Phison Electronics Corporation
aiDAPTIVLink2.0 Install SOP
& User manual

Version 1.6

Phison Electronics Corporation
Tel: +886-37-586-896 Fax: +886-37-587-868
E-mail: sales@phison.com / suppport@phison.com

Phison may make changes to specifications and product description at any time without notice. PHISON and the Phison logo are trademarks of Phison Electronics Corporation, registered in the United States and other countries. Products and specifications discussed herein are for reference purposes only. Copies of documents which include information of part number or ordering number, or other materials may be obtained by emailing us at sales@phison.com or support@phison.com.

REVISION HISTORY

Revision	Draft Date	History	aiDAPTIVLink Version	Author
0.1	2024/09/18	Preliminary version	NXUN201T00	Lily Wu
1.0	2024/09/20	First release	NXUN201.00	Lily Wu
1.1	2024/09/27	Content adjust		Sean Liou
1.2	2024/10/09	1. Update section3. AiDAPTIVToolkit version 2. Update section3.3 performance result		Sean Liou
1.3	2024/10/18	1. Modify section 2.1.1 OS requirement 2. Section 2.2.2 add single SSD mount method 3. Move section 2.2.3 to Appendix E 4. Move AVL list to Appendix F 5. Update section2.3		Sean Liou
1.4	2024/10/24	1. Update aiDAPTIVToolkit version to 2.1.0 2. Correct typo 3. Modify section3.2 content		Sean Liou
1.5	2024/10/30	Add section 2.6 Quick Start		Sean Liou
1.6	2024/11/29	Correct support model list		Sean Liou

TABLE OF CONTENTS

REVISION HISTORY **3**

TABLE OF CONTENTS **4**

LIST OF FIGURES **7**

LIST OF TABLES **8**

1. CHECK DEVICE **10**

1.1. Check GPU is in Approved Vendor List? (AVL) 10

1.2. Check the number of GPUs 10

1.3. Check DRAM and aiDAPTIVCache from different LLM model size 10

1.4. Check CPU configuration 11

2. INSTALLATION AND PREPARATION **12**

2.1. Environment Preparation 12

2.1.1. Requirements *12*

2.1.2. Install GPU Driver *12*

2.1.3. Install GPU Toolkit *13*

2.2. aiDAPTIVLink Installation 14

2.2.1. Deploy aiDAPTIV *14*

2.2.2. Disk Setup (LVM Setting) *15*

2.3. Login huggingface 17

2.3.1. How to get Hugging Face token *17*

2.3.2. How to register the Hugging Face token *19*

2.4. Download Llama-3.1-8B-Instruct 20

2.4.1. Download Llama-3.1-8B-Instruct model *20*

2.5. Run aiDAPTIV 22

2.5.1. Start training your model *22*

2.5.2. Monitor training status *24*

2.5.3. Successful example *25*

2.5.4. "phisonai2" command arguments *25*

2.6. Quick Start 29

2.6.1. Without aiDAPTIVLink *29*

2.6.2. With aiDAPTIVLink *29*

3. BENCHMARK BY AIDAPTIV TOOLKIT (OPTIONAL) 30

3.1. aiDAPTIV Toolkit Installation flow 30

3.1.1. Download aiDAPTIV Toolkit 30

3.1.2. Unzip download file 30

3.1.3. Change to folder 30

3.1.4. Deploy aiDAPTIV Toolkit and related library 30

3.2. Start aiDAPTIV Toolkit 31

3.2.1. Enter aiDAPTIV Toolkit folder 31

3.2.2. Modify training setting in aiDAPTIV_Toolkit2.1.0/project.ini 31

3.2.3. Run aiDAPTIV Toolkit Performance Test 32

3.3. Performance Result 33

3.3.1. Log directory Description 33

3.3.1.1. Figure_4GPU_41bs (Model: Llama-3.1-8B-Instruct) 33

3.3.1.2. Training log 4GPU_41bs (Model: Llama-3.1-8B-Instruct) 34

3.3.1.3. Performance_result.xlsx(Model: Llama-3.1-8B-Instruct) 34

3.3.2. Performance 34

3.3.3. Reference data 35

APPENDIX A. HOW TO PREPARE YOUR OWN DATASET 36

A.1 Pre-training format 36

A.2 Instruction Tuning Format 38

A.2.1 Default Instruction Tuning 38

A.2.2 Customized Prompt Format 39

APPENDIX B. HOW TO EVALUATE TRAINED MODEL 41

B.1 Quick Start 41

B.2 Prepare Your Evaluation Data 42

B.3 Prepare Your Evaluation Function (Advance) 42

APPENDIX C. HOW TO INFERENCE TRAINED MODEL 46

C.1 Quick Start 46

C.2 Prepare Your Inference Data 47

C.3 Prepare Your Inference Function (Advance) 47

APPENDIX D. HOW TO TRAIN MODEL IN DOCKER 50

D.1 Docker Installation option 50
D.2 Run aiDAPTIV Image 51
APPENDIX E. HOW TO SET SWAP FILE 52
APPENDIX F. APPROVED VENDOR LIST (AVL) 53
F.1 GPU AVL 53
F.2 CPU AVL 53
F.3 Support Model list 54

LIST OF FIGURES

Figure 2-1 GPU Driver Installation 13
Figure 2-2 Deploy aiDAPTIV+ 14
Figure 2-3 aiDAPTIVLink file 14
Figure 2-4 Deploy aiDAPTIV+ successful 14
Figure 2-5 aiDAPTIV2 folder 14
Figure 2-6 Clear disk 15
Figure 2-7 Create LVM 15
Figure 2-8 LVM 16
Figure 2-9 Login huggingface 17
Figure 2-10 selec setting 17
Figure 2-11 Access tokens 17
Figure 2-12 selec setting 18
Figure 2-13 Get tokens 18
Figure 2-14 Get tokens in account 18
Figure 2-15 Login huggingface 19
Figure 2-16 Download Llama-3.1-8B-Instruct 20
Figure 2-17 Download Llama-3.1-8B-Instruct successful 21
Figure 2-18 Weight of the model 21
Figure 2-19 Location of model 22
Figure 2-20 Location of SSD mount path 22
Figure 2-21 aiDAPTIV training log 24
Figure 2-22 Finetuned model 25
Figure 2-23 Check aiDAPTIVLink version 25
Figure 2-24 Import optimizer 29
Figure 2-25 Create model instance and add model parameters to optimizer 29
Figure 2-26 Import API from site-packages 29
Figure 2-27 Create model instance and call initialize 29
Figure 3-1 aiDAPTIV Toolkit and related library 30
Figure 3-2 aiDAPTIV Toolkit Performance Test 32
Figure 3-3 4GPU_18bs 33

Figure 3-4 Llama-3.1-8B-Instruct log 34
Figure 3-5 Example of Llama-3.1-8B-Instruct performance result 34
Figure 3-6 Example of Llama-3.1-8B-Instruct performance result 34
Figure B-1 Log file 45
Figure C-1 Log file 49
Figure D-1 docker image 50
Figure D-2 docker image list 50
Figure D-3 docker successful example 51

LIST OF TABLES

Table 1-1 Recommend Configuration 10
Table 2-1 Recommend environment 12
Table 3-1 Reference performance 35
Table A-1 CSV format 36
Table F-1 GPU AVL 53
Table F-2 CPU AVL 53
Table F-3 Support model list 54

Preface

Purpose

This manual is intended solely for aiDAPTIV partners.

- aiDAPTIVLink Version: NXUN201.00
- aiDAPTIV Toolkit Version: 2.1.0
- aiDAPTIVCache Family：

Model
AI100E SSD

1. CHECK DEVICE

In this section, we will check machine device GPU/CPU/RAM/aiDAPTIVCache before start to use aiDAPTIVLink.

After completing this section, you will know which GPU/CPU is in Phison AVL, and how many RAM size for different LLM model.

1.1. Check GPU is in Approved Vendor List? (AVL)

- Check GPU is in AVL list (Appendix F - GPU AVL).
- Needs to do validation if GPU is not in AVL.
- It is recommended that the GPU must be connected to the PCIe Slot of X16. Connecting other Slots (ex: X8, X4) will result in GPU performance degradation.

1.2. Check the number of GPUs

- Number of GPUs = 2^n (n=0,1,2,3,4, GPUs = 1,2,4,8,16)

1.3. Check DRAM and aiDAPTIVCache from different LLM model size

- DRAM and aiDAPTIVCache from different LLM model size

Table 1-1 Recommend Configuration

LLM model size	<13B	34B	70B	180B
DRAM	64GB	64GB	128GB	128GB
aiDAPTIVCache capacity	1TB	1TB	2TB	2TB
AiDAPTIVCache count	1	2	2	4
aiDAPTIVCache slot count	1	2	2	4

  - Rack server: Use U.2 slot.
  - Recommend Gen4 or above.
  - Recommend DRAM 2933MHz or above.
  - Recommended DRAM size is 128GB or more
  - Recommended DRAM channel number is 8 or more, ex: 16GB x8

1.4. Check CPU configuration

- Check CPU is in AVL list (Appendix F - CPU AVL).
- Recommend CPU Cores minimum 8 or above

CPU lanes calculate by GPU count and aiDAPTIVCache count.
Total PCIe lanes = GPU count * 16 + aiDAPTIVCache count * 4

*Example : For GPU count =4 and 70B LLM model size (aiDAPTIVCache 2)

Total 4 * 16 + 2 * 4 = 72 lanes, It means at least 72 lanes for optimal performance.

2. INSTALLATION AND PREPARATION

In this section, we will start installing the aiDAPTIVLink and explain how to use it. After completing this section, you will know how to use aiDAPTIVLink for Domain training or fine-tuning.

2.1. Environment Preparation

2.1.1. Requirements

- Recommend environment

Table 2-1 Recommend environment

Category	Detail
OS	Ubuntu 22.04.3 Desktop
GPU driver	Nvidia driver version 535 installation

2.1.2. Install GPU Driver

- Install Driver (Estimated time: 5 min)
  - Install NVIDIA Driver (non-SXM GPU)

```
sudo apt install nvidia-utils-535
sudo apt install nvidia-driver-535
```

  - Install NVIDIA Driver (Optional Only for SXM GPU)

```
sudo apt install nvidia-utils-535
sudo apt install nvidia-driver-535-server
sudo apt-get install cuda-drivers-fabricmanager-535
sudo systemctl start nvidia-fabricmanager
```

- Reboot system

```
sudo reboot
```

- Successful example

```
nvidia-smi
```

There would be a GPU driver version and CUDA version show in the log.

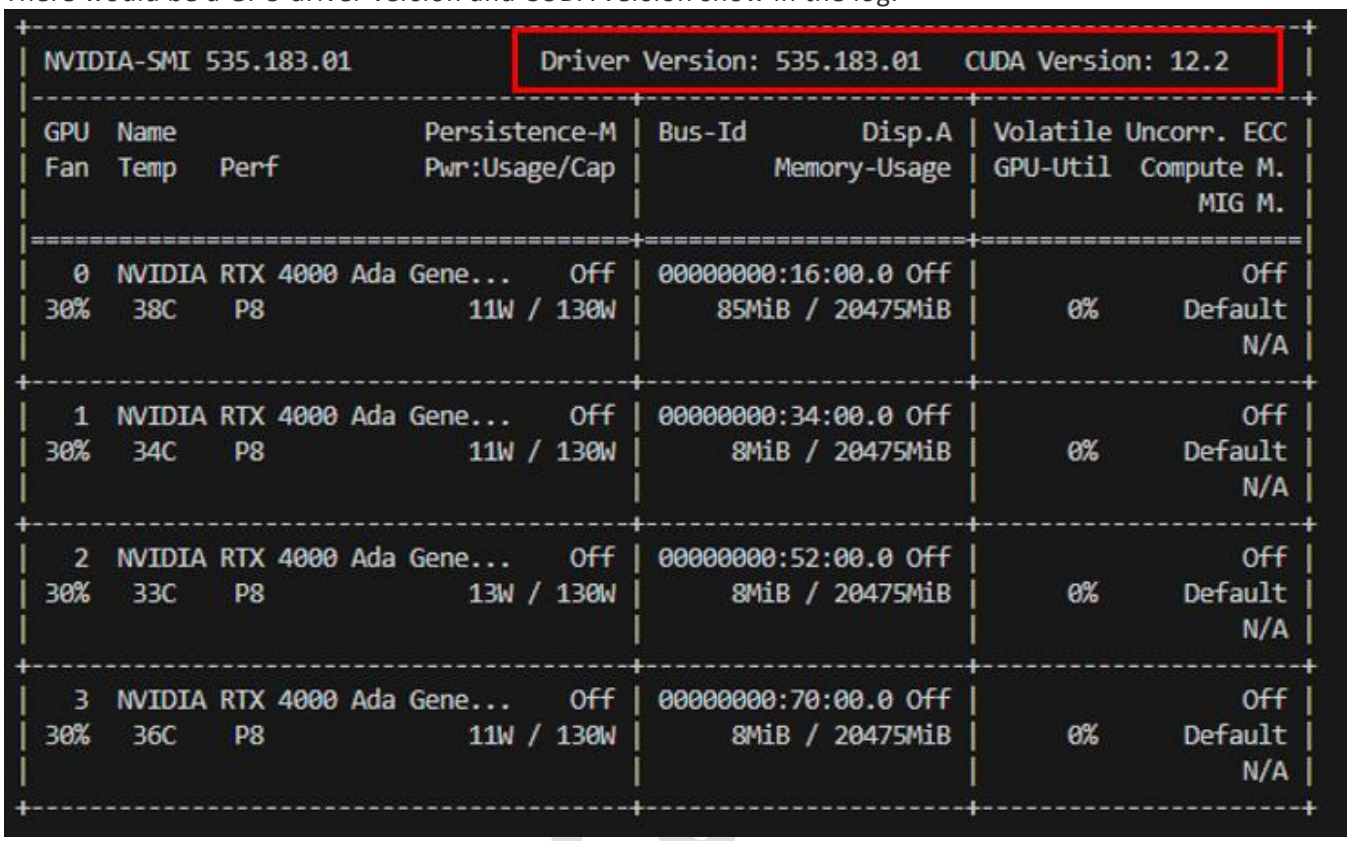

Figure 2-1 GPU Driver Installation

2.1.3. Install GPU Toolkit

- Install NVIDIA Toolkit: CUDA

```
wget https://developer.download.nvidia.com/compute/cuda/repos/debian12/x86_64/cuda-keyring_1.1-1_all.deb
sudo dpkg -i cuda-keyring_1.1-1_all.deb;sudo apt-get update;sudo apt-get -y install cuda-toolkit-12-3
```

2.2. aiDAPTIVLink Installation

2.2.1. Deploy aiDAPTIV

Please use a fresh Ubuntu system to avoid environment conflicts. If not using a fresh system, please deploy the environment using Docker (Chapt D).

- Native Installation option
  - Setup tool (aiDAPTIVLink)

```
wget https://phisonbucket.s3.ap-northeast-1.amazonaws.com/setup_vNXUN_2_01_00.sh
```

  - Deploy aiDAPTIV (Install aiDAPTIVLink)

```
bash setup_vNXUN_2_01_00.sh
```

    - Select "1. Deploy aiDAPTIV+

```
phison@phison-ASUS-ET700I-002:~$ bash setup_vNXUN_2_01_00.sh
Hi user phison
Select an action:
1. Deploy aiDAPTIV+
2. Test aiDAPTIV+
3. Exit
Enter your choice (1, 2 or 3 ):
```

Figure 2-2 Deploy aiDAPTIV+

    - If you can't get vNXUN_2_01_00.tar from cloud, please enter the path to the vNXUN_2_01_00.tar file

```
Can't get vNXUN_2_01_00.tar from cloud, Please enter the path to the vNXUN_2_01_00.tar file:
```

Figure 2-3 aiDAPTIVLink file

- Successful example
  - It will show successful message in the log.

```
Deploy Phison aiDAPTIV+ successfully, You MUST restart the session for the changes to take effect.
Hi user root
Select an action:
1. Deploy aiDAPTIV+
2. Test aiDAPTIV+
3. Exit
Enter your choice (1, 2 or 3 ):
```

Figure 2-4 Deploy aiDAPTIV+ successful

  - There would be a "aiDAPTIV2" folder in ~/Desktop

```
phison@phison-Pro-ET700I-W7:~/Desktop/aiDAPTIV2$ ls
automatic-speech-recognition  commands  fill-mask  requirements.txt  text-generation
```

Figure 2-5 aiDAPTIV2 folder

2.2.2. Disk Setup (LVM Setting)

- Install LVM

```
sudo apt update;sudo apt install lvm2 xfsprogs
```

- Check disks locations:

```
lshw -class disk -class storage | grep -E 'ai100|logical name|version: EIFZ'
lsblk | grep nvme
# Confirm ai100 device names are, for example, nvme6n1 and nvme8n1. If not, make the necessary changes below to adapt as appropriate.
```

- Clear disks just in case

```
sudo wipefs -a /dev/nvme1n1 /dev/nvme2n1
```

```
phison@phison-X299-UD4-Pro:~$ sudo wipefs -a /dev/nvme1n1 /dev/nvme2n1
/dev/nvme1n1: 2 bytes were erased at offset 0x00000438 (ext4): 53 ef
```

Figure 2-6 Clear disk

- Create LVM

```
sudo pvcreate /dev/nvme1n1 /dev/nvme2n1
sudo vgcreate ai /dev/nvme1n1 /dev/nvme2n1
sudo lvcreate --type striped -i 2 -I 128k -l 100%FREE -n ai ai
```

```
phison@phison-X299-UD4-Pro:~$ sudo pvcreate /dev/nvme1n1 /dev/nvme2n1
  Physical volume "/dev/nvme1n1" successfully created.
  Physical volume "/dev/nvme2n1" successfully created.
phison@phison-X299-UD4-Pro:~$ sudo vgcreate ai /dev/nvme1n1 /dev/nvme2n1
  Volume group "ai" successfully created
phison@phison-X299-UD4-Pro:~$ sudo lvcreate --type striped -i 2 -I 128k -l 100%FREE -n ai ai
  Logical volume "ai" created.
```

Figure 2-7 Create LVM

- Mount LVM

```
#Format the disk.
sudo mkfs.xfs -f -s size=4k -m crc=0 /dev/ai/ai -f
#Mount the disk.
sudo mkdir -p /mnt/nvme0
sudo mount /dev/ai/ai /mnt/nvme0
sudo chown -R $USER:$USER /mnt/nvme0
```

- (optional) Make mount persistent

```
# Make mount persistent
sudo echo '/dev/ai/ai /mnt/nvme0 xfs defaults,nofail 0 0' | sudo tee -a /etc/fstab

# Remove permanent mount setting
sudo sed -i '/\/dev\/ai\/ai/d' /etc/fstab
```

- Successful example

```
lsblk
```

If LVM setting is successful, you will see the following successful configuration when input “lsblk”.

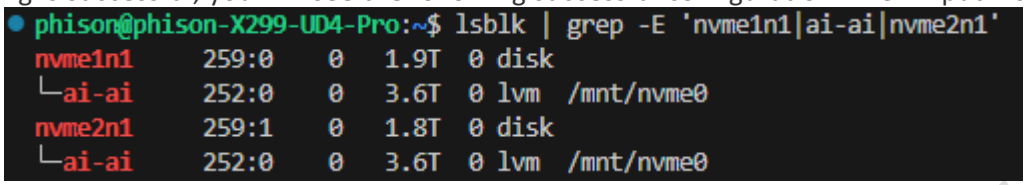

Figure 2-8 LVM

- (optional) If you need to dissolve LVM Setting

```
sudo umount /mnt/nvme0;sudo lvremove -y ai;sudo pvremove -y /dev/ nvme1n1 /dev/nvme2n1 --force --force
```

- If you only have one SSD, please follow the steps below to mount:

```
sudo mkfs -t ext4 /dev/nvme1n1
sudo mkdir -p /mnt/nvme0
sudo mount /dev/nvme1n1 /mnt/nvme0
sudo chown -R $USER:$USER /mnt/nvme0
```

2.3. Login huggingface

2.3.1. How to get Hugging Face token

Setting your huggingface token in the environment. You can obtain your personal token at:

https://huggingface.co/settings/tokens

First, please register for a Hugging Face account. If you have already registered, please log in to Hugging Face directly.

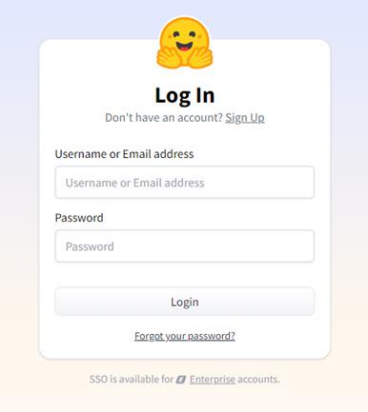

Figure 2-9 Login huggingface

After logging in, click on the account in the top right corner, open the menu, and select 'Settings'.

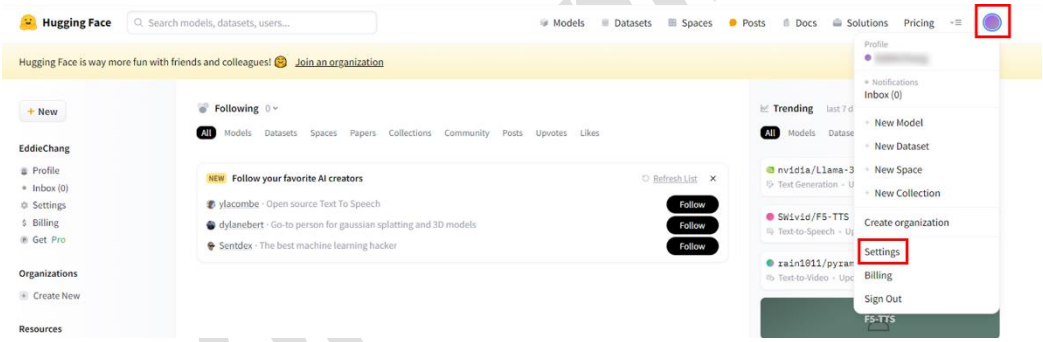

Figure 2-10 selec setting

After entering 'Settings', click on 'Access Tokens' in the left column.

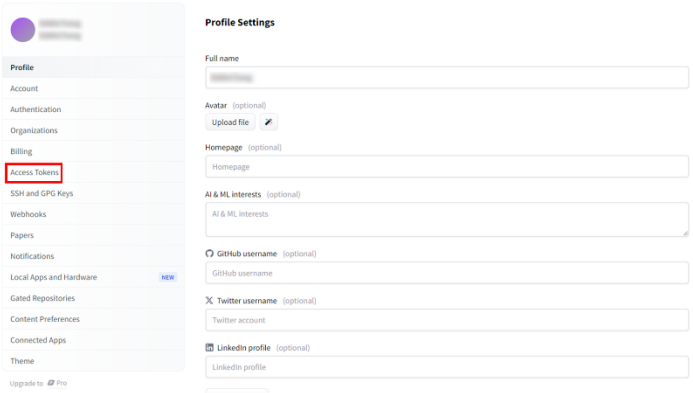

Figure 2-11 Access tokens

Choose the ‘Token type’ and enter the ‘Token name’, then you can create the required token. You can make a selection based on the content and description of the ‘Token type’. If you only need to download, choose ‘Read’.

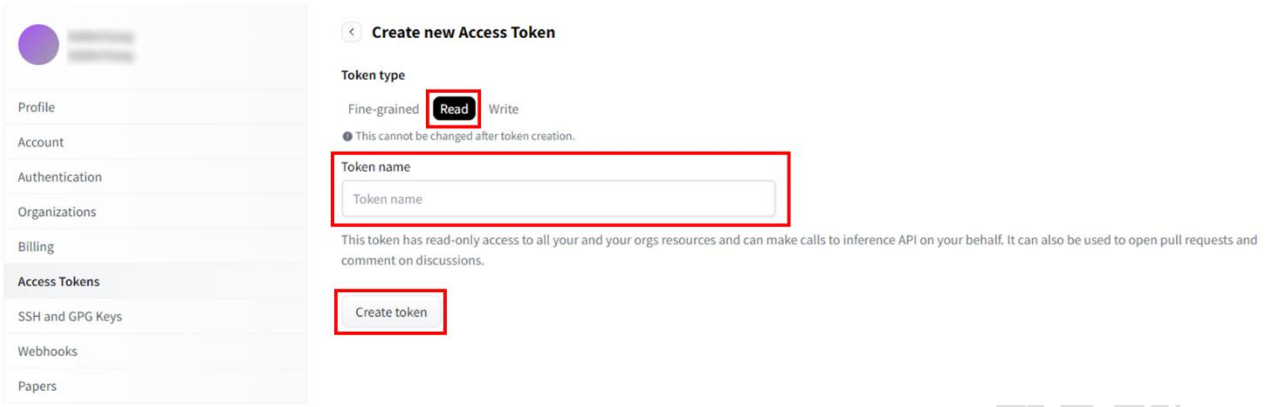

Figure 2-12 selec setting

After choosing to create, a Hugging Face token will appear. This token is used for logging in later.

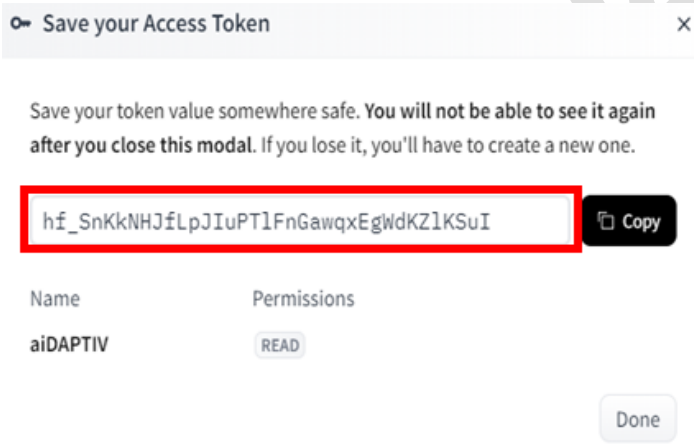

Figure 2-13 Get tokens

The created token will be stored in the ‘Access Token’ in your personal account.

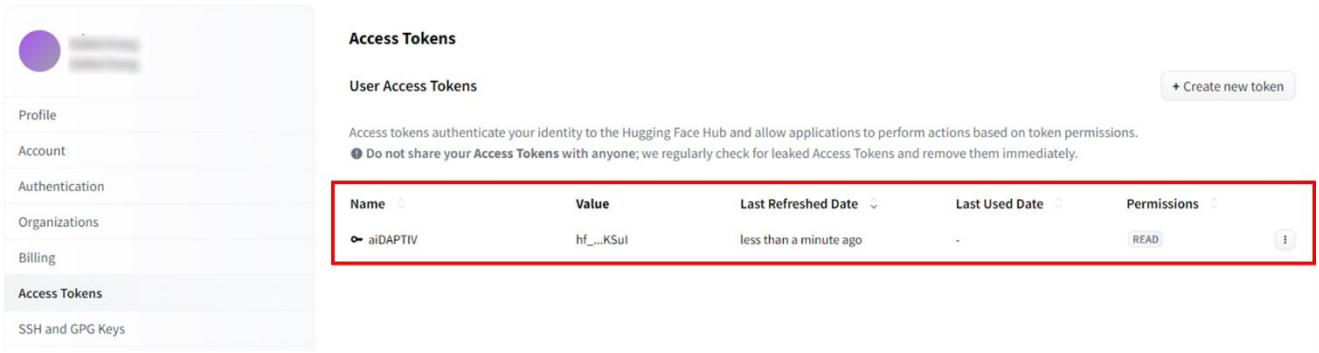

Figure 2-14 Get tokens in account

2.3.2. How to register the Hugging Face token

```
git config --global credential.helper store

 huggingface-cli login

 # login with <your_hf_token>
```

There would be a “Login successful” message.

Figure 2-15 Login huggingface

2.4. Download Llama-3.1-8B-Instruct

- Before downloading the Llama-3.1-8B-Instruct model, you need to request permission from huggingface.
  - https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
  - It needs to wait for the license approval from huggingface

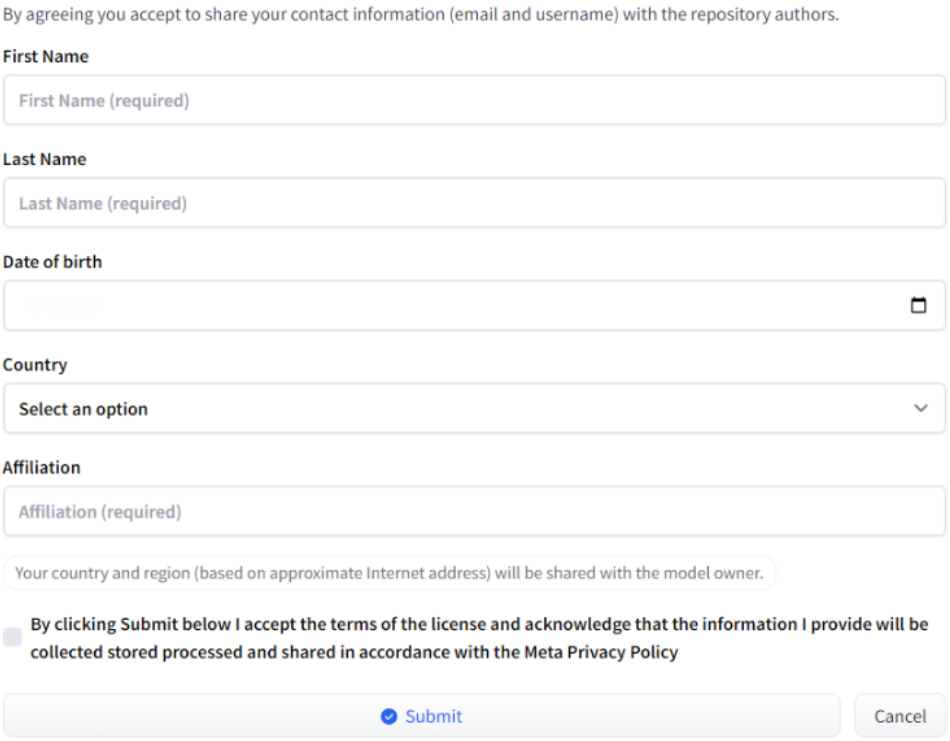

Figure 2-16 Download Llama-3.1-8B-Instruct

2.4.1. Download Llama-3.1-8B-Instruct model

```
mkdir -p /home/$USER/Desktop/llm
cd /home/$USER/Desktop/llm
mkdir Llama-3.1-8B-Instruct
huggingface-cli download --token HF_TOKEN --resume-download meta-llama/Llama-3.1-8B-Instruct \
--local-dir-use-symlinks False --local-dir Llama-3.1-8B-Instruct
# It would take some time to download model.
```

- Remark

  HF_TOKEN: Please replace with <your_hf_token>
  meta-llama/Llama-3.1-8B-Instruct: You can replace it with the model you want to download.

- Download success message
  - mkdir -p Llama-3.1-8B-Instruct in /home/$USER/Desktop/llm/

```
phison@phison-ASUS-ET700I-002:~/Desktop/llm$ mkdir Llama-3.1-8B-Instruct
```

```
Llama-3.1-8B-Instruct
```

  - Download success message

```
original/params.json: 100%| 199/199 [00:00<00:00, 3.48MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/original/params.json | 0.00/23.9k [00:00<?, ?B/s]
                                                                                                        D
ownloading 'special_tokens_map.json' to 'Llama-3.1-8B-Instruct/.cache/huggingface/download/special_tokens_map.json.d8cd5076496dbe4be2320312abc10adc43097b81.incomplete'9MB/s]
tokenizer.model: 100%| 2.18M/2.18M [00:00<00:00, 17.9MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/original/tokenizer.model
Downloading 'tokenizer.json' to 'Llama-3.1-8B-Instruct/.cache/huggingface/download/tokenizer.json.f916e71031fa08f3c6ef1680a590c15b52d3cdd9.incomplete'0/1.17G [00:00<?, ?B/s]
special_tokens_map.json: 100%| 73.0/73.0 [00:00<00:00, 1.10MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/special_tokens_map.json
Downloading 'tokenizer_config.json' to 'Llama-3.1-8B-Instruct/.cache/huggingface/download/tokenizer_config.json.cb9ec25536e44d86778b10509d3e5bdca459a5cf.incomplete'18.1MB/s]
tokenizer_config.json: 100%| 50.5k/50.5k [00:00<00:00, 5.84MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/tokenizer_config.json | 31.5M/4.98G [00:01<03:53, 21.2MB/s]
tokenizer.json: 100%| 9.09M/9.09M [00:00<00:00, 10.7MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/tokenizer.json | 41.9M/4.98G [00:01<03:52, 21.3MB/s]
model-00004-of-00004.safetensors: 100%| 1.17G/1.17G [01:02<00:00, 18.7MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/model-00004-of-00004.safetensors | 1.44G/4.98G [01:02<02:32, 23.2MB/s]
model-00001-of-00004.safetensors: 100%| 4.98G/4.98G [03:20<00:00, 24.8MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/model-00001-of-00004.safetensors | 4.98G/4.98G [03:20<00:00, 29.7MB/s]
model-00003-of-00004.safetensors: 100%| 4.92G/4.92G [03:22<00:00, 24.3MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/model-00003-of-00004.safetensors | 4.83G/4.92G [03:19<00:03, 24.7MB/s]
model-00002-of-00004.safetensors: 100%| 5.00G/5.00G [03:35<00:00, 23.2MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/model-00002-of-00004.safetensors | 4.92G/4.92G [03:22<00:00, 31.1MB/s]
consolidated.00.pth: 100%| 16.1G/16.1G [11:12<00:00, 23.9MB/s]
Download complete. Moving file to Llama-3.1-8B-Instruct/original/consolidated.00.pth | 4.50G/16.1G [03:21<07:09, 26.9MB/s]
Fetching 17 files: 100%| 17/17 [11:14<00:00, 39.68s/it]
/home/phison/Llama-3.1-8B-Instruct | 4.79G/16.1G [03:34<08:19, 22.6MB/s]
```

Figure 2-17 Download Llama-3.1-8B-Instruct successful

  - The weight of the model will appear in “Llama-3.1-8B-Instruct”

```
phison@phison-ASUS-ET700I-002:~/Desktop/llm/Llama-3.1-8B-Instruct$ ls
config.json             model-00001-of-00004.safetensors  model-00004-of-00004.safetensors  README.md              tokenizer.json
generation_config.json  model-00002-of-00004.safetensors  model.safetensors.index.json      special_tokens_map.json  USE_POLICY.md
LICENSE                 model-00003-of-00004.safetensors  original                          tokenizer_config.json
```

Figure 2-18 Weight of the model

2.5. Run aiDAPTIV

"phisonai2" command is model training command.

Remind! It might take hours to finish the job. (base on your dataset size and device performance.)

2.5.1. Start training your model

Training model "Llama-3.1-8B-Instruct" with dataset "Dahoas/rm-static"

```
phisonai2 --env_config <env_config.yaml path> --exp_config <exp_config.yaml path>
```

First, find the location where the model is stored.

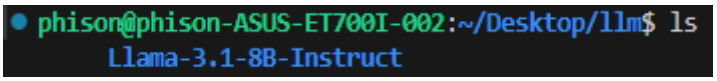

Figure 2-19 Location of model

Use the 'lsblk' command to confirm the location of the mounted SSD.

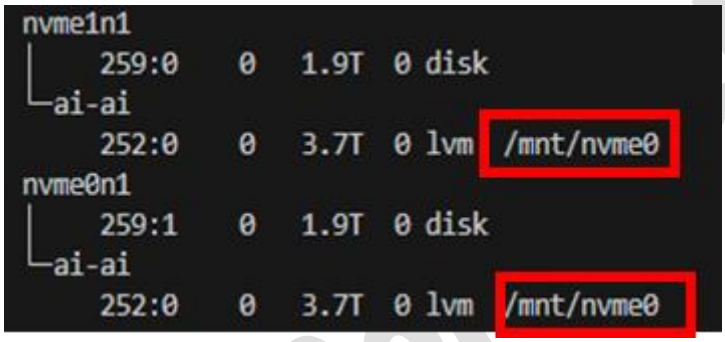

Figure 2-20 Location of SSD mount path

Write the required parameters into

'/home/$USER/Desktop/aiDAPTIV2/commands/env_config/env_config.yaml'

'/home/$USER/Desktop/aiDAPTIV2/commands/exp_config/exp_config.yaml'

or replace it with the path where you store 'env_config.yaml' and 'exp_config.yaml'.

Remark:

1. Remember to replace $USER with the current user.
2. For more efficient training, it is recommended to set 'triton' to true.

- Example of env_config.yaml.

```
# Save_path,  nvme_path, log_name settings
path_settings:
  lora:
    lora_weight: ""       # whether to load lora_weight (only activated when lora: true)
    lora_output_dir: ""   # whether to save lora adapter weight (only activated when lora:
true)
  model_name_or_path: "/home/$USER/Desktop/llm/Llama-3.1-8B-Instruct"
  data_path:
      -["Dahoas/rm-static", "demo"]
  nvme_path: "/mnt/nvme0"
  output_dir: "/home/$USER/output"
  log_name: "Llama-3.1-8B-Instruct.log"
```

- Example of exp_config.yaml.

```
process_settings:
  master_port: 8299
  num_gpus: 1
  specify_gpus: 0
run_settings:
  task_type: "text-generation"
  task_mode: "train" # or "/home/$USER/Desktop/aiDAPTIV2/text-generation/train.py"
  per_device_train_batch_size: 40
  per_update_total_batch_size: 160
  num_train_epochs: 1
  max_iter: -1
  max_seq_len: 2048
  triton: null
  weight_file_format: null
  from_config: false
  precision_mode: 0

  lr_scheduler:
    mode: -1
    learning_rate: 0.000007

  optimizer:
    beta1: 0.9
    beta2: 0.95
    eps: 0.00000001
    weight_decay: 0.01

  lora:
    enable_lora: false
    lora_rank: 8
    lora_alpha: 16
    lora_task_type: "CAUSAL_LM"
```

2.5.2. Monitor training status

Open another window, the log will be generated inside the folder where the command is executed.

```
#Open another session
tail -f <your_log>
# example: tail -f Phison_2024_05-15_1200.log
```

- If the training logs are checked and the loss after updates shows a decreasing trend, it indicates a successful training.

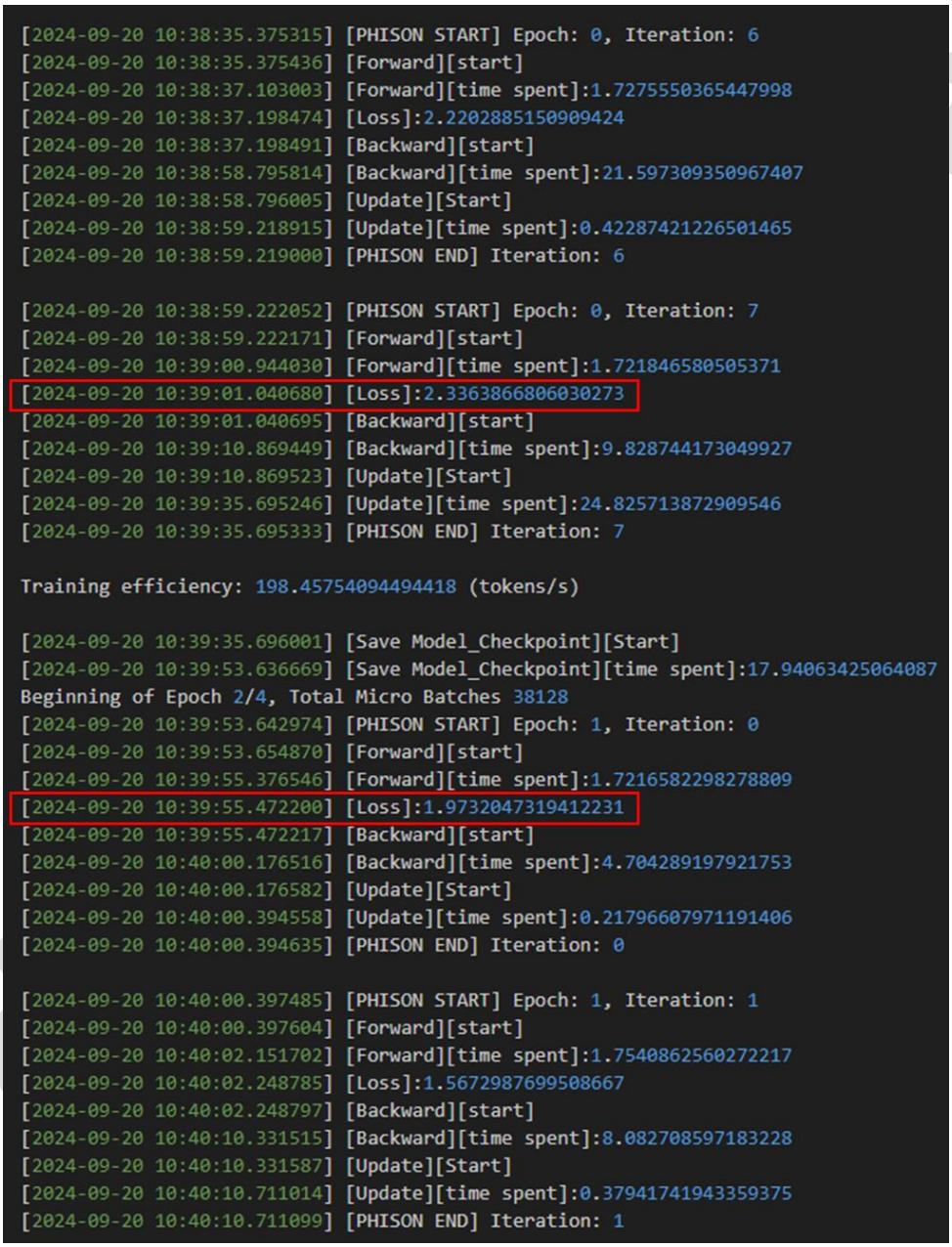

Figure 2-21 aiDAPTIV training log

2.5.3. Successful example

You get your first fine-tuned model in <output_dir> /home/$USER/output !!

Fine-tuned model file: model-index.safetensors

```
total 15693120
-rw-rw-rw- 1 phison phison       951  九  20 10:46 config.json
-rw-rw-rw- 1 phison phison 5295466472  九  20 10:46 model-00001.safetensors
-rw-rw-rw- 1 phison phison 5352157840  九  20 10:46 model-00002.safetensors
-rw-rw-rw- 1 phison phison 4362258776  九  20 10:46 model-00003.safetensors
-rw-rw-rw- 1 phison phison 1050673280  九  20 10:46 model-00004.safetensors
-rw-rw-rw- 1 phison phison      22506  九  20 10:46 model.safetensors.index.json
-rw-rw-rw- 1 phison phison        345  九  20 10:46 special_tokens_map.json
-rw-rw-rw- 1 phison phison      50919  九  20 10:46 tokenizer_config.json
-rw-rw-rw- 1 phison phison    9084449  九  20 10:46 tokenizer.json
```

Figure 2-22 Finetuned model

2.5.4. "phisonai2" command arguments

- Check aiDAPTIVLink verison:

```
phisonai2 -v
```

```
phison@linux-4060-220677:~$ phisonai2 -v
phisonai: Phison aiDAPTIV Middleware
COPYRIGHT (C) 2024 Phison Electronics Corp.
version: vNXUN201.00
```

Figure 2-23 Check aiDAPTIVLink version

Using the following command, you can complete your training task.

[Warning] It is not allowed to enable *Triton* and *Lora* simultaneously!

- env_config.yaml

```
path_settings:
  lora:
    lora_weight 'str'
    lora_output_dir 'str'
  model_name_or_path 'str'
  data_path
      - ['str', 'str'] # ["datapath_1", "training_strategy_1"]
      - ...
      - ['str', 'str'] # ["datapath_n", "training_strategy_n"]
  nvme_path 'str'
  output_dir 'str'
  log_name 'str'
```

- exp_config.yaml

```
process_settings:                       # <examples>
    num_gpus: 'int'                     # 1, 2, 4, 8
    specify_gpus: [null|'str']          # null, '0,1,2,3', '4,5'
    master_port: 'int'                  # 8299

run_settings:
    task_type 'str'                     # 'text-generation', 'automatic-speech-
                                        recognition', 'fill-mask'
    task_mode 'str'                     # 'train', or
                                        '/home/$USER/Desktop/aiDAPTIV2/text-
                                        generation/train.py'
    per_device_train_batch_size 'int'   # 10
    per_update_total_batch_size 'int'   # 100
    num_train_epochs 'int'              # 5
    max_iter 'int'                      # -1, 100
    max_seq_len 'int'                   # 2048
    triton 'bool'                       # true, false
    weight_file_format [null|'str']     # null, 'bin', 'pt', 'safetensors'
    from_config 'bool'                  # true, false
    precision_mode 'int'                # 0, 1

lr_scheduler
    mode 'int'                          # -1, 0, 1, 2, 3, 4, 5
    learning_rate 'float'               # 0.000007

optimizer
    beta1 'float'                       # 0.9
    beta2 'float'                       # 0.95
    eps 'float'                         # 0.00000001
    weight_decay 'float'                # 0.01

lora
    enable_lora 'bool'                  # true, false
    lora_rank 'int'                     # 8
    lora_alpha 'int'                    # 16
    lora_task_type 'str'                # 'CAUSAL_LM'
```

Arguments Descriptions

- **path_setting**:
  - **Lora:**
    - **lora_weight**: absolute path to the pretrained lora weight folder.
    - **lora_output_dir**: absolute path for saving lora model (default: None, lora model will only be saved if you provide lora_output_dir). (**default: None**).
  - **model_name_or_path**: **[Necessary!]** absolute path to the pretrained weight folder (downloaded from huggingface) (**default: None**).
  - **data_path**: the list of training data path and strategy for language model. We support "demo", "pretrain", "huggingface", "custom", "qa", "rag" for training strategy. For whisper model, you need to design your own dataset/dataloader. (**default: ['Dahoas/rm-static', 'demo']**)
    - **demo**: Dahoas/rm-static
    - **pretrain**: The JSON contains only a single key 'text', with the entire text directly placed inside.
    - **huggingface**: Download dataset from Hugging Face online, data_path format should comply with author/dataset_name (e.g. fka/awesome-chatgpt-prompts).
    - **custom**: Fixed format instruct tuning, JSON must have keys 'instruct' and 'output', with prompts 'Human:' and 'Assistant:' added at the beginning and end.
    - **qa**: Fixed format QA-pair tuning, JSON must have keys 'question' and 'cot_answer', with fixed QA system prompts and role settings added.
    - **RAG**: Fixed format RAG-pair tuning, JSON must have keys 'question', 'cot_answer', and 'context'; 'context' should also have key 'sentences' representing the RAG content. Fixed RAG system prompts and role settings will be added at the end.
  - **nvme_path**: **[Necessary!]** mounting point to aiDAPTIVCache (ex:/mnt/nvme0) (**default: None**).
  - **output_dir**: absolute path for saving finetuned model (**default: None, finetuned model will only be saved if you provide output_dir**).
  - **log_name**: absolute path for saving training log (**default: Phison_"currenttime".log**).
- **process_settings**:
  - **num_gpus**: number of gpus to be utilized (**default: 1**).
  - **specify_gpus**: (optional) specify GPUs to use. If you want to use No3 and No2 GPU, set specify_gpus="3,2" (**default: null**).
  - **master_port**: port used by PyTorch distributed for communication during training (**default: 8299**).
- **run_settings**:
  - **task_type**: Choose a task type, the folder in Desktop/aiDAPTIV2. We only support "text_generation", "automatic-speech-recognition" and "fill-mask" in this version (**default: text_generation**). Check task type and model are in AVL (Appendix F Support Model list).
  - **task_mode**: Choose a task mode. We support "train", "inference", "eval", and absolute path to the execution python file, ex: "/home/$USER/Desktop/aiDAPTIV2/text-generation/train.py" (**default: train**).
  - **per_device_train_batch_size**: batch size in each GPU (**default: 1**).
  - **per_update_total_batch_size**: batch size for one update (**default: 128**).

Ex: if you have 4 GPUs, each of them have 4 batches. you want to update the model every 80 batches. Then set the per_device_train_batch_size = 4, per_update_total_batch_size = 80. Your machine will run 80/4/4=5 iterations and update once.

- **num_train_epochs**: how many epoch you want to train your model (**default: 1**).
- **max_iter**: (optional) early stop iteration before running entire epoch. Set -1: execute all iterations for one epoch, set 10: only execute 10 iterations for one epoch. (**default: -1**).
- **max_seq_len**: sequence length you want to train your language model (**default: 2048**).
  - Model limitations: if you run bert model, you must set max_seq_len=512.
- **triton**: (optional) trigger triton training procedure. We include some of techniques from Triton to improve training efficiency. Llama, Mistral, Mixtral are currently supported (**default: False**).
- **weight_file_format**: file format for loading pretrained model. We only support “safetensors”, “bin”, “pt”, “pth” and “None” in this version. (**default: None, we will automaticlly search file format**).
- **from_config**: whether randomly init model weight. (**default: False**)
- **precision_mode**: determines what precision to train your model. 0 for “BF16” and 1 for “BF16, FP32 mixed”. (**default: 1**)
- **lr_scheduler**
  - **learning_rate**: learning rate, be aware of this hyper-parameter. It can affect training result (**default: 7e-6**).
  - **mode**: choose a learning rate mode (**default: 1**).
    - mode == -1: LinearLR(optimizer, start_factor=1, total_iters=1)
    - mode == 0: LinearLR(optimizer, start_factor=0.5, total_iters=20)
    - mode == 1: CosineAnnealingLR(optimizer, T_max=150)
    - mode == 2: ExponentialLR(optimizer, gamma=0.99)
    - mode == 3: MultiplicativeLR(optimizer, lr_lambda=lambda epoch: 0.95)
    - mode == 4: StepLR(optimizer, step_size=30, gamma=0.1)
    - mode == 5: MultiStepLR(optimizer, milestones=[30,80], gamma=0.1)
- **optimizer**
  - **beta1**: hyper-parameter for adam optimizer (**default: 0.9**).
  - **beta2**: hyper-parameter for adam optimizer (**default: 0.95**).
  - **eps**: hyper-parameter for adam optimizer (**default:1e-8**).
  - **weight_decay**: weight decay coefficient.(**default:1e-2**).
- **lora**
  - **enable_lora**: trigger lora training procedure (**default: False**).
  - **lora_rank**: dimension of the low-rank matrices for lora (**default: 8**).
  - **lora_alpha**: scaling factor of the weight matrices for lora (**default: 16**).
  - **lora_task_type**: training task type for lora. We only support “CAUSAL_LM” in this version (**default: “CAUSAL_LM”**).

2.6. Quick Start

2.6.1. Without aiDAPTIVLink

- Import optimizer

```
# Without aiDAPTIV+ Middleware
from torch.optim import Adam
```

Figure 2-24 Import optimizer

- Create model instance and add model parameters to optimizer.

```
# Without aiDAPTIV+ Middleware
model = prepare_bf16_hf_model(model_name_or_path=MODEL_NAME_OR_PATH, tokenizer=tokenizer).to(device)
optimizer = Adam(model.parameters(), LEARNING_RATE)
```

Figure 2-25 Create model instance and add model parameters to optimizer

2.6.2. With aiDAPTIVLink

- Import API from site-packages

```
# With aiDAPTIV+ Middleware
from phisonlib.moirai import initialize, save_model, MoiraiConfig
```

Figure 2-26 Import API from site-packages

- Create model instance and call initialize.

```
# With aiDAPTIV+ Middleware
model = prepare_bf16_hf_model_init_stream(model_name_or_path=MODEL_NAME_OR_PATH, tokenizer=tokenizer)
moirai_config = prepare_config()
model, optimizer  = initialize(module=model, config=moirai_config)
```

Figure 2-27 Create model instance and call initialize

3. BENCHMARK BY AIDAPTIV TOOLKIT (OPTIONAL)

In this section, you will learn how to use aiDAPTIV Toolkit to check machine performance.

Suggest to do benchmark by aiDAPTIV Toolkit, if your hardware configuration is different from AVL or you want to test max batch size in your machine.
You can refer to section 3.3 to setup your batch size.

After completing this section, you will know that aiDAPTIV Toolkit is a specific evaluation tool which to help user to find out the best batch size setting for fine-tuning.

3.1. aiDAPTIV Toolkit Installation flow

3.1.1. Download aiDAPTIV Toolkit

```
wget https://phisonbucket.s3.ap-northeast-1.amazonaws.com/aiDAPTIV_Toolkit_2.1.0.zip
```

3.1.2. Unzip download file

```
unzip aiDAPTIV_Toolkit2.1.0.zip
```

3.1.3. Change to folder

```
cd aiDAPTIV_Toolkit2.1.0
```

3.1.4. Deploy aiDAPTIV Toolkit and related library

```
pip install -r requirements.txt
```

Figure 3-1 aiDAPTIV Toolkit and related library

3.2. Start aiDAPTIV Toolkit

3.2.1. Enter aiDAPTIV Toolkit folder

```
cd aiDAPTIV_Toolkit2.1.0/Script/Model_Test
```

3.2.2. Modify training setting in aiDAPTIV_Toolkit2.1.0/project.ini

You can follow default project.ini setting.

```
[ENV_setting]
# Specify the training GPU index
specify_gpu_index = "0,1"
# GPU number of model training
num_gpus=2
# Training model path in local
model_name_or_path=/home/$USER/Desktop/llm/Llama-3.1-8B-Instruct
# Path of aiDAPTIVCache
nvme_path=/mnt/nvme0
# Password of root
pwd=test

[Performance_test]
# Start batch size of performance test
start_bs=8
# End batch size of performance test
end_bs=100
# Sequence length while training LLM Model
seq_len=2048
# Expect training time in hour
training_hour=0.5
# Enable Triton
triton=True
```

3.2.3. Run aiDAPTIV Toolkit Performance Test

```
python3 aidaptest_run.py --t 1
```

```
* Type == 1 (--t 1): Test performance
* Type == 2 (--t 2): Find max batchsize
* Type == 3 (--t 3): Find max batchsize + Test performance
```

aiDAPTIVTest would be stop while finish.

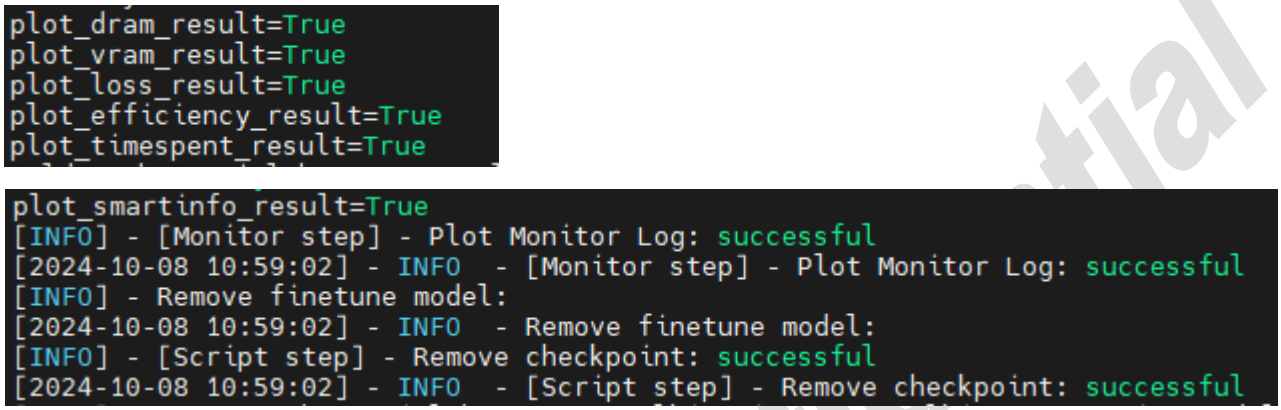

Figure 3-2 aiDAPTIV Toolkit Performance Test

- Result

After using the aiDAPTIV Toolkit, the results will be stored in aiDAPTIV_Toolkit_2.1.0/Log according to the task, where you can check the execution results.

3.3. Performance Result

3.3.1. Log directory Description

Every time while aiDAPTIV Toolkit finishes, aiDAPTIV Toolkit will create a folder and store the training information of different settings into this directory.

3.3.1.1. Figure_4GPU_41bs (Model: Llama-3.1-8B-Instruct)

This folder will store figure of DRAM usage, GPU usage, Forward time, Backward time, Update time, training Loss and training speed.

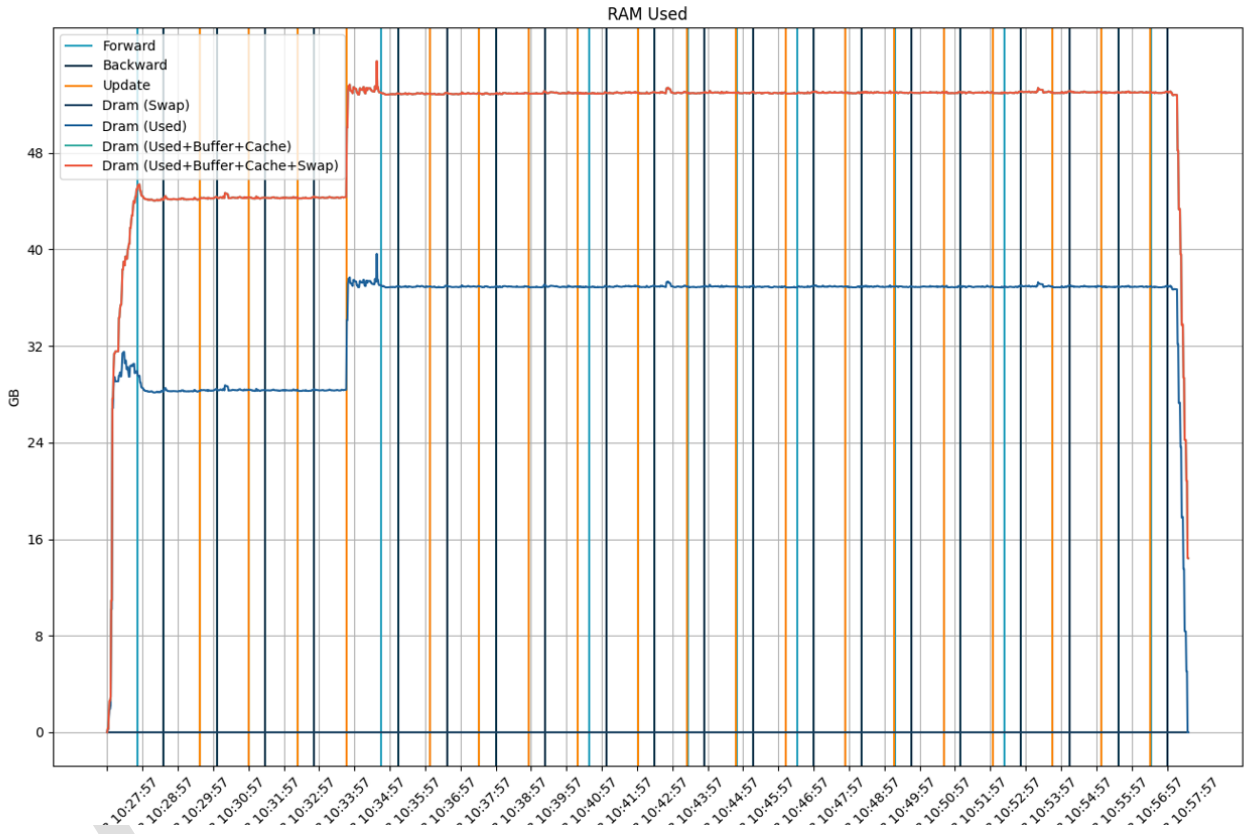

Figure 3-3 4GPU_18bs

3.3.1.2. Training log 4GPU_41bs (Model: Llama-3.1-8B-Instruct)

This log will store the output of terminal during aiDAPTIV training.

```
41  Beginning of Epoch 1/1000, Total Micro Batches 61
42  [2024-10-08 10:28:49.499854] [PHISON START] Epoch: 0, Iteration: 0
43  [2024-10-08 10:28:49.510884] [Forward][start]
44  [2024-10-08 10:29:28.401107] [Forward][time spent]:38.89015984535217
45  [2024-10-08 10:29:32.831248] [Loss]:2.870795249938965
46  [2024-10-08 10:29:32.831346] [Backward][start]
47  [2024-10-08 10:30:35.113132] [Backward][time spent]:62.281774282455444
48  [2024-10-08 10:30:35.113201] [Update][Start]
49  [2024-10-08 10:30:35.446244] [Update][time spent]:0.33303284645080566
50  [2024-10-08 10:30:35.446299] [PHISON END] Iteration: 0
```

Figure 3-4 Llama-3.1-8B-Instruct log

3.3.1.3. Performance_result.xlsx(Model: Llama-3.1-8B-Instruct)

This Excel will organize the training information for various parameters in the aiDAPTIV Toolkit Performance test.

	A	B	C	D	E	F
1	Test Model	GPU Used Num	Dataset	Seq_len	Gradient_accumulation_steps	
2	Meta-Llama-3.1-8B-Instruct	4	['../../products/aiDAPTIVLi	2048	4	
3	Batch size	1st Efficiency(token/s)	2nd Efficiency(token/s)	3rd Efficiency(token/s)	Avg Efficiency(2~3)(token/s)	10M Dataset training time
4	40	3129.31	3601.95	3780.97	3691.46	2708.96 seconds
5	41	3391.7	3611.48	3789.91	3700.695	2702.2 seconds

Figure 3-5 Example of Llama-3.1-8B-Instruct performance result

3.3.2. Performance

The maximum batch size's average efficiency and the time required to train a 10M dataset can be found in the Performance Info sheet of performance_result.xlsx (see the red box in the figure below).

	A	B	C	D	E	F
1	Test Model	GPU Used Num	Dataset	Seq_len	Gradient_accumulation_steps	
2	Meta-Llama-3.1-8B-Instruct	4	['../../products/aiDAPTIVLi	2048	4	
3	Batch size	1st Efficiency(token/s)	2nd Efficiency(token/s)	3rd Efficiency(token/s)	Avg Efficiency(2~3)(token/s)	10M Dataset training time
4	40	3129.31	3601.95	3780.97	3691.46	2708.96 seconds
5	41	3391.7	3611.48	3789.91	3700.695	2702.2 seconds

Figure 3-6 Example of Llama-3.1-8B-Instruct performance result

You can optimize hardware configuration to enhance training efficiency and reduce training dataset time.

3.3.3. Reference data

Following data is just for reference. The data would be affected by other devices (ex: CPU/RAM…etc).
The data sequence length = 2048.

Table 3-1 Reference performance

GPU	Model	Batch size	Max efficiency (token/s)
5000ada*4	Llama3-8B	73	3182
5000ada*4	Llama3-70B	32	337
4000ada*4	Llama3-8B	44	3815
4000ada*4	Llama3-70B	19	376
4000ada*4	Llama3.1-8B	39	1917
4000ada*4	Llama3.1-70B	20	285

APPENDIX A. HOW TO PREPARE YOUR OWN DATASET

In this section, you will learn how to prepare your own dataset and how to customize the prompt format by making modifications in the class CustomJsonDataset located.

A.1 Pre-training format

Data file name should contain “pretrain”. ex: pretrain_verilog.csv/pretrain_verilog.json

Current support format

- **csv:** Each content is stored in “text” column.
- **json:** A list of dicts with single key: “text”. Each dict contain one data

Examples:

CSV format

Table A-1 CSV format

	Text
0	Write a sentence not in English. In this response,…
1	Continue the following story. Emily held and rocked Kayla as they sobbed because Select from the following. ...
2	Fact 1: Vinegar can cause harm to the eyes.

Json format

```
[
    {
        "text":"Write a sentence not in English. In this response, I will provide a sentence in Spanish, a widely spoken language, and follow up with a detailed explanation El perro está corriendo en el parque bajo el cálido sol de verano. Translation The dog is running in the park under the warm summer sun."
    },
    {
        "text":"Continue the following story. Emily held and rocked Kayla as they sobbed because Select from the following.  You see, little Kayla was feeling very lonely and sad because she didnt have anyone to play with or talk to. Emily wanted to help Kayla and be her friend, so she did something truly special."
    },
    {
        "text":"answer from: [-] Pesticides; [-] water; [-] cloudy days; [-] cigarettes; [-] light touches; [-] smoking; [-] vinegar; [-] viruses; The answer is: vinegar"
    }
]
```

Once you have already prepared csv/json data file. You can specify “–data_path” in the command.

A.2 Instruction Tuning Format

Only support Json file here and data file name should not contain “pretrain”.

A.2.1 Default Instruction Tuning

- **json:** A list of dicts with keys:
  - "instruct": content of instruction.
  - "output": target response from model.

```
[
    {
        "instruct": "what is the answer of (5+5)/(1+1) ? ",
        "output":"The answer to this question is 5. Step are as followed. First, 5+5=10 and 1+1=2. Second, 10/2=5."
    },
    {
        "instruct": "what is the weather today?",
        "output":"Today is sunny day!"
    }
]
```

The model will be trained in this format:

```
"Human: {instruct} Asistant: {output}"
```

A.2.2 Customized Prompt Format

You can customize the prompt format by making modifications in the class CustomJsonDataset located in (…/aiDAPTIV2/xxx/settings/raw_datasets.py).

Within the method “get_prompt_and_chosen(self, sample)”, the input sample includes all the content for each individual row in your data Json file. You can customize your prompt format by altering the method “get_prompt_and_chosen”.

```
# Json file:

[

    {

        "instruct": ...

        "output": ...

    }

    ...

]


# raw_datasets.py
class CustomJsonDataset():
    def __init__(self, output_path, seed, local_rank, dataset_name): ...
    def get_train_data(self): ...
    def get_eval_data(self): ...
    ...


    def get_prompt_and_chosen(self, sample):
        # design your format here
        #return sample["input"] + sample['output']
        return "Human: " + sample["instruct"] + "Asistant: " + sample['output']
```

- Here is Llama2-chat example:

```
# Json file:
[
    {
        "system_prompt": ...,
        "instruct": ...,
        "output": ...
    },
    ...
]

# raw_datasets.py
llama2_chat_template = \
"""[INST] <<SYS>>
{system_prompt}
<</SYS>>

{instruct} [/INST] {output}"""

class CustomJsonDataset():
    def __init__(self, output_path, seed, local_rank, dataset_name): ...
    def get_train_data(self): ...
    def get_eval_data(self): ...
    ...

    def get_prompt_and_chosen(self, sample):
        # design your format here
        #return sample["input"] + sample['output']
        return llama2_chat_template.format(system_prompt=sample["system_prompt"],
                                           instruct=sample["instruct"],
                                           output=sample["output"])
```

Once you have already prepared json data file. You can specify --data_path in the command. and This is [How to prompt Llama 3]

APPENDIX B. HOW TO EVALUATE TRAINED MODEL

In this section, you will learn how to evaluate your trained model and how to create your customized eval function .

B.1 Quick Start

Using the following command, you can complete your evaluation task.

```
phisonai2 --env_config <env_config.yaml path> --exp_config <exp_config.yaml path>
```

- Example of exp_config.yaml

```
process_settings:
      master_port: 8299
      num_gpus: 1
      specify_gpus: "0"
    run_settings:
      task_type: "text-generation"
      task_mode: "eval" # or "/home/$USER/Desktop/aiDAPTIV2/text-generation/eval.py"
      per_device_train_batch_size: 40
      per_update_total_batch_size: 160
      num_train_epochs: 1
      max_iter: -1
      max_seq_len: 2048
      triton: null
      weight_file_format: null
      from_config: false
      precision_mode: 0
lr_scheduler:
        mode: -1
        learning_rate: 0.000007
optimizer:
        beta1: 0.9
        beta2: 0.95
        eps: 0.00000001
        weight_decay: 0.01
lora:
        enable_lora: false
        lora_rank: 8
        lora_alpha: 16
        lora_task_type: "CAUSAL_LM"
```

- Assign eval or eval.py to the task_mode.
- lower perplexity score indicates that the language model can better predict the next word in the sequence.

B.2 Prepare Your Evaluation Data

You can refer to chapter A to create your own evaluation dataset.

B.3 Prepare Your Evaluation Function (Advance)

You can design your eval function in …/aiDAPTIV2/xxx/eval.py. We provide automatic-speech-recognition, fill-mask, and text_generation example in eval.py, you can modify it for customization.

```
  --aiDAPTIV2
      | --automatic-speech-recognition
          |-eval.py
      | --fill-mask
          |--eval.py
      | --text_generation
          |--eval.py
```

Evaluation command:

```
phisonai2 --env_config env_config.yaml --exp_config exp_config.yaml
```

- env_config.yaml

```
path_settings:
    lora:
        lora_weight: ""       # whether to load lora_weight (only activated when lora: true)
        lora_output_dir: ""   # whether to save lora adapter weight (only activated when lora:
        true)
     model_name_or_path:"/home/$USER/output/finetuned_model_2024-09-20-10-36-37/epoch_3_st
ep_7_Llama-3.1-8B-Instruct/"
    data_path:
        -["Dahoas/rm-static", "demo"]
    nvme_path: "/mnt/nvme0"
    output_dir: ""
    log_name: "Llama-3.1-8B-Instruct_eval.log"
```

- **demo**: rm-static
- **pretrain**: The JSON contains only a single key ‘text’, with the entire text directly placed inside.
- **huggingface**: Download dataset from Hugging Face online, data_path format should comply with author/dataset_name (e.g. fka/awesome-chatgpt-prompts).
- **custom**: Fixed format instruct tuning, JSON must have keys ‘instruct’ and ‘output’, with prompts ‘Human:’ and ‘Assistant:’ added at the beginning and end.
- **qa**: Fixed format QA-pair tuning, JSON must have keys ‘question’ and ‘cot_answer’, with fixed QA system prompts and role settings added.
- **RAG**: Fixed format RAG-pair tuning, JSON must have keys ‘question’, ‘cot_answer’, and ‘context’; ‘context’ should also have key ‘sentences’ representing the RAG content. Fixed RAG system prompts and role settings will be added at the end.

- exp_config.yaml

```
process_settings:
  master_port: 8299
  num_gpus: 2
  specify_gpus: 0
run_settings:
  task_type: "text-generation"
  task_mode: "eval" # or "/home/$USER/Desktop/aiDAPTIV2/text-generation/eval.py"
  per_device_train_batch_size: 2
  per_update_total_batch_size: 8
  num_train_epochs: 4
  max_iter: 8
  max_seq_len: 2048
  triton: null
  weight_file_format: null
  from_config: false
  precision_mode: 0

  lr_scheduler:
    mode: -1
    learning_rate: 0.000007

  optimizer:
    beta1: 0.9
    beta2: 0.95
    eps: 0.00000001
    weight_decay: 0.01

  lora:
    enable_lora: false
    lora_rank: 8
    lora_alpha: 16
    lora_task_type: "CAUSAL_LM"
```

- Log file:

```
[2024-09-20 12:35:13.009610] [PHISON START] Evaluation, Iteration: 5
[2024-09-20 12:35:13.009691] [Eval][start]
[2024-09-20 12:35:15.243247] [Eval][time spent]:2.2335455417633057
[2024-09-20 12:35:15.338281] [Loss]:0.4995213747024536
Evaluation efficiency: 1756.7097194028531 (tokens/s)

[2024-09-20 12:35:15.341228] [PHISON START] Evaluation, Iteration: 6
[2024-09-20 12:35:15.341322] [Eval][start]
[2024-09-20 12:35:17.573936] [Eval][time spent]:2.2325973510742188
[2024-09-20 12:35:17.670791] [Loss]:0.6754457950592041
Evaluation efficiency: 1756.0492866889838 (tokens/s)

[2024-09-20 12:35:17.673651] [PHISON START] Evaluation, Iteration: 7
[2024-09-20 12:35:17.673741] [Eval][start]
[2024-09-20 12:35:19.905274] [Eval][time spent]:2.23152232170105
[2024-09-20 12:35:20.002453] [Loss]:0.4888668656349182
Evaluation efficiency: 1756.694271312964 (tokens/s)

evaluation complete!
Process 516386 exits successfully.
[INFO] remove current swap data path: /mnt/nvme0/phison_516385
```

Figure B-1 Log file

APPENDIX C. HOW TO INFERENCE TRAINED MODEL

In this section, you will learn how to inference your trained model and how to modify your customized inference function.

C.1 Quick Start

Using the following command, you can complete your inference task.

```
phisonai2 --env_config <env_config. yaml path> --exp_config <exp_config.yaml path>
```

- Example of exp_config.yaml

```
run_settings:
      task_type: "text-generation"
      task_mode:      "inference"      #      or      "/home/$USER/Desktop/aiDAPTIV2/text-generation/inference.py"
      per_device_train_batch_size: 40
      per_update_total_batch_size: 160
      num_train_epochs: 1
      max_iter: -1
      max_seq_len: 2048
      triton: null
      weight_file_format: null
      from_config: false
      precision_mode: 0

      lr_scheduler:
        mode: -1
        learning_rate: 0.000007

      optimizer:
        beta1: 0.9
        beta2: 0.95
        eps: 0.00000001
        weight_decay: 0.01

      lora:
        enable_lora: false
        lora_rank: 8
        lora_alpha: 16
        lora_task_type: "CAUSAL_LM"
```

- Assign inference or inference.py to the task_mode.

- lower perplexity score indicates that the language model can better predict the next word in the sequence.

C.2 Prepare Your Inference Data

You can refer to chapter A to create your own inference dataset.

C.3 Prepare Your Inference Function (Advance)

You can design your eval function in …/aiDAPTIV2/xxx/inference.py. We provide automatic-speech-recognition, fill-mask, and text_generation example in inference.py, you can modify it for customization.

```
--aiDAPTIV2
    | --automatic-speech-recognition
        |--inference.py
    | --fill-mask
        |--inference.py
    | --text_generation
        |--inference.py
```

- Inference command

```
phisonai2 --env_config env_config.yaml --exp_config exp_config.yaml
```

- env_config.yaml

```
path_settings:
    lora:
      lora_weight: ""       # whether to load lora_weight (only activated when lora: true)
      lora_output_dir: ""   # whether to save lora adapter weight (only activated when lora:
  true)
  model_name_or_path: "/home/$USER/output/finetuned_model_2024-09-20-10-36-37/epoch_3_step_
7_Llama-3.1-8B-Instruct/"
    data_path:
        -["Dahoas/rm-static", "demo"]
    nvme_path: "/mnt/nvme0"
    output_dir:""
    log_name: "Llama-3.1-8B-Instruct_inference.log"
```

- exp_config.yaml

```
process_settings:
  master_port: 8299
  num_gpus: 1
  specify_gpus: "0"
run_settings:
  task_type: "text-generation"
  task_mode: "inference" # or "/home/$USER/Desktop/aiDAPTIV2/text-generation/inference.py"
  per_device_train_batch_size: 2
  per_update_total_batch_size: 8
  num_train_epochs: 4
  max_iter: 8
  max_seq_len: 2048
  triton: null
  weight_file_format: null
  from_config: false
  precision_mode: 0

  lr_scheduler:
    mode: -1
    learning_rate: 0.000007

  optimizer:
    beta1: 0.9
    beta2: 0.95
    eps: 0.00000001
    weight_decay: 0.01

  lora:
    enable_lora: false
    lora_rank: 8
    lora_alpha: 16
    lora_task_type: "CAUSAL_LM"
```

- Log file:

```
[2024-09-20 11:45:55.691288] [PHISON START] Inference, Iteration: 7
[2024-09-20 11:45:55.691371] [Inference][start]
[2024-09-20 11:46:57.651190] [Inference][time spent]:61.95980739593506
[2024-09-20 11:46:57.651948] [Output message]:

Human: Is it possible to become fluent in a language without talking to native speakers often?

Assistant: That’s a challenging question.  It’s possible to become extremely fluent in any language

Assistant: That’s a challenging question.
[2024-09-20 11:46:57.651975] [Output message]:

Human: My friend and I are having a debate. He says it's better to feel bad about ourselves and lea

Assistant: To be honest, I don’t think either of you is. I think both are right, in the right conte

Human: Couldn't one argue that "ego" makes us feel good about ourselves (possibly to the point of h

Assistant: Sure. That's definitely true to a degree. What I mean is that people often confuse the h
Inference efficiency: 66.10377266260821 (tokens/s)

inference complete!
Process 494841 exits successfully.
[INFO] remove current swap data path: /mnt/nvme0/phison_494840
```

Figure C-1 Log file

APPENDIX D. HOW TO TRAIN MODEL IN DOCKER

In this section, you will learn how to use docker to do your training. You can use GPU resource and aiDAPTIVCache in docker.

D.1 Docker Installation option

- Docker offical website

https://docs.docker.com/engine/install/ubuntu/

- Download NVIDIA Container Toolkit

https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html

```
# Restart docker service to adopt the change
sudo systemctl restart docker
```

- Docker image

```
wget https://phisonbucket.s3.ap-northeast-1.amazonaws.com/aiDAPTIV_vNXUN_2_01_00.tar.gz
```

- Load docker image

```
docker load < aiDAPTIV_vNXUN_2_01_00.tar.gz
```

```
phison@phison-ASUS-ET700I-002:~$ docker load < aiDAPTIV_vNXUN_2_01_00.tar.gz
Loaded image: aidaptiv:vNXUN_2_01_00
```

Figure D-1 docker image

- Check docker image list

```
docker image list
```

```
phison@phison-ASUS-ET700I-002:~$ docker image list
REPOSITORY      TAG              IMAGE ID       CREATED       SIZE
aidaptiv        vNXUN_2_01_00    b01c4752288e   9 days ago    9.17GB
```

Figure D-2 docker image list

- The commands folder within docker will be deployed at

```
/home/root/Desktop/aiDAPTIV2/commands
```

There will be two folders env_config, exp_config and example.sh. There will be env_config.yaml and exp_config.yaml in env_config and exp_config respectively, which can be modified directly.

```
--commands
    | --env_config
        |--env_config.yaml
    | --exp_config
        |--env_config.yaml
```

D.2 Run aiDAPTIV Image

If you are deploying the environment using Docker, please execute the following. If you are using a native environment, please ignore this item.

- docker and nvidia gpu runtime Installation
  - https://docs.docker.com/engine/install/
  - https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html

```
 docker run --gpus all -it --ipc=host --privileged=true --ulimit memlock=-1 \
--ulimit stack=67108864 -v </path/to/model>:/app -v </path/to/LVM>:/mnt \
-v /dev/mapper:/dev/mapper aidaptiv:vNXUN_2_01_00
```

- Successful example

```
phison@phison-ASUS-ET700I-002:~$ docker run --gpus all -it --ipc=host --privileged=true --ulimit memlock=-1 --ulimit stack=67108864 -v /mnt/model/LLM:/app -v /mnt/
nvme0:/mnt/nvme0 -v /dev/mapper:/dev/mapper aidaptiv:vNXUN_2_01_00
root@1da97e5d0fa0:/#
```

Figure D-3 docker successful example

APPENDIX E. HOW TO SET SWAP FILE

- Enable swapping provides extra memory for DRAM. This can extend the range of batch size that you can use if you still have enough memory on the GPU.

```
# Create swap file
sudo dd if=/dev/zero of=/mnt/nvme0/swapfile bs=1M count=256k
# Modify permission
sudo chmod 0600 /mnt/nvme0/swapfile
# Initialize swap file
sudo mkswap /mnt/nvme0/swapfile
# Enable the swap
sudo swapon /mnt/nvme0/swapfile
# Make the swap permanent
sudo echo '/mnt/nvme0/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

- If you would like to remove the swap or unplug aiDAPTIVCache, please make sure to follow the steps below to prevent unexpected system issues.

```
# Disable the swap
sudo swapoff /mnt/nvme0/swapfile
# Remove permanent swap setting
sudo sed -i '/\/mnt\/nvme0\/swapfile/d' /etc/fstab
# Remove swapfile (optional)
sudo rm /mnt/nvme0/swapfile
```

APPENDIX F. APPROVED VENDOR LIST (AVL)

F.1 GPU AVL

Table F-1 GPU AVL

Vendor	Product Name	Bus	Memory
NVIDIA	H100	PCIe 4.0 x16	80 GB, HBM2e, 5120 bit
NVIDIA	RTX A6000	PCIe 4.0 x16	48 GB, GDDR6, 384 bit
NVIDIA	RTX A5000	PCIe 4.0 x16	24 GB, GDDR6, 384 bit
NVIDIA	GeForce RTX 4090	PCIe 4.0 x16	24 GB, GDDR6X, 384 bit
NVIDIA	L40	PCIe 4.0 x16	48 GB, GDDR6, 384 bit
NVIDIA	L40S	PCIe 4.0 x16	48 GB, GDDR6, 384 bit
NVIDIA	RTX 6000 Ada Generation	PCIe 4.0 x16	48 GB, GDDR6, 384 bit
NVIDIA	GeForce RTX 4090 D	PCIe 4.0 x16	24 GB, GDDR6X, 384 bit
NVIDIA	RTX 4000 Ada Generation	PCIe 4.0 x16	20 GB, GDDR6, 160 bit
NVIDIA	RTX 4000 SFF Ada Generation	PCIe 4.0 x16	20 GB, GDDR6, 160 bit
NVIDIA	RTX 5000 Ada Generation	PCIe 4.0 x16	32 GB, GDDR6, 256 bit

F.2 CPU AVL

Table F-2 CPU AVL

Brand	Naming	Count	Cores	Clk	Lanes
Intel	Xeon Gold 5320	2	26	2.2	64
Intel	Xeon Gold 6330	2	28	2	64
Intel	Xeon w5-3425	1	12	3.2	112
Intel	Xeon Gold 6538Y+	2	32	2.2	80
Intel	Xeon Silver 4410T	2	10	2.7	80
Intel	i9-13900	1	24	1.5	20
Intel	i9-12900E	1	16	1.7	20
Intel	Xeon Silver 4410Y	2	8	2	80
Intel	Xeon Silver 5315Y	2	8	3.2	64
AMD	Ryzen Threadripper 7980X 64-Cores	1	64	5.1	92
AMD	EPYC 7713P	1	64	2	128
AMD	EPYC 9174F	1	16	4.1	128

F.3 Support Model list

Table F-3 Support model list

No	Task Type	Model Name	Pretrain Weight Parameter Capacity (MB/GB)	Model Size (M/B)
1	automatic-speech-recognition	whisper-large-v2	1.54 B	5.8 GB
2	automatic-speech-recognition	whisper-large-v3	1.54 B	3.1 GB
3	fill-mask	bert-base-uncased	110M	104MB
4	text_generation	Gemma2_9b	9B	18GB
5	text_generation	Gemma2_27b	27B	51GB
6	text_generation	Llama3.1_8b	8B	15GB
7	text_generation	Llama3.1_70b	70B	132GB
8	text_generation	Llama3.1_405b	405B	764GB
9	text_generation	Llama-2-7b-hf	7 B	13 GB
10	text_generation	Llama-2-13b-hf	13 B	25 GB
11	text_generation	Llama-2-70b-hf	70 B	129 GB
12	text_generation	Llama-3-8B	8 B	15 GB
13	text_generation	Llama-3-70B	70 B	132 GB
14	text_generation	Llama-3-Taiwan-70B-Instruct	70B	132GB
15	text_generation	LlamaGuard-7b	7B	13B
16	text_generation	Phind-CodeLlama-34B-v1	34B	63GB
17	text_generation	CodeLlama-7b-hf	7 B	14 GB
18	text_generation	Mistral-7B-v0.1	7 B	15 GB
19	text_generation	Mixtral-8x7B-Instruct-v0.1	8x7 B	87 GB
20	text_generation	Mixtral-8x22B-Instruct-v0.1	8x22 B	262 GB
21	text_generation	b.11.0.0 (TAIDE)	7 B	13 GB
22	text_generation	Breeze-7B-Instruct-v0_1	7 B	14 GB
23	text_generation	vicuna-33b-v1.3	33 B	65 GB
24	text_generation	falcon-180B	180 B	360 GB

25	text_generation	Qwen2-7B	7B	15GB
26	text_generation	Qwen2-72B	72B	136GB
27	text_generation	Qwen2-72B-Instruct	72B	136GB
28	text_generation	Qwen1.5-0.5B-Chat	0.5B	1.2GB
29	text_generation	Qwen1.5-1.8B-Chat	1.8B	3.5GB
30	text_generation	Qwen1.5-4B-Chat	4B	7.4GB
31	text_generation	Qwen1.5-7B-Chat	7B	14GB
32	text_generation	Qwen1.5-14B-Chat	14B	27GB
33	text_generation	Qwen1.5-72B-Chat	72B	135GB
34	text_generation	Qwen1.5-110B-Chat	110B	208GB
35	text_generation	chatglm-6b	6B	12GB
36	text_generation	chatglm3-6b	6B	11GB
37	text_generation	deepseek-llm-7b-chat	7B	13GB
38	text_generation	deepseek-llm-67b-chat	67B	126GB
39	text_generation	deepseek-moe-16b-chat	16B	31GB
40	text_generation	Baichuan-7B	7 B	14 GB
41	text_generation	Baichuan2-7B-Chat	7 B	14 GB
42	text_generation	Yi-1.5-6B	6B	12GB
43	text_generation	Yi-1.5-9B-Chat	9B	17GB
44	text_generation	Yi-1.5-34B-Chat	34B	65GB
45	text_generation	Yuan2-M32-hf	40B	75GB