Fine-Tuning Llama 3 on AMD Radeon™ GPUs

Garrett Byrd
Dr. Joe Schoonover
Fluid Numerics

AMD
together we advance_

Disclaimer and Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

What is Fine-Tuning?

What is Fine-Tuning?

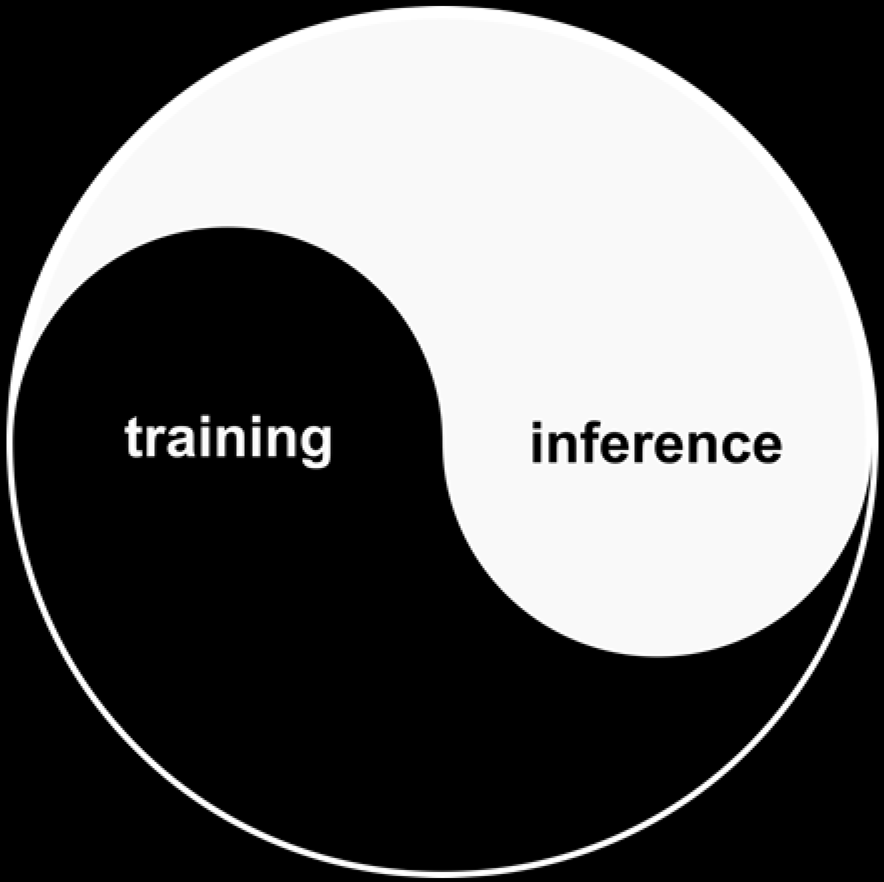
Fine-tuning a large language model (LLM) is the process of increasing a model's performance for a specific task.

E.g.,

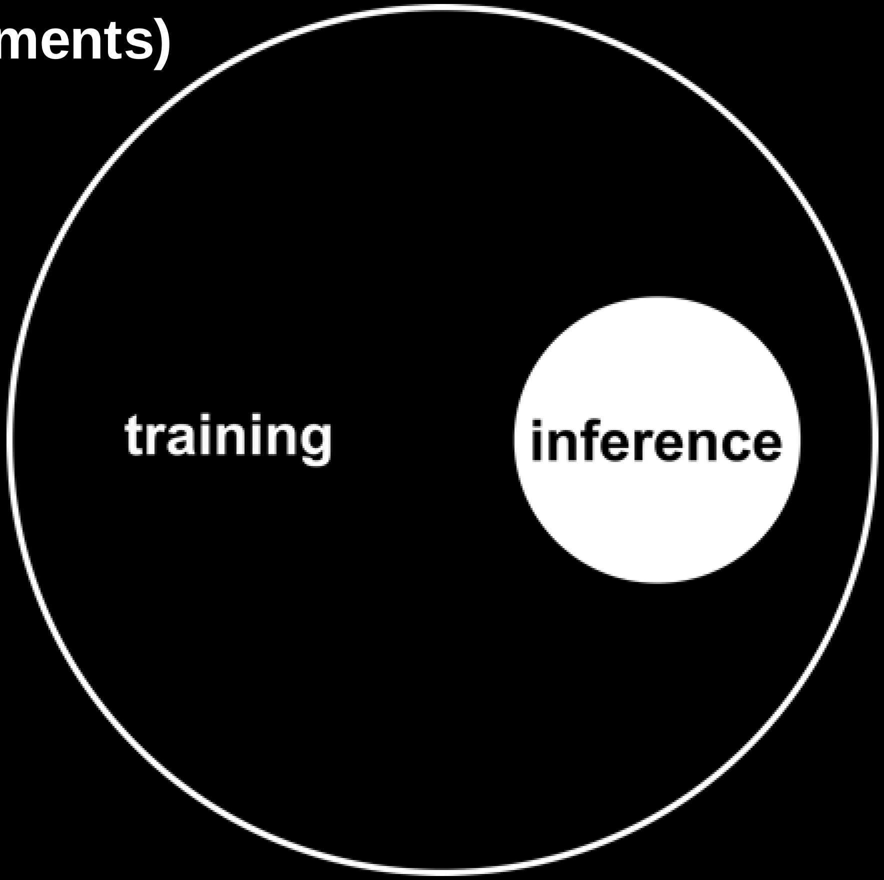
- Specializing a model to generate responses in a question-answer format
- Updating a model's knowledge base for a specific area of application
- Tuning a model to obscure information

Fine-tuning adjusts the parameters of a pretrained model, compared to training a model from scratch.

Two Phases of LLMs



Two Phases of LLMs (memory requirements)



Problem

We want to specialize a model to perform better at a certain task (i.e., fine tune it).

BUT

To do this, we need to retrain the model, which vastly increases the memory requirement.

Solution

Reduce the memory footprint of the (fine-tuning) training process

PEFT and LoRA

Parameter-Efficient Fine-Tuning (PEFT) methods enable efficient adaptation of large pretrained models to various downstream applications by only fine-tuning a small number of (extra) model parameters instead of all the model's parameters. This significantly decreases the computational and storage costs.

https://github.com/huggingface/peft

Low Rank Adaptation (LoRA) is low-rank decomposition method to reduce the number of trainable parameters which speeds up finetuning large models and uses less memory.

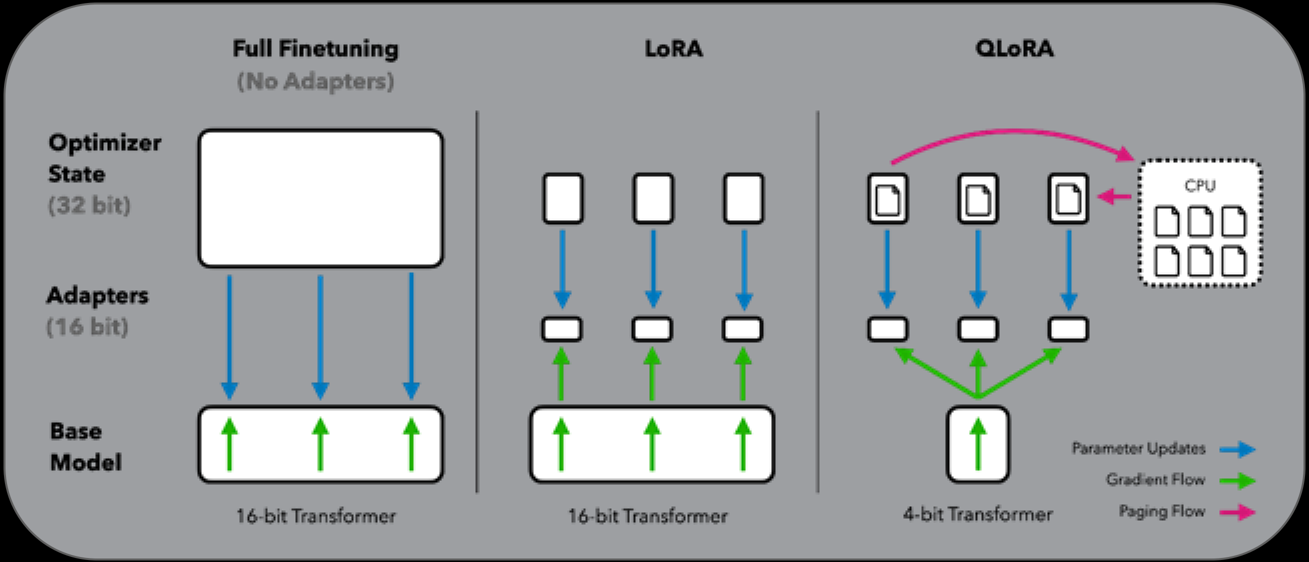
LoRA allows us to train some dense layers in a neural network indirectly by optimizing rank decomposition matrices of the dense layers' change during adaptation instead, while keeping the pre-trained weights frozen.

https://huggingface.co/docs/peft/en/developer_guides/lora

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).

…and QLoRA (Quantized LoRA)

QLoRA utilizes quantization to further reduce the memory footprint of low-rank adaptation methods.



Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. "Qlora: Efficient finetuning of quantized llms." Advances in Neural Information Processing Systems 36 (2024).

What is Quantization?

Quantization

Quantization is the process of discretizing an input from a representation that holds more information to a representation with less information. In the context of LLMs, quantization is used to recast model parameter (often stored with high floating point precision) as lower precision floating point values, or even as integers.

Most often (as in LoRA), we see this as converting LLM parameters stored as 32-bit floats to 8-bit integers.

Quantization Algorithms

AQLM (Additive Quantization of Language Models)

AWQ (Activation-aware Weight Optimization)

bitsandbytes (Block-Wise Quantization)

EETQ (Easy & Efficient Quantization for Transformers)

GPTQ (Accurate Post-Training Quantization)

HQQ (Half-Quadratic Quantization)

FBGEMM FP8 (FaceBook General Matrix Multiplication)

TorchAO (PyTorch Architecture Optimization)

Quantization Algorithms

~~AQLM (Additive Quantization of Language Models)~~

~~AWQ (Activation-aware Weight Optimization)~~

bitsandbytes (Block-Wise Quantization)

~~EETQ (Easy & Efficient Quantization for Transformers)~~

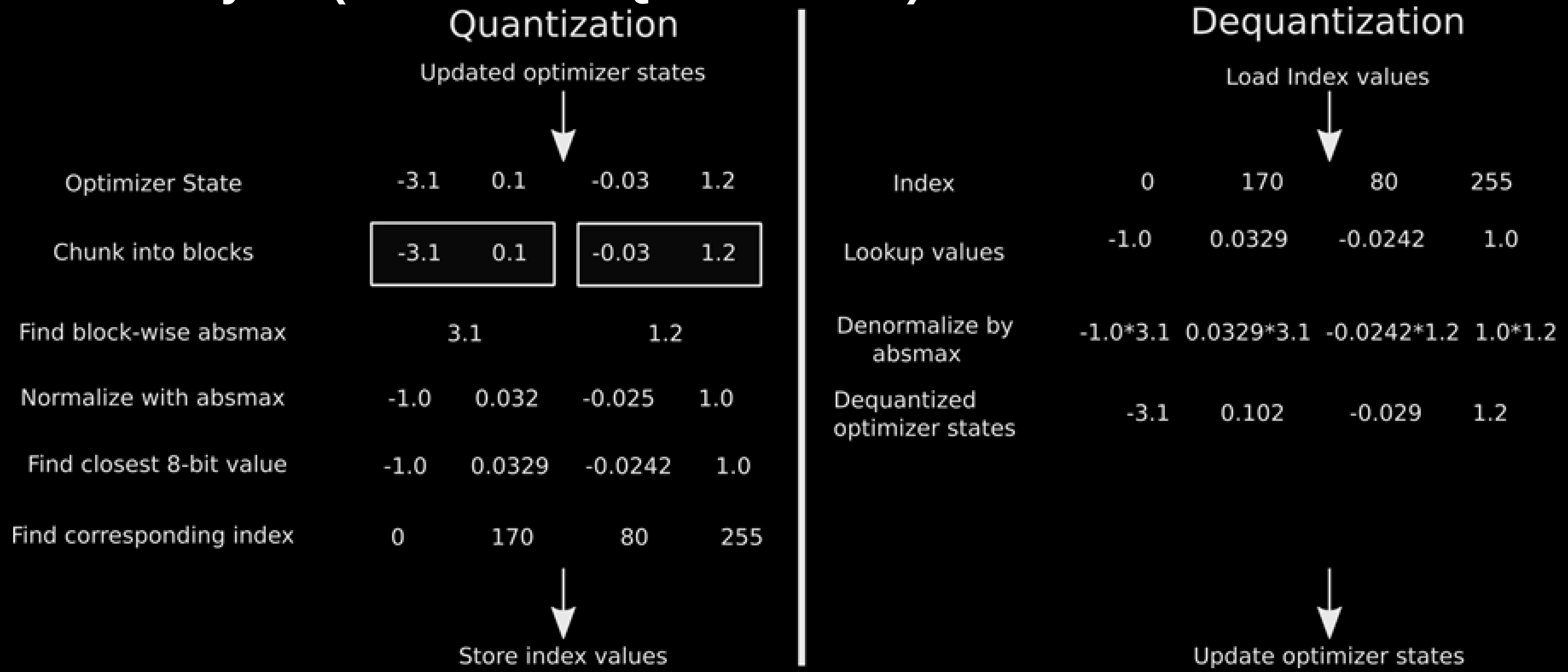
~~GPTQ (Accurate Post-Training Quantization)~~

~~HQQ (Half-Quadratic Quantization)~~

~~FBGEMM FP8 (FaceBook General Matrix Multiplication)~~

~~TorchAO (PyTorch Architecture Optimization)~~

bitsandbytes (Block-Wise Quantization)



Dettmers, Tim, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. "8-bit optimizers via block-wise quantization." arXiv preprint arXiv:2110.02861 (2021).

A grayscale image is just a matrix

Full size, full color (1000x1000)

Resized to 192x192

Grayscale

Gorillaz. "Plastic Beach." Recorded June 2008 – November 2009. Parlophone, compact disc, vinyl LP, streaming.

A "naive" approach (posterization)

In image processing, **posterization** is the process of re-depicting an image using fewer tones. Analogously, in data processing, we can think of this as recasting *n*-bit data (e.g., 32-bit `long int`) to a lower-precision datatype (`uint8_t`).

For a grayscale image using 8-bit color, this can be seen as utilizing 2^n gray tones to depict the image.

Essentially, all values are just being rounded to the nearest value in the target precision.

AMD
together we advance_

Block-Wise Quantization

Llama 3 Embedding Matrix (Example Calculation)

32-bit Single Precision (default):

(vocabulary size) x (length of token embedding) x (floating point precision)

128256 x 4096 x 32 = 2004 MiB ≈ 2 GB

With 8-bit block-wise quantization:

(vocabulary size) x (length of token embedding) x (floating point precision)

+ (number of blocks) x (floating point precision)

(128256 x 4096 x 8) + (128256 x 32) = 1002 MiB ≈ 1 GB

Space saving: 0.5

Model and Dataset

Llama3-8B

- Open source model developed by Meta Platforms, Inc.
- Pretrained with 15 trillion tokens
- 8 billion and 70 billion parameter versions
- Context length of 8K tokens
- High scores on various LLM benchmarks (e.g., MMLU)
- The Llama family has 5 million+ downloads on Hugging Face
- Vocabulary size: 128256
- Embedding token length: 4096

MetaMathQA (Example Data)

query:

If $2a + 1 = 1$ and $b - a = 1$, what is the value of b?

original_question:

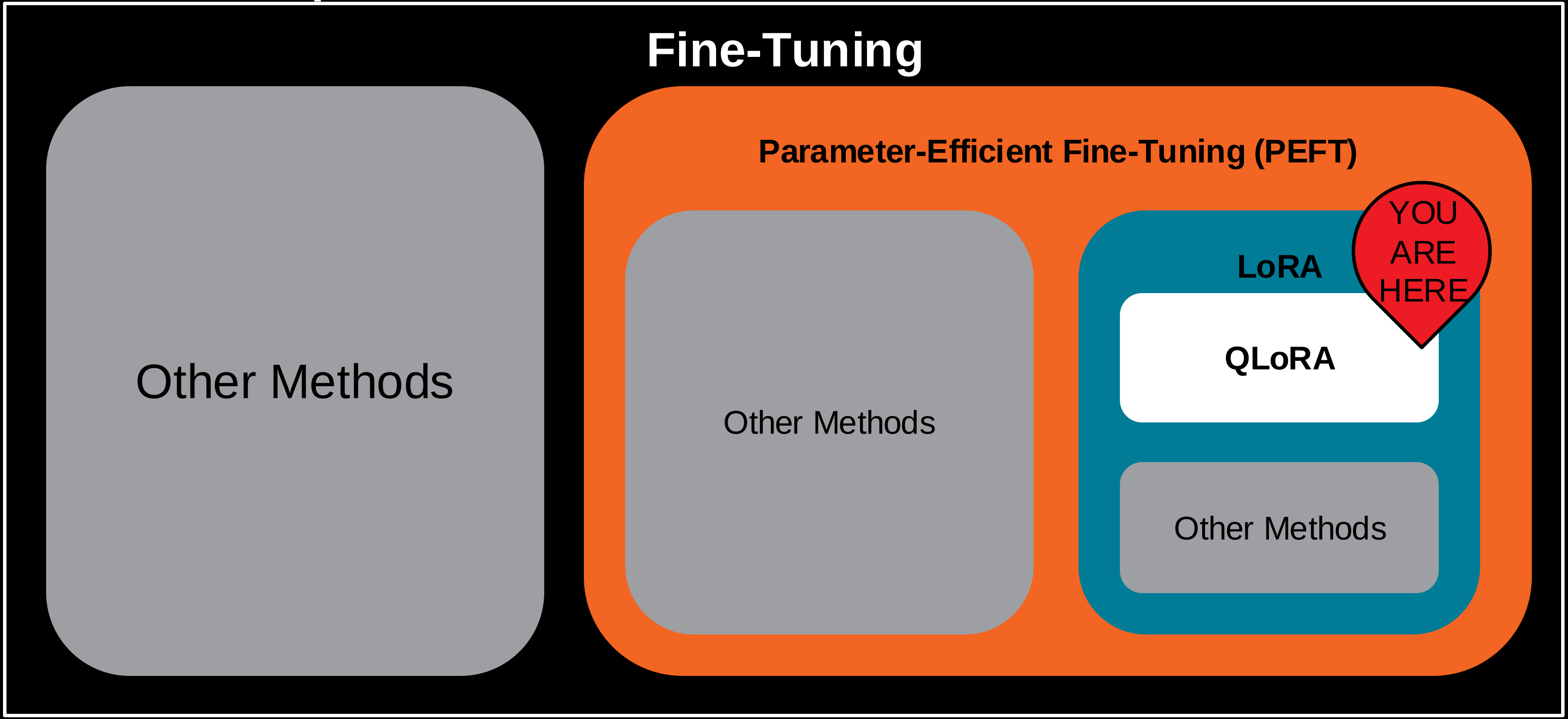
We have that $2a + 1 = 1$ and $b - a = 1.$ What is the value of b?

response:

From the first equation, we have $2a = 0$, so $a = 0$. Substituting this into the second equation, we have $b - 0 = 1$, so $b = \boxed{1}$. The answer is: 1

https://huggingface.co/datasets/meta-math/MetaMathQA *(last accessed September 13 2024)*
https://huggingface.co/meta-llama/Meta-Llama-3-8B *(last accessed September 13 2024)*

The Roadmap



Fine-Tuning
Other Methods
Parameter-Efficient Fine-Tuning (PEFT)
Other Methods
LoRA
YOU ARE HERE
QLoRA
Other Methods

vLLM

- A toolkit and library for large language model inference and serving
- Deploys the PagedAttention algorithm, which reduces memory consumption and increases throughput by leveraging dynamic key and value allocation in GPU memory
- Incorporates many recent LLM acceleration and quantization algorithms
- Easy to get started with Docker

```
docker pull rocm/vllm:rocm6.2_mi300_ubuntu22.04_py3.9_vllm_7c5fd50
```

https://github.com/ROCm/MAD/blob/develop/benchmark/vllm/README.md *(last accessed September 13 2024)*
https://docs.vllm.ai/en/latest/getting_started/amd-installation.html *(last accessed September 13 2024)*
Kwon, Woosuk, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. "Efficient memory management for large language model serving with pagedattention." In Proceedings of the 29th Symposium on Operating Systems Principles, pp. 611-626. 2023.

Coding Example

Follow along:

github.com/FluidNumerics/amd-ml-examples

AMD ROCm™ Platform Blogs

- **Performing natural language processing tasks with LLMs on ROCm running on AMD GPUs** https://rocm.blogs.amd.com/artificial-intelligence/llm-tasks/README.html
- **Optimizing RoBERTa: Fine-Tuning with Mixed Precision on AMD** https://rocm.blogs.amd.com/artificial-intelligence/roberta_amp/README.html
- **Instruction fine-tuning of StarCoder with PEFT on multiple AMD GPUs** https://rocm.blogs.amd.com/artificial-intelligence/starcoder-fine-tune/README.html
- **ResNet for image classification using AMD GPUs** https://rocm.blogs.amd.com/artificial-intelligence/resnet/README.html
- **Enhancing LLM Accessibility: A Deep Dive into QLoRA Through Fine-tuning Llama 2 on a single AMD GPU** https://rocm.blogs.amd.com/artificial-intelligence/llama2-Qlora/README.html
- **Fine-tune Llama 2 with LoRA: Customizing a large language model for question-answering** https://rocm.blogs.amd.com/artificial-intelligence/llama2-lora/README.html

https://rocm.blogs.amd.com/ *(last accessed September 13 2024)*

Find Fluid Numerics Online

www.fluidnumerics.com

r/fluidnumerics

github.com/fluidnumerics

youtube.com/@FluidNumerics

linkedin.com/company/fluidnumerics

Q & A

Endnote(s)

Links to third party sites are provided for convenience and unless explicitly stated, AMD is not responsible for the contents of such linked sites and no endorsement is implied. GD-97.