# A PERFORMANCE ON ADA LOVELACE GPUS

Tables 6 and 7 present an additional set of end-to-end results on 2 NVIDIA RTX 4090 GPUs. To make the appendix layout more compact in single-column format, we reorganize the table by placing benchmarks on rows and methods on columns, and split AL and TPS into separate lines for readability. Similar to the main results, PRISM consistently achieves the best acceptance length and throughput across all benchmarks and target models. In particular, the advantage of PRISM over Standard, EAGLE-2, and HASS remains clear under different temperatures, showing that the benefit of our parametrically refactoring architecture generalizes beyond datacenter accelerators to commodity GPUs. Although the absolute throughput differs from that on A800 or H800 due to hardware characteristics, the relative ranking of methods remains unchanged, which further strengthens our conclusion.

*Table 6.* LLaMA-2-7B results on 2 NVIDIA RTX 4090 GPUs.

| Benchmark | Temp | Metric | Vanilla | Standard | EAGLE-2 | HASS | PRISM |
|---|---|---|---|---|---|---|---|
| MT-bench | T = 0 | AL | N/A | 2.77 | 4.12 | 4.42 | **4.69** |
| | | TPS | 101.30 | 134.85 | 223.32 | 238.65 | **254.18** |
| | T = 1 | AL | N/A | 2.78 | 4.12 | 4.43 | **4.65** |
| | | TPS | 102.27 | 136.56 | 223.28 | 241.22 | **253.61** |
| HumanEval | T = 0 | AL | N/A | 2.72 | 4.75 | 5.12 | **5.26** |
| | | TPS | 103.57 | 133.32 | 251.94 | 271.59 | **287.77** |
| | T = 1 | AL | N/A | 2.62 | 4.48 | 4.74 | **4.88** |
| | | TPS | 103.27 | 129.76 | 218.69 | 233.54 | **250.55** |
| GSM8K | T = 0 | AL | N/A | 2.88 | 4.26 | 4.51 | **4.68** |
| | | TPS | 104.25 | 140.77 | 222.95 | 238.27 | **256.28** |
| | T = 1 | AL | N/A | 2.85 | 4.24 | 4.50 | **4.63** |
| | | TPS | 102.54 | 138.21 | 205.44 | 218.77 | **234.73** |
| Alpaca | T = 0 | AL | N/A | 2.90 | 4.09 | 4.40 | **4.62** |
| | | TPS | 103.89 | 143.16 | 217.64 | 233.42 | **256.10** |
| | T = 1 | AL | N/A | 2.85 | 3.92 | 4.18 | **4.34** |
| | | TPS | 103.80 | 137.00 | 192.05 | 207.23 | **223.28** |
| CNN/DM | T = 0 | AL | N/A | 2.05 | 3.89 | 4.20 | **4.35** |
| | | TPS | 99.73 | 96.59 | 197.64 | 213.89 | **217.02** |
| | T = 1 | AL | N/A | 2.02 | 3.71 | 4.01 | **4.18** |
| | | TPS | 99.60 | 94.20 | 174.54 | 190.18 | **194.20** |
| Natural Q. | T = 0 | AL | N/A | 2.84 | 3.84 | 4.08 | **4.24** |
| | | TPS | 104.52 | 136.31 | 201.01 | 217.04 | **232.58** |
| | T = 1 | AL | N/A | 2.79 | 3.64 | 3.78 | **3.96** |
| | | TPS | 103.38 | 134.66 | 173.72 | 179.97 | **195.25** |

# B SCALING EXPERIMENTS WITH DIFFERENT SETTINGS

As shown in Figure10, we conduct another scaling experiment with a more realistic tree structure (4-step + 1 bonus-token, 4-branch, 16-token-to-verify). We observe that PRISM also achieves better performance than EAGLE2, HASS, and EAGLE-3, and the performance gap between PRISM and EAGLE-3 does not converge when the training data volume increases. In the figure, PRISM* denotes a PRISM variant that, like EAGLE-3, uses hidden states from three target-model layers.

*Table 7.* LLaMA-3-8B results on 2 NVIDIA RTX 4090 GPUs.

| Benchmark | Temp | Metric | Vanilla | Standard | EAGLE-2 | HASS | PRISM |
|---|---|---|---|---|---|---|---|
| MT-bench | T = 0 | AL | N/A | 5.08 | 3.76 | 4.01 | **4.27** |
| | | TPS | 91.60 | 149.47 | 166.75 | 184.12 | **198.21** |
| | T = 1 | AL | N/A | 5.08 | 3.77 | 4.01 | **4.30** |
| | | TPS | 91.72 | 150.61 | 168.10 | 183.37 | **198.65** |
| HumanEval | T = 0 | AL | N/A | 5.93 | 4.57 | 5.16 | **5.44** |
| | | TPS | 91.51 | 171.76 | 198.19 | 230.42 | **252.18** |
| | T = 1 | AL | N/A | 5.36 | 4.35 | 4.75 | **5.08** |
| | | TPS | 91.02 | 154.60 | 154.91 | 169.27 | **182.53** |
| GSM8K | T = 0 | AL | N/A | 5.75 | 4.26 | 4.74 | **5.06** |
| | | TPS | 91.42 | 166.41 | 186.82 | 212.79 | **235.85** |
| | T = 1 | AL | N/A | 5.31 | 4.02 | 4.45 | **4.71** |
| | | TPS | 90.83 | 152.54 | 142.84 | 157.93 | **170.26** |
| Alpaca | T = 0 | AL | N/A | 4.80 | 3.63 | 3.91 | **4.21** |
| | | TPS | 91.32 | 139.88 | 160.91 | 178.80 | **198.16** |
| | T = 1 | AL | N/A | 4.28 | 3.37 | 3.58 | **3.85** |
| | | TPS | 91.01 | 124.80 | 120.82 | 127.26 | **139.94** |
| CNN/DM | T = 0 | AL | N/A | 4.75 | 3.58 | 3.93 | **4.15** |
| | | TPS | 88.51 | 135.04 | 153.15 | 170.92 | **178.01** |
| | T = 1 | AL | N/A | 4.30 | 3.34 | 3.58 | **3.71** |
| | | TPS | 88.46 | 121.33 | 115.65 | 124.07 | **125.98** |
| Natural Q. | T = 0 | AL | N/A | 4.48 | 3.24 | 3.41 | **3.58** |
| | | TPS | 90.88 | 130.60 | 142.68 | 155.38 | **168.44** |
| | T = 1 | AL | N/A | 3.93 | 0.98 | 3.13 | **3.21** |
| | | TPS | 90.97 | 115.29 | 107.00 | 112.40 | **117.48** |

# C ARTIFACT

## C.1 Abstract

This appendix provides detailed guidance for reproducing the results presented in *PRISM: Parametrically Refactor Inference for Speculative Decoding Models*. The principal numerical results reported in our tables and figures can be replicated using the data and checkpoints we have released.

## C.2 Artifact check-list (meta-information)

- **Algorithm: Prism.**
- **Data sets: MT-Bench, HumanEval, GSM8K, Alpaca, CNN/Daily Mail and Natural Questions.**
- **Run-time environment: Linux and Do not need root access.**
- **Hardware: GPU with $\geq$ 80 GB HBM. Recommend 8 x NVIDIA A100 80GB.**
- **Metrics: Throughput (tokens/s) and acceptance length.**
- **Output: Tables and figures.**
- **How much disk space required (approximately)?: 250 GB.**

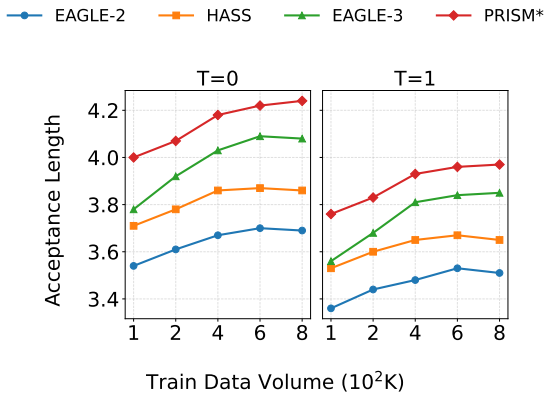


*Figure 10.* Acceptance length under different training data scales and sampling settings.

- **How much time is needed to prepare workflow (approximately)?: Depends on your internet speed; about 150 GB of model files need to be downloaded.**
- **How much time is needed to complete experiments (approximately)?: 8 hours.**
- **Publicly available?: Yes.**
- **Code licenses (if publicly available)?: Creative Commons Attribution 4.0 International**
- **Archived (provide DOI)?: 10.5281/zenodo.18987960.**

## C.3 Description

### *C.3.1 How delivered*

The artifact is available at `https://github.com/Akemiiii/prism-mlsys-ae.git`. It is also archived at `https://doi.org/10.5281/zenodo.18987960`. The corresponding docker image is available at `https://hub.docker.com/r/akemiiii/prism-mlsys26-ae`.

The models are available at `https://www.modelscope.cn/models/Akemiiii/Prism-AE` and `https://huggingface.co/Akem1i/Prism-AE`.

### *C.3.2 Hardware dependencies*

The minimum hardware requirement for running the system is one GPU with at least 80 GB of VRAM and at least 220 GB of available disk space. We recommend a multi-GPU server with 8 NVIDIA A800 80GB GPUs.

### *C.3.3 Software dependencies*

Conda and Python 3.12. The required python packages are detailed in the requirement files.

### *C.3.4 Data sets*

In this work, we conduct experiments across 6 benchmarks (MT-Bench, HumanEval, GSM8K, Alpaca, CNN/Daily, Natural Questions), 5 drafter architectures (Eagle2, Eagle3, HASS, HASS-2, and our proposed PRISM) trained with 5 different data scales, and 2 target models (Llama-2-7B-chat-hf and Llama-3-8B-Instruct).

## C.4 Installation

**Recommended**: Use the provided docker image `akemiiii/prism-mlsys26-ae:latest` to run the experiments. **If you use the docker image, you can skip steps C.4.1–C.4.5 below.**

Please note: if you are using a shared server with other reviewers, please avoid environment conflicts such as container names and SGLang ports. Additionally, we have refined our training procedures since the initial submission, yielding improved checkpoints; as a result, practitioners reproducing our experiments may observe stronger performance for the proposed PRISM model relative to the baselines reported.

### *C.4.1 Clone the repository with:*

```
$ git clone https://github.com/Akemiiii/
    prism-mlsys-ae.git
```

### *C.4.2 Install the PRISM implementation built on top of SGLang:*

```
$ conda create -n prism-sglang python=3.12
    -y
$ conda activate prism-sglang
$ pip install -r sglang/requirements.txt
$ pip install -e "python[all]"
$ cd ../..
```

### *C.4.3 Install another python environment for some experiments with SGLang 0.5.4:*

```
$ conda activate prism-sglang
$ python -m venv sglang054
$ ./sglang054/bin/activate
$ pip install sglang==0.5.4
$ deactivate
```

### *C.4.4 Install python enviroment for Prism implemented on top of pytorch:*

```
$ conda create -n prism python=3.12 -y
$ conda activate prism
$ pip install -r PRISM/requirements.txt
```

#### C.4.5 *Install python environment for eagle3:*

```
$ conda create -n eagle3 python=3.12 -y
$ conda activate eagle3
$ pip install -r Eagle3/requirements_eagle3.
    txt
```

#### C.4.6 *Download the models with:*

```
$ cd evaluation/
$ bash model_download.sh
$ cd ..
```

By default, `model_download.sh` downloads models from ModelScope. If the download speed is too slow, you can use the backup Hugging Face script instead `model_download_hf.sh`.

#### C.4.7 *Use the docker image.*

If you use the docker image to run all the experiments, please follow the steps below:

```
$ docker pull akemiiii/prism-mlsys26-ae:
    latest
$ docker run -it -v ./models:/prism/models
    --gpus all prism-mlsys26-ae:latest
```

### C.5 Evaluation and expected result

Here are step-by-step instructions for reproducing the evaluation results in the paper. You can also refer the `README.md`.

Please note that, due to the difference machine setups, the evaluation results will be different from what shown in the paper. However, the evaluation results should follow similar trend with what we've shown in the paper and support the same claims.

#### C.5.1 *Preparations for Figure 1, 4, 5, 6 (6 hours for 8 x NVIDIA A800 GPUs)*

To reproduce the results in Figure 1, 4, 5, 6. We first evaluate all draft model architectures on all target models, which will generate log files recoding accept length and conditional acceptance ratio. The log files will later be parsed for drawing the figures. There are 8 combinations of target and draft models. For each combination, accept length experiments on 6 benchmarks and 2 different temprature settings are conducted. Each combination on all 12 runs takes around 45 minutes. To conduct the experiment, run

```
$ cd evaluation/prep_figure1456
$ bash eval_acceptance_length.sh --gpus
    0,1,2,3,4,5,6,7
```

Use `--gpus` to assign GPUs. This will take a long time to finish. If you prefer less benchmarks to save your time, pass valid benchmark name (`mt_bench, humaneval, gsm8k, alpaca, sum, qa`) to the script so that experiments for each combination will just run on selected benchmarks. For example:

```
$ bash eval_acceptance_length.sh --gpus
    0,1,2,3,4,5,6,7 --benches humaneval,qa
```

The results are saved in `evaluation/prep_figure1456/outputs`. The log file for each experiment is saved at `evaluation/prep_figure1456/outputs/<TARGET MODEL NAME>/<DRAFT MODEL NAME>/latest/logs/`, where the acceptance length and conditional acceptance rate are reported.

#### C.5.2 *Figure 1 (No extra time)*

```
$ cd evaluation/figure1
$ bash draw_figure1.sh
```

The output image should be stored in `evaluation/figure1/figure1.pdf`. Please compare it with Figure 1 in the paper. Expected results are superior conditional acceptance rates of the PRISM drafter compared to other baselines.

#### C.5.3 *Figure 4 (No extra time)*

```
$ cd evaluation/figure4
$ bash draw_figure4.sh
```

The output image should be stored in `evaluation/figure4/figure4.pdf`. Please compare it with Figure 4 in the paper. We expect to see better scaling of the PRISM drafter.

#### C.5.4 *Figure 5 (No extra time)*

```
$ cd evaluation/figure5
$ bash draw_figure5.sh
```

The output image should be stored in `evaluation/figure5/figure5.pdf`. Please compare it with Figure 5 in the paper. Expected result is a better or at least comparable scaling curve of the PRISM model.

#### C.5.5 *Figure 6 (No extra time)*

```
$ cd evaluation/figure6
$ bash draw_figure6.sh
```

The output image should be stored in `evaluation/figure6/figure6.pdf`. Please compare it with Figure 6 in the paper. Expected result is a better scaling curve of the PRISM model.

#### C.5.6 *Figure 7 (1.5 hours)*

```
$ conda activate prism-sglang
$ cd evaluation/figure7
$ ./run_figure7.sh
```

The output image should be stored in `evaluation/figure7/figure7.pdf`. Please compare it with Figure 7 in the paper.

#### C.5.7 *Figure 8 (1 hour)*

```
$ conda activate prism-sglang
$ cd evaluation/figure8
$ ./run_figure8.sh
```

The output image should be stored in `evaluation/figure8/figure8.pdf`. Please with Figure 8 in the paper.

#### C.5.8 *Table 4 and Table 5 (2 hours for 2 x NVIDIA A800 GPUs)*

On a single machine in this AE setup, you can only reproduce Table 4. To reproduce Table 5, you need another machine with a different GPU type.

Please use the following commands to run the Table 4 experiments. If you have two or more GPUs, pass two GPU IDs (comma-separated). The script will use the first two IDs and run LLaMA-2 and LLaMA-3 in parallel. For example:

```
$ conda activate prism-sglang
$ cd evaluation/table4
$ ./run_table4.sh 0,1
```

If you only have one GPU, pass a single GPU ID. The script will run LLaMA-2 and LLaMA-3 sequentially on the same GPU. For example:

```
$ conda activate prism-sglang
$ cd evaluation/table4
$ ./run_table4.sh 0
```

The output table should be stored in `evaluation/table4/table4.pdf`. Please compare it with Table 4 in the paper.

### C.6 Experiment customization

#### C.6.1 *Figure 1, 4, 5, 6*

You could also go into the code files to conduct a single experiment. For example, if you want to test PRISM on `Llama-3-8B-Instruct` on `MT-bench` and use GPU 0,1, run:

```
$ cd PRISM
$ CUDA_VISIBLE_DEVICES=0,1 PROJECT=Llama
    -3-8B MODEL=PRISM BENCHES=mt_bench bash
     eval_all.sh
```

#### C.6.2 *Figure 7*

If you are interested in experimenting with different batch size settings, you can pass a custom list of values via the `--batch-size` argument when invoking `evaluation/figure7/e2e_llama2_bs_sweep.py` directly. For example:

```
$ python evaluation/figure7/
    e2e_llama2_bs_sweep.py --batch-size 10
    12
```

The default sweep is `2 4 8 16 32`, matching the settings used in the paper. You can also run

```
$ python evaluation/figure7/
    e2e_llama2_bs_sweep.py --help
```

to see all available options. The average and detailed result of this python script will be shown in `evaluation/figure7/e2e_llama2_bs_sweep_avg.csv` and `evaluation/figure7/e2e_llama2_bs_sweep_detail.csv`.

#### C.6.3 *Figure 8*

If you are interested in experimenting with different tree settings, you can pass a custom list of values via the `--sweep-configs` argument when invoking `evaluation/figure8/e2e_llama2_tree_verify_sweep.py` directly. For example:

```
$ python evaluation/figure8/
    e2e_llama2_tree_verify_sweep.py --sweep-
    configs 3,2,6 5,8,32 8,10,64
```

The default tree sweep config is matching the settings used in the paper. You can also run

```
$ python evaluation/figure8/
    e2e_llama2_tree_verify_sweep.py --help
```

to see all available options. The result of this python script will be shown in `evaluation/figure8/e2e_llama2_tree_verify_sweep_avg.csv` and `evaluation/figure8/e2e_llama2_tree_verify_sweep_detail.csv`.