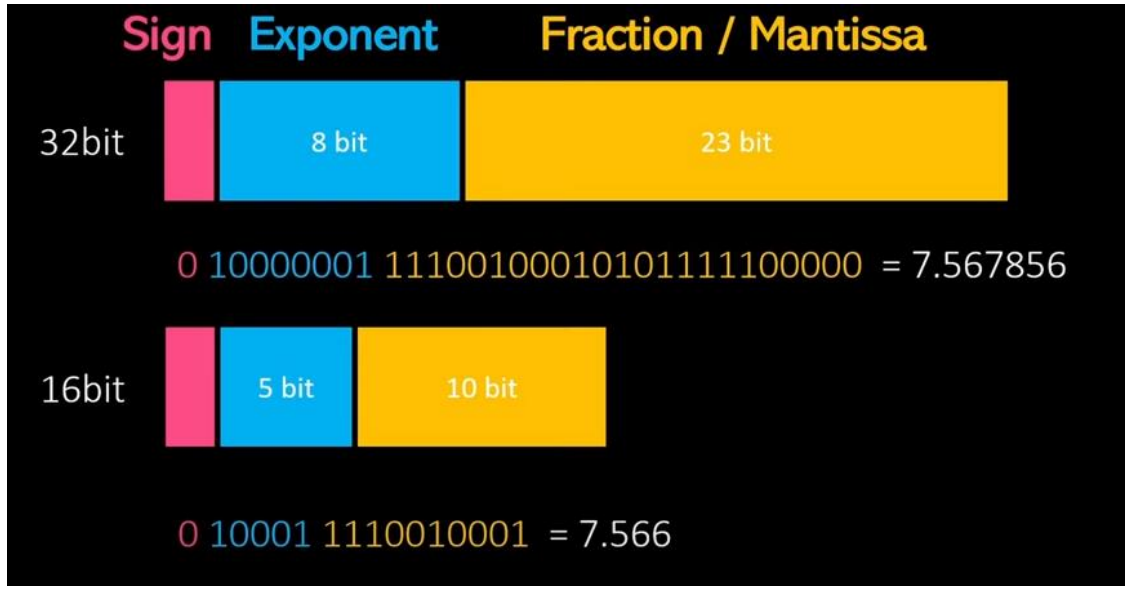
LLMs Compute Requirements & PEFT (LoRa)

Ramendra Kumar

LLM Compute Requirements

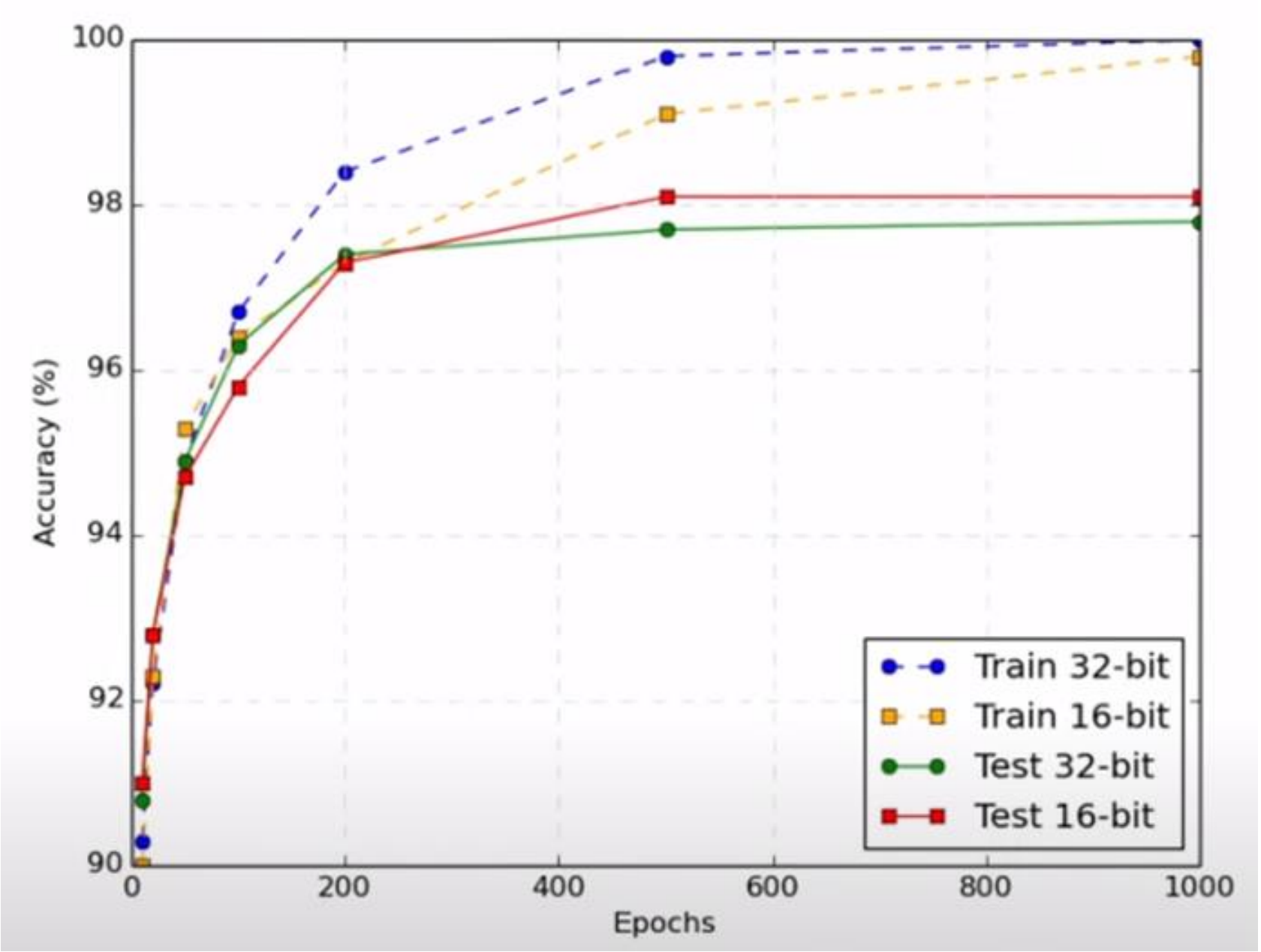
- Each parameter in an LLM takes up 4 bytes (32-bit precision) just to store the model
  - 4GB of GPU Ram per 1B parameters | $\left(Model\ Size = \text{size}_{Data\ Type} * Number\ of\ weightes\right)$
  - $\frac{1000000000 \times 4bytes}{1024 \times 1024 \times 1024}$ GB =3.74GB ~ 4GB
- GPT-3 has 175B parameters =700GB of GPU RAM, just to store the model



But Wait ! Training a model need additional storage to store –optimizer states, gradients activations & temp variables.

- Converting to 16-bit(FP16) requires 2 bytes per parameters → ½ storage requirements

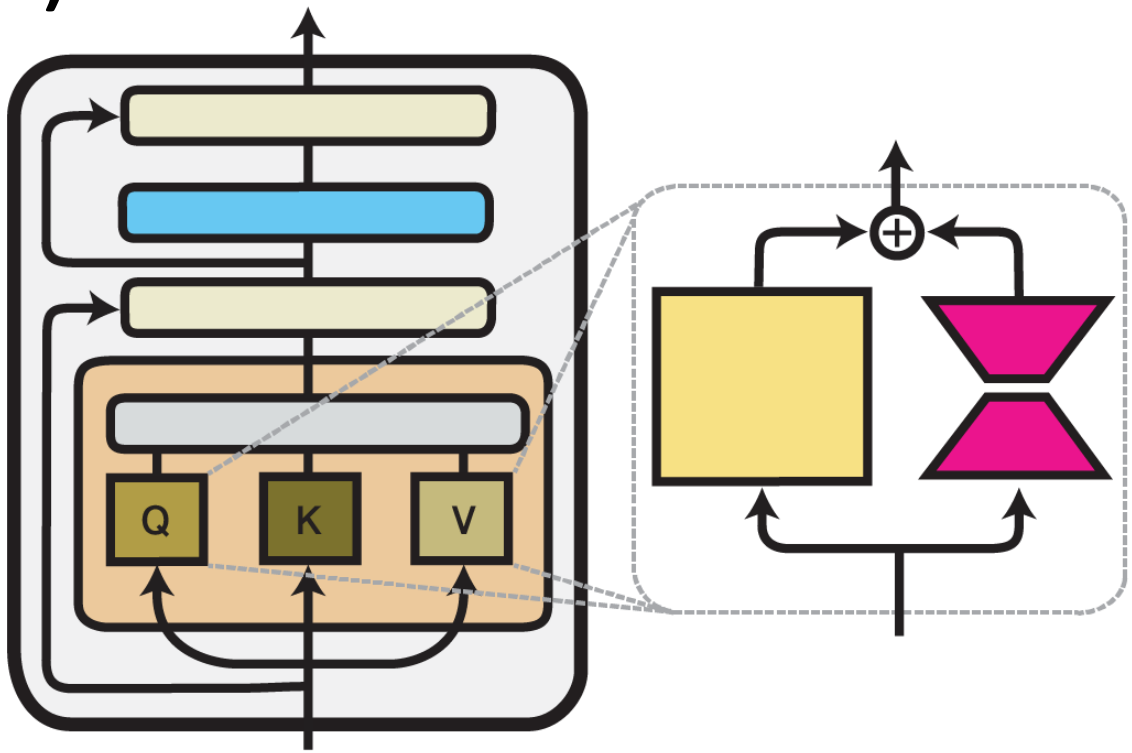
In defence of Pure 16-bit floating-point neural networks, Yun et al(2023)

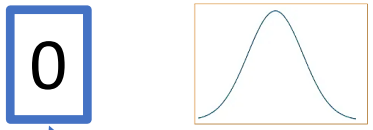


Low-rank Adaptation(LoRA)

min independent rows/columns

Fine Tuning of model





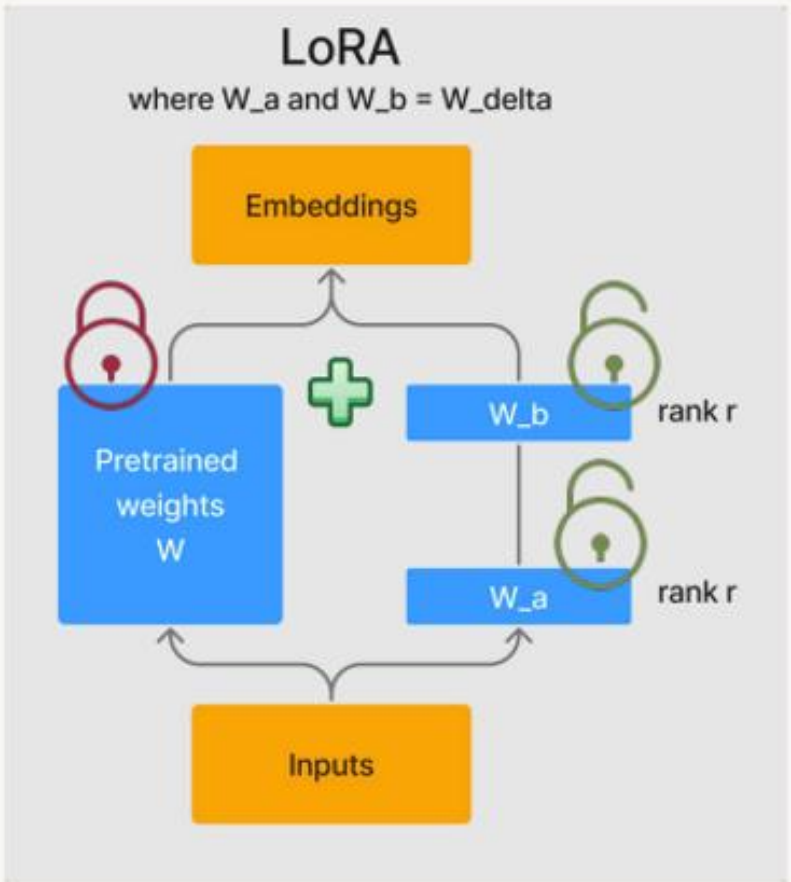
$$(W_0 + \Delta W) = (W_0 + BA)$$

- $B \in R^{d \times r}$
- $A \in R^{r \times k}$

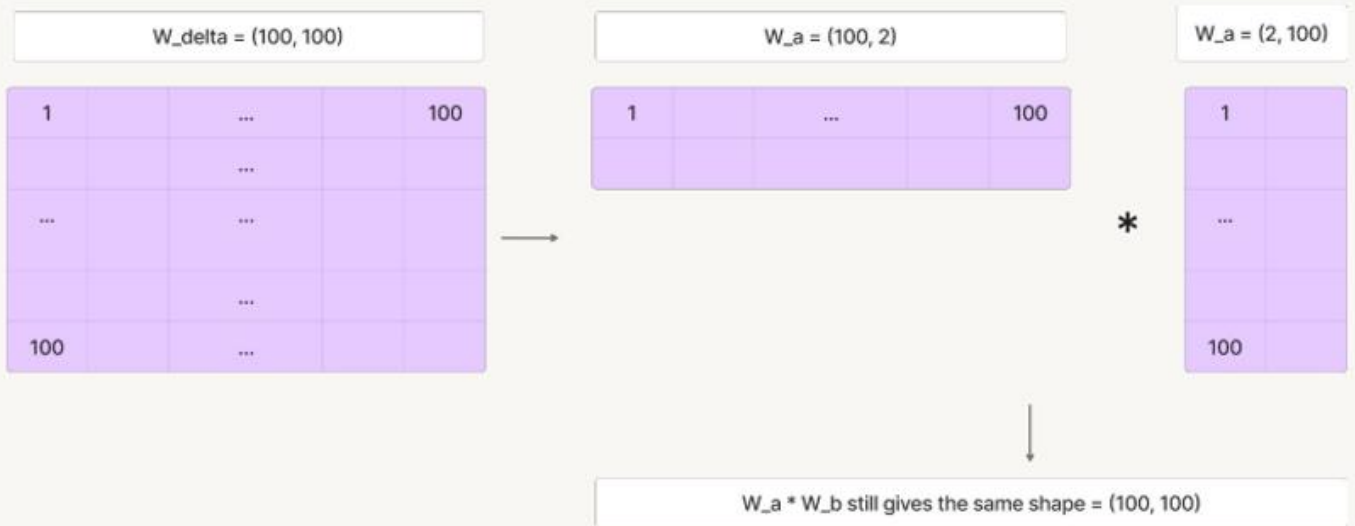
$\frac{\alpha}{r}$ ← Scaling Factor

← Rank

How does weight matrix decomposition work?
Actual rank of the attention weight matrices is low



$$W_{delta} = W_a * W_b$$



- Total parameters = (100 x 2) + (2 x 100) = 400
- Original parameters = (100 x 100) = 10,000 parameters
- Reduction = 10,000 – 400 = 96%!

PEFT

```
from transformers import AutoModelForSeq2SeqLM
from peft import get_peft_config, get_peft_model, LoraConfig, TaskType
model_name_or_path = "bigscience/mt0-large"
tokenizer_name_or_path = "bigscience/mt0-large"

peft_config = LoraConfig(
    task_type=TaskType.SEQ_2_SEQ_LM, inference_mode=False, r=8, lora_alpha=32, lora_dropout=0.1
)

model = AutoModelForSeq2SeqLM.from_pretrained(model_name_or_path)
model = get_peft_model(model, peft_config)
model.print_trainable_parameters()
# output: trainable params: 2359296 || all params: 1231940608 || trainable%: 0.19151053100118282
```

https://huggingface.co/docs/peft/en/index
https://docs.adapterhub.ml/methods.html
https://lightning.ai/pages/community/article/lora-llm/
https://www.philschmid.de/fine-tune-flan-t5-peft

PEFT Limitation

- Difficult to match the performance of full fine-tuning
- Doesn’t make inference more efficient
- Doesn’t reduce the cost of storing massive foundation models
- Requires full forward and backward passes

Decoder Task

- Predict the next token in a sequence using only previous tokens
- E.g. “the quick brown fox jumps over the lazy _____”(target :dog)
- Each document turns into multiple effective samples:

“the” → “quick”
“the quick” → “brown”
“the quick brown” → “fox”

- Language model, modelling : p(x(t) | x(t-1), x(t-1), x(t-2), …, x(1))