facebookresearch / **llama-recipes**

Code | Issues 108 | Pull requests 21 | Actions | Projects | Security | Insights

Scripts for fine-tuning Llama2 with composable FSDP & PEFT methods to cover single/multi-node GPUs. Supports default & custom datasets for applications such as summarization & question answering. Supporting a number of candid inference solutions such as HF TGI, VLLM for local or cloud deployment.Demo apps to showcase Llama2 for WhatsApp & Messenger

View license

Code of conduct

Security policy

7.5k stars **1k** forks **70** watching **43** Branches **0** Tags Activity Custom properties

Public repository

main | 43 Branches | 0 Tags | Go to file t | Go to file | + | Add file

mreso last week

.github	refactored PyTest workflow to r...	last week
.vscode	Create finetuning data formatte...	2 months ago
benchmarks	Update README.md	2 weeks ago
demo_apps	Add missing copyright header	last week
docs	update readme	2 weeks ago
eval	Remove copyright header from ...	last week
examples	Add missing copyright header	last week
scripts	Added blacklist to check_copyri...	last week
src/llama_recipes	Add missing copyright header	last week
tests	Make tests run on cpu only ma...	last week
.gitignore	Initial commit	7 months ago
CODE_OF_CONDUCT.md	Initial commit	7 months ago

CONTRIBUTING.md	Added install from source com…	5 months ago
LICENSE	Initial commit	7 months ago
README.md	Merge branch 'main' into ssdp	2 weeks ago
UPDATES.md	Fix broken link in UPDATES.md	5 months ago
USE_POLICY.md	Initial commit	7 months ago
dev_requirements.txt	Added dev_req.txt files	5 months ago
pyproject.toml	remove decapoda-research/lla…	2 months ago
requirements.txt	Merging with main	2 months ago

README Code of conduct License Security

Llama 2 Fine-tuning / Inference Recipes, Examples, Demo Apps

[Update Feb. 5, 2024] We added support for Code Llama 70B instruct in our example inference script. F for Code Llama 70B instruct model please refer to this document.

[Update Dec. 28, 2023] We added support for Llama Guard as a safety checker for our example inferen inference with an example script and prompt formatting. More details here. For details on formatting c provide a script and sample usage here.

[Update Dec 14, 2023] We recently released a series of Llama 2 demo apps here. These apps show how on-prem), how to use Azure Llama 2 API (Model-as-a-Service), how to ask Llama questions in general c live), how to integrate Llama with WhatsApp and Messenger, and how to implement an end-to-end cha Augmented Generation).

The 'llama-recipes' repository is a companion to the Llama 2 model. The goal of this repository is to provic with fine-tuning for domain adaptation and how to run inference for the fine-tuned models. For ease of u converted versions of the models. See steps for conversion of the model here.

In addition, we also provide a number of demo apps, to showcase the Llama 2 usage along with other eco locally, in the cloud, and on-prem.

Llama 2 is a new technology that carries potential risks with use. Testing conducted to date has not — and order to help developers address these risks, we have created the Responsible Use Guide. More details car well. For downloading the models, follow the instructions on Llama 2 repo.

Table of Contents

Quick Start

Llama 2 Jupyter Notebook: This jupyter notebook steps you through how to finetune a Llama 2 model on samsum. The notebook uses parameter efficient finetuning (PEFT) and int8 quantization to finetune a 7B c gpu memory.

Installation

Llama-recipes provides a pip distribution for easy install and usage in other projects. Alternatively, it can b

Install with pip

```
pip install --extra-index-url https://download.pytorch.org/whl/test/cu118 llama-recipes
```

Install with optional dependencies

Llama-recipes offers the installation of optional packages. There are three optional dependency groups. To required dependencies with:

```
pip install --extra-index-url https://download.pytorch.org/whl/test/cu118 llama-recipes[tests]
```

For the vLLM example we need additional requirements that can be installed with:

```
pip install --extra-index-url https://download.pytorch.org/whl/test/cu118 llama-recipes[vllm]
```

To use the sensitive topics safety checker install with:

```
pip install --extra-index-url https://download.pytorch.org/whl/test/cu118 llama-recipes[auditnlg]
```

Optional dependencies can also be combines with [option1,option2].

Install from source

To install from source e.g. for development use these commands. We're using hatchling as our build backe as well as setuptools package.

```
git clone git@github.com:facebookresearch/llama-recipes.git
cd llama-recipes
pip install -U pip setuptools
pip install --extra-index-url https://download.pytorch.org/whl/test/cu118 -e .
```

For development and contributing to llama-recipes please install all optional dependencies:

```
git clone git@github.com:facebookresearch/llama-recipes.git
cd llama-recipes
pip install -U pip setuptools
pip install --extra-index-url https://download.pytorch.org/whl/test/cu118 -e .[tests,auditnlg,vll
```

⚠ **Note** ⚠ Some features (especially fine-tuning with FSDP + PEFT) currently require PyTorch nightlies to install the nightlies if you're using these features following this guide.

Note All the setting defined in config files can be passed as args through CLI when running the script, the files directly.

For more in depth information checkout the following:

- Single GPU Fine-tuning
- Multi-GPU Fine-tuning
- LLM Fine-tuning
- Adding custom datasets
- Inference
- Evaluation Harness
- FAQs

Where to find the models?

You can find Llama 2 models on Hugging Face hub here, where models with `hf` in the name are already c checkpoints so no further conversion is needed. The conversion step below is only for original model weig Hugging Face model hub as well.

Model conversion to Hugging Face

The recipes and notebooks in this folder are using the Llama 2 model definition provided by Hugging Face

Given that the original checkpoint resides under models/7B you can install all requirements and convert th

```
## Install Hugging Face Transformers from source
pip freeze | grep transformers ## verify it is version 4.31.0 or higher

git clone git@github.com:huggingface/transformers.git
cd transformers
pip install protobuf
python src/transformers/models/llama/convert_llama_weights_to_hf.py \
   --input_dir /path/to/downloaded/llama/weights --model_size 7B --output_dir /output/path
```

Fine-tuning

For fine-tuning Llama 2 models for your domain-specific use cases recipes for PEFT, FSDP, PEFT+FSDP hav datasets. For details see LLM Fine-tuning.

Single and Multi GPU Finetune

If you want to dive right into single or multi GPU fine-tuning, run the examples below on a single GPU like parameters in the examples and recipes below need to be further tuned to have desired results based on t hand.

Note:

- To change the dataset in the commands below pass the `dataset` arg. Current options for integrated `alpaca_dataset` and `samsum_dataset`. Additionally, we integrate the OpenAssistant/oasst1 dataset as description of how to use your own dataset and how to add custom datasets can be found in Dataset `alpaca_dataset` please make sure you use the suggested instructions from here to set them up.
- Default dataset and other LORA config has been set to `samsum_dataset`.
- Make sure to set the right path to the model in the training config.
- To save the loss and perplexity metrics for evaluation, enable this by passing `--save_metrics` to the f plotted using the plot_metrics.py script, `python examples/plot_metrics.py --file_path path/to/metri`

Single GPU:

```
#if running on multi-gpu machine
export CUDA_VISIBLE_DEVICES=0

python -m llama_recipes.finetuning  --use_peft --peft_method lora --quantization --model_name /pa
```

Here we make use of Parameter Efficient Methods (PEFT) as described in the next section. To run the comr `peft_method` arg which can be set to `lora`, `llama_adapter` or `prefix`.

Note if you are running on a machine with multiple GPUs please make sure to only make one of them visi `CUDA_VISIBLE_DEVICES=GPU:id`

Make sure you set `save_model` parameter to save the model. Be sure to check the other training param in the config folder as needed. All parameter can be passed as args to the training script. No need to al

Multiple GPUs One Node:

NOTE please make sure to use PyTorch Nightlies for using PEFT+FSDP. Also, note that int8 quantization fro supported in FSDP.

```
torchrun --nnodes 1 --nproc_per_node 4  examples/finetuning.py --enable_fsdp --use_peft --peft_me
```

Here we use FSDP as discussed in the next section which can be used along with PEFT methods. To make u sure to pass `use_peft` and `peft_method` args along with `enable_fsdp`. Here we are using `BF16` for traini

Flash Attention and Xformer Memory Efficient Kernels

Setting `use_fast_kernels` will enable using of Flash Attention or Xformer memory-efficient kernels based would speed up the fine-tuning job. This has been enabled in `optimum` library from Hugging Face as a on

```
torchrun --nnodes 1 --nproc_per_node 4  examples/finetuning.py --enable_fsdp --use_peft --peft_me
```

Fine-tuning using FSDP Only

If you are interested in running full parameter fine-tuning without making use of PEFT methods, please us to change the `nproc_per_node` to your available GPUs. This has been tested with `BF16` on 8xA100, 40GB

```
torchrun --nnodes 1 --nproc_per_node 8  examples/finetuning.py --enable_fsdp --model_name /path_o
```

Fine-tuning using FSDP on 70B Model

If you are interested in running full parameter fine-tuning on the 70B model, you can enable `low_cpu_fsdp`
This option will load model on rank0 only before moving model to devices to construct FSDP. This can dra
loading large models like 70B (on a 8-gpu node, this reduces cpu memory from 2+T to 280G for 70B mod
on 16xA100, 80GB GPUs.

```
torchrun --nnodes 1 --nproc_per_node 8 examples/finetuning.py --enable_fsdp --low_cpu_fsdp --fsdp_
```

In case you are dealing with slower interconnect network between nodes, to reduce the communication o
flag.

HSDP (Hybrid sharding Data Parallel) helps to define a hybrid sharding strategy where you can have FSDP
can be the minimum number of GPUs you can fit your model and DDP between the replicas of the model

This will require to set the Sharding strategy in fsdp config to `ShardingStrategy.HYBRID_SHARD` and specify
`sharding_group_size` and `replica_group_size` where former specifies the sharding group size, number o
form a replica of a model and latter specifies the replica group size, which is world_size/sharding_group_si

```
torchrun --nnodes 4 --nproc_per_node 8 examples/finetuning.py --enable_fsdp --low_cpu_fsdp --fsdp_
```

Multi GPU Multi Node:

```
sbatch multi_node.slurm
# Change the num nodes and GPU per nodes in the script before running.
```

You can read more about our fine-tuning strategies here.

Evaluation Harness

Here, we make use `lm-evaluation-harness` from `EleutherAI` for evaluation of fine-tuned Llama 2 models
other optimizations for inference of Llama 2 model such as quantization. Please use this get started doc.

Demo Apps

This folder contains a series of Llama2-powered apps:

- Quickstart Llama deployments and basic interactions with Llama

1. Llama on your Mac and ask Llama general questions

2. Llama on Google Colab
3. Llama on Cloud and ask Llama questions about unstructured data in a PDF
4. Llama on-prem with vLLM and TGI
5. Llama chatbot with RAG (Retrieval Augmented Generation)
6. Azure Llama 2 API (Model-as-a-Service)

- Specialized Llama use cases:

1. Ask Llama to summarize a video content
2. Ask Llama questions about structured data in a DB
3. Ask Llama questions about live data on the web
4. Build a Llama-enabled WhatsApp chatbot

Benchmarks

This folder contains a series of benchmark scripts for Llama 2 models inference on various backends:

1. On-prem - Popular serving frameworks and containers (i.e. vLLM)
2. (WIP) Cloud API - Popular API services (i.e. Azure Model-as-a-Service)
3. (WIP) On-device - Popular on-device inference solutions on Android and iOS (i.e. mlc-llm, QNN)
4. (WIP) Optimization - Popular optimization solutions for faster inference and quantization (i.e. AutoAW

Repository Organization

This repository is organized in the following way: benchmarks: Contains a series of benchmark scripts for l backends.

configs: Contains the configuration files for PEFT methods, FSDP, Datasets.

docs: Example recipes for single and multi-gpu fine-tuning recipes.

datasets: Contains individual scripts for each dataset to download and process. Note: Use of any of the da the dataset's underlying licenses (including but not limited to non-commercial uses)

demo_apps: Contains a series of Llama2-powered apps, from quickstart deployments to how to ask Llama structured data, live data, and video summary.

examples: Contains examples script for finetuning and inference of the Llama 2 model as well as how to u

inference: Includes modules for inference for the fine-tuned models.

model_checkpointing: Contains FSDP checkpoint handlers.

policies: Contains FSDP scripts to provide different policies, such as mixed precision, transformer wrapping along with any precision optimizer (used for running FSDP with pure bf16 mode).

utils: Utility files for:

- `train_utils.py` provides training/eval loop and more train utils.
- `dataset_utils.py` to get preprocessed datasets.
- `config_utils.py` to override the configs received from CLI.
- `fsdp_utils.py` provides FSDP wrapping policy for PEFT methods.
- `memory_utils.py` context manager to track different memory stats in train loop.

License

See the License file here and Acceptable Use Policy here

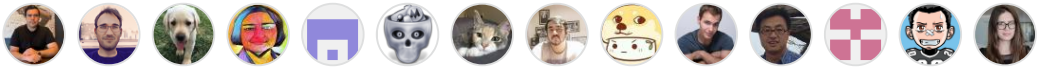
Releases

No releases published

Packages

No packages published

Contributors 46



+ 32 contributors

Languages

Jupyter Notebook 68.2% Python 31.6% Shell 0.2%