lm-sys / **FastChat** Public

Code | Issues 877 | Pull requests 151 | Actions | Security and quality | Insights

main | **33** Branches | **23** Tags | Go to file | Go to file | Code

infwinston Update constants.py (#3733) ✓ 587d5cf · 11 months ago

.github	update .github	3 years ago
assets	Update server_arch.png	3 years ago
data	Update train code to support the new token…	3 years ago
docker	[server] migrate pydantic from v1 to v2 (#33…	2 years ago
docs	Update monitor_md.py (#3494)	2 years ago
fastchat	Update constants.py (#3733)	11 months ago
playground	Code sync (#3546)	2 years ago
scripts	Update the version to 0.2.34 (#2793)	3 years ago
tests	Update load test, use threads instead of as…	2 years ago
.gitignore	Add Multimodal Textbox (#3297)	2 years ago
.pylintrc	[lint] Add code formatter and linter (#511)	3 years ago
LICENSE	Initial commit	3 years ago
README.md	fix doc to make command consistent (#3619)	2 years ago
format.sh	Add scripts for chat data cleaning and anal…	3 years ago
pyproject.toml	p2l stuff (#3660)	last year
theme.json	[WIP] Fixed Leaderboard dropdown bug an…	2 years ago

README | Apache-2.0 license

FastChat

| **Demo** | **Discord** | **X** |

FastChat is an open platform for training, serving, and evaluating large language model based chatbots.

- FastChat powers Chatbot Arena (lmarena.ai), serving over 10 million chat requests for 70+ LLMs.
- Chatbot Arena has collected over 1.5M human votes from side-by-side LLM battles to compile an online LLM Elo leaderboard.

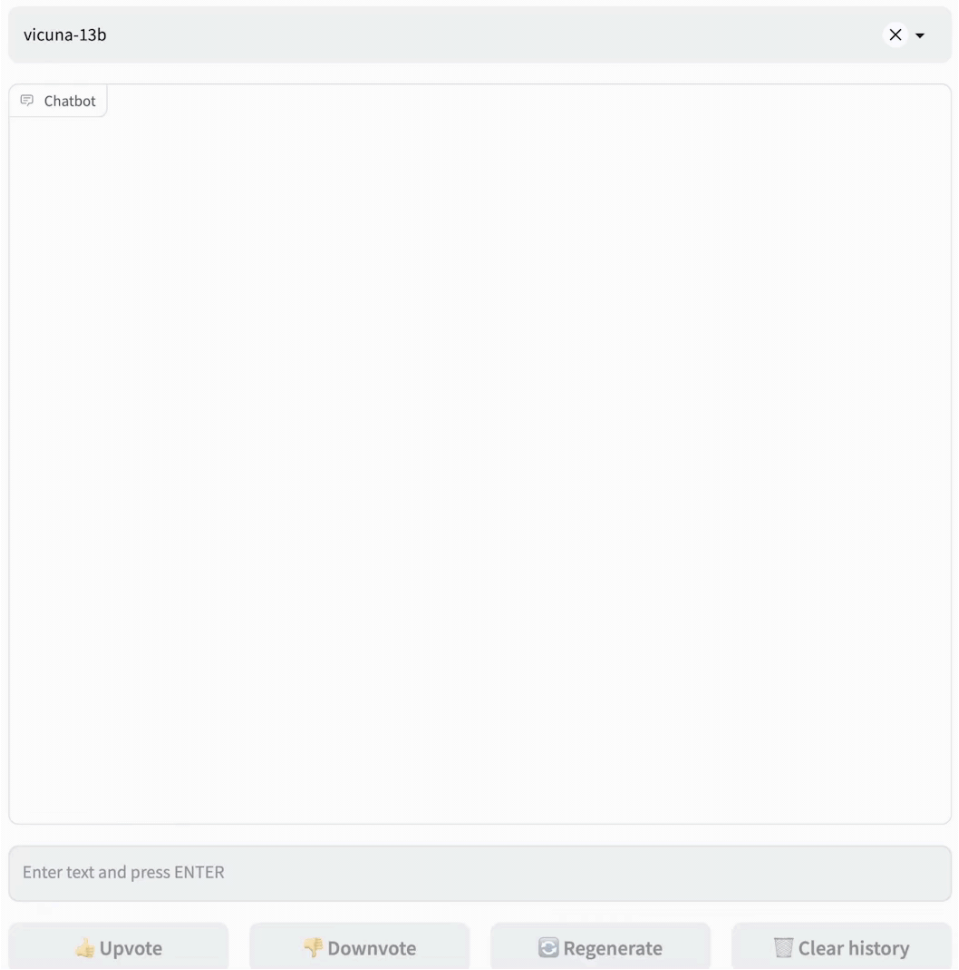
FastChat's core features include:

- The training and evaluation code for state-of-the-art models (e.g., Vicuna, MT-Bench).
- A distributed multi-model serving system with web UI and OpenAI-compatible RESTful APIs.

News

- [2024/03] 🔥 We released Chatbot Arena technical report.
- [2023/09] We released **LMSYS-Chat-1M**, a large-scale real-world LLM conversation dataset. Read the report.
- [2023/08] We released **Vicuna v1.5** based on Llama 2 with 4K and 16K context lengths. Download weights.
- [2023/07] We released **Chatbot Arena Conversations**, a dataset containing 33k conversations with human preferences. Download it here.

▶ More



Contents

Install

Method 1: With pip

```
pip3 install "fschat[model_worker,webui]"
```

Method 2: From source

1. Clone this repository and navigate to the FastChat folder

```
git clone https://github.com/lm-sys/FastChat.git
cd FastChat
```

If you are running on Mac:

```
brew install rust cmake
```

2. Install Package

```
pip3 install --upgrade pip  # enable PEP 660 support
pip3 install -e ".[model_worker,webui]"
```

Model Weights

Vicuna Weights

Vicuna is based on Llama 2 and should be used under Llama's model license.

You can use the commands below to start chatting. It will automatically download the weights from Hugging Face repos. Downloaded weights are stored in a `.cache` folder in the user's home folder (e.g., `~/.cache/huggingface/hub/<model_name>`).

See more command options and how to handle out-of-memory in the "Inference with Command Line Interface" section below.

NOTE: `transformers>=4.31` is required for 16K versions.

Size	Chat Command	Hugging Face Repo
7B	`python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5`	lmsys/vicuna-7b-v1.5
7B-16k	`python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5-16k`	lmsys/vicuna-7b-v1.5-16k
13B	`python3 -m fastchat.serve.cli --model-path lmsys/vicuna-13b-v1.5`	lmsys/vicuna-13b-v1.5
13B-16k	`python3 -m fastchat.serve.cli --model-path lmsys/vicuna-13b-v1.5-16k`	lmsys/vicuna-13b-v1.5-16k
33B	`python3 -m fastchat.serve.cli --model-path lmsys/vicuna-33b-v1.3`	lmsys/vicuna-33b-v1.3

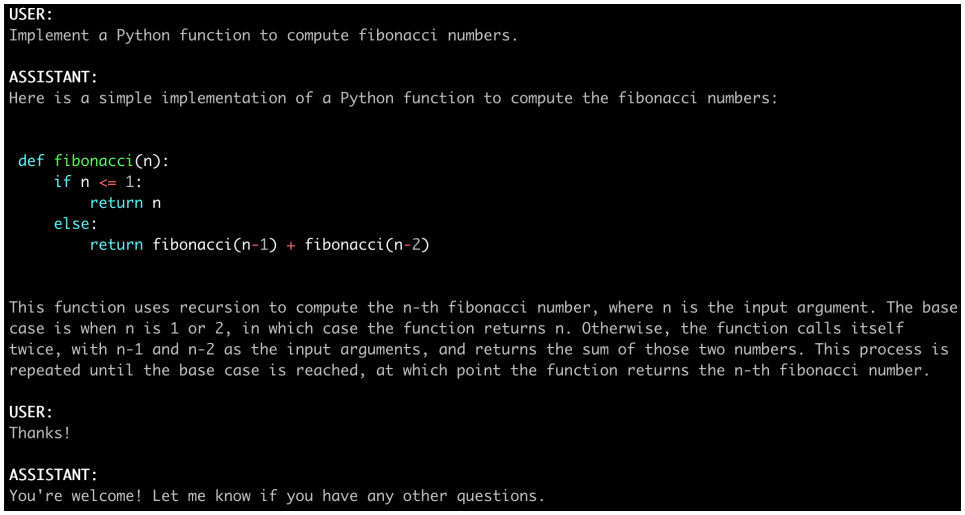
Old weights: see docs/vicuna_weights_version.md for all versions of weights and their differences.

Other Models

Besides Vicuna, we also released two additional models: LongChat and FastChat-T5. You can use the commands below to chat with them. They will automatically download the weights from Hugging Face repos.

Model	Chat Command	Hugging Face Repo
LongChat-7B	`python3 -m fastchat.serve.cli --model-path lmsys/longchat-7b-32k-v1.5`	lmsys/longchat-7b-32k
FastChat-T5-3B	`python3 -m fastchat.serve.cli --model-path lmsys/fastchat-t5-3b-v1.0`	lmsys/fastchat-t5-3b-v1.0

Inference with Command Line Interface



(Experimental Feature: You can specify `--style rich` to enable rich text output and better text streaming quality for some non-ASCII content. This may not work properly on certain terminals.)

Supported Models

FastChat supports a wide range of models, including LLama 2, Vicuna, Alpaca, Baize, ChatGLM, Dolly, Falcon, FastChat-T5, GPT4ALL, Guanaco, MTP, OpenAssistant, OpenChat, RedPajama, StableLM, WizardLM, xDAN-AI and more.

See a complete list of supported models and instructions to add a new model here.

Single GPU

The command below requires around 14GB of GPU memory for Vicuna-7B and 28GB of GPU memory for Vicuna-13B. See the "Not Enough Memory" section below if you do not have enough memory. `--model-path` can be a local folder or a Hugging Face repo name.

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5
```

Multiple GPUs

You can use model parallelism to aggregate GPU memory from multiple GPUs on the same machine.

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --num-gpus 2
```

Tips: Sometimes the "auto" device mapping strategy in huggingface/transformers does not perfectly balance the memory allocation across multiple GPUs. You can use `--max-gpu-memory` to specify the maximum memory per GPU for storing model weights. This allows it to allocate more memory for activations, so you can use longer context lengths or larger batch sizes. For example,

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --num-gpus 2 --max-gpu-memory 8GiB
```

CPU Only

This runs on the CPU only and does not require GPU. It requires around 30GB of CPU memory for Vicuna-7B and around 60GB of CPU memory for Vicuna-13B.

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --device cpu
```

Use Intel AI Accelerator AVX512_BF16/AMX to accelerate CPU inference.

```
CPU_ISA=amx python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --device cpu
```

Metal Backend (Mac Computers with Apple Silicon or AMD GPUs)

Use `--device mps` to enable GPU acceleration on Mac computers (requires torch >= 2.0). Use `--load-8bit` to turn on 8-bit compression.

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --device mps --load-8bit
```

Vicuna-7B can run on a 32GB M1 Macbook with 1 - 2 words / second.

Intel XPU (Intel Data Center and Arc A-Series GPUs)

Install the Intel Extension for PyTorch. Set the OneAPI environment variables:

```
source /opt/intel/oneapi/setvars.sh
```

Use `--device xpu` to enable XPU/GPU acceleration.

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --device xpu
```

Vicuna-7B can run on an Intel Arc A770 16GB.

Ascend NPU

Install the Ascend PyTorch Adapter. Set the CANN environment variables:

```
source /usr/local/Ascend/ascend-toolkit/set_env.sh
```

Use `--device npu` to enable NPU acceleration.

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --device npu
```

Vicuna-7B/13B can run on an Ascend NPU.

Not Enough Memory

If you do not have enough memory, you can enable 8-bit compression by adding `--load-8bit` to commands above. This can reduce memory usage by around half with slightly degraded model quality. It is compatible with the CPU, GPU, and Metal backend.

Vicuna-13B with 8-bit compression can run on a single GPU with 16 GB of VRAM, like an Nvidia RTX 3090, RTX 4080, T4, V100 (16GB), or an AMD RX 6800 XT.

```
python3 -m fastchat.serve.cli --model-path lmsys/vicuna-7b-v1.5 --load-8bit
```

In addition to that, you can add `--cpu-offloading` to commands above to offload weights that don't fit on your GPU onto the CPU memory. This requires 8-bit compression to be enabled and the bitsandbytes package to be installed, which is only available on linux operating systems.

More Platforms and Quantization

- For AMD GPU users, please install ROCm and the ROCm version of PyTorch before you install FastChat. See also this post.
- FastChat supports ExLlama V2. See docs/exllama_v2.md.
- FastChat supports GPTQ 4bit inference with GPTQ-for-LLaMa. See docs/gptq.md.

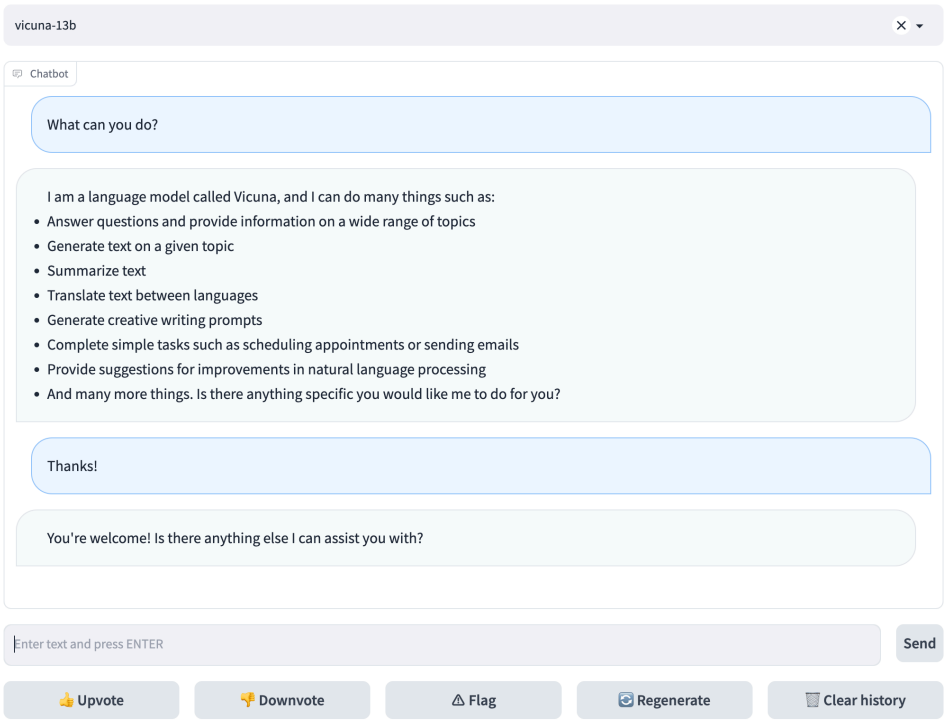
- FastChat supports AWQ 4bit inference with mit-han-lab/llm-awq. See docs/awq.md.
- MLC LLM, backed by TVM Unity compiler, deploys Vicuna natively on phones, consumer-class GPUs and web browsers via Vulkan, Metal, CUDA and WebGPU.

Use models from modelscope

For Chinese users, you can use models from www.modelscope.cn via specify the following environment variables.

```
export FASTCHAT_USE_MODELSCOPE=True
```

Serving with Web GUI



To serve using the web UI, you need three main components: web servers that interface with users, model workers that host one or more models, and a controller to coordinate the webserver and model workers. You can learn more about the architecture here.

Here are the commands to follow in your terminal:

Launch the controller

```
python3 -m fastchat.serve.controller
```

This controller manages the distributed workers.

Launch the model worker(s)

```
python3 -m fastchat.serve.model_worker --model-path lmsys/vicuna-7b-v1.5
```

Wait until the process finishes loading the model and you see "Uvicorn running on ...". The model worker will register itself to the controller .

To ensure that your model worker is connected to your controller properly, send a test message using the following command:

```
python3 -m fastchat.serve.test_message --model-name vicuna-7b-v1.5
```

You will see a short output.

Launch the Gradio web server

```
python3 -m fastchat.serve.gradio_web_server
```

This is the user interface that users will interact with.

By following these steps, you will be able to serve your models using the web UI. You can open your browser and chat with a model now. If the models do not show up, try to reboot the gradio web server.

Launch Chatbot Arena (side-by-side battle UI)

Currently, Chatbot Arena is powered by FastChat. Here is how you can launch an instance of Chatbot Arena locally.

FastChat supports popular API-based models such as OpenAI, Anthropic, Gemini, Mistral and more. To add a custom API, please refer to the model support doc. Below we take OpenAI models as an example.

Create a JSON configuration file `api_endpoint.json` with the api endpoints of the models you want to serve, for example:

```
{
    "gpt-4o-2024-05-13": {
        "model_name": "gpt-4o-2024-05-13",
        "api_base": "https://api.openai.com/v1",
        "api_type": "openai",
        "api_key": [Insert API Key],
        "anony_only": false
    }
}
```

For Anthropic models, specify `"api_type": "anthropic_message"` with your Anthropic key. Similarly, for gemini model, specify `"api_type": "gemini"`. More details can be found in api_provider.py.

To serve your own model using local gpus, follow the instructions in Serving with Web GUI.

Now you're ready to launch the server:

```
python3 -m fastchat.serve.gradio_web_server_multi --register-api-endpoint-file api_endpoint.json
```

(Optional): Advanced Features, Scalability, Third Party UI

- You can register multiple model workers to a single controller, which can be used for serving a single model with higher throughput or serving multiple models at the same time. When doing so, please allocate different GPUs and ports for different model workers.

```
# worker 0
CUDA_VISIBLE_DEVICES=0 python3 -m fastchat.serve.model_worker --model-path lmsys/vicuna-7b-v1.5 --
controller http://localhost:21001 --port 31000 --worker http://localhost:31000
# worker 1
CUDA_VISIBLE_DEVICES=1 python3 -m fastchat.serve.model_worker --model-path lmsys/fastchat-t5-3b-v1.0 --
controller http://localhost:21001 --port 31001 --worker http://localhost:31001
```

- You can also launch a multi-tab gradio server, which includes the Chatbot Arena tabs.

```
python3 -m fastchat.serve.gradio_web_server_multi
```

- The default model worker based on huggingface/transformers has great compatibility but can be slow. If you want high-throughput batched serving, you can try vLLM integration.

- If you want to host it on your own UI or third party UI, see Third Party UI.

API

OpenAI-Compatible RESTful APIs & SDK

FastChat provides OpenAI-compatible APIs for its supported models, so you can use FastChat as a local drop-in replacement for OpenAI APIs. The FastChat server is compatible with both openai-python library and cURL commands. The REST API is capable of being executed from Google Colab free tier, as demonstrated in the FastChat_API_GoogleColab.ipynb notebook, available in our repository. See docs/openai_api.md.

Hugging Face Generation APIs

See fastchat/serve/huggingface_api.py.

LangChain Integration

See docs/langchain_integration.

Evaluation

We use MT-bench, a set of challenging multi-turn open-ended questions to evaluate models. To automate the evaluation process, we prompt strong LLMs like GPT-4 to act as judges and assess the quality of the models' responses. See instructions for running MT-bench at fastchat/llm_judge.

MT-bench is the new recommended way to benchmark your models. If you are still looking for the old 80 questions used in the vicuna blog post, please go to vicuna-blog-eval.

Fine-tuning

Data

Vicuna is created by fine-tuning a Llama base model using approximately 125K user-shared conversations gathered from ShareGPT.com with public APIs. To ensure data quality, we convert the HTML back to markdown and filter out some inappropriate or low-quality samples. Additionally, we divide lengthy conversations into smaller segments that fit the model's maximum context length. For detailed instructions to clean the ShareGPT data, check out here.

We will not release the ShareGPT dataset. If you would like to try the fine-tuning code, you can run it with some dummy conversations in dummy_conversation.json. You can follow the same format and plug in your own data.

Code and Hyperparameters

Our code is based on Stanford Alpaca with additional support for multi-turn conversations. We use similar hyperparameters as the Stanford Alpaca.

Hyperparameter	Global Batch Size	Learning rate	Epochs	Max length	Weight decay
Vicuna-13B	128	2e-5	3	2048	0

Fine-tuning Vicuna-7B with Local GPUs

- Install dependency

```
pip3 install -e ".[train]"
```

- You can use the following command to train Vicuna-7B with 4 x A100 (40GB). Update `--model_name_or_path` with the actual path to Llama weights and `--data_path` with the actual path to data.

```
torchrun --nproc_per_node=4 --master_port=20001 fastchat/train/train_mem.py \
    --model_name_or_path meta-llama/Llama-2-7b-hf \
    --data_path data/dummy_conversation.json \
    --bf16 True \
    --output_dir output_vicuna \
    --num_train_epochs 3 \
    --per_device_train_batch_size 2 \
    --per_device_eval_batch_size 2 \
    --gradient_accumulation_steps 16 \
    --evaluation_strategy "no" \
    --save_strategy "steps" \
    --save_steps 1200 \
    --save_total_limit 10 \
    --learning_rate 2e-5 \
    --weight_decay 0. \
    --warmup_ratio 0.03 \
    --lr_scheduler_type "cosine" \
    --logging_steps 1 \
    --fsdp "full_shard auto_wrap" \
    --fsdp_transformer_layer_cls_to_wrap 'LlamaDecoderLayer' \
    --tf32 True \
    --model_max_length 2048 \
    --gradient_checkpointing True \
    --lazy_preprocess True
```

Tips:

- If you are using V100 which is not supported by FlashAttention, you can use the memory-efficient attention implemented in xFormers. Install xformers and replace `fastchat/train/train_mem.py` above with fastchat/train/train_xformers.py.
- If you meet out-of-memory due to "FSDP Warning: When using FSDP, it is efficient and recommended... ", see solutions here.
- If you meet out-of-memory during model saving, see solutions here.
- To turn on logging to popular experiment tracking tools such as Tensorboard, MLFlow or Weights & Biases, use the `report_to` argument, e.g. pass `--report_to wandb` to turn on logging to Weights & Biases.

Other models, platforms and LoRA support

More instructions to train other models (e.g., FastChat-T5) and use LoRA are in docs/training.md.

Fine-tuning on Any Cloud with SkyPilot

SkyPilot is a framework built by UC Berkeley for easily and cost effectively running ML workloads on any cloud (AWS, GCP, Azure, Lambda, etc.). Find SkyPilot documentation here on using managed spot instances to train Vicuna and save on your cloud costs.

Citation

The code (training, serving, and evaluation) in this repository is mostly developed for or derived from the paper below. Please cite it if you find the repository helpful.

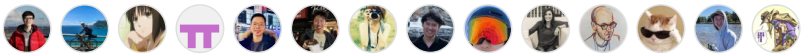
```
@misc{zheng2023judging,
      title={Judging LLM-as-a-judge with MT-Bench and Chatbot Arena},
      author={Lianmin Zheng and Wei-Lin Chiang and Ying Sheng and Siyuan Zhuang and Zhanghao Wu and
Yonghao Zhuang and Zi Lin and Zhuohan Li and Dacheng Li and Eric. P Xing and Hao Zhang and Joseph E.
Gonzalez and Ion Stoica},
      year={2023},
      eprint={2306.05685},
      archivePrefix={arXiv},
```

Releases 14

Release v0.2.36 Latest
on Feb 11, 2024

+ 13 releases

Contributors 255



+ 241 contributors

Languages

Python 91.6% **Jupyter Notebook** 7.7% **Other** 0.7%