FINE-TUNING LLM

Akila Peiris

SUMMARY

- Fine-tuning LLM
- Challenges
- Solution
- LoRA
- Quantization
- Cloud GPU
- Runpod
- Axolotl
- Setps to fine-tune LLM
- Recommendations

FINE-TUNING LLM

TO PERFORM SPECIFIC TASKS

TO EMBED DOMAIN KNOWLEDGE

NEED APPROPRIATE DATA SET

COMPLEX PROCESS

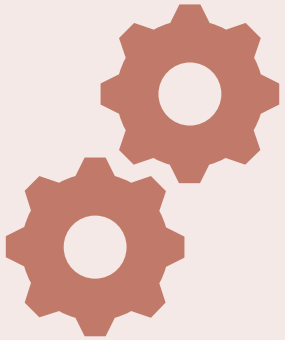
ADJUSTS BILLIONS OF PARAMETERS

TIME AND RESOURCE INTENSIVE

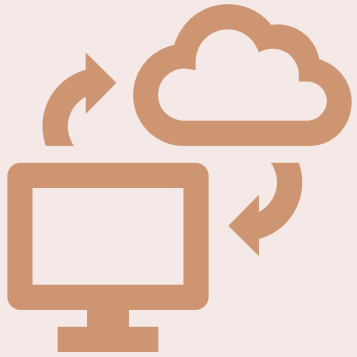
CHALLENGES IN FINE-TUNING LLM

- Highly resource intensive
- Cannot be accomplished on consumer hardware
- Time consuming
- Larger the data set, longer and more complex the fine-tuning

2 PRONGED SOLUTION



Optimize fine-tuning algorithms and process



Use better hardware. Use cloud infrastructures

OPTIMIZE FINE-TUNING - LoRA

- Low Rank Adaptation (LoRA) [1]
- Freeze layers of pre-trained model except top few layers
- Inject smaller weights into it
- Fine-tune the smaller weights set
- Reduces the number of parameters to train and the memory required to do so
- https://nisansads.staff.uom.lk/GroupMeeting/065_06_09_2023/slides.pdf
- https://drive.google.com/file/d/1D5khrhytjeWixTU91hBk4g0q6FQVJrET/view?usp=sharing

[1] Edward J Hu, 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.

OPTIMIZE FINE-TUNING - QUANTIZATION

- Reduce the model size and computation by reducing the precision of the model weights
- Based on research on 8 and 4-bit quantization [2][3]
- 16 bit to a lower number
- Quality vs. efficiency
- Ex: Lossless fine-tuning of 65 billion parameter model on a single GPU and 48GB memory
- Also reduces model size
- QLoRA - Quantized Low-Rank Adaptation
- Huggingface Transformer library compatible

[2] Tim Dettmers. 8-bit approximations for parallelism in deep learning. arXiv preprint arXiv:1511.04561, 2015.

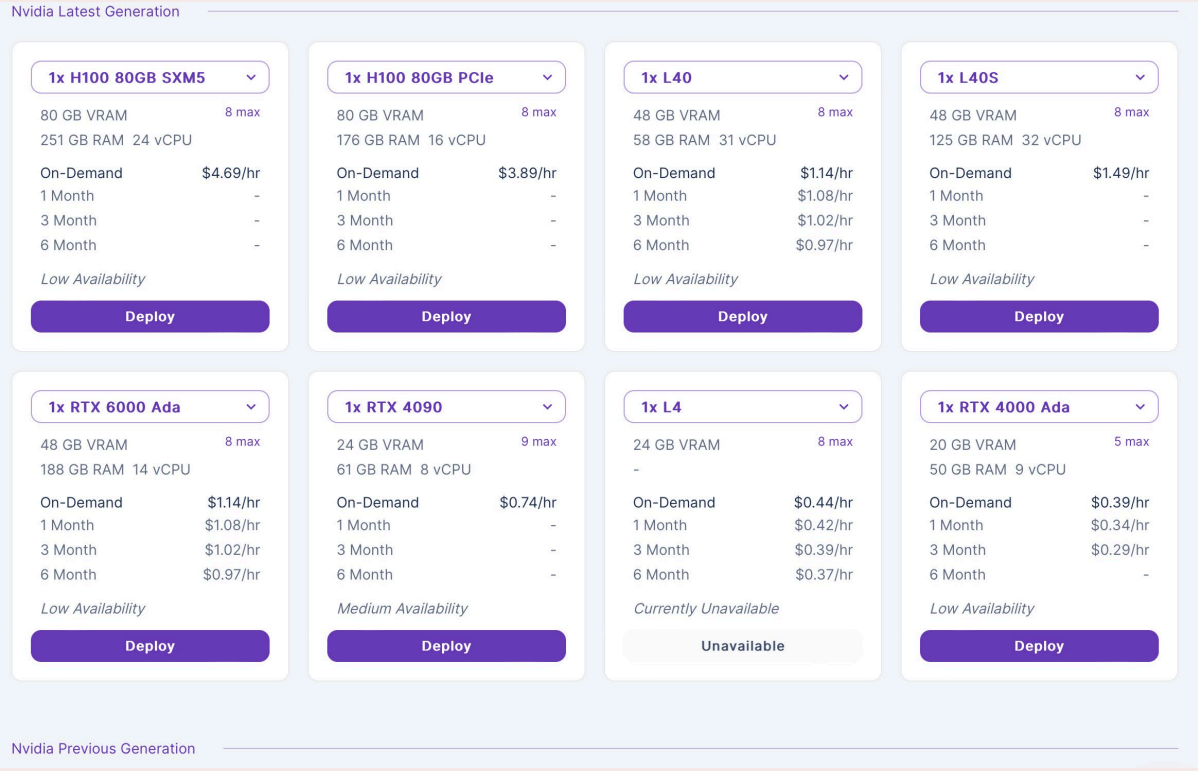
[3] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. arXiv preprint arXiv:2212.09720, 2022.

CLOUD RESOURCES

- Infrastructure as a Service
- Rent GPUs
- Multiple cloud GPU vendors
- AWS, GCP, Runpod
- Runpod is specific for ML work using GPU

RUNPOD

- https://runpod.io
- Distributed cloud GPU vendor
- Multiple templates for different LLM tasks
- Templates include text-generation-webui
- Different GPUs and configurations available



AXOLOTL

- oobabooga text-generation-webui can train models
  - But only a handful of architectures including LLaMA
  - Can’t train Mistral
  - 3rd part plugin Training PRO
- **Axolotl by OpenAccess-AI-Collective**
- https://github.com/OpenAccess-AI-Collective/axolotl
- Train various Huggingface models (ex: llama, pythia, falcon, mpt, Mistral)
- Supports full fine-tune, lora, qlora, and gptq
- Has own Runpod Template

DATA SET

- Data set needs to be JSONL
- Different formats for,
  - Completion (raw corpus) - {"text": "..."}
  - Instruction (alpaca) - {"instruction": "...", "input": "...", "output": "..."}
  - input_output (template free prompts) - {"segments": [{"label": true|false, "text": "..."}]}
  - Sharegpt - {"conversations": [{"from": "...", "value": "..."}]}
- https://github.com/OpenAccess-AI-Collective/axolotl?tab=readme-ov-file#dataset

AXOLOTL CONFIG

- Templates for different models included in axolotl repo
- Will publish to HuggingFace after training
  - Card contains training info
- hub_model_id – the HuggingFace model name for your new model
- datasets – Huggingface data set path

```yaml
base_model: mistralai/Mistral-7B-v0.1
base_model_config: mistralai/Mistral-7B-v0.1
model_type: MistralForCausalLM
tokenizer_type: LlamaTokenizer
is_mistral_derived_model: true
hub_model_id: DragonMistral-7b

load_in_8bit: false
load_in_4bit: true
strict: false

datasets:
  - path: Akila/ForgottenRealmsWikiDataset
    data_files:
      - specific_formats/FRW-J-axolotl-completion.jsonl
    type: completion
dataset_prepared_path:
val_set_size: 0.02
output_dir: ./qlora-out

#using lora for lower cost
```

STEPS

- We will set up manually - better control
- Setup Runpod and install axolotl
  - Select RunPod Pytorch 2.1 template
  - Add custom ports 7860,5001 and increase disk sizes to 100 GP
  - Connect to Jupyter server
  - Open Terminal
  - git clone https://github.com/OpenAccess-AI-Collective/axolotl
  - cd axolotl
  - pip3 install packaging
  - pip3 install -e '.[flash-attn,deepspeed]'
- Run fine-tuning
  - accelerate launch -m axolotl.cli.train your_train_config.yml

PITFALLS AND RECOMMENDATIONS

- Running out of Container Disk and Volume Disk
  - Set to 100GB each (for a 7B model, this is plenty)
- For reference, for Mistral 7B
  - 1x H800 80GB
  - FRW-J (converted to JSONL)
  - ~6 hours of training
  - Didn't use 100% of VRAM
  - 24 CPUs were overkill (but can't adjust)
- For converting to GGUF (QLoRA)
  - Use llama.cpp's converter
  - https://github.com/ggerganov/llama.cpp/discussions/2948

PITFALLS AND RECOMMENDATIONS

- When converting to GGUF,
  - Q4 and Q5 models are preferred (optimal size vs. performance)
  - K_M models are preferred over K_S (e.g. Q5_K_M)
- Remember to add the quantization level info to the model file
  - E.g. Mistral-of-Realms-7b-v1.Q5_K_M.gguf
- For GGUF models create a new model repo on HuggingFace
  - E.g. Akila/Mistral-of-Realms-7b-gguf
  - Add all the different quantized versions to this repo
  - E.g. Mistral-of-Realms-7b-v1.Q5_K_M.gguf, Mistral-of-Realms-7b-v1.Q8_0.gguf
- Make sure to stop and delete the pod once done. Otherwise, you will be charged

REFERENCES

[1] Edward J Hu, 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.

[2] Tim Dettmers. 8-bit approximations for parallelism in deep learning. arXiv preprint arXiv:1511.04561, 2015.

[3] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. arXiv preprint arXiv:2212.09720, 2022.