Fine tuning LLMs, 

Insop, 11/10/2023

Take aways

- Why do we fine-tune
- How to fine-tune
- Things to consider

Outlines

🤖 LLM Overview

Why consider your own LLMs

Which LLMs to choose

Train your LLMs

Preparing Dataset for training

Evaluating LLMs

How to serve your own LLMs

LLM

API based LLM access

- Proprietary and Open source LLMs

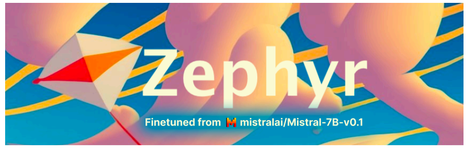
Approaches:

- Zero-shot, Few-show prompt
- Crafted prompt
- Retrieval augmented generation

ANTHROP\C

cohere

Meta Llama2, 🦙



* how_to_use_llm_talk

Outlines

LLM Overview

? Why consider your own LLMs

Which LLMs to choose

Train your LLMs

Preparing Dataset for training

Evaluating LLMs

How to serve your own LLMs

Why consider custom LLMs

Data residency and privacy

Intellectual property

Customization

Cost

Performance

Latency

Control over updates

Ownership

Transparency

Performance of Fine-tuned LLM

- How is the performance of fine-tuned LLM?
  - *It depends*
  - For a specific task, fine-tuning open source LLM can improve the performance. It could surpass closed models' capability
  - Depends on the complexity of task, degree of improvement can vary

OSS LLM Fine-tuning 🦙

- Finetuned llama2 outperform gpt-4 on **SQL generation** task

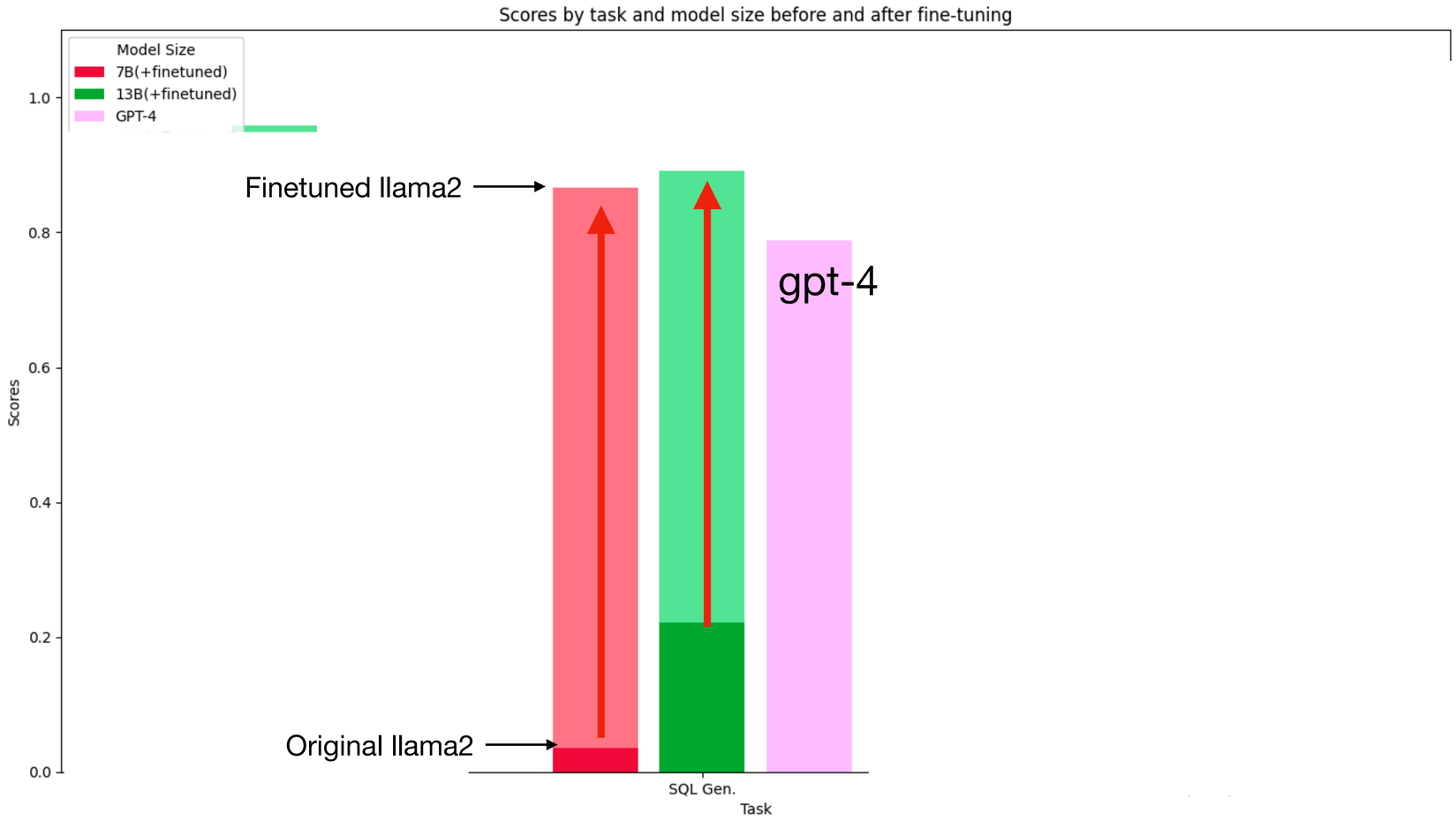


Image source: https://www.anyscale.com/blog/fine-tuning-llama-2-a-comprehensive-case-study-for-tailoring-models-to-unique-applications

OSS LLM Fine-tuning 🦙

- Finetuned llama2 still behind gpt-4 on GSM8k (grade school math)

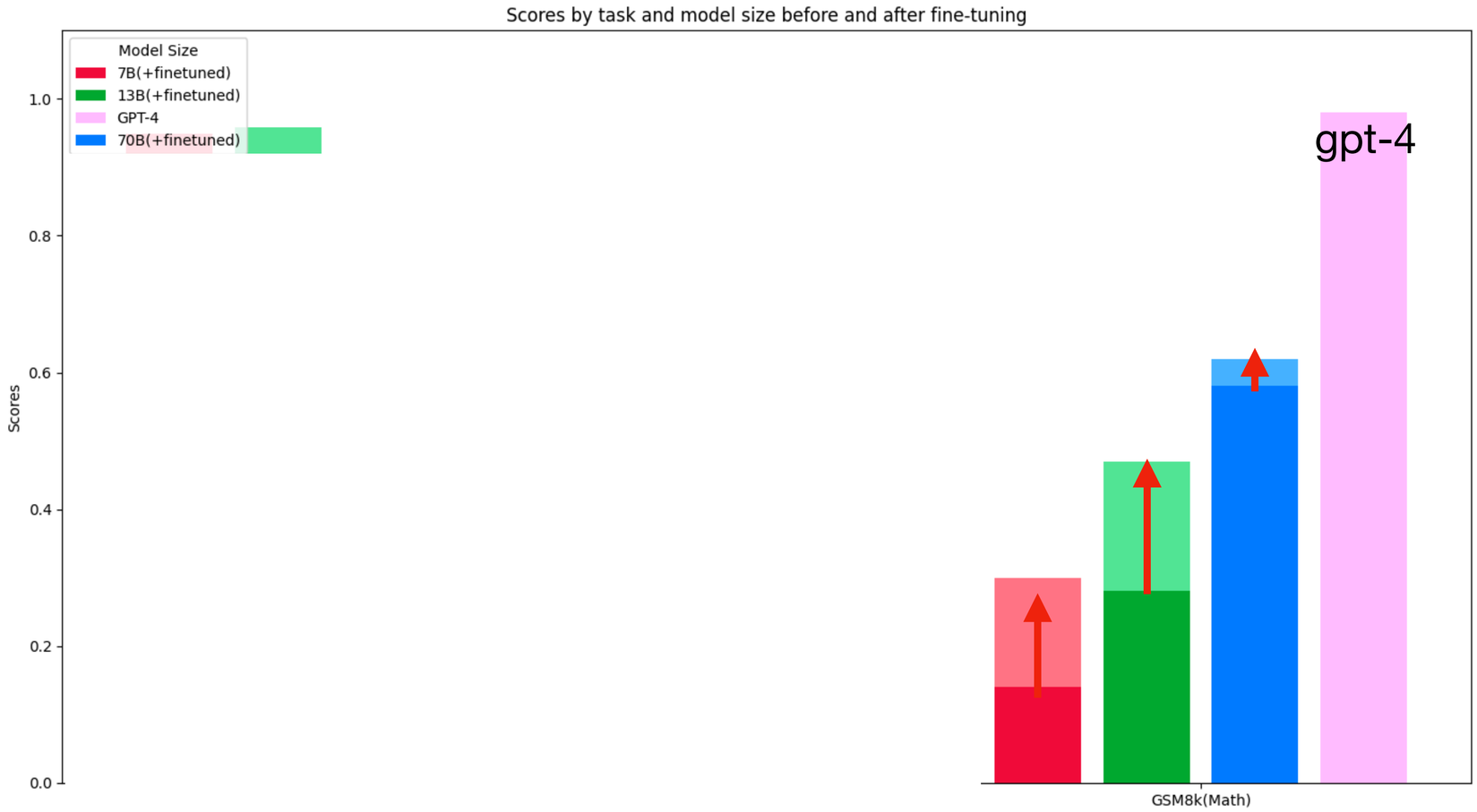


Image source: https://www.anyscale.com/blog/fine-tuning-llama-2-a-comprehensive-case-study-for-tailoring-models-to-unique-applications

What type of tasks?

- Start with simple method first and gradually move the more involved methods
  - In context learning
  - RAG:
    - When dealing with factual data
  - Fine tuning
    - Need to improve few shot behavior, domain adaptation
    - Reduce the known prompt patterns

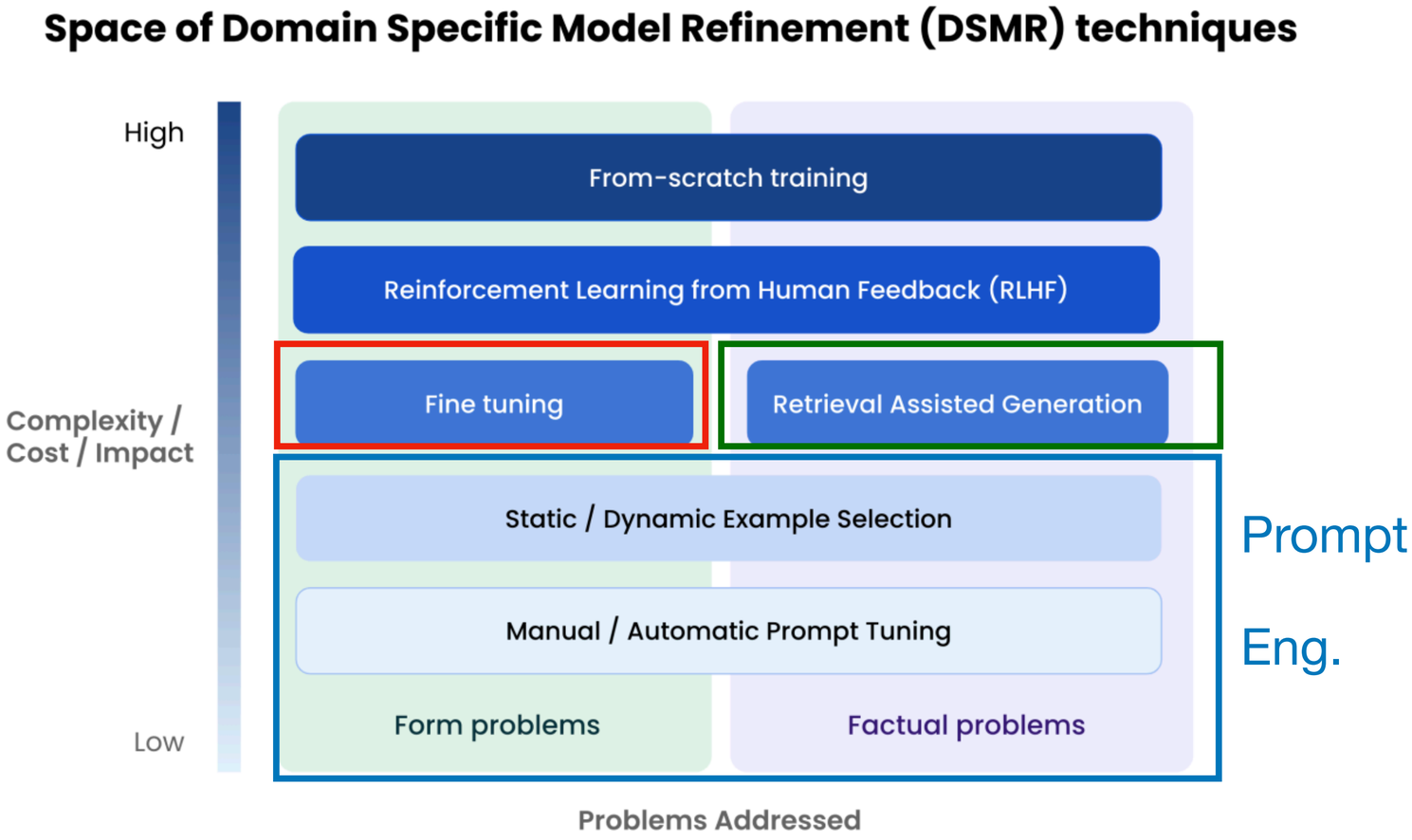


Image source: https://www.anyscale.com/blog/fine-tuning-is-for-form-not-facts

Customization options

Fine-tuning commercial LLMs

Fine-tuning OSS LLMs

OpenAI fine-tuning service*:

- Prepare and upload training data
- Train a new fine-tuned model
- Evaluate results and go back to step 1 if needed
- Use your fine-tuned model

OpenAI fine-tune dataset example

```
{"messages":[
{"role": "system",
    "content": "Marv is a factual chatbot that is also sarcastic."},
{"role": "user",
    "content": "What's the capital of France?"},
{"role": "assistant",
    "content": "Paris, as if everyone doesn't know that already."}
]}
```

*: GPT-3.5 Turbo
* GPT-4-turbo fine-tuning service is just started

https://platform.openai.com/docs/guides/fine-tuning
https://docs.cohere.com/docs/training-custom-models
https://www.bloomberg.com/company/press/bloomberggpt-50-billion-parameter-llm-tuned-finance/
https://blog.replit.com/llm-training
https://openai.com/pricing
https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset
https://openai.com/blog/openai-partners-with-scale-to-provide-support-for-enterprises-fine-tuning-models

Outlines

LLM Overview

Why consider your own LLMs

🦙 Which LLMs to choose

Train your LLMs

Preparing Dataset for training

Evaluating LLMs

How to serve your own LLMs

Which Open source LLMs

Target tasks

Requirements

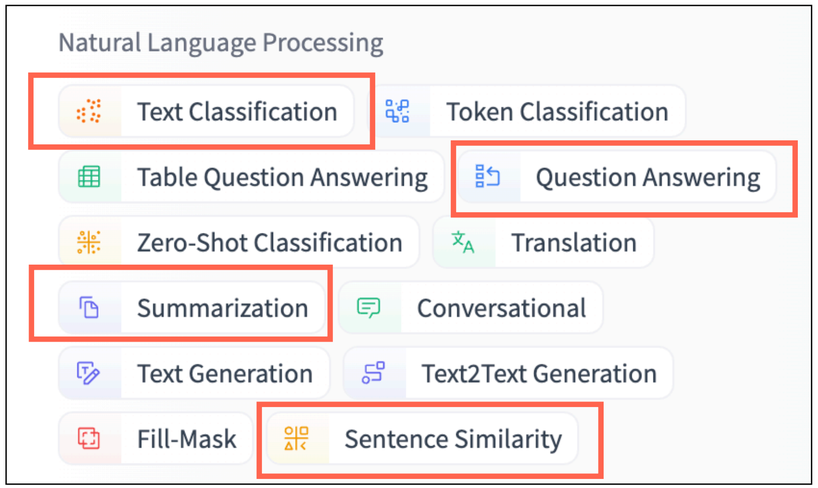
Model: size, structure, license

Dataset used

Training type: base, instruction-tuned

Evaluation: leaderboard

Iterative development



https://huggingface.co/models

Which LLMs

Start with smaller model first

Target tasks

Requirements

Model: size, structure, license

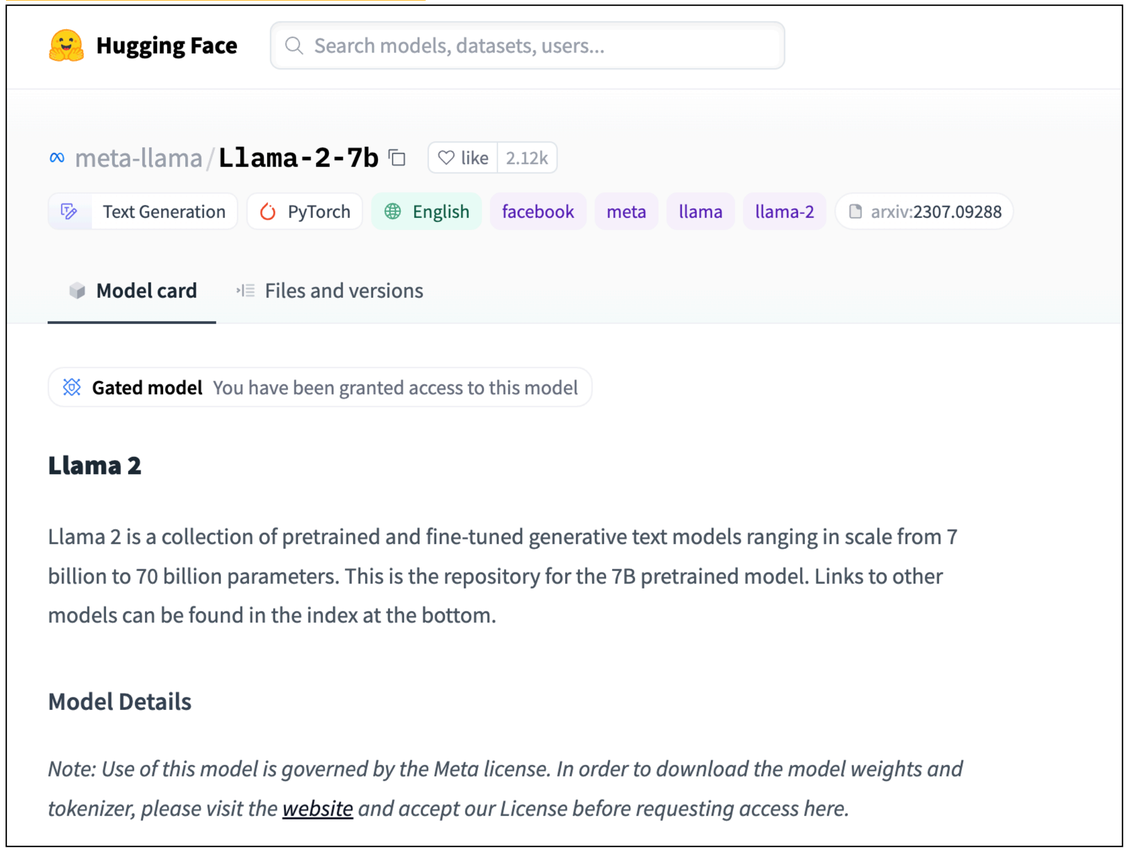
Dataset used

Training type: base, instruction-tuned

Evaluation: leaderboard

Iterative development

Example model card



https://huggingface.co/meta-llama/Llama-2-7b

Which LLMs

Target tasks

Requirements

Model: size, structure, license

Dataset used

Training type: base, instruction-tuned

Evaluation:

Iterative development

- See if the model is used in similar tasks
- Check the evaluations score if the evaluation is similar to your task
- Start with small model and small size of dataset
- Run fraction of epoch to measure the performance
- Increase model size and/or increase dataset
- Iterate

Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

🤗 Train your LLMs

Preparing Dataset for training

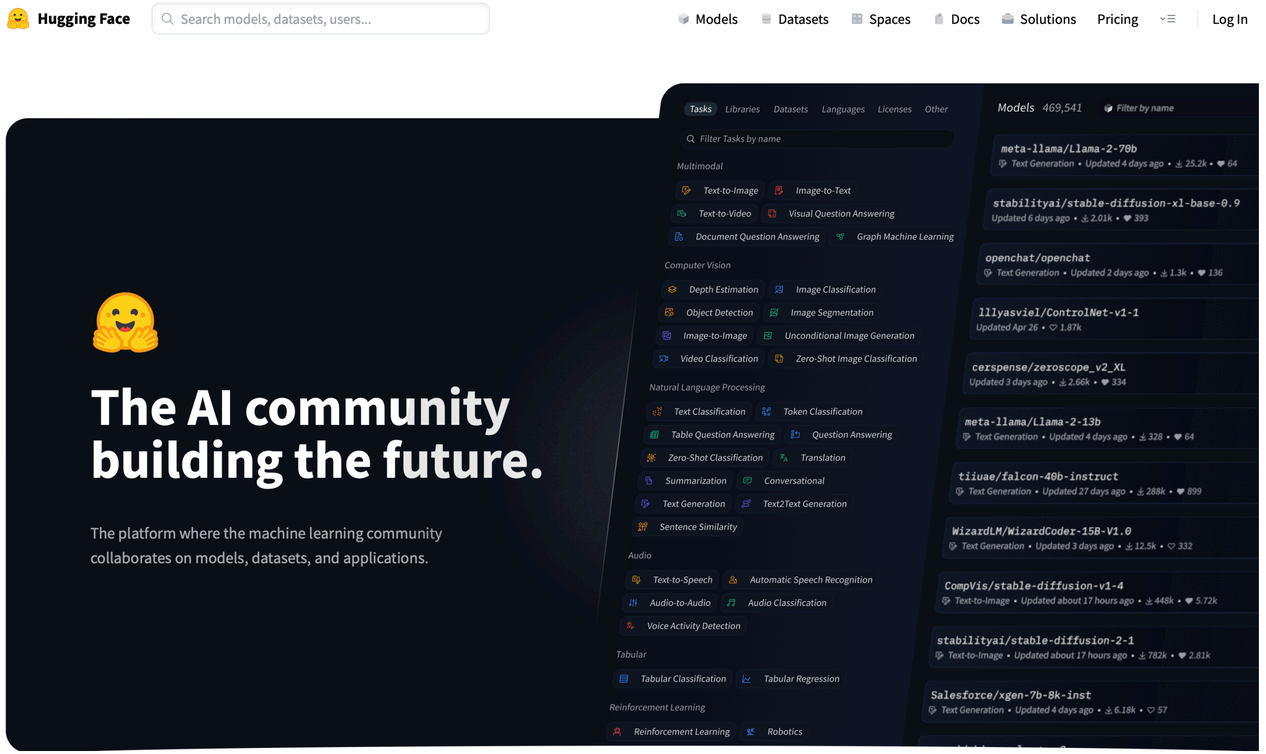
Evaluating LLMs

How to serve your own LLMs

Train your LLMs

Huggingface:

- One stop shop for LLMs
- Model repo
- Dataset repo
- LLM libraries
- Model serving
- Model training



There are other tools on top of huggingface to even simplify training, such as, Ludwig, Efficient Fine-Tuning for Llama-v2-7b on a Single GPU

Train your LLMs

Dataset preparation

Train your LLMs

Model training

- Start with examples*
- Simple training code
- Systematic experiments

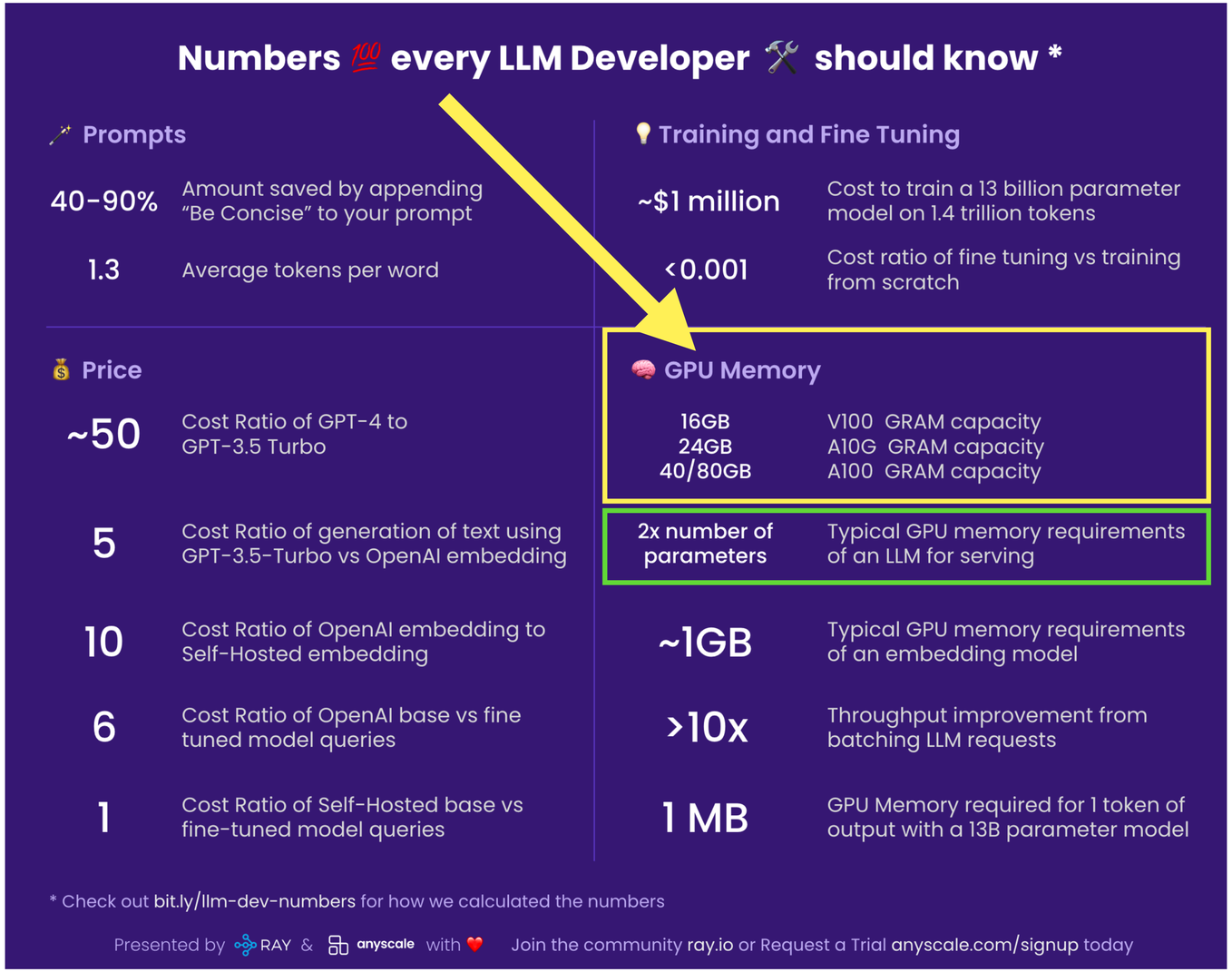
```
# load model
model = AutoModelForCausalLM.from_pretrained(
    model_path, device_map="auto",
)

# load tokenizer
tokenizer = AutoTokenizer.from_pretrained(
    model_path)

# load/set training arguments
training_arguments = TrainingArguments(...)

# load dataset
dataset = load_dataset(dataset_name)
train_data, val_data = dataset["train"], dataset["validation"]

# configure train
trainer = Trainer(
    model=model, args=training_args, train_dataset=train_data,
eval_dataset=val_data
)

# train!!!
trainer.train()
```

* see examples in reference section; due to rapid changes in libraries, you may need to change code/api depends on the library version you use

Compute resource

GPU *memory*:

- RTX GPUs: ~24 GB
- A10G GPU: 24GB
- A100 GPU: 40/80 GB

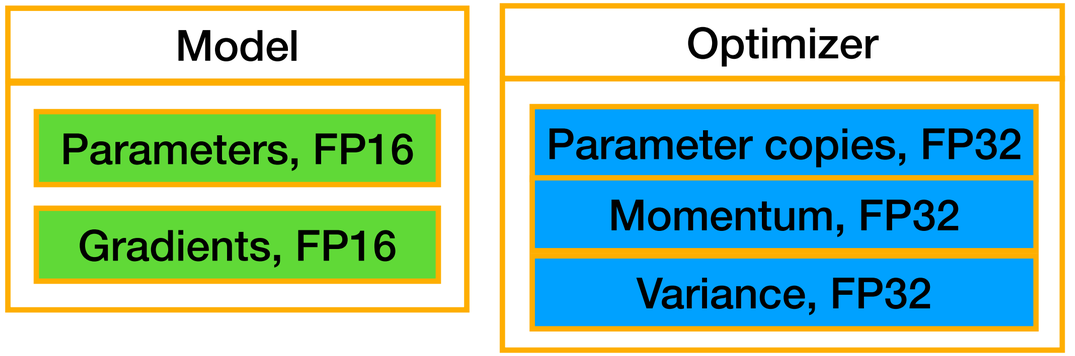


https://github.com/ray-project/llm-numbers

Param. & GPU memory

To train, need to store the followings on the GPU memory

- model parameters, 2 Bytes (floating point 16, FP16)
- model gradients, 2 Bytes
- optimizer states (Adam), 12 Bytes



n : number of parameters

Lower bound of static memory using Adam optimizer:

- ***16*** *x n Bytes, (16=2+2+12)*

ex> llama 2-7b, *7 b param*. require ~ 112 GB GPU memory

This will not fit to 80GB memory on A100 GPU!

http://web.stanford.edu/class/cs224u/slides/sidd-fantastic-lms-cs224u.pdf
FP16: floating point 16 bit, 2bytes, FP32: 4 byes

Parameter efficient fine tuning (PEFT)

- train additional small subset, additive
- train small set of model, re-parameterization
- *Low rank adaptation (LoRa)*

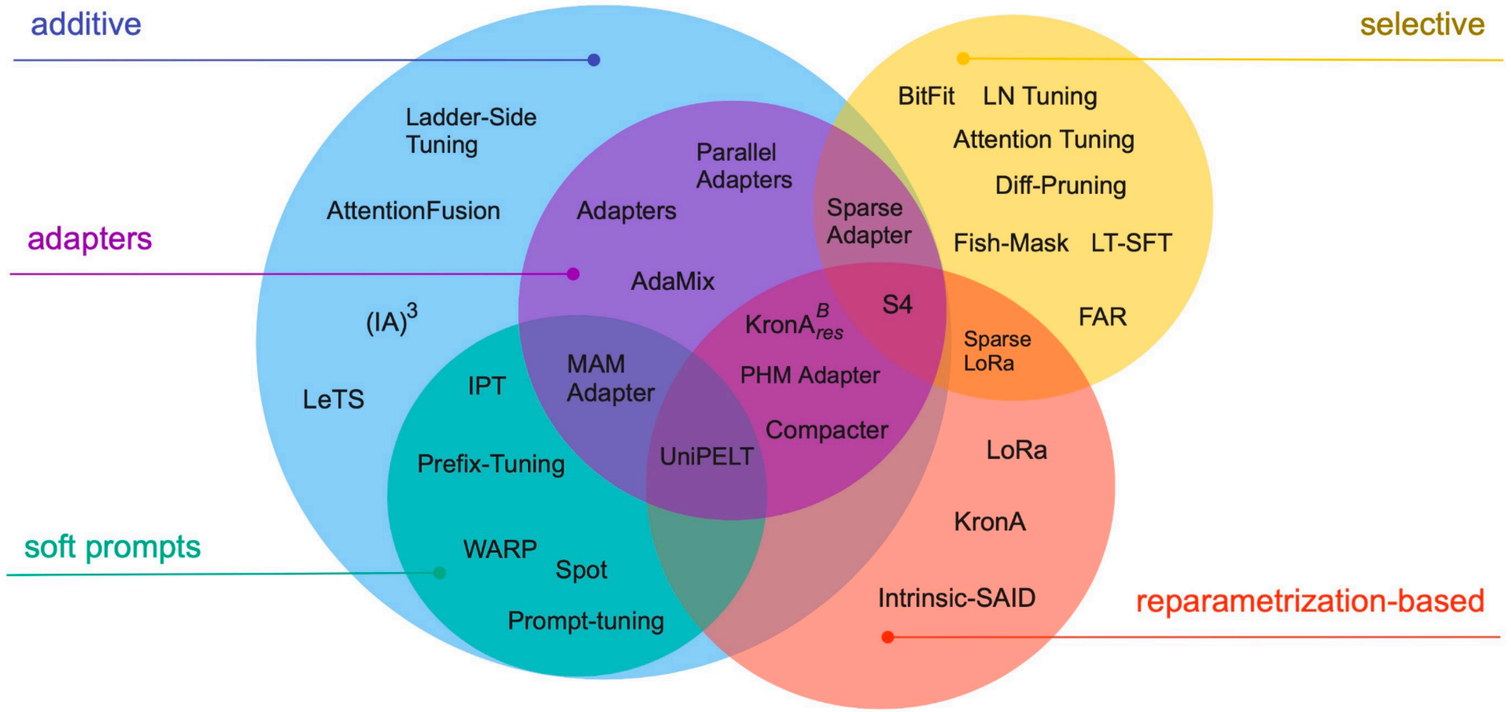
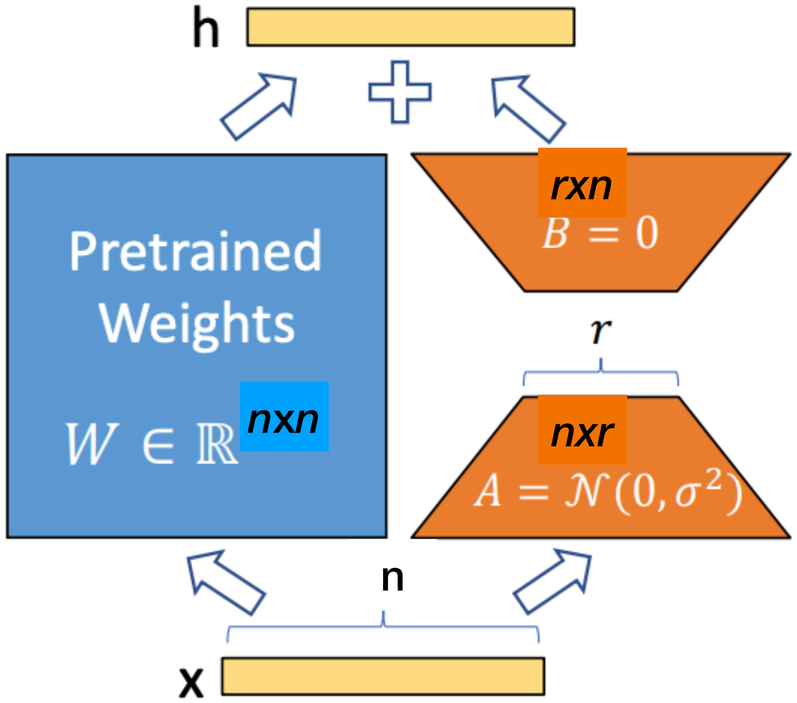


Image from Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning: https://arxiv.org/abs/2303.15647

🤗 https://huggingface.co/docs/peft/index

Low rank adaptation (LoRa)



Low rank adaptation (LoRa)

- Freeze pretrained weights
- Only train lower rank matrices

```
def lora_linear(x):
    """LoRA
    W: nxn, frozen param
    W_A: nxr, W_B: rxn, where r < n, learnable param
    """
    h = x @ W # regular linear
    h += x @ W_A @ W_B # low-rank update
    return scale * h
```

Image from Lora: https://arxiv.org/pdf/2106.09685.pdf
🤗 https://huggingface.co/docs/peft/index

Low rank adaptation (LoRa)

- Rank configuration
- target_module: which modules to use

```
peft_config = LoraConfig(
    r=args.lora_r,
    lora_alpha=args.lora_alpha,
    lora_dropout=args.lora_dropout,
    target_modules=modules,
    bias="none",
    task_type=TaskType.CAUSAL_LM,
)

model = get_peft_model(model, peft_config)
```

Trainable parameter with lora rank 8 w.r.t original number of trainable parameters: 0.3 %

Memory required llama 2 7b model

- 112GB = 16x7

- *14GB is for keep the frozen weight*

=> (112*0.003 + 14) GB ~ 14 GB

Full fine-tuning vs LoRA

- How is the performance of LoRA compared to full fine-tuning?
  - It *depends*, but small
  - Simple task (**SQL** generation)
    - Small performance difference

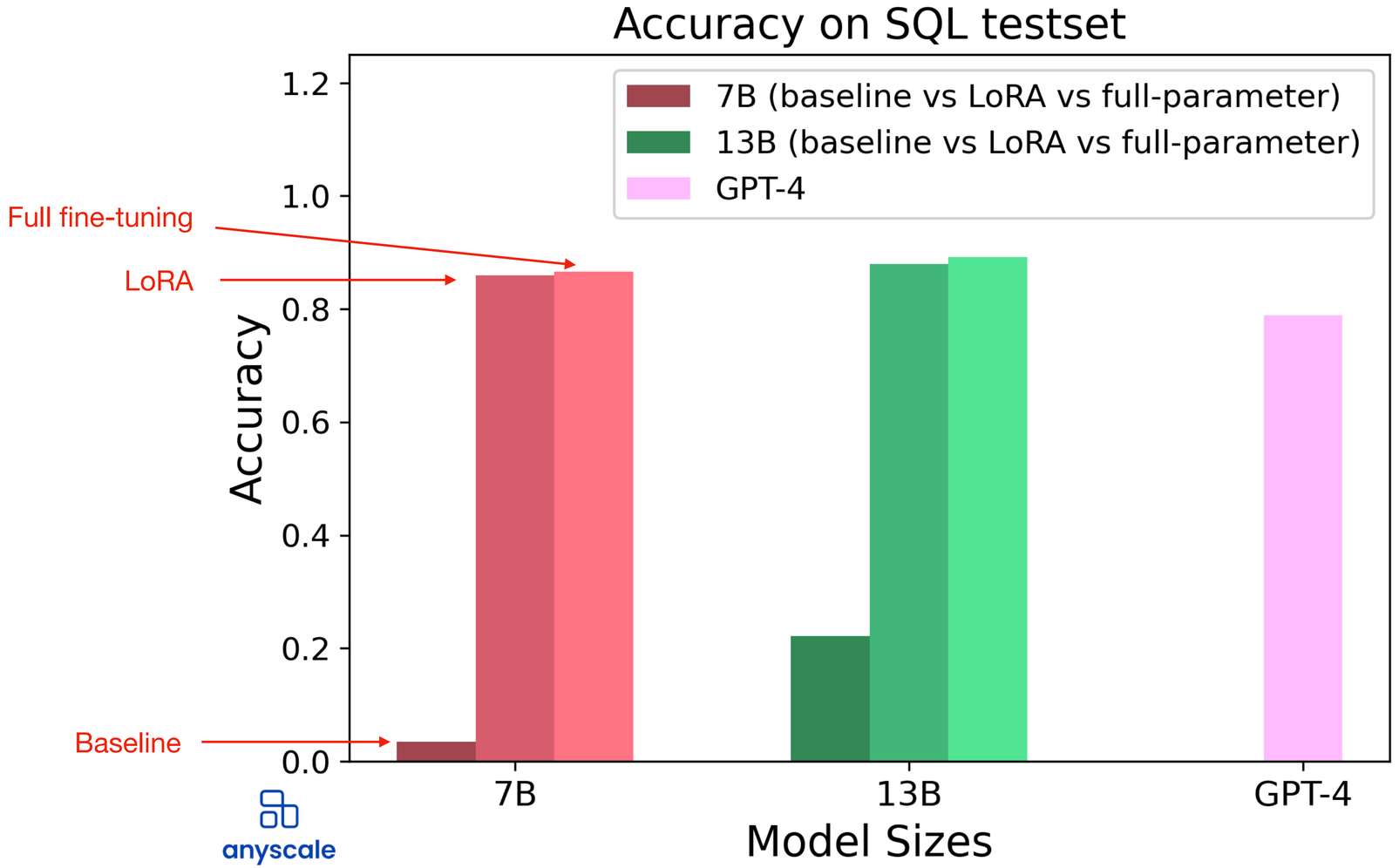


Image from: https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-llama-2

Full fine-tuning vs LoRA

- Complex task (**GSM8K**, grade school math)
  - Large performance difference in smaller model
  - small performance difference in larger model

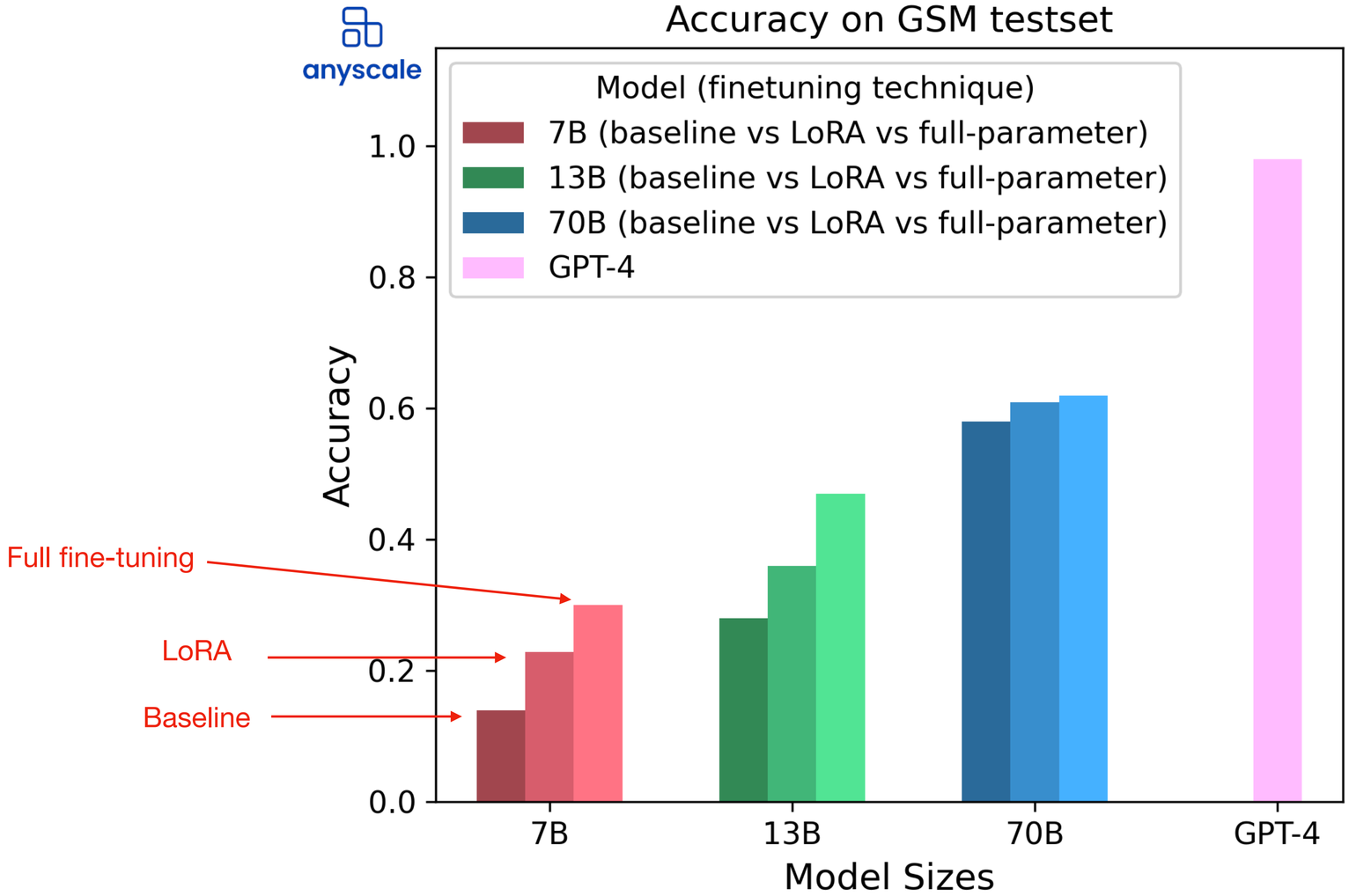


Image from: https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-llama-2

Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

Train your LLMs

 Preparing Dataset for training

Evaluating LLMs

How to serve your own LLMs

Dataset preparation

Dataset for domain adaptation

Dataset for instruction tuning

- Prepare text dataset
- Verify dataset manually
- Apply filtering & dedup
- Continuous pre-training
- Next token prediction objective

Dataset preparation

Dataset for domain adaptation

Dataset for instruction tuning

- Prepare Q&A pair dataset
- Allows chatgpt-like behavior after this training
- High quality dataset is important
- Suggested number of dataset varies from 100 to 1,000 or more. So test incrementally based on the your own results
- Self-instruct method: use LLM to generate/ filter/improve Q&A pairs for dataset. There are many dataset follows this idea.

https://platform.openai.com/docs/guides/fine-tuning
LIMA, less is more, https://arxiv.org/pdf/2305.11206.pdf
Self-instruct: https://arxiv.org/abs/2212.10560
dolly-dataset:https://www.databricks.com/blog/2023/03/24/hello-dolly-democratizing-magic-chatgpt-open-models.html
Codellama2: https://ai.meta.com/research/publications/code-llama-open-foundation-models-for-code/

*: frozen model only used to generate text
**: model under training

Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

Train your LLMs

Preparing Dataset for training

 Evaluating LLMs

How to serve your own LLMs

Evaluation

- Very important to have evaluations metric early
- Find out metric that can measure the performance of the task
- LLM based evaluation can be effective

Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

Train your LLMs

Preparing Dataset for training

Evaluating LLMs

🏃🦙How to serve your own LLMs

How to serve LLMs

Post train model conversion

End point serving

- GPTQ: post train quantization (8, 4, 3, 2-bit precision) using the GPTQ algorithm
- AutoGPTQ: library that applies GPTQ for LLMs
  - Huggingface integrates AutoGPTQ library, which makes it easier to use

GPTQ: https://arxiv.org/pdf/2210.17323.pdf
AutoGPTQ:https://github.com/PanQiWei/AutoGPTQ/tree/main
🤗 https://huggingface.co/blog/gptq-integration

How to serve LLMs

Post train model conversion

End point serving

- 🤗 tgi (text generation inference)
  - LLM serving tool support
  - Support most OSS models
  - Feature rich tool
- vllm
  - Easy, Fast, and Cheap LLM Serving with PagedAttention
  - Performance number is very promising

tgi: https://huggingface.co/text-generation-inference
vllm: https://vllm.ai/

Summary

OSS LLMs for custom LLM use cases

Based on requirements, find OSS LLMs using 🤗 model hub

Use huggingface or similar framework to train model

Use PEFT method to reduce compute requirements

Prepare raw dataset and/or instruction following dataset, based on the application requirements.

Evaluate early and improve the evaluation metrics as needed

Serve LLM as API end point

Questions?

References

- Insop, how_to_use_llm_talk
- Anyscale, Fine-Tuning LLMs: LoRA or Full-Parameter? An in-depth Analysis with Llama 2
- Jeremy Howard, A Hackers' Guide to Language Models
- Andrej Karpathy, Let's build GPT: from scratch, in code, spelled out.
- Ludwig, Efficient Fine-Tuning for Llama-v2-7b on a Single GPU
- Huggingface, Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA
- Huggingface, Training a Llama in your backyard
- QLoRA example, Fine-tune LLaMA 2 (7-70B) on Amazon SageMaker

References

- AWS w/ *deeplearning.ai*, Generative AI with Large Language Models
- *Databricks, EdX*, Large Language Models: Foundation Models from the Ground Up
- *Databricks, EdX*, Large Language Models: Application through Production
- *deeplearning.ai*, Finetuning Large Language Models
- *Databricks*, Data+AI Summit 2023
- *AWS*, Generative AI Foundations on AWS Technical Deep Dive Series
- Anyscale, Fine-Tuning Llama-2: A Comprehensive Case Study for Tailoring Models to Unique Applications
- Anyscale, Fine tuning is for form, not facts

Customization options

Fine-tuning commercial LLMs

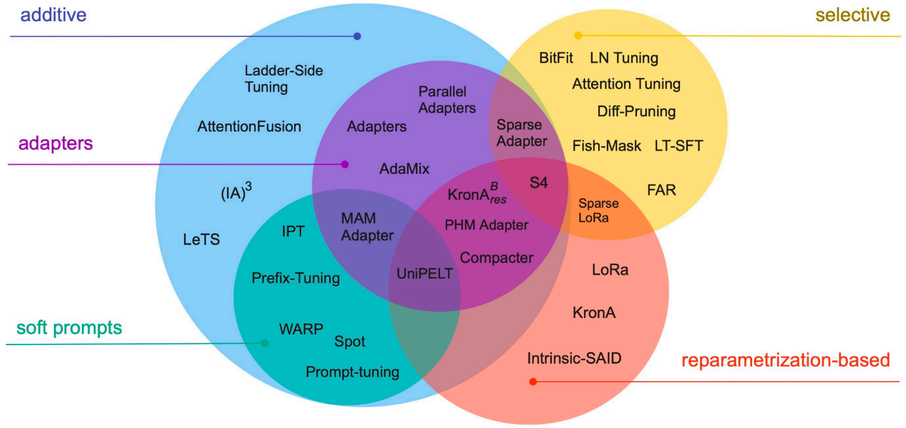
Fine-tuning OSS LLMs

Train your LLMs from scratch

- Requires significant engineering and budget
- Design model structure and scales
- Dataset preparation
- There are commercial services* to help pre-training process, but it is required to build in-house expertise
- Examples:
  - bloomberggpt
  - replit LLM

https://www.bloomberg.com/company/press/bloomberggpt-50-billion-parameter-llm-tuned-finance/
https://blog.replit.com/llm-training
*: https://www.mosaicml.com/
*: https://huggingface.co/training-cluster
*: openAI will work with selected customers to build custom model, additional domain specific pre-training, to running a custom RL post-training process

Train your LLMs



Parameter efficient fine tuning (PEFT)

- train additional small subset, additive
- train small set of model, re-parameterization

Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning: https://arxiv.org/abs/2303.15647

https://huggingface.co/docs/peft/index

Soft prompt tuning

- Concatenate trainable parameters to the input embeddings
- Consist with task description

```
def soft_prompted_model(input_ids):
    """Soft prompt tuning
    soft_prompt is a learnable parameter
    """
    x = Embed(input_ids)
    x = concat([soft_prompt, x], dim=seq)
    return model(x)
```

Low rank adaptation (LoRa)

- Freeze pretrain weight
- Only train lower rank matrices

```
def lora_linear(x):
    """LoRA
    W: nxn, frozen param
    W_A: nxr, W_B: rxn, where r < n, learnable param
    """
    h = x @ W # regular linear
    h += x @ W_A @ W_B # low-rank update
    return scale * h
```