A. Artifact Appendix

A.1 Abstract

Transformer-based pre-trained models have revolutionized NLP for superior performance and generality. Fine-tuning pre-trained models for downstream tasks often requires private data, for which federated learning is the de-facto approach (i.e., FedNLP). However, our measurements show that FedNLP is prohibitively slow due to the large model sizes and the resultant high network/computation cost. Towards practical FedNLP, we identify as the key building blocks *adapters*, small bottleneck modules inserted at a variety of model layers. A key challenge is to properly configure the depth and width of adapters, to which the training speed and efficiency is highly sensitive. No silver-bullet configuration exists: the optimal choice varies across downstream NLP tasks, desired model accuracy, and mobile resources. To automate adapter configuration, we propose `AdaFL`, a framework that enhances the existing FedNLP with two novel designs. First, `AdaFL` progressively upgrades the adapter configuration throughout a training session; the principle is to quickly learn shallow knowledge by only training fewer and smaller adapters at the model's top layers, and incrementally learn deep knowledge by incorporating deeper and larger adapters. Second, `AdaFL` continuously profiles future adapter configurations by allocating participant devices to trial groups. Extensive experiments show that `AdaFL` can reduce FedNLP's model convergence delay to no more than several hours, which is up to 155.5× faster compared to vanilla FedNLP and 48× faster compared to strong baselines.

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

- **Model: BERT-base**
- **Data set: 20NEWS**
- **Run-time environment: Ubuntu 18.04.1**
- **Hardware: desktop computer and GPU server**
- **Execution: python and bash scripts**
- **Metrics: time-to-accuracy**
- **Output: 20news.csv**
- **Experiments: measure the time-to-accuracy of AdaFL**
- **How much disk space required (approximately)?: 25G**
- **How much time is needed to prepare workflow (approximately)?: 10 mins**
- **How much time is needed to complete experiments (approximately)?: 20 hours**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: MIT**

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 deployed on a desktop computer and a GPU server. The desktop computer has 8 cores CPU and 16G memory. The GPU server has 64 cores CPU, 755G memory and 8 NVIDIA A40 GPU.

We also experiment on 4 NVIDIA Tesla V100 GPUs. It works well after remapping clients on GPU (Section A.8.2).

[1] `https://github.com/UbiquitousLearning/AdaFL`

[2] `https://hub.docker.com/repository/docker/caidongqi/adafl/general`

acc	dep	wid	round
0.0	0	8	-1
0.468	0	8	79
0.603	2	8	88
0.742	2	16	180

Table 1. First four rows of our "20news.csv".

A.3.3 Software dependencies

We use docker to pack our code and all dependencies.

A.3.4 Data sets

Required datasets have been downloaded in the docker image.

A.3.5 Models

We have downloaded the required pre-trained models in the docker image.

A.4 Installation

```
docker pull caidongqi/adafl:1.0.1
docker run -it --gpus all --network host \
caidongqi/adafl:1.0.1 bash
```

A.5 Experiment workflow

The following instructions are executed on the docker image. We strongly recommend you to read Section A.8 first to avoid possible bugs.

```
# Go to the training workspace
cd experiments/distributed
cd transformer_exps/run_tc_exps
# Modify clients mapping in gpu_mapping.yaml
python trial_error.py \
  --dataset 20news\
  --round -1 \
  --depth 0 \
  --width 8 \
  --time_threshold 60 \
  --max_round 3000
# Wait for 20 trail rounds (about 20 hours)
# Process the results
python process.py
```

A.6 Evaluation and expected results

A.6.1 Expected reuslts on BERT-base+20NEWS

The emulation results are stored in 'results/20news.csv'. File '20news.csv' has four columns: 'acc', 'dep', 'wid', 'round'. For example, as shown in Table 1, the last row means that the validation accuracy is 0.742 at round 180. And adapter configuration is depth 2 and width 16.

You could draw Figure 5(a) in our paper based on your newly obtained results by running the following commands:

```
# Go to root directory
cd /app
cd exps_data
python draw-performance-baseline.py
```

For reference, our results are stored in 'exps_data/nice_results/20news.csv'.

A.6.2 Other experiment results

You could act as above to run other model+datasets. For simplicity, we provide all of our experiment results in folder 'exps_data'. You

could reproduce all figures in our paper by running the following commands:

```
# Go to root directory
cd /app
cd exps_data
bash run.sh
```

Generated figures will be stored in 'figs'. And we have provided all figures in our paper in folder 'figs/ref' for reference.

A.7 Experiment customization

For customized datasets, please refer to our README.md and folder 'data'. For customized models, you could refer to file 'initializer.py', Huggingface will download the pre-trained models automatically.

A.8 Notes

A.8.1 Clean processes.

Ctrl-C could not kill all related processes. Please run the provided shell to stop running processes so as to release resource.

```
# Go to the training workspace
cd experiments/distributed
cd transformer_exps/run_tc_exps
bash clean.sh
```

A.8.2 How to map clients to GPUs?

Specifically, you need to modify line 13 in gpu_mapping.yaml according to your GPU capacity. For example, if you have 8 GPUs, you could map 5 clients (and 1 server) to them as follows.

```
NVIDIA_A40: [2, 0, 0, 1, 1, 0, 2, 0]
```

Each simulated client needs 2GB GPU memory, 10GB RAM memory. Please remember that we have three sideline training groups, so you need at least 30GB GPU memory and 150GB RAM Memory run all the clients.

A.8.3 How to monitor and debug our system?

We still take BERT-base+20NEWS as an example. After running 'train_error.py', detailed client training logs will be stored in the folder 'results/reproduce/20news-Trail-0-60'. Those logs will be summarized into file 'results/20news-depth-0-freq-60.log'. You could obtain error information in the detailed logs and monitor the training process by tailing the summary log.