A. Artifact Appendix

A.1 Abstract

On-device training Neural Networks (NNs) has been a crucial catalyst towards privacy-preserving and personalized mobile intelligence. Recently, a novel training paradigm, namely Parameter-Efficient Training (PET), is attracting attention in both the machine learning and system community. In our preliminary measurements, we find PET well-suited for on-device scenarios; yet, its parameter efficiency does not translate coequal to time efficiency on resource-constrained devices, as the training time is dominated by the frozen layers.

To this end, this work presents `PieBridge`, an on-device training framework with both time and parameter efficiency. Its key idea is to dynamically approximate the frozen layers to cheaper ones (subnets) with data awareness during PET. To achieve effective and efficient approximate training, we introduce (1) a pre-training-assisted on-cloud subnets generation method and (2) an edge-friendly on-device data-aware subnets routing method. The subnets generation method performs fine-grained pruning and latent space alignment to generate a series of high-quality proxy subnets with varying speed-accuracy trade-offs for the deployment-ready NN. The subnets routing method perceives data diversity from two unique perspectives (referred to as importance and difficulty). The routing strategy is provided by an offline-learning and online-estimation fusion, which is accurate, end-to-end and cost-effective on devices. Through extensive experiments, we show that `PieBridge` exhibits up to 2.5× training speedup compared to state-of-the-art PET methods, and up to 6.6× speedup compared to traditional full model training and other on-device training frameworks, without compromising parameter efficiency and accuracy.

A.2 Artifact check-list (meta-information)

- **Model: ResNet**
- **Data set: Caltech101, Caltech256, Dogsvscats, DTD**
- **Run-time environment: Ubuntu18.04.1**
- **Hardware: Jetson TX2**
- **Execution: Python and bash scripts**
- **Metrics: Wall-clock training time, best test accuracy**
- **Output: training logs and processed csv**
- **Experiments: measure the on-device training time and accuracy.**
- **How much disk space required (approximately)?: 25GB**
- **How much time is needed to prepare workflow (approximately)?: 1 Hour**
- **How much time is needed to complete experiments (approximately)?: 24–48 Hours**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: MIT**
- **Data licenses (if publicly available)?: MIT**
- **Archived (provide DOI)?: Not yet**

A.3 Description

A.3.1 How to access

Our code is publicly available at GitHub[1], and we have packed our code into a docker image, which can be downloaded from DockerHub[2].

A.3.2 Hardware dependencies

Our artifact is depoyed on an edge board Jetson TX2 with 8GB RAM and 16 GB swap size. We attach an extra 900 GB NVME SSD to TX2 since the born disk size is insufficient.

(Recommended) Please run the following code to configure the swapping files.

```
sudo systemctl disable nvzramconfig
sudo fallocate -l 16G /mnt/16GB.swap
sudo mkswap /mnt/16GB.swap
sudo swapon /mnt/16GB.swap
```

Change the mounted path to your own.

A.3.3 Software dependencies

We recommend you directly using the docker image, or checking the software version details in the docker image and manually installing them.

```
docker pull yinwangsong2000/piebridge_ae
```

Then, enter the docker container and go ahead for the following preparation of data and weights.

A.3.4 Data sets and weights

The datasets and model weights are pre-uploaded in Google Drive[3]. You can download them manually by

```
gdown <file-id>
```

Put the dowloaded data in the corresponding path of *./datasets/** and *./proxynetworks/**.

We also provide an automated script for downloading in *./scripts/downloading.sh*.

A.4 Experiment workflow

Execute the following instructions in the docker container for reproducing the main results.

```
cd PieBridge
bash ./scripts/run_e2e.sh 0
```

The cmd parameter "0" means GPU ID, which is fixed in Jetson TX2. After about 24 hours, the training logs will be recorded in *./res/*/log.txt*, including `PieBridge` (Ours) and its baselines.

A.5 Evaluation and expected results

The main metrics of our system are the wall-clock on-device training time and the best test accuracy. After running the experiments, you can either check the training log, or use the following scripts to generate a csv file that contains the evaluated metrics of each case.

Running the experiments in one click:

```
cd PieBridge
bash ./scripts/run_e2e.sh 0
```

Generating the CSV file:

```
cd PieBridge
python3 PieBridge/res/train_log/resnet/gen_csv.py
```

The generated CSV file should look like:

For reference, we also provide the training logs abtained on our device in Google Drive. You can directly download them and generate the csv file for a quick reproduction.

[1] `https://github.com/yinwangsong/PieBridge`

[2] `https://hub.docker.com/r/yinwangsong2000/piebridge_ae`

[3] `https://drive.google.com/drive/folders/1aoZ9SorbMS_hEw9stMvm-nMfC6y0dKNG?usp=sharing`

	Ours	PET	FTAll
Time	0.44	0.65	2.18
Acc.	91.85	91.15	92

Table 1. The generated CSV file of ResNet, Caltech101.

A.6 Notes

A.6.1 Training time

Please note that the actual wall-clock time for the experiment is longer than the training time reported in our paper and the csv file. This is mainly due to our system performs on-device evaluation for the trained model periodically to monitor the training status of the current model. In real-world applications of on-device training, such an extra overhead is unnecessary. As a result, we only record the pure training time of each method.