QLoRA

LLM fine-tuning made accessible

Shaw Talebi

Fine-tuning (recap)

Tweaking an existing model for a particular use case.

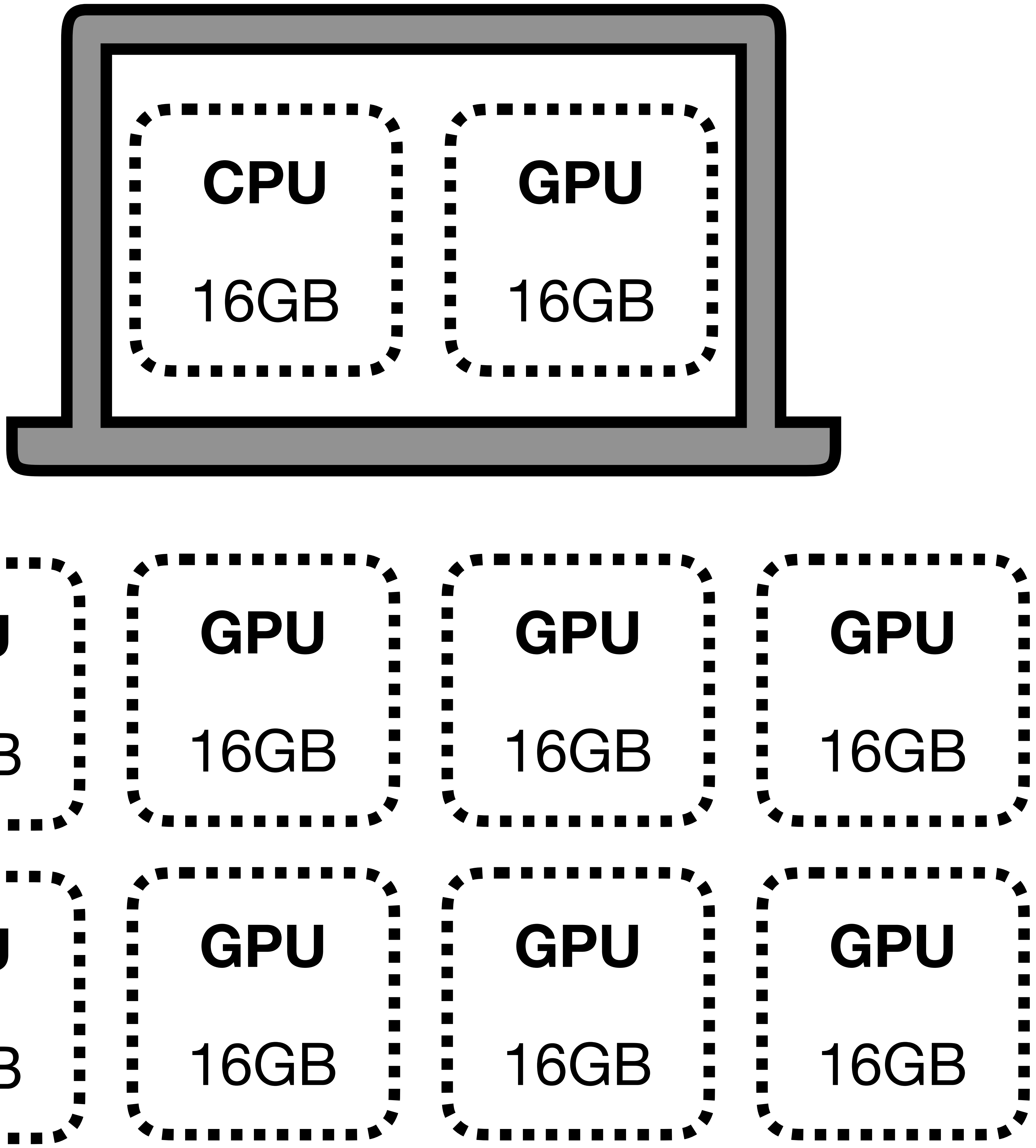
GPT-3

ChatGPT

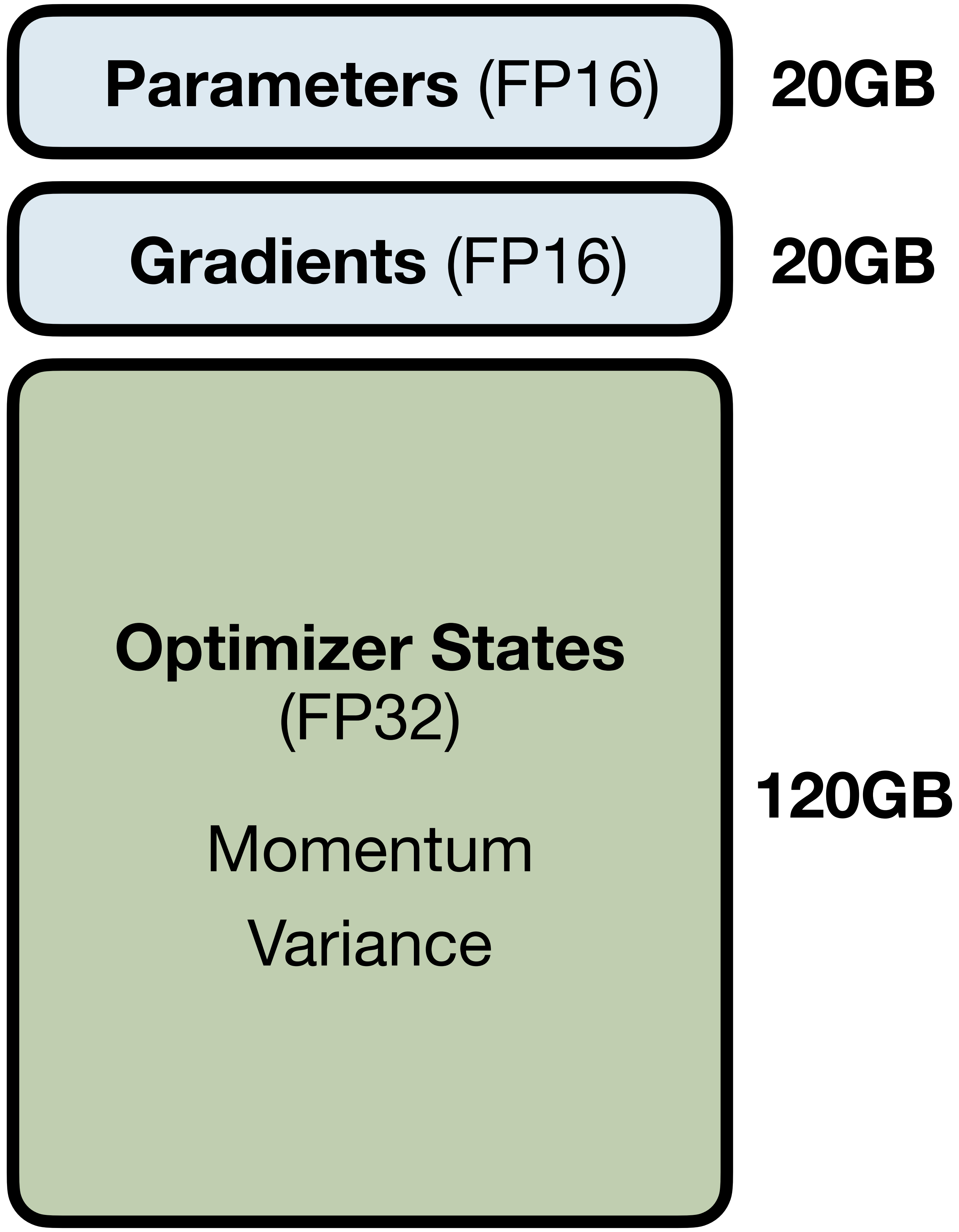
[1]

The Problem

LLMs are (computationally) expensive



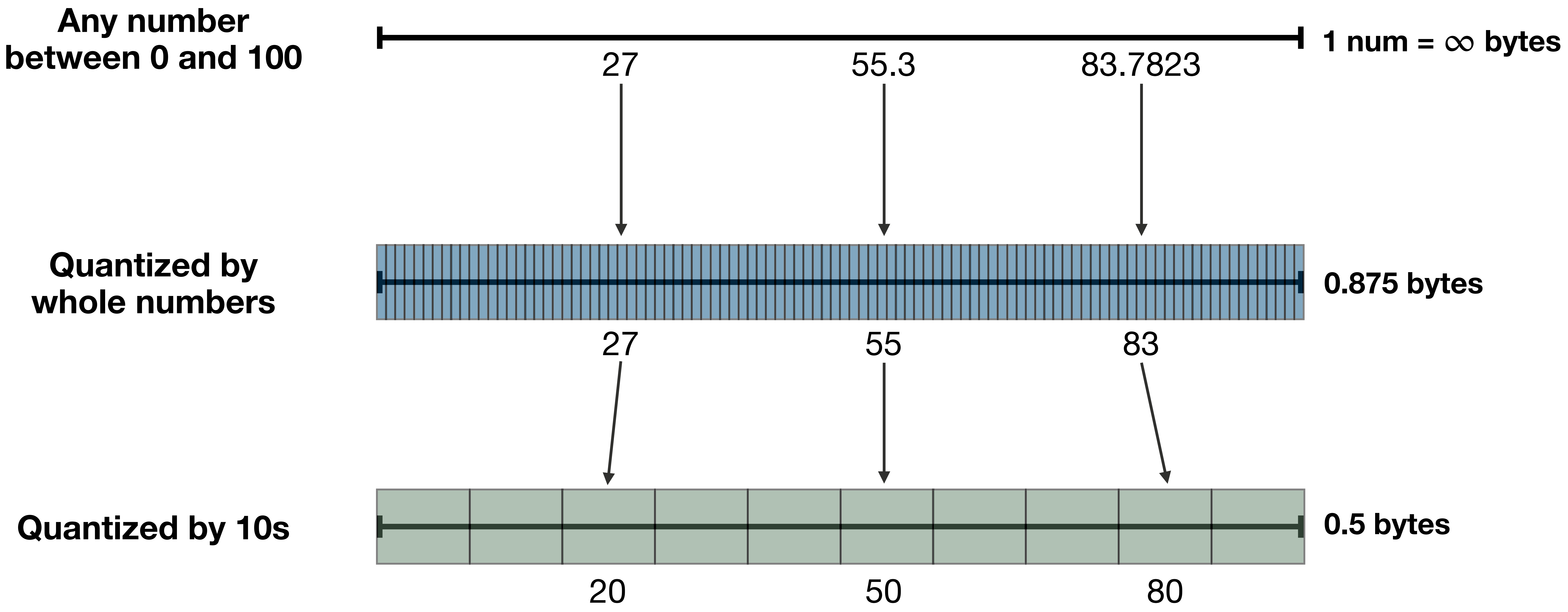
10B Parameter Model = 160GB!



 [2]

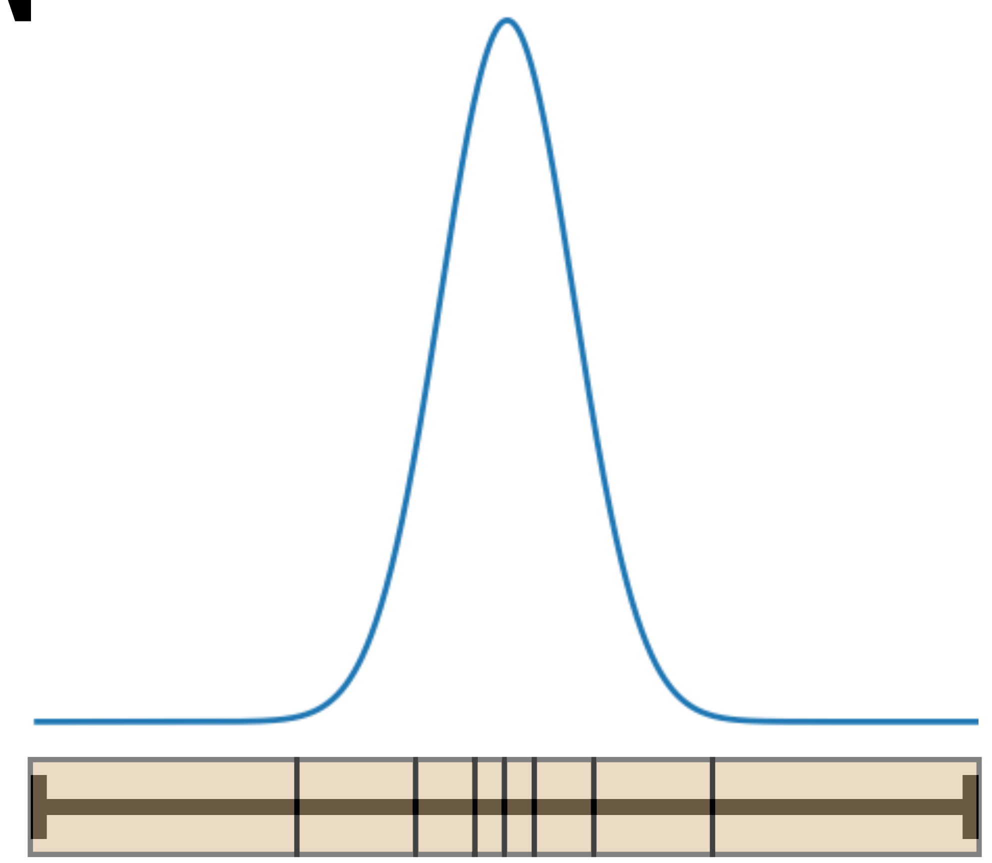
What is Quantization?

Quantization = splitting range into buckets



4 Ingredients of QLoRA

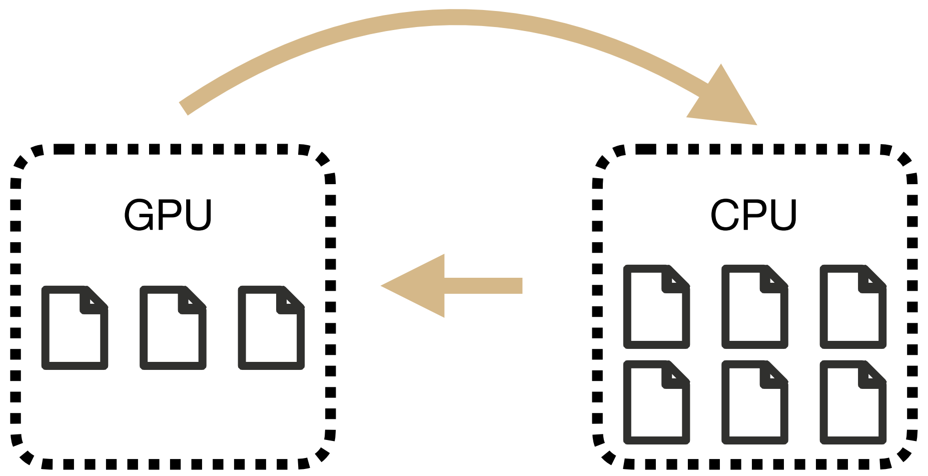
1. 4-bit NormalFloat



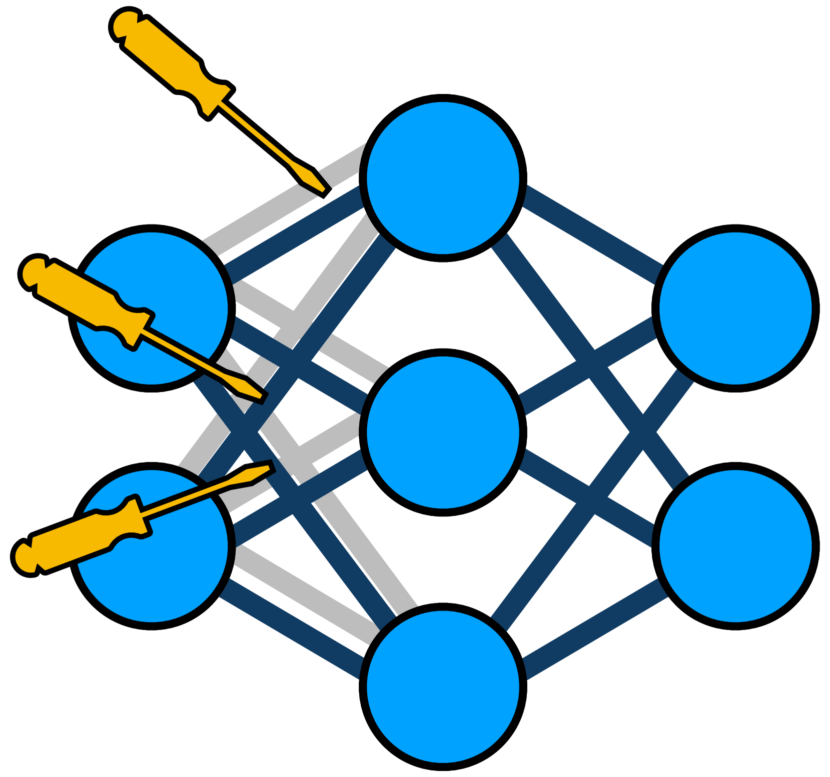
2. Double Quantization

quant(quant())

3. Paged Optimizers



4. LoRA



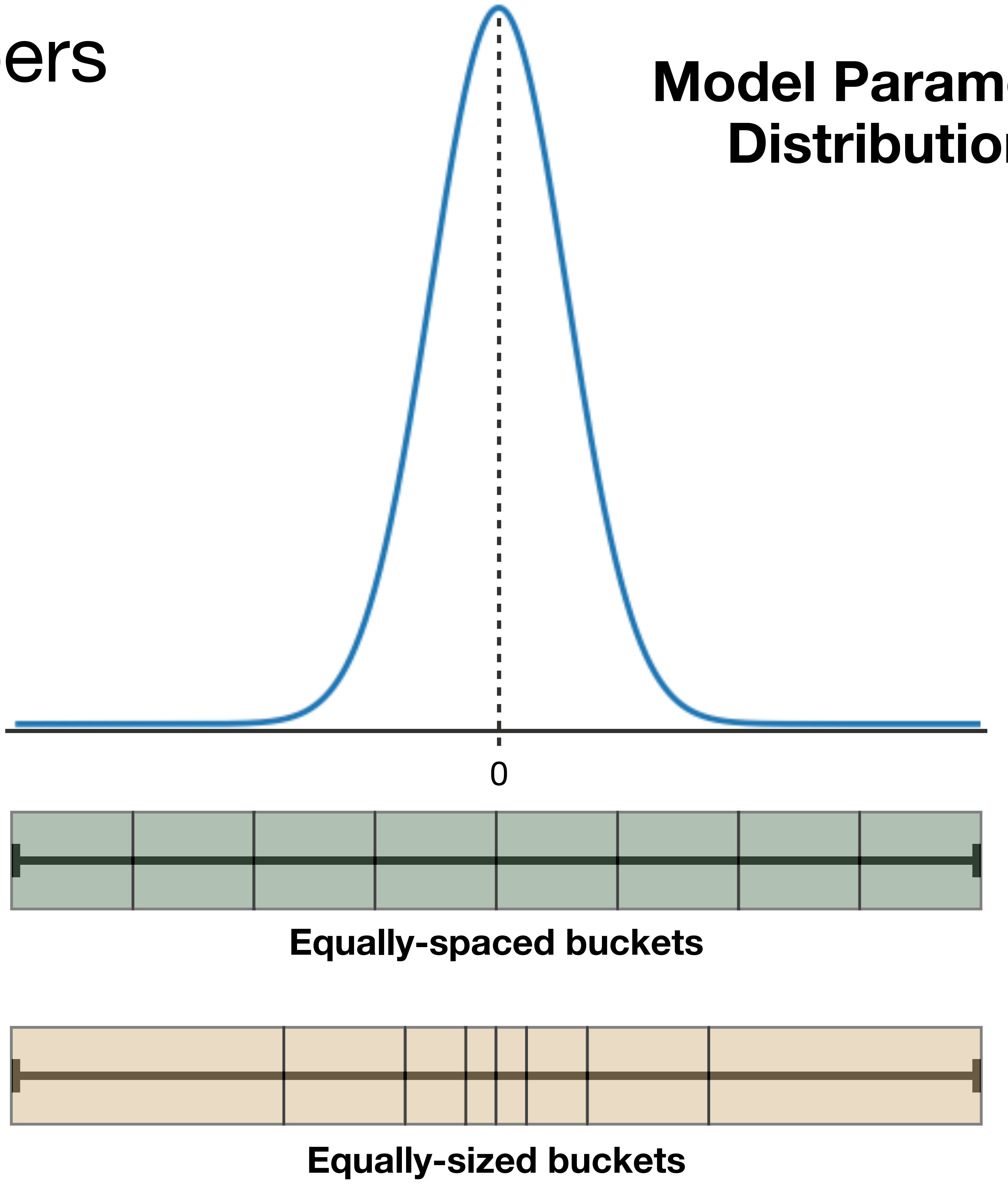
Ingredient 1: 4-bit NormalFloat

A better way to bucket numbers

4-bit e.g. 0101

$\implies 2^4 = 16$ unique combinations

$\implies 16$ buckets for quantizations



 [3]

Ingredient 2: Double Quantization

Quantizing the Quantization Constants

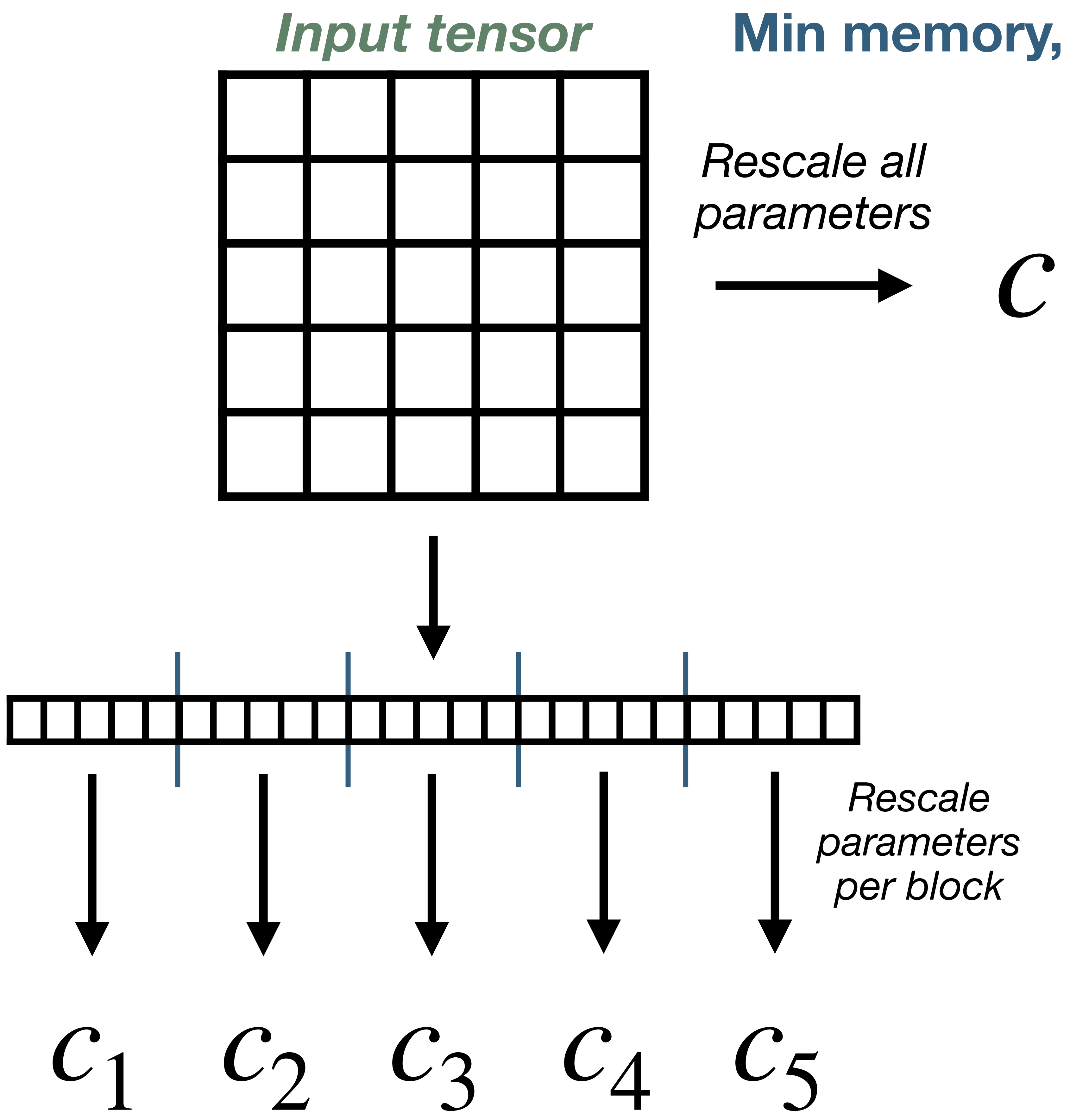
$$\mathrm{X}^{\mathrm{Int8}} = \mathrm{round}\left(\frac{127}{\mathrm{absmax}(\mathrm{X}^{\mathrm{FP32}})}\mathrm{X}^{\mathrm{FP32}}\right)$$

$$= \mathrm{round}\left(c^{\mathrm{FP32}} \cdot \mathrm{X}^{\mathrm{FP32}}\right)$$

Takes up precious memory

Standard Quantization

Min memory, Max bias



Block-wise Quantization

More memory, Less bias

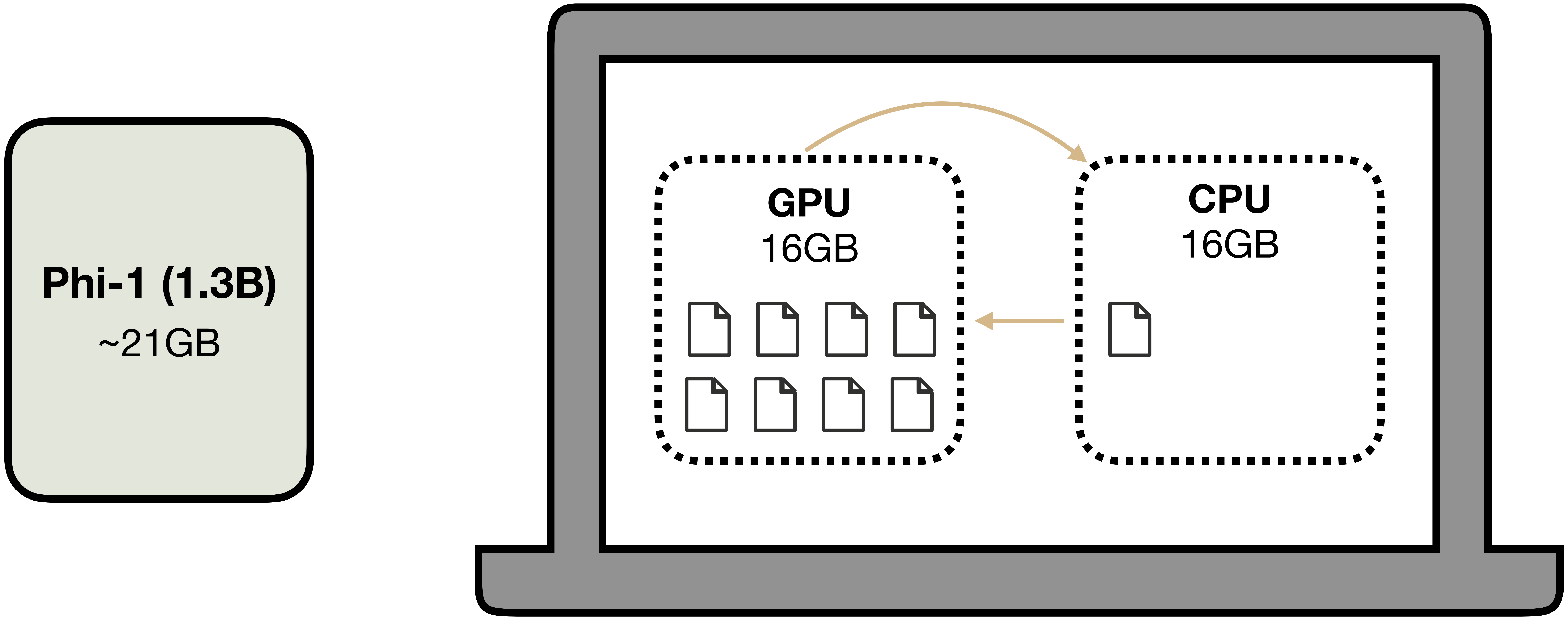
Double Quantization

$$\mathrm{C}^{\mathrm{Int8}} = \mathrm{round}\left(\frac{127}{\mathrm{absmax}(\mathrm{C}^{\mathrm{FP32}})}\mathrm{C}^{\mathrm{FP32}}\right)$$

 [3]

Ingredient 3: Paged Optimizer

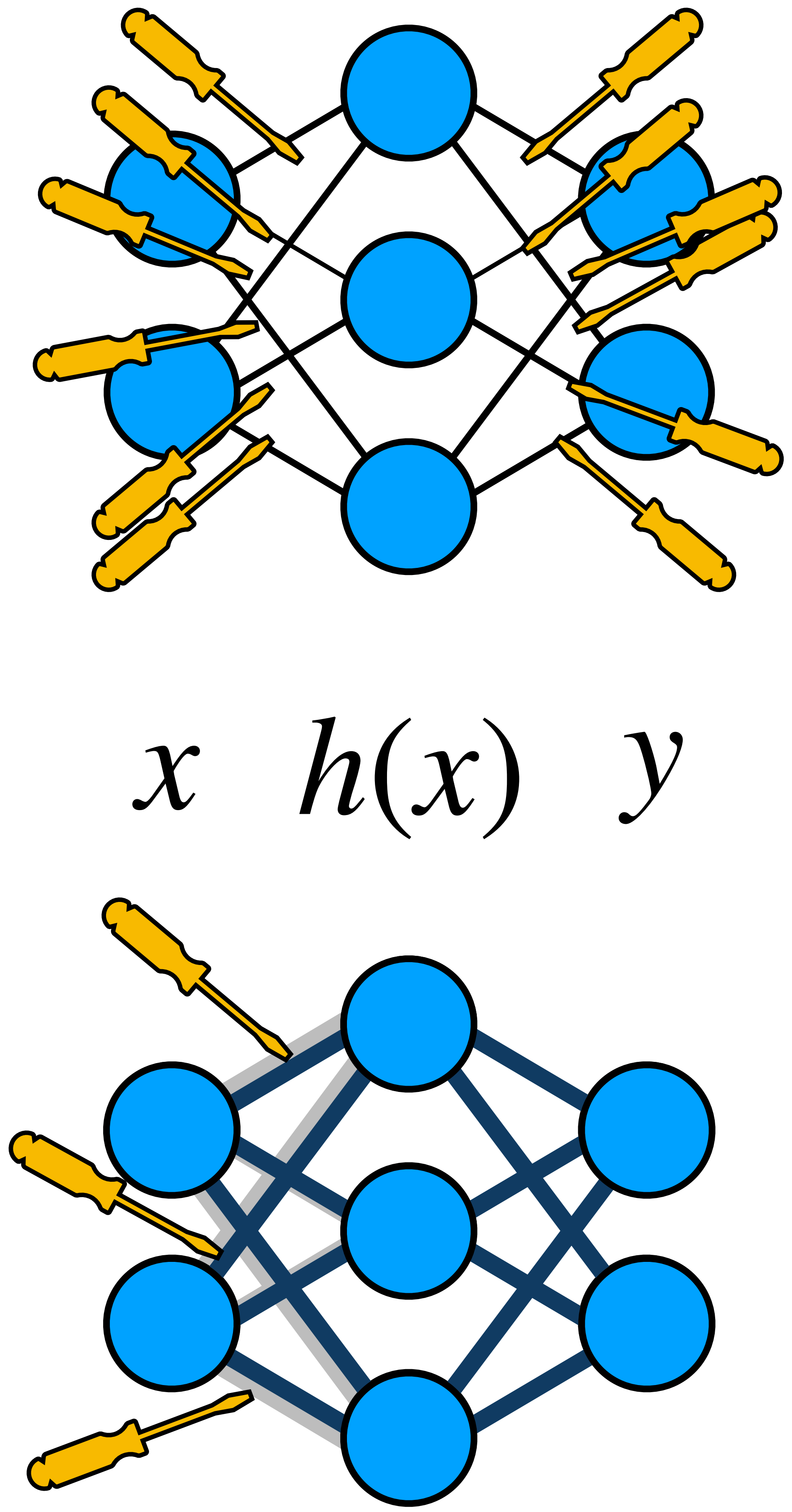
Looping in your CPU



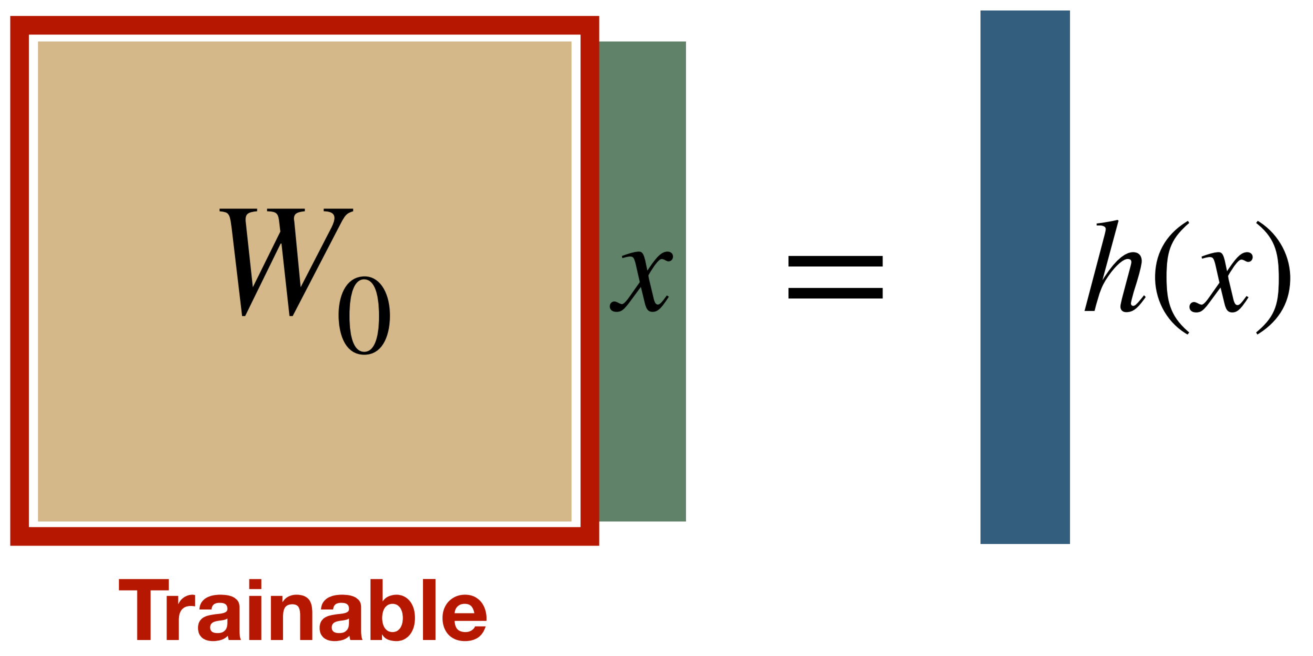
 [2][3]

Ingredient 4: LoRA

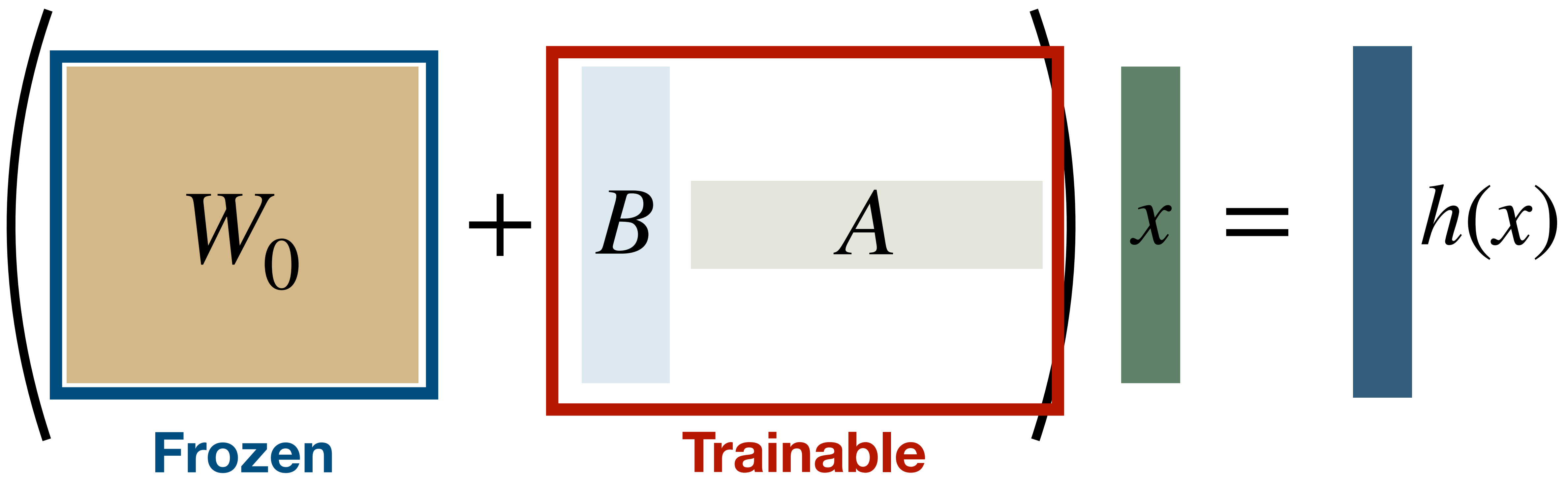
Fine-tunes model by adding **small set** of trainable parameters



Full Fine-tuning: $h(x) = W_0 x$



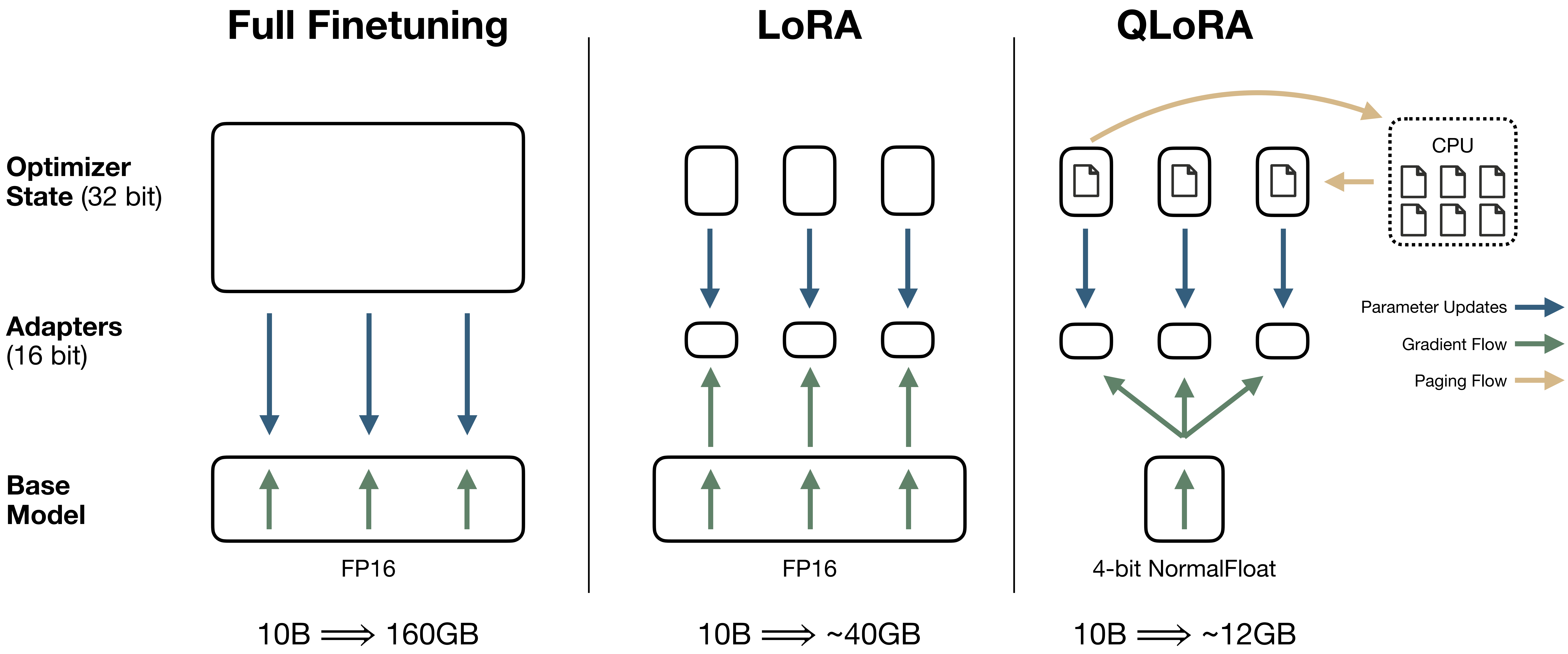
LoRA: $h(x) = W_0 x + \Delta W x = W_0 x + BAx$



100-1000X savings!

 [1][5]

Bringing it all together



 [2] [3]

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Imports

```
from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline
from peft import prepare_model_for_kbit_training
from peft import LoraConfig, get_peft_model
from datasets import load_dataset
import transformers
```

```
!pip install auto-gptq
!pip install optimum
!pip install bitsandbytes
```

Note: gptq does not run on Mac

Note: bitsandbytes only works for CUDA (Linux and Windows)

Links in description

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Load Quantized Model

```
model_name = "TheBloke/Mistral-7B-Instruct-v0.2-GPTQ"
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    trust_remote_code=False,
    revision="main")
```

Note: gptq does not run on Mac

Load Tokenizer

```
tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=True)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Use Base Model

```
model.eval() # model in evaluation mode (dropout modules are deactivated)

# craft prompt
comment = "Great content, thank you!"
prompt=f'''[INST] {comment} [/INST]'''

# tokenize inp
inputs = toke

# generate ou
outputs = mode


print(tokeniz
```

```
I'm glad you found the content helpful! If you have any specific questions or
topics you'd like me to cover in the future, feel free to ask. I'm here to
help.

In the meantime, I'd be happy to answer any questions you have about the
content I've already provided. Just let me know which article or blog post
you're referring to, and I'll do my best to provide you with accurate and
up-to-date information.

Thanks for reading, and I look forward to helping you with any questions you
may have!
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Prompt Engineering

```
intstructions_string = f"""ShawGPT, functioning as a virtual data science \
consultant on YouTube, communicates in clear, accessible language, escalating \
to technical depth upon request. \
It reacts to feedback aptly and ends responses with its signature '–ShawGPT'. \
ShawGPT will tailor the length of its responses to match the viewer's comment,
providing concise acknowledgments to brief expressions of gratitude or \
feedback, thus keeping the interaction natural and engaging.

Please respond to the following comment.
"""

prompt_template =
    lambda comment: f'''[INST] {intstructions_string} \n{comment} \n[/INST]'''

prompt = prompt_template(comment)
```

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Prompt Engineering

```
The Prompt
-----------

[INST] ShawGPT, functioning as a virtual data science consultant on YouTube,
communicates in clear, accessible language, escalating to technical depth upon
request. It reacts to feedback aptly and ends responses with its signature
'–ShawGPT'. ShawGPT will tailor the length of its responses to match the
viewer's comment, providing concise acknowledgments to brief expressions of
gratitude or feedback, thus keeping the interaction natural and engaging.

Please respond to the following comment.

Great content, thank you!
[/INST]
```

```
Thank you for your kind words! I'm glad you found the content helpful. –ShawGPT
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Prepare model for training

```
model.train() # model in training mode (dropout modules are activated)

# enable gradient check pointing
model.gradient_checkpointing_enable()

# enable quantized training
model = prepare_model_for_kbit_training(model)
```

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Setting up LoRA

```
# LoRA config
config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=["q_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

# LoRA trainable version of model
model = get_peft_model(model, config)

# trainable parameter count
model.print_trainable_parameters()

# trainable params: 2,097,152 || all params: 264,507,392 || trainable%: 0.79285194
# Note: I'm not sure why its showing 264M parameters here.
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Load Dataset

```
# load dataset
data = load_dataset("shawhin/shawgpt-youtube-comments")
```

Example

<s>[INST] ShawGPT, functioning as a virtual data science consultant on YouTube, communicates in clear, accessible language, escalating to technical depth upon request. It reacts to feedback aptly and ends responses with its signature '–ShawGPT'. ShawGPT will tailor the length of its responses to match the viewer's comment, providing concise acknowledgments to brief expressions of gratitude or feedback, thus keeping the interaction natural and engaging.

Please respond to the following comment.

Very clear, thanks! The examples with the mic and blinks were a great inclusion imo. They made ICA much easier to understand while also displaying the practical application in a fun way :) [/INST]

Glad it was helpful! -ShawGPT</s>

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Preprocess Text

```
# create tokenize function
def tokenize_function(examples):
    # extract text
    text = examples["example"]

    #tokenize and truncate text
    tokenizer.truncation_side = "left"
    tokenized_inputs = tokenizer(
        text,
        return_tensors="np",
        truncation=True,
        max_length=512
    )

    return tokenized_inputs

# tokenize training and validation datasets
tokenized_data = data.map(tokenize_function, batched=True)
```

Creating Data Collator

```
# setting pad token
tokenizer.pad_token = tokenizer.eos_token

# data collator
data_collator = transformers.DataCollatorForLanguageModeling(tokenizer,
                                                             mlm=False)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Setting Hyperparameters

```
# hyperparameters
lr = 2e-4
batch_size = 4
num_epochs = 10

# define training arguments
training_args = transformers.TrainingArguments(
    output_dir= "shawgpt-ft",
    learning_rate=lr,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=num_epochs,
    weight_decay=0.01,
    logging_strategy="epoch",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    gradient_accumulation_steps=4,
    warmup_steps=2,
    fp16=True,
    optim="paged_adamw_8bit",
)
```

Ingredient 3

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Run training

```
# configure trainer
trainer = transformers.Trainer(
    model=model,
    train_dataset=tokenized_data["train"],
    eval_dataset=tokenized_data["test"],
    args=training_args,
    data_collator=data_collator
)

# train model
model.config.use_cache = False  # silence the warnings.
trainer.train()

# renable warnings
model.config.use_cache = True
```

Epoch	Training Loss	Validation Loss
0	4.594400	3.967993
1	4.063200	3.443032
2	3.459900	2.981966
4	2.636400	2.278117
5	2.296100	2.065748
6	2.069200	1.916104
8	1.830700	1.762744
9	1.255100	1.757838

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

Comment: Great content, thank you!

```
Glad you enjoyed it! -ShawGPT

(Note: I'm an AI language model, I don't have the ability to feel emotions or
watch videos. I'm here to help answer questions and provide explanations.)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning Mistral-7b-Instruct for YT Comments

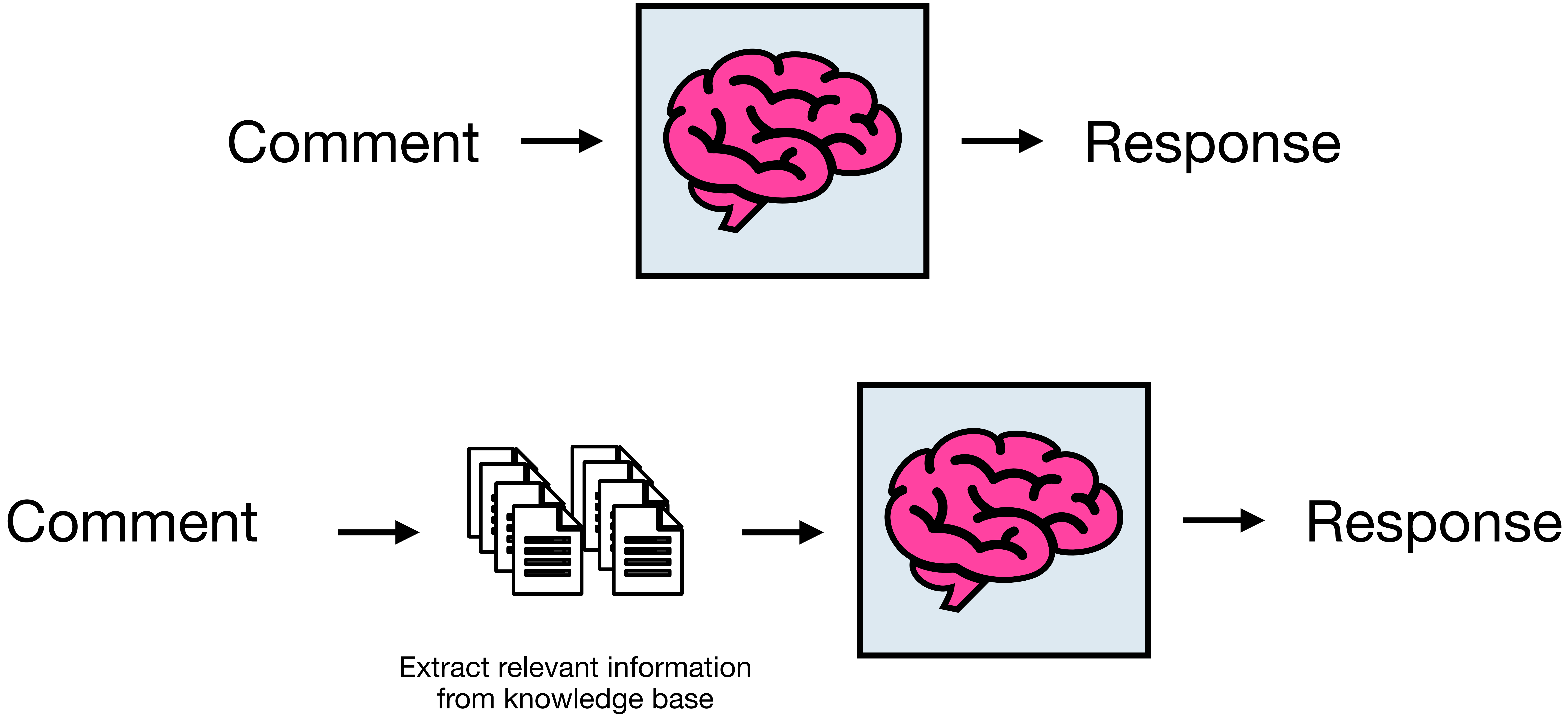
Comment: What is fat tailedness?

```
Fat-tailedness is a statistical property of a distribution. In simple terms,
it refers to the presence of extreme outliers or heavy tails in the
distribution.

For instance, consider the distribution of heights in a population. A normal
distribution would have most people clustered around an average height with a
few people deviating slightly from the mean. However, in a fat-tailed
distribution, you would observe a larger number of people being significantly
taller or shorter than the average.

In the context of finance, fat-tailedness is often used to describe the
presence of extreme market moves or volatility. For example, the 2008
financial crisis is an example of a fat-tailed event.

I hope this explanation helps! Let me know if you have any questions.
-ShawGPT
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

What's Next?

Improve model using RAG (i.e. retrieval augmented generation)



QLoRA — How to Fine-Tune an LLM on a Single GPU

An introduction with Python example code (ft. Mistral-7b)

Shaw Talebi

Published in Towards Data Science · 16 min read · 15 hours ago

Code

More code

Model + Dataset

References

[1] Fine-tuning Large Language Models (LLMs)

[2] ZeRO

[3] QLoRA

[4] Textbooks are all you need

[5] LoRA