Big Models, Small Tweaks: Exploring the LoRA Way of Fine-Tuning

Preethi Srinivasan, Shruti Dhavalikar

Sahaj Software

Into the talk..

- The concept
- Deep dive into fine tuning
- LoRA
- Set the toy stage
- Fine Tuning with LoRA
- Model sharing
- Limitations of LoRA

The Concept

Do you know them??

ChatGPT

Llama

Gemini

CLIP

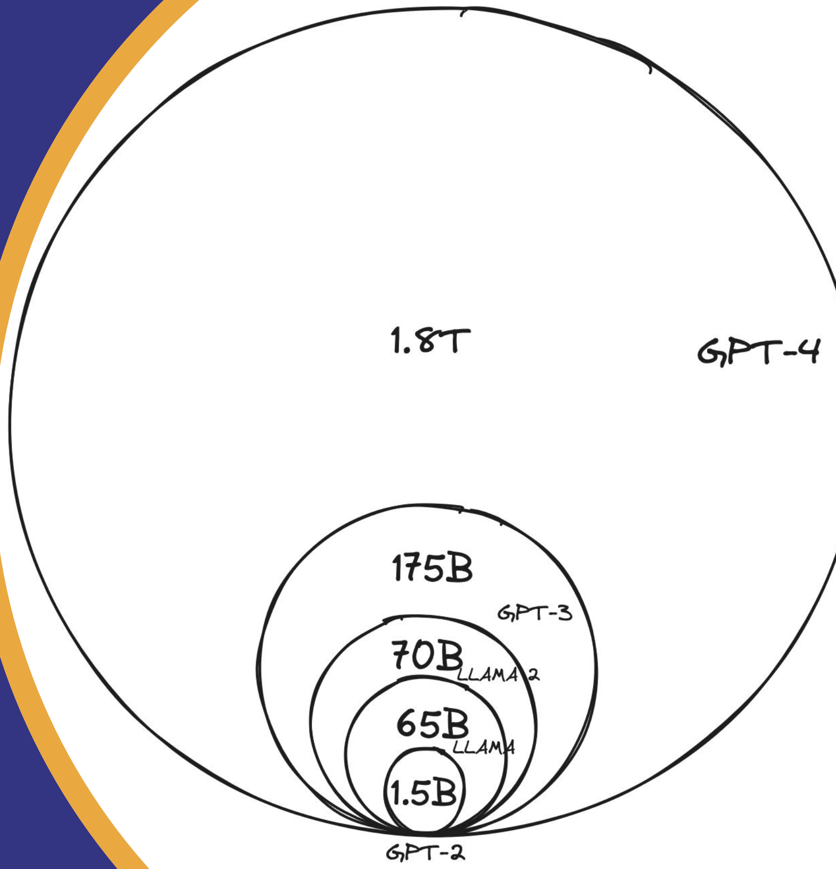
Mistral

Claude

Phi

Source:
https://accelerationeconomy.com/wp-content/uploads/2023/04/Generative-AI-Requires-Human-Intelligence-and-Creativity-scaled-e1682537728608.jpeg

The
Large Language Models



Source: Obsidian

LLMs are huuuge..

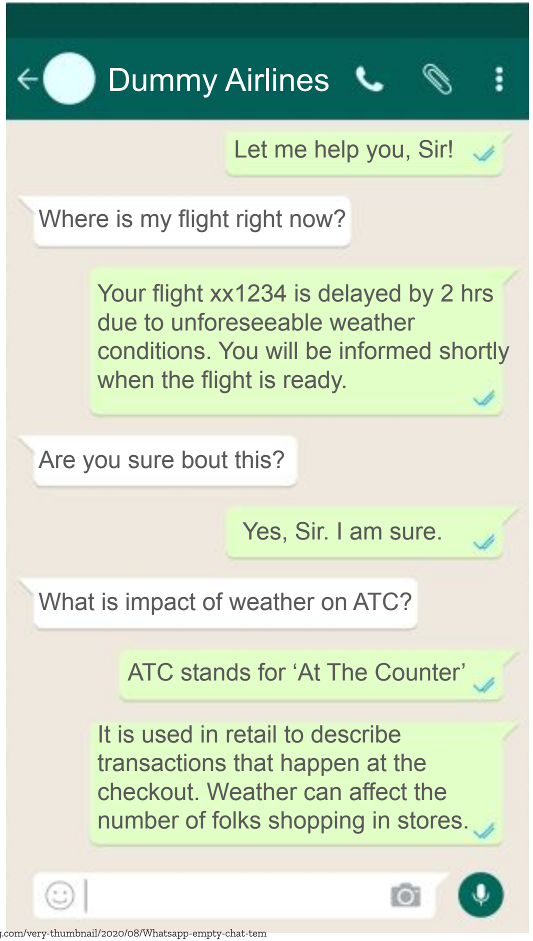
- *Large memory requirement*
- *Let's assume we are rich, then go ahead!*

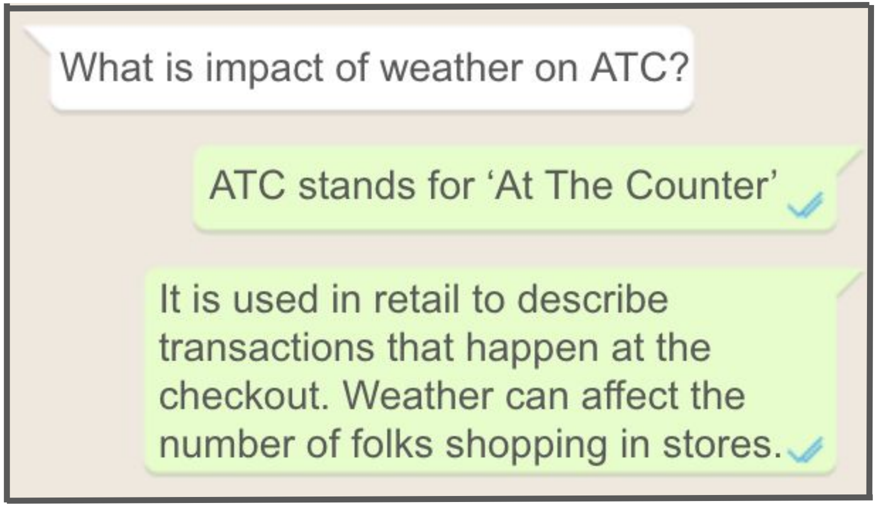
Memory required for LLaMa weights

# of parameters (B)	GB of RAM (float32s)	GB of RAM (float16s)	GB of RAM (int8s)	GB of RAM (int4s)
7	28	14	7	3.5
13	52	26	13	6.5
32.5	130	65	32.5	16.25
65.2	260.8	130.4	65.2	32.6

Source: https://gifdb.com/images/high/really-big-larry-david-7npr76q6tkyslc4i.gif
https://finbarr.ca/static/images/llama-memory-weights.png

Do they always work?





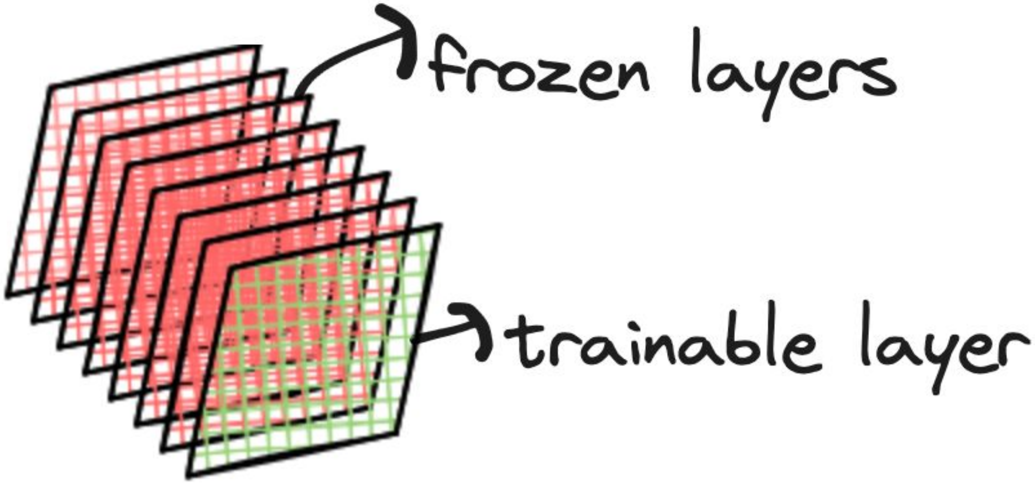
- **Prompts are not exhaustive!**
- **Fine tuning is not off the chart**

Source:
https://image.similarpng.com/very-thumbnail/2020/08/Whatsapp-empty-chat-template-vector-PNG.png

Deep dive into fine tuning

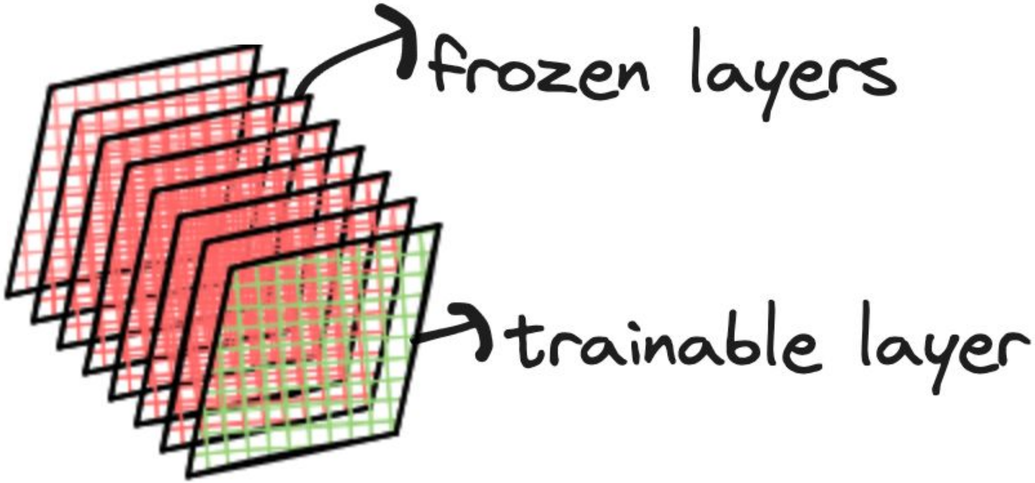
What is the conventional way of fine tuning?

- Layers == matrix of numbers
- Train all/some layers



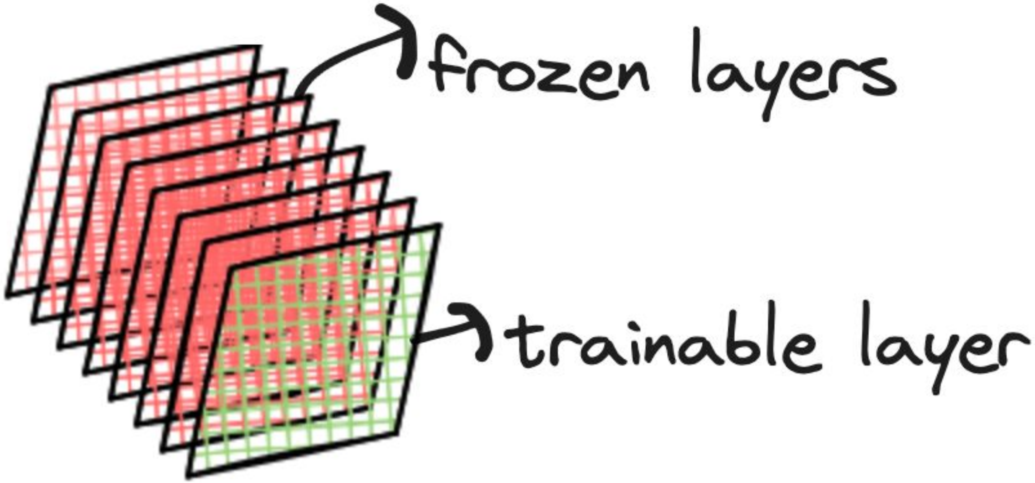
Conventional fine-tuning in the era of large models

- Llama 3.1 - 8B
- 32 Layers
- Each layer has 218M params



Conventional way is Resource and Memory intensive!

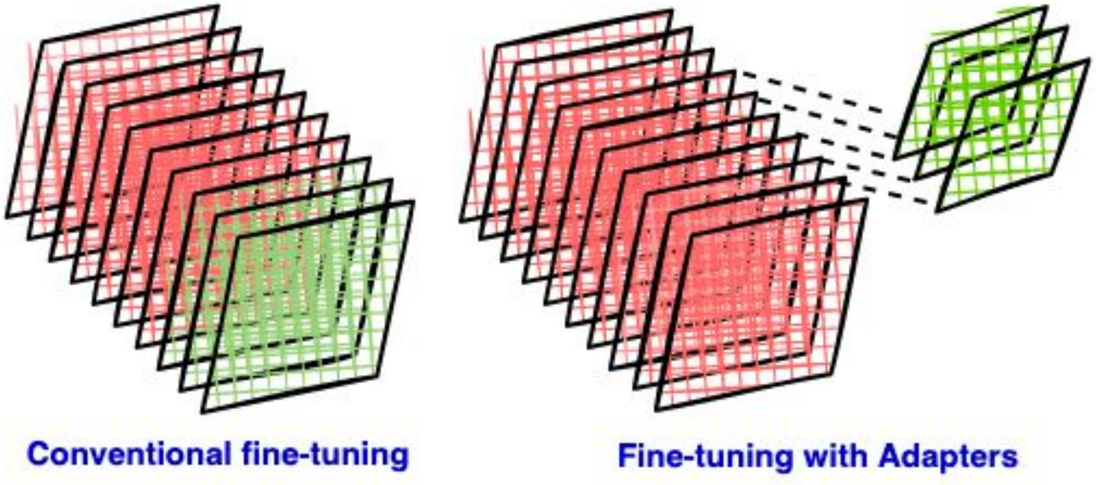
- **Memory**: 8B params = 8 x 10^9 x 4 Bytes/param
- **Resource**: 218M params = 218 x 10^6 x 4 Bytes/param x 3 (gradients, 2 moments)



Can fine tuning be made more efficient?

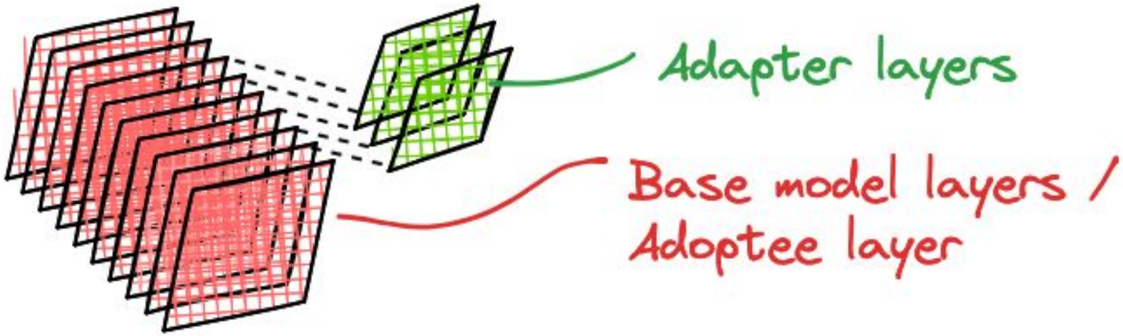
Can fine-tuning be made more efficient?

- Parameter Efficient Fine Tuning(PEFT)
- Adapter based PEFT!
- Learn a few extra parameters
- Less memory requirement



Fundamentals of PEFT

- Adapters vs Adoptees
- Small in size
- Initialization should not disrupt the training process

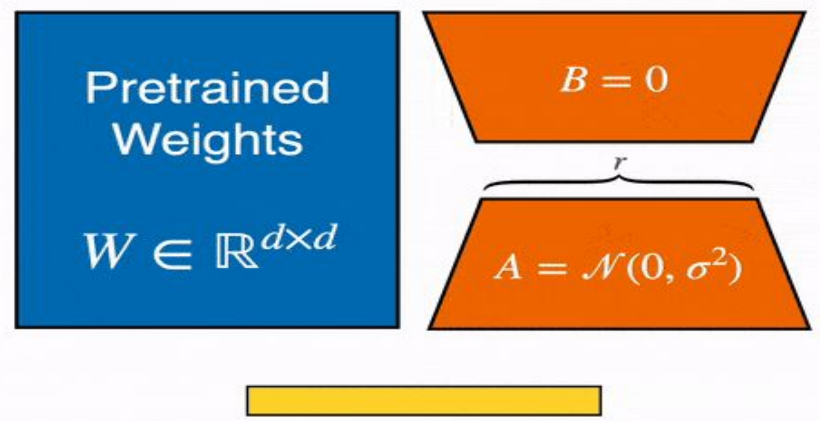


Fine-tuning with Adapters

LoRA

Analysing LoRA

- *Low Rank Adaptation(LoRA) Fine tuning*



Validating LoRA through its implementation

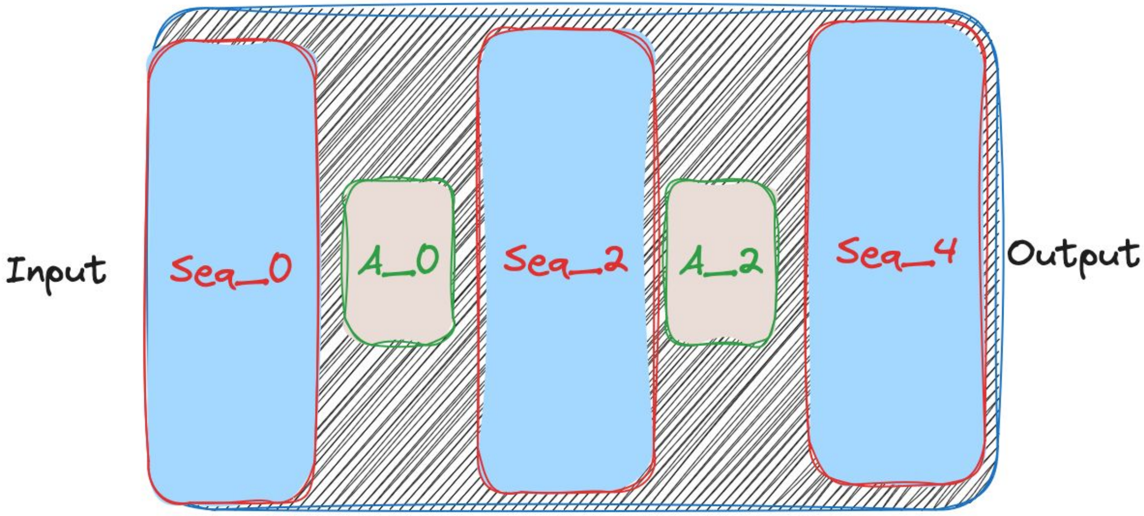
Source: https://imgflip.com/i/944607

Moving to the code

How to inject adaptors?

Sequential

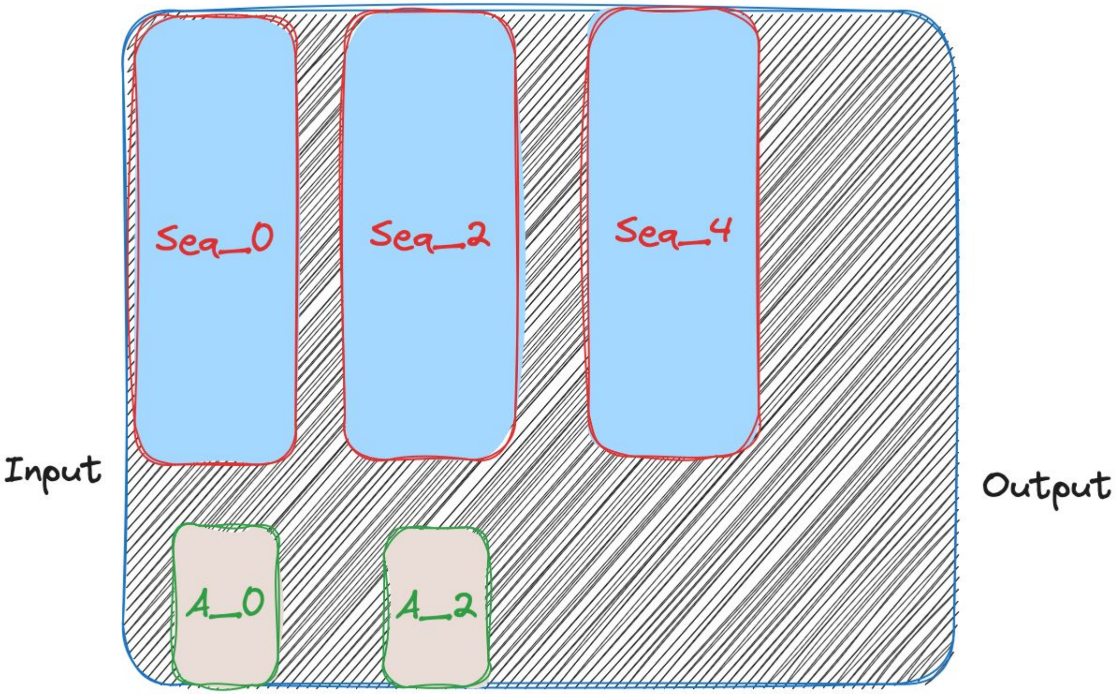
- GPUs memory won't be fully utilized.
- Training and Inference time is longer.



Source: https://medium.com/towards-data-science/dive-into-lora-adapters-38f4da488de

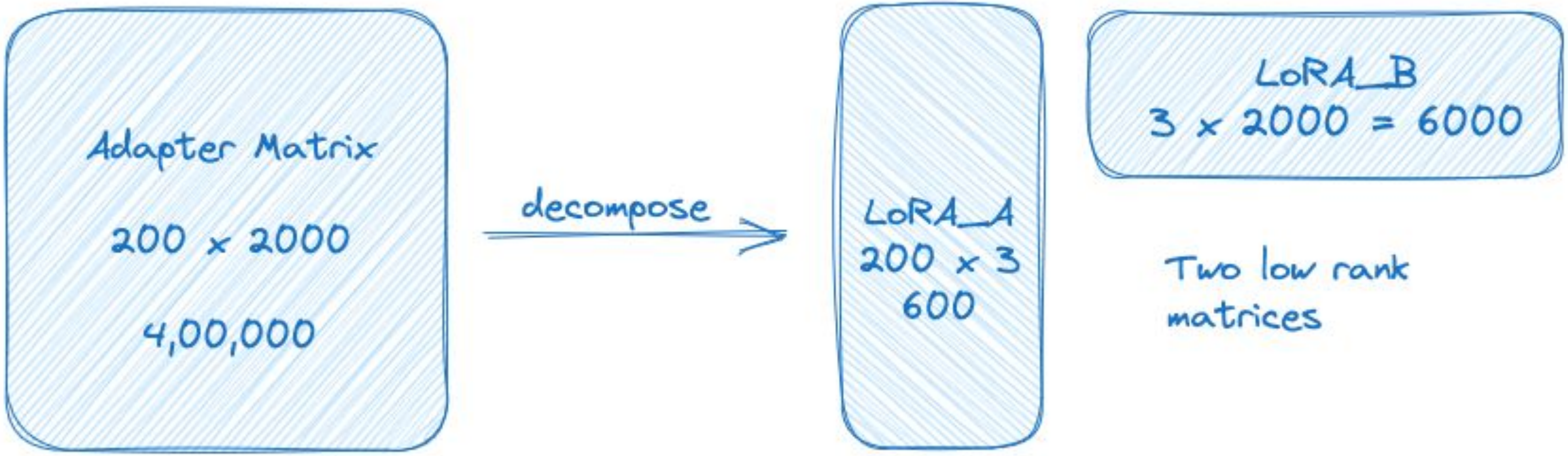
How to inject adaptors?

LoRA proposed : Parallel



Source: https://medium.com/towards-data-science/dive-into-lora-adapters-38f4da488ede

Adapters Before and After LoRA

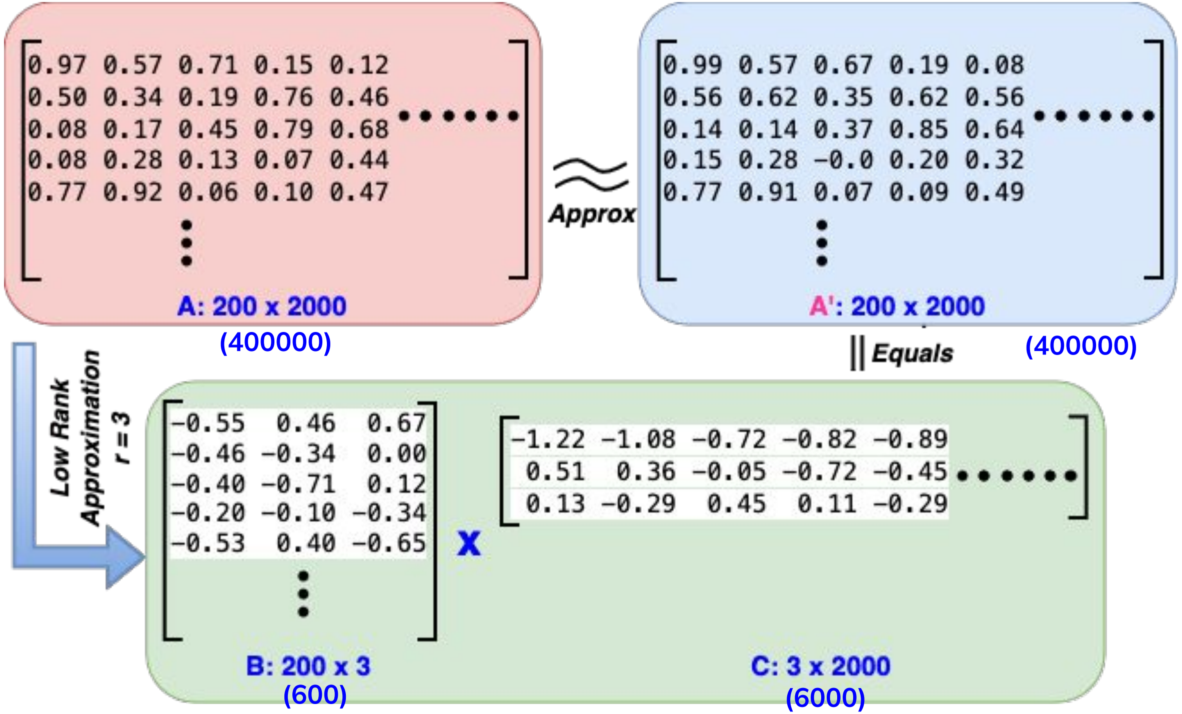


- *Smaller matrix still full rank matrix*

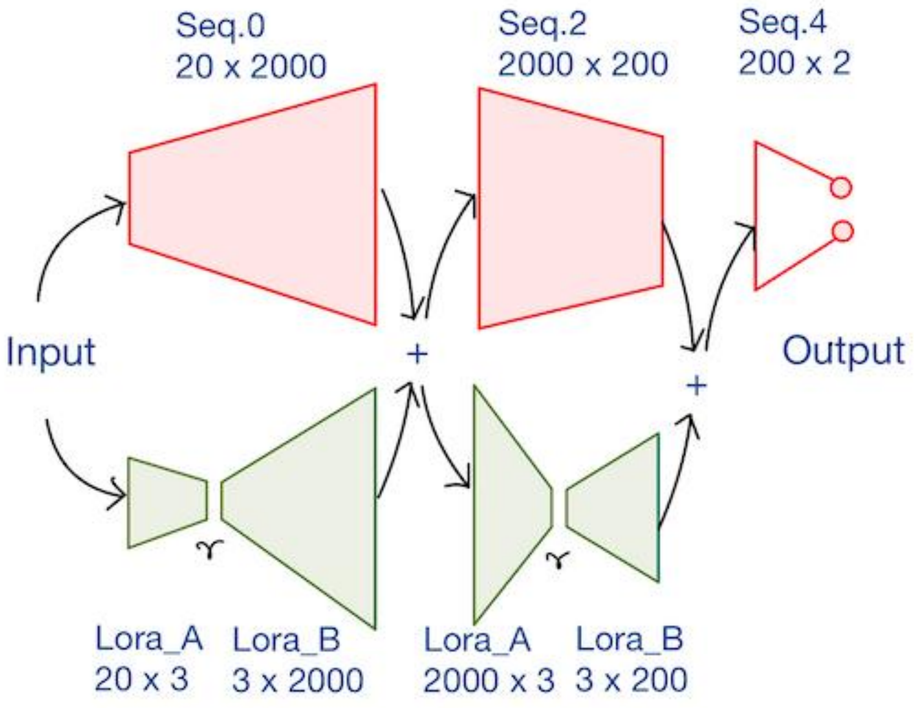
Idea behind LoRA?

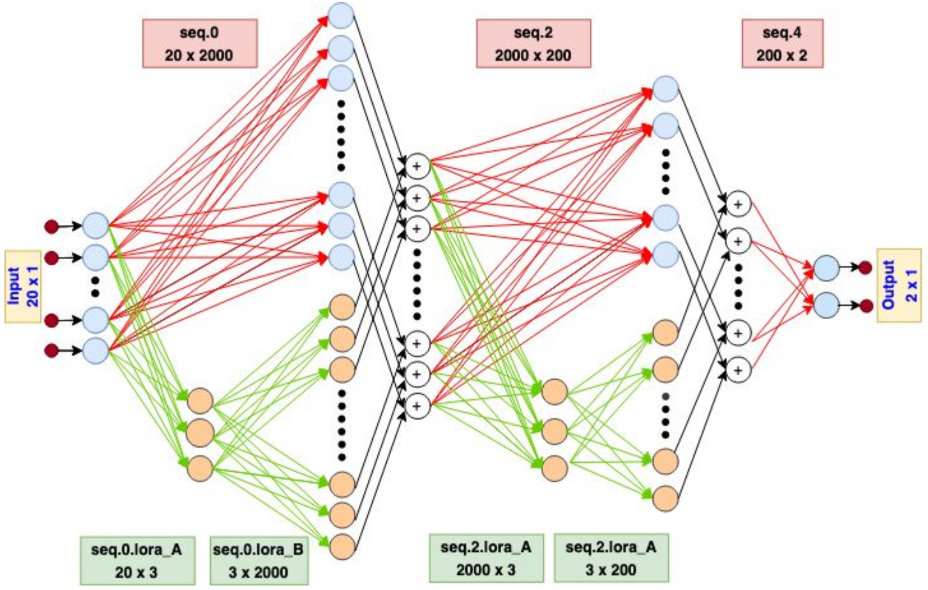
Recap: SVD (Singular Value Decomposition)

- SVD identifies B and C for a given A and r.
- LoRA learns B and C, for a given specific downstream task.



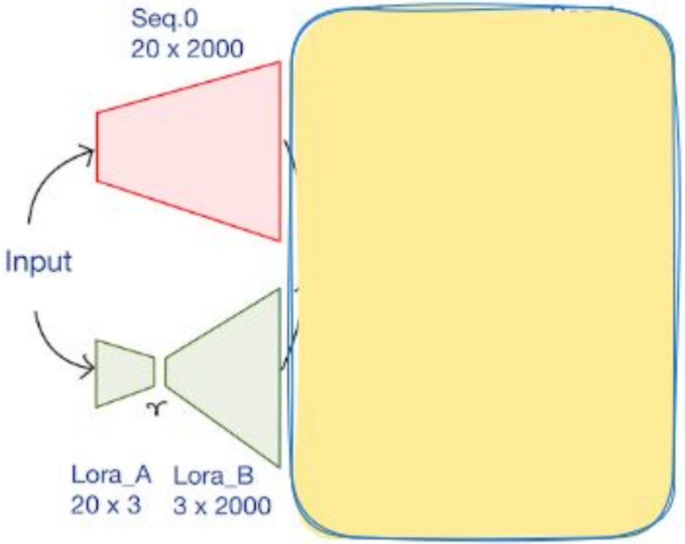
How did LoRA design the fine tuning architecture?





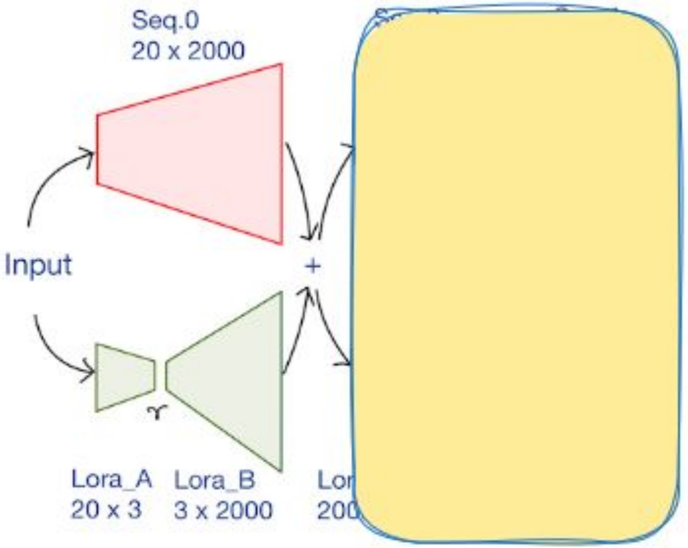
How does the forward pass look like?

```python
def forward(x):
  seq.0_out = seq.0(x)
  lora_A_out = seq.0.lora_A(x)
  lora_B_out = seq.0.lora_B(lora_A_out)
```



How does the forward pass look like?

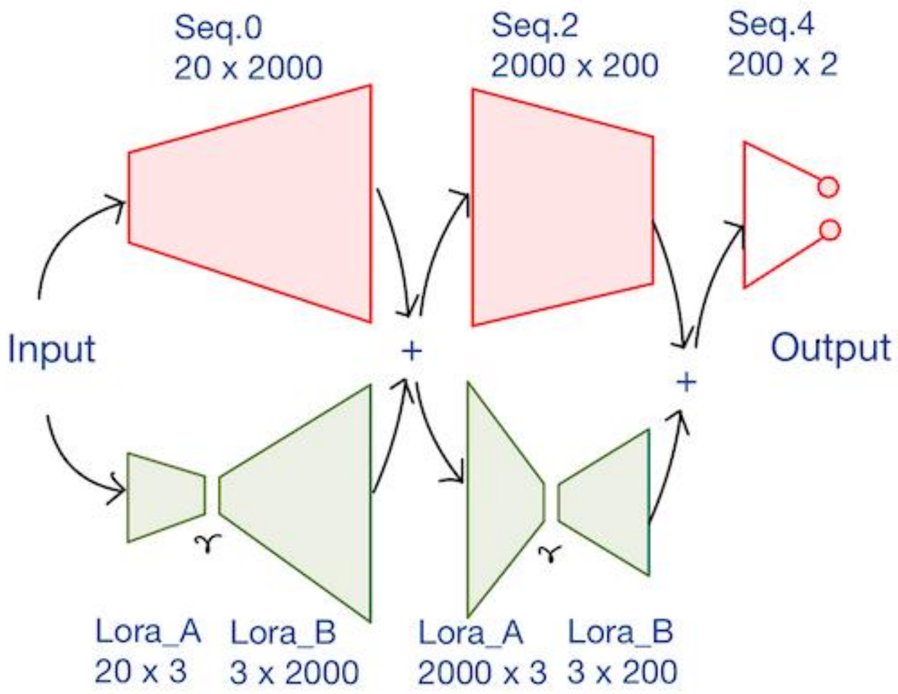
```
lora_B_out = lora_B_out * alpha
seq.0_lora_out = seq.0_out + lora_B_out
seq.0_lora_out = ReLU(seq.0_lora_out)
```



Alpha decides how much influence should the fine-tuning have on the pretrained models.

How does the forward pass look like?

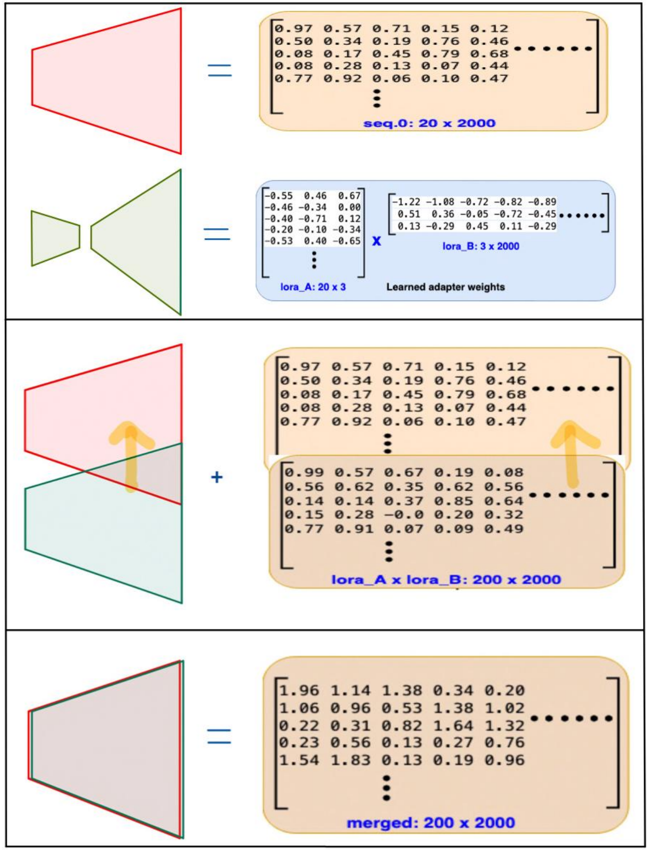
```
def forward(x):
  seq.0_out = seq.0(x)
  lora_A_out = seq.0.lora_A(x)
  lora_B_out = seq.0.lora_B(lora_A_out)
  lora_B_out = lora_B_out * alpha
  seq.0_lora_out = seq.0_out + lora_B_out
  seq.0_lora_out = ReLU(seq.0_lora_out)

  # Repeat for seq.2
  seq.2(seq.2.lora_out)
  ...
```



Moving to notebook

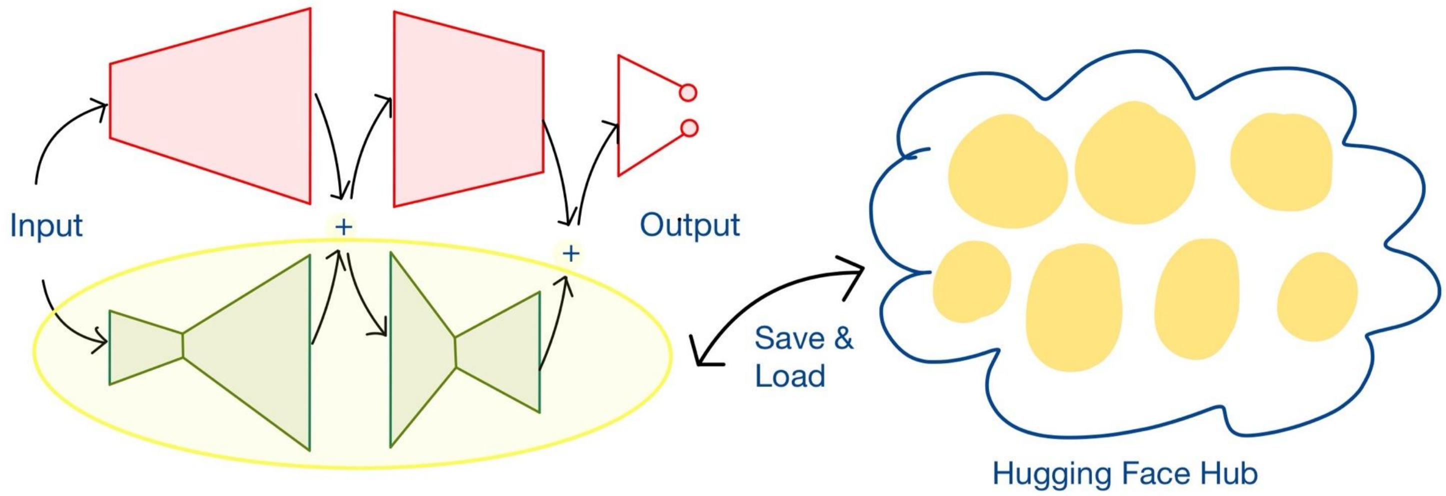
Merging the adaptors

- Expanded network - Additional adapter matrices.
- LoRA adapters are strategically designed to merge with adoptee matrices.



Moving to notebook

Sharing the model through HF hub



Performance

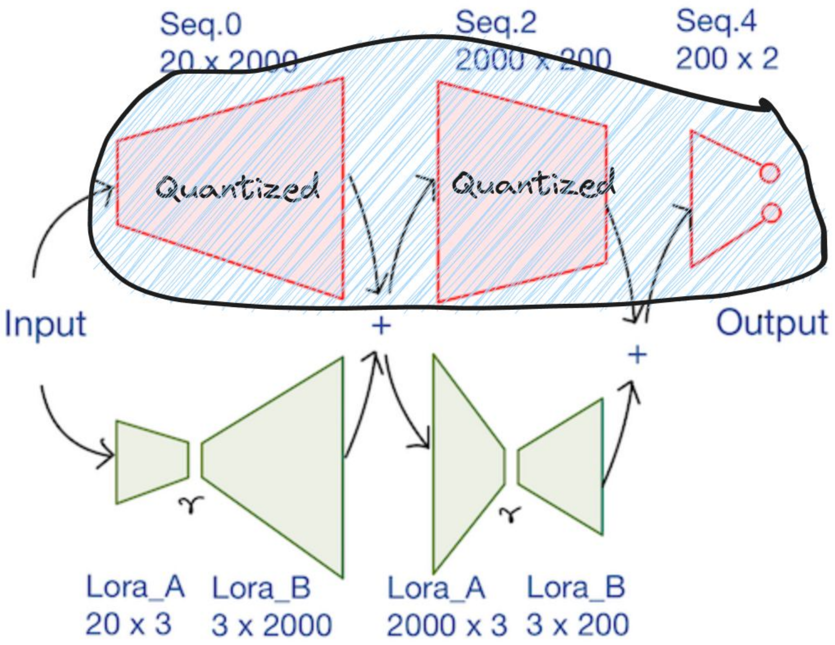
- Storage efficient : Llama-8B ; 5M LoRA params ; r = 2
- Compute efficient
- Amplification factor (A)
- A (r == 2) > A(r == 64)

Limitations

- Multi tasking - One adapter per task - Cannot load multiple adapters for batch with multiple tasks
- Memory Requirement - Need both base model (GBs) and adapter (MBs) for finetuning and inference.

Potential Solution

QLoRA (Quantized LoRA)



References