(x+\delta, y; \theta)\right]. \tag{1}$$

For each set of sample x, a set of perturbations $\delta \in \mathbb{S} \subseteq \mathbb{R}^d$ is added on them, thus generating adversarial examples. The maximum range of the perturbations is limited by $\mathbb{S}$. The generated samples x' appear visually identical to the original clean samples to the human eyes. This is usually measured using L_p-normalization as:

$$\mathbb{S} = \{\delta \in \mathbb{R}^d \mid \|\delta\|_p \leq \epsilon\}, \epsilon > 0, \tag{2}$$

where ϵ is the perturbation budget that controls the adversarial robustness of the model. Under the same normalization method, the larger value ϵ indicates a stronger perturbation.

B. Related Work

Online DA techniques adapt DNN models to a target domain and they can be divided into two categories: (i) BatchNorm (BN) tuning techniques adapt the BN layer in the DNN model to the target domain. Some studies have argued that updating only the normalized statistics of the BN layer [10], while other approaches have focused on training the linear transformation parameters of the BN layer [9]. (ii) Feature alignment techniques align features (e.g., feature maps or feature distributions) between the source and target domains by training the feature extractors of the target domain model, aiming to mitigate the domain bias. These techniques use distance metrics to calculate and minimize the distance of feature distributions (e.g. Maximum Mean Discrepancy (MMD) [11], correlation alignment [14], and entropy [3]), or use adversarial network techniques [15] to achieve feature alignment by adversarially training the target domain's feature extractor and discriminator. Continual DA [16], [17] and Test-Time Adaptation (TTA) [18], [19] are closely related to these areas. Continual DA focuses on optimizing model performance across both the current and all previously seen domains. It involves multiple training iterations followed by evaluation. In contrast, TTA focuses on improving model accuracy and accelerating its training for the current domain.

1) Adversarial Attack: Fast Gradient Sign Method (FGSM) [20] is proposed to find adversarial samples efficiently. BIM [21] attempts to extend the FGSM by taking multiple iterations of small gradient steps. Projected Gradient Descent (PGD) [6] extends the FGSM from a single iteration to a multiple iteration version. AutoAttack [22] proposes a hybrid attack method that runs four attacks sequentially, which automatically adjusts step size. Other studies such as Square [23] and FAB [24] focus on the vulnerability to attacks of deep neural networks.

2) Adversarial Defense: Previous research has proposed several effective defenses against adversarial attacks. Adversarial Training (AT) [6] is currently the most representative and robust defense. TRADES [25] proposes a compromise adversarial defense method which measures model accuracy and robustness with two separate loss terms derived theoretically. Inspired by the seminal work of TRADES, several unsupervised adversarial training methods like Label-Free Adversarial Training (LFAT) [12] and Pseudo-Label Adversarial Training (PLAT) [13] are proposed for conducting adversarial training on unlabeled samples. However, these techniques are designed to a pre-specified domain, and their robustness may significantly decrease when domain shift happens.

3) Online DA and Adversarial Robustness: At present, some preliminary work has been proposed to study both domain generalizability and robustness [26]. It explores the relationship between adversarial robustness and transfer learning to deal with Out of Distribution(OOD) data in the target domain, and finds that a model's robustness accurcy mainly depends on the extracted robust features in training. Existing methods can be divided into two types: (i) the *transfer-based* methods are inspired by the exploration of robust transfer in [26]. For instance, RFA [27] uses an external pre-trained robust model for robust feature distillation during the DA process. Although RFA is effective, its performance is limited by the perturbation budget of the teacher model and is sensitive to the architecture of the teacher model. (ii) The *self training-based* methods often require the generation of pseudo-labels in the target domain for adversarial training. Specifically, UCAT [28] applies adversarial training in unlabeled target domain by training a model on the source domain. ASSUDA [13] uses a pre-trained UDA model in the semantic segmentation scenario to generate pseudo-labels. SRoUDA [29] empoys a meta-learning technique to mitigate the error propagation of noisy pseudo-labels. When applying in edge scenarios, existing methods has two limitations. First, they are designed for one given target domain, rather than continuously evolving domains at edge. Second, their training needs source domain data, large numbers of iterations and other extra information, thus may incur high computation costs.

III. Method

A. Overview

We design RobustDA to enable adaptations to evolving domains on edge devices while maintaining a high level of adversarial robustness in each domain. The key idea of RobustDA is the low-rank adapter, termed LoRA for short, which jointly optimizes natural accuracy and adversarial robustness in the target domain. This adapter enables quick retraining and supports lightweight computation on devices

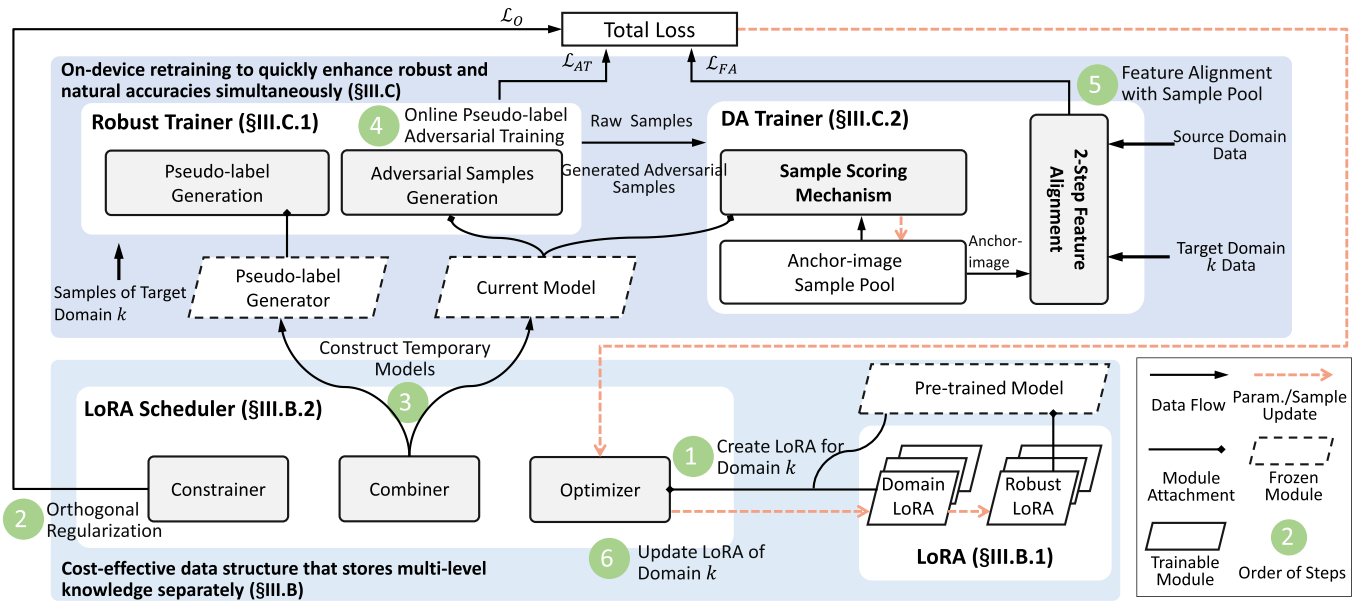

Fig. 2. The overview of RobustDA, which shows how its internal components work when encountering a new target domain k.

with limited resources. In addition, RobustDA can leverage the retained adapters to efficiently generate high-quality adversarial samples for improving model robustness. As illustrate in Figure 2, our approach consists of four modules and is designed to achieve two objectives.

1) Cost-Effective Data Structure That Stores Multi-Level Knowledge Separately (§III.B): For evolving domains, RobustDA designs a set of LoRAs that utilize compact model parameters to store knowledge for each learned target domain. Specifically, it separates domain-specific and robustness-related parameters into distinct LoRAs, thus preventing the dilemma of robust accuracy and natural accuracy in model training. RobustDA is also able to merge all learned LoRA modules with the original model to avoid inference latency increase.

- *LoRA* (§III.B.1). RobustDA attaches a set of LoRAs to the original DNN model. During retraining, it only updates the parameters of these adapters, while storing the learned domain-specific knowledge and robustness-related knowledge in separate adapters.
- *LoRA Scheduler* (§III.B.2). The scheduler consists of three components: a constrainer, a combiner, and an optimizer. The *constrainer* provides constraints based on the parameters between LoRA modules to mitigate forgetting caused by mutual coverage of these modules' knowledge. The *combiner* combines the original model with existing LoRA modules to generate temporary models for specific purposes, such as a Pseudo-label Generator for generating pseudo-labels and the model of current domain for generating adversarial samples. The *optimizer* updates the corresponding LoRA parameters with the online retraining results.

2) On-Device Retraining to Quickly Enhance Robust and Natural Accuracies Simultaneously (§III.C): Generating high-quality (low-noise) pseudo-labels is a time-consuming process in traditional approaches. RobustDA aims to quickly generate adversarial samples by leveraging the stored knowledge of all adapters. It also designs a sample pool to store a small amount of filtered target domain data and their pseudo-labels, and uses the pool for feature alignment between target and source domains.

- *Robust trainer* (§III.C.1). This trainer has two stages: the first stage generates pseudo-labels and filters training data via a confidence threshold; and the second stage generates adversarial samples using the filtered data and pseudo-labels. These samples are then used in adversarial training.
- *DA trainer* (§III.C.2). RobustDA designed a sample scoring mechanism that utilizes the sample robustness information from the Robust Trainer to evaluate the importance of new samples and stored samples in the pool. More important samples indicate they are more difficult to be attacked. In other words, they are more helpful for feature alignment. RobustDA updates the sample pool at each iteration by adding the more important samples. RobustDA also designs a two-step feature alignment method based on the sample pool to avoid conflicts between natural accuracy and robust accuracy.

Overall, RobustDA operates at both offline pre-training and online retraining. At the offline stage, RobustDA creates a robust and accurate model by first training with source domain samples, then adding and training a Domain LoRA module with adversarial samples while freezing the former model's parameters. At the online stage, RobustDA adapts this pre-trained model to evolving target domains using *6 steps*. Given a new target domain, step 1 initializes a new LoRA module, freezes the pre-trained model and existing LoRA modules. The following four steps calculate and optimize three loss functions: (i) orthogonality regularization loss ($\mathcal{L}_{\mathcal{O}}$) to prevent interference with previously learned knowledge

(step 2); (ii) adversarial training loss ($\mathcal{L}_{AT}$) to enhance robustness in the current domain (steps 3 and 4); and (iii) feature alignment loss ($\mathcal{L}_{FA}$) to align source and target domain features (step 5). Finally, step 6 only updates the current domain's LoRAs to simultaneously improve robustness and accuracy.

B. Cost-Effective Data Structure That Stores Multi-Level Knowledge Separately

1) Design of LoRA: Developed from the LoRA technique [30] originally proposed for fine tuning large language model, RobustDA designs its *robust LoRA structure* with four purposes. (i) In the online training stage, the LoRA architecture is unique in its optimization of low-rank small parameters. This significantly reduces memory usage, enabling efficient model training even on resource-constrained edge devices. (ii) Compared to other DA methods, LoRA doesn't increase inference latency in the inference stage. This means that in edge environments, LoRA can provide faster and responsive inference results, meeting the requirements of high real-time applications. (iii) The flexibility of LoRA is evident in the ability to freely adjust the weights of each module. If each module represents different domains or tasks, adjusting the weights allows for flexible combinations of models with different functionalities, thus effectively satisfying DA or robustness requirements at the edge. (iv) LoRA's universality allows it to support different model architectures like CNNs and Transformers, catering to a wide range of complex applications.

Formally, for a pre-trained weight matrix W_0 of size $d \times h$, the updated weight matrix W after fine-tuning can be expressed as $W = W_0 + \Delta W$, where ΔW represents the difference between W_0 and W. Decomposing ΔW can significantly reduce the number of parameters. Thus, it can be decomposed into $\Delta W = BA$, where B and A being the new small weight matrices of size $d \times r$ and $r \times h$ respectively. Here, r represents the LoRA dimension, i.e., rank, with $r \ll d, h$.

We integrate LoRA into DA to enhance robustness without significantly increasing model size. During the offline pretraining stage, we designed **Robust LoRA** and applied them to enhance robustness in the source domain and ensure robustness against domain changes in future edge deployments. This approach significantly improves robustness without significantly increasing model size. Specifically, instead of directly employing adversarial training to obtain robust source domain models using adversarial samples, we choose to pre-train with natural samples. Subsequently, we add LoRAs (with similar structures) to each layer of the feature extractor parameters and freeze all other parameters, fine-tuning only the newly added LoRA part with adversarial samples (i.e., adversarial training). By adding LoRA modules, the model size increased slightly. Note that it doesnot improve the actual inference latency and computational cost, as the parameters are merged during inference. Similarly, during current target domain retraining, each time a new target domain is encountered, new adapters are added to preserve the learned domain-specific parameters. These adapters are referred to as **Domain LoRA**.

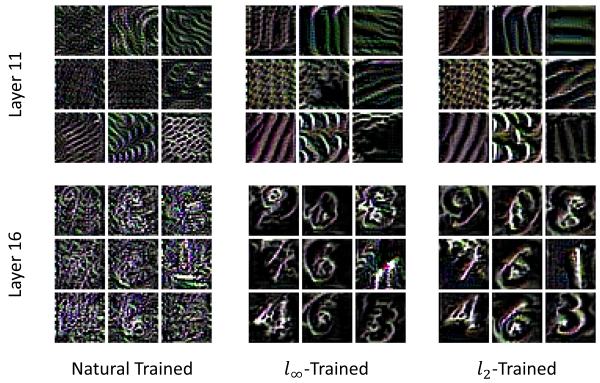

Fig. 3. Visualization results of the features extracted by partial convolutional kernels at the 11th and 16th layers of the three class of WideResNet-16-2 models [7]. It can be seen that the features extracted by models trained on natural samples (both low-level and high-level features) are mixed with high-frequency noised features. However, after incorporating the trained LoRA modules, these noisy features are purified.

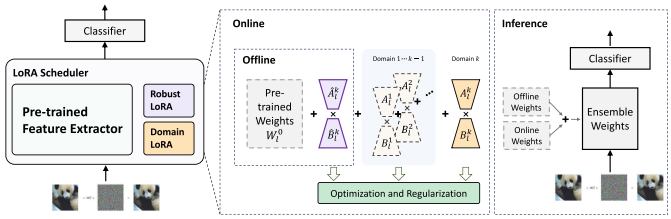

Fig. 4. The schematic diagram illustrates our LoRA Scheduler. It visualizes the training process of the LoRA modules attached to a layer's parameters in the feature extractor during both offline and online stages. When deploying inference, the parameters trained in both stages are merged seamlessly without incurring additional inference latency.

With the above mechanism, we can achieve both natural accuracy and robustness at a very low overhead. In addition, high-quality natural features and robust features can be obtained simultaneously in the retraining stage. This aids task-specific feature alignment or fine-tuning in the subsequent online stage. Moreover, as shown in Fig. 3, colorblackRobust LoRA has demonstrated the capability of cleaning noisy features and thus maintaining model robustness when migrating to a new target domain.

2) Design of LoRA Scheduler: The LoRA Scheduler can manage various LoRA modules with different functionalities on edge devices, providing an interface for online retraining and interaction between models. Fig. 4 illustrates the structure and functionality of the LoRA Scheduler. The specific design is as follows.

Formally, we deploy the feature extractor, classifier, and Robust LoRA trained in the offline stage to the target domain. To enable the model to adapt to diverse target domains flexibly and improve its generalization capability, for each new target domain $\mathcal{D}^k$, we initialize a new set of Domain LoRAs on every layer parameters of the feature extractor. These new modules, denoted as $\{A^k, B^k\} = \bigcup_{l=1}^{L}\{A_l^k, B_l^k\}$, where L is the total number of layers with optimizable parameters in the feature extractor, are concatenated with the existing LoRA modules. Correspondingly, the parameters of Robust LoRA are denoted as $\{\hat{A}^k, \hat{B}^k\} = \bigcup_{l=1}^{L}\{\hat{A}_l^k, \hat{B}_l^k\}$, representing the Robust LoRA retrained in the k-th domain.

For each domain, our optimization strategy focuses on optimizing only the parameters of the additional new modules, i.e., Domain LoRA $\{A^k, B^k\}$, and the inherited parameters of the Robust LoRA module from the previous domain $\{\hat{A}^k := \hat{A}^{k-1}, \hat{B}^k := \hat{B}^{k-1}\}$. The remaining parameters are frozen and not involved in online training. The purpose of adding and optimizing Domain LoRA is to enable the model to learn the ability to confuse domains, thereby achieving feature alignment between the target domain and the source domain. However, in our scenario where there are continuously incoming target domains, for this incremental learning, we need to set a constraint to ensure that the model does not forget the features of old domains while learning new domain features and accumulating knowledge. This ensures that the model performs well on all seen target domains by retaining and incorporating knowledge over time.

To achieve this goal, we introduce a constraint in the loss function that ensures the optimization direction of the parameters for the current target domain is orthogonal to the optimization directions of all past domains. In other words, we optimize the model towards a more integrated direction rather than overlapping directions, which helps prevent changes in the loss of previous tasks when learning parameters for new tasks. This ensures that the knowledge between different target domains remains mutually independent.

Since each of our Domain LoRA modules represents parameter updates for a target domain, we can define orthogonality between two LoRA modules as follows:

$$\mathcal{O}\left(A_{l_1}^j, A_{l_2}^k\right) = \left\| {A_{l_1}^j}^T A_{l_2}^k \right\|^1, \tag{3}$$

where l_1 and l_2 represent a certain layer of the DNN, $\cdot^T$ denotes transpose, and $\|\cdot\|^1$ represents the L_1-norm. We use the L_1-norm instead of the L_2-norm because the L_1-norm better represents the sparsity of tensors, which is beneficial for optimizing towards mutually orthogonal objectives. We only use parameter A here because A is randomly initialized, while B is initialized to zeros. Therefore, when $\mathcal{O}$ approaches 0, we can consider the two modules as approaching orthogonality. Specifically, for the target domain $\mathcal{D}^k$, the complete constraint is formulated as follows:

$$\mathcal{L}_O = \mathop{\mathbb{E}}_{\substack{1\le i\le k \\ 1\le l_1,l_2\le L}} \mathbb{1}_{[i\neq k, l_1=l_2]}\mathcal{O}\left(A_{l_1}^j, A_{l_2}^k\right) + \mathbb{1}_{[l_1=l_2]}\mathcal{O}\left(\hat{A}_{l_1}^k, A_{l_2}^k\right), \tag{4}$$

where L denotes the total number of layers with optimizable parameters in the feature extractor. $\mathbb{1}_{[]}$ represents indicator function. Additionally, to prevent the sparsity constraint from affecting Robust LoRA and causing it to forget relevant robust knowledge, we need to ensure that the $\hat{A}_{l_1}^{k-1}$ involved in the calculation of the above equation are copies of the inherited Robust LoRA from the previous target domain, rather than the actual parameters. This prevents the backward propagation of gradients.

Note that when target domains are uncorrelated, the optimization of Equation 4 updates each Domain LoRA's parameters in an orthogonal direction, ensuring that each Domain LoRA retains only the domain-specific knowledge. On othe other hand, when target domains are highly correlated, RobustDA can merge LoRAs of all similar domains to preserve domain-specific knowledge and reduce computation and storage costs.

C. On-Device Retraining to Quickly Enhance Robust and Natural Accuracies Simultaneously

1) Design of Robust Trainer: To conduct adversarial training during the online stage, RobustDA designs a pseudo-label generator to assign pseudo-labels to a small amount of unlabeled data in the target domain. One advantage of our approach is the ability to freely set the weight coefficients of each LoRA module during inference. Specifically, the total parameters of the model for the l-th layer should be:

$$W_l := W_l^0 + \alpha_l \hat{B}_l^k \hat{A}_l^k + \sum_{j=1}^{k} \beta_l^j B_l^j A_l^j, \tag{5}$$

where α represents the weight coefficient of Robust LoRA, and β^j represents the weight coefficient of the Domain LoRA corresponding to each target domain j.

Since in the optimization strategy, we have a set of LoRA modules responsible for robustness, while the remaining sets of LoRA modules focus on optimizing the corresponding target domains (aligning the features of the target domain with the source domain). To eliminate interference caused by the inherent trade-off between robustness and natural accuracy [25] and to minimize noise in pseudo-labels, we set α to 0. For the remaining coefficients, we employ an optimization strategy of entropy minimization. We solve for a combination of Domain LoRA according to the following constraints.

$$\begin{aligned} &\min_{\beta} -\sum_c p_c\left(x_i;\beta\right)\log p_c\left(x_i;\beta\right) \\ &\text{s.t.}\quad p_c\left(x_i;\beta\right) = T\left(F\left(x_i;W(\beta)\right),c\right) \\ &\qquad W(\beta) = \left\{W_l^0 + \sum_{j=1}^{k}\beta_l^j B_l^j A_l^j\right\}_{l=1}^{L} \end{aligned} \tag{6}$$

The solved model combination scheme is handed over to the Combiner to assemble a temporary model, serving as the pseudo-label generator. This approach quickly combines a more confident pseudo-label generator using the knowledge stored in lightweight data structures, eliminating the need for costly maintenance of additional models. Specifically, the process of generating pseudo-labels is as follows: We combine the parameters $W_P := W^0 + \sum_{j=1}^{k}\beta^j B^j A^j$, and use this parameter group to initialize a new feature extractor F_P. This new feature extractor is then feed through a classifier T pretrained on the source domain to obtain the probability function $p(x_i) = T(F_P(x_i))$. Pseudo-labels are obtained based on the following equation:

$$\hat{\mathcal{T}}_\gamma^k = \left\{\left\langle x_i, \hat{y}_i\right\rangle \mid \left\langle x_i, \hat{y}_i\right\rangle \in \hat{\mathcal{T}}^k, \max(p(x_i)) \ge \gamma\right\}, \tag{7}$$

where γ represents the pseudo-label threshold, indicating that only samples with class probability scores greater than or equal to the threshold are selected for inclusion in the pseudo-label

training set. $\hat{\mathcal{T}}^k$ denotes the set of sample pairs where the predicted class with the highest probability is used as the pseudo-label, i.e., $\hat{\mathcal{T}}^k = \bigcup_{c=1}^{C} \left\{ \langle x_i, c \rangle \mid \underset{\tilde{c} \in \{1,\ldots,C\}}{\arg\max}\, p_{\tilde{c}}(x_i) = c \right\}$, where C is the number of classes. It's evident that $\hat{\mathcal{T}}_\gamma^k \subseteq \hat{\mathcal{T}}^k$.

The samples in the pseudo-label training set are used for adversarial training, with the loss function given by:

$$\begin{aligned} \mathcal{L}_{AT} = & \mathop{\mathbb{E}}_{(x_i, \hat{y}_i) \in \hat{\mathcal{T}}_\gamma^k} \ell_{CE}(T(F_k(x_i')), \hat{y}_i) \\ \text{s.t.} \quad & \|\delta\|_\infty \leq \epsilon \\ & x_i' = \arg\max_{\delta} \ \ell_{CE}(T(F_k(x_i + \delta)), \hat{y}_i) \end{aligned} \tag{8}$$

where F_k denotes the feature extractor, initialized with parameters $W_k := W^0 + \hat{B}^k \hat{A}^k + \sum_{j=1}^{k} B^j A^j$. x_i' represents the adversarial sample. Through adversarial training, we update the Robust LoRA and the current Domain LoRA.

Algorithm 1 Online Adversarial Training of Robust Trainer

Input: D^k: k-th target domain data, x: a batch of samples from D^k, W^0: weights of the feature extractor in source domain, T: classifier of source domain model, γ: threshold of pseudo-label, n_iters: iterations for adversarial training, $step$: steps for adversarial training

Output: robustness loss $\mathcal{L}_{AT}$

1: **while** β does not satisfy the constraint 6 **do**
2: $W_p \leftarrow W^0 + \sum_{j=1}^{k} \beta^j B^j A^j$
3: $F_p \leftarrow$ Initialize the feature extractor using W_p
4: $p \leftarrow$ Use x to forward propagation through the F_p and T
5: $\beta \leftarrow \min_\beta - \sum_c p_c \log p_c$
6: **end while**
7: $\hat{\mathcal{T}}_\gamma^k \leftarrow \emptyset$
8: **for** $x_i \in x$ **do**
9: $p_i \leftarrow$ Use x_i to forward propagation through the F_p and T
10: **if** $max(p_i) \geq \gamma$ **then**
11: $\hat{y}_i \leftarrow \underset{\tilde{c} \in \{1,\ldots,C\}}{\arg\max}\, p_{\tilde{c}}(x_i) = c$
12: Add $\langle x_i, \hat{y}_i \rangle$ to $\hat{\mathcal{T}}_\gamma^k$
13: **end if**
14: **end for**
15: $W_k \leftarrow W^0 + \hat{B}^k \hat{A}^k + \sum_{j=1}^{k} B^j A^j$
16: $F_k \leftarrow$ Initialize the feature extractor using W_k
17: $\mathcal{L}_{AT} \leftarrow 0$
18: **for** $\langle x_i, \hat{y}_i \rangle$ in $\hat{\mathcal{T}}_\gamma^k$ **do**
19: $x_i' \leftarrow x_i$
20: **for** each n_iters **do**
21: $p_i \leftarrow$ Use x_i to forward propagation through the F_k and T
22: $x_i' \leftarrow Clip_{x_i', \epsilon}(x_i' + step \cdot sign(\nabla_{x_i'} \ell_{CE}(p_i, \hat{y}_i)))$
23: **end for**
24: $p_i \leftarrow$ Use x_i' to forward propagation through the F_k and T
25: $\mathcal{L}_{AT} \leftarrow \mathcal{L}_{AT} + \ell_{CE}(p_i, \hat{y}_i)$
26: **end for**

Output: $\mathcal{L}_{AT}$

Algorithm 1 presents the process of online adversarial training of Robust Trainer. First, it iteratively optimizes the parameters β to find the optimal values that enables composing all LoRA modules and the original model into a high-quality pseudo-label generator (lines 1-6). Next, with the generator (F_p with T), the algorithm generates pseudo-label for each sample x_i and store it into $\hat{\mathcal{T}}_\gamma^k$ (lines 8-14). Then the algorithm construct feature extractor F_k for generating adversarial samples (lines 15-16). The generation of adversarial samples is calculating cross-entropy loss on the pseudo-label set, applies gradient ascent to update the adversarial sample x_i' (lines 20-24). Finally, the algorithm return the robust loss $\mathcal{L}_{AT}$ by using these adverarial samples (lines 25-26).

2) Design of DA Trainer: We design DA Trainer to effectively optimize the natural accuracy through the intermediate information generated by Robust Trainer. To this end, we designed an sample pool and update it using a *sample scoring mechanism*. With the sample pool, we then proposed a *two-step feature alignment* method to optimize the natural accuracy.

a) Sample scoring mechanism: Although the model is susceptible to attacks, not all samples can be used to attack the model. The principle of adversarial attacks is to find an adversarial sample in the neighborhood $\mathbb{S}$ of the sample that maximizes the Loss of the current model. If the attack succeeds in causing the model to produce an incorrect classification result, it indicates that the adversarial sample has crossed the classifier's decision boundary in the feature space.

The optimization goal of DA, in essence, is to narrow the intra-class distance while widening the inter-class distance. This means that samples of the same class should move closer to the centroid of that class in the feature space, while samples of different classes should move away from each other.

Since our model is robust on the target domain, we can fully leverage the advantages of the robust model to assist DA. Suppose a sample x_i and its pseudo-label $\hat{y}_i$ have a classification loss ℓ_i^{nat} under the current model. When transformed into an adversarial sample, the classification loss is re-calculated as ℓ_i^{adv}. We can then intuitively establish the importance score of the sample under the current model as follows:

$$\begin{aligned} \mathcal{S}\left(\langle x_i, \hat{y}_i \rangle\right) &= \ell_i^{adv} - \ell_i^{nat} \\ &= \ell_{CE}\left(T\left(F\left(x_i'\right)\right), \hat{y}_i\right) - \ell_{CE}\left(T\left(F\left(x_i\right)\right), \hat{y}_i\right) \\ &= \log \frac{p_{\hat{y}_i}\left(x_i'\right)}{p_{\hat{y}_i}\left(x_i\right)}. \end{aligned} \tag{9}$$

where $\mathcal{S}$ is a function used to score the samples. The smaller the score, the less likely the image x_i is to be disturbed in the feature space of the current model, indicating that its features are close to the class centroid. This prevents them from approaching the decision boundary and reduces susceptibility to attacks. This approach ensures that DA does not compromise robustness. Just as its function suggests, we can refer to this type of image sample as an *anchor*.

During online training, we construct the image sample pool $\mathbb{M} = \left\{\langle x_i, \hat{y}_i \rangle\right\}_{i=1}^{N}$, where N is the current number of samples in the sample pool. We dynamically maintain the sample pool on the target domain in real-time. During each iteration,

we calculate the scores for a batch of data $\mathbb{B}$ along with the data in the original sample pool. When the total number of samples exceeds the sample pool capacity N_{max}, we select the top N_{max} samples based on their scores and add them to the pool. Therefore, the updated sample pool is:

$$\mathbb{M}^* = \arg\min_{\mathbb{M}} \sum_{\langle x_i, \hat{y}_i \rangle \in \{\mathbb{M}, \mathbb{B}\}} \mathcal{S}(\langle x_i, \hat{y}_i \rangle). \tag{10}$$

In order to ensure that the sample pool maintains high quality and optimizes the effect of feature alignment, we have also considered adding two mechanisms to it.

i) Class balancing mechanism: Since samples from the same class typically tends to cluster together in feature space, the existing scoring system favors the classes that naturally distant from other class centroids. This could lower the scores of these classes, causing class imbalance. Therefore, we further incorporate a penalty term $\mathcal{B}(\hat{y}_i)$ into the scoring, which increases the importance score of samples from over-represented classes while decreasing the score of samples from under-represented classes. The scoring mechanism with class balancing is as follows:

$$\begin{aligned} s_i &= \mathcal{S}\left(\langle x_i, \hat{y}_i \rangle\right) - \eta \mathcal{B}\left(\hat{y}_i\right) \\ &= \log \frac{p_{\hat{y}_i}\left(x_i'\right)}{p_{\hat{y}_i}\left(x_i\right)} - \eta \frac{\sum_{\langle x_j, \hat{y}_j \rangle \in \mathbb{M}} \mathbb{1}[\hat{y}_j = \hat{y}_i]}{N}, \end{aligned} \tag{11}$$

where η is the penalty coefficient. For samples from the same class, the penalty term $\mathcal{B}(\hat{y}_i)$ we added is the same.

ii) Smooth updating mechanism: To prevent large-scale oscillations in the samples entering the pool, which could lead to frequent changes in the optimization direction of feature alignment, we introduce a smoothing update mechanism for sample scoring. That is, at time $t+1$, the score of the sample x_i is s_i^{t+1}, and after smoothing, the final importance score s^{*t+1}_i for sample x_i is given by:

$$s_i^{*t+1} = \begin{cases} \mu s_i^t + (1-\mu) s_i^{t+1}, & \langle x_i, \hat{y}_i \rangle \in \mathbb{M}^t \\ s_i^{t+1}, & \langle x_i, \hat{y}_i \rangle \notin \mathbb{M}^t \end{cases}, \tag{12}$$

where μ is the smoothing coefficient, and taking $\mu = 0$ is equivalent to removing smoothing. In other words, if a sample is already in the sample pool at time t, its score at time $t+1$ is the weighted average of the old score and the new score.

Algorithm 2 introduces the process of updating the sample pool. First, it construct a feature extractor model F_k for subsequent scoring. (lines 1-2). Then score a sample based on F_k and store the sample with its score into a dictionary $score\_dict$ (lines 4-9). Subsequently, the algorithm updates the sample scores based on Equation 11 and 12 (lines 10-16). Finally, the algorithm sorts the dictionary $score\_dict$ by the samples' scores, and selects the top $N_{\max}$ samples with the lowest scores to update the sample pool M_{t+1} (lines 17 and 18).

b) Two-step feature alignment: The existing means of feature alignment typically involve aligning all samples. However, samples of the same class may still show variation in feature distributions, causing feature alignment instability across batches. In contrast, the *anchor* images in the sample pool consists of filtered anchor samples, and feature alignment based on them is of high quality.

Algorithm 2 Maintaining of *Anchors* in Sample Pool

Input: D^k: k-th target domain data, $\mathbb{M}$: sample pool at time t, $\langle x, \hat{y} \rangle$: a batch of sample features x with their pseudo-labels $\hat{y}$ at time $t+1$, W^0: weights of the feature extractor in the source domain, T: classifier of source domain model, $\{\hat{A}, \hat{B}\}$: Robust LoRA parameters, $\{A, B\}$: Domain LoRA parameters, N_{max}: capacity of sample pool, η, μ

Output: sample pool $\mathbb{M}^{t+1}$ at time $t+1$

1: $W_k \leftarrow W^0 + \hat{B}^k \hat{A}^k + \sum_{j=1}^{k} B^j A^j$
2: $F_k \leftarrow$ Initialize the feature extractor using W_k
3: $score\_dict \leftarrow \{\}$ ▷ A dictionary to store samples and their scores.
4: **for** $\langle x_i, \hat{y}_i \rangle \in (\langle x, \hat{y} \rangle \cup \mathbb{M}^t)$ **do**
5: $p_i \leftarrow$ Use x_i to forward propagation through the F_k and T
6: $x_i' \leftarrow$ adversarial sample of x_i
7: $p_i' \leftarrow$ Use the adversarial sample x_i' to forward propagation through the F_k and T
8: $score\_dict[\langle x_i, \hat{y}_i \rangle] \leftarrow \log \frac{p_{\hat{y}_i}(x_i')}{p_{\hat{y}_i}(x_i)}$
9: **end for**
10: **for** $\langle x_i, \hat{y}_i \rangle \in (\langle x, \hat{y} \rangle \cup \mathbb{M}^t)$ **do**
11: $\mathcal{B}(\hat{y}_i) \leftarrow$ the number of samples whose labels are $\hat{y}_i$ within the $\mathbb{M}^t$
12: $s_i \leftarrow score\_dict(\langle x_i, \hat{y}_i \rangle) - \eta \mathcal{B}(\hat{y}_i)$
13: **if** $\langle x_i, \hat{y}_i \rangle$ in $\mathbb{M}^t$ **then**
14: $s_i \leftarrow \mu \mathbb{M}^t[\langle x_i, \hat{y}_i \rangle] + (1-\mu) s_i$
15: **end if**
$score\_dict[\langle x_i, \hat{y}_i \rangle] \leftarrow s_i$
16: **end for**
17: $score\_dict \leftarrow Sort(score\_dict)$ ▷ Sort the $score\_dict$ in ascending order based on the sample score
18: $\mathbb{M}^{t+1} \leftarrow$ Update sample pool $\mathbb{M}$ at time $t+1$ using top N_{max} elements in $score\_dict$

Output: $\mathbb{M}^{t+1}$

On one hand, we align the features of anchor images with those of source domain images. On the other hand, we align the features of all target domain samples with the features of anchor images in the sample pool. The loss function for our feature alignment is given by:

$$\mathcal{L}_{FA} = \mathop{\mathbb{E}}_{\substack{x_i \in \mathcal{S}, x_j \in \mathcal{D}^k \\ x_m \in \mathbb{M}}} \mathcal{D}\left(F(x_m), F_0(x_i)\right) + \mathcal{D}\left(F(x_m), F(x_j)\right), \tag{13}$$

where $\mathcal{D}$ represents the distance metric function, and F_0 is the feature extractor of source domain. We use Maximum Mean Discrepancy (MMD) to measure the difference between distributions.

Overall, the joint online retraining loss is defined as

$$\mathcal{L} = \mathcal{L}_{AT} + \lambda_O \mathcal{L}_O + \lambda_{FA} \mathcal{L}_{FA}, \tag{14}$$

where λ_O and λ_{FA} are positive weight coefficients.

D. Running Example

Fig. 5 shows the online retraining process of RobustDA when encountering the 3rd target domain. In this example,

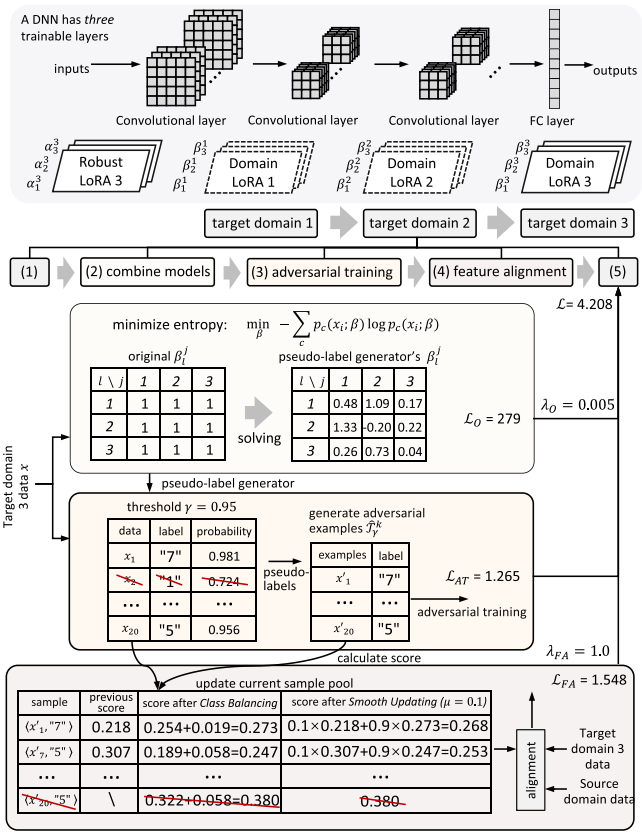

Fig. 5. Example process of online retraining with RobustDA. It demonstrates how to compute the three loss functions in RobustDA to improve both natural and robust accuracies in DA.

we use a simple three-layer DNN and select one iteration to demonstrate how RobustDA works.

Following the steps outlined in Fig. 2. We first attach a new Domain LoRA to the 3rd target domain and inherit the Robust LoRA from the previous domain. Next, we use entropy minimization as the optimization objective to solve for the weight coefficients β of all Domain LoRAs using target domain data.Then combine these modules into a pseudo-label generator. Additionally, we simultaneously compute the orthogonal regularization term as $\mathcal{L}_O = 279$. Then, we generate pseudo-labels for the corresponding target domain samples. In this example, the pseudo-label threshold γ is set to 0.95, resulting in the removal of noise from the sample x_2 with a probability of 0.724. Subsequently, we use these pseudo-labels to generate adversarial samples and conduct on-device adversarial training, resulting in $\mathcal{L}_{AT} = 1.265$. Afterward, we calculate the importance scores utilizing the adversarial samples and their corresponding original samples and update the sample pool using the class balancing and smooth updating mechanisms. Based on a small amount of data in the sample pool, a small amount of source domain data, and the current target domain data, we perform feature alignment, resulting in $\mathcal{L}_{FA} = 1.548$.

Finally, we multiply the three loss functions by their corresponding weight coefficients to obtain the joint online retraining loss $\mathcal{L} = 4.208$. We then update the parameters of the corresponding LoRA module using the gradients obtained from backpropagation, completing one iteration.

TABLE I
SUMMARY OF TWO WORKLOADS IN EVALUATION

Task	Source Domain	Target Domain	Acc. metric
Digit-Five (low-resolution image classification)	MNIST Synthetic Digits	MNIST-M USPS SVHN	top-1 acc.
Object-Sign (high-resolution image classification)	ImageNet SYN SIGNS	Caltech256 GTSRB DomainNet	top-1 acc.

IV. EVALUATION

In this section, we evaluate the full implementation of RobustDA on top of PyTorch with an extensive set of evaluations.

A. Experimental Settings

1) Testbed: We choose multiple heterogeneous edge platforms. For high-resource devices, we selected an NVIDIA AGX Xavier with a 512-core NVIDIA Volta GPU and 32GB of memory. For low-resource devices, we chose a Jetson Nano featuring NVIDIA Maxwell architecture with 128 NVIDIA CUDA cores and 4 GB of memory. All NVIDIA platforms run Ubuntu 18.04.5 LTS and support DNNs in PyTorch 1.9.0 (Python 3.6.9 and CUDA 10.2).

2) Datasets: Following the setting of EdgeVisionBench [1], we use multiple datasets for the evaluation as source and target domains. The specific dataset settings are detailed in Table I. For low-resolution image classification workload, we adopt the Digit-Five task in [31], which includes five independent digit recognition datasets: MNIST, MNIST-M, Synthetic Digits, SVHN, and USPS. Each dataset contains 10 classes ranging from "0" to "9". MNIST and USPS are grayscale datasets, while the others are color datasets. To ensure consistency in input shape, all images from these datasets are resized to 32 × 32 pixels. Additionally, 2 channels are added to MNIST and USPS to unify the image shape across all datasets to 32 × 32 × 3. For high-resolution image classification workload, we employ the Object-Sign task in [32], which includes five independent datasets: ImageNet, SYN SIGNS, Caltech256, GTSRB, and DomainNet. SYN SIGNS and GTSRB are sign recognition datasets, while the others are object recognition datasets. The combined task encompasses 59 classes. All images from these datasets are resized to 224 × 224 pixels.

3) DA Setting: The evaluation tests image classification tasks using both low resolution and high resolution (shown in Table I). Each task uses five corresponding datasets. The first two datasets are combined to form the source domain, while the remaining three datasets serve are used to construct target domains. Specifically, each dataset is randomly sampled to generate 10 target domains and three datasets are alternatively run to generate a total of 30 target domains in an evaluation.

4) Robustness Setting: We consider two types of threat model [33] for attack: white-box and black-box, both using L_∞ norm to measure the size of the adversarial perturbation. We mainly focus on the white-box attacks, while additional experiments (shown in Tables II and III) also explore the

TABLE II
THE AVERAGE ROBUST ACCURACY OF VGG-11 IN THE DIGIT-FIVE SCENARIO ACROSS 30 DOMAINS

Method	APGD	TPGD	FGSM	BIM	FAB	Square	Avg.
Source-only	30.61	58.05	39.87	31.11	32.46	33.99	37.68
CUA	27.43	58.10	40.98	28.29	30.08	31.14	36.00
SHOT	26.30	58.03	42.33	27.35	30.34	31.68	36.00
Tent	30.10	59.36	43.07	30.86	32.73	33.94	38.34
CUA&LFAT	36.77	59.20	43.77	37.20	37.89	38.93	42.29
CUA&PLAT	31.79	58.75	42.35	32.51	34.14	35.06	39.10
SHOT&LFAT	36.69	61.36	46.04	37.26	38.77	40.00	43.35
SHOT&PLAT	35.59	61.67	46.33	36.23	38.09	39.24	42.85
Tent&LFAT	34.80	61.47	45.46	35.57	37.28	38.28	42.14
Tent&PLAT	31.80	60.01	44.40	32.70	34.55	35.71	39.86
RFA_{CUA}	35.65	60.57	42.74	36.08	36.44	37.98	41.57
RFA_{SHOT}	33.15	59.85	42.71	33.89	35.24	35.95	40.13
RFA_{Tent}	28.07	56.58	38.45	28.65	30.50	32.07	35.72
UCAT	36.55	63.02	47.35	37.23	38.70	39.80	43.77
SRoUDA	42.17	62.73	47.99	42.57	41.54	42.91	46.65
Ours	**44.10**	**63.42**	**49.53**	**44.52**	**43.26**	**44.64**	**48.25**

TABLE III
THE AVERAGE ROBUST ACCURACY OF WIDERESNET-16-2 IN THE DIGIT-FIVE SCENARIO ACROSS 30 DOMAINS

Method	APGD	TPGD	FGSM	BIM	FAB	Square	Avg.
Source-only	26.95	57.05	39.31	27.66	28.69	30.21	34.97
CUA	29.37	58.41	41.33	30.05	31.24	32.49	37.14
SHOT	30.83	59.44	43.18	31.54	32.88	34.24	38.68
Tent	29.73	58.24	41.26	30.46	31.69	33.01	37.39
CUA&LFAT	39.89	61.89	46.55	40.29	40.50	41.66	45.13
CUA&PLAT	34.33	60.11	44.47	35.11	36.08	36.94	41.17
SHOT&LFAT	35.52	60.57	43.97	36.02	36.76	37.93	41.79
SHOT&PLAT	36.03	62.17	46.47	36.66	37.76	38.86	42.99
Tent&LFAT	33.43	60.24	44.69	34.46	35.48	36.23	40.75
Tent&PLAT	33.34	60.67	45.36	34.14	35.47	36.39	40.89
RFA_{CUA}	31.48	59.42	40.66	31.94	32.51	33.90	38.31
RFA_{SHOT}	30.99	59.23	39.17	31.41	31.83	33.47	37.68
RFA_{Tent}	27.19	56.88	39.11	27.86	29.12	30.53	35.11
UCAT	38.57	63.86	48.82	39.18	40.20	41.38	45.33
SRoUDA	42.64	61.91	47.70	42.95	41.34	42.66	46.53
Ours	**47.02**	**67.63**	**54.36**	**47.50**	**47.26**	**49.27**	**52.17**

black-box attacks using the Square attack. Specifically, the adversarial perturbation budget isϵ uniformly set to 8/255 in the Digit-Five scenario. Throughout all training or fine-tuning stages, the number of iterations for Projected Gradient Descent (PGD) is set to 10, with a step size of 2/255.

5) Compared Baselines: We implement and compare a total of 14 state-of-the-art baselines, categorized into three groups: (i) *three online DA methods* including Tent [9], CUA [11] and Source HypOthesis Transfer (SHOT) [3]; (ii) *six concatenation methods* combining online DA methods with UAT. The UAT includes Label-Free AT [34] and Pseudo-Label AT [13]; (iii) *five robust unsupervised DA (UDA) methods* including RFA [27] (based on Tent, CUA and SHOT respectively), UCAT [28] and SRoUDA [29].

B. Comparative Evaluations on Accuracy Improvement in Evolving Domain

This section's evaluation compares model natural accuracy and robust accuracy between our method and 14 baseline techniques. In comparison, the model is trained using the same initial weights, training samples. We also set all methods to have the same retraining window for fairness in evaluation. For the hyperparameters, we performed a grid search in the source domain and then adopted the selected value in the target domain.

1) Evaluation Settings: In low-resolution scenarios, we evaluate the CNN models VGG-11 [35] and WideResNet-16-2 [7] on all baselines to test their natural accuracy and robust accuracy. All accuracies are obtained as the average performance across 30 target domains after adaptation. In high-resolution scenarios, evaluations were conducted on select baselines using the Transformer model ViT-B-16 [36]. For evaluating robustness, the widely used PGD adversarial attack was employed. During testing, the PGD attack was configured with 20 steps for low-resolution scenarios and 10 steps for high-resolution scenarios.

2) Comparison Results: Fig. 6 displays the comparison results of the 14 techniques and we have three key observations:

a) Advantage in robust accuracy: Across all three workloads, RobustDA consistently maintains the highest robust accuracy compared to all baselines in continual evolving domains, as shown in Fig. 6 (a), (c), and (e). This is mainly attributed to the two factors: (i) The pseudo-label generator combined with the LoRA scheduler produces high-quality pseudo-labels, enhancing the effect of online adversarial training. (ii) The orthogonality constraint of the LoRA scheduler ensures that domain knowledge does not overwrite the Robust LoRA attributes inherited from the source domain.

b) Balance between natural and robust accuracy: While improving robust accuracy, RobustDA effectively maintains high natural accuracy, as depicted in Fig. 6 (b), (d), and (f). In contrast, although SRoUDA can enhance robust accuracy in the target domain, it comes at the cost of a significant decrease in natural accuracy, particularly evident in the Digit-Five workload (Fig. 6 (b) and (d)), where its natural accuracy is lower than all baselines. Meanwhile, UCAT achieves high natural accuracy in all Digit-Five workloads but exhibits suboptimal robust accuracy, comparable to concatenation methods like CUA&LFAT.

c) Effective to complex DNNs: When facing more challenging scenarios involving complex deep neural networks (Fig. 6 (e) and (f)), RobustDA's performance significantly surpasses existing methods. This is because baseline methods have overlapping parameters learned across different domains, tasks, and robustness aspects, leading to a substantial decrease in optimization effectiveness.

3) Results: Compared to baseline techniques, RobustDA increases robust accuracy by an average of 11.41% while also increasing natural accuracy an average of by 6.34%.

C. Comparative Evaluations on Retraining Overhead

Following the settings of the previous section, this section's evaluation focuses on the online retraining overhead, including retraining time and memory footprint.

1) Evaluation Settings: We conduct evaluations on both NVIDIA Jetson AGX Xavier and NVIDIA Jetson Nano using two workloads: VGG-11 and WideResNet-16-2. On the

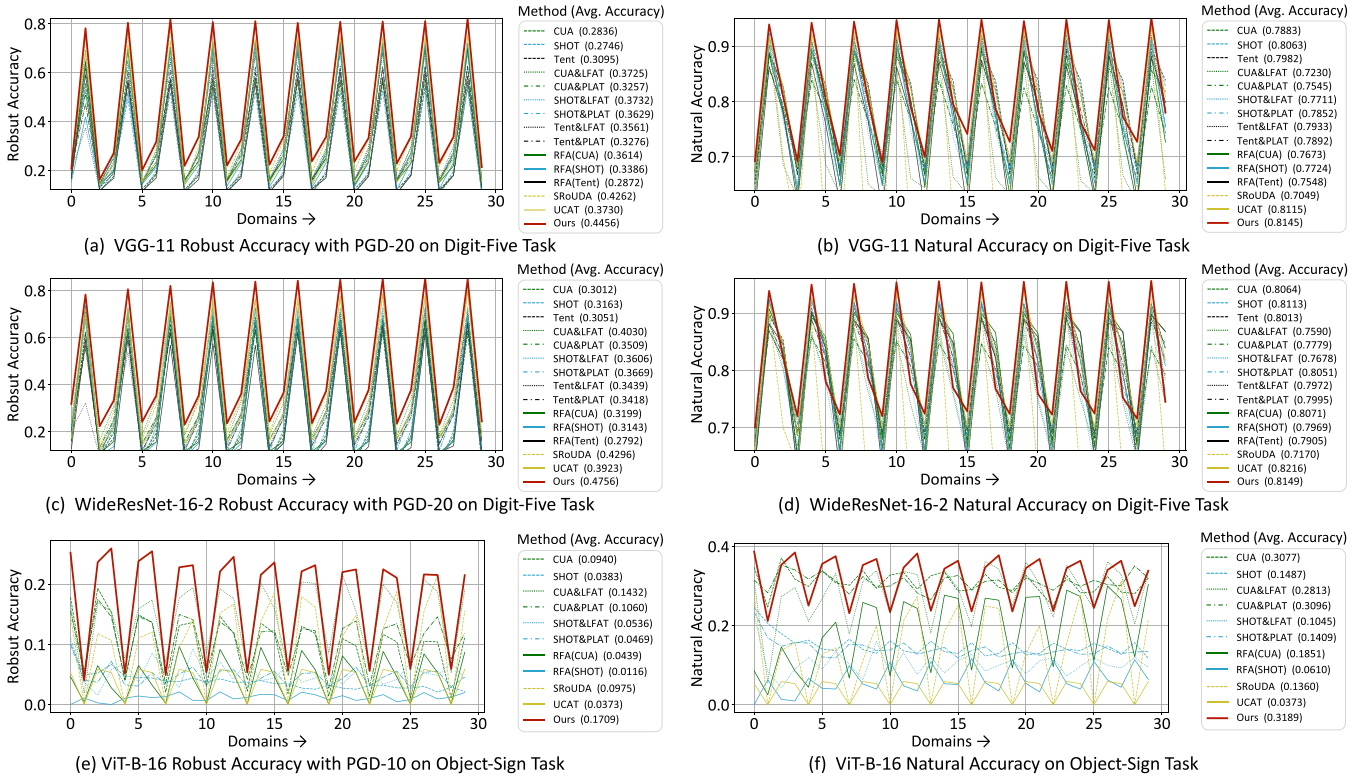

Fig. 6. Comparison of model robust accuracy and natural accuracy improvement in evolving domain.

NVIDIA Jetson AGX Xavier, which has ample memory of 32GB, we set the batch size for all methods to 100. However, for the NVIDIA Jetson Nano with only 4GB of memory, we set the batch size to 20. To compare baseline methods, we adjust the number of iterations while keeping other settings constant to achieve similar robust accuracy to our method. We exclude the RFA(Tent) method and the online DA method because, regardless of how we adjust the number of iterations, their robust accuracy consistently remains much lower than other baseline methods.

2) Comparison Results: The evaluation results are presented in Fig. 7 and Fig. 8. We analyze the results from the following three perspectives:

a) Impact of robust accuracy: Due to the varying learning capabilities of each baseline method, some methods fail to achieve the same robust accuracy as others in certain scenarios. For example, in Fig. 7 (a), the robust accuracy of RFA(SHOT) is on average 4.05% lower than other baselines. In contrast, RobustDA can achieve similar or higher robust accuracy compared to baseline methods. Specifically, it averages 3.88% higher on the NVIDIA Jetson AGX Xavier and 3.76% higher on the NVIDIA Jetson Nano.

b) Impact of memory usage: The difference between RobustDA and baseline methods is not that obvious, with an average reduction of only 79.42MB on the NVIDIA Jetson AGX Xavier and 34.64MB on the NVIDIA Jetson Nano. This is because the evaluated VGG-11 and WideResNet-16-2 models are relatively small. Even with LoRAs, only a small portion of the model parameters is reduced, accounting for a minor fraction of the total memory usage. For larger models like ViT, memory savings would be more significant.

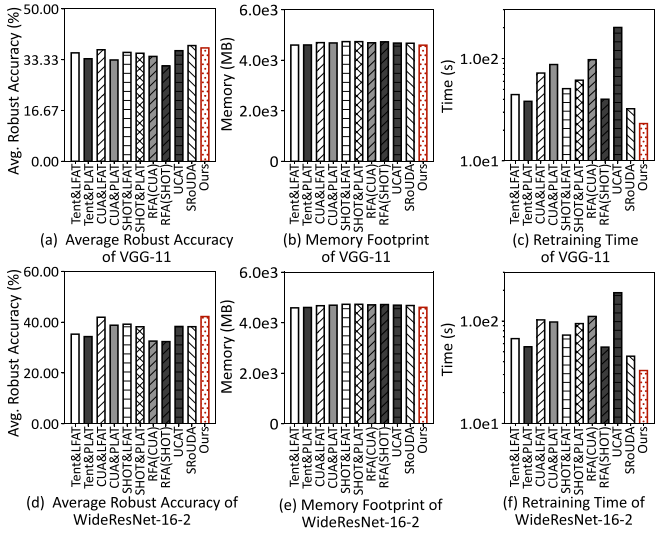

Fig. 7. Robust accuracy and overheads when baselines have similar robust accuracy to RobustDA on NVIDIA Jetson AGX Xavier.

c) Impact of retraining time: RobustDA substantially outperforms baseline methods, speeding up the retraining time by an average of 2.93x on NVIDIA Jetson AGX Xavier and 5.42x on NVIDIA Jetson Nano. RobustDA achieves the shortest retraining time for two reasons: (i) Before retraining, RobustDA can fully leverage accumulated knowledge by solving the adapter composition problem to generate higher-quality

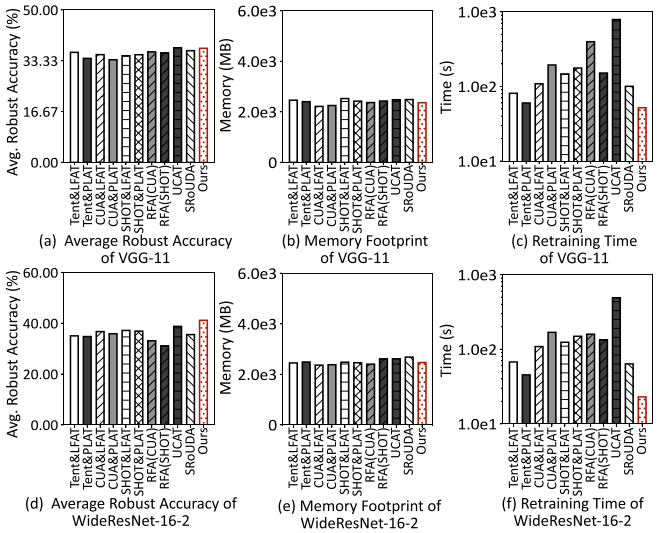

Fig. 8. Robust accuracy and overheads when baselines have similar robust accuracy to RobustDA on NVIDIA Jetson Nano.

pseudo-labels, enhancing robust learning. (ii) During training, the sample pool for new DA helps the robust optimization. It avoids conflicts in the optimization direction that could weaken the robust optimization, thus speeding up the convergence of the training.

3) Results: Compared to baseline techniques, RobustDA significantly reduces the retraining time by an average of 4.09x, while delivering higher accuracy and similar memory footprint.

D. Comparative Evaluations on Robustness With Different Adversarial Attacks

For further evaluating robustness, we conduct extended experiments employing six other popular adversarial attacks (APGD, TPGD, FGSM, BIM, FAB, and Square), covering five white-box attacks and one black-box attack (Square). Within white-box attacks, there are four iterative attacks and one single-step attack (FGSM).

1) Evaluation Settings: We evaluated the VGG-11 and WideResNet-16-2 models on the Digit-Five task. The reason for excluding the Object-Sign task is due to its large image resolution, which makes significant evaluation time-consuming. For all iterative attacks, we set the number of iterations to 20. APGD adopts the cross-entropy loss function with a threshold set to 0.75. For the black-box attack Square, the number of queries is set to 5000.

2) Comparison Results: We present the comprehensively comparison with different methods in Table II and Table III. With the result, we have two key observations:

a) Applicability of RobustDA to diverse attacks: RobustDA achieves the highest robust accuracy across all types of adversarial attack. Specifically, compared to the baselines, RobustDA's maximum improvement on VGG-11 is 10.75%, and the minimum is 3.37%. On WideResNet-16-2, RobustDA's maximum improvement is 13.21%, and the minimum is 7.41%. In contrast, the baselines exhibit varying robustness against different attacks. For instance, in the VGG-11 robustness evaluation (Table II), SRoUDA achieves suboptimal robust accuracy in most adversarial attacks, but fails to reach the robustness level of UCAT against TPGD. Similarly, the phonomenon is also observed in the WideResNet-16-2 robustness evaluation (Table III).

b) Applicability of RobustDA to diverse DNNs: RobustDA achieves the highest robust accuracy across different model architectures. Specifically, RobustDA achieves an average improvement of 10.57% on VGG-11 and 17.20% on WideResNet-16-2 compared to the source-only method. VGG-11 represents traditional deep neural network architectures based on repeated convolutions and pooling layers, while WideResNet-16-2 is a typical deep neural network architecture with residual connections. In contrast, the Baseline methods exhibit different robust accuracies across different architectures. For instance, compared to the source-only method on VGG-11 (Table II), SHOT&LFAT and Tent&LFAT relatively improve robust accuracy by 5.67% and 4.46%, respectively. These improments are higher than SHOT&PLAT and Tent&PLAT that improves only 5.1% and 2.1%. However, when applied to WideResNet-16-2 (Table III), the opposite result occurs.

3) Results: Compared to baseline techniques, RobustDA shows improvements ranging from 5.39% to 11.98% in resisting various adversarial attacks, with an average improvement of 9.62%; Compared to the source-only method, RobustDA demonstrates enhancements ranging from 7.98% to 16.78% in resisting various adversarial attacks, with an average improvement of 13.89%.

TABLE IV
COMPARISON OF USING DIFFERENT LORA DIMENSION ON WIDERESNET-16-2

Rank	Nat. Acc. (%)	Rob. Acc (%)	Size (MB)	Avg. Retraining Time (s)
$r = 2$	80.57	44.53	**2.768**	**77.21**
$\boldsymbol{r = 4}$	**81.49**	**47.26**	2.829	78.36
$r = 8$	80.38	44.71	2.950	80.30
$r = 16$	78.49	42.74	3.193	81.40

E. Discussion of LoRA Module Dimension

1) Evaluation Settings: We conducted evaluations on the WideResNet-16-2 model, keeping the number of iterations and other hyperparameters while varying only the LoRA module dimension r (i.e., rank). A range of LoRA module dimensions from 2 to 16 was selected, and the results are presented in Table IV.

a) Impact of LoRA module dimension on performance: As the LoRA module dimension increases, the model size increases continuously (e.g., the count of remained parameter grows from 3.32% to 19.19%). Correspondingly, the average retraining time increases as well. Notably, both natural and robust accuracies reach their maximum values at $r = 4$. This is because the unlabeled data of each target domain is limited, and there is an inherent rank based on its complexity. If the rank of our LoRA module is smaller than the rank of the data, information loss occurs due to compression, leading to a decrease in accuracy. Conversely, if the rank of our LoRA

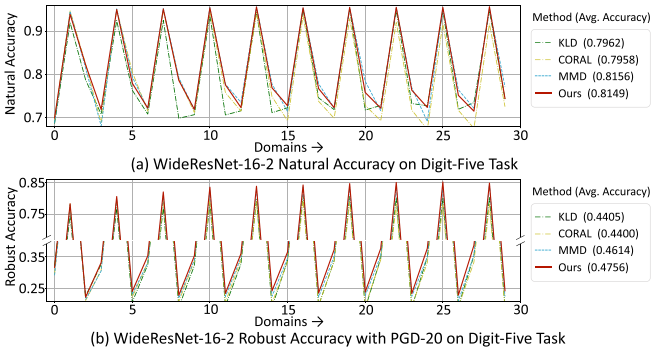

Fig. 9. Comparison of using different feature alignment method on WideResNet-16-2.

module is larger than the rank of the data, overfitting is likely during learning, also resulting in a decrease in accuracy. Therefore, when RobustDA addresses diffrent scenarios, it is crucial to select the appropriate LoRA module dimension to achieve optimal natural accuracy and robust accuracy.

F. Discussion of Feature Alignment Method

1) Evaluation Settings: We evaluated the WideResNet-16-2 model on the Digit-Five task, keeping the settings of other modules unchanged while only varying the feature alignment method used by the DA Trainer. The methods tested included KL divergence (KLD), Correlation Alignment (CORAL), Maximum Mean Discrepancy (MMD), and the two-step feature alignment adopted by RobustDA. The results are shown in Fig. 9. In Fig. 9 (b), the y-axis was truncated to clearly display the differences between methods.

a) Impact of feature alignment method on accuracy: Fig. 9 demonstrates that both natural and robust accuracies significantly improve with either RobustDA's two-step feature alignment or MMD, compared to KLD and CORAL. Specifically, the two-step alignment improves natural accuracy by an average of 1.89% and robust accuracy by an average of 3.54%. While MMD and the two-step alignment perform similarly in natural accuracy, the latter improves robust accuracy by 1.42%. This is because MMD alone does not adequately account for model robustness, leading to a greater trade-off between natural accuracy and robust accuracy. In contrast, our method uses samples with lower robust loss as anchors, ensuring that features of a particular class stay close to the class centroid. This keeps features away from the decision boundary, making them less susceptible to attacks and reducing this trade-off. Therefore, the two-step feature alignment adopted in RobustDA outperforms other methods in feature alignment.

G. Discussion of the Effectiveness of Loss Items and Scoring Mechanism

To demonstrate the importance of key components (i.e. three loss items and a sample scoring mechanism) employed by RobustDA. This section conducts ablation studies by progressively optimizing each component.

1) Evaluation Settings: We evaluate the WdiResNet-16-2 model on the Digt-Five task, folloing the setting of hyperparameters (e.g. learning rate, batch size, etc.) in Section IV-E.

TABLE V

THE AVERAGE NATURAL ACCURACY AND ROBUST ACCURACY USING DIFFERENT COMPONENTS EMPLOYED BY ROBUSTDA

$\mathcal{L}_{AT}$	$\mathcal{L}_O$	$\mathcal{L}_{FA}$	*scoring mechanism*	**Nat.Acc (%)**	**Rob.Acc (%)**
✓				50.43	30.97
✓	✓			80.22	46.87
✓	✓	✓		81.14	46.31
✓	✓	✓	✓	81.49	47.26

TABLE VI

AVERAGE ACCURACY OF SEEN DOMAINS BEFORE AND AFTER ADAPTING TO NEW DOMAIN

Number of Seen Domains	10	15	20	25	29
Avg Nat.Acc (Before)	80.04	74.95	77.20	79.94	79.97
Avg Nat.Acc (After)	81.95 ↑	76.39	80.90 ↑	78.29	80.31 ↑
Avg Rob.Acc (Before)	48.07	44.81	44.07	45.49	44.01
Avg Rob.Acc (After)	48.13 ↑	43.72	43.97	44.13	45.11 ↑

The evaluation contains four incremental steps: step 1 optimizes only the adversarial loss $\mathcal{L}_{AT}$; step 2 adds the orthogonality regularization loss $\mathcal{L}_O$; step 3 introduces the feature alignment loss $\mathcal{L}_{FA}$, and step 4 incorporates the scoring mechanism.

a) Impact of different components: The results in Table V show that optimizing all components together indeed produces the highest natural accuracy (81.49%) and robust accuracy (47.26%), thereby confirming the positive interaction among these losses and the scoring mechanism. At **step 1**, when optimizing only $\mathcal{L}_{AT}$, the model has the lowest performance due to parameter overlapping between domains and the lack of feature alignment, which is crucial for effective adaptation. At **step 2**, adding the orthogonality regularization loss $\mathcal{L}_O$ significantly improves both accuracy metrics. This is because $\mathcal{L}_O$ ensures that each Domain LoRA retains domain-specific knowledge, thereby producing more reliable adversarial samples and enhancing adversarial training. At **step 3**, the feature alignment loss $\mathcal{L}_{FA}$ further increases the natural accuracy but with lower robust accuracy, indicating the unstable adversarial training without the scoring mechanism for training samples. Finally, when incorporating with the scoring mechanism at **step 4**, both natural accuracy and robust accuracy improve, promoting the adversarial training and feature alignment for domain adaption.

b) Impact of loss $\mathcal{L}_O$ on seen domains: We extend the above evaluation to illustrate its importance in preserving knowledge of seen domains. Specifically, Table VI reports the average natural accuracy and robust accuracy of all seen domains before and after training with the new Domain LoRA. We can see that there is no significant accuracy decrease before and after the model adapts to a new domain. Therefore, $\mathcal{L}_O$ ensures effectively adapting to a new domain without interference the knowledge retained by seen domains.

H. Discussion of Different Hyperparameter Configurations

To investigate the sensitivity of RobustDA to hyperparameters, we conduct a sensitivity analysis on the three key hyperparameters: (1) λ_O, which reduces knowledge interference between Domain LoRAs; (2) λ_{FA}, which focus on natural accuracy; and (3) γ, which controls the reliability of generated pseudo-labels.

TABLE VII
PERFORMANCE OF ROBSTDA UNDER DIFFERENT γ

Items	$\lambda_O = 0.005, \lambda_{FA} = 1$			
	$\gamma = 0.65$	$\gamma = 0.75$	$\gamma = 0.85$	$\gamma = 0.95$
Nat.Acc(%)	78.02	79.28	79.43	81.26
Rob.Acc(%)	42.27	44.31	44.38	45.89

TABLE VIII
PERFORMANCE OF ROBSTDA UNDER DIFFERENT λ_{FA}

Items	$\lambda_O = 0.005, \gamma = 0.95$			
	$\lambda_{FA} = 0.1$	$\lambda_{FA} = 1$	$\lambda_{FA} = 10$	$\lambda_{FA} = 100$
Nat.Acc(%)	80.42	81.26	76.38	73.97
Rob.Acc(%)	46.16	45.89	39.61	32.95

TABLE IX
PERFORMANCE OF ROBSTDA UNDER DIFFERENT λ_O

Items	$\lambda_{FA} = 1, \gamma = 0.95$			
	$\lambda_O = 0.5$	$\lambda_O = 0.05$	$\lambda_O = 0.005$	$\lambda_O = 0.0005$
Nat.Acc(%)	80.96	81.03	81.26	60.77
Rob.Acc(%)	45.86	45.88	45.89	35.21

1) Experimental Setup: To ensure fairness, when evaluating one hyperparameter, the other two ones keep fixed with their optimal values. Specifically, we search λ_O from 0.0005 to 0.5 (as this loss usually exceeds 100); search λ_{FA} from 0.1 to 100 (as this loss is typically less than 1); and search γ from 0.65 to 0.95 (higher values indicate more reliable pseudo-labels).

a) Impact of different hyperparameter configurations: Table VII shows that as the γ increases, both natural and robust accuracies improve. This indicates that a higher γ value makes the generated pseudo-labels more reliable, thereby promoting the quality of subsequent adversarial samples and anchor images. In addition, Table VIII shows that when λ_{FA} is set to 1.0, RobustDA achieves the best trade off between natural and robust accuracy. When λ_{FA} is 0.1, the model favors adversarial training, but neglects the DA performance. However, when λ_{FA} is 100, it overlooks the orthogonal constraint between Domain LoRAs, negatively impacting subsequent adversarial training and online DA. Furthermore, Table IX shows that adjusting λ_O within the range of 0.005 to 0.5 results in slight performance changes. However, when the value is too small (e.g. 0.0005), performance significantly drops due to the disruption of the orthogonal constraint between Domain LoRAs, because it affects the effective execution of subsequent steps.

I. Discussion of Applicability to DA Datasets

To demonstrate the applicability of RobustDA in DA datasets, this section tests the ViT-B model using three prevalent DA datasets: PACS [37], OfficeHome [38], and VisDA-C [39].

1) Evaluation Settings: For DA setting, we set images from *photo* domain as the source domain of PACS dataset, using remaining domains (*art*, *sketch*, and *cartoon*) to form target domains. For the OfficeHome dataset, we select *Real World* domain as the source domain, and other domains (*Art*, *Clipart*, and *Product*) serve as target domains. For the VisDA-C dataset, the *real* domain is selected as the source domain, and *synthetic* domain is used as the target domain. In evaluation, we also tested three baselines that achieve the best performance in high-resolution scenarios, and assign the same retraining window for RobustDA and the three baselines.

TABLE X
THE AVERAGE NATURAL ACCURACY AND ROBUST ACCURACY OF VIT-B IN PACS, OFFICEHOME AND VISDA-C DATASETS

Method	PACS		OfficeHome		VisDA-C	
	Nat.Acc	Rob.Acc	Nat.Acc	Rob.Acc	Nat.Acc	Rob.Acc
CUA	42.65	16.86	11.04	2.90	32.68	9.30
CUA&PFAT	43.32	18.98	10.22	3.27	33.20	10.47
SRoUA	20.15	17.49	6.22	3.61	17.44	10.65
Ours	**44.22**	**30.49**	**11.45**	**5.25**	**33.88**	**16.82**

a) Applicability to more datasets: Table X shows that RobustDA achieves significant advantages in both natural and robust accuracies than other baselines in all three DA datasets. Specifically, CUA achieves similar natural accuracy but lacks mechanisms to optimize robust performance. CUA&PFAT improves robustness by incorporating pseudo-labeled adversarial training, but the overlap in domain knowledge reduces the quality of adversarial samples, thus weakening the training effect. In addition, SRoUA improves the model robustness in some cases (e.g., the OfficeHome and VisDA-C datasets in Table X) but fails to maintain natural accuracy due to the absence of DA mechanisms. In contrast, RobustDA facilitates high-quality adversarial sample generation, and supports both robust training and effective feature alignment in DA. This is because it considers both the natural accuracy and the robust accuracy in its optimization objectives. That is, RobustDA prevents knowledge interference by retaining domain-related and robustness-related knowledge in different adapters, and selects the most useful training samples to improve both accuracies.

V. CONCLUSION

This paper presents the design, implementation and evaluation of RobustDA, an approach allowing on-device lightweight DNN co-optimization for adapting evolving vision domains and resisting adversarial attacks at edge. Our approach can provide ideal robust accuracy across evolving vision domain shifts, while also preserving high natural accuracy with minimal overhead. Extensive evaluation results prove RobustDA's superior accuracy, efficiency, and applicability compared to existing techniques.

APPENDIX

This section introduces three categories of latest DA techniques, which are used as baselines in this work's evaluation of online domain adaptation and adversarial robustness.

Online DA methods enable models to continuously adapt to evolving data distribution of target domains, particularly

in scenarios where the test data distribution cannot be predicted during the training phase. We introduce three typical techniques as follows.

Tent [9] is the information entropy minimization method. It effectively improves the prediction confidence of new category data by reducing the entropy value of the model output.

CUA [11] adapts to continuously shifting target domains. During this process, it aims to avoid Catastrophic forgetting by ensuring consistent classification of previously seen examples in each DA.

SHOT [3] freezes the classifier module of the source model and learns the target-specific feature extraction module by exploiting both information maximization and self-supervised pseudo-labeling. This method implicitly aligns representations from the target domains to the source hypothesis.

Concatenation methods combine online DA with unsupervised online robust retraining, which uses unlabeled target domain data in adversarial training to improve robust accuracy. We introduce two typical techniques as follows.

LFAT [34] (&CUA/SHOT/Tent) uses unlabeled data as a competitive alternative to train adversarially robust models, and aims to improve adversarial robustness by training models to be invariant to adversarial perturbations.

PLAT [13] (&CUA/SHOT/Tent) maximizes the agreement between clean images and their adversarial examples by a contrastive loss in the output space.

Robust unsupervised DA methods aim to improve natural and robust accuracies simultaneously. This means the model can maintain high predictive performance even when facing evolving input data and adversarial attacks. We introduce three typical techniques as follows.

RFA [27] aligns the features of the UDA model with the robust features learned by ImageNet pre-trained models in DA. It utilizes both labeled and unlabeled data and instills robustness without any adversarial intervention or label requirement during domain adaptation training.

Unsupervised Cross-domain Adversarial Training (UCAT) [28] addresses the problem of adversarial training in unlabeled target domain. This method effectively leverages the knowledge of the labeled source domain to prevent the adversarial samples from misleading the training process.

SRoUDA [29] is a meta self-training pipeline for improving adversarial robustness in unsupervised DA. This method starts with pre-training a source model using labeled source data and unlabeled target data with a developed random masked augmentation (RMA). It then alternates between adversarial target model training on pseudo-labeled target data and fine-tuning source model by a meta step.

Conclusion When performing continuous DA at edge, RobustDA outperforms existing techniques with highers natural and robust accuracies for three reasons. *First*, RobustDA leverages the LoRA structure to separate domain knowledge from robust knowledge, thus storing them in distinct LoRA parameters. *Second*, RobustDA optimizes the domain-specific LoRA parameters orthogonally, ensuring that the knowledge learning of one domain does not interfere other domains. Our approach further utilizes learned LoRA parameters to efficiently generate adversarial samples for adversarial training. *Finally*, by maintaining an *anchor* image sample pool, RobustDA can perform more effective feature alignment between the source and target domains to improved natural accuracy.

REFERENCES

[1] Q. Zhang, R. Han, C. H. Liu, G. Wang, and L. Y. Chen, "EdgeVisionBench: A benchmark of evolving input domains for vision applications at edge," in *Proc. IEEE 39th Int. Conf. Data Eng. (ICDE)*, Apr. 2023, pp. 3643–3646.

[2] H. Liu, M. Long, J. Wang, and Y. Wang, "Learning to adapt to evolving domains," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, 2020, pp. 22338–22348.

[3] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6028–6039.

[4] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 2, Jul. 2015, pp. 1180–1189.

[5] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[7] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[8] F. Croce et al., "RobustBench: A standardized adversarial robustness benchmark," 2020, *arXiv:2010.09670*.

[9] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," 2020, *arXiv:2006.10726*.

[10] M. Mancini, H. Karaoguz, E. Ricci, P. Jensfelt, and B. Caputo, "Kitting in the wild through online domain adaptation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1103–1109.

[11] A. Bobu, E. Tzeng, J. Hoffman, and T. Darrell, "Adapting to continuously shifting domains," in *Proc. ICLR Workshop*, 2018, pp. 1–4.

[12] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang, "Unlabeled data improves adversarial robustness," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 11192–11203.

[13] J. Yang et al., "Exploring robustness of unsupervised domain adaptation in semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9194–9203.

[14] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Computer Vision—ECCV*, Amsterdam, The Netherlands. Cham, Switzerland: Springer, 2016, pp. 443–450.

[15] Y. Ganin et al., "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 59, pp. 1–35, Jan. 2016.

[16] A. M. N. Taufique, C. S. Jahan, and A. Savakis, "Continual unsupervised domain adaptation in data-constrained environments," *IEEE Trans. Artif. Intell.*, vol. 5, no. 1, pp. 167–178, Jan. 2023.

[17] W. Ahmed, P. Morerio, and V. Murino, "Continual source-free unsupervised domain adaptation," in *Proc. Int. Conf. Image Anal. Process.* Cham, Switzerland: Springer, 2023, pp. 14–25.

[18] J. Lee et al., "Entropy is not enough for test-time adaptation: From the perspective of disentangled factors," 2024, *arXiv:2403.07366*.

[19] S. Gui, X. Li, and S. Ji, "Active test-time adaptation: Theoretical analyses and an algorithm," 2024, *arXiv:2404.05094*.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[21] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*. London, U.K.: Chapman & Hall, 2018, pp. 99–112.

[22] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2206–2216.

[23] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 484–501.

[24] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2196–2205.

[25] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, vol. 97, 2019, pp. 7472–7482.
[26] A. Shafahi et al., "Adversarially robust transfer learning," 2019, *arXiv:1905.08232*.
[27] M. Awais et al., "Adversarial robustness for unsupervised domain adaptation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 8568–8577.
[28] J. Zhang, H. Chao, and P. Yan, "Toward adversarial robustness in unlabeled target domains," *IEEE Trans. Image Process.*, vol. 32, pp. 1272–1284, 2023.
[29] W. Zhu, J.-L. Yin, B.-H. Chen, and X. Liu, "SRoUDA: Meta self-training for robust unsupervised domain adaptation," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 3, pp. 3852–3860.
[30] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," 2021, *arXiv:2106.09685*.
[31] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1406–1415.
[32] Q. Zhang, R. Han, C. H. Liu, G. Wang, and L. Y. Chen, "ElasticDNN: On-device neural network remodeling for adapting evolving vision domains at edge," *IEEE Trans. Comput.*, vol. 73, no. 6, pp. 1616–1630, Jun. 2024.
[33] Y. Dong et al., "Benchmarking adversarial robustness on image classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 321–331.
[34] J.-B. Alayrac, J. Uesato, P.-S. Huang, A. Fawzi, R. Stanforth, and P. Kohli, "Are labels required for improving adversarial robustness?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 12214–12223.
[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[36] A. Dosovitskiy et al., "An image is worth 16 × 16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[37] K. Zhou, Y. Yang, T. Hospedales, and T. Xiang, "Deep domain-adversarial image generation for domain generalisation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 13025–13032.
[38] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5018–5027.
[39] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "VisDA: The visual domain adaptation challenge," 2017, *arXiv:1710.06924*.

Xinyu Guo is currently pursuing the master's degree with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include edge intelligence and domain adaptation.

Xiaojiang Zuo is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include federated learning and edge computing.

Rui Han received the M.Sc. degree (Hons.) from Tsinghua University, China, in 2010, and the Ph.D. degree from Imperial College London, U.K., in 2014. He is currently an Associate Professor with the School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include system optimization for cloud data center workloads (in particular highly parallel services and deep learning applications). He has over 40 publications in these areas, including papers at MobiCOM, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, INFOCOM, and ICDCS.

Junyan Ouyang is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include differential privacy, federated learning, and edge computing.

Jing Xie received the B.Eng. degree from Beijing Institute of Technology, Beijing, China. Her research interests include artificial intelligence and reinforcement learning.

Chi Harold Liu (Senior Member, IEEE) received the B.Eng. degree from Tsinghua University, Beijing, China, and the Ph.D. degree from Imperial College London, London, U.K. He is currently a Full Professor and the Vice Dean of the School of Computer Science and Technology, Beijing Institute of Technology, Beijing. Before that, he was with IBM Research China and Deutsche Telekom Laboratories, Berlin, Germany, and the IBM T. J. Watson Research Center, USA. His current research interests include big data analytics, mobile computing, and deep learning. He is a fellow of IET and the Royal Society of the Arts. He is an Associate Editor of IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

Qinglong Zhang is currently pursuing the master's degree with the School of Computer Science and Technology, Beijing Institute of Technology. His work focuses on edge intelligence and deep learning applications.

Ying Guo is currently a Research Fellow with the School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan, China. Her research interests include cloud computing, big data, and software engineering.

Jing Chen is currently a Research Fellow with the School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Science), Jinan, China. Her research interests include cloud computing, artificial intelligence, and computing power networks.

Lydia Y. Chen (Senior Member, IEEE) received the B.A. degree from National Taiwan University and the Ph.D. degree from The Pennsylvania State University. She is currently an Associate Professor with the Department of Computer Science, Delft University of Technology (TU Delft). Prior to joining TU Delft, she was a Research Staff Member with the IBM Zurich Research Laboratory from 2007 to 2018. She has published more than 80 articles in journals, such as IEEE TRANSACTIONS ON DISTRIBUTED SYSTEMS and IEEE TRANSACTIONS ON SERVICE COMPUTING, and conference proceedings, such as INFOCOM, Sigmetrics, DSN, and Eurosys. Her research interests include dependability management, resource allocation, and privacy enhancement for large scale data processing systems and services. She was a co-recipient of the Best Paper Award from CCgrid 2015 and eEnergy 2015.