Fine-tuning LLMs on custom Dataset!🚀

powered by: LightningAI's Lit-GPT!⚡

@akshay_pachaar

Lightning AI recently launched Lit-GPT!

A clean, solid & optimised LLM implementation with pre-training & fine-tuning support using LoRA & Adapter.

We'll go step-by-step:
- Installations
- model download
- data prep.
- fine-tuning

1️⃣ Installation & Setup:

Swipe 👉

@akshay_pachaar

Installation & Setup

```
git clone <https://github.com/Lightning-AI/lit-gpt
cd lit-gpt
```

We are using FlashAttention, a fast and memory-efficient implementation of attention,

@akshay_pachaar

```
# for cuda
pip install --index-url <https://download.pytorch.org/whl/nightly/cu118> --pre 'torch⩾2.1.0dev'

# for cpu
pip install --index-url <https://download.pytorch.org/whl/nightly/cpu> --pre 'torch⩾2.1.0dev'

# install all the dependencies
pip install -r requirements.txt
```

! All the instructions & code provided at the end !

2️⃣ Downloading the model weights

Lit-GPT supports following pretrained weights:

- StableLM
- Pythia
- RedPajama-INCITE

Here we gonna use RedPajama-INCITE 3B weights!

And doing this doesn't get easier!

Swipe 👉

@akshay_pachaar

Download & convert weights to Lit-GPT format

@akshay_pachaar

```
# download the model weights
python scripts/download.py --repo_id togethercomputer/RedPajama-INCITE-Base-3B-v1

# convert the weights to Lit-Parrot format
python scripts/convert_hf_checkpoint.py --checkpoint_dir \n
checkpoints/togethercomputer/RedPajama-INCITE-Base-3B-v1
```

3️⃣ Download Dataset for fine-tuning

We'll use the Dolly 2.0 instruction dataset!

Instruction datasets typically have 3 keys:
- instruction
- input (optional context for given instruction)
- expected response from the LLM.

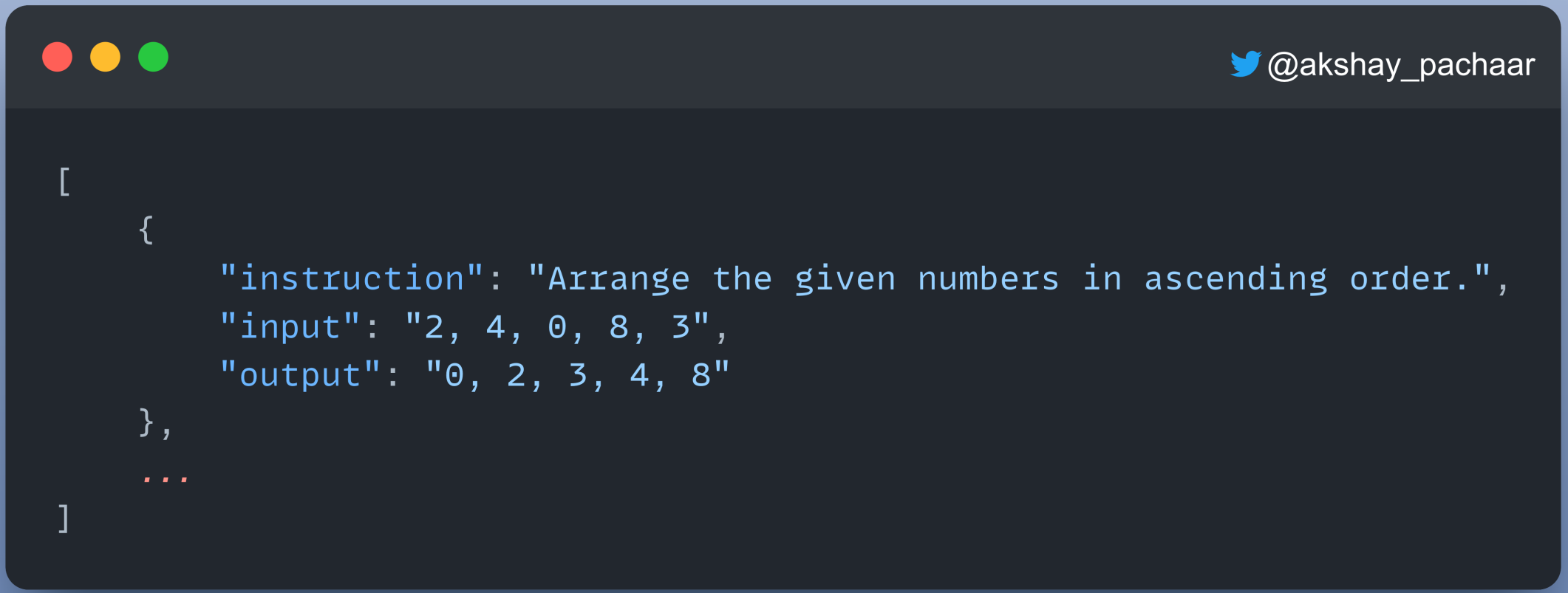
Check the example!

Swipe 👉

@akshay_pachaar

Dolly-2.0

Instruction Dataset Sample



4️⃣ Prepare the dataset

What needs to be done:
- loading the raw instruction dataset
- creating prompts
- tokenization

Following code does it all for you
(I'll provide link to all the code)

Swipe 👉

@akshay_pachaar

```python
DATA_FILE = "https://huggingface.co/datasets/databricks/databricks-dolly-15k/resolve/main/databricks-dolly-15k.jsonl"
DATA_FILE_NAME = "dolly_data_cleaned_archive.json"

def prepare(
    destination_path: Path = Path("data/dolly"),
    checkpoint_dir: Path = Path("checkpoints/togethercomputer/RedPajama-INCITE-Base-3B-v1"),
    test_split_size: int = 2000,
    max_seq_length: int = 256,
    seed: int = 42,
    mask_inputs: bool = False,  # as in alpaca-lora
    data_file_name: str = DATA_FILE_NAME,
) -> None:
    """Prepare the Dolly dataset for instruction tuning.

    The output is a training and validation dataset saved as `train.pt` and `val.pt`,
    which stores the preprocessed and tokenized prompts and labels.
    """
    destination_path.mkdir(parents=True, exist_ok=True)
    file_path = destination_path / data_file_name
    download(file_path)

    tokenizer = Tokenizer(checkpoint_dir / "tokenizer.json", checkpoint_dir / "tokenizer_config.json")

    with open(file_path, "r") as file:
        data = file.readlines()
        data = [json.loads(line) for line in data]
    for item in data:
        item["input"] = item.pop("context")
        item["output"] = item.pop("response")

    # Partition the dataset into train and test
    train_split_size = len(data) - test_split_size
    train_set, test_set = random_split(
        data, lengths=(train_split_size, test_split_size), generator=torch.Generator().manual_seed(seed)
    )
    train_set, test_set = list(train_set), list(test_set)

    print(f"train has {len(train_set):,} samples")
    print(f"val has {len(test_set):,} samples")

    print("Processing train split ...")
    train_set = [prepare_sample(sample, tokenizer, max_seq_length, mask_inputs) for sample in tqdm(train_set)]
    torch.save(train_set, file_path.parent / "train.pt")

    print("Processing test split ...")
    test_set = [prepare_sample(sample, tokenizer, max_seq_length, mask_inputs) for sample in tqdm(test_set)]
    torch.save(test_set, file_path.parent / "test.pt")
```

5️⃣ Fine-tuning the RedPajama-INCITE model

Once you have the data prepared, you can start fine-tuning right away! 🚀

You need to run the `finetune_adapter .py` script by providing your data path.

Swipe 👉

@akshay_pachaar

Finetuining

@akshay_pachaar

```
python finetune_adapter.py \
    --data_dir data/dolly \
    --checkpoint_dir checkpoints/togethercomputer/RedPajama-INCITE-Base-3B-v1
    --out_dir out/adapter/dolly
```

And we are done! 🎉

Let's play with the fine-tuned model!

Swipe 👉

@akshay_pachaar

Playing with the model !

@akshay_pachaar

```
python generate_adapter.py \
    --adapter_path out/adapter/dolly/lit_model_adapter_finetuned.pth \
    --checkpoint_dir checkpoints/togethercomputer/RedPajama-INCITE-Base-3B-v1 \
    --prompt "who is the author of Game of thrones?"
```

```
Loading model ...
Time to load model: 11.32 seconds.
George R. R. Martin is the author of Game of Thrones.


Time for inference: 2.16 sec total, 7.86 tokens/sec
Memory used: 5.72 GB
```

That's a wrap!

If you interested in:

- Python 🐍
- Data Science 📈
- Machine Learning 🤖
- MLOps 🛠️
- NLP 🗣️
- Computer Vision 📽️
- LLMs 🧠

Follow me on LinkedIn ✔️
Everyday, I share tutorials on above topics!

Cheers!! 🙂

@akshay_pachaar