Quantization

CPSC 471 / 571: Trustworthy Deep Learning

Rex Ying

Readings

- Readings are updated on the website (syllabus page)

- **Quantization readings**:
  - I-BERT: Integer-only BERT Quantization (ICML'21 Oral)
  - QLoRA: Efficient Finetuning of Quantized LLMs (NeurIPS'23 Spotlight)

Outline

1. Basics of Quantization

2. Integer-only Transformers

3. Efficient fine-tuning methods

Outline

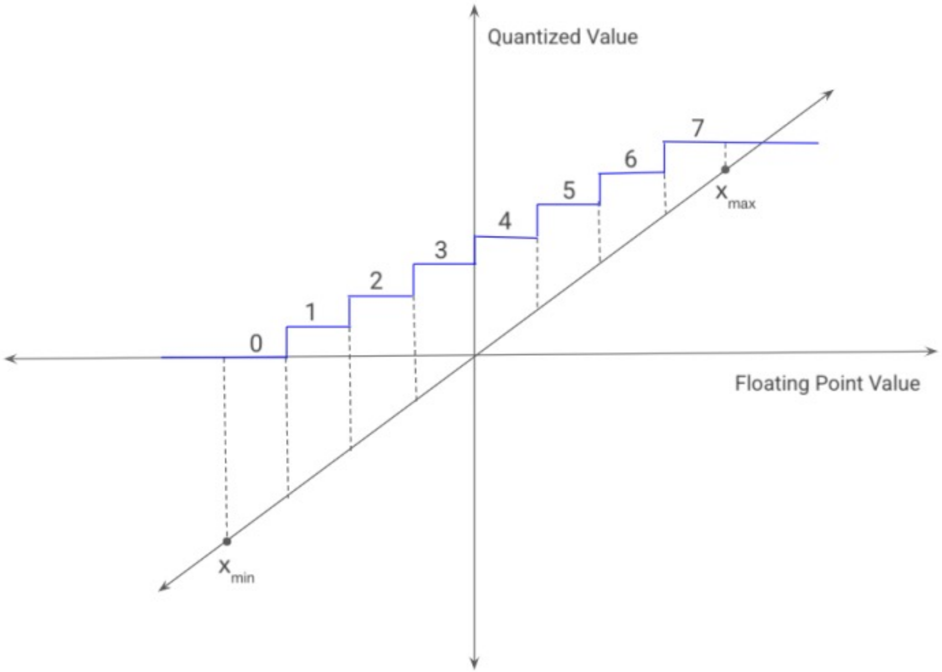
1. Basics of quantization

2. Integer-only Transformers

3. Efficient fine-tuning methods

What is quantization?

- Quantization is a well-studied technique for model optimization
- Significant reduction in model size (often 4× when using 8-bit quantization) and inference latency
  - Usual floating point is 32-bit
- **Weight** quantization
- **Activation** quantization
- **Caveat**: quality loss during inference



What are the tradeoffs?

- Quantization has two competing goals:
  - **Maximize** the precision of the target computation
  - **Minimize** the number of bits in the discrete representation

How do we choose the discrete set of values?

- The choice of the discrete set of values is dependent on the application. However, **integers** are often a convenient choice.
- In many cases, this set of values consists of low-precision integers, such as signed 8-bit integers with range [-128, 128).

0.463	0.072	9.138	5.292
4.409	5.081	7.772	5.874
6.736	4.181	6.789	0.892
2.971	3.070	4.143	0.802

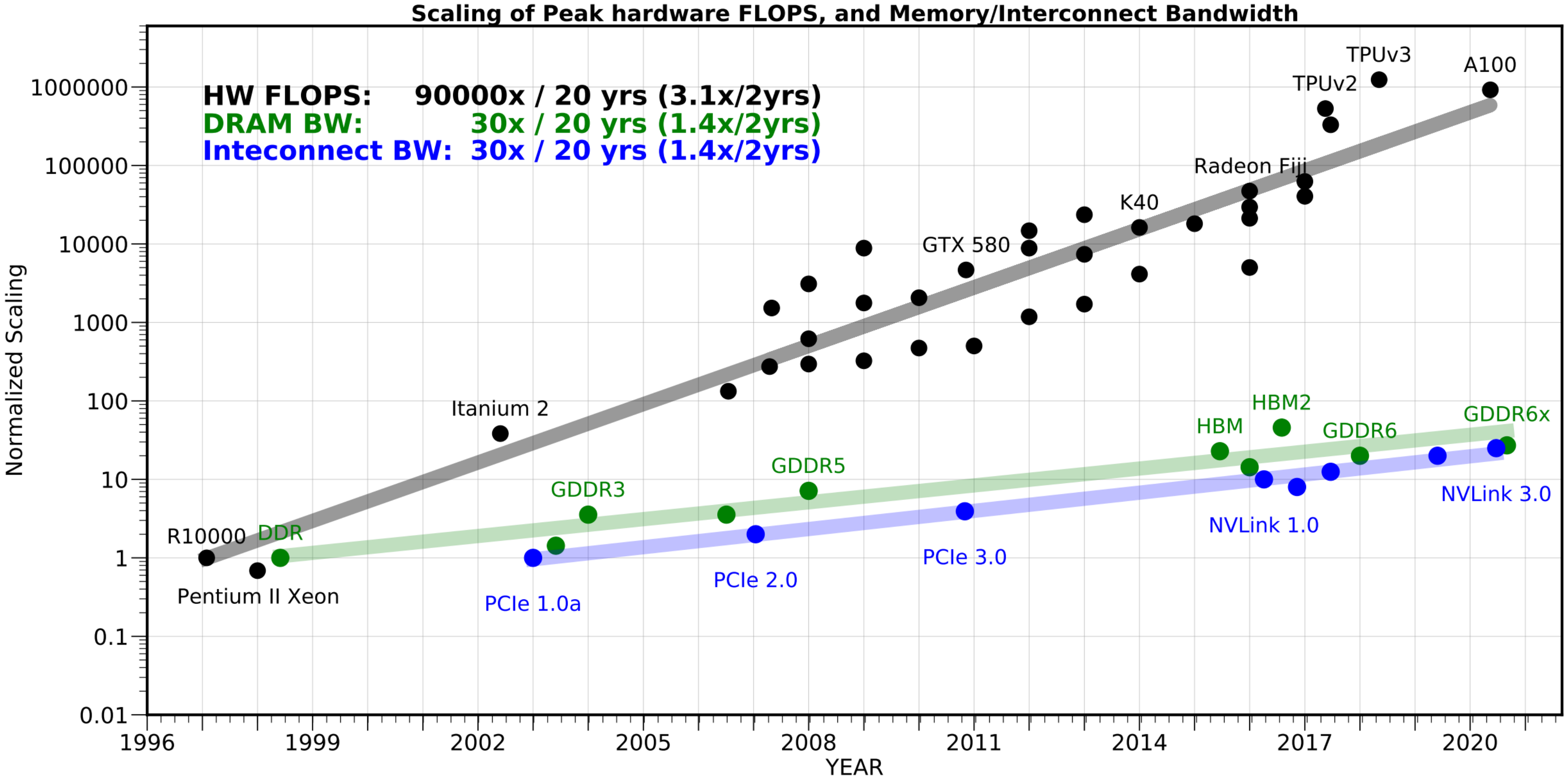
→

-116	-126	106	8
-16	2	70	22
44	-20	46	-106
-52	-50	-22	-108

Why do we care about quantization?

- **Memory wall problem**
  - Limited capacity of on-device memory / cache
  - Limited bandwidth of device communication channels
- **Environmental impact**
  - Integer ALUs are smaller and more energy-efficient relative to FPUs.
  - This is especially important for mobile devices and embedded systems.

Memory Wall



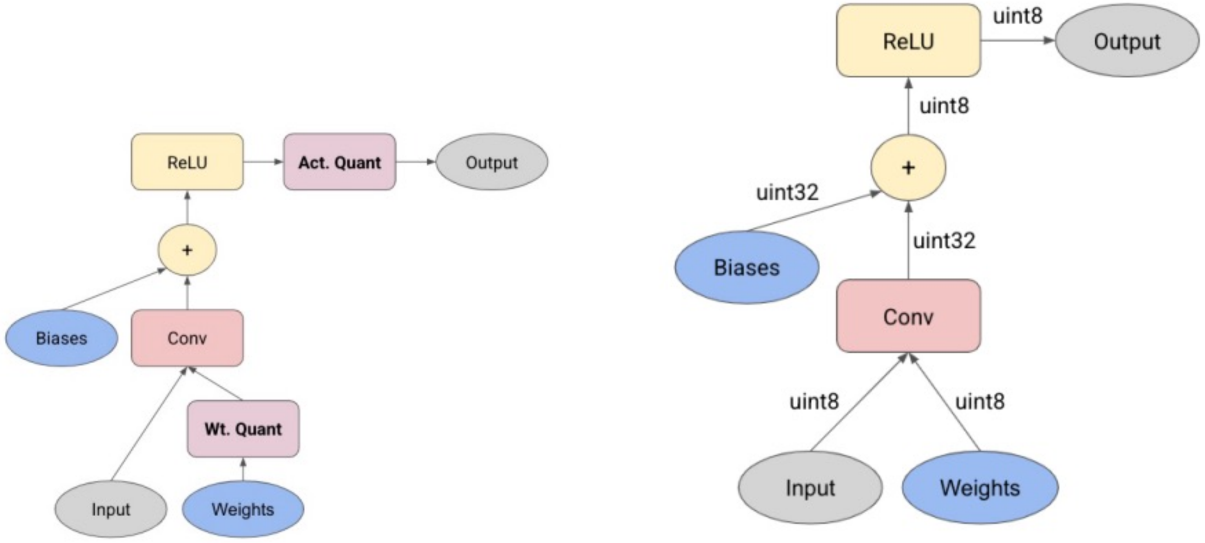
Reference: AI and Memory Wall (2021)

Quantization-Aware Training

- The training phase of the model is quantization-aware
  - Not necessarily trained with low precision
  - “**Fake quantized**” forward pass

$$
\begin{aligned}
\widehat{\mathbf{X}} &= \mathrm{FakeQuant}(\mathbf{X}) \\
&= \mathrm{Dequantize}(\mathrm{Quantize}(\mathbf{X})) \\
&= s\left(\left(\mathrm{round}\left(\frac{\mathrm{clamp}(\mathbf{X}, x_{min}, x_{max})}{s}\right) + z\right) - z\right) \\
&= s\left(\mathrm{round}\left(\frac{\mathrm{clamp}(\mathbf{X}, x_{min}, x_{max})}{s}\right)\right)
\end{aligned}
$$

Reference: [1712.05877] Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference (CVPR 2018)



(a) Quantization-Aware Training

(b) Final fixed-point inference graph

What else can we quantize?

- **Embeddings**
  - Reduces storage and serving costs (saves millions of dollars / year at scale)
  - This technique is widely used in production at Google, Meta, Pinterest, etc.
- **Optimizers**
  - Optimizer state is often quite large, impacting memory usage
  - 8-bit optimizers can be a drop-in replacement for many models
- **KV caches**
  - Useful for quantization-aware training of large language models
  - Challenge: LLM fine-tuning datasets are typically limited in size

References: 8-bit Optimizers via Block-wise Quantization (ICLR 2022 Spotlight)
LLM-QAT: Data-Free Quantization Aware Training for Large Language Models (2023)

How do we measure success?

- **Basic metrics**
  - Size of model weights
  - Runtime memory usage
  - Inference latency
  - Inference throughput
- **Utilization metrics**
  - **Hardware FLOPs utilization (HFU)**
  - **Model FLOPs utilization (MFU)**

Model FLOPs Utilization

- **Model FLOPs utilization**
  - Ratio of achieved throughput to theoretical peak throughput R
  - **Example**: $R = \frac{P}{6N+12LHQT}$ for **Transformer decoder-only model**

  - P: Theoretical peak matmul throughput (FLOPs per second)
  - N: Model parameter count
  - L, H, Q, T: layers, attention heads, head dim, sequence length
  - Non-attention operations in the model account for $6N$ FLOPs per token seen:
    - $2N$ for forward pass and $4N$ for backward pass.
  - Self-attention accounts for $6LH(2QT)$ FLOPs per token seen.

For more information on how the coefficients are estimated:
https://www.adamcasson.com/posts/transformer-flops
https://medium.com/@dzmitrybahdanau/the-flops-calculus-of-language-model-training-3b19c1f025e4

Outline

1. Basics of quantization

2. **Integer-Only Transformers**

3. Efficient fine-tuning methods

Simulated Quantization

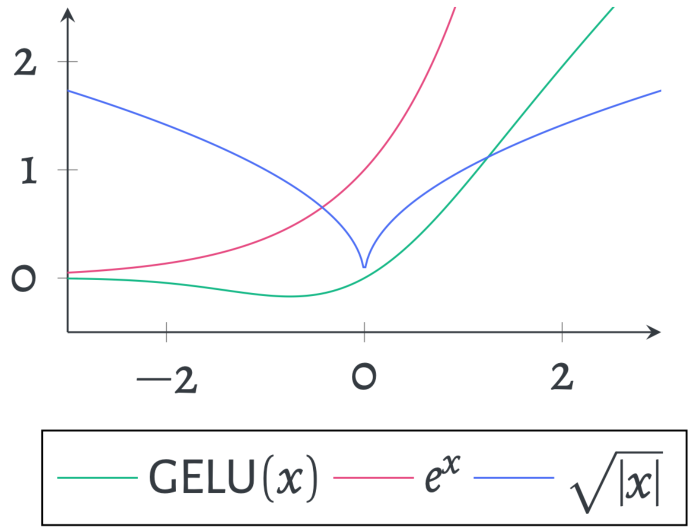
- **Inefficient baseline**: store model parameters in low-precision integer format, compute in floating-point format.
- Prior work on Transformer quantization used this approach (Bhandare et al., 2019; Shen et al., 2020; Zafrir et al., 2019).
- Reduces the cost of model transmission and loading, but does not take full advantage of fast, energy-efficient hardware.

CNN Quantization

- Many papers have focused on quantization of CNNs.
- Easy to quantize linear operations and RELU activations.
  - (1) Dequantize value, perform linear operation on real value
  - (2) Quantize value, perform linear operation, dequantize later
- Hard to quantize non-linear operations (e.g., BatchNorm).

Non-linear Operations

- The Transformer model contains three non-linear operations that are challenging to quantize with integer-only arithmetic.
- These non-linear operations are GELU, Softmax, and LayerNorm.
  - Softmax relies on the exponential function
  - LayerNorm relies on the square root function
- **I-BERT** approximates these building blocks



Polynomial Approximation

- **Goal:** Find a unique polynomial of degree at most n that passes through all $n+1$ unique points $\{(x_0, f_0), \ldots, (x_n, f_n)\}$.
- **Solution:** Lagrange interpolation formula

$$L(x) = \sum_{i=0}^{n} f_i l_i(x) \text{ where } l_i(x) = \prod_{\substack{0 \leqslant j \leqslant n \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

Integer-Only GELU

- **Activation function**

$$\mathrm{GELU}(x) = x \cdot \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

- **Error function**

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

- **Approximation strategy**
  - Approximate error function in a tight interval
  - Clip values at the boundaries of the initial approximation
  - Create a symmetric extension of the clipped approximation

Integer-Only GELU

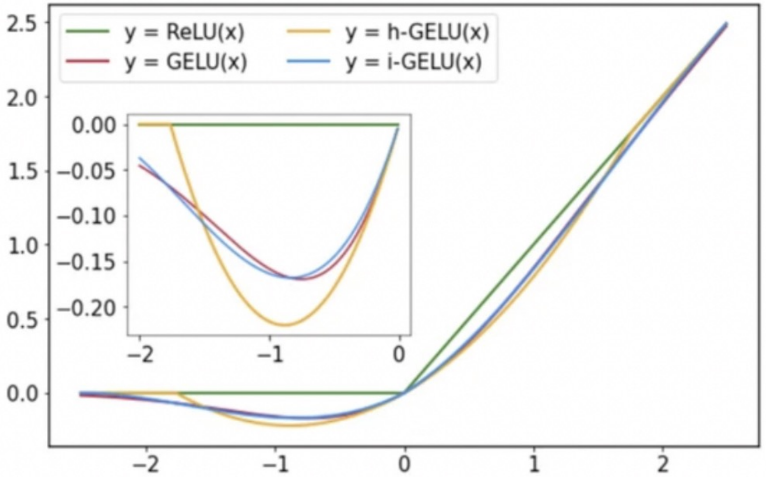
- Search for degree-2 polynomial interpolation of erf that minimizes L^2 distance of i-GELU and GELU in a small range (subject to tuning).

$$L(x) = \mathrm{sgn}(x) \cdot a(\mathrm{clip}(|x|, 0, -b) + b)^2 + 1.$$

$$a = -0.2888 \text{ and } b = -1.769$$

$$\text{i-GELU}(x) = x \cdot \frac{1}{2}\left[1 + L\left(\frac{x}{\sqrt{2}}\right)\right]$$

By assuming that $L(x) = a(x+b)^2 + c$ and use it to approximate GELU



Distance from GELU	Int-only	L^2 dist	L^∞ dist
$x\sigma(1.702x)$ [1]	✗	0.012	0.020
h-GELU [2]	✓	0.031	0.068
i-GELU (Ours)	✓	**0.0082**	**0.018**

The most accurate approximation method

Integer-Only Softmax

- **Softmax definition**

$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \text{ where } x = [x_1, \ldots, x_k]$$

- **Approximation strategy**
  - Rewrite softmax such that e^x is only evaluated in negative domain.
  - Hence only need to approximate e^x in negative domain.
  - High-degree polynomials do not work well. Instead find an approximation in a tight interval and extend to full domain.

Integer-Only Softmax

- Rewrite the softmax function:

$$\text{Softmax}(x)_i = \frac{e^{x_i - x_{max}}}{\sum_{j=0}^{k} e^{x_j - x_{max}}} \text{ where } x_{max} = \max\{x_1, \ldots, x_k\}$$

- Search for degree-2 polynomial interpolation of e^x that minimizes L^2 distance in the range $[-\ln 2, 0]$.
- Rewrite real value $x = (-\ln 2) \cdot z + p$ where the real value $p \in [-\ln 2, 0]$.
- Calculate $e^x = 2^{-z} \cdot e^p$ with bitshift and the polynomial approximation for e^p.

Integer-Only Layer Normalization

- **Layer Normalization function:**

$$\text{LayerNorm}(x; \gamma, \beta) = \gamma \frac{x - \mu_L}{\sigma_L} + \beta$$

- μ_L, σ_L are the mean and standard deviation of the input across the channel dimension. **Calculated at runtime!**
- Challenge is the square root term in the calculation of the standard deviation. Solve with fast iterative algorithm.
- Here we obtain an **exact result**: (see paper for details)

$$\text{isqrt}(n) = \lfloor \sqrt{n} \rfloor$$

I-BERT Results

- No accuracy degradation on GLUE benchmark.
- Improvements could be due to reduced ability to overfit.

(a) RoBERTa-Base

	Precision	Int-only	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
Baseline	FP32	✗	**87.8**	**87.4**	**90.4**	**92.8**	94.6	61.2	**91.1**	90.9	78.0	86.0
I-BERT	INT8	✓	87.5	**87.4**	90.2	**92.8**	**95.2**	**62.5**	90.8	**91.1**	**79.4**	**86.3**
Diff			-0.3	0.0	-0.2	0.0	+0.6	+1.3	-0.3	+0.2	+1.4	+0.3

(b) RoBERTa-Large

	Precision	Int-only	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
Baseline	FP32	✗	90.0	89.9	92.8	94.1	96.3	68.0	**92.2**	91.8	86.3	89.0
I-BERT	INT8	✓	**90.4**	**90.3**	**93.0**	**94.5**	**96.4**	**69.0**	**92.2**	**93.0**	**87.0**	**89.5**
Diff			+0.4	+0.4	+0.2	+0.4	+0.1	+1.0	0.0	+1.2	+0.7	+0.5

I-BERT Results

- Up to 4.0x speedup with NVIDIA T4 GPUs. [Kim et al. 2021]
- Up to 39.6x speedup with custom hardware. [Kim et al. 2023]

SL	128				256				Avg.
BS	1	2	4	8	1	2	4	8	
Base	2.42	3.36	3.39	3.31	3.11	2.96	2.94	3.15	3.08
Large	3.20	4.00	3.98	3.81	3.19	3.51	3.37	3.40	3.56

Outline

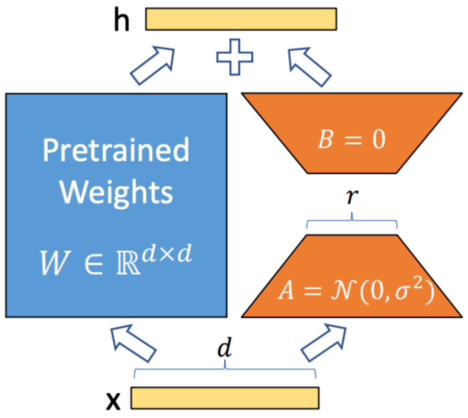
1. Basics of quantization

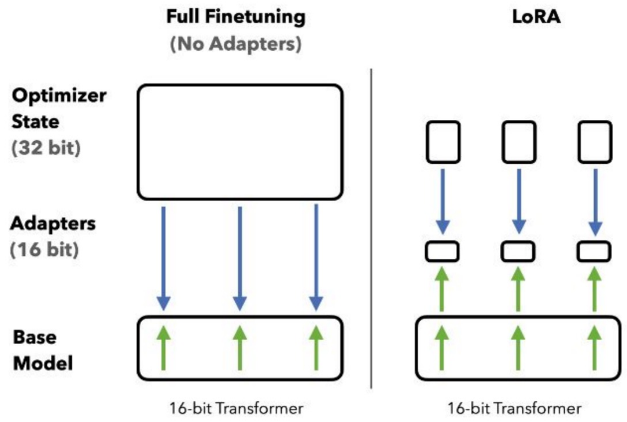
2. Integer-only Transformers

3. Efficient fine-tuning methods

Low-rank adapters

- **Low-rank adapters**
  - Low-rank adapters are small, additional neural network modules inserted between the layers of a pre-trained model. They are designed to adapt the model to new tasks with minimal additional parameters.
- **Key benefits**
  - They introduce a minimal number of parameters, making the fine-tuning process more efficient.
  - Adapters allow for task-specific tuning while keeping the original model weights frozen.





Reference: [LoRA: Low-Rank Adaptation of Large Language Models (2021)]

QLoRA

- **Motivating question**
  - How to make fine-tuning LLMs more accessible?
- **Key techniques**
  - Low rank adapters
    - Same as LoRA
  - 4-bit quantized base model
    - Uses new data type (NF4)
  - Paged optimizers
  - Double quantization
    - See paper for details

Model	Fine-tuning memory
T5-11B	132 GB
Mistral-7B	84 GB
LLaMA2-70B	840 GB

QLoRA

Model	Fine-tuning memory
T5-11B	6 GB
Mistral-7B	5 GB
LLaMA2-70B	46 GB

Reference: [QLoRA: Efficient Finetuning of Quantized LLMs (NeurIPS 2023)]

Paged Optimizers

- **Problem**
  - Varying sequence lengths can cause GPU memory spikes in fine-tuning
- **Solution**
  - NVIDIA GPUs support the *unified memory* feature, which does automatic page-to-page transfers between the CPU and GPU for error-free GPU processing in the scenario where the GPU occasionally runs out-of-memory.
  - This feature is used to allocate paged memory for the optimizer states which are then automatically evicted to CPU RAM when the GPU runs out-of-memory and paged back into GPU memory when the memory is needed in the update step. This solves the key problem for single GPU fine-tuning.

4-bit NormalFloat (NF4)

- Information-theoretically optimal data type for normal distributions