A. Artifact Appendix

A.1 Abstract

The artifact appendix provides the dataset, code, procedure, hyper-parameter settings, etc. needed to reproduce the HyC-LoRA project. The overall training framework is implemented by PyTorch, some of the operators are implemented by OpenAI Triton, and can be run on the Linux operating system with NVIDIA GPU support. This artifact appendix evaluates the HyC-LoRA project's code availability, algorithmic effectiveness, and system performance. Github project: `https://github.com/Ther-nullptr/HyC-LoRA-release`

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

- **Program:** Python 3.10+
- **Models:** Checkpoints from huggingface: llama-2-7b (`https://huggingface.co/meta-llama/Llama-2-7b-hf`), llama-2-13b (`https://huggingface.co/meta-llama/Llama-2-13b-hf`), mistral-7b (`https://huggingface.co/mistralai/Mistral-7B-v0.1`), roberta-base (`https://huggingface.co/FacebookAI/roberta-base`)
- **Dataset:** *Auto download from huggingface*: GSM8K, Wikitext-2, GLUE; *Download use script*: Math10K, SVAMP, mawps, AQuA; *Download from google drive*: redpajama, proof-pile, PG-19. The download path can be seen in Github project.
- **Run-time environment:** Ubuntu 20.04 LTS, CUDA Version 12.0+, PyTorch version 2.0+
- **Hardware:** CPU: x86 architecture; GPU: NVIDIA GPU (Ampere arch. is recommend) with 16GB+ memory.
- **Execution:** See the `README.md` in attachments provided
- **Metrics:** loss, accuracy, memory consumption, throughput
- **Output:** training log, task accuracy/perplexity, checkpoints
- **Experiments:** See **A.5 Experiment workflow, Evaluation and expected result** part.
- **How much disk space required (approximately)?:** 200GB+
- **How much time is needed to prepare workflow (approximately)?:** Download the checkpoints/datasets: 30min; Build the Python environment: 10min.
- **How much time is needed to complete experiments (approximately)?:** Full experiment: about 2-3 days. Demo experiment: about 2h.
- **Publicly available?:** Yes
- **Code licenses (if publicly available)?:** Apache License 2.0
- **Data licenses (if publicly available)?:** MIT license
- **Workflow framework used?:** No

A.3 Description

A.3.1 How delivered

The artifact is distributed as an environment configuration manual. A companion GitHub repository `https://github.com/Ther-nullptr/HyC-LoRA-release` contains:

- Source code
- Detailed descriptions of scripts
- Links of models and datasets

A.3.2 Hardware dependencies

- CPU: x86 architecture. Since the computation in this project mainly relies on the GPU, there is no special requirement for CPU performance. It is worth noting that although some of our experiments were conducted on devices with **arm** architecture (e.g., NVIDIA Jetson Orin series), the open-source repository code is mainly compatible with **x86** architecture.
- GPU: Proved GPUs: **NVIDIA A800 80GB PCIe, NVIDIA RTX 6000 Ada, NVIDIA GeForce RTX 3090**. Other GPUs were not tested, but might work if meeting the following conditions: ❶ HBM: 16GB+ is recommended, more specific minimum memory requirements can be obtained from **Figure 8** in the original paper. ❷: Arch: Ampere or Hopper architecture for bf16/tf32 tensor core support.
- Disk: 200GB+: models (about 70GB), datasets (about 50GB) and rest for generated checkpoints.

A.3.3 Software dependencies

See **Quick Start** part in `README.md` of Github Project.

A.3.4 Datasets

- GSM8K: experiments for **Table 3**
- Wikitext2: experiments for **Table 3**
- Math10K, SVAMP, mawps, AQuA: experiments for **Table 4**
- redpajama, proof-pile, PG-19: experiments for **Table 5**
- GLUE: experiments for **Table 6**

A.4 Installation

See **Quick Start** part in `README.md` of Github Project. One possible organization of the directory tree is as follows:

Listing 1. dir tree

```
.
|-- figures # Thesis illustration
|   |-- main-intra-inter.jpg
|   |-- main-intra-inter.pdf
|-- models # Model structure implementation
|   |-- llama
|   |-- llama_flash_attn
|   |-- mistral
|   |-- roberta
|   |-- utils
|       |-- compute_utils.py
|-- operators # Triton kernels for certain operator
|   |-- compress_function_kernel.py
..   ..
|   |--
    triton_fuse_lora_silu_hadamard_forward_kernels.py
|-- ckpt # Needs to be created on your own, for
    placing models checkpoints (e.g. llama-2-7b)
|-- README.md
|-- requirements.txt # Dependency libraries required
    by the project
|-- download_dataset.sh # Download the datasets for `
    run_multitask.sh`
|-- run_glue.py
|-- run_glue.sh
|-- run_gsm8k.py # main process of gsm8k experiment
|-- run_gsm8k.sh # script for running gsm8k
    experiment
|-- run_longseq.py
|-- run_longseq.sh
|-- run_multitask.py
|-- run_multitask.sh
|-- run_wikitext2.py
|-- run_wikitext2.sh
```

```
|-- utils # utils functions
    |-- accuracy
    |-- glue
    |-- longseq
    |-- math_10k
```

A.5 Experiment workflow, Evaluation and expected result

Full experiments require a lot of arithmetic and time. If the device has less arithmetic or limited time, try *demo experiments*.

(E1) Basic Algorithm Experiments – demo *[5 min human-time and about 10 min compute-time]*

- ***Preparation:*** ❶ Setup the environment following **Quick Start** part in `README.md`. ❷ Download the models on huggingface, e.g., llama-2-7b (`https://huggingface.co/meta-llama/Llama-2-7b-hf`), and put the model dir to a certain position e.g., `ckpt/`. ❸ Change the `model_name` and `model_dir` config in `run_gsm8k.sh` to your own path like:

  ```
  model_name=<your model name>
  model_dir=<your model dir>
  model_name_full=${model_dir}/${model_name}
  ```

- ***Execution:*** Run `bash run_gsm8k.sh`
- ***Results:***
  ❶ After launch, **the configuration of the core parameters** will be printed on the terminal in green color:

  ```
  ********** HycLora Configuration **********
  [INFO] HycLora type: intra_inter
  [INFO] Iteration Threshold: 5
  [INFO] Softmax Outlier Ratio: 0.05
  [INFO] Layernorm Outlier Ratio: 0.005
  [INFO] Quantization Bit: 2
  ******************************************
  ```

  ❷ After a few minutes, a **progress bar** recording the training process and **degrading model loss record** will appear on the terminal, proving that the model is training properly (then the program can be canceled manually):

  ```
  {'loss': 1.0592, 'grad_norm':
      0.158203125, 'learning_rate':
      3.529411764705882e-05, 'epoch':
      0.02}
  {'loss': 0.9622, 'grad_norm': 0.234375,
      'learning_rate': 7.058823529411764e
      -05, 'epoch': 0.04}
  {'loss': 0.682, 'grad_norm': 0.375, '
      learning_rate':
      0.00010588235294117647, 'epoch':
      0.06}
  {'loss': 0.5484, 'grad_norm':
      0.1689453125, 'learning_rate':
      0.00014117647058823528, 'epoch':
      0.09}
    2%|--                                |
        49/2802 [02:48<1:19:14,  1.73s/it]
  ```

- ***Notes:*** The first time you start the programme, it will automatically download the required datasets, which may take a few minutes, so please ensure you have a good internet connection. The program may display the following error if your network encounters an issue. In this case, please check your network connection:

  ```
  Traceback (most recent call last):
    File "/home/abc/HyC-LoRA-release/run_gsm8k.py",
        line 569, in <module>
      train_and_eval()
    ...
    File "/home/abc/anaconda3/envs/hyclora/lib/
        python3.10/site-packages/datasets/load.py",
        line 1780, in dataset_module_factory
      raise ConnectionError(f"Couldn't reach '{path
          }' on the Hub ({type(e).__name__})")
  ConnectionError: Couldn't reach 'gsm8k' on the Hub
      (SSLError)
  ```

(E2) Basic Algorithm Experiments – full *[5 min human-time and about 1.5 h compute-time on A800 / about 3 h compute-time on 3090]*

- ***Preparation:*** Same as **E1**.
- ***Execution:*** Same as **E1**.
- ***Results:***
  ❶ When the model has been trained, the terminal will display a summary of the training phase:

  ```
  {'loss': 0.145, 'grad_norm': 0.296875, '
      learning_rate': 4.8529260605706386e
      -08, 'epoch': 5.95}
  {'loss': 0.1338, 'grad_norm':
      0.27734375, 'learning_rate':
      1.4439004654120956e-08, 'epoch':
      5.97}
  {'loss': 0.1266, 'grad_norm':
      0.30859375, 'learning_rate':
      4.0108971875452144e-10, 'epoch':
      5.99}
  {'train_runtime': 4772.7021, '
      train_samples_per_second': 9.395, '
      train_steps_per_second': 0.587, '
      train_loss': 0.28048270989706653, '
      epoch': 6.0}
  ```

  ❷ After that, the script will step into evaluation stage. Final evaluation result will be displayed on the terminal:

  ```
  prediction [18.0, 3.0, ..., 11.0]
  ground truth [18.0, 3.0, ..., 14.0]
  adapter: None | GSM8K test accuracy:
      29.42% | full precision: False
  ```

  ❸ A `.log` file will be generated in `exp_results_gsm8k/`, recording the configuration of the experiment, the training process and the evaluation process.
- ***Corresponding Content:*** **Table 3: Algorithm performance ... (the same as below)** in original paper.
- ***Notes:*** If you want to complete the flow of the full experiment while saving time, you can modify the `num_train_epochs` config in `run_gsm8k.sh`.

(E3) Reset Configurations – demo/full *[time consumptions same as **E1**]*

- ***Preparation:*** Same as **E1**.
- ***Execution:***
  ❶ Modify the core HyC-LoRA hyper-parameters in `run_gsm8k.sh` script:

```
use_hyclora=True
layer_type=intra_inter
iteration_threshold=5
softmax_outlier_ratio=0.05
layernorm_outlier_ratio=0.005
q_bit=2
```

The exact meaning of the hyper-parameters can be found in the **Configuration Guide** part in `README.md`.

❷ Rerun `bash run_gsm8k.sh`

- ***Result:***
  Same as **E1** and **E2**. The training and evaluation logs change with configuration changes.
- ***Corresponding Content:*** **Table 3: Algorithm performance ... (the same as below)** in original paper.

(E4) Memory Consumption Evaluation *[5 min human-time and about 5 min compute-time]*

- ***Preparation:*** Same as **E1** (for `run_memory_throughput.sh`).
- ***Execution:***
  ❶ Modify the evaluation config and the core HyC-LoRA hyper-parameters in `run_memory_throughput.sh` script:

```
#! evaluation config
# evaluation sequence length
seq_len=512
# evaluation batch size
per_device_train_batch_size=4
# whether to evaluate memory or
    throughput
evaluate_memory=True
evaluate_throughput=False

#! HyCLoRA core parameters
use_hyclora=True
layer_type=intra_inter
iteration_threshold=5
softmax_outlier_ratio=0.05
layernorm_outlier_ratio=0.005
q_bit=2
```

❷ Run `bash run_memory_throughput.sh`

- ***Result:***
  The profiled memory consumption should be printed in the terminal:

```
  0%|                    | 0/2802 [00:00<?,
      ?it/s]
torch.cuda.memory_allocated (static):
    4327.07 MiB
torch.cuda.memory_allocated: 6045.48 MiB
torch.cuda.memory_allocated: 6134.20 MiB
```

  The program will automatically stop after two iterations.
- ***Corresponding Content:*** **Figure 8: Measured Memory...** in original paper.
- ***Notes:*** The "(static)" part means the **static memory allocation** during training, including the weight and optimizer state; The subsequent data represent the **dynamic peak memory allocation** after considering the buffered activation memory consumption. The two above can be subtracted to get the buffered activation memory usage.

(E5) Throughput Evaluation *[5 min human-time and about 10 min compute-time]*

- ***Preparation:*** Same as **E1**.
- ***Execution:*** Modify the evaluation config and the core HyC-LoRA hyper-parameters in `run_memory_throughput.sh` script:

```
# whether to evaluate memory or
    throughput
evaluate_memory=False
evaluate_throughput=True
```

- ***Result:*** The training throughput data can be read from the right side of the progress bar:

```
1%|-   |20/2802[02:06<1:56:26,2.51s/it]
1%|-   |30/2802[02:31<1:55:34,2.50s/it]
1%|--  |40/2802[02:56<1:54:51,2.50s/it]
2%|--  |49/2802[03:18<1:54:23,2.49s/it]
```

  In the original paper's setting, per iteration has 16 sequences (`per_device_train_batch_size=4` and `gradient_accu_steps=4`), so the sequence per second can be calculated as: $\frac{16}{\text{second per iteration}}$. After the data has been read, the program can be canceled manually.
- ***Corresponding Content:*** **Figure 7: Throughput...** in original paper.
- ***Notes:*** Due to the **cold start** and **short calibration phase** of the machine, we strongly recommend reading the throughput data after the operation has stabilized (after about 20-30 iterations).

A.6 Experiment customization

None.

A.7 Notes

None.

A.8 Methodology

Submission, reviewing and badging methodology:

- `http://cTuning.org/ae/submission-20190109.html`
- `http://cTuning.org/ae/reviewing-20190109.html`
- `https://www.acm.org/publications/policies/artifact-review-ba`