



**CS 598**

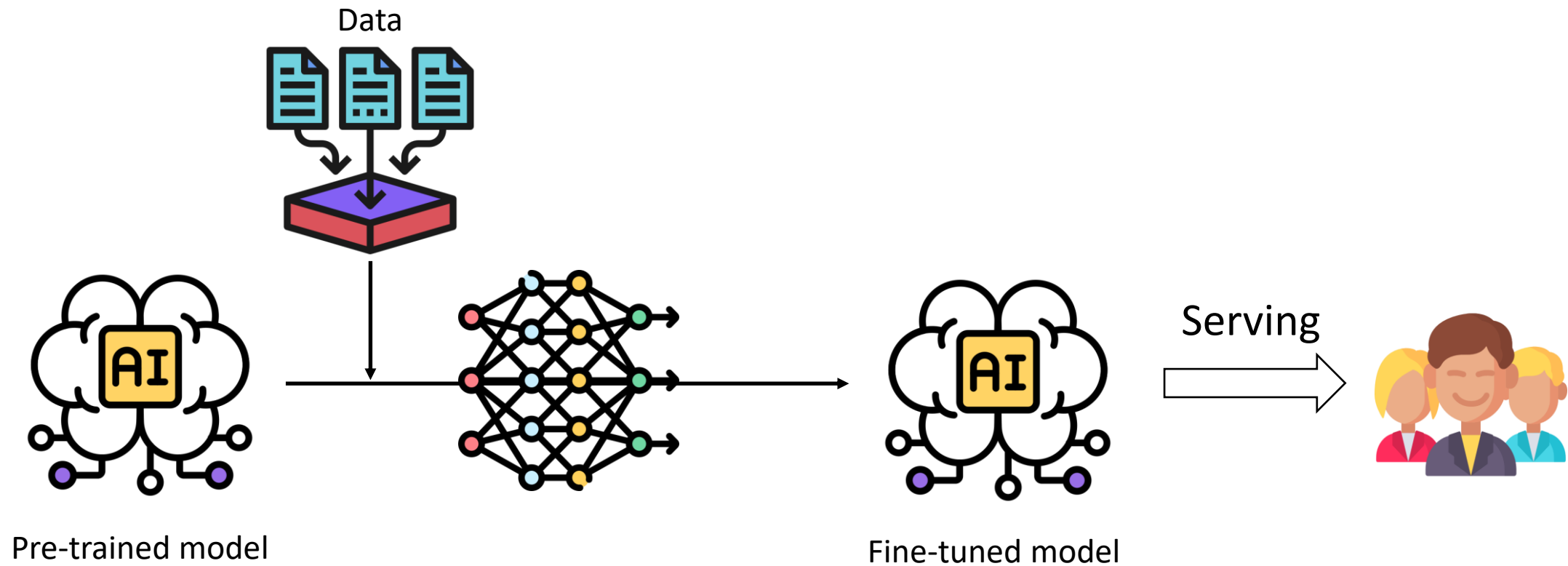
# **AI Efficiency: Systems and Algorithms**

## **Overview & Key Challenges in AI Model Serving**

Minjia Zhang

Computer Science Department

# What is Model Serving?



# Serving Scenario 1: Online ChatBot

What are some fun places to visit at Urbana Champaign?

✓ Searching for: **fun places to visit at Urbana Champaign**

✓ Generating answers for you...

There are many fun places to visit at Urbana Champaign, depending on your interests and preferences. Here are some of the most popular ones:

- If you love nature and animals, you might enjoy visiting the **Anita Purves Nature Center**<sup>1</sup>, where you can explore trails, feed goats, and learn about local wildlife.
- If you are into art and culture, you might want to check out the **Krannert Art Museum**<sup>2</sup>, which has a diverse collection of artworks from various regions and periods, as well as exhibitions and events.
- If you are looking for some entertainment and history, you might like the **Virginia Theatre**<sup>3</sup>, which is a restored historic venue that hosts concerts, movies, and shows.
- If you are feeling adventurous and sporty, you might have fun at the **University of Illinois Ice Arena**, where you can skate, play hockey, or watch games.


These are just some of the fun places to visit at Urbana Champaign. You can find more information and reviews on [Tripadvisor](#) or [Bing](#). I hope you have a great time exploring the city! 😊

Learn more ▾ 1 [experiencecu.org](#) 2 [kam.illinois.edu](#) 3 [bing.com](#)


👍 🗨 📄 ⬇ 🔄 2 of 30 ●

A map of the Urbana-Champaign area in Illinois. The map shows major roads like I-57, I-74, and I-150. Several locations are marked with red pins: Anita Purves Nature Center (top right), Krannert Art Museum (center), Virginia Theatre (center), and University of Illinois Ice Arena (bottom center). Other labels on the map include Sawgrass, Staley, Champaign, Urbana, and Spurlock Museum.






# Serving Scenario 2: Online Image Generation

 **You**

Digital illustration of a beach scene crafted from yarn. The sandy beach is depicted with beige yarn, waves are made of blue and white yarn crashing onto the shore. A yarn sun sets on the horizon, casting a warm glow. Yarn palm trees sway gently, and little yarn seashells dot the shoreline.


 **Copilot**

I'll try to create that.




1 of 30 responses


AI-generated content may be incorrect



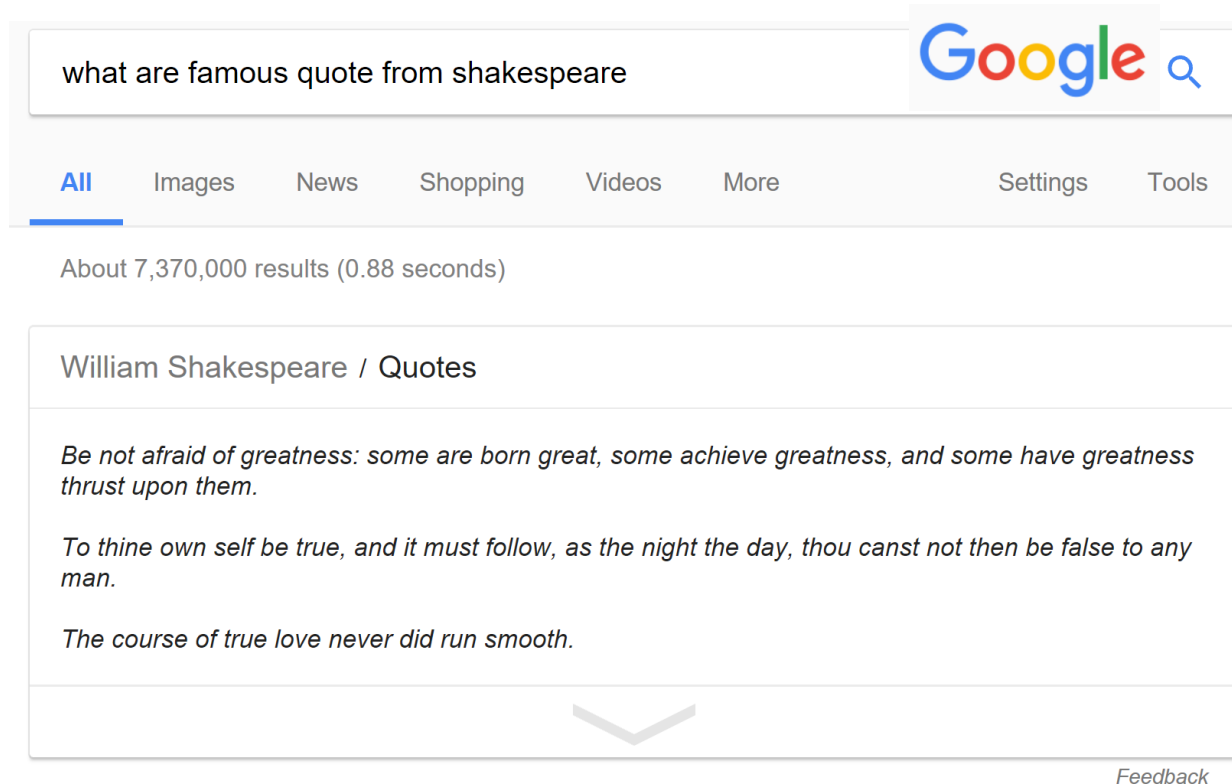
Digital illustration of a beach scene crafted from yarn. The sandy ...

 Designer

Powered by DALL·E 3

 14

# Serving Scenario 3: Online Q&A

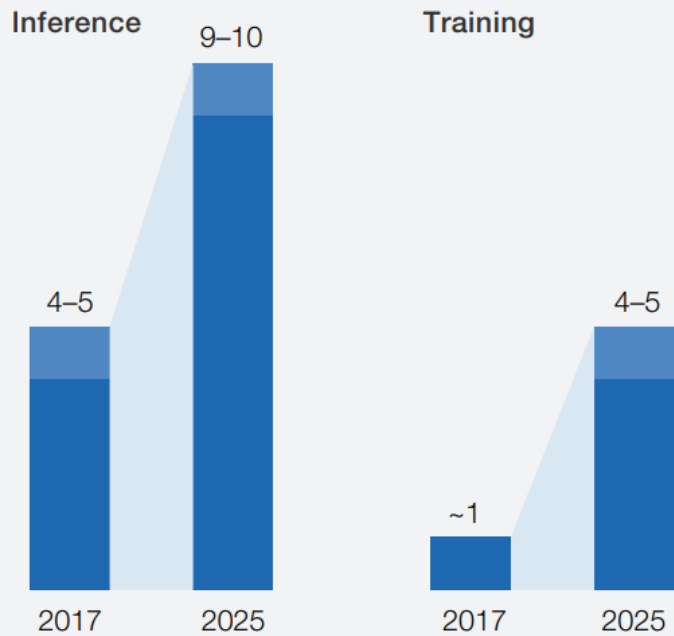


- Question and Answer Scenario
  - Direct answer not a list of webpages
  - Good quality of answer

# Training -> Inference

**Exhibit 5** At both data centers and the edge, demand for training and inference hardware is growing.

Data center, total market, \$ billion



Edge, total market, \$ billion



Source: Expert interviews; McKinsey analysis

# Inference Challenges

**Training**

vs

**Inference**

Runtime

Weeks or months

Milliseconds or seconds

Challenges

TCO (Cost, Energy)

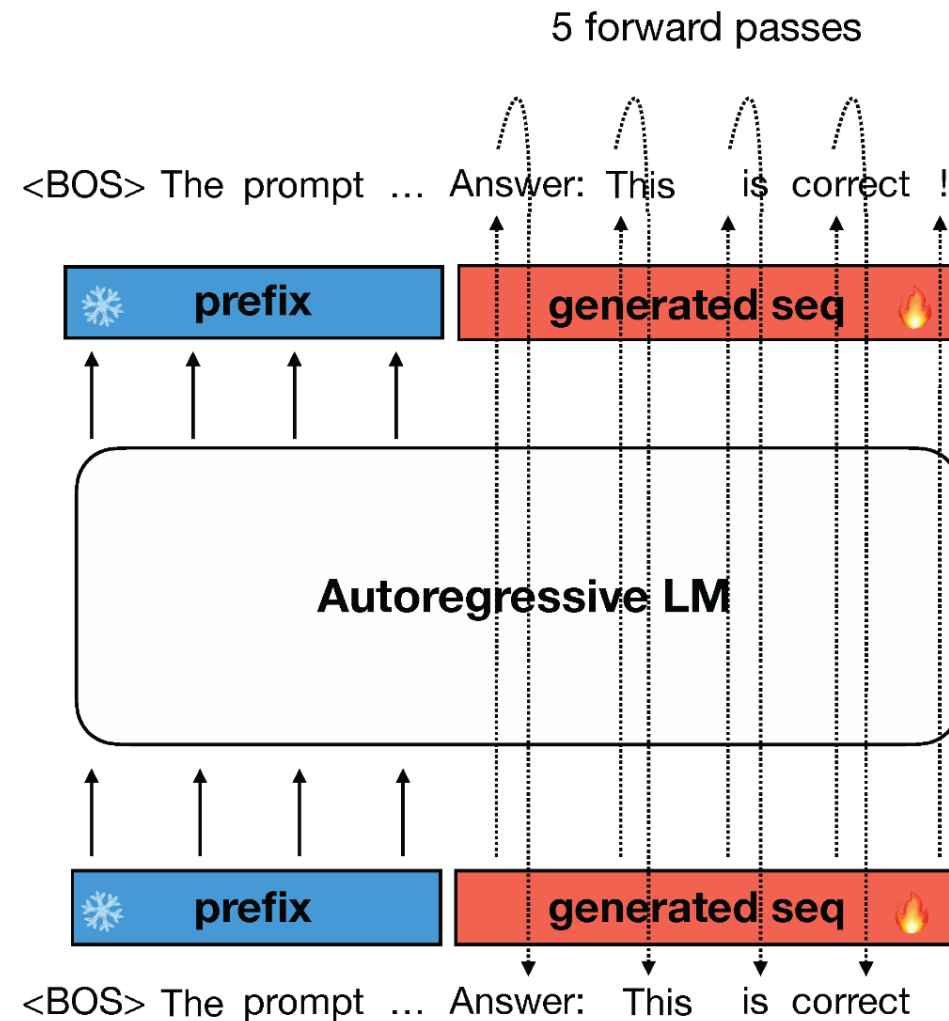
TCO (Cost, Energy)

**Speed** (LLM: token rates)

**Model size**

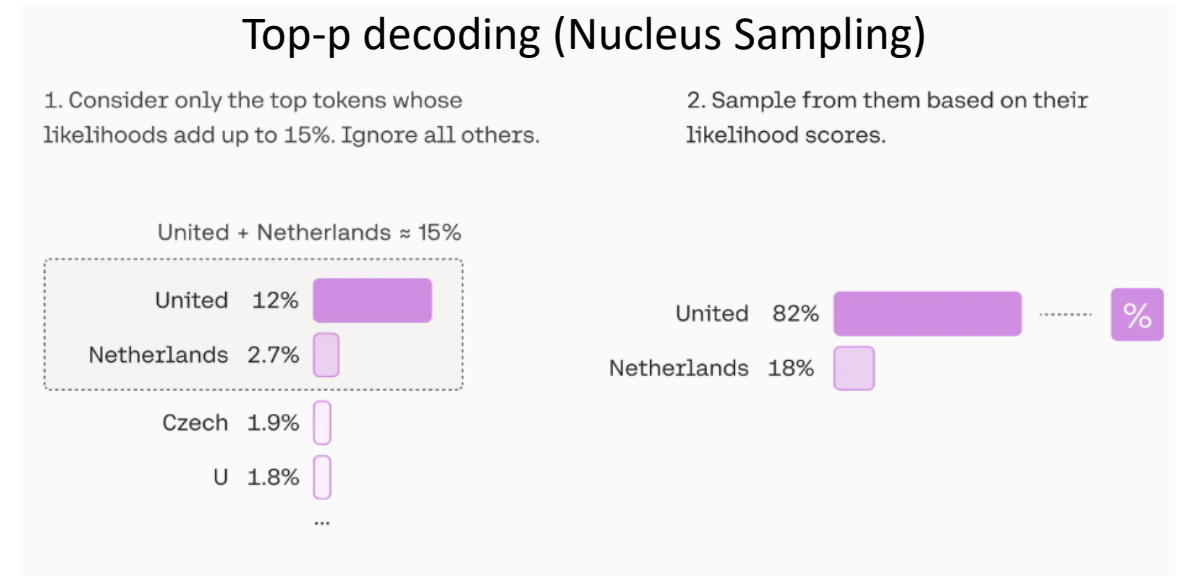
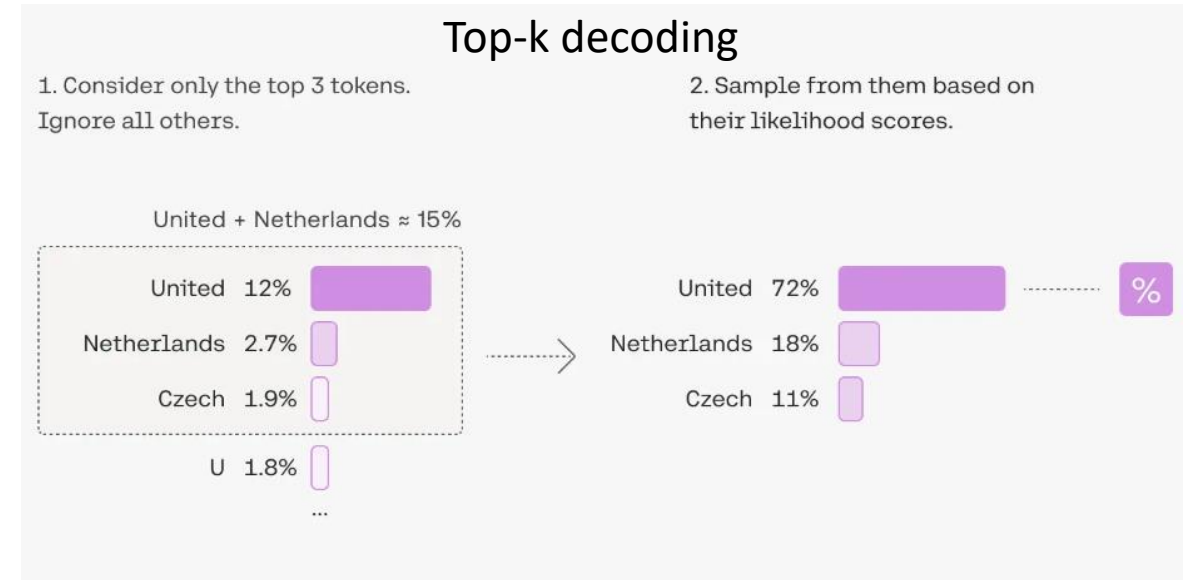
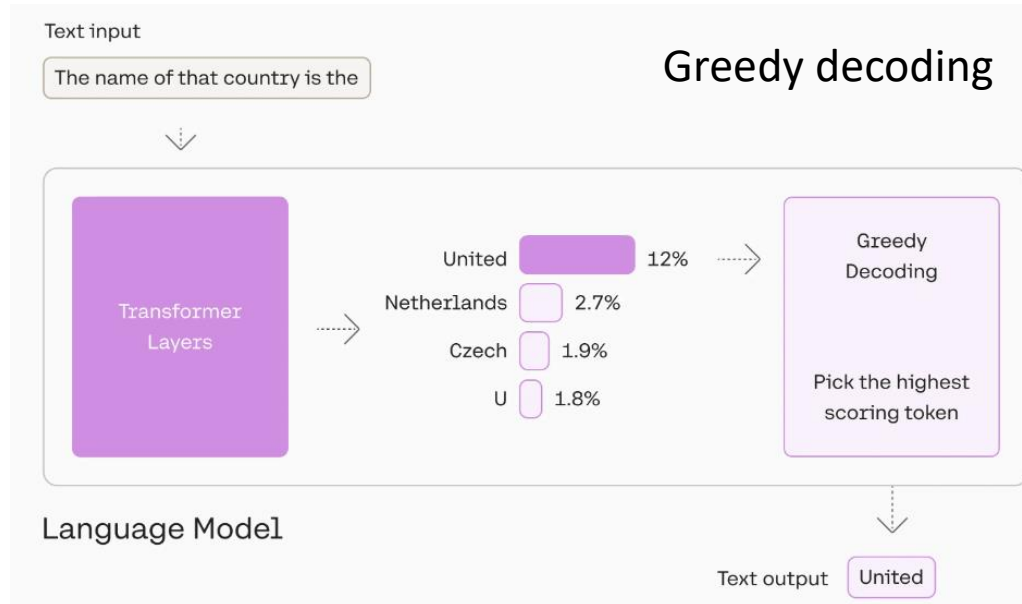
- Parameter volume
- LLM: Context length

# LLM Autoregressive Generation





# Decoding Strategies

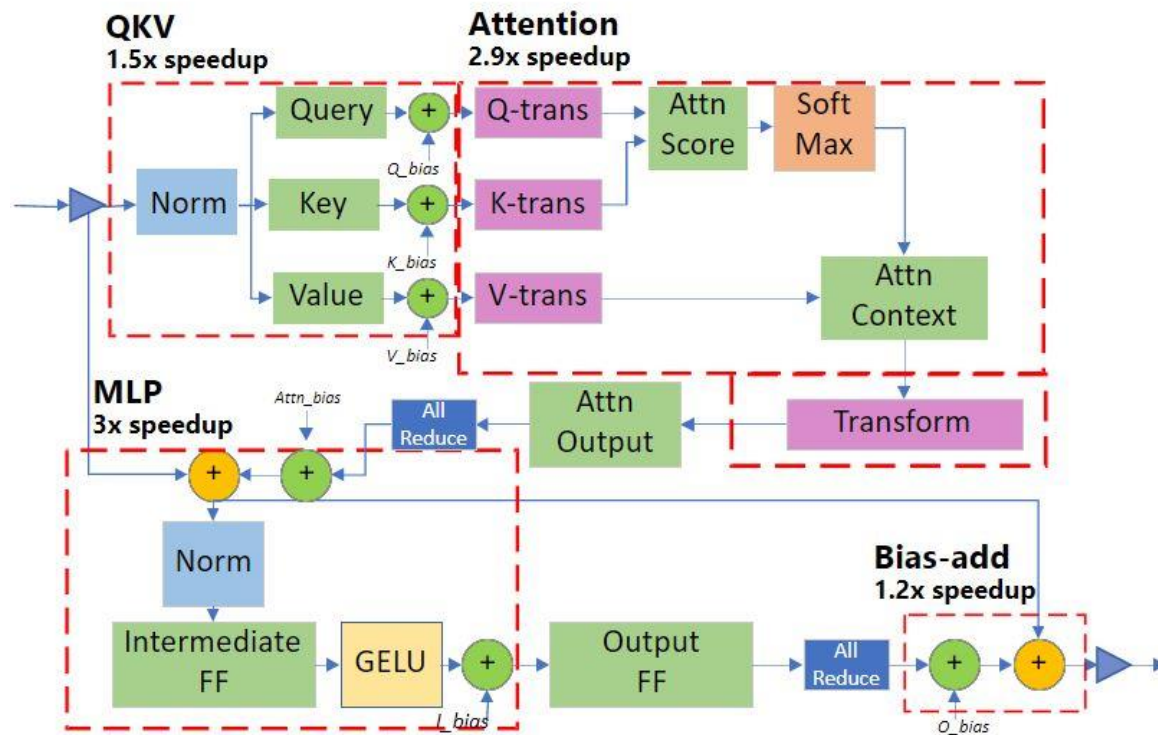


# Serving Challenge: Long Latency

- Long serving latency blocks deployment
- Support advance models while meeting latency SLA and saving cost

DL Scenarios	Original Latency	Latency Target
Turing Prototype 2	~100ms	< 10ms
Turing Prototype 3	~107ms	< 10ms
Deep Query Document Similarity	10~12ms for [query, 1 doc] x 33 docs	< 6ms
Malta Click Features	10ms for [query, 1 passage] x 150 passages	< 5ms
Ads seq2seq model for query rewriting	~51ms	< 5ms

# Customized Kernels



DeepSpeed-Inference: enabling efficient inference of transformer models at unprecedented scale, SC 2022

## ByteTransformer: A High-Performance Transformer Boosted for Variable-Length Inputs, 2023

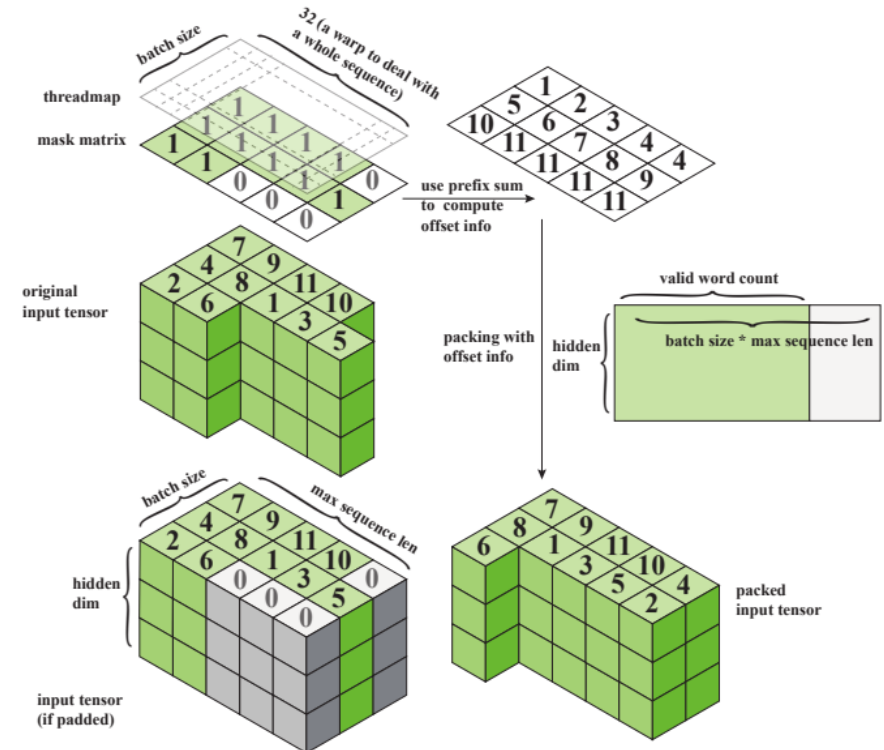
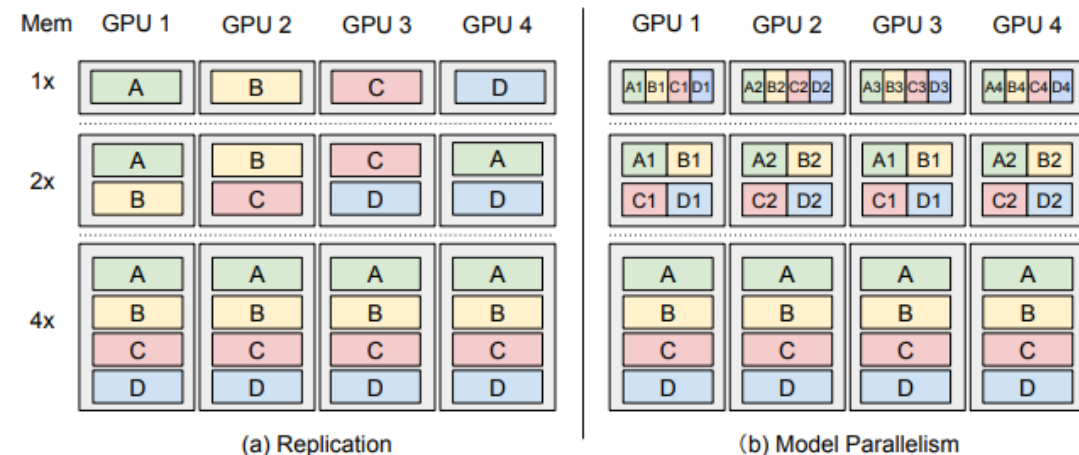
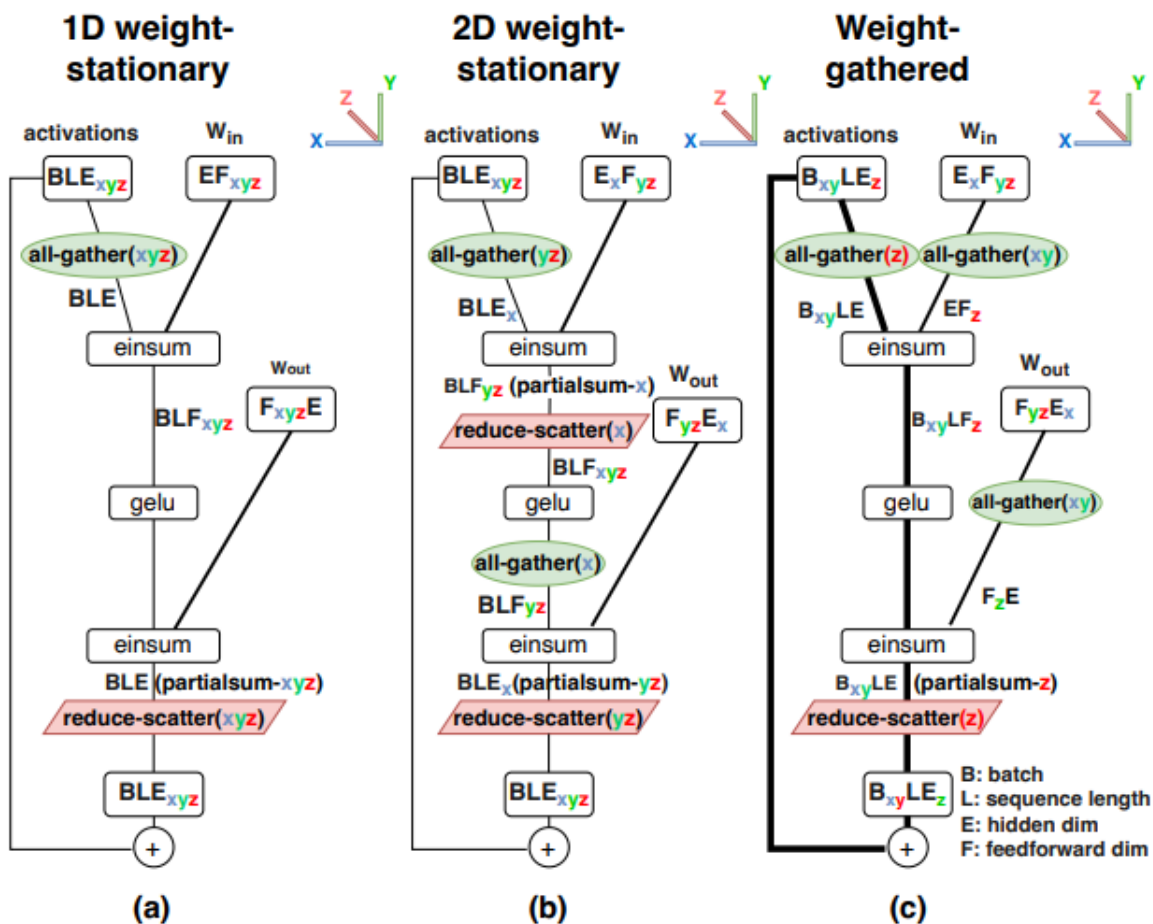


Fig. 4: The zero padding algorithm.

# Multi-GPU Inference via Partitioned Layouts

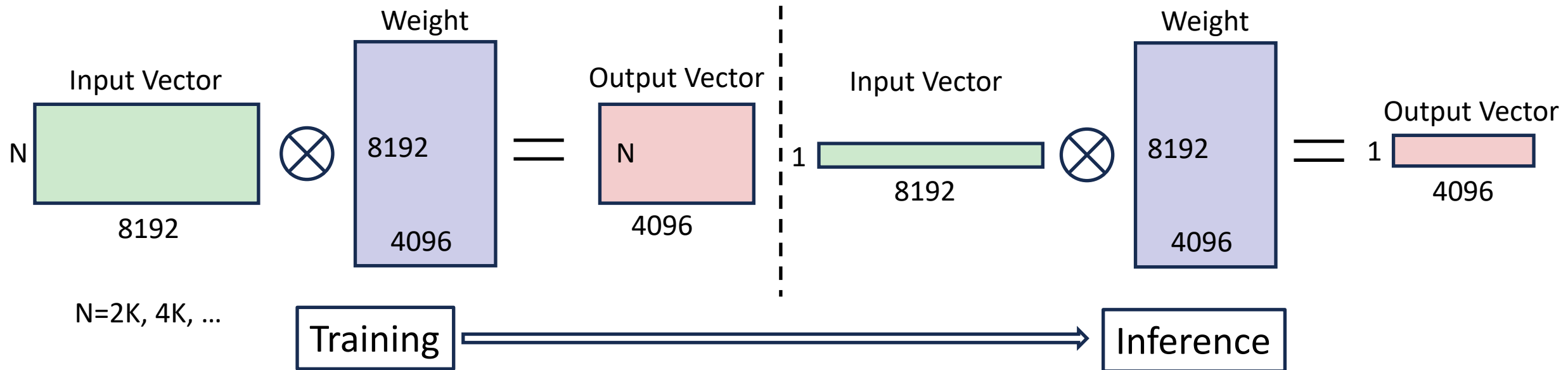
AlpaServe: Statistical Multiplexing with Model Parallelism for Deep Learning Serving, OSDI 2023



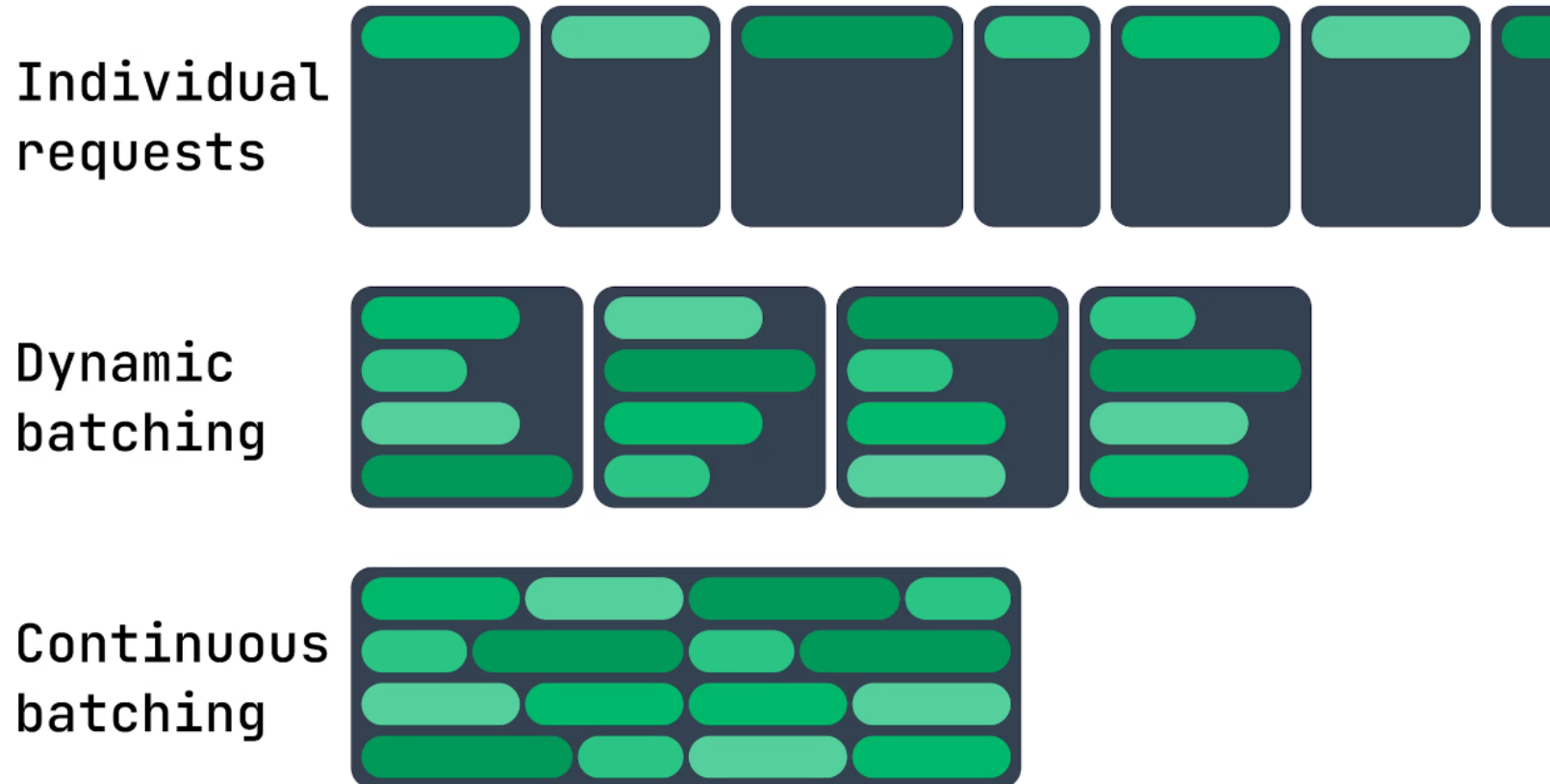
Efficiently Scaling Transformer Inference, MLSys 2023

# Inference Challenge: Limited Parallelism

- Small batch size  $\implies$  Low data reuse
- Autoregressive generation  $\implies$  Sequential dependency



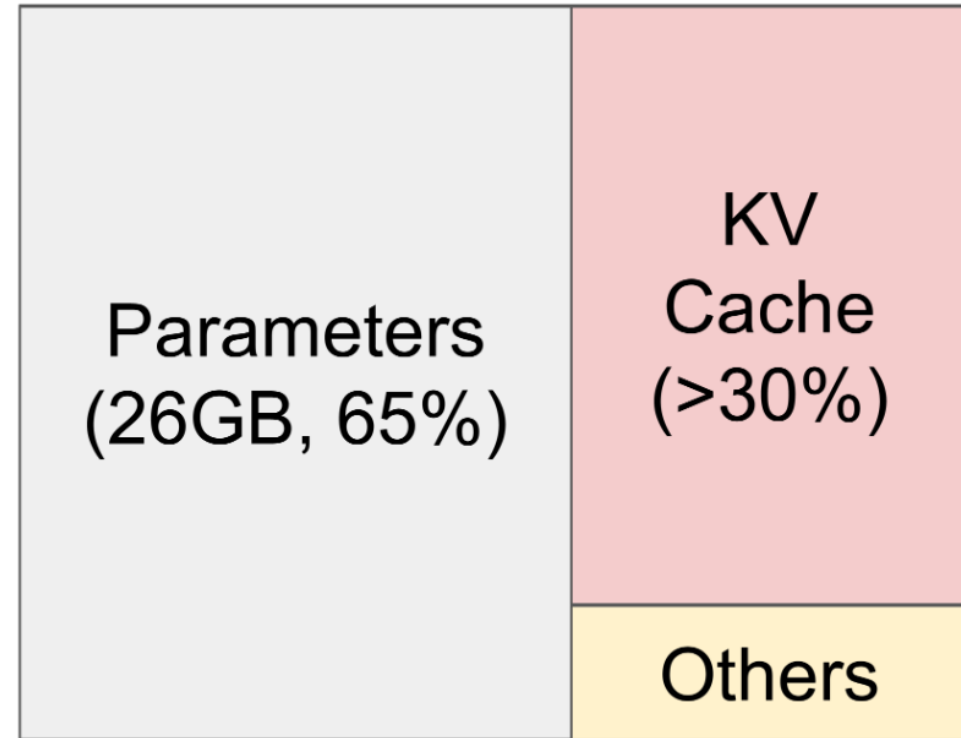
# Batching Strategies for LLM Inference



Orca: A Distributed Serving System for Transformer-Based Generative Models, OSDI 2022

# Inference Challenge: Large Memory Footprint

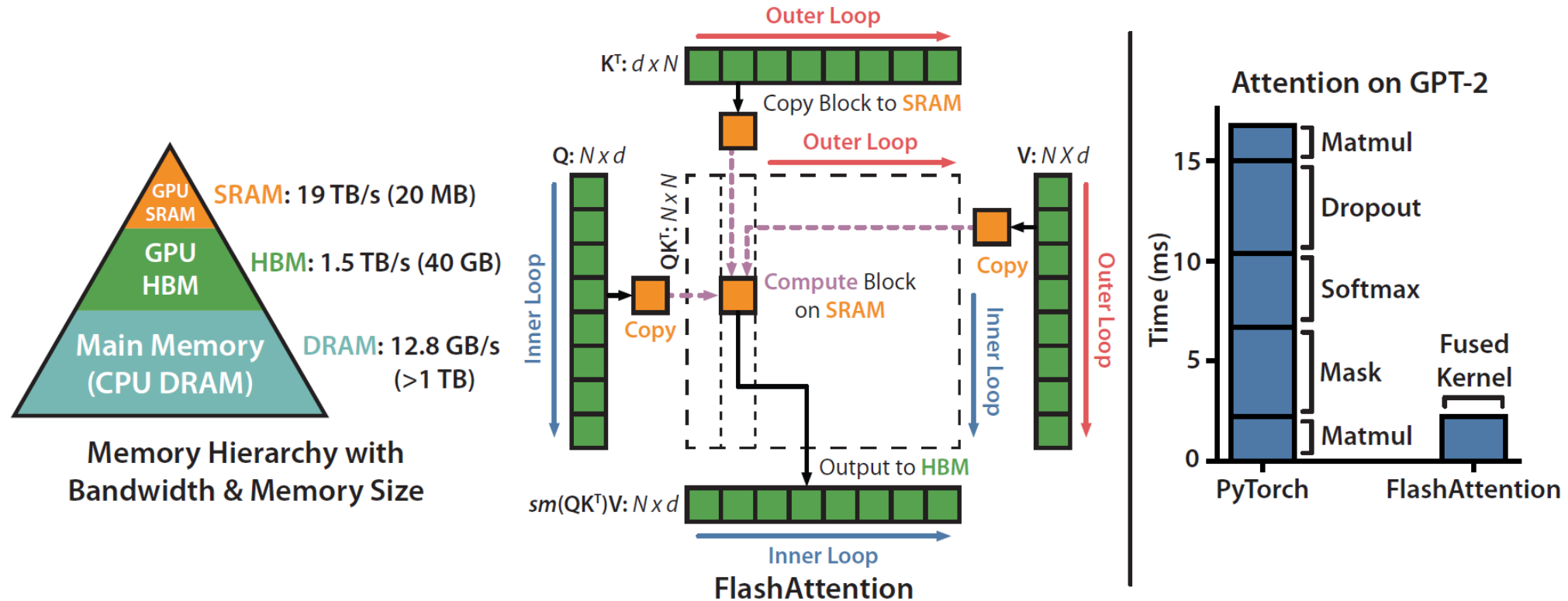
- Model parameters
  - # Layers
  - # Hidden dim
- KV cache
  - Batch size
  - Sequence length
  - # Layers
  - # Hidden
- Activation and others



OPT-13B on A100 40 GB

[Efficient Memory Management for Large Language Model Serving with PagedAttention](#), by Kwon et al., 2023

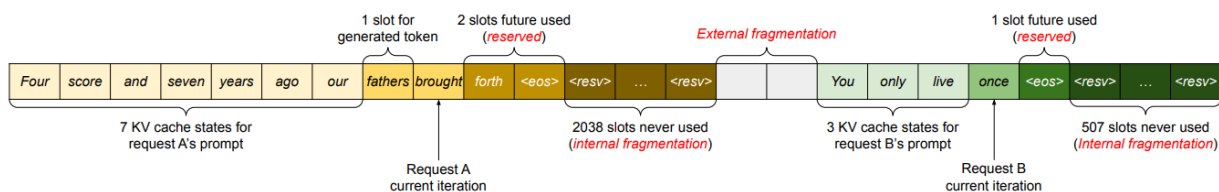
# FlashAttention



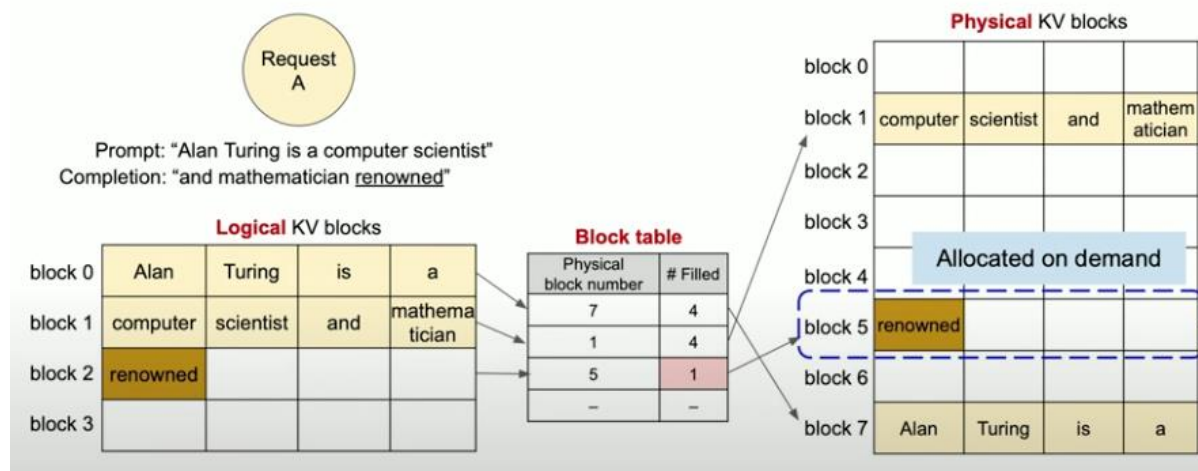
Fast and Memory-Efficient Exact Attention with IO-Awareness, 2023



# PagedAttention

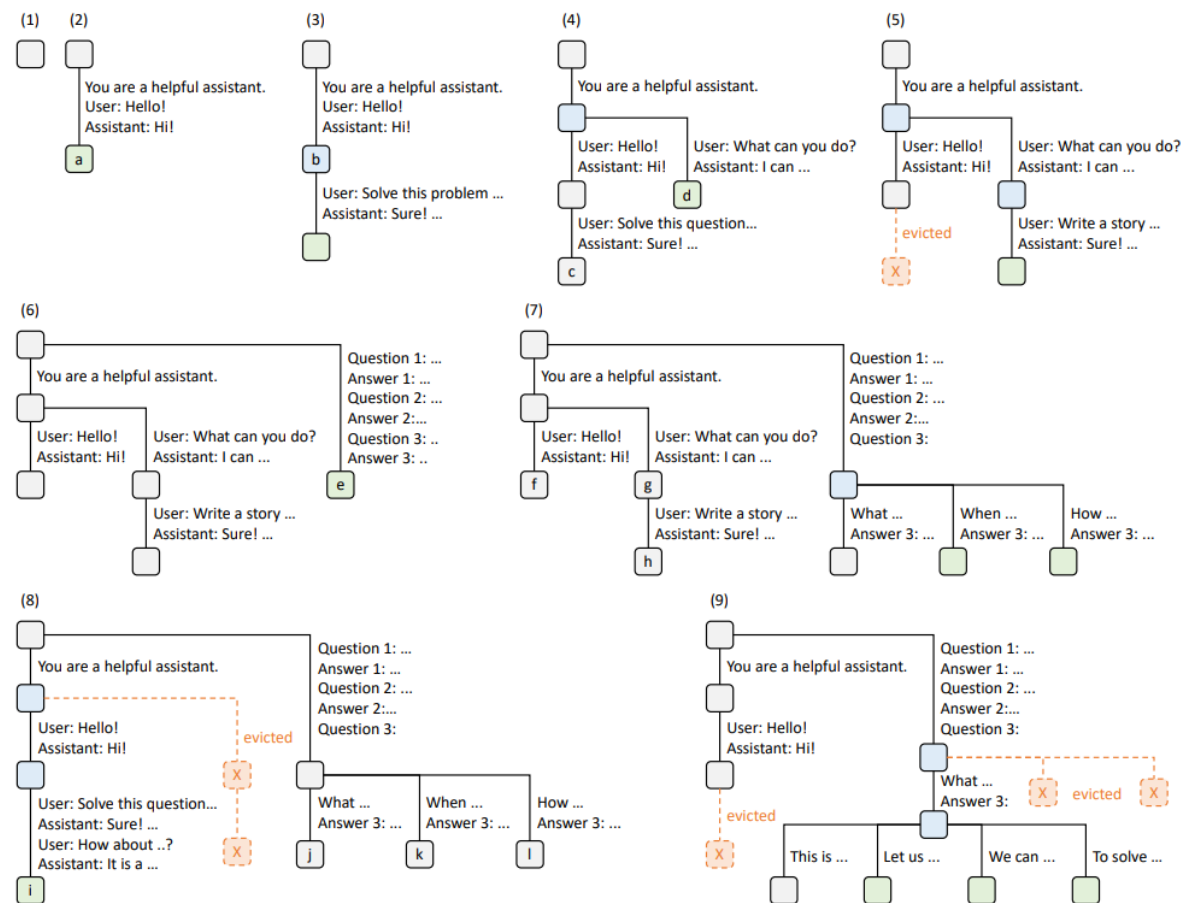


## Logical & physical KV blocks



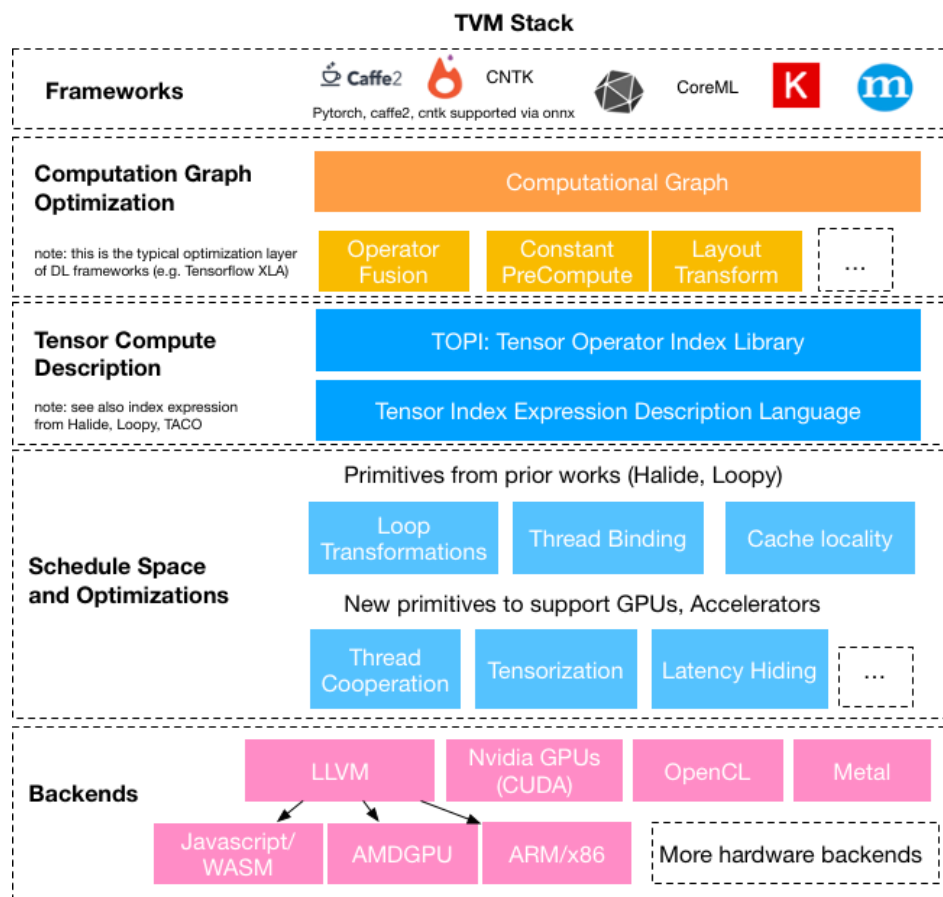
## Efficient Memory Management for Large Language Model Serving with PagedAttention, 2023

## SGLang: Efficient Execution of Structured Language Model Programs, 2024



# DL Compilation

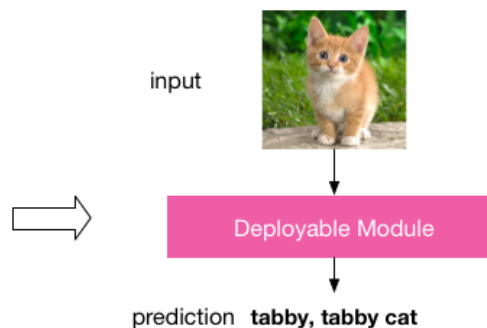
Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations, 2019



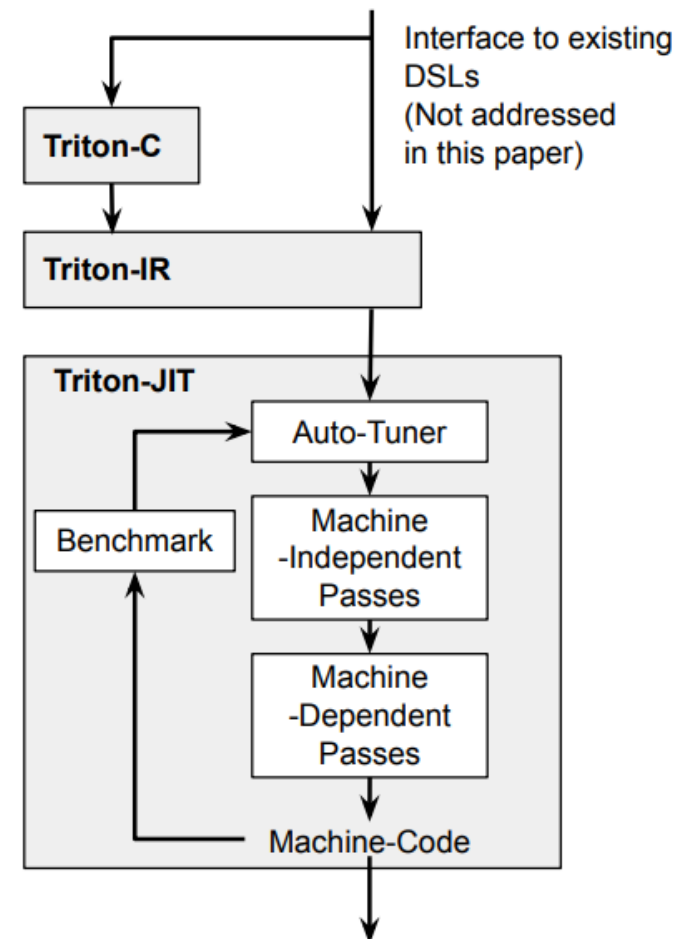
TVM: An Automated End-to-End Optimizing Compiler for Deep Learning, 2018

**Runtime: Lightweight and Cross Platform**

```
module = runtime.create(graph, lib, tvml.gpu(0))
module.set_input(**params)
module.run(data=data_array)
output = tvml.nd.empty(out_shape, ctx=tvml.gpu(0))
module.get_output(0, output)
```



**Deploy Languages and Platforms**



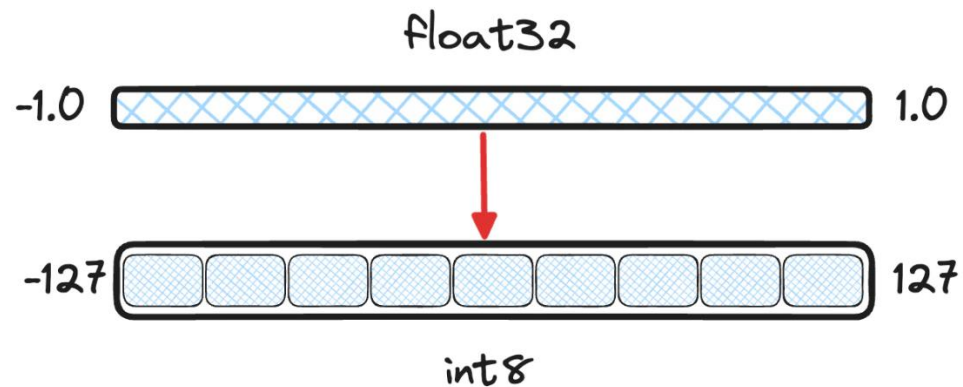
# Efficient and Effective Algorithms

# Compression Strategies

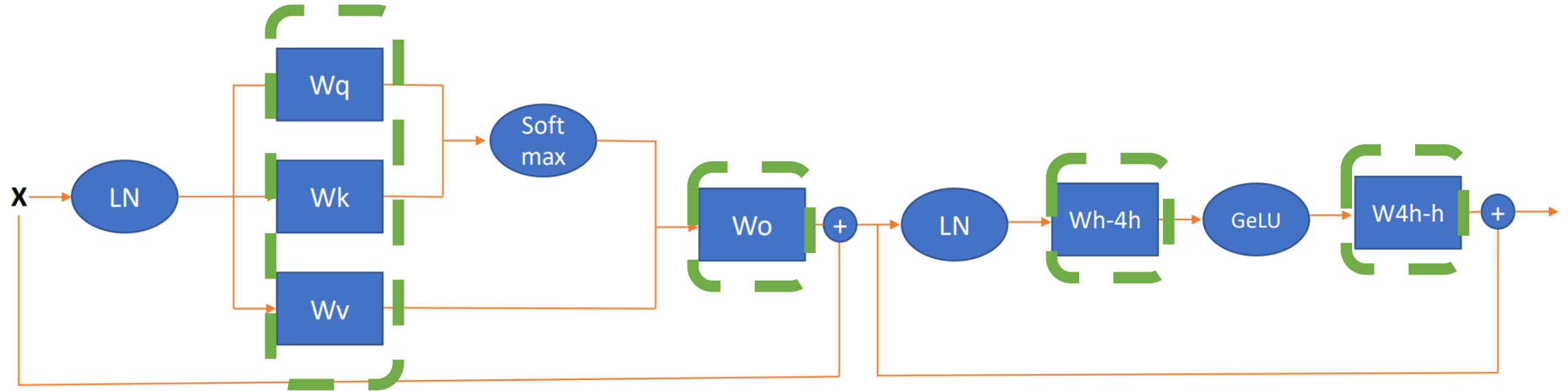
- Quantization
- Sparsification
- Distillation
- Low-Rank Decomposition
- ...

# Quantization: Quick Recap

- Reduce the bits per weight, saving memory consumption
- Accelerate inference speed on supporting hardware



# 8-bit Weight Quantization



- 8-bit weight quantization

$$\mathbf{x}_{quantize} = round \left( clamp \left( \frac{\mathbf{x}}{S}, -2^{bit-1}, 2^{bit-1} - 1 \right) \right)$$

FP32 weight matrix

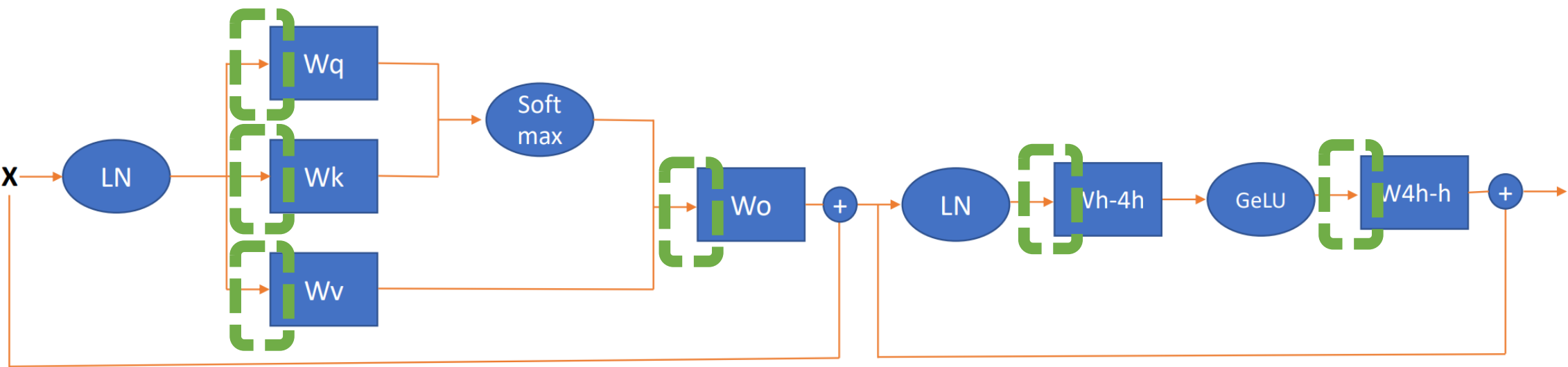
1.1	2.2	0.1	-0.1	-5.5	-6.6
...					
...					
...					
...					
1.1	2.1	0.1	-0.1	-4.8	-6.6

Scaling  
Factor  
 $1/S$   
 $\approx 0.05 *$

8-bit quantization

21	42	2	-2	-106	-127
...					
...					
...					
...					
21	40	2	-2	-92	-127

# 8-bit Activation Quantization



- 8-bit activation  
(Input to the linear layer)

$$x_{quantize} = round \left( clamp \left( \frac{x}{S}, -2^{bit-1}, 2^{bit-1} - 1 \right) \right)$$

FP32 input matrix

1.1	2.2	0.1	-0.1	-5.5	-6.6
...					
...					
...					
...					
1.1	2.1	0.1	-0.1	-4.8	-6.6

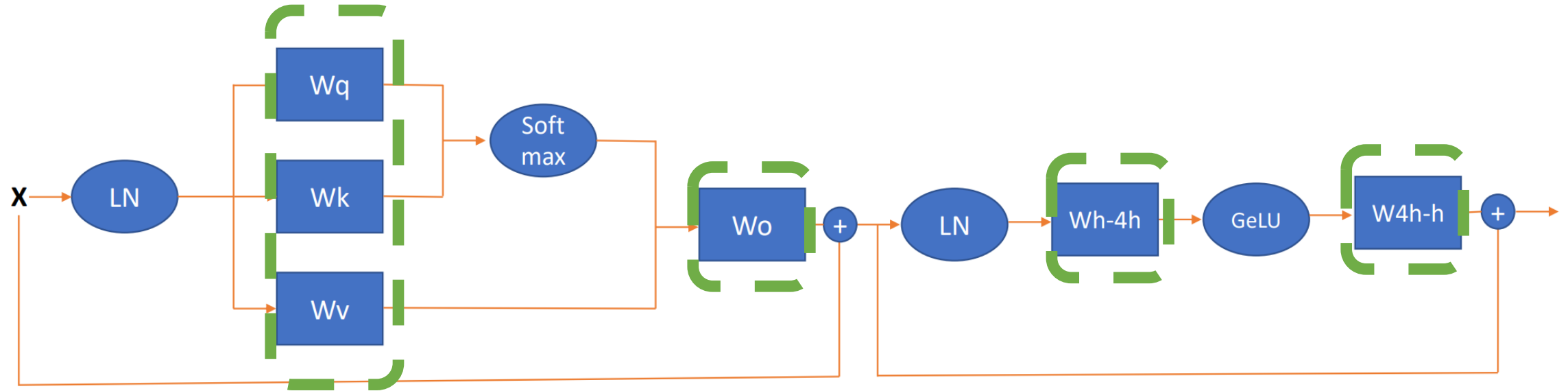
Scaling  
Factor  
 $1/S$

$\approx 0.05^*$

8-bit quantization

21	42	2	-2	-106	-127
...					
...					
...					
...					
21	40	2	-2	-92	-127

# Weight Ternarization



- Ternarization (weight)**

$W$ : weight matrix, FP32.

$Q(W)$ : Quantization mapping, 2-bit.

With  $\alpha = \|W\|_1/n$ , for some scalar  $s$

$$Q(W_{ij}) = \begin{cases} \alpha \cdot \text{sign}(W_{ij}) & \text{when } |W_{ij}| > s \\ 0 & \text{when } |W_{ij}| < s \end{cases}$$

FP32 weight matrix

1.1	2.2	0.1	-0.1	-5.5	-6.6
...					
...					
...					
...					
1.1	2.1	0.1	-0.1	-4.8	-6.0

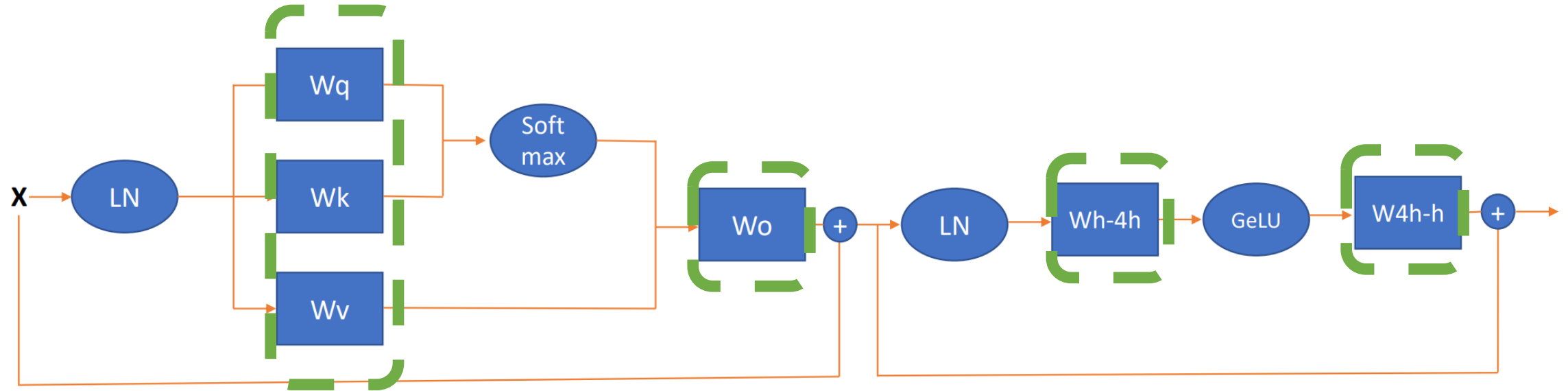
Scaling  
Factor  $\alpha$   
 $\approx 2.06 *$

2-bit quantization

1	1	0	0	-1	-1
...					
...					
...					
...					
...					
1	1	0	0	-1	-1



# Weight Binarization



- Binarization (weight)**

$W$ : weight matrix, FP32.

$Q(W)$ : Quantization mapping, 1-bit.

With  $\alpha = \|W\|_1/n$

$$Q(W_{ij}) = \alpha \cdot \text{sign}(W_{ij})$$

FP32 weight matrix

1.1	2.2	0.1	-0.1	-5.5	-6.6
...					
...					
...					
...					
1.1	2.1	0.1	-0.1	-4.8	-6.0

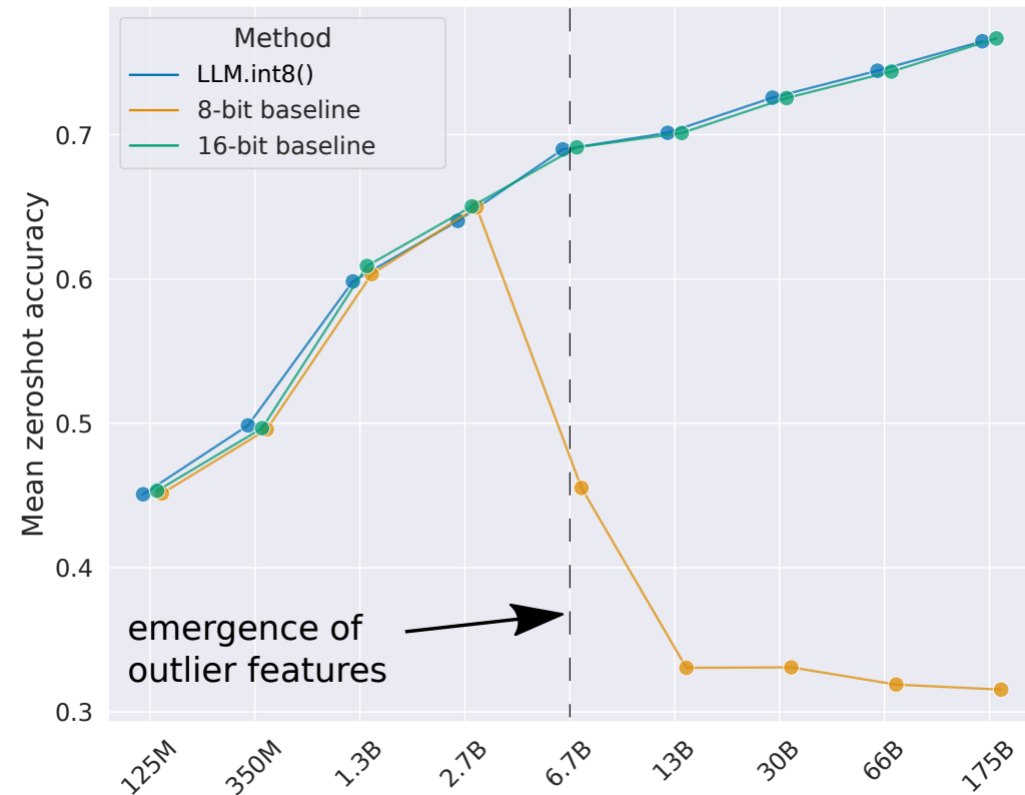
1-bit quantization

$\approx 1.4 \times$   
Scaling  
Factor  $\alpha$

1	1	1	-1	-1	-1
...					
...					
...					
...					
1	1	1	-1	-1	-1

# Challenges to Quantize LLMs

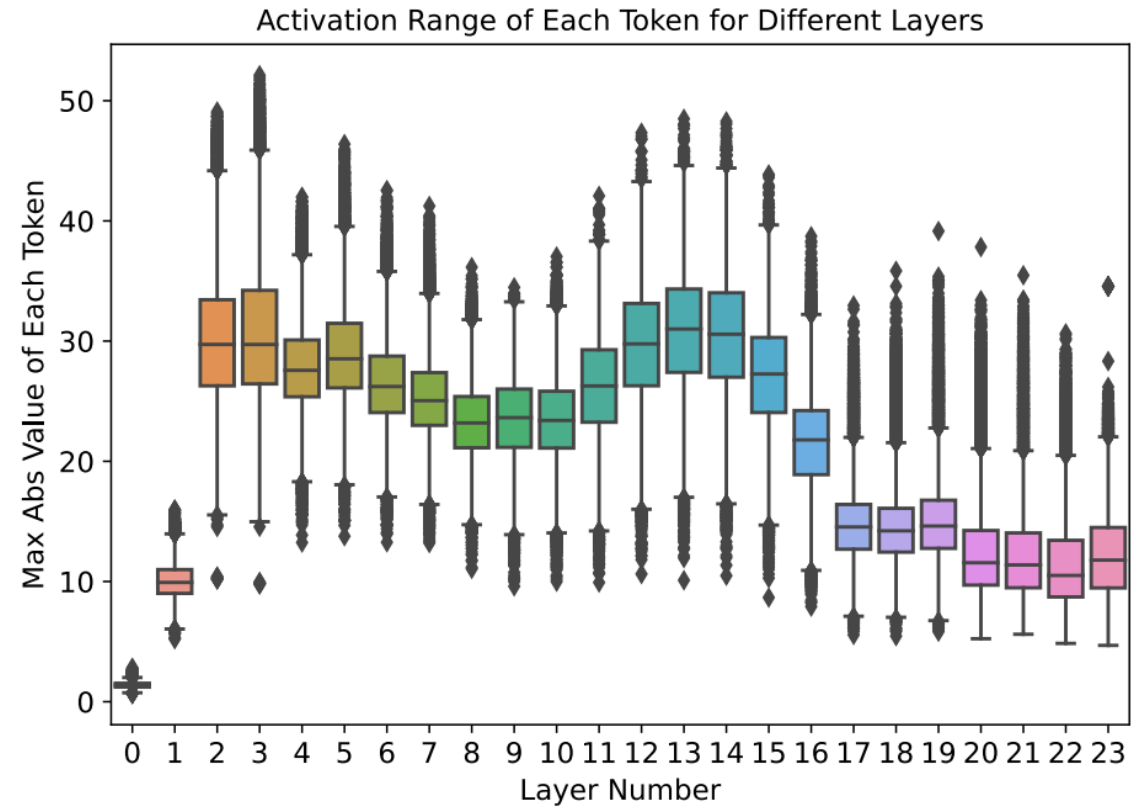
- Standard quantization strategy leads to catastrophic accuracy drop



LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, 2023

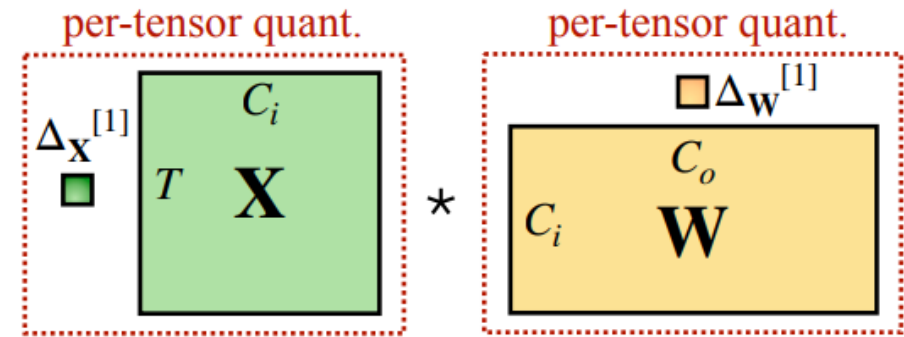
# Challenges to Quantize LLMs

- High dynamic ranges of activation, leading to large quantization errors

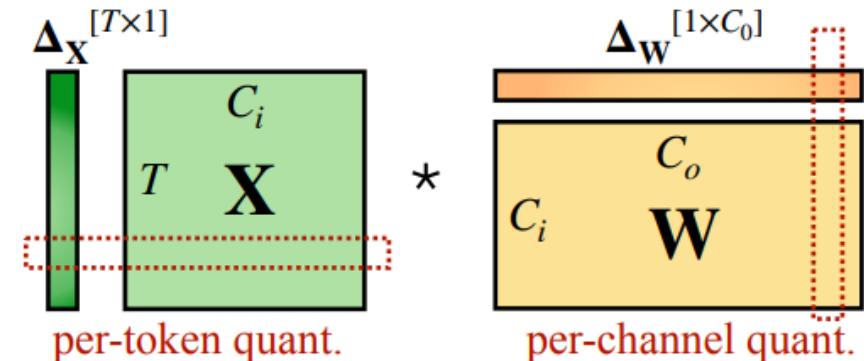


# Fine-grained Quantization

- Per-tensor quantization
  - Low accuracy
  - Fast to quantize/dequantize
- Per-token/channel quantization
  - High accuracy
  - Slower to quantize/dequantize
  - Custom kernels required



(a) per-tensor quantization

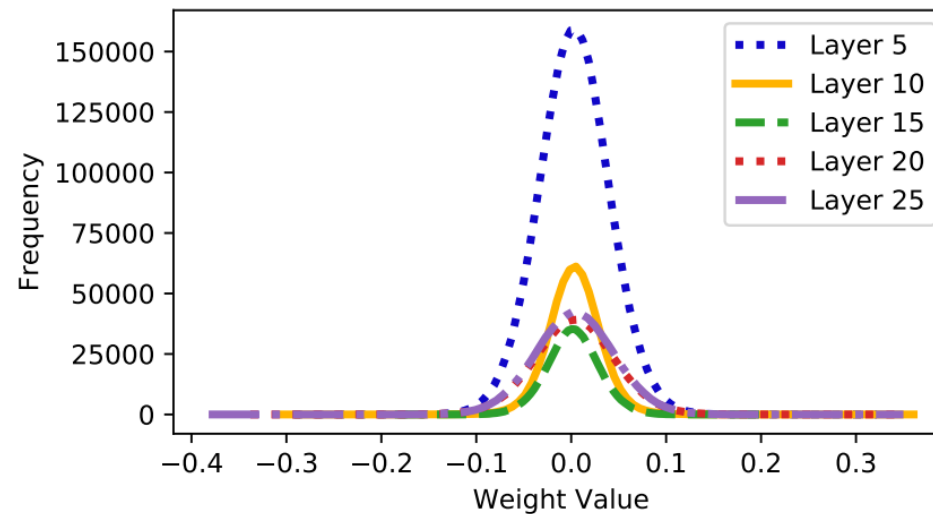


(b) per-token + per-channel quantization

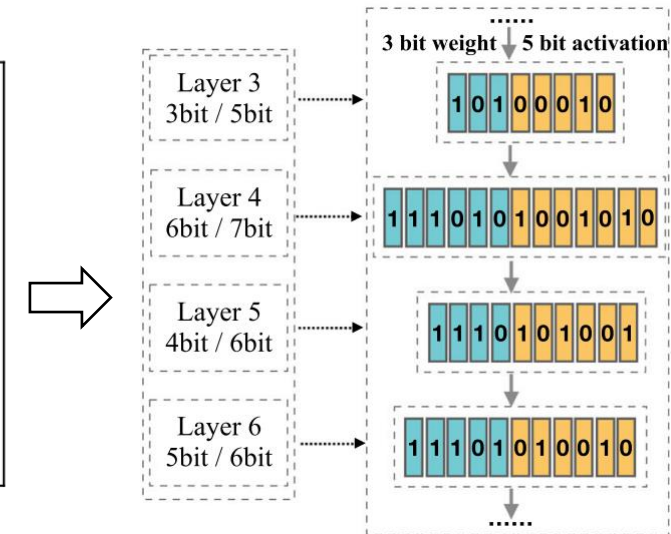
ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers, NeurIPS 2022

# Mixed Precision Quantization

- Weights follow Gaussian distribution
- Outliers remain in original form, quantize the rest of the values
- Different bits for different layers



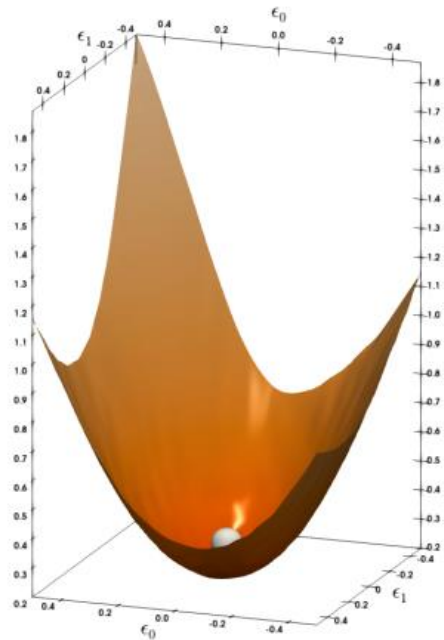
Per-layer weight distribution of BERT model



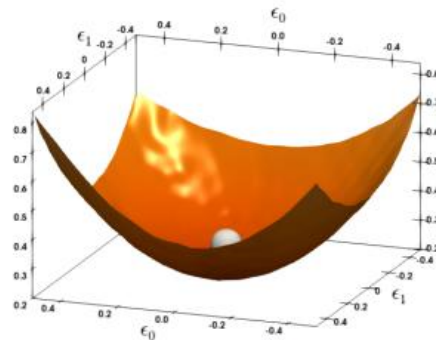
GOBO: Quantizing Attention-Based NLP Models for Low Latency and Energy Efficient Inference, MICRO 2020

# Second Order Information

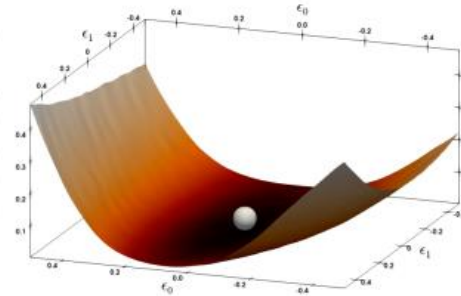
- Analyze the loss curvature (Hessian matrices) to help identify layer sensitivity



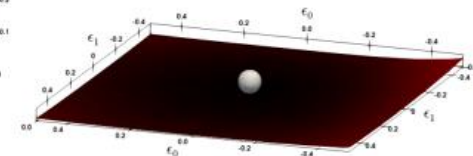
(a) MNLI 4<sup>th</sup> layer



(b) MNLI 10<sup>th</sup> layer



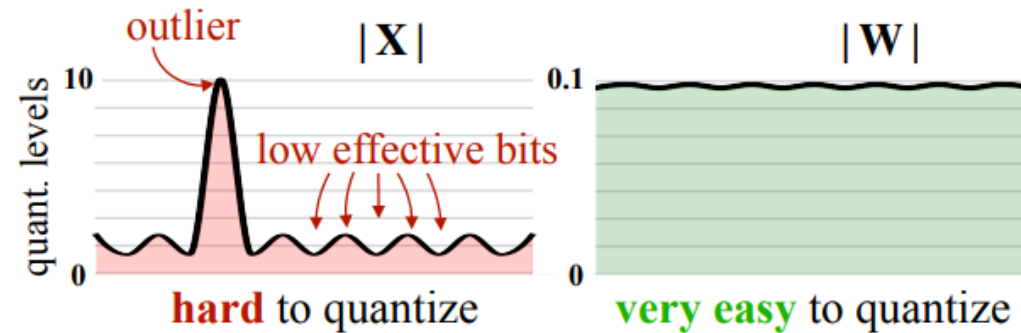
(c) CoNLL-03 4<sup>th</sup> layer



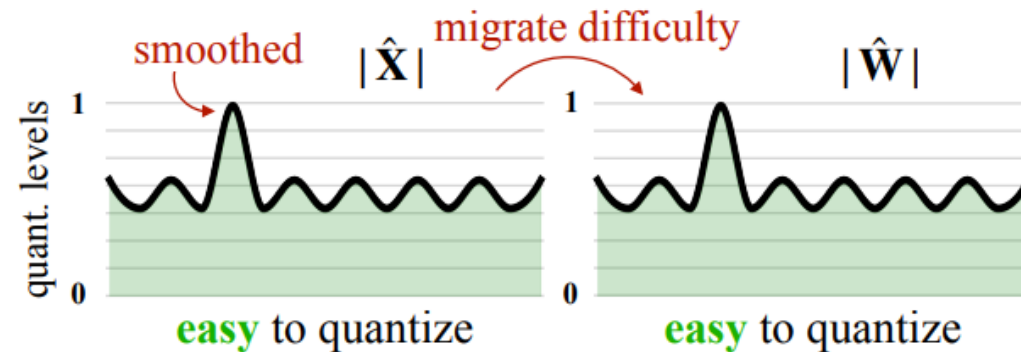
(d) CoNLL-03 11<sup>th</sup> layer

GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers, ICLR 2023

# Outlier Smoothing



(a) Original



(b) SmoothQuant

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models, ICML 2023

# Model Pruning

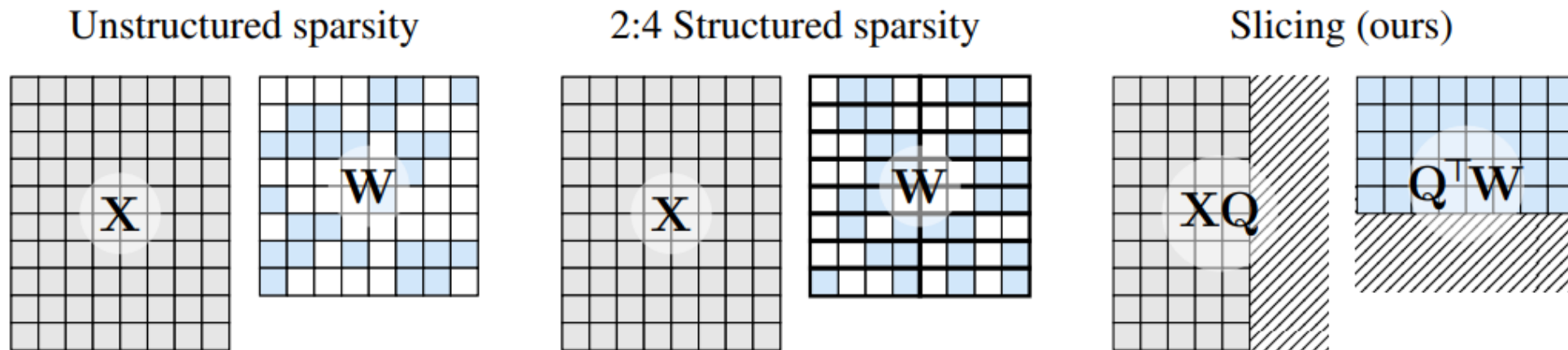
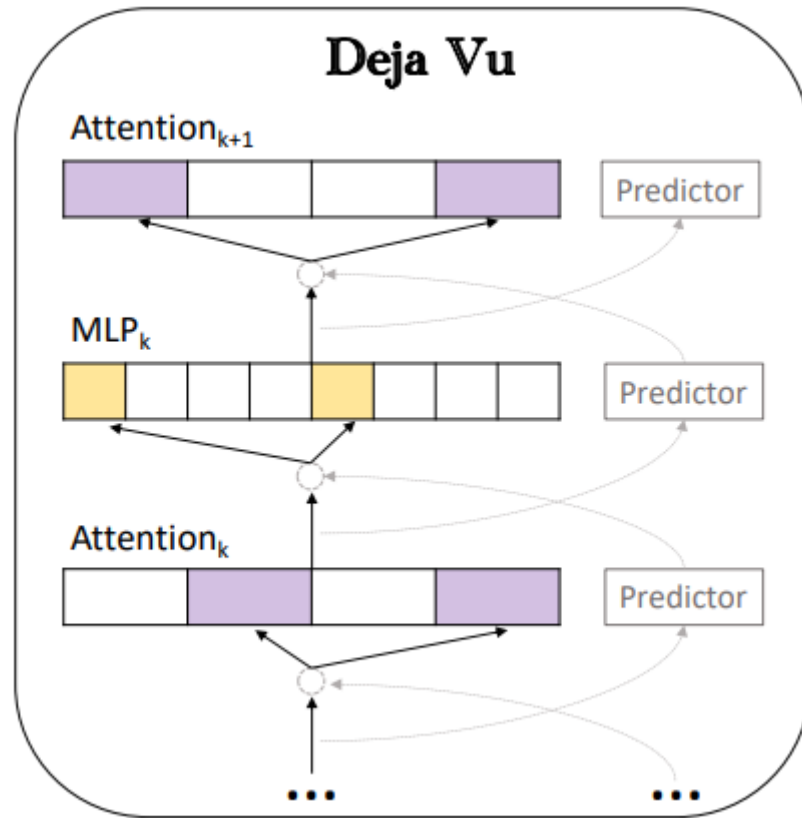


Figure 1: Matrix multiplication of the signal  $\mathbf{X}$  and a weight matrix  $\mathbf{W}$  under different types of sparsity. **Left:** unstructured sparsity, where some elements of  $\mathbf{W}$  are zero, and  $\mathbf{X}$  is dense. **Middle:** 2:4 structured sparsity, where each block of four weight matrix entries contains two zeros, and  $\mathbf{X}$  is dense. **Right:** SliceGPT, where after introducing transformation  $\mathbf{Q}$ , all the sparsity is arranged to the bottom rows of  $\mathbf{W}$  and the corresponding columns of  $\mathbf{X}$  are removed.

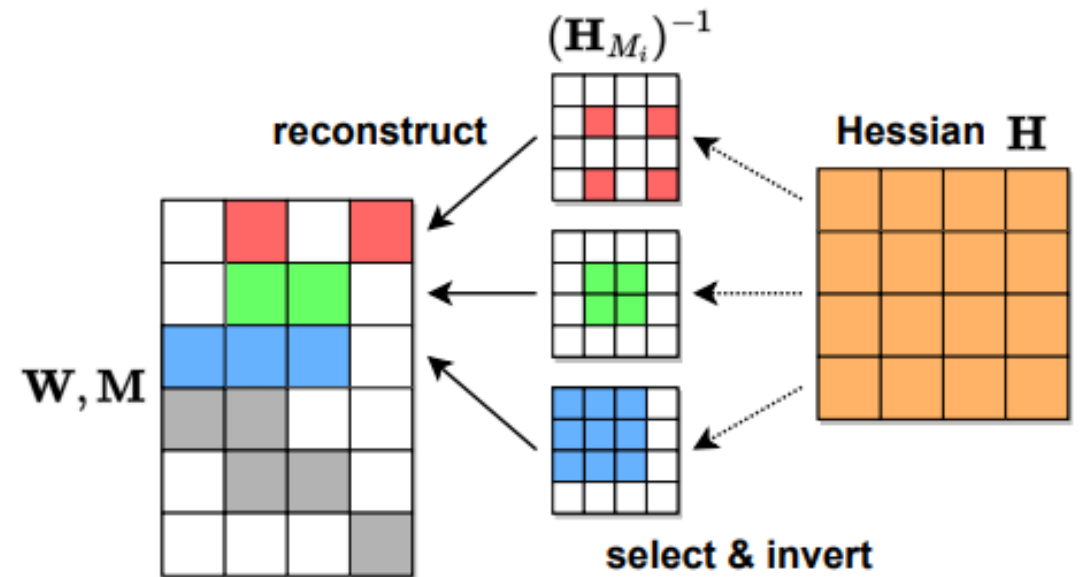


# Model Pruning



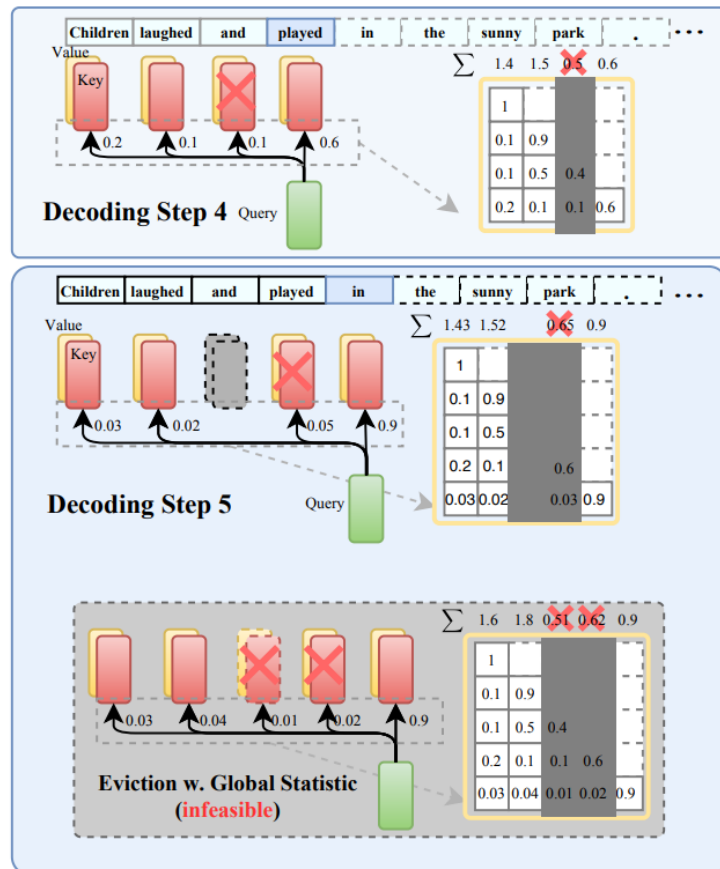
Deja Vu: Contextual Sparsity for Efficient LLMs at Inference Time, 2023

SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot, 2023

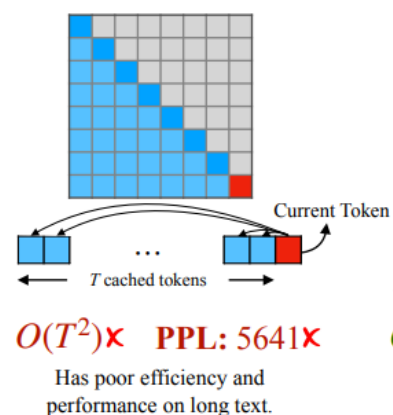


# KV Cache Compression

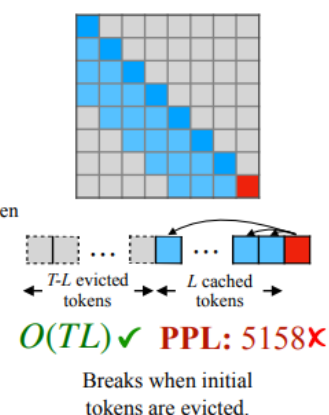
## Efficient Streaming Language Models with Attention Sinks, ICL 2024



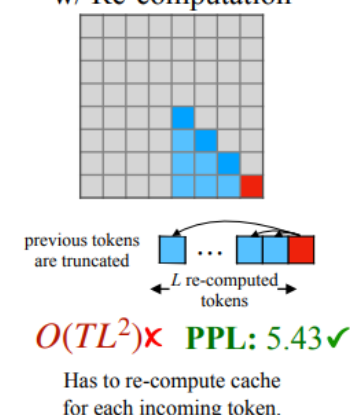
(a) Dense Attention



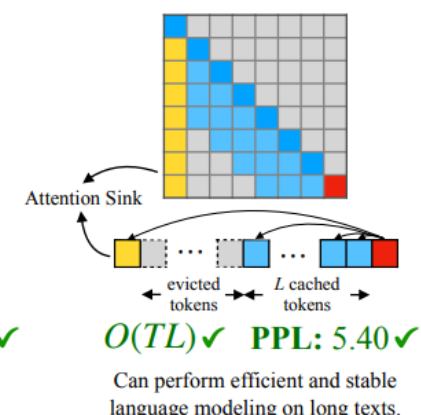
(b) Window Attention



(c) Sliding Window w/ Re-computation

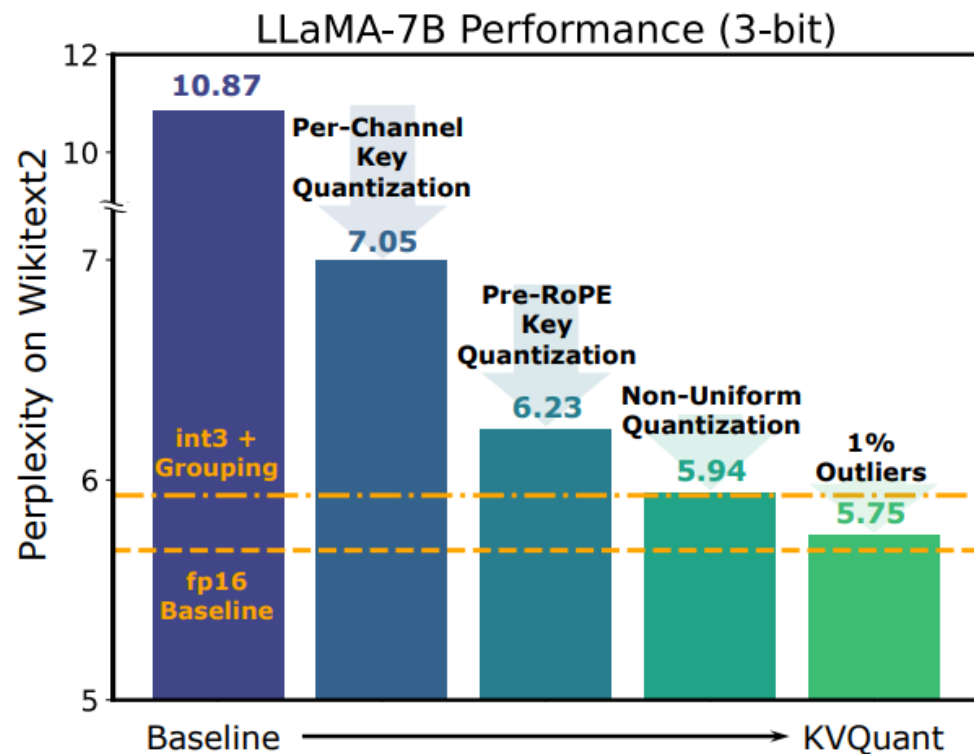


(d) StreamingLLM (ours)

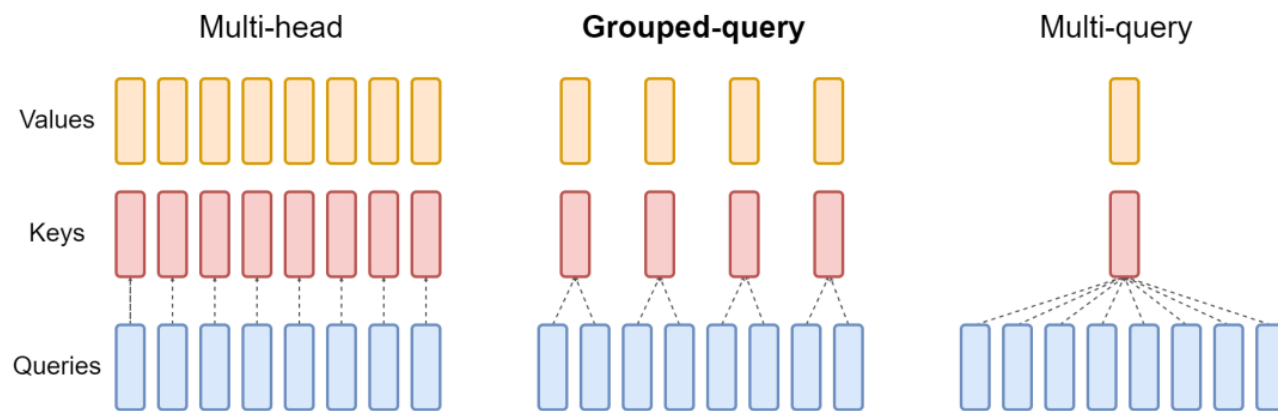


H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models, 2023

# KV Cache Compression



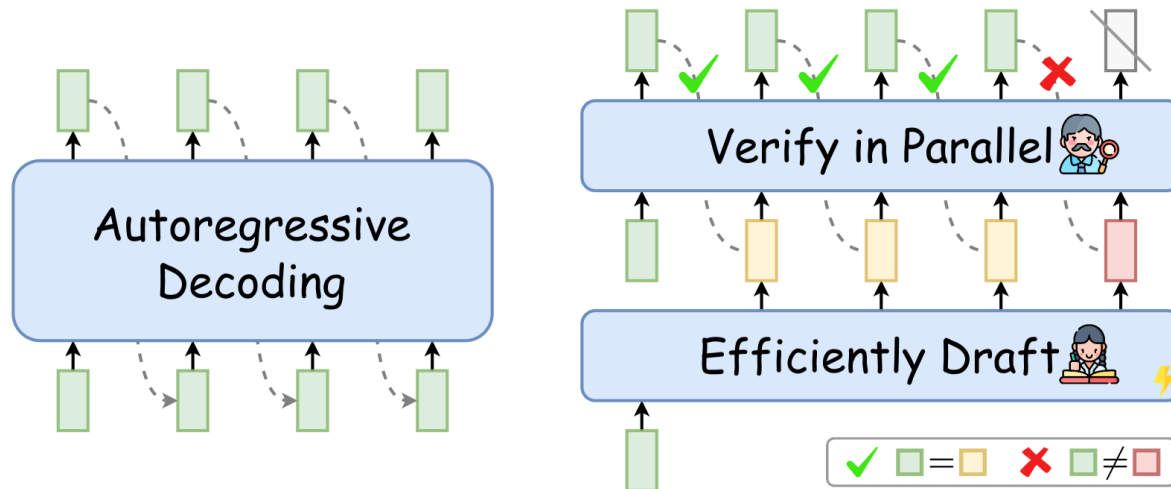
GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints, 2023



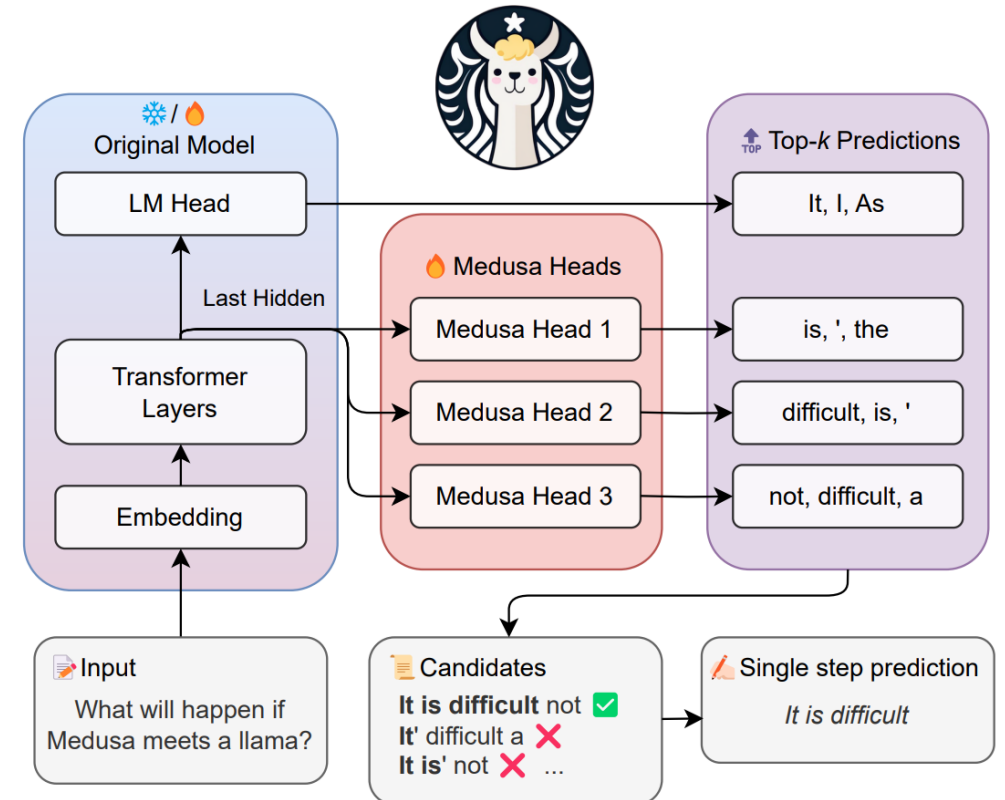
KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization, 2024

# Speculative/Parallel Decoding

## MEDUSA: Simple LLM Inference Acceleration Framework with Multiple, 2024



Fast Inference from Transformers via Speculative Decoding, 2023



# Inference Optimizations

- Reduce the inference latency to satisfy latency SLA
- Improve the inference throughput to save cost
- Reduce the memory footprint of the model by using fewer GPU devices and less GPU memory
- Improve agility from DNN prototype to deployment

QA