[Tutorial] Inference Optimization for Foundation Models on AI Accelerators

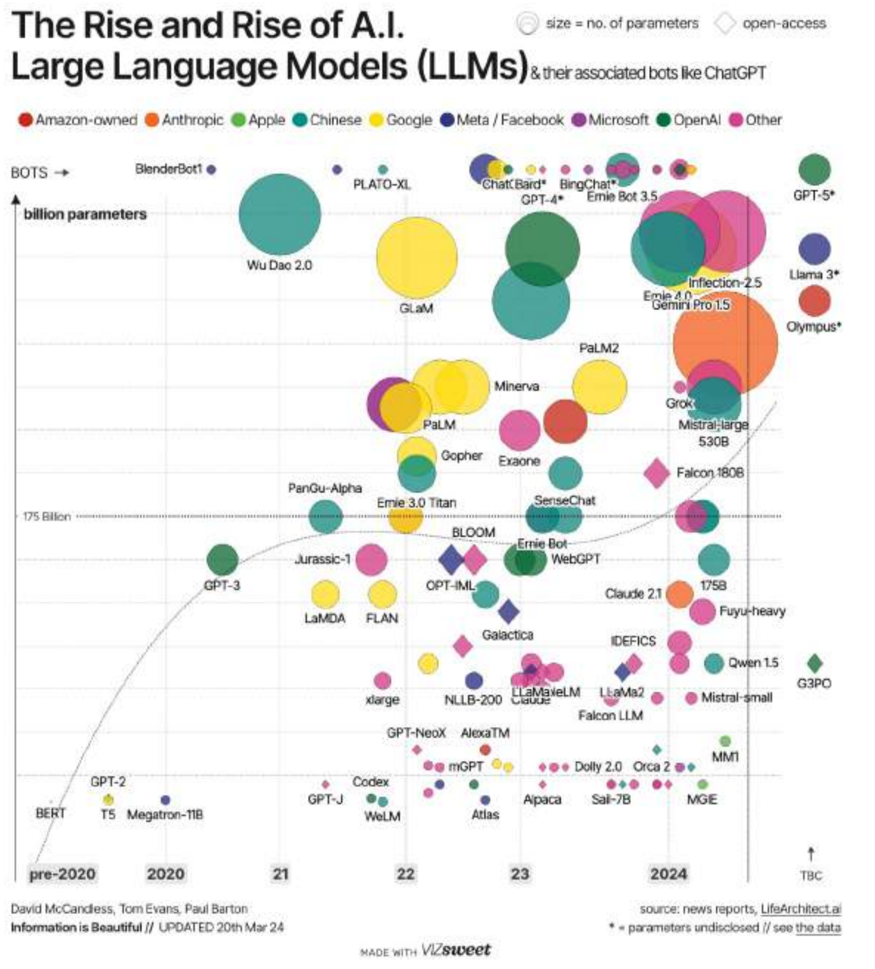
Yan Xu

Houston Machine Learning

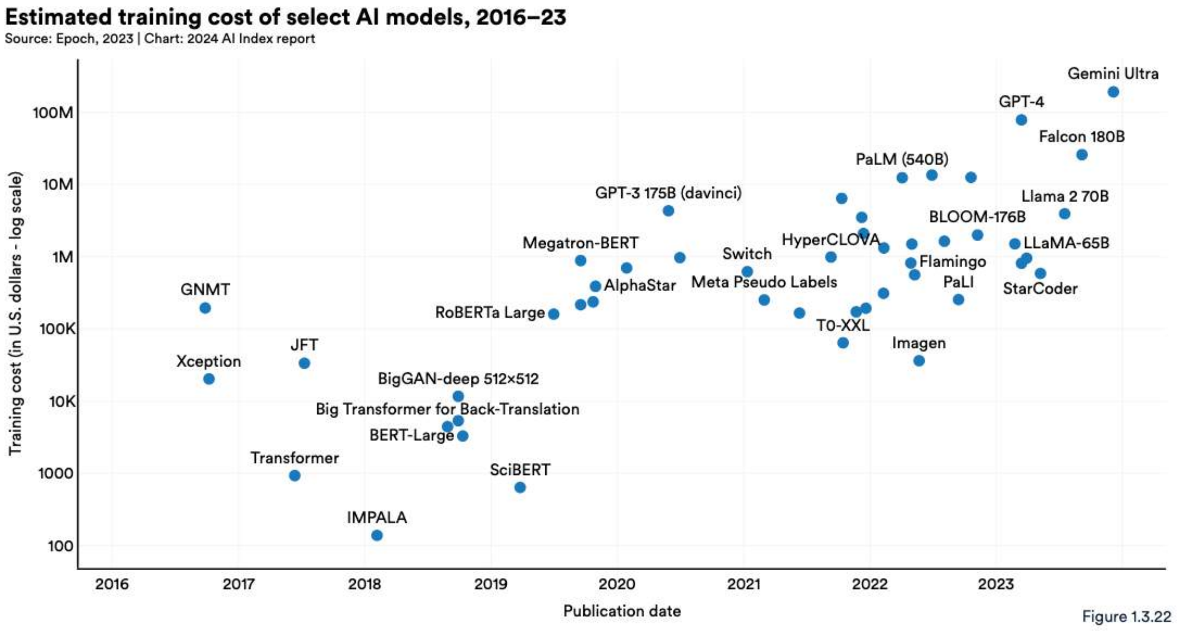
Sep 6, 2024

Reference: KDD tutorial - Inference Optimization of Foundation Models on AI Accelerators.

Rise of Large Language Models and Cost



The rise and rise of AI-based Large Language Models (LLMs) like GPT4, LaMDA, LLaMa, PaLM and Jurassic-2.



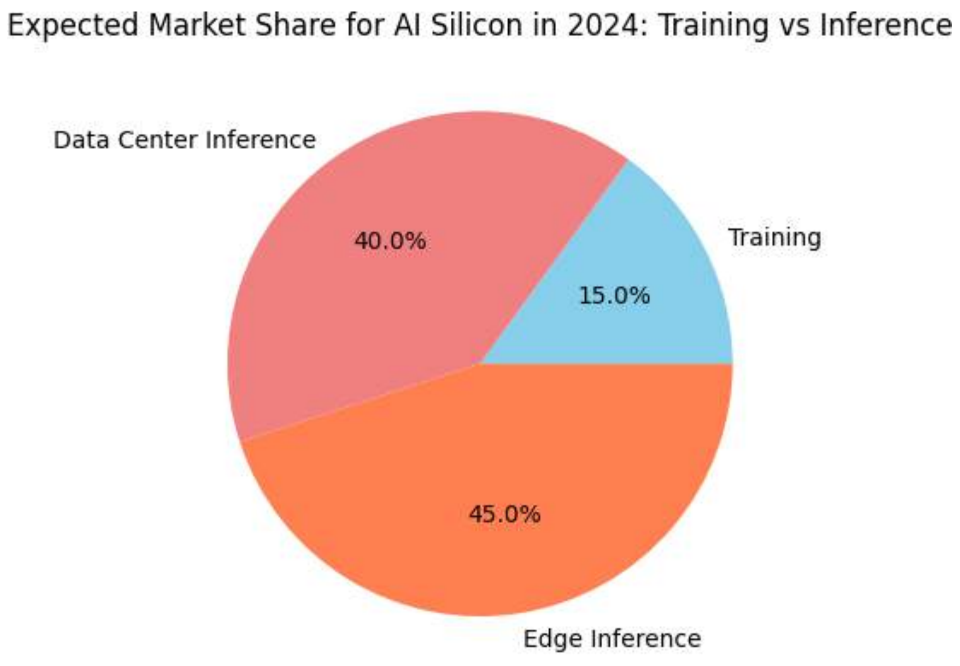
[Image credit:
(left) informationisbeautiful.net, LifeArchitect.ai
(right) Stanford AI Report, 2024]

amazon | science

Why inference optimization?

- Projected market size of inference vs training
  - LLM pre-training: one time, or update every month
  - LLM inference: far more frequent for serving millions of users with low latency.

- Inference optimization is essential to meet serving criteria of latency and energy cost
  - E.g., GPT3 (175 billion parameters), inference optimization can reduce from a few second to millisecond or lower to serve ChatGPT.



[Source: “The AI chip market landscape”, TECHSPOT, 2023]

Overview of attention mechanism

- Self-attention captures relationship within input seq $X = [x_1, \ldots, x_L]$
- Attention procedures
  - Step 1: for each input, compute query (q_j), key (k_j), and value (v_j) via linear projection.
  - Step 2: compute attention scores $\{\alpha_{ij}\}$ across inputs w.r.t. current input query x_i.
  - Step 3: retrieve output (z_j) by weighting values $\{v_j\}$ with scores $\{\alpha_{ij}\}$.

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

$$Z = \mathrm{Attention}(Q, K, V) = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

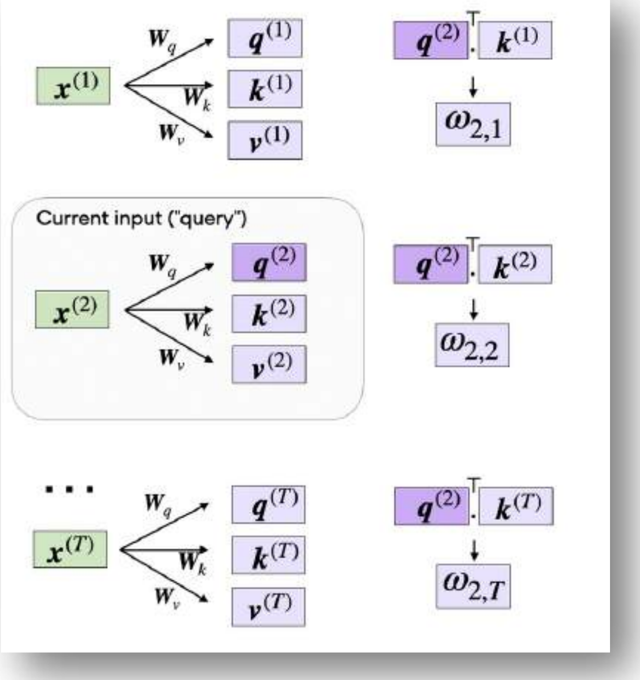


Image source: [Sebastian Raschka, /sebastianraschka.com/blog/2023]

Overview of attention mechanism

- Self-attention captures relationship within input seq $X = [x_1, \ldots, x_L]$
- Attention procedures
  - Step 1: compute query (q_j), key (k_j), and value (v_j) for each input via linear projection.
  - Step 2: compute attention scores $\{\alpha_{ij}\}$ across inputs w.r.t. current input query q_i.
  - Step 3: retrieve output (z_j) by weighting values $\{v_j\}$ with scores $\{\alpha_{ij}\}$.

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

$$Z = \text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

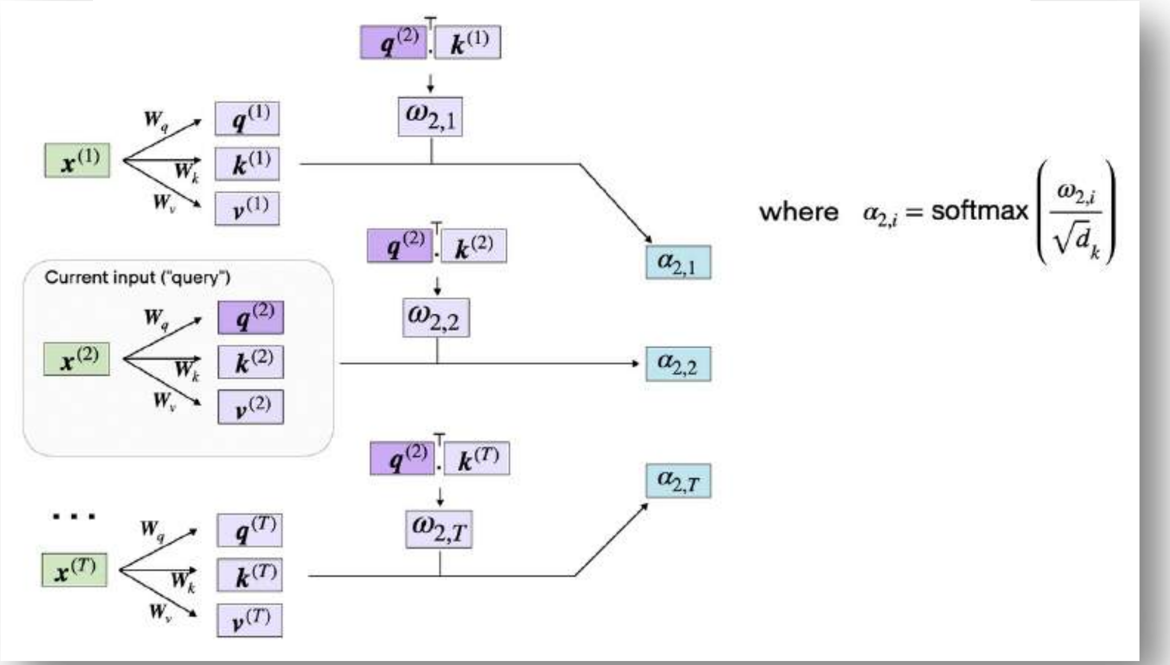


Image source: [Sebastian Raschka, /sebastianraschka.com/blog/2023]

Overview of attention mechanism

- Self-attention captures relationship within input seq $X = [x_1, \ldots, x_L]$
- Attention procedures
  - Step 1: compute query (q_j), key (k_j), and value (v_j) for each input via linear projection.
  - Step 2: compute attention scores $\{\alpha_{ij}\}$ across inputs w.r.t. current input query x_i.
  - Step 3: retrieve output z_i by combining values $\{v_j\}$ with attention scores $\{\alpha_{ij}\}$.

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

$$Z = \text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

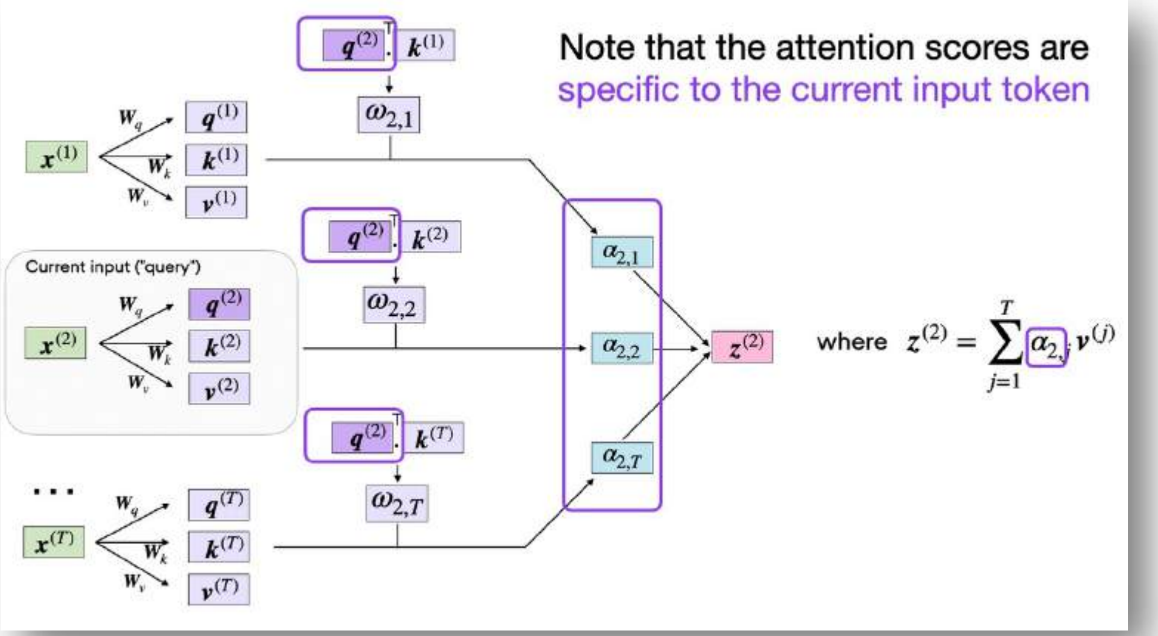


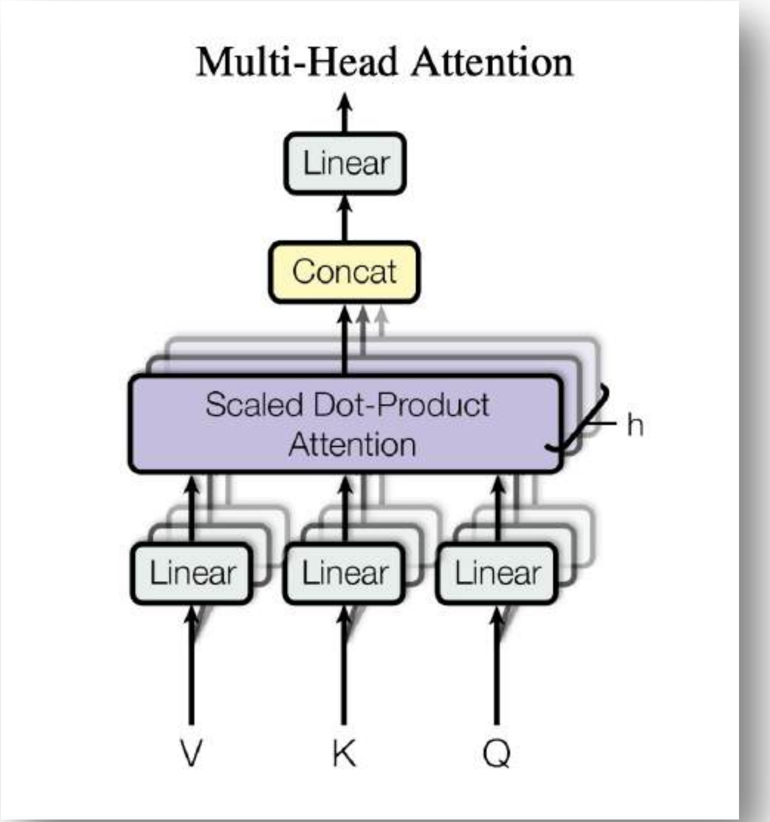
Image source: [Sebastian Raschka, /sebastianraschka.com/blog/2023]

Multi-head attention (MHA) block

- MHA has these h (self-)attentions
  - Hidden dimension (D) for queries, keys, and values is evenly split into multiple parts—each corresponding to a different *attention head*.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

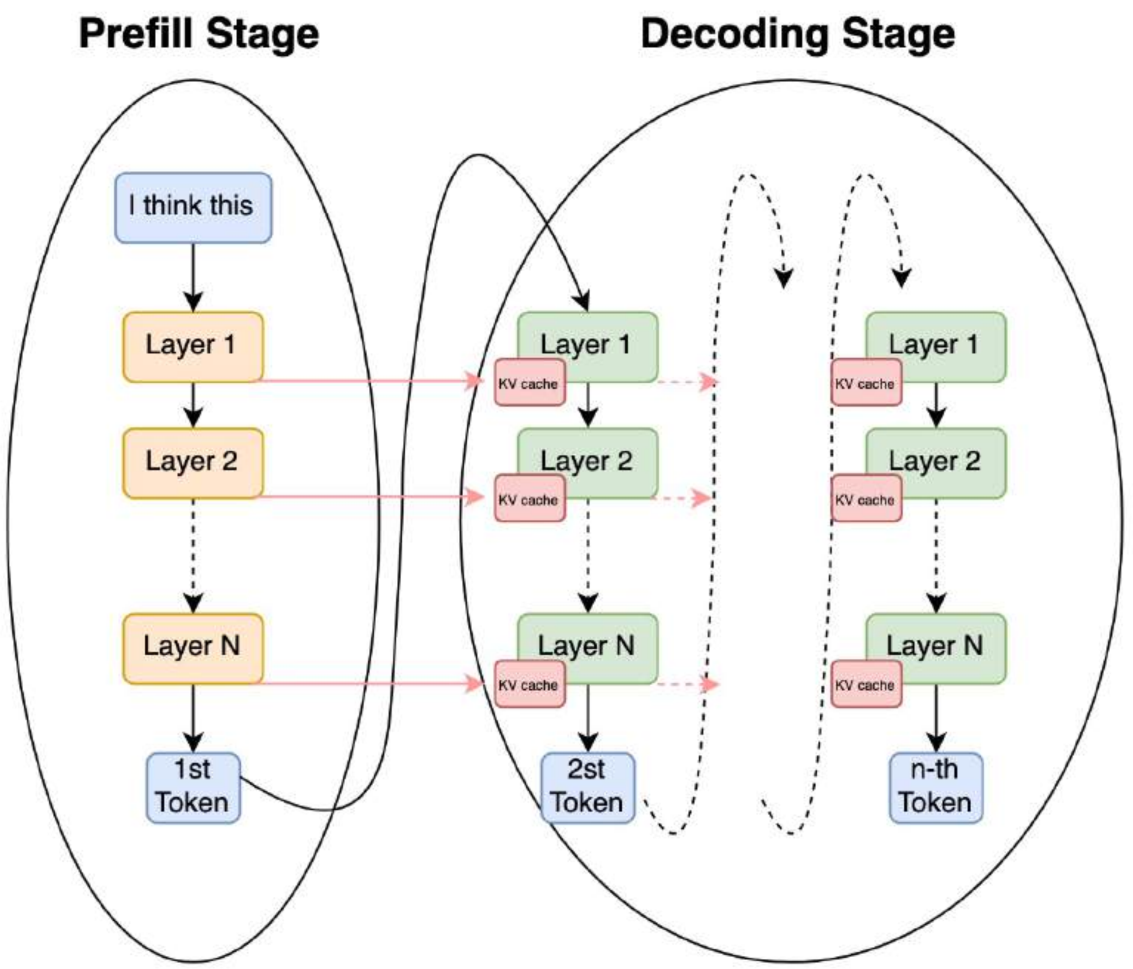
  - Mult-heads, instead of single head, allows to capture distinct patterns within sequence



[Vaswani et al., 2017] the Transformer architecture

Overview of inference computations

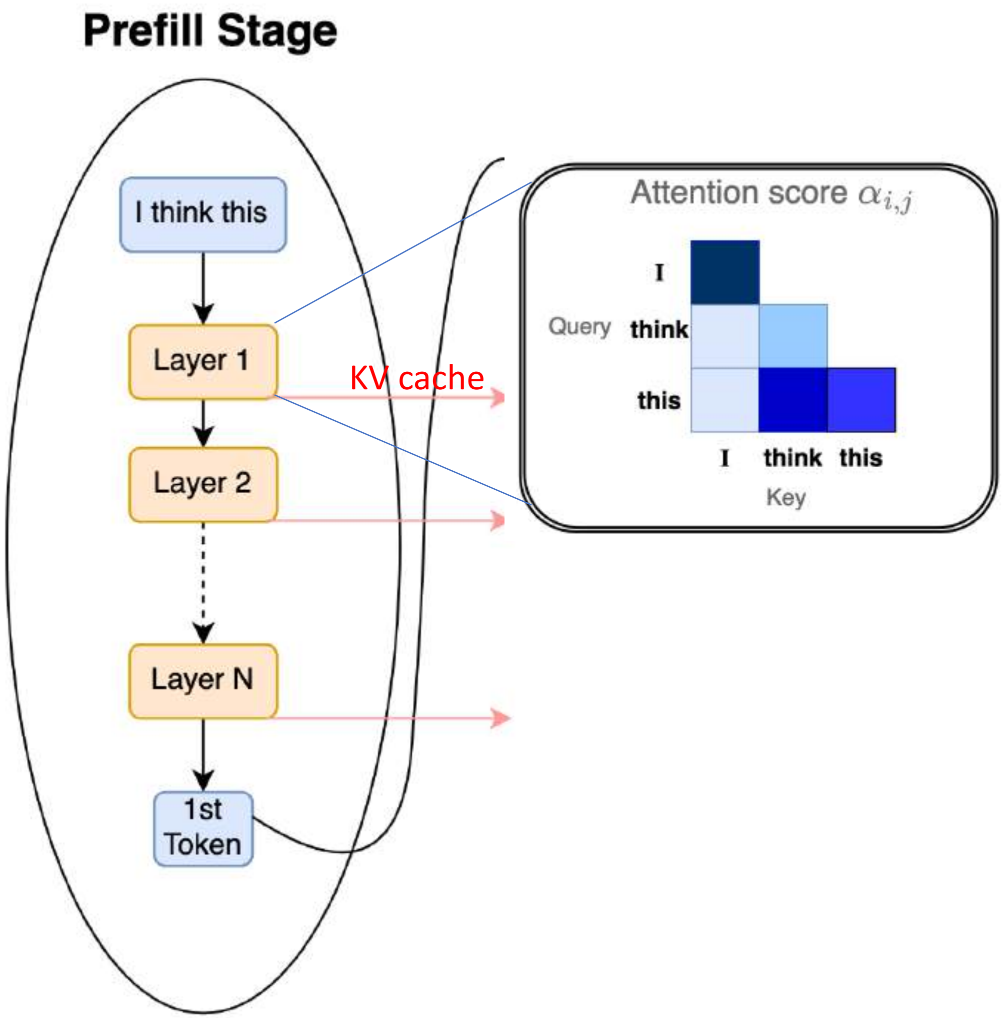
- Inference task
  - Given an input sequence X, use LLMs to generate next n-continuation Y.
- Two-step solution
  - Step 1. Prefill stage
  - Step 2. Decoding stage



amazon | science

Overview of inference computations

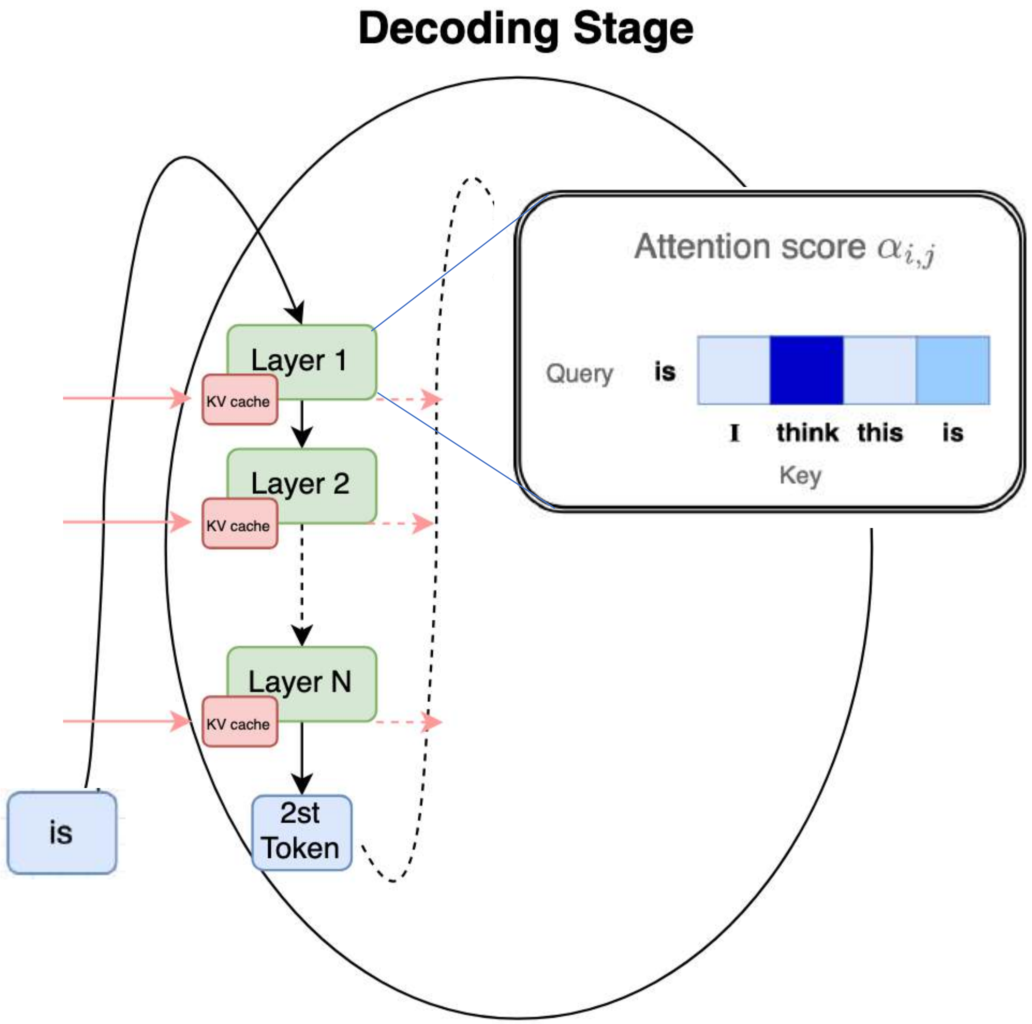
- Inference task
  - Given an input sequence X, use LLMs to generate next n-continuation Y.
- Two-step solution
  - Step 1. Prefill stage:
    - Perform a forward pass on X and save key and value in each layer for input tokens (KV cache).
    - KV cache can be used for the decoding stage



amazon | science

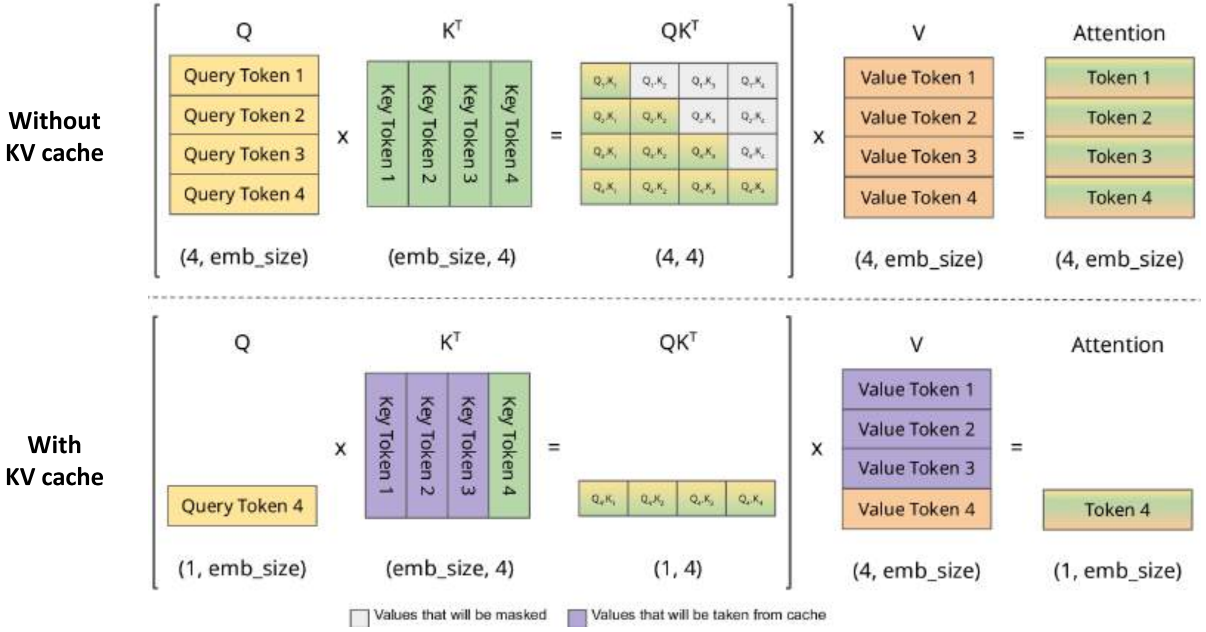
Overview of inference computations

- Inference task
  - Given an input sequence X, use LLMs to generate next n-continuation Y.
- Two-step solution
  - Step 1. Prefill stage
  - Step 2. Decoding stage
    - Generate tokens of Y (and its representation) in an autoregressive manner i.e., sequentially one-at-a-time
    - Can reuse some of previous representation, KV cache
    - Use a token sampling method, e.g., greedy, beam search, etc



Importance of KV-cache

- Essential for reducing compute complexity of generating successive token → linear from quadratic.
- However, inference with KV-cache consumes more memory.
  - Memory for KV-cache for LLaMa2 7B:
    - 2K input → 1GB
    - 8K input → 8GB
    - Batch of 4x8K sequences → 32GB. ← larger than the model's parameters (14GB) !
  - Can reduce device utilization



For n^{th} query token, KV- cache [pope eta al, 202] stores and reuses these past key-value pairs, eliminating the need of recalculation for every new token.
Image course: [João Lages, https://medium.com/@joaolages/kv-caching-explained]

The many ways to optimize inference

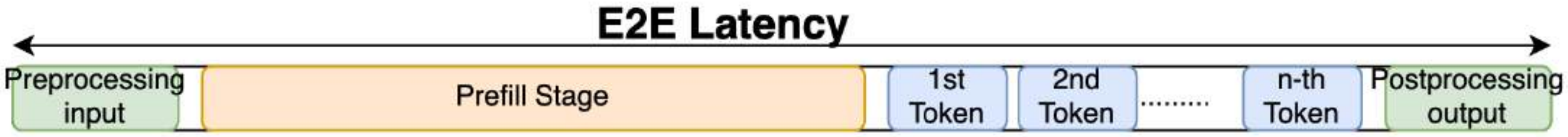
Inference metrics, optimization space

What gets measured gets improved

Performance metrics

- **Prefill latency**
  - time-to-first-token latency (TTFT)
  - Critical waiting time under real-time LLM response to users (e.g., chatbot)
  - Affecting factors: model size, large input length, hardware
- **Decoding throughput (tokens / second)**
  - Inter-token latency (ITL) after the first token
  - Important under real time interaction (e.g. chatbot)
    - Faster is not always desirable. 6 English words/sec may be sufficient for chatbot
  - Affecting factors: batchsize, hardware

Performance metrics

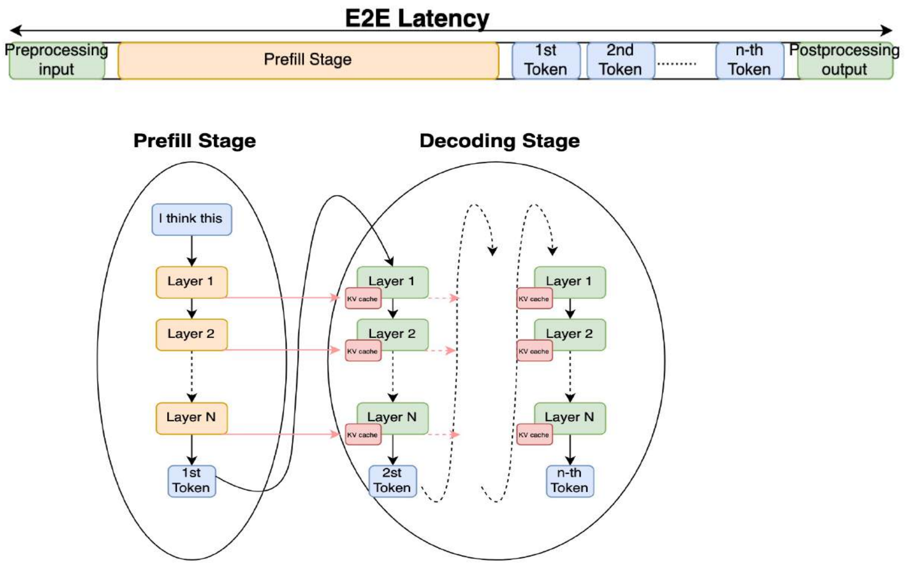


- **End-to-end latency**
  - Latency to complete inference process
  - crucial for assessing the overall user experience
    - often dependent on other components, e.g., processing JSON
  - Affecting factors: network latency, total tokens to generate, overall system, etc
- **Maximum request rate, a.k.a. QPS, (queries/second)**
  - measures how many concurrent queries can be handled per second
  - Important for serving models in production environments under multiple users
    - Example: Assume 20 QPS can be served (for P90 latency) via single GPU, then 300/20=15 GPUs is required to support QPS 300
  - Affecting factors: hardware resources, load balancing, etc.

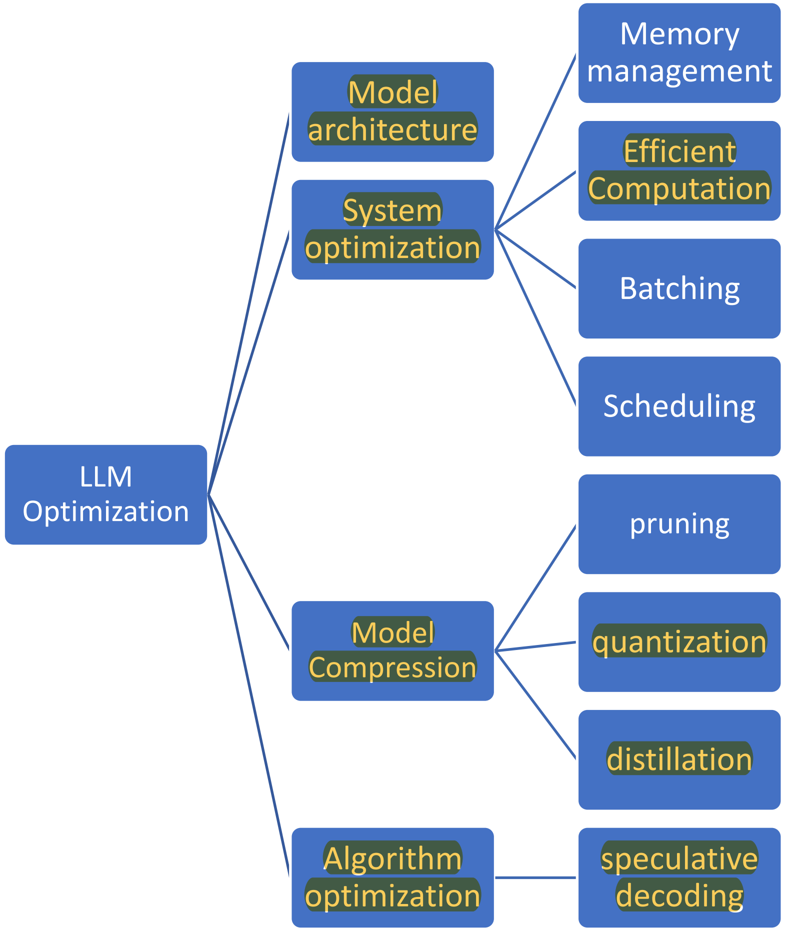
The space of inference optimizations

Metric:

Inference Stages:



- With these metrics, optimize prefill, decoding and other stages of LLM via
  - Model architecture choice
  - System optimizations
  - Model compressions
  - Algorithmic optimization



amazon | science

Primer on model architecture choice

Group Query Attention, Mixture of Experts

Architecture Choice for Fast Inference

- As scaling up model parameters to 70+B, encounter memory bound and compute bound.
  - For half precision Llama2 70B, multiple GPUs just to load 140G weights.
  - Quadratic computational $O(L^2d + Ld^2)$ during prefill
  - KV cache memory can consume large memory for long seq or large batch

Architecture Choice for Fast Inference

- As scaling up model parameters to 70+B, encounter memory bound and compute bound.
  - For half precision Llama2 70B, multiple GPUs just to load 140G weights.
  - Quadratic computational $O(L^2d + Ld^2)$ during prefill
  - KV cache memory can consume large memory for large input token

- Motivate to have more efficient Transformation architectures
  - Can we reduce K, V, Q computations and memory? -> Group Query Attention
  - Can we reduce projection computations? -> Mixture of Experts

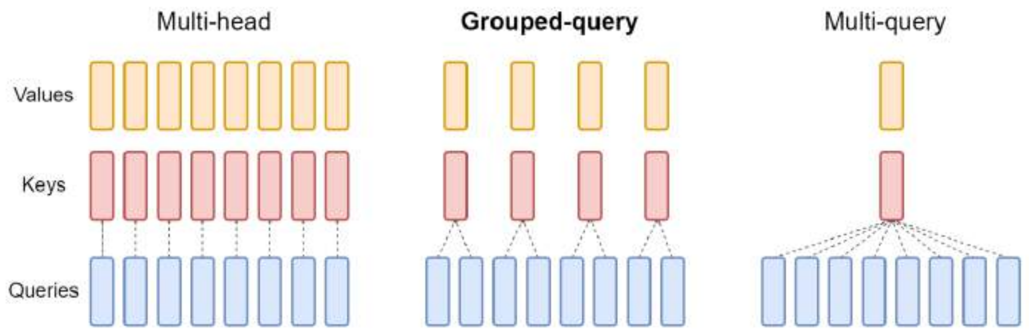
Architecture Choice for Fast Inference

- As scaling up model parameters to 70+B, encounter memory bound and compute bound.
  - For half precision Llama2 70B, multiple GPUs just to load 140G weights.
  - Quadratic computational $O(L^2d + Ld^2)$ during prefill
  - KV cache memory can consume large memory for long seq or large batch

- Motivate to have more efficient Transformation architectures
  - Can we reduce K, V, Q computations and memory? -> Group Query Attention
  - Can we reduce projection computations? -> Mixture of Experts

Group Query Attention

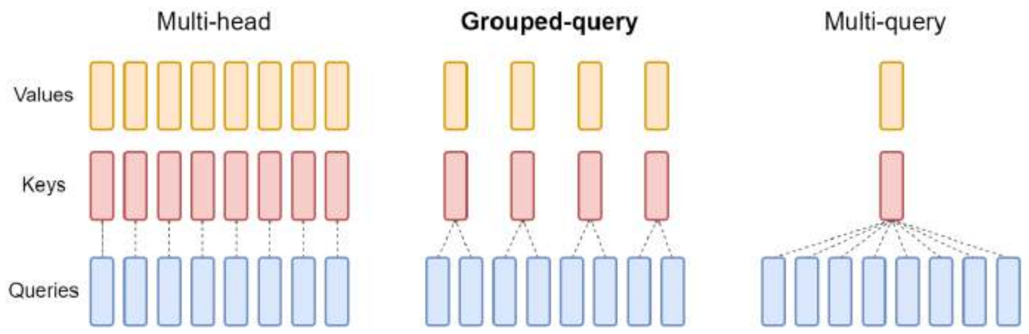
- Multi-headed attention (MHA): distinct query (Q), key (K), and value (V) for each head
  - K and V heads increase linearly to Q heads
- Multi-query attention (MQA): single K and V head shared across all query heads
  - Leverages repeated K and V, reduces inference latency
  - but sacrifices prediction quality
- Grouped-query attention (GQA): single K and V head shared within each query group
  - Balances between MHA and MQA, reducing memory/latency but preserving accuracy



[Ainselie et al. 2023] Overview of MHA (left), MQA (middle), and GQA (right)

Group Query Attention

- Under distributed setting, evenly distribute KV heads to devices
  - KV heads/caches can be reused for multiple query heads (within each devices)
  - Llama2 70B has 64 Q heads, which are grouped with 8 KV heads,
  - i.e., one KV head per core in case of 8xA100

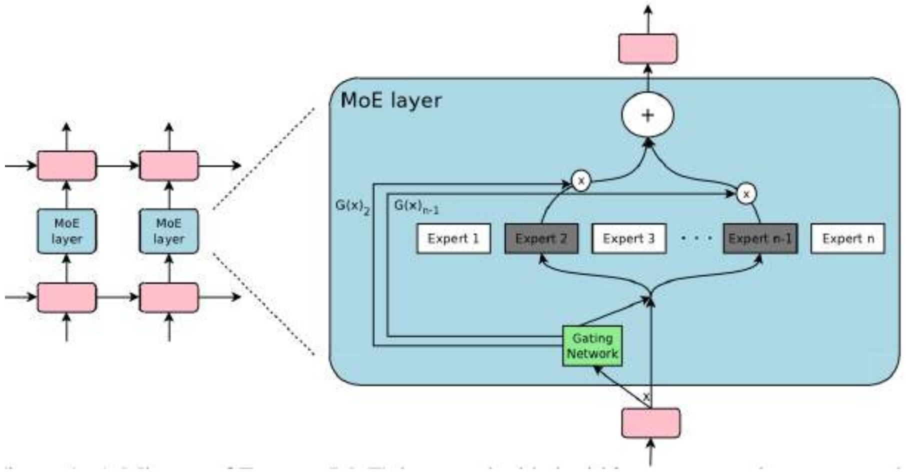


[Ainselie et al. 2023] Overview of MHA (left), MQA (middle), and GQA (right)

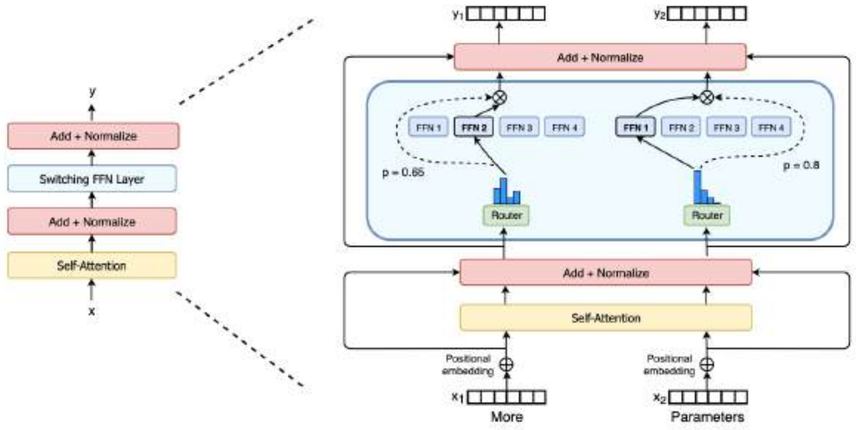
- GQA requires less memory of KV cache than MHA due to smaller # KV heads
  - E.g., KV cache of LLaMa2 70B (with GQA) is smaller than one of LLaMa2 7B.

Mixture of Experts for Transformer

YanAITalk Youtube Channel: Mixtral 8x7B



[Shazeer et al . 2017] MoE to LSTM

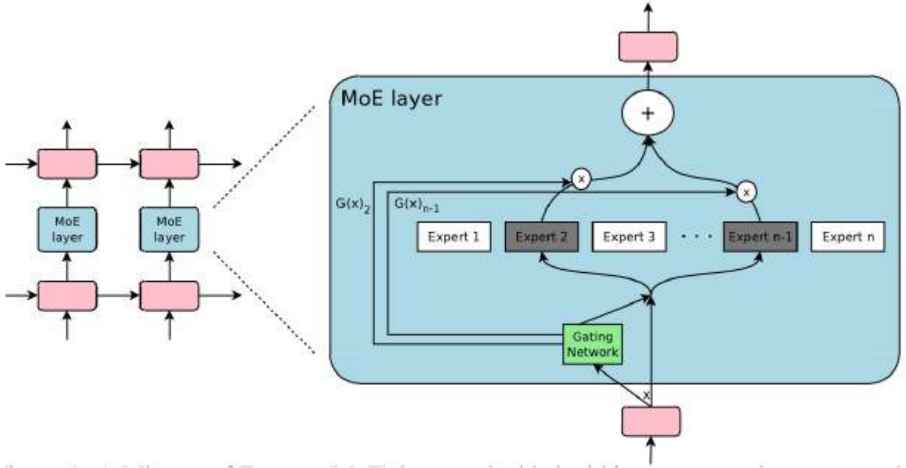


[Fedus et al . 2022] MoE (a sparse Switch FFN layer) to Transformer layer

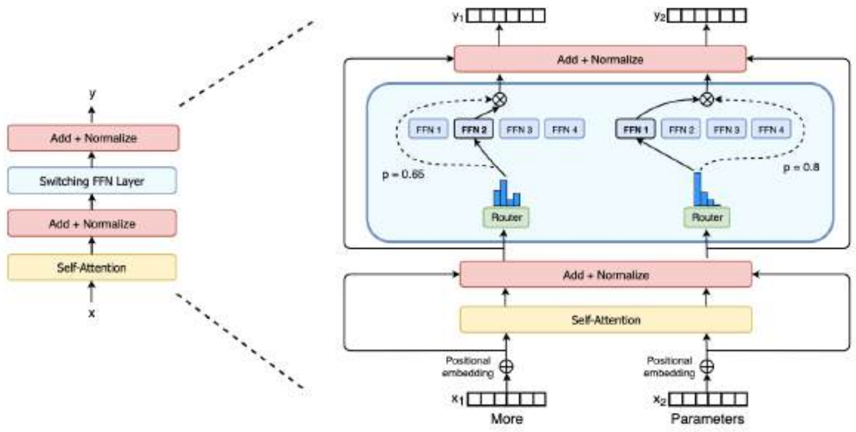
- Mixture of Experts (MoE) is one way to scale up without increasing inference latency
  - Model size of hundreds of billion parameters
  - E.g., Mixtral 8 x 7B [Jiang et al, 2023] uses 8 experts
- Replaces dense Feed Forward Network (FFN) layer with sparse MoE FFN.
  - Few e experts are selected and activated to a given input (token or segment)
  - Effective (activated) number of parameters is the same as experts increase

amazon | science

Mixture of Experts for Transformer



[Shazeer et al . 2017] MoE to LSTM

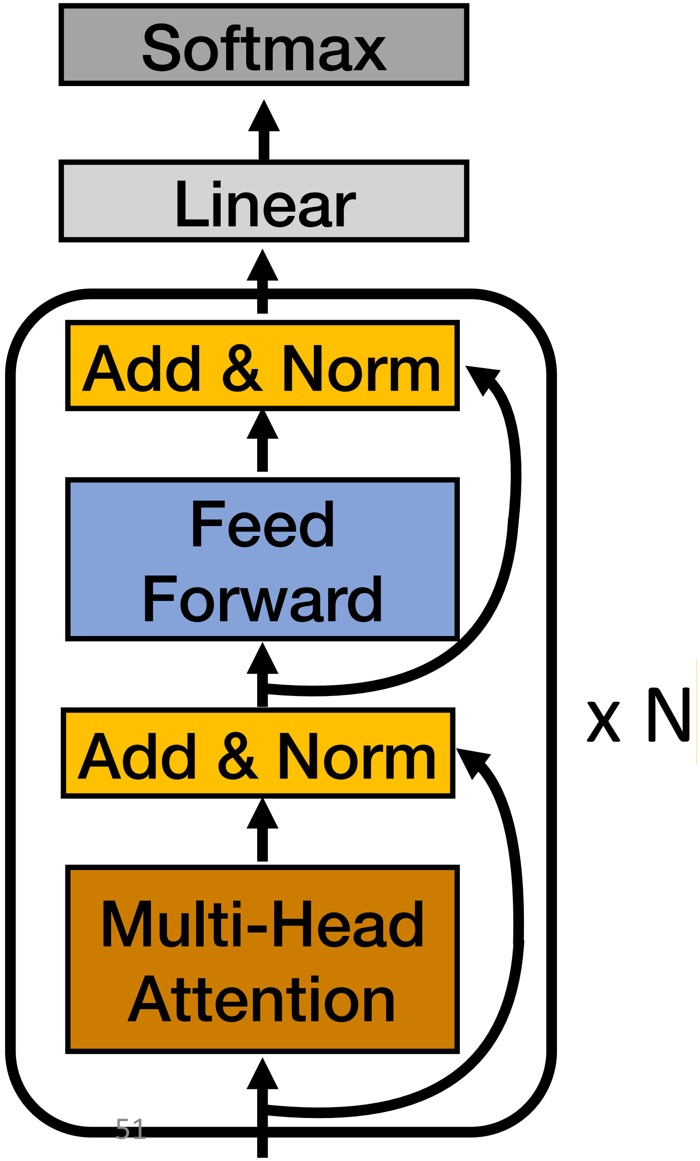


[Fedus et al . 2022] MoE (a sparse Switch FFN layer) to Transformer layer

- Routing mechanism (or gate network)
  - Assigns few experts to a given input based attention weights, expert domain, etc
  - Critical in increasing inference throughput via load balancing over experts and expert parallelism over devices
- Expert Parallelism (EP) is effective under distributed setting
  - keeping each expert within a small group of devices
  - alleviating collective communications and dynamic loading, leading to fast inference
  - E.g., Mitral 8 x 7B uses 8 experts with choosing 2 of experts group for each token

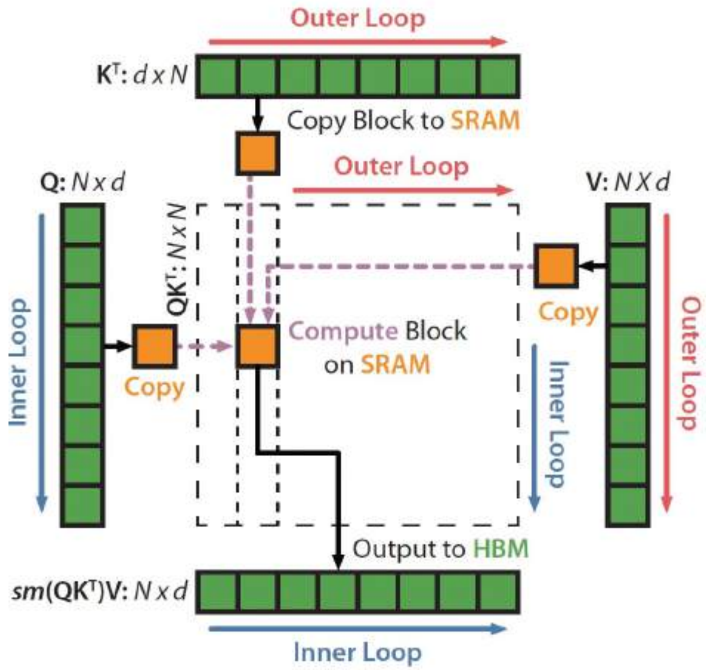
amazon | science

Optimize the forward computation



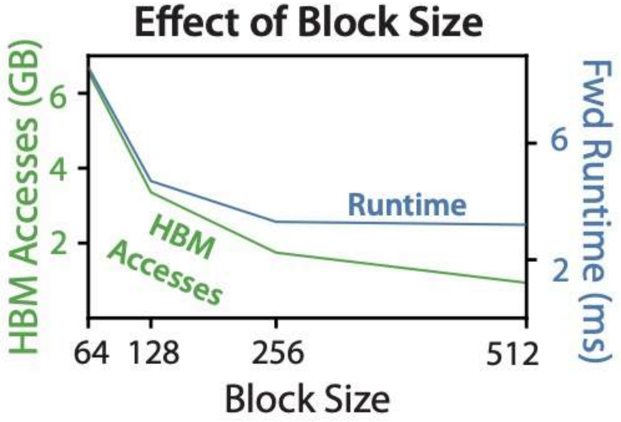
- Goal: Finish one forward computation as fast as possible
  - FlashAttention for prefill
  - FlashDecoding for decoding
  - Distributed inference

FlashAttention



- Standard self-attention: $softmax(\frac{QK^T}{\sqrt(d_k)})V$
  - $S = QK^T$, read QK ($\mathbb{R}^{Nd}$), write S ($\mathbb{R}^{N^2}$)
  - $P = softmax(S)$, read S ($\mathbb{R}^{N^2}$), write P ($\mathbb{R}^{N^2}$)
  - $O = PV$, read PV ($\mathbb{R}^{N^2}, \mathbb{R}^{Nd}$), write O ($\mathbb{R}^{Nd}$)

- FlashAttention: attention kernel with reduced ***IO*** (i.e., data transfer between HBM and SRAM on accelerators)
  - Decompose (tiling) softmax by scaling
  - Fuse the S, P, O computation tile-by-tile
  - Tile size is dependent to the SRAM size
    - GPU: 40 MB L2 shared, Trainium: 24 MB per core



Dao, Tri, et al. "FlashAttention: Fast and memory-efficient exact attention with IO-Awareness." *Advances in Neural Information Processing Systems* 35 (2022)

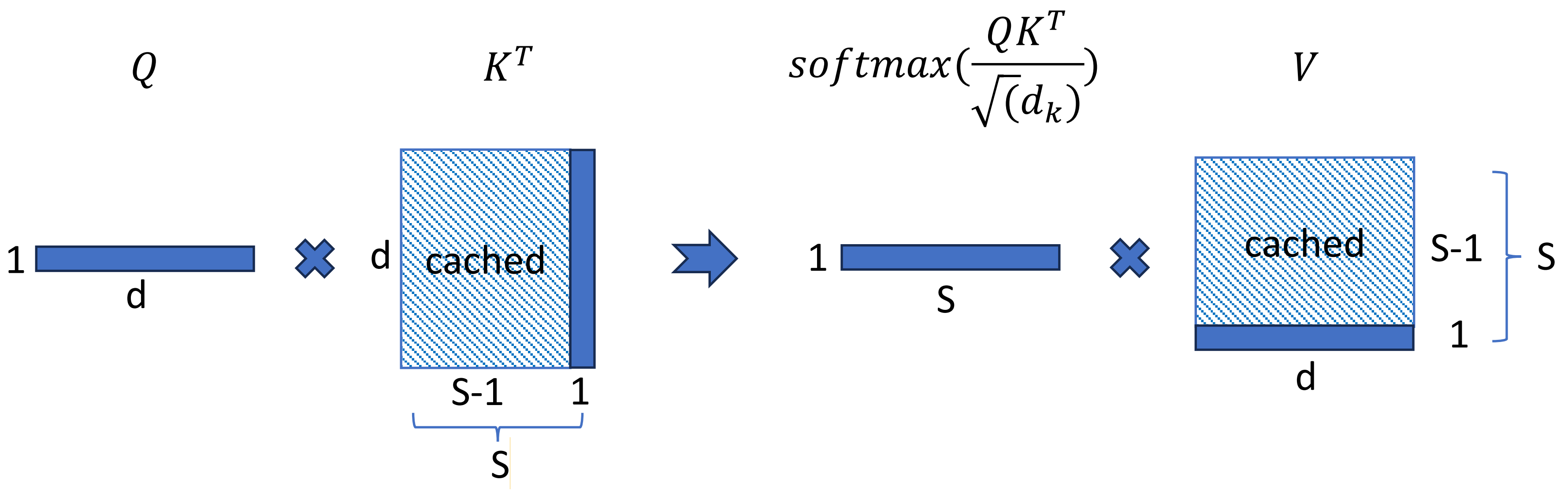
FlashAttention (conti.)

- Calculate P_i of softmax ($P_i = \frac{e^{S_i}}{\sum_j e^{S_j}}$) requires a complete matrix of S
- FlashAttention decomposes softmax to remove this constraint, so that P can be computed tile-by-tile

Dao, Tri, et al. "FlashAttention: Fast and memory-efficient exact attention with IO-Awareness." *Advances in Neural Information Processing Systems* 35 (2022)

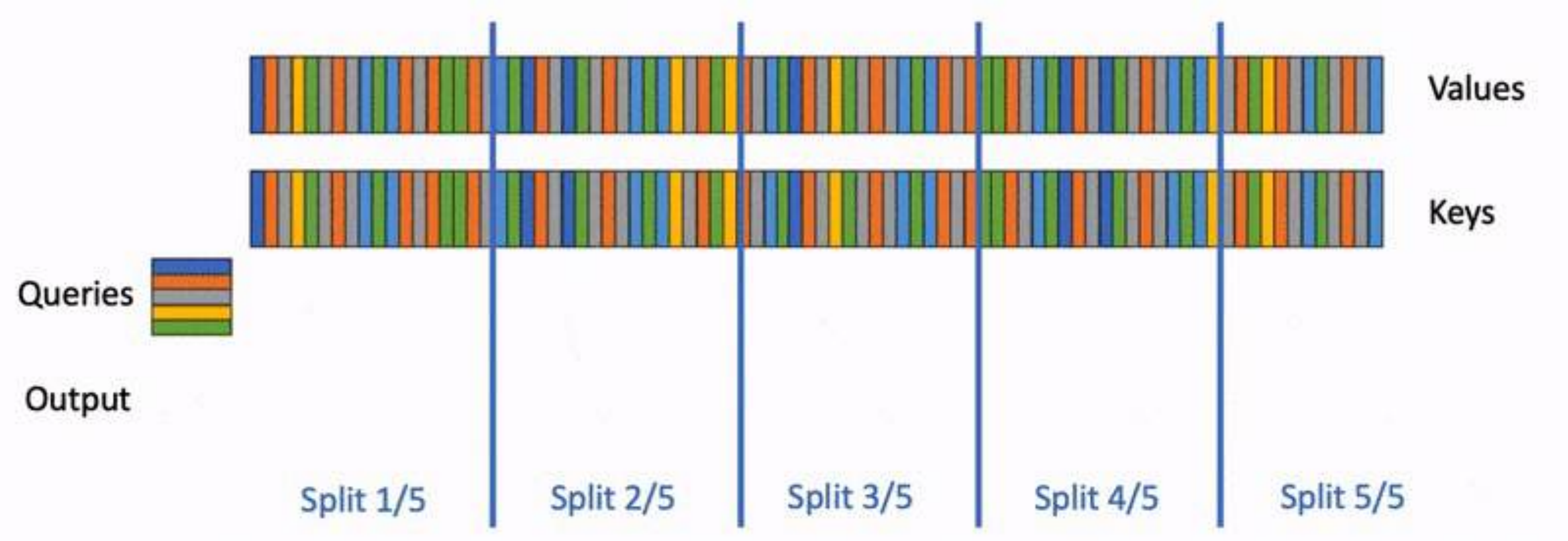
Decoding with KV-cache

- Standard self-attention: $softmax(\frac{QK^T}{\sqrt{(d_k)}})V$
  - Q becomes a vector
  - QK^T and $softmax(\frac{QK^T}{\sqrt{(d_k)}})V$ becomes matrix-vector multiplication (GEMV)



FlashDecoding

- One more parallelization degree: Sequence length



Dao, Tri, et al. https://crfm.stanford.edu/2023/10/12/flashdecoding.html

Distributed inference

- Tensor Parallelism (TP) on top of multi-head
  - Group query attention, #kv heads=8
  - What if #devices per node > 8?
    - Duplication
    - Partition
- Pipeline Parallelism (PP) across multiple nodes
  - Enable inference on very large models
  - Larger latency due to lower bandwidth between nodes
- Prefill-Decode Disaggregation
  - Run Prefill and decode on **separate** machines with **different** parallelization and hardware configurations

Primer on model compression

1. Motivation
2. Quantization
3. Distillation
4. Pruning

Motivation

Current Trend: Larger models tend to do better on benchmarks.

Category	Benchmark	Llama 3 8B	Gemma 2 9B	Mistral 7B	Llama 3 70B	Mixtral 8x22B	GPT 3.5 Turbo	Llama 3 405B	Nemotron 4 340B	GPT-4 (0125)	GPT-4o	Claude 3.5 Sonnet
General	MMLU (5-shot)	69.4	**72.3**	61.1	**83.6**	76.9	70.7	87.3	82.6	85.1	89.1	**89.9**
	MMLU (0-shot, CoT)	**73.0**	72.3$^{\triangle}$	60.5	**86.0**	79.9	69.8	88.6	78.7$^{\triangleleft}$	85.4	**88.7**	88.3
	MMLU-Pro (5-shot, CoT)	**48.3**	–	36.9	**66.4**	56.3	49.2	73.3	62.7	64.8	74.0	**77.0**
	IFEval	**80.4**	73.6	57.6	**87.5**	72.7	69.9	**88.6**	85.1	84.3	85.6	88.0
Code	HumanEval (0-shot)	**72.6**	54.3	40.2	**80.5**	75.6	68.0	89.0	73.2	86.6	90.2	**92.0**
	MBPP EvalPlus (0-shot)	**72.8**	71.7	49.5	**86.0**	78.6	82.0	88.6	72.8	83.6	87.8	**90.5**
Math	GSM8K (8-shot, CoT)	**84.5**	76.7	53.2	**95.1**	88.2	81.6	**96.8**	92.3$^{\diamond}$	94.2	96.1	96.4$^{\diamond}$
	MATH (0-shot, CoT)	**51.9**	44.3	13.0	**68.0**	54.1	43.1	73.8	41.1	64.5	**76.6**	71.1
Reasoning	ARC Challenge (0-shot)	83.4	**87.6**	74.2	**94.8**	88.7	83.7	**96.9**	94.6	96.4	96.7	96.7
	GPQA (0-shot, CoT)	32.8	–	28.8	**46.7**	33.3	30.8	51.1	–	41.4	53.6	**59.4**
Tool use	BFCL	**76.1**	–	60.4	84.8	–	**85.9**	88.5	86.5	88.3	80.5	**90.2**
	Nexus	**38.5**	30.0	24.7	**56.7**	48.5	37.2	**58.7**	–	50.3	56.1	45.7
Long context	ZeroSCROLLS/QuALITY	81.0	–	–	90.5	–	–	**95.2**	–	**95.2**	90.5	90.5
	InfiniteBench/En.MC	65.1	–	–	78.2	–	–	**83.4**	–	72.1	82.5	–
	NIH/Multi-needle	98.8	–	–	97.5	–	–	98.1	–	**100.0**	**100.0**	90.8
Multilingual	MGSM (0-shot, CoT)	**68.9**	53.2	29.9	**86.9**	71.1	51.4	**91.6**	–	85.9	90.5	**91.6**

Image Source: The Llama 3 Herd of Models, https://llama.meta.com/

Motivation

Challenge: Inference is *expensive* and *slow* with *large* models

- **Need multiple AI accelerator chips**

  Llama-3.1 405B

  - ≈ 810 GB in native precision (BF16)
  - 512 GB off-chip shared memory (16 chips) in an Amazon EC2 trn1n.32xlarge instance
  - Need at least 2 trn1n.32xlarge to load *weights only* in BF16
    USD 434, 505 in annual on-demand cost (at USD 24.78 / hr)

AWS EC2 Trainium Source: https://aws.amazon.com/ec2/instance-types/trn1/ (date: 8/18/2024)

Key Idea

- Compress a large model into a smaller model
  - Use low-precision data types
  - Use fewer parameters
- Trade-off between accuracy and inference improvement

Benefits

- Reduces the number of flops
  - Applies to *some* model compression approaches
    (e.g., pruning but not quantization)
  - Speeds up prefill
    Removing 50% of the parameters of the model reduces the flops by a half

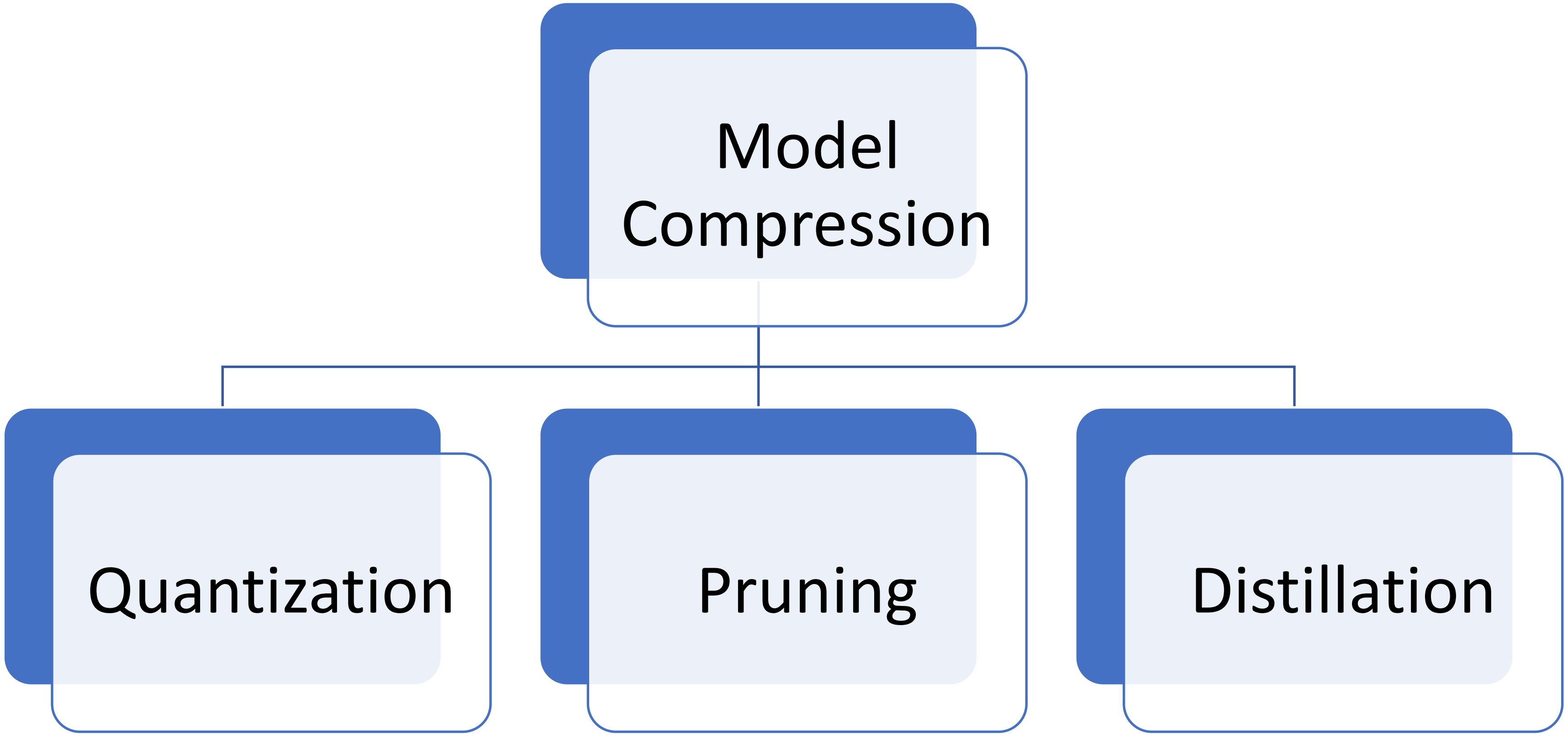
- Reduces the memory footprint of LLMs
  - Applies to *all* model compression methods
  - Number of AI accelerators to serve the model goes down
  - Speeds up decoding

Model compression measurement

Compression Ratio (r) = $\frac{Uncompressed\ Model\ Size}{Compressed\ Model\ Size}$

r > 1 ⇒ effective compression

Model compression methods



Model quantization

Quantization

- Computes and stores *tensors* at lower bit-widths

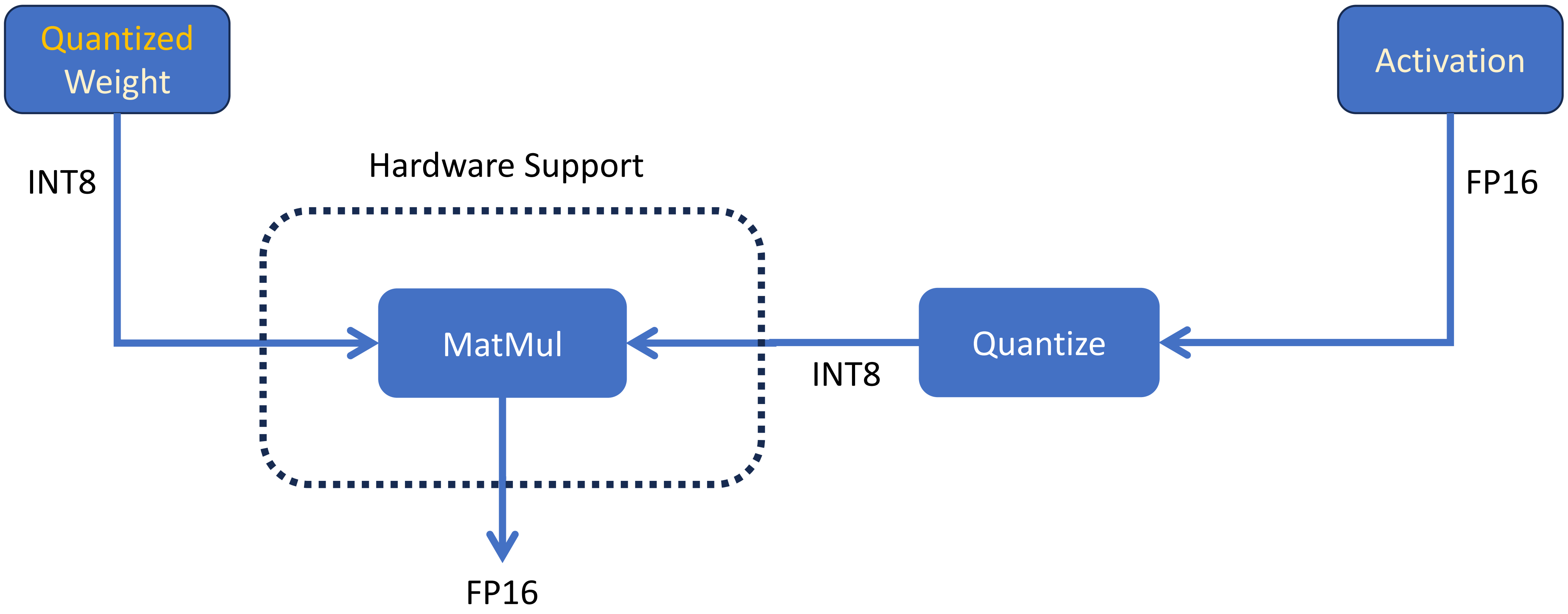
16 bit (e.g., BF16) → 8 bit (e.g., INT8) ⇒ $r = 2$

16 bit (e.g., BF16) → 4 bit (e.g., INT4) ⇒ $r = 4$

- Supported by popular AI accelerators

INT8	FP8
Amazon EC2 trn1/inf2 Neuron Chips NVIDIA Tensor Cores	Amazon EC2 trn2/inf3 Neuron Chips (upcoming!) NVIDIA Hopper Tensor Cores

Example Operation



Typically quantized components in LLMs

- Weights of linear layers in attention and feedforward layers
- Activations
- KV Cache

How to quantize different components?

Dynamic quantization

- Quantize weights offline
- Learn quantization parameters on-the-fly
- Read / write activations from / to memory in higher bit-width (e.g., FP32)
- Quantize activations on-the-fly
- Perform tensor operations in lower bit-width (e.g., INT8)

Ref: https://pytorch.org/docs/stable/quantization.html

How to quantize different components?

Static quantization

- Quantize weights offline
- Learn quantization parameters offline with calibration data
- Read / write activations from / to memory in lower bit-width (e.g., INT8)
- Quantize activations on-the-fly
- Perform tensor operations in lower bit-width (e.g., INT8)

Ref: https://pytorch.org/docs/stable/quantization.html

When to quantize?

- Post-training quantization
- Quantization-aware training

1.5-bit LLM deep dive

BitNet Architecture: Transformer with nn.Linear replaced by BitLinear

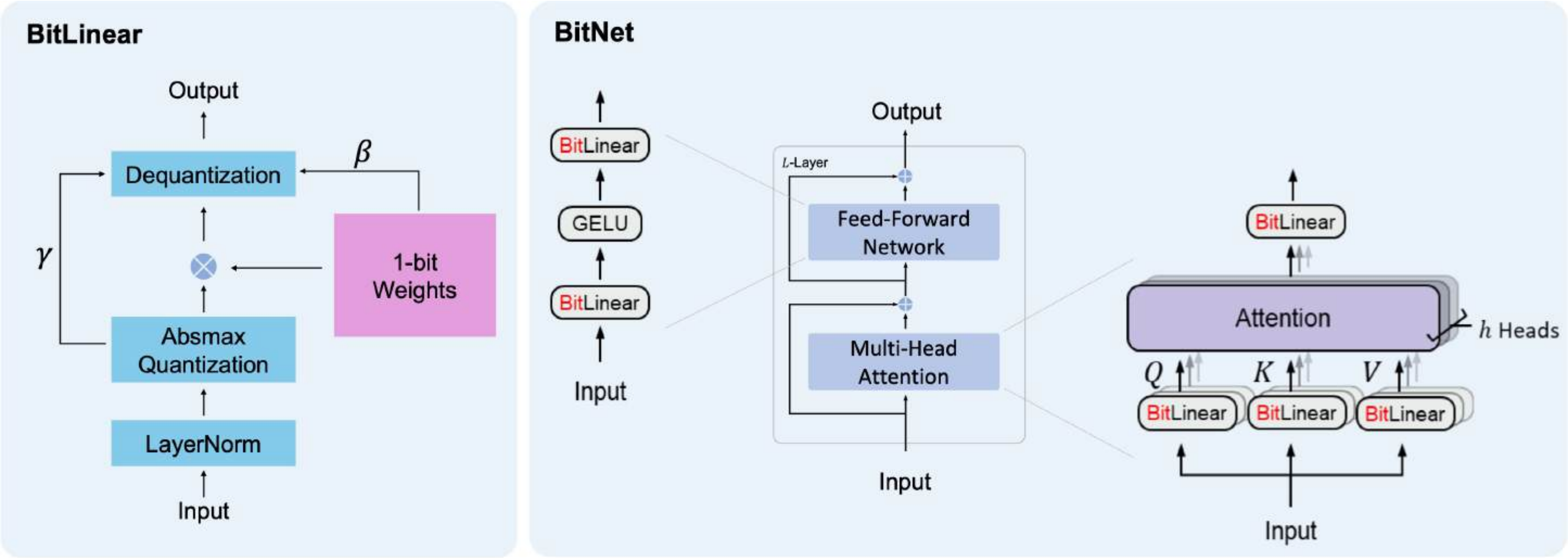
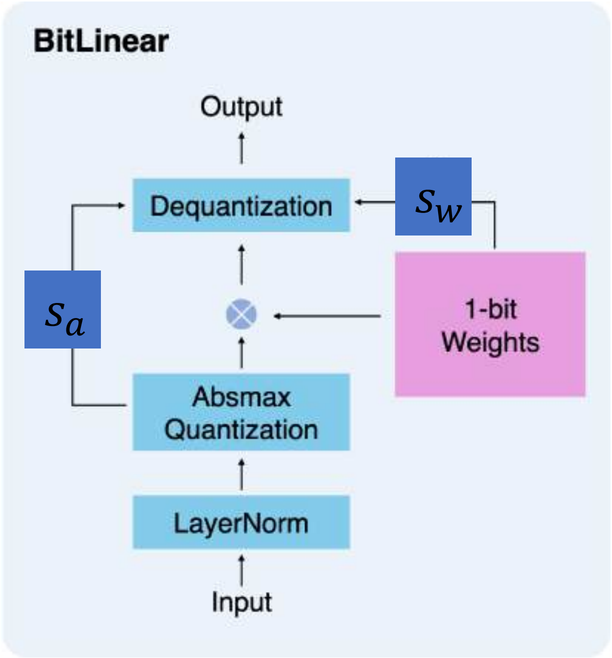


Image Source: Ma & Wang et al., The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits, arXiv 2024

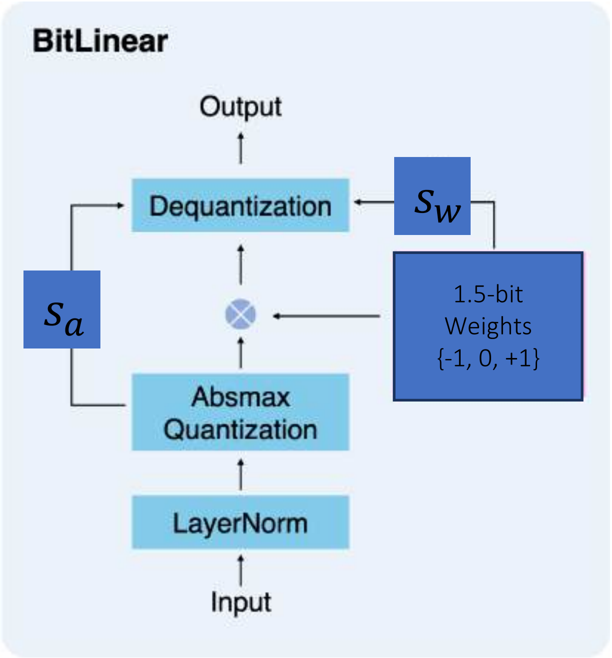
amazon | science

1.5-bit LLM deep dive – architecture

1-bit BitLinear



1.5-bit BitLinear



- 1.5-bit LLM replaces 1-bit BitLinear with 1.5-bit BitLinear layer
- s_a and s_w are scaling parameters for activation and weights resp.

Image Source: Ma & Wang et al., The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits, arXiv 2024

1.5-bit LLM deep dive – accuracy results

- BitNet b1.58 can match the performance of the full precision baseline starting from a 3B size.

Models	Size	ARCe	ARCc	HS	BQ	OQ	PQ	WGe	Avg.
LLaMA LLM	700M	54.7	23.0	37.0	60.0	20.2	68.9	54.8	45.5
BitNet b1.58	700M	51.8	21.4	35.1	58.2	20.0	68.1	55.2	44.3
LLaMA LLM	1.3B	56.9	23.5	38.5	59.1	21.6	70.0	53.9	46.2
BitNet b1.58	1.3B	54.9	24.2	37.7	56.7	19.6	68.8	55.8	45.4
LLaMA LLM	3B	62.1	25.6	43.3	61.8	24.6	72.1	58.2	49.7
BitNet b1.58	3B	**61.4**	**28.3**	**42.9**	**61.5**	**26.6**	**71.5**	**59.3**	**50.2**
BitNet b1.58	3.9B	**64.2**	**28.7**	**44.2**	**63.5**	**24.2**	**73.2**	**60.5**	**51.2**

Image Source: Ma & Wang et al., The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits, arXiv 2024

Model distillation

Model distillation history

Ensemble of models:

Performance(Ensemble of models) > Performance(single model)
Inference Cost(Ensemble of models) > Inference Cost(single model)

Model distillation:
Compress ensemble of models / Large model → smaller model
(Bucila et al., KDD 2006, Hinton et al., NeurIPS 2014)

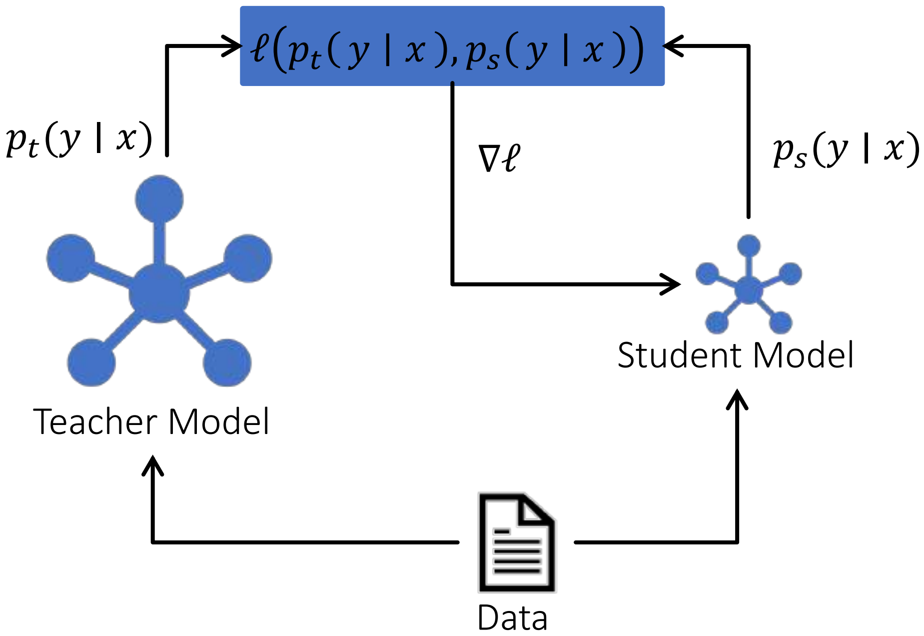
Similar scenario with LLMs with billions of parameters

Bucila et al., Model Compression, KDD 2006
Hinton et al., Distilling the Knowledge in a Neural Network, NeurIPS 2014

Key idea

Analogy from formal education:
Larger model (**teacher**) teaches the smaller model (**student**)

Train the student to match the teacher's perf. via a **distillation loss**



Benefits

- Train a smaller model (of same/different arch.) with fewer parameters
- Smaller memory footprint
  - Number of AI accelerators to host the model goes down
- Speeds up prefill
  - fewer flops during prefill
- Speeds up decoding
  - reduces the memory I/O for loading parameters and intermediate states

How to match a student to a teacher?

- Output logits (Hinton et al, NeurIPS DL Workshop 2014)
  - Compute probabilities $\left[p^{(1)}, \ldots, p^{(v)}\right]$ from logits $[z^{(1)}, \ldots, z^{(v)}]$ using softmax (for teacher & student) with v being the vocabulary size
  - Compute the cross-entropy loss
- Intermediate Weights (Romero et al., ICLR 2015)
  - Add an L_2 loss between teacher and student weights in addition to cross entropy loss
  - Apply linear transformation to match dimensionalities

Hinton et al., Distilling the Knowledge in a Neural Network, NeurIPS 2014
Romero et al., FitNets: Hints for Thin Deep Nets, ICLR 2015

How to match a student to a teacher?

- Intermediate Features (Huang and Wang, arXiv 2017)
  - Minimize the distance between intermediate activations of teacher/student
  - Maximum Mean Discrepancy, L_2 distance

Huang and Wang, Like What You Like: Knowledge Distill via Neuron Selectivity Transfer, arXiv 2017

How to match student to teacher?

- Intermediate Gradients (Zagoruyko & Komodakis, ICLR 2017)
  - L_2 distance between gradients $\nabla(Y)$ of intermediate activations Y_t and Y_s of teacher and student respectively
  - Compute probabilities $\left[p^{(1)}, \dots, p^{(v)}\right]$ from logits $[z^{(1)}, \dots, z^{(v)}]$ using Softmax (for teacher & student) with v being the vocabulary size
  - Compute the cross-entropy loss

- Sparsity Pattern, Relation between layers, …

Zagoruyko & Komadakis, Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer, ICLR 2017

Model distillation knowledge transfer set

Typical approach:

Use an external dataset also called transfer set

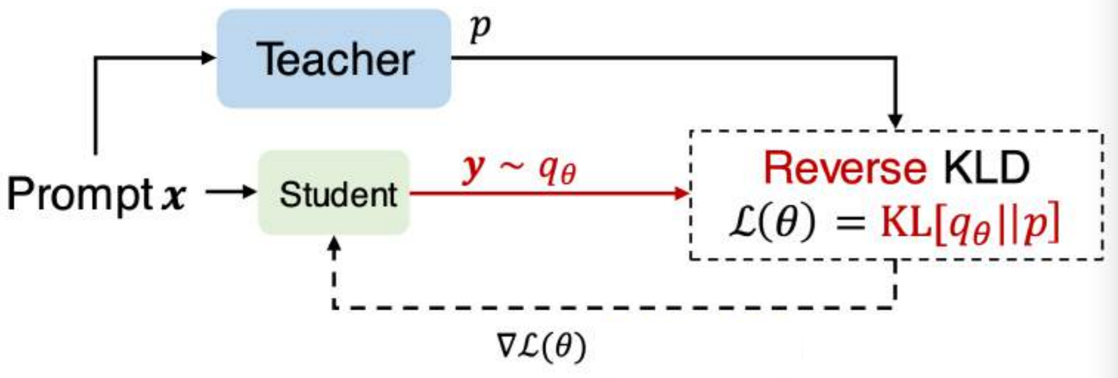
Another approach (Symbolic distillation):
Teacher generates synthetic data (West et al., ACL 2022)

West et al., Symbolic Knowledge Distillation: from General Language Models to Commonsense Models, ACL 2022

amazon | science

MiniLLM deep dive

- Minimizes reverse KL divergence between student distribution q_θ & teacher distribution p



$$\begin{aligned}\theta &= \arg\min_{\theta} \mathcal{L}(\theta) = \arg\min_{\theta} \mathrm{KL}[q_\theta || p] \\ &= \arg\min_{\theta} \left[- \mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}, \boldsymbol{y} \sim q_\theta} \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{q_\theta(\boldsymbol{y}|\boldsymbol{x})} \right]\end{aligned}$$

Image Source: Gu et al., MiniLLM: Knowledge Distillation of Large Language Models, ICLR 2024

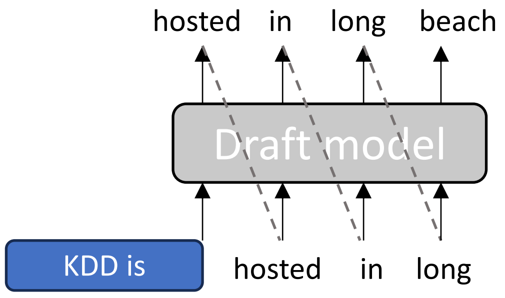
Model distillation takeaways

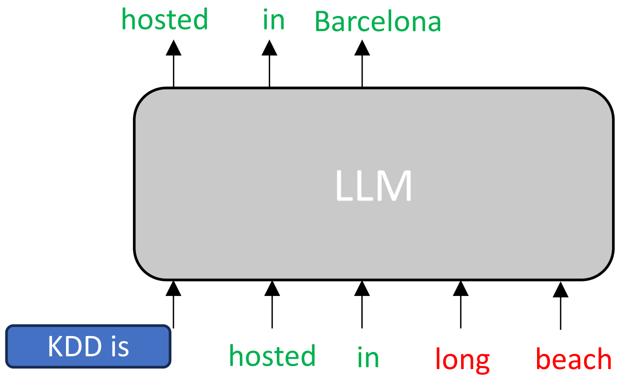
- **Expensive.** Both models must be loaded into AI accelerators
- **Train small or distill?** Unclear how many training steps are needed during distillation compared to training the small model from scratch
- Performance of student hinges on the transfer set
- Difficulties in optimization (Stanton et al., NeurIPS 2021)
  - Fidelity of the student to its teacher vs generalization ability of the student in predicting unseen data

Stanton et al., Does Knowledge Distillation Really Work?, NeurIPS 2021

Speculative Decoding

- LLM Decoding is autoregressive, and memory bounded
- Speculative Decoding (Leviathan et. al, 2023) mitigates the memory bound
  - Draft tokens from a smaller model: $x \sim q(x)$
  - Verify (in parallel) the drafted tokens using the LLM: $p(x)$
  - Example
    - Draft 4 tokens, given the prompt
    - Input the prompt + draft tokens to LLM, reject when the LLM indicates a different generation





amazon | science

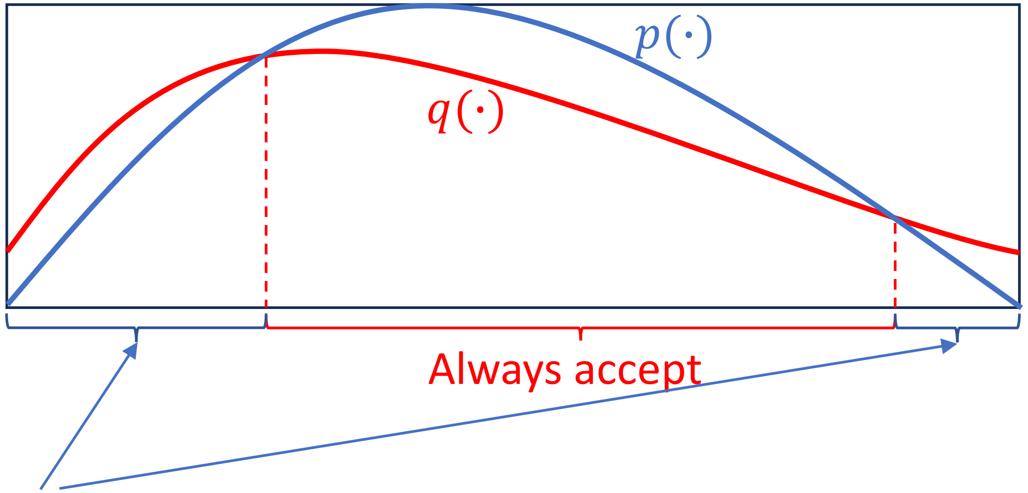
How to verify

- Greedy: Reject if $\arg\max p(x)$ does not equal the drafted
  - e.g., previous example: $\arg\max p(x) =$ "Barcelona" $\neq$ "long"
  - so, reject from "long" and afterwards
- More generally, rejection sampling based
  - draw $x \sim q(x)$
  - Accept x w.p. $\min\left\{\frac{p(x)}{q(x)}, 1\right\}$
    - So always accept when $q(x) < p(x)$
  - Reject x w.p. $1 - \min\left\{\frac{p(x)}{q(x)}, 1\right\}$ and redraw

$$x \sim \text{Normalize}(\max\{p(x) - q(x), 0\})$$

Rejection Sampling based verification

- An illustration

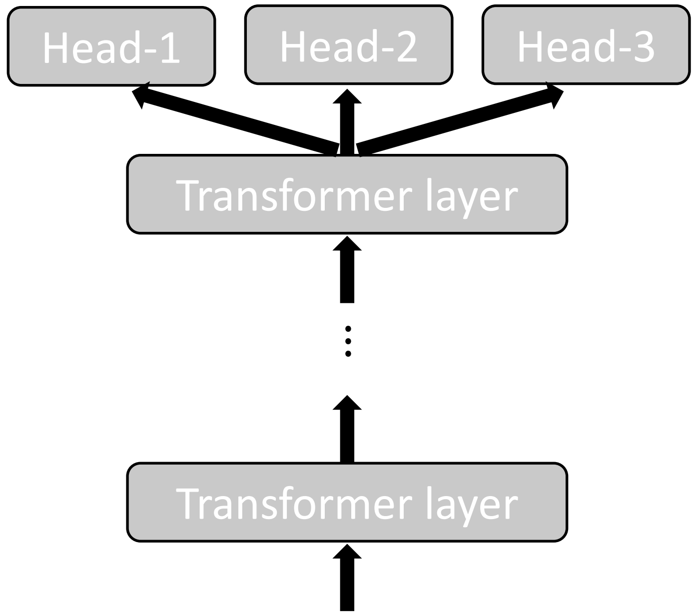


Accept w.p. $\frac{p(x)}{q(x)}$; redraw if rejected

- Guaranteed to be equivalent to sampling from $p(\cdot)$!

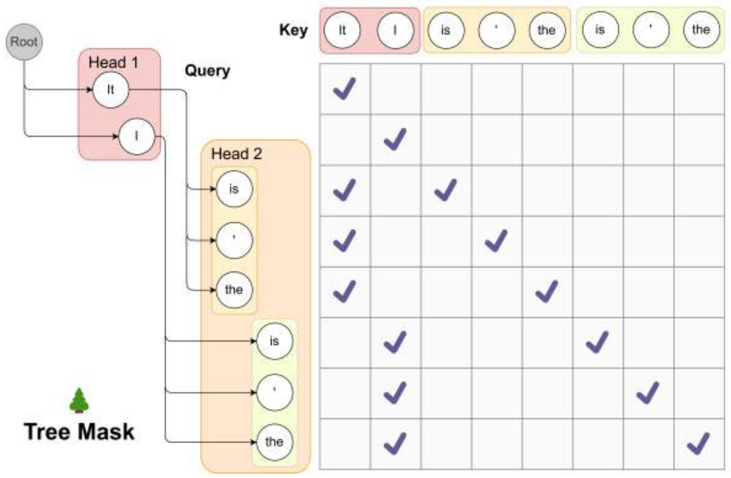
Choice of Draft Model

- Speed up depends on:
  - Closeness between $q(\cdot)$ and $p(\cdot)$
  - Speed ratio between the draft model and the LLM
- Up to 2-3x speedup reported
- How to choose the Draft Model?
  - A smaller model in the same family as the LLM (e.g., Tiny-Llama for Llama2-70B)
  - Share backbone of LLM, but use light-weight head(s) to predict multiple next tokens (e.g., MEDUSA, Cai et. al, 2023, illustrated right)



Improve Acceptance Rate

- Make $q(\cdot)$ and $p(\cdot)$ closer
  - "Align" the draft model to the LLM via Knowledge Distillation
  - Various distillation losses (e.g., DistillSpec, Zhou et. al, 2023)
- Tree-based drafting
  - Multiple draft token per time step
  - Proceed as long as one of them is accepted



Tree-based drafting adopted in Medusa

Image Source: *MEDUSA: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads,* Cai et al. ICML'24

amazon | science

Optimal Draft Length

- Too short draft length may not fully exploit the power of the draft model
- Draft sequence can be longer if
  - $q(\cdot)$ and $p(\cdot)$ are close
  - Draft model is much faster than the LLM
- The closeness between $q(\cdot)$ and $p(\cdot)$ varies,
  - depending on the prefix
  - Adaptive draft length (see, right)

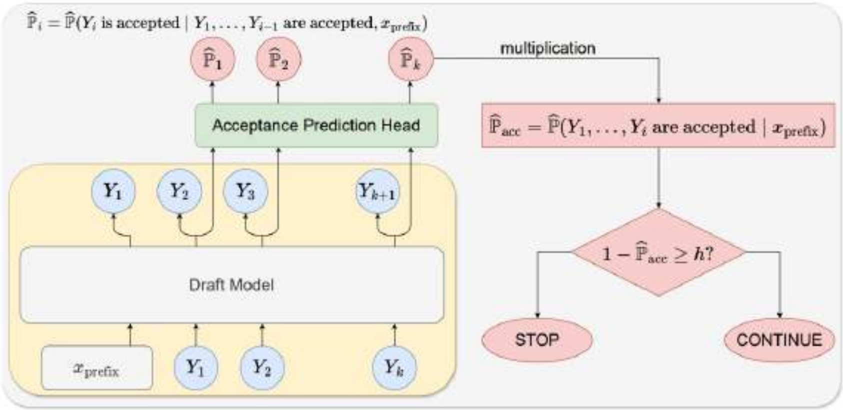


Image Source: *SpecDec++: Boosting Speculative Decoding via Adaptive Candidate Lengths,* Huang et al., 2024

Speculative Decoding Takeaways

- Speculative Decoding is lossless!
- Reduces # of target model forward calls and increases arithmetic int.
- Supported for example on Neuron devices (transformers-neuronx) or GPU (e.g. vllm).
- Various methods to generate drafts -- differ along:
  - Prerequisites: Needs to be trained or not.
  - Overhead: additional memory and latency for draft generation.
  - Quality: Alignment of the drafts with target model.
- Choice of method depends also on the model and application.