starriver030515/**LLaVA** ♡ like 0

arxiv:9 papers

Model card | Files and versions xet | Community

Edit model card

YAML Metadata Warning: empty or missing yaml metadata in repo card (https://huggingface.co/docs/hub/model-cards#model-card-metadata)

🌋 LLaVA: Large Language and Vision Assistant

Visual instruction tuning towards large language and vision models with GPT-4 level capabilities.

[📢 LLaVA-NeXT Blog] [Project Page] [Demo] [Data] [Model Zoo]

🤝 Community Contributions: [llama.cpp] [Colab] [🤗 Space] [Replicate] [AutoGen] [BakLLaVA]

Improved Baselines with Visual Instruction Tuning [Paper] [HF]
Haotian Liu, Chunyuan Li, Yuheng Li, Yong Jae Lee

Visual Instruction Tuning (NeurIPS 2023, **Oral**) [Paper] [HF]
Haotian Liu*, Chunyuan Li*, Qingyang Wu, Yong Jae Lee (*Equal Contribution)

Release

- [03/10] Releasing **LMMs-Eval**, a highly efficient evaluation pipeline we used when developing LLaVA-NeXT. It supports the evaluation of LMMs on dozens of public datasets and allows new dataset onboarding, making the dev of new LMMs much faster. [Blog] [Codebase]
- [1/30] 🔥 LLaVA-NeXT (LLaVA-1.6) is out! With additional scaling to LLaVA-1.5, LLaVA-NeXT-34B outperforms Gemini Pro on some benchmarks. It can now process 4x more pixels and perform more tasks/applications than before. Check out the blog post, and explore the demo! Models are available in Model Zoo. Training/eval data and scripts coming soon.
- [11/10] LLaVA-Plus is released: Learning to Use Tools for Creating Multimodal Agents, with LLaVA-Plus (LLaVA that Plug and Learn to Use Skills). [Project Page] [Demo] [Code] [Paper]
- [11/2] LLaVA-Interactive is released: Experience the future of human-AI multimodal interaction with an all-in-one demo for Image Chat, Segmentation, Generation and Editing. [Project Page] [Demo] [Code] [Paper]
- [10/26] 🔥 LLaVA-1.5 with LoRA achieves comparable performance as full-model finetuning, with a reduced GPU RAM requirement (ckpts, script). We also provide a doc on how to finetune LLaVA-1.5 on your own dataset with LoRA.
- [10/12] Check out the Korean LLaVA (Ko-LLaVA), created by ETRI, who has generously supported our research! [🤗 Demo]
- [10/5] 🔥 LLaVA-1.5 is out! Achieving SoTA on 11 benchmarks, with just simple modifications to the original LLaVA, utilizes all public data, completes training in ~1 day on a single 8-A100 node, and surpasses methods like Qwen-VL-Chat that use billion-scale data. Check out the technical report, and explore the demo! Models are available in Model Zoo. The training data and scripts of LLaVA-1.5 are released here, and evaluation scripts are released here!
- [9/26] LLaVA is improved with reinforcement learning from human feedback (RLHF) to improve fact grounding and reduce hallucination. Check out the new SFT and RLHF checkpoints at project [LLavA-RLHF]
- [9/22] LLaVA is accepted by NeurIPS 2023 as **oral presentation**, and LLaVA-Med is accepted by NeurIPS 2023 Datasets and Benchmarks Track as **spotlight presentation**.

▶ More

Code License Apache 2.0 **Usage and License Notices**: This project utilizes certain datasets and checkpoints that are subject to their respective original licenses. Users must comply with all terms and conditions of these original licenses, including but not limited to the OpenAI Terms of Use for the dataset and the specific licenses for base language models for checkpoints trained using the dataset (e.g. Llama community license for LLaMA-2 and Vicuna-v1.5). This project does not impose any additional constraints beyond those stipulated in the original licenses. Furthermore, users are reminded to ensure that their use of the dataset and checkpoints is in compliance with all applicable laws and regulations.

Contents

Install

If you are not using Linux, do *NOT* proceed, see instructions for macOS and Windows.

1. Clone this repository and navigate to LLaVA folder

```
git clone https://github.com/haotian-liu/LLaVA.git
cd LLaVA
```

2. Install Package

```
conda create -n llava python=3.10 -y
conda activate llava
pip install --upgrade pip  # enable PEP 660 support
pip install -e .
```

3. Install additional packages for training cases

```
pip install -e ".[train]"
pip install flash-attn --no-build-isolation
```

Upgrade to latest code base

```
git pull
pip install -e .

# if you see some import errors when you upgrade,
# please try running the command below (without #)
# pip install flash-attn --no-build-isolation --no-cache-dir
```

Quick Start With HuggingFace

▶ Example Code

LLaVA Weights

Please check out our Model Zoo for all public LLaVA checkpoints, and the instructions of how to use the weights.

Demo

Gradio Web UI

To launch a Gradio demo locally, please run the following commands one by one. If you plan to launch multiple model workers to compare between different checkpoints, you only need to launch the controller and the web server *ONCE*.

```
flowchart BT
    %% Declare Nodes
    gws("Gradio (UI Server)")
    c("Controller (API Server):<br/>PORT: 10000")
    mw7b("Model Worker:<br/>llava-v1.5-7b<br/>PORT: 40000")
    mw13b("Model Worker:<br/>llava-v1.5-13b<br/>PORT: 40001")
    sglw13b("SGLang Backend:<br/>llava-v1.6-34b<br/>http://localhost:30000")
    lsglw13b("SGLang Worker:<br/>llava-v1.6-34b<br/>PORT: 40002")

    %% Declare Styles
    classDef data fill:#3af,stroke:#48a,stroke-width:2px,color:#444
    classDef success fill:#8f8,stroke:#0a0,stroke-width:2px,color:#444
    classDef failure fill:#f88,stroke:#f00,stroke-width:2px,color:#444

    %% Assign Styles
    class id,od data;
    class cimg,cs_s,scsim_s success;
    class ncimg,cs_f,scsim_f failure;

    subgraph Demo Connections
        direction BT
        c<-->gws

        mw7b<-->c
        mw13b<-->c
        lsglw13b<-->c
        sglw13b<-->lsglw13b
    end
```

Launch a controller

```
python -m llava.serve.controller --host 0.0.0.0 --port 10000
```

Launch a gradio web server.

```
python -m llava.serve.gradio_web_server --controller http://localhost:10000 --
model-list-mode reload
```

You just launched the Gradio web interface. Now, you can open the web interface with the URL printed on the screen. You may notice that there is no model in the model list. Do not worry, as we have not launched any model worker yet. It will be automatically updated when you launch a model worker.

Launch a SGLang worker

This is the recommended way to serve LLaVA model with high throughput, and you need to install SGLang first. Note that currently `4-bit` quantization is not supported yet on SGLang-LLaVA, and if you have limited GPU VRAM, please check out model worker with quantization.

```
pip install "sglang[all]"
```

You'll first launch a SGLang backend worker which will execute the models on GPUs. Remember the `--port` you've set and you'll use that later.

```
# Single GPU
CUDA_VISIBLE_DEVICES=0 python3 -m sglang.launch_server --model-path
liuhaotian/llava-v1.5-7b --tokenizer-path llava-hf/llava-1.5-7b-hf --port 30000

# Multiple GPUs with tensor parallel
CUDA_VISIBLE_DEVICES=0,1 python3 -m sglang.launch_server --model-path
liuhaotian/llava-v1.5-13b --tokenizer-path llava-hf/llava-1.5-13b-hf --port
30000 --tp 2
```

Tokenizers (temporary): `llava-hf/llava-1.5-7b-hf`, `llava-hf/llava-1.5-13b-hf`, `liuhaotian/llava-v1.6-34b-tokenizer`.

You'll then launch a LLaVA-SGLang worker that will communicate between LLaVA controller and SGLang backend to route the requests. Set `--sgl-endpoint` to `http://127.0.0.1:port` where `port` is the one you just set (default: 30000).

```
python -m llava.serve.sglang_worker --host 0.0.0.0 --controller
http://localhost:10000 --port 40000 --worker http://localhost:40000 --sgl-
endpoint http://127.0.0.1:30000
```

Launch a model worker

This is the actual *worker* that performs the inference on the GPU. Each worker is responsible for a single model specified in `--model-path`.

```
python -m llava.serve.model_worker --host 0.0.0.0 --controller
http://localhost:10000 --port 40000 --worker http://localhost:40000 --model-
path liuhaotian/llava-v1.5-13b
```

Wait until the process finishes loading the model and you see "Uvicorn running on ...". Now, refresh your Gradio web UI, and you will see the model you just launched in the model list.

You can launch as many workers as you want, and compare between different model checkpoints in the same Gradio interface. Please keep the `--controller` the same, and modify the `--port` and `--worker` to a different port number for each worker.

```
python -m llava.serve.model_worker --host 0.0.0.0 --controller
http://localhost:10000 --port <different from 40000, say 40001> --worker
http://localhost:<change accordingly, i.e. 40001> --model-path <ckpt2>
```

If you are using an Apple device with an M1 or M2 chip, you can specify the mps device by using the `--device` flag: `--device mps`.

Launch a model worker (Multiple GPUs, when GPU VRAM <= 24GB)

If the VRAM of your GPU is less than 24GB (e.g., RTX 3090, RTX 4090, etc.), you may try running it with multiple GPUs. Our latest code base will automatically try to use multiple GPUs if you have more than one GPU. You can specify which GPUs to use with `CUDA_VISIBLE_DEVICES`. Below is an example of running with the first two GPUs.

```
CUDA_VISIBLE_DEVICES=0,1 python -m llava.serve.model_worker --host 0.0.0.0 --
controller http://localhost:10000 --port 40000 --worker http://localhost:40000
--model-path liuhaotian/llava-v1.5-13b
```

Launch a model worker (4-bit, 8-bit inference, quantized)

You can launch the model worker with quantized bits (4-bit, 8-bit), which allows you to run the inference with reduced GPU memory footprint, potentially allowing you to run on a GPU with as few

Downloads last month
-
Downloads are not tracked for this model. How to track

Inference Providers NEW

This model isn't deployed by any Inference Provider. Ask for provider support as 12GB VRAM. Note that inference with quantized bits may not be as accurate as the full-precision model. Simply append `--load-4bit` or `--load-8bit` to the **model worker** command that you are executing. Below is an example of running with 4-bit quantization.

```
python -m llava.serve.model_worker --host 0.0.0.0 --controller
http://localhost:10000 --port 40000 --worker http://localhost:40000 --model-
path liuhaotian/llava-v1.5-13b --load-4bit
```

Launch a model worker (LoRA weights, unmerged)

You can launch the model worker with LoRA weights, without merging them with the base checkpoint, to save disk space. There will be additional loading time, while the inference speed is the same as the merged checkpoints. Unmerged LoRA checkpoints do not have `lora-merge` in the model name, and are usually much smaller (less than 1GB) than the merged checkpoints (13G for 7B, and 25G for 13B).

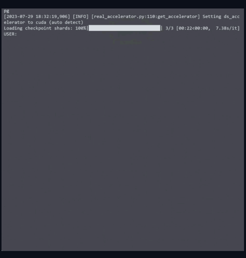
To load unmerged LoRA weights, you simply need to pass an additional argument `--model-base`, which is the base LLM that is used to train the LoRA weights. You can check the base LLM of each LoRA weights in the model zoo.

```
python -m llava.serve.model_worker --host 0.0.0.0 --controller
http://localhost:10000 --port 40000 --worker http://localhost:40000 --model-
path liuhaotian/llava-v1-0719-336px-lora-vicuna-13b-v1.3 --model-base
lmsys/vicuna-13b-v1.3
```

CLI Inference

Chat about images using LLaVA without the need of Gradio interface. It also supports multiple GPUs, 4-bit and 8-bit quantized inference. With 4-bit quantization, for our LLaVA-1.5-7B, it uses less than 8GB VRAM on a single GPU.

```
python -m llava.serve.cli \
    --model-path liuhaotian/llava-v1.5-7b \
    --image-file "https://llava-vl.github.io/static/images/view.jpg" \
    --load-4bit
```



```
[2023-07-29 18:32:19,906] [INFO] [real_accelerator.py:110:get_accelerator] Setting ds_acc
elerator to cuda (auto detect)
Loading checkpoint shards: 100%|██████████| 3/3 [00:22<00:00,  7.38s/it]
USER:
```

Train

Below is the latest training configuration for LLaVA v1.5. For legacy models, please refer to README of this version for now. We'll add them in a separate doc later.

LLaVA training consists of two stages: (1) feature alignment stage: use our 558K subset of the LAION-CC-SBU dataset to connect a *frozen pretrained* vision encoder to a *frozen LLM*; (2) visual instruction tuning stage: use 150K GPT-generated multimodal instruction-following data, plus around 515K VQA data from academic-oriented tasks, to teach the model to follow multimodal instructions.

LLaVA is trained on 8 A100 GPUs with 80GB memory. To train on fewer GPUs, you can reduce the `per_device_train_batch_size` and increase the `gradient_accumulation_steps` accordingly. Always keep the global batch size the same: `per_device_train_batch_size` x `gradient_accumulation_steps` x `num_gpus`.

Hyperparameters

We use a similar set of hyperparameters as Vicuna in finetuning. Both hyperparameters used in pretraining and finetuning are provided below.

1. Pretraining

Hyperparameter	Global Batch Size	Learning rate	Epochs	Max length	Weight decay
LLaVA-v1.5-13B	256	1e-3	1	2048	0

2. Finetuning

Hyperparameter	Global Batch Size	Learning rate	Epochs	Max length	Weight decay
LLaVA-v1.5-13B	128	2e-5	1	2048	0

Download Vicuna checkpoints (automatically)

Our base model Vicuna v1.5, which is an instruction-tuned chatbot, will be downloaded automatically when you run our provided training scripts. No action is needed.

Pretrain (feature alignment)

Please download the 558K subset of the LAION-CC-SBU dataset with BLIP captions we use in the paper here.

Pretrain takes around 5.5 hours for LLaVA-v1.5-13B on 8x A100 (80G), due to the increased resolution to 336px. It takes around 3.5 hours for LLaVA-v1.5-7B.

Training script with DeepSpeed ZeRO-2: `pretrain.sh`.

- `--mm_projector_type mlp2x_gelu`: the two-layer MLP vision-language connector.
- `--vision_tower openai/clip-vit-large-patch14-336`: CLIP ViT-L/14 336px.

▶ Pretrain takes around 20 hours for LLaVA-7B on 8x V100 (32G)

Visual Instruction Tuning

1. Prepare data

Please download the annotation of the final mixture our instruction tuning data llava_v1_5_mix665k.json, and download the images from constituting datasets:

- COCO: train2017
- GQA: images
- OCR-VQA: download script, **we save all files as** `.jpg`
- TextVQA: train_val_images
- VisualGenome: part1, part2

After downloading all of them, organize the data as follows in `./playground/data`,

```
├── coco
│   └── train2017
├── gqa
│   └── images
├── ocr_vqa
│   └── images
├── textvqa
│   └── train_images
└── vg
    ├── VG_100K
    └── VG_100K_2
```

2. Start training!

You may download our pretrained projectors in Model Zoo. It is not recommended to use legacy projectors, as they may be trained with a different version of the codebase, and if any option is off, the model will not function/train as we expected.

Visual instruction tuning takes around 20 hours for LLaVA-v1.5-13B on 8x A100 (80G), due to the increased resolution to 336px. It takes around 10 hours for LLaVA-v1.5-7B on 8x A100 (40G).

Training script with DeepSpeed ZeRO-3: `finetune.sh`.

If you are do not have enough GPU memory:

- Use LoRA: `finetune_lora.sh`. We are able to fit 13B training in 8-A100-40G/8-A6000, and 7B training in 8-RTX3090. Make sure `per_device_train_batch_size*gradient_accumulation_steps` is the same as the provided script for best reproducibility.
- Replace `zero3.json` with `zero3_offload.json` which offloads some parameters to CPU RAM. This slows down the training speed.

If you are interested in finetuning LLaVA model to your own task/data, please check out `Finetune_Custom_Data.md`。

New options to note:

- `--mm_projector_type mlp2x_gelu`: the two-layer MLP vision-language connector.
- `--vision_tower openai/clip-vit-large-patch14-336`: CLIP ViT-L/14 336px.
- `--image_aspect_ratio pad`: this pads the non-square images to square, instead of cropping them; it slightly reduces hallucination.
- `--group_by_modality_length True`: this should only be used when your instruction tuning dataset contains both language (e.g. ShareGPT) and multimodal (e.g. LLaVA-Instruct). It makes the training sampler only sample a single modality (either image or language) during training, which we observe to speed up training by ~25%, and does not affect the final outcome.

Evaluation

In LLaVA-1.5, we evaluate models on a diverse set of 12 benchmarks. To ensure the reproducibility, we evaluate the models with greedy decoding. We do not evaluate using beam search to make the inference process consistent with the chat demo of real-time outputs.

See Evaluation.md.

GPT-assisted Evaluation

Our GPT-assisted evaluation pipeline for multimodal modeling is provided for a comprehensive understanding of the capabilities of vision-language models. Please see our paper for more details.

1. Generate LLaVA responses

```
python model_vqa.py \
    --model-path ./checkpoints/LLaVA-13B-v0 \
    --question-file \
    playground/data/coco2014_val_qa_eval/qa90_questions.jsonl \
    --image-folder \
    /path/to/coco2014_val \
    --answers-file \
    /path/to/answer-file-our.jsonl
```

2. Evaluate the generated responses. In our case, `answer-file-ref.jsonl` is the response generated by text-only GPT-4 (0314), with the context captions/boxes provided.

```
OPENAI_API_KEY="sk-***********************************" python
llava/eval/eval_gpt_review_visual.py \
    --question playground/data/coco2014_val_qa_eval/qa90_questions.jsonl \
    --context llava/eval/table/caps_boxes_coco2014_val_80.jsonl \
    --answer-list \
    /path/to/answer-file-ref.jsonl \
    /path/to/answer-file-our.jsonl \
    --rule llava/eval/table/rule.json \
    --output /path/to/review.json
```

3. Summarize the evaluation results

```
python summarize_gpt_review.py
```

Citation

If you find LLaVA useful for your research and applications, please cite using this BibTeX:

```
@misc{liu2024llavanext,
    title={LLaVA-NeXT: Improved reasoning, OCR, and world knowledge},
    url={https://llava-vl.github.io/blog/2024-01-30-llava-next/},
    author={Liu, Haotian and Li, Chunyuan and Li, Yuheng and Li, Bo and Zhang,
Yuanhan and Shen, Sheng and Lee, Yong Jae},
    month={January},
    year={2024}
}

@misc{liu2023improvedllava,
      title={Improved Baselines with Visual Instruction Tuning},
      author={Liu, Haotian and Li, Chunyuan and Li, Yuheng and Lee, Yong Jae},
      publisher={arXiv:2310.03744},
      year={2023},
}

@misc{liu2023llava,
      title={Visual Instruction Tuning},
      author={Liu, Haotian and Li, Chunyuan and Wu, Qingyang and Lee, Yong
Jae},
      publisher={NeurIPS},
      year={2023},
}
```

Acknowledgement

- Vicuna: the codebase we built upon, and our base model Vicuna-13B that has the amazing language capabilities!

Related Projects

- Instruction Tuning with GPT-4
- LLaVA-Med: Training a Large Language-and-Vision Assistant for Biomedicine in One Day
- Otter: In-Context Multi-Modal Instruction Tuning

For future project ideas, please check out:

- SEEM: Segment Everything Everywhere All at Once
- Grounded-Segment-Anything to detect, segment, and generate anything by marrying Grounding DINO and Segment-Anything.