An overview of methods for efficient generative AI training & inference

Constantine Dovrolis

The Cyprus Institute (CyI)
and
Georgia Tech

THE CYPRUS INSTITUTE
RESEARCH•TECHNOLOGY•INNOVATION

Georgia Tech

Some LLMs and their training requirements

Model	Parameter Size	Data Scale	GPUs Cost	Training Time
GPT-3 (Brown et al., 2020)	175B	300B tokens	-	-
GPT-NeoX-20B (Black et al., 2022)	20B	825GB corpus	96 A100-40G	-
OPT (Zhang et al., 2022a)	175B	180B tokens	992 A100-80G	-
BLOOM (Scao et al., 2022)	176B	366B tokens	384 A100-80G	105 days
GLM (Zeng et al., 2022)	130B	400B tokens	786 A100-40G	60 days
LLaMA (Touvron et al., 2023a)	65B	1.4T tokens	2048 A100-80G	21 days
LLaMA-2 (Touvron et al., 2023b)	70B	2T tokens	A100-80G	71,680 GPU days
Gopher (Rae et al., 2021)	280B	300B tokens	1024 A100	13.4 days
LaMDA (Thoppilan et al., 2022)	137B	768B tokens	1024 TPU-v3	57.7 days
GLaM (Du et al., 2022)	1200B	280B tokens	1024 TPU-v4	574 hours
PanGu-α (Zeng et al., 2021)	13B	1.1TB corpus	2048 Ascend 910	-
PanGu-$\sum$ (Ren et al., 2023b)	1085B	329B tokens	512 Ascend 910	100 days
PaLM (Chowdhery et al., 2022)	540B	780B tokens	6144 TPU-v4	-
PaLM-2 (Anil et al., 2023)	-	3.6T tokens	TPUv4	-
WeLM (Su et al., 2022b)	10B	300B tokens	128 A100-40G	24 days
Flan-PaLM (Chung et al., 2022)	540B	-	512 TPU-v4	37 hours
AlexaTM (Soltan et al., 2022)	20B	1.3 tokens	128 A100	120 days
Codegeex (Zheng et al., 2023)	13B	850 tokens	1536 Ascend 910	60 days
MPT-7B (Team, 2023)	7B	1T tokens	-	-

Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Learning in LLMs

A Pretraining

Source:A, Radford et al., “Improving Language Understanding by Generative Pre-train

Large corpus
(unlabeled text)

“Would you tell me, please, which way I ought to go from here?”
“That depends a good deal on where you want to get to,” said the Cat.
“I don’t much care where—” said Alice.
“Then it doesn’t matter which way you go,” said the Cat.
“—so long as I get *somewhere*,” Alice added as an explanation.
“Oh, you’re sure to do that,” said the Cat, “if you only walk long enough.”

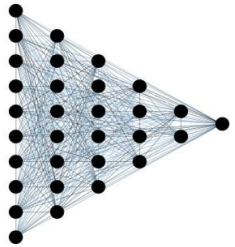
Original text

Masking

“Would you tell me, [redacted], which way I [redacted] to go from here?”
“That [redacted] a [redacted] deal on where you want to get to,” said the Cat.
“I [redacted] much care where—” [redacted] Alice.
“Then it doesn’t matter [redacted] [redacted] you go,” said the Cat.
“—so long as I get *somewhere*,” Alice [redacted] as an explanation.
“Oh, [redacted] [redacted] to do that,” said the Cat, “if [redacted] only [redacted] long enough.”

Masked text

Language model



“Would you tell me, sir, which way I need to go from here?”
“That depends a good deal on where you want to get to,” said the Cat.
“I don’t much care where—” said Alice.
“Then it doesn’t matter which way you go,” said the Cat.
“—so long as I get *somewhere*,” Alice added as an explanation.
“Oh, no need to do that,” said the Cat, “if one only waits long enough.”

Predicted text

Loss

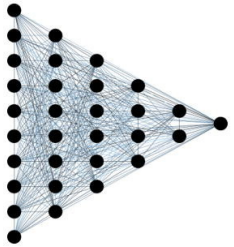
B Fine-tuning

Small labeled
dataset

We wish to suggest a structure for the salt of deoxyribose nucleic acid (D.N.A.). This structure has novel features which are of considerable biological interest.

Text

Fine-tuned model

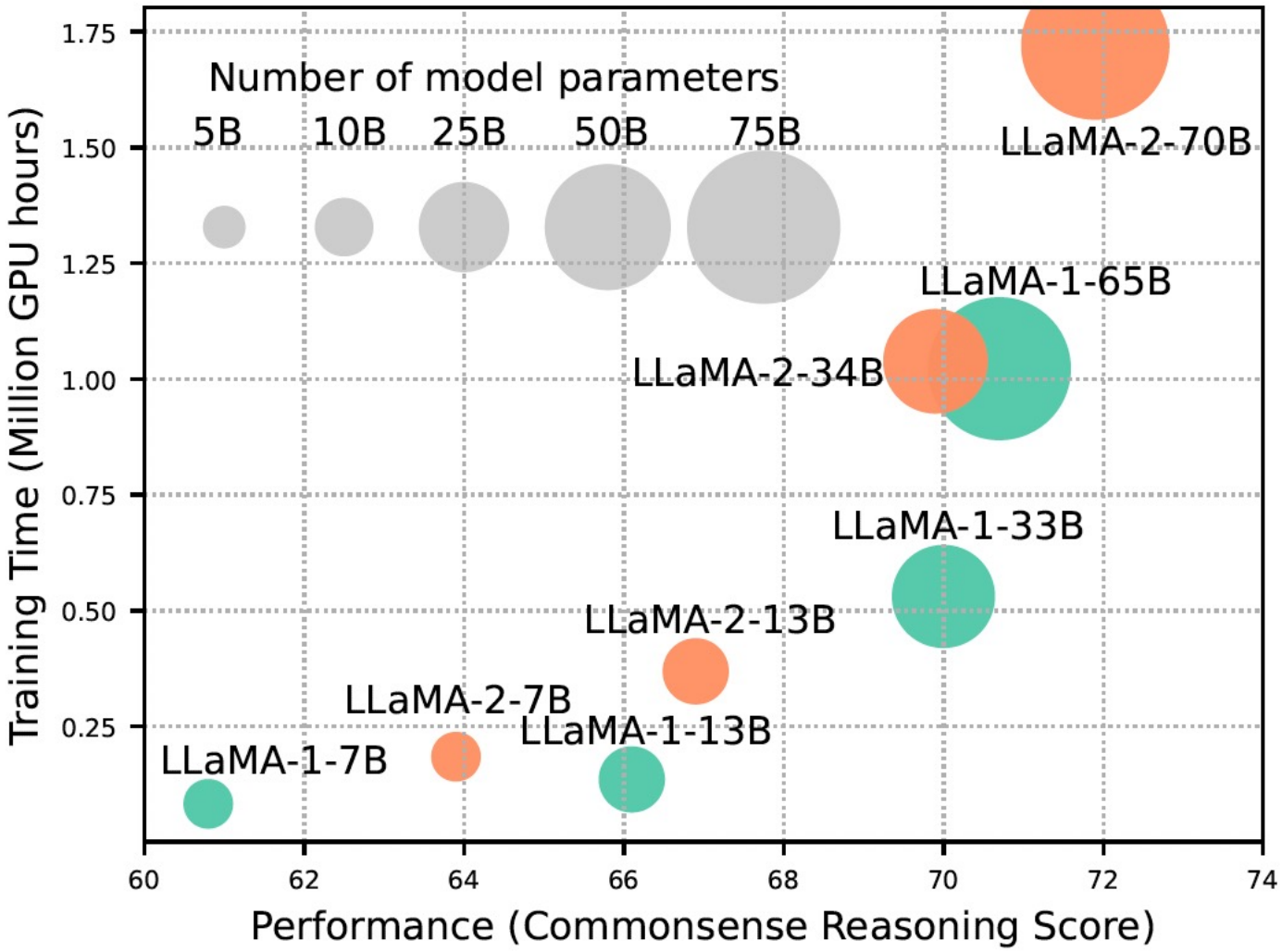


Topic: Biology (97%)

Prediction

Source: D.Ofer et al., “The language of proteins: NLP, machine learning & protein sequences”

Exponential scaling of training time with performance

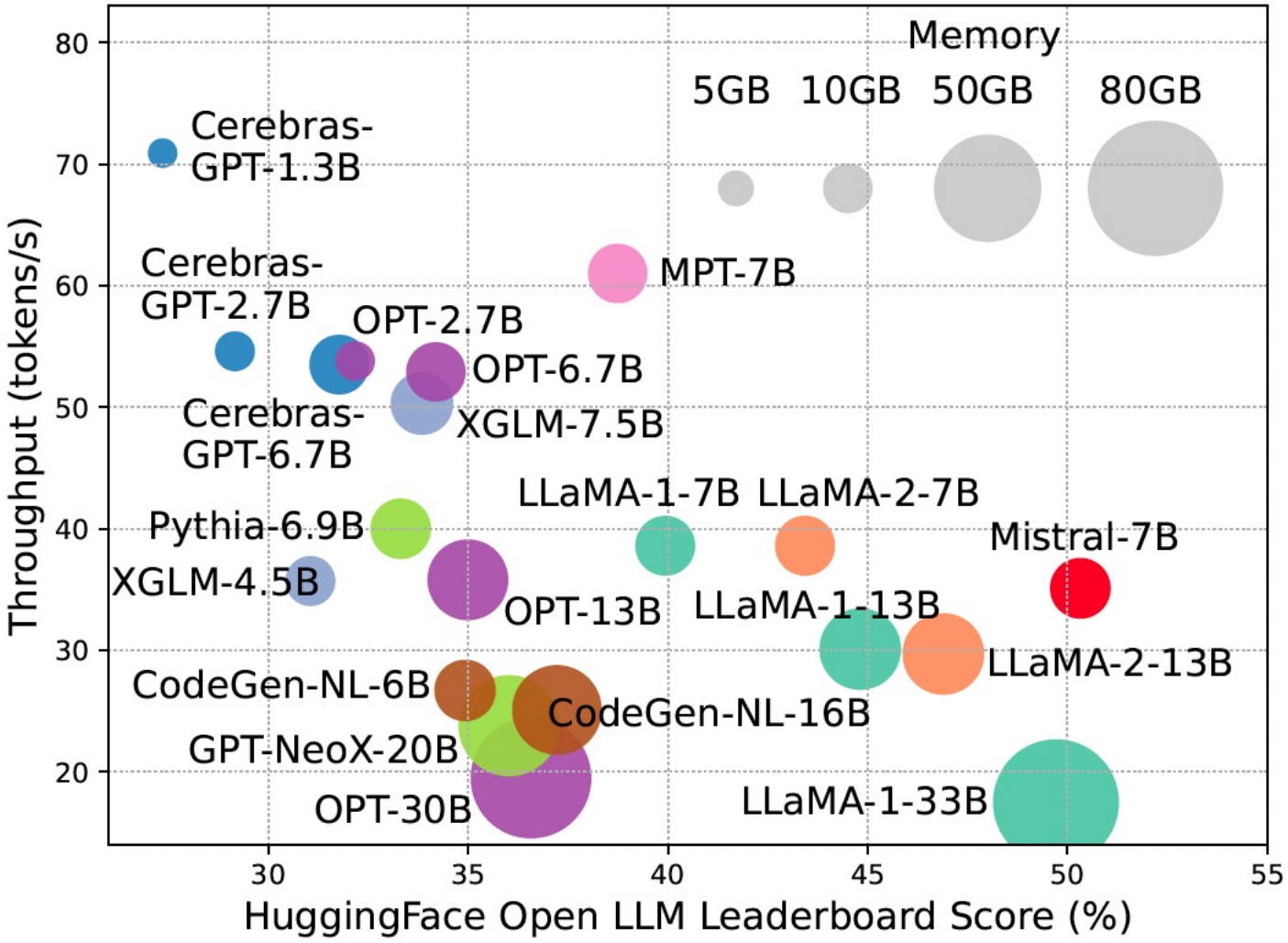


Source: “LLaMA: Open and efficient foundation language models” by Touvron et al. 2023

Inference is also computationally heavy

Note Mistral-7B:

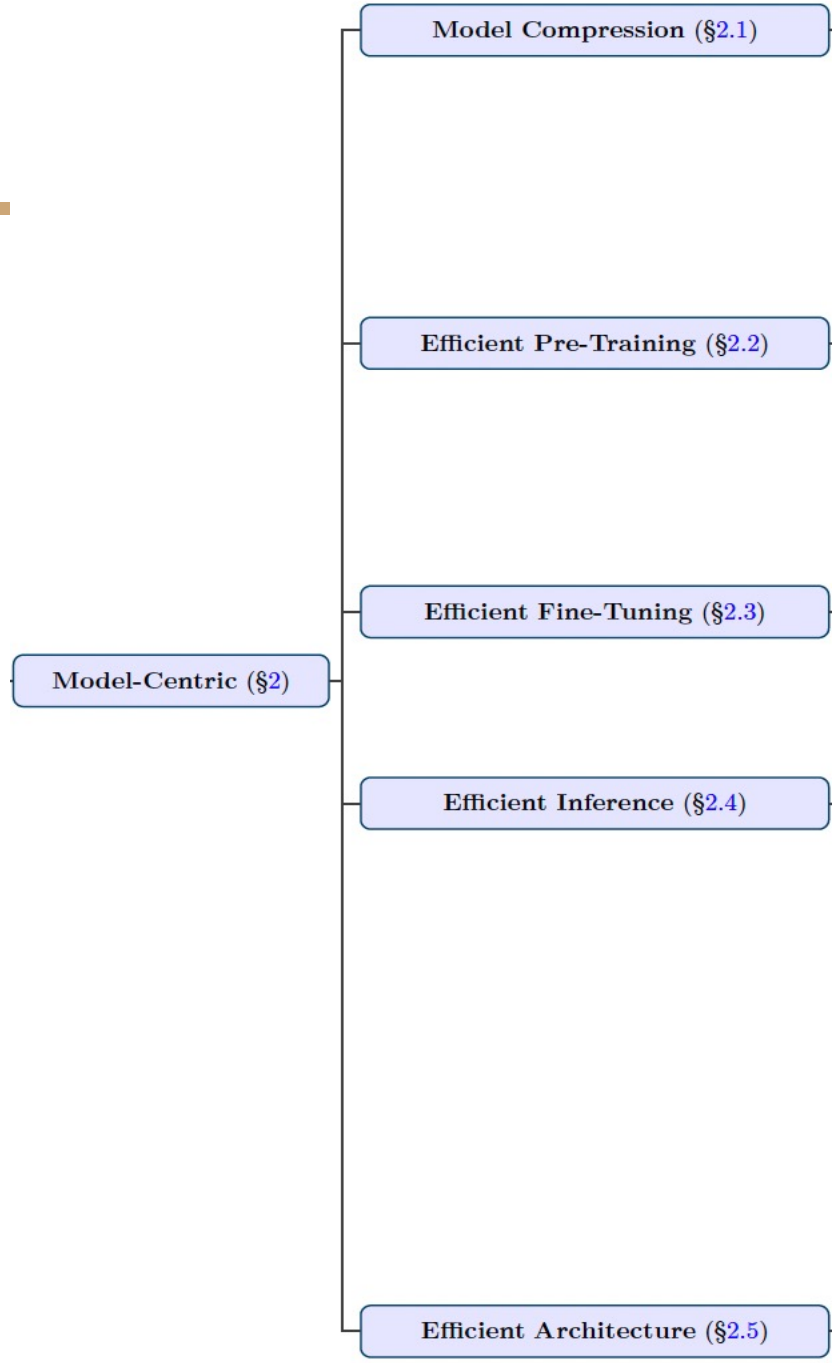
- Grouped-query attention
- Sliding window attention



Source: Hugging Face – LLM perf leaderboard - 2023

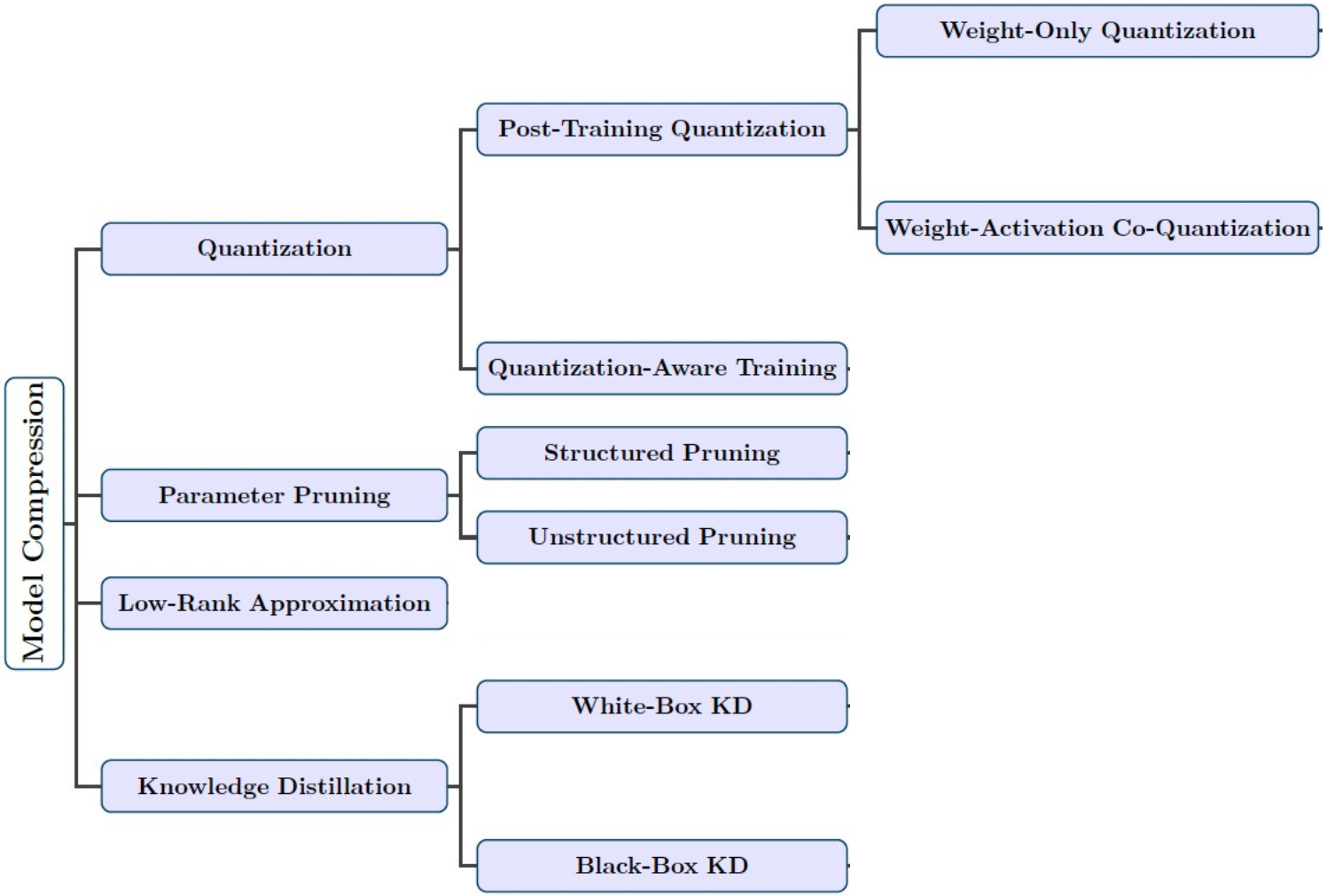
Overview & high-level taxonomy

- This presentation will focus on Model-Centric methods for efficient generative AI
- But note that there are also other approaches for efficient generative AI
  - Data-Centric methods (e.g., data selection)
  - Frameworks (e.g., Megatron, DeepSpeed)



Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Model compression methods



Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Quantization methods

$$\mathbf{X}^{\mathrm{L}} = \mathrm{Round}\left(\frac{\mathrm{absmax}\left(\mathbf{X}^{\mathrm{L}}\right)}{\mathrm{absmax}\left(\mathbf{X}^{\mathrm{H}}\right)}\mathbf{X}^{\mathrm{H}}\right) = \mathrm{Round}\left(\mathcal{K}\cdot\mathbf{X}^{\mathrm{H}}\right),$$

- Example: from 32-bit floating point to 8-bit integer representations of weights and/or activations

- Post-Training Quantization (PTQ)
  - Weight-Only quantization (e.g., LLM.int8() to reduce memory usage during inference)
  - Weight-Activation co-quantization: much harder because activations often include outliers
  - It only requires a small validation dataset for final tuning of weights

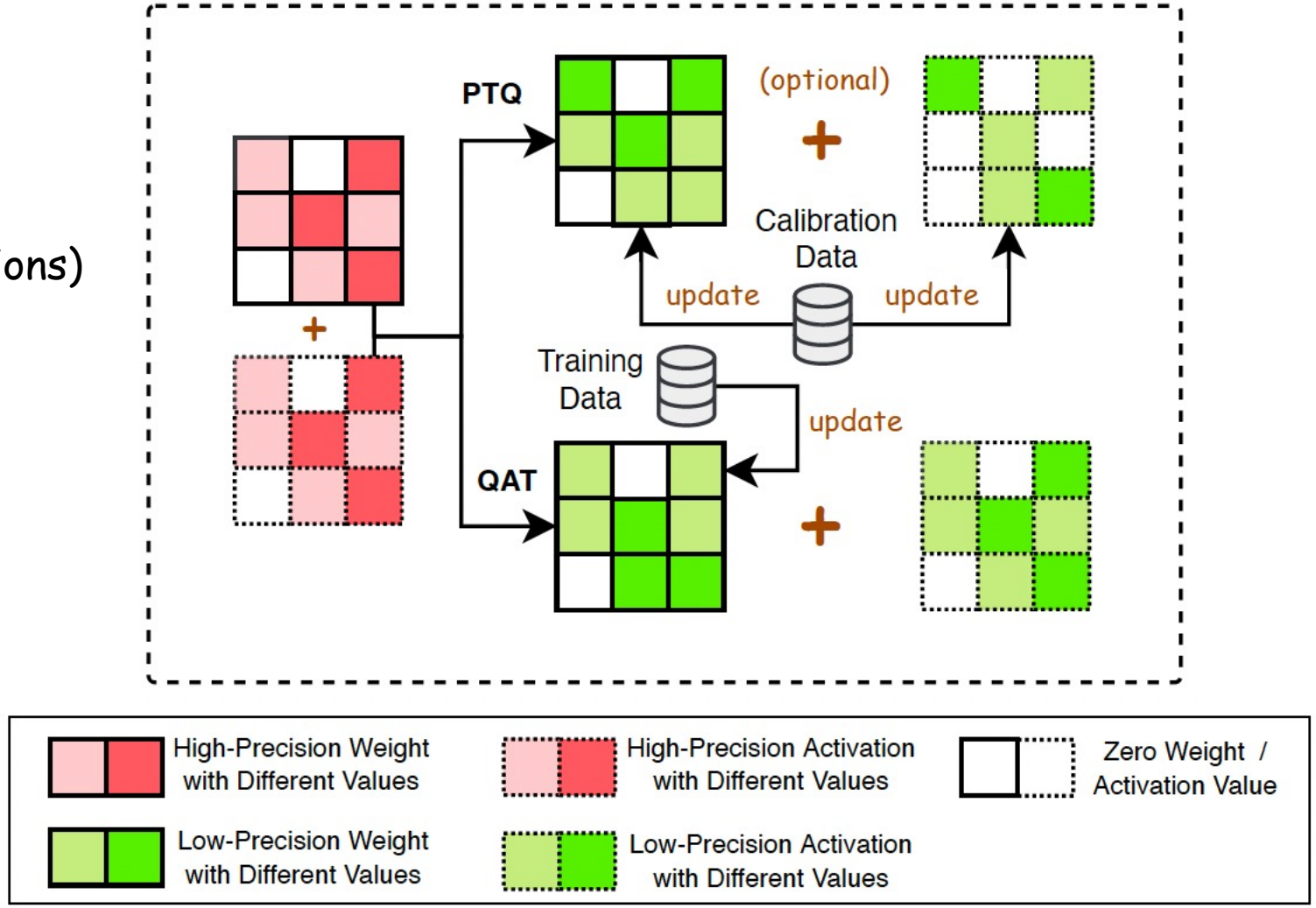
- Quantization-Aware Training (QAT)
  - Goal: learn quantization-friendly representations
  - But requires the complete training dataset

Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Quantization methods

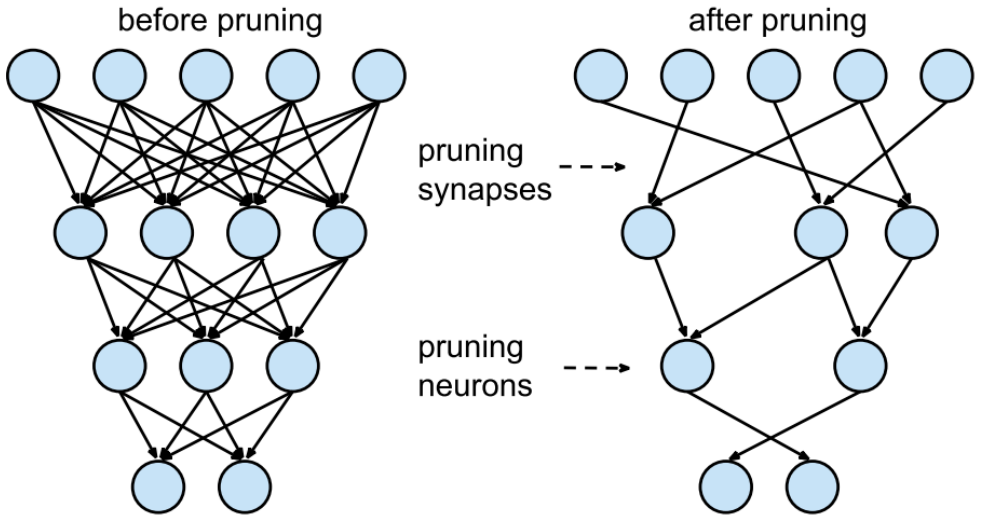
Examples:

- GPTQ (3-4 bits/weight)
- ZeroQuant (PTQ weights & activations)
- QuantGPT (QAT)



Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Neural network pruning

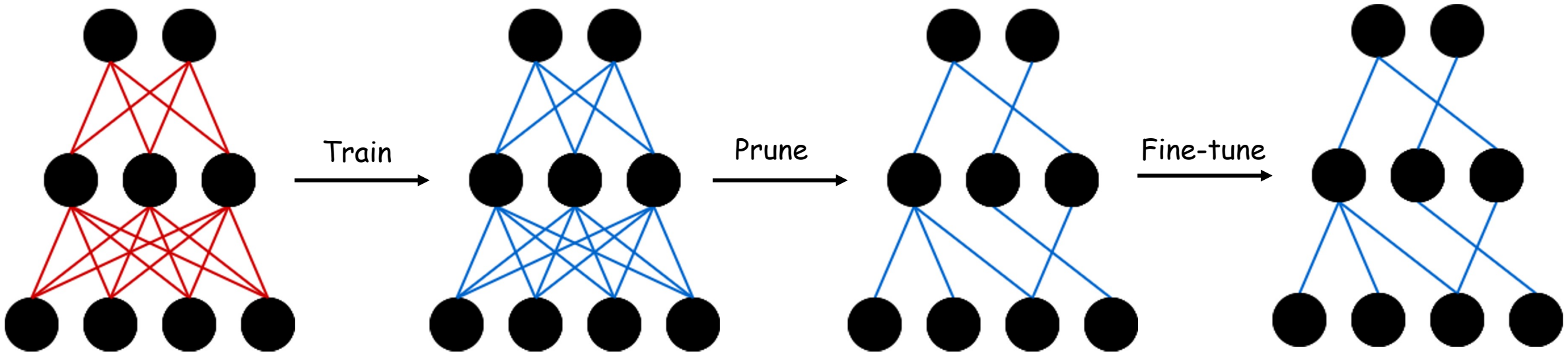


- The concept has been around since the late 80s
- But mostly performed after training – only improves inference

Neural network pruning – after training

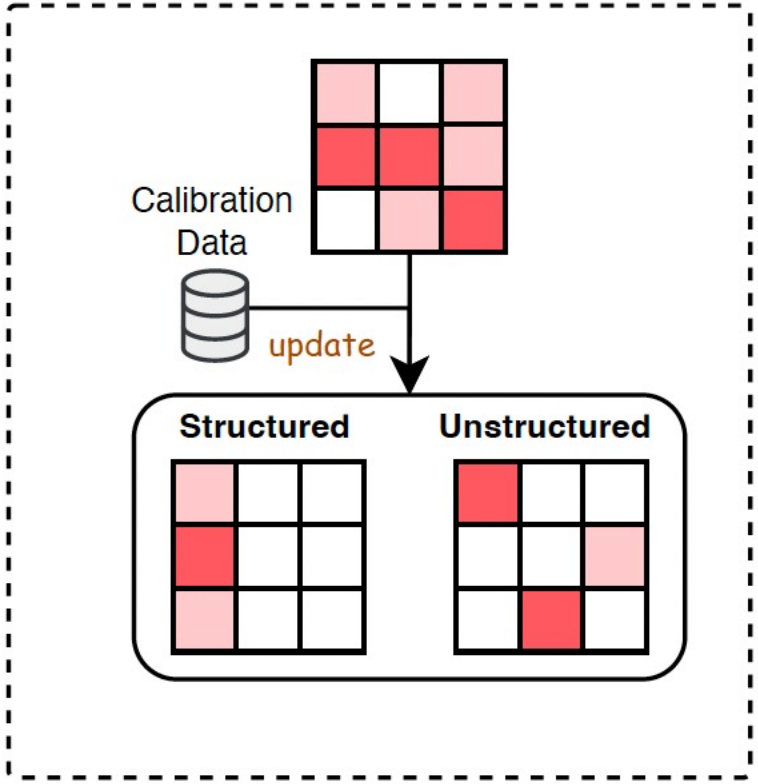
1) Train the given dense network
2) Remove insignificant components
3) Re-train from scratch
4) Optionally : repeat steps 2

Weight → Edges, neurons, filter channel etc.



Structured vs Unstructured pruning

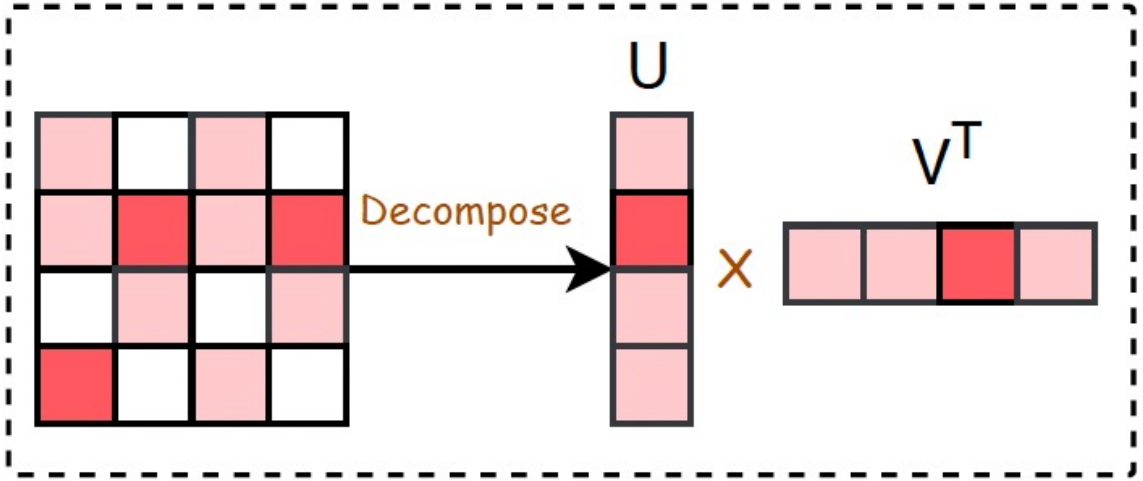
- Structured: remove entire layers, filters in conv-nets, attention heads, rows/columns/subblocks of LLM weight matrices
    - Sheared LLaMMA: Prune LLaMMA-2-7B down to 1.3B parameters
- Unstructured: removes individual weights – more flexible. But it may not be supported by underlying hardware.
    - SparseGPT: completely avoids retraining – prunes 60% of the OPT-135B model parameters with minimal loss of performance



Source: "Efficient Large Language Models: A Survey" by Wan et al. 2024

Low-Rank Approximations

- Reduce number of parameters
- Example: TensorGPT compresses embedding layers of LLMs using Tensor-Train Decomposition
- Some loss in performance should be expected depending on size of dimension r

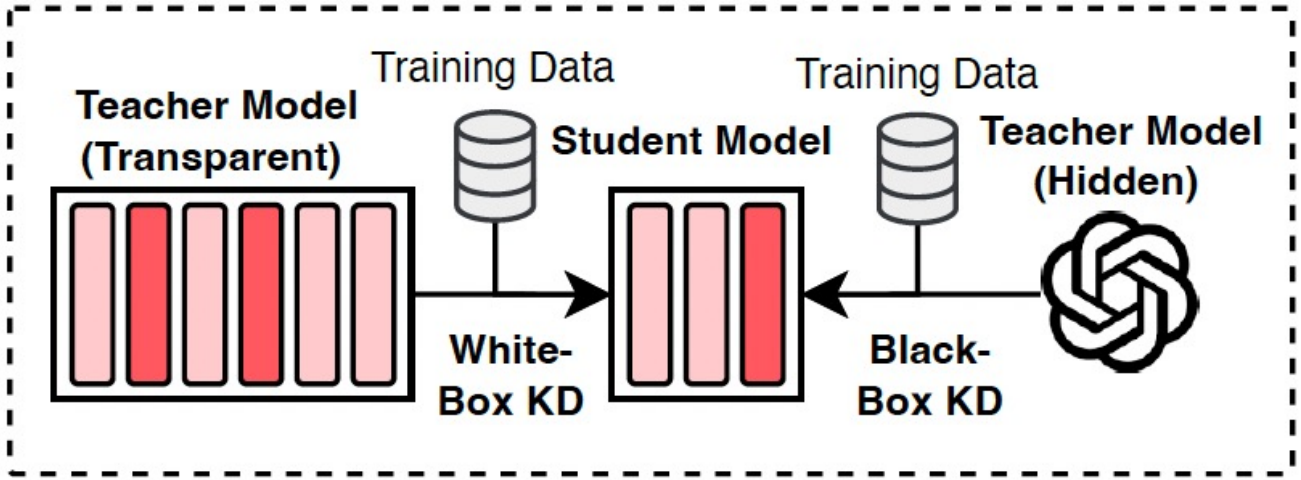


$$\mathbf{W} \approx \mathbf{U}\mathbf{V}^{\top}, \text{ where } \mathbf{U} \in \mathbb{R}^{m \times r}, \mathbf{V} \in \mathbb{R}^{n \times r}$$

Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

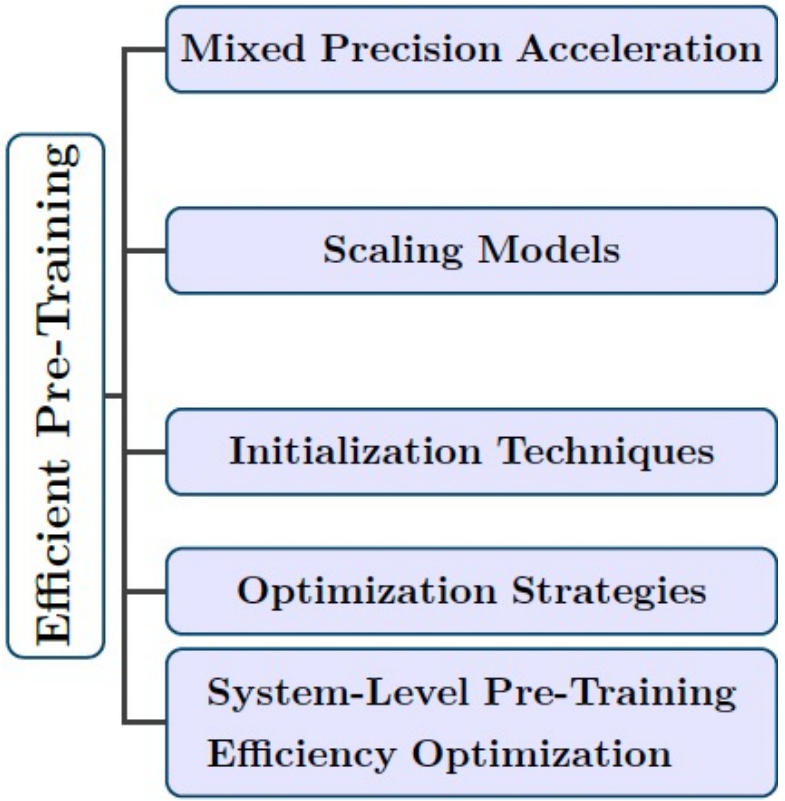
Knowledge Distillation (white-box vs black-box)

- Student model is much smaller than teacher model
- Student model is trained to emulate teacher model
- Student model is much more efficient during inference
- E.g. (white box): BabyLLaMA (58M parameters)
- E.g. (black box): DISCO prompts LLM to generate counterfactual data, distilled on student models



Source: "Efficient Large Language Models: A Survey" by Wan et al. 2024

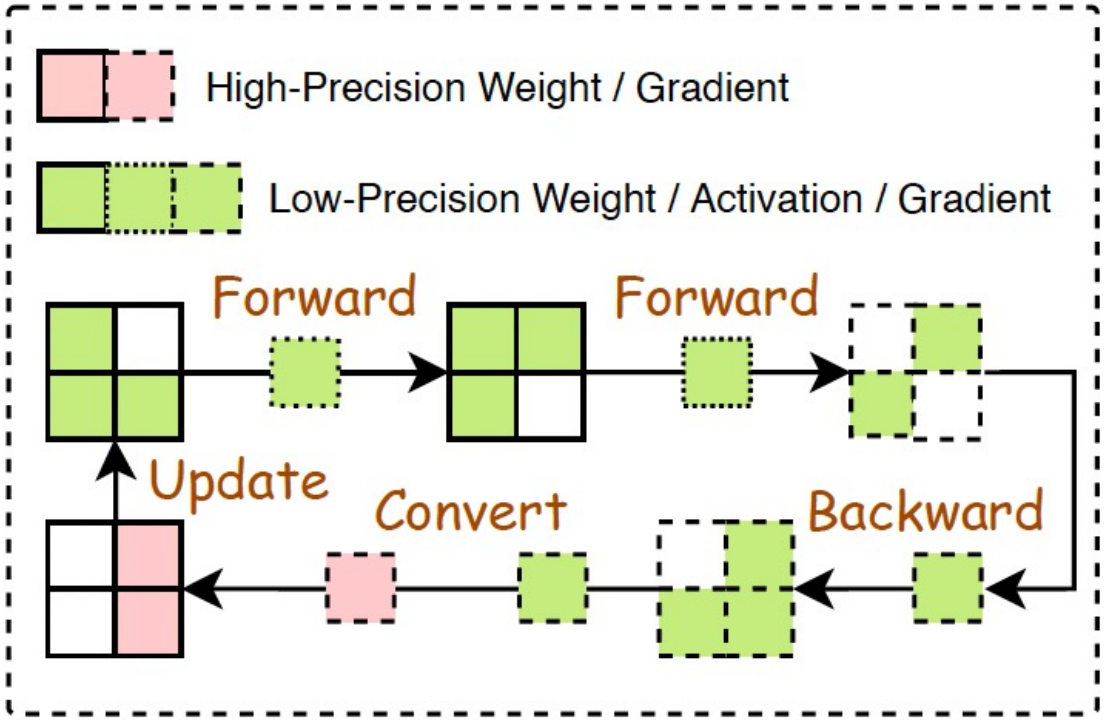
Methods for efficient pre-training



Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Mixed-Precision Acceleration

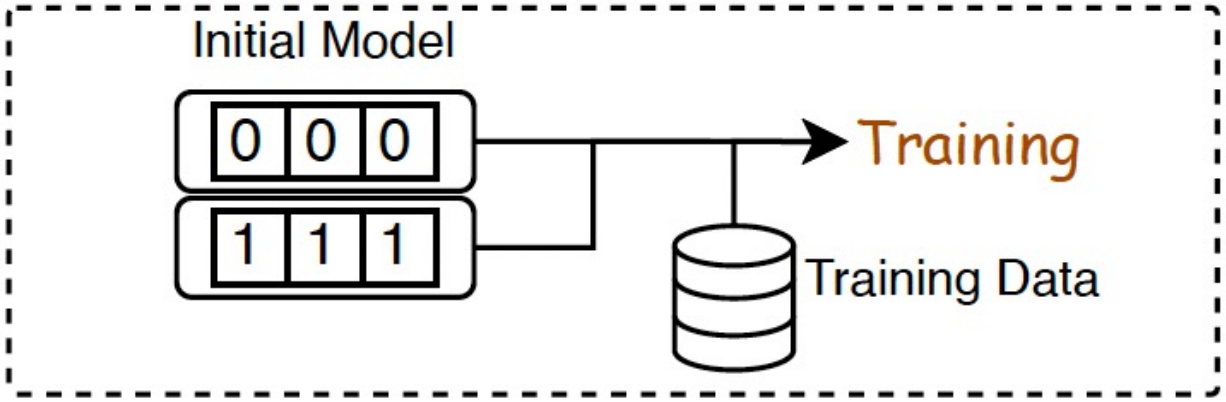
- Use LP model for forward and backward propagation – but convert LP gradients to HP gradients before updating HP weights
- Example: AMP (automatic mixed precision) keeps copy of FP32 weights for updates – but weights, activations, gradients are stored in FP16 for arithmetic operations



Source: "Efficient Large Language Models: A Survey" by Wan et al. 2024

Initialization methods

- Initial weights can strongly affect learning speed and generalizability

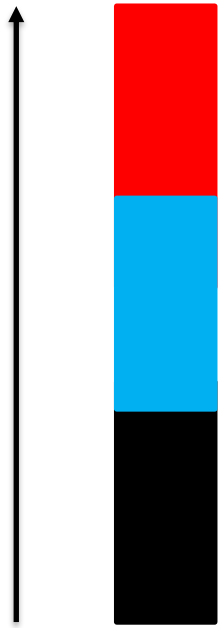


- "PHEW: Constructing sparse networks that learn fast and generalize well without training data"
  Shreyas Malakarjun Patil and Constantine Dovrolis, ICML 2021

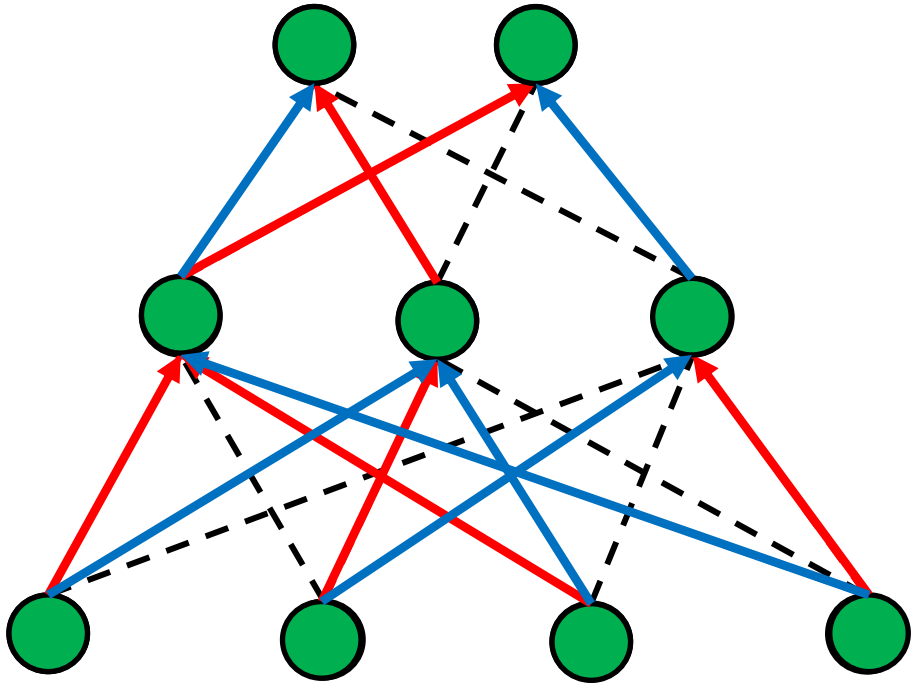
PHEW: Paths with High Edge-Weights

Given a dense, randomly initialized neural network and a target number of weights / parameters ($m = 12$)

Number of weights : [redacted] / 12



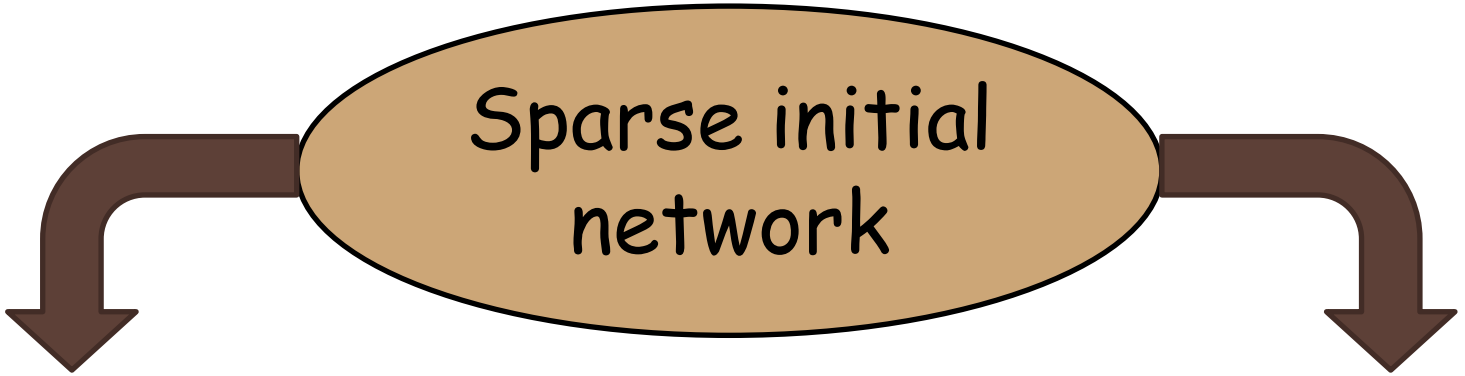
$$Q(j,i) = \frac{|\theta(j,i)|}{\sum_k |\theta(k,i)|}$$



Increasing weight magnitude

Starting unit selected through round robin from both inputs and outputs

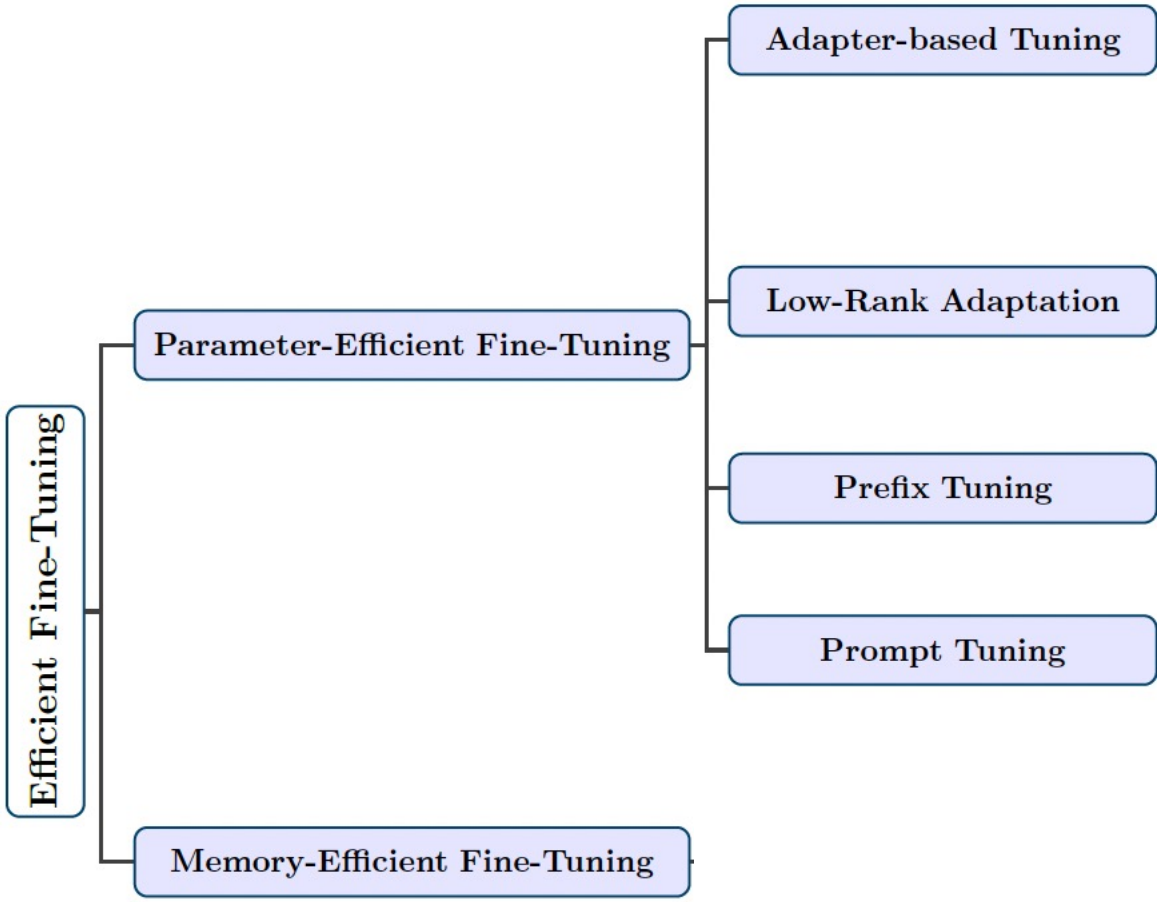
What we get from PHEW



Faster convergence/ efficient pre-training compared to dense network

Better generalization than other sparse networks of the same density

Methods for efficient fine-tuning

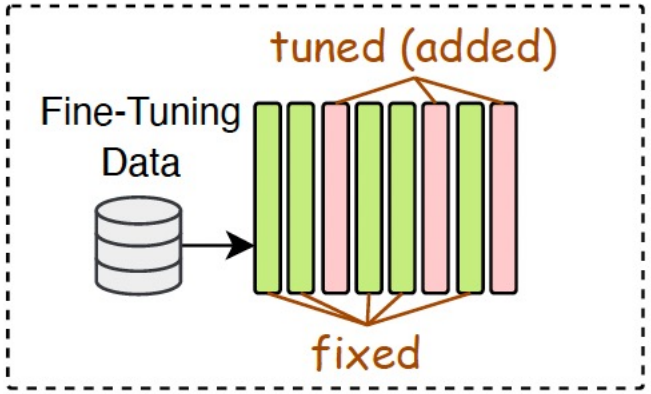


Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

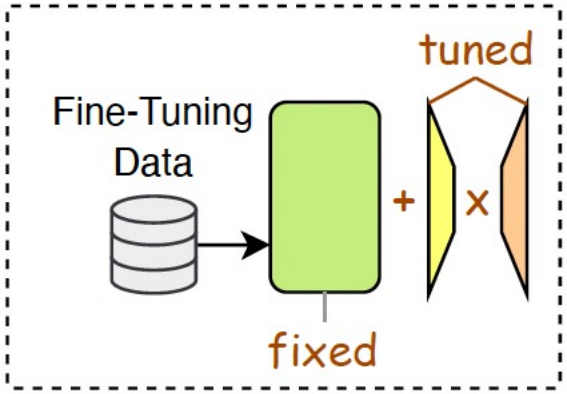
Methods for parameter-efficient fine-tuning

Bottleneck modules added in fixed architecture for specific tasks

E.g. LLM-Adapters



(a) Adapter-based Tuning



(b) Low-Rank Adaptation

$\mathbf{W} \in \mathbb{R}^{m \times n}$ as $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$

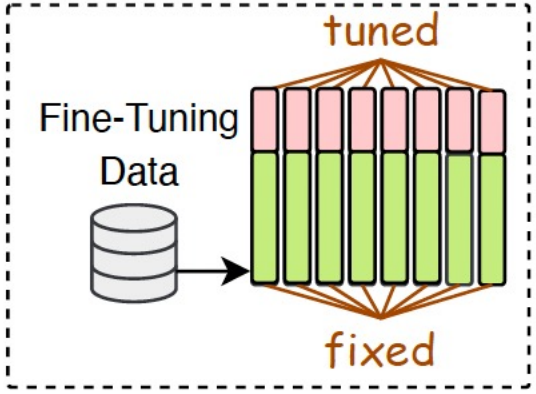
$\Delta\mathbf{W}$ as $\Delta\mathbf{W} = \mathbf{A} \cdot \mathbf{B}$

$\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times n}$

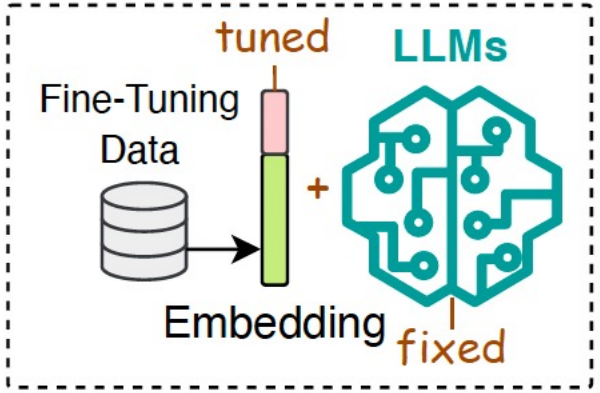
E.g. LoraHub

Add short trainable vectors for each task in each layer

E.g., LLaMA-Adapter



(c) Prefix Tuning



(d) Prompt Tuning

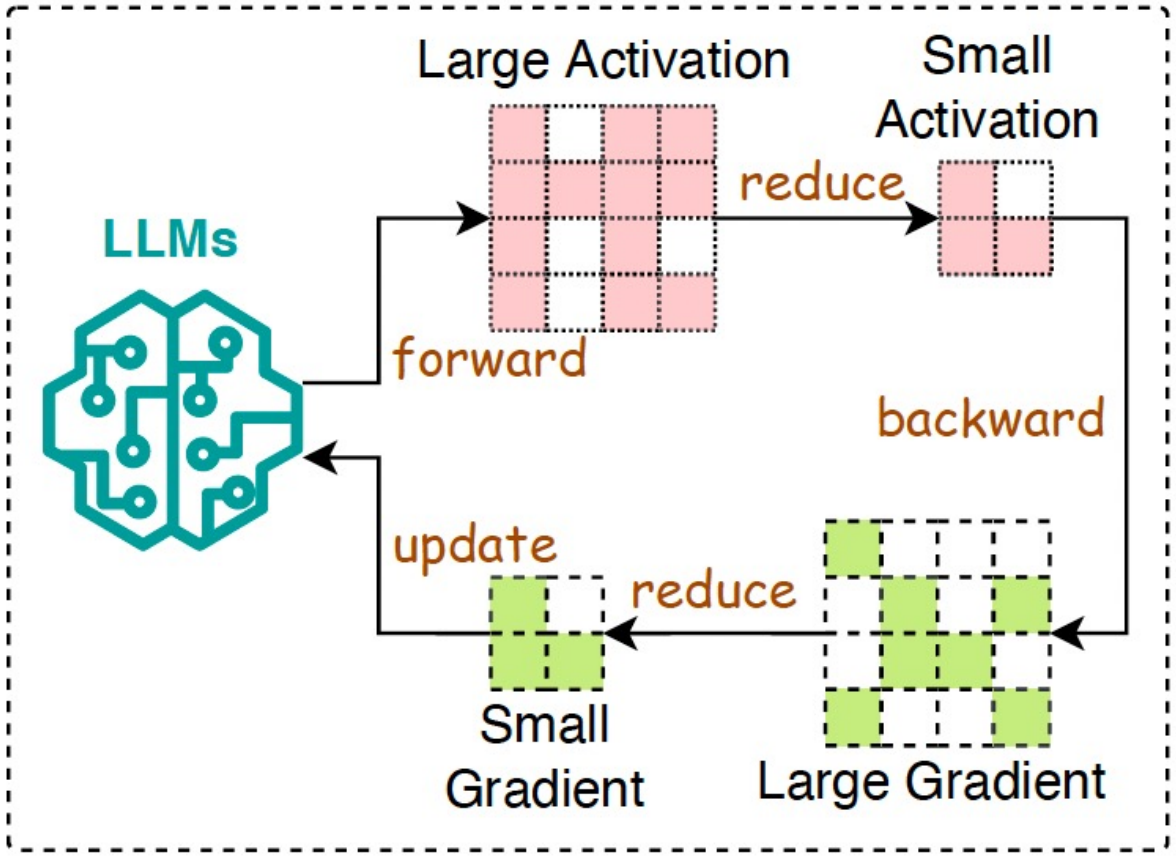
The LLM model remains fixed – only trainable prompt tokens are added/encoded at input layer
E.g., P-Tuning

Source: "Efficient Large Language Models: A Survey" by Wan et al. 2024

Methods for memory-efficient fine-tuning

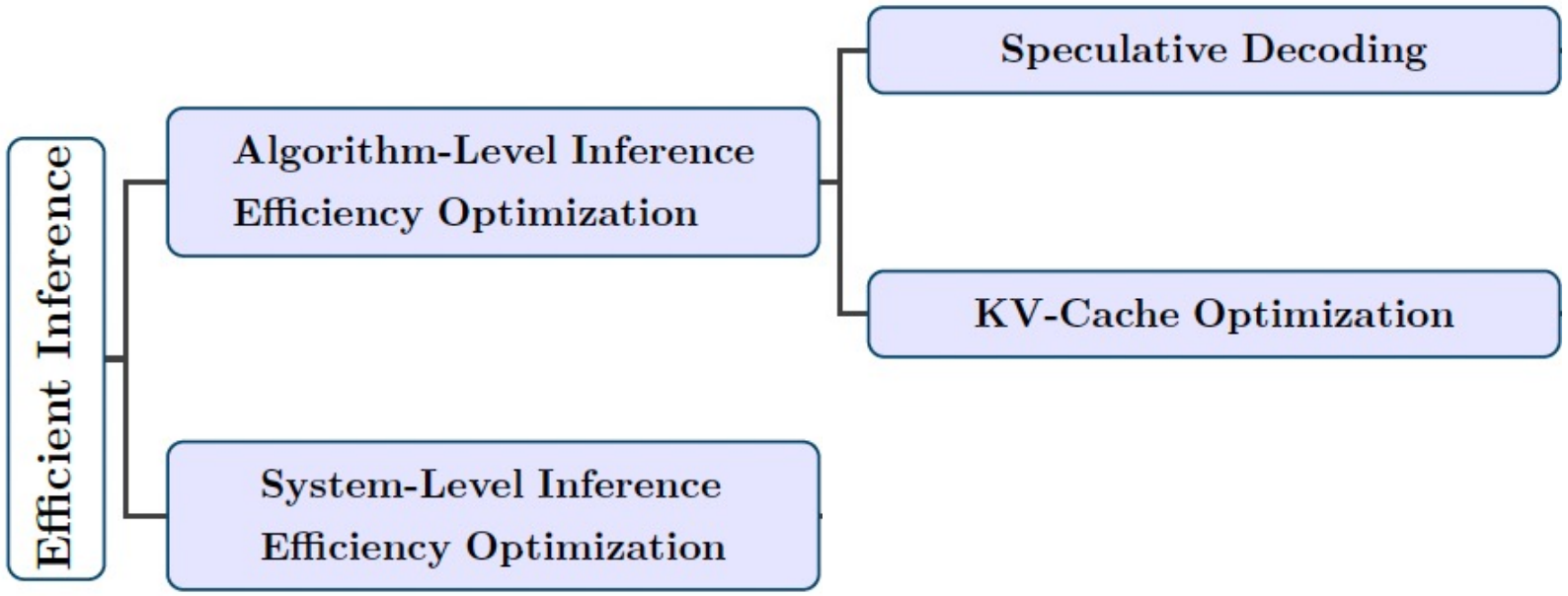
Quantize model into low-precision – then fine-tune quantized model with LoRA

E.g. QLoRA



Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Methods for efficient inference



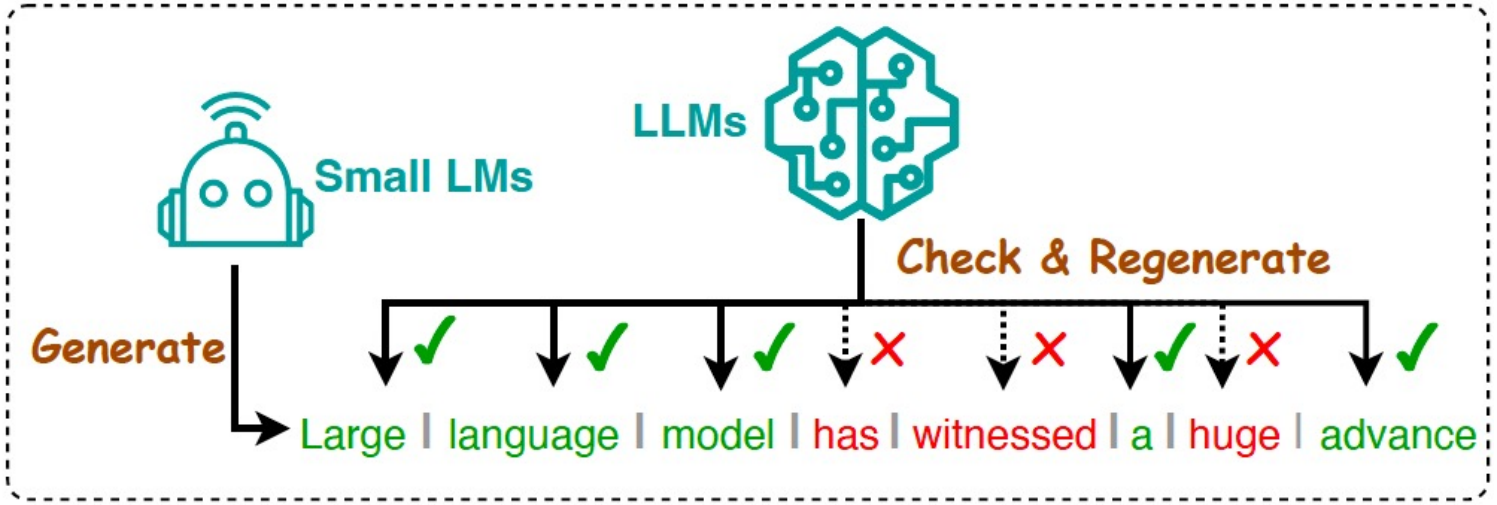
Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

Speculative Decoding

Smaller (faster) models generate speculative tokens for the target model.

Those can be rejected or revisited by the target model

E.g. BiLD, SpecInfer



Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

KV-Cache Optimization

Without KV-cache, attention computation requires quadratic compute with sequence length.

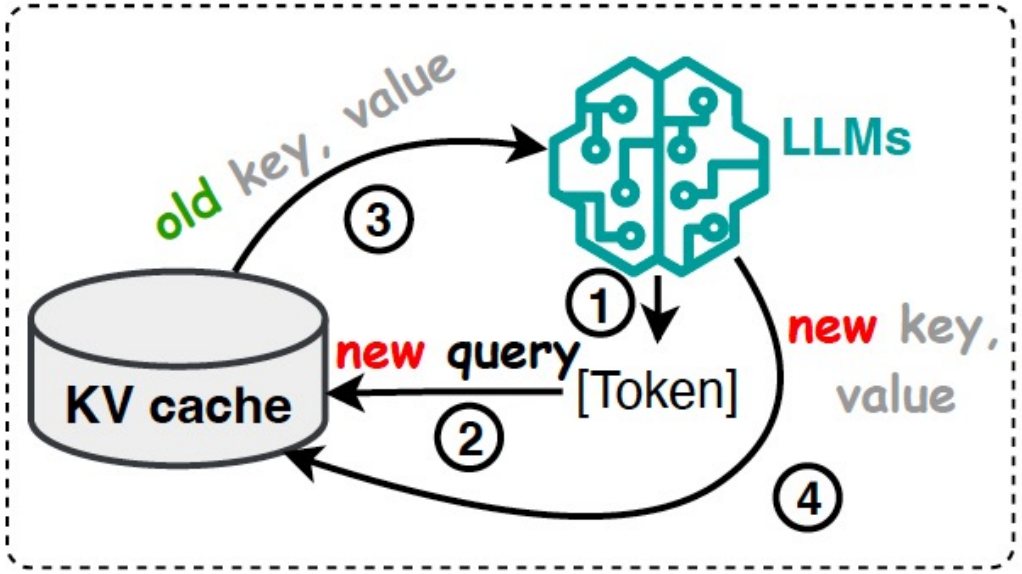
Masking of subsequent tokens (not yet generated) creates redundant computations in self-attention.

Keys and Values for previous tokens can be reused in attention computations for subsequent tokens – and so they are cached.

- Linear scaling with sequence length

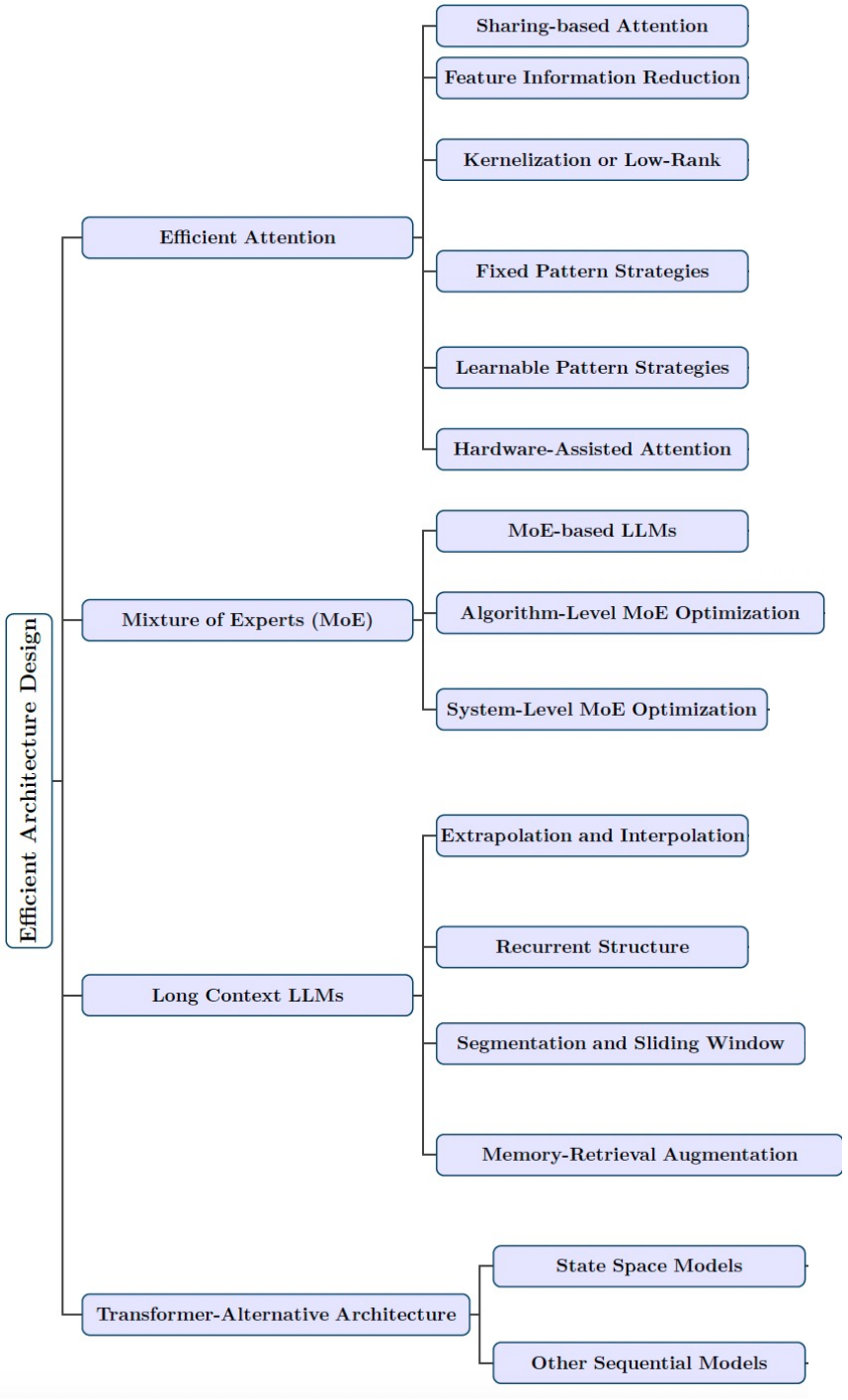
Many possible optimizations for KV-Cache:

- different eviction strategies
- E.g. Dynamic Context Pruning



Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

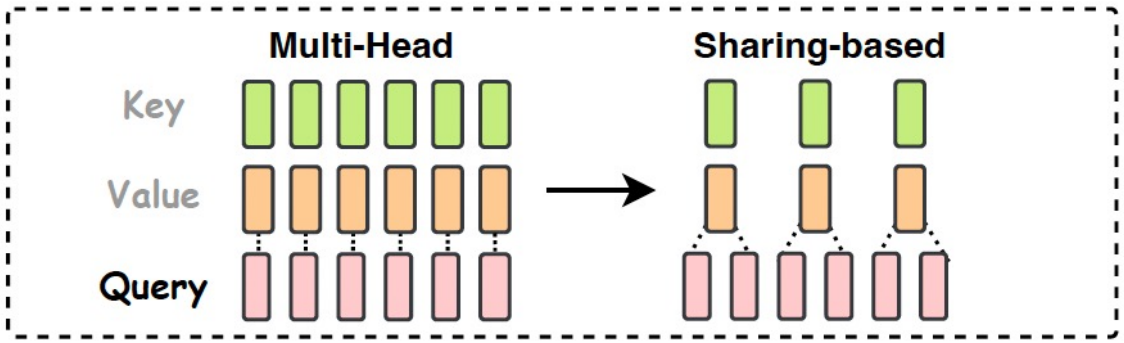
Efficient architectures



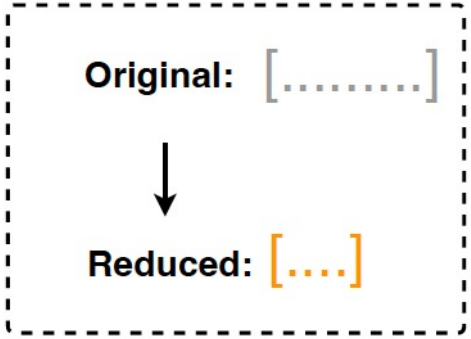
Source: “Efficient Large Language Models: A Survey” by Wan et al. 2024

More advanced attention architectures

Share keys, values across attention heads. E.g., MQA, GQA

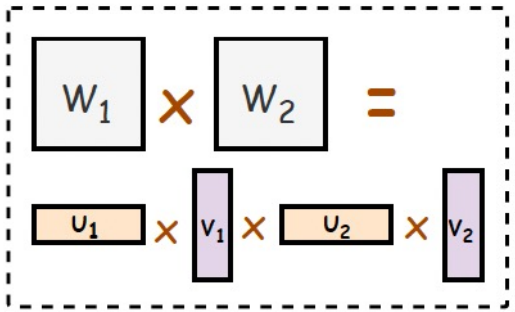


(a) Sharing-based Attention

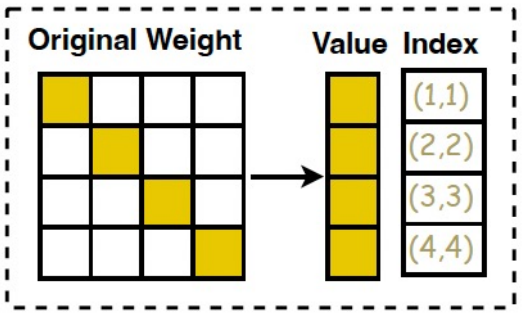


(b) Feature Information Reduction

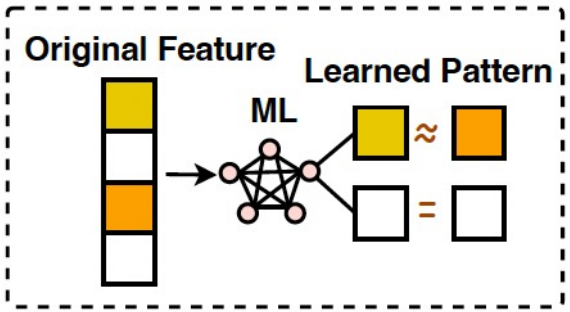
Reduce dimensionality of attention keys, values. Approximate attention calculations.



(c) Kernelization or Low-Rank



(d) Fixed Pattern Strategies



(e) Learnable Pattern Strategies

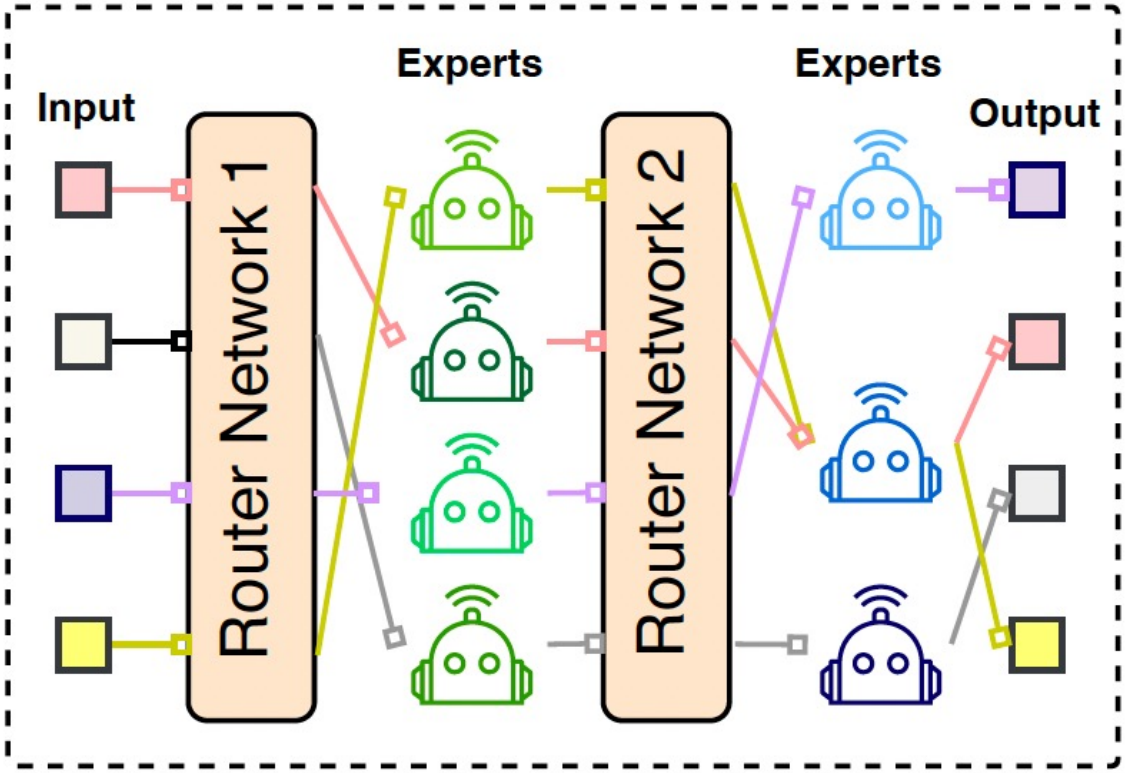
Source: "Efficient Large Language Models: A Survey" by Wan et al. 2024

Mixture-of-Experts models

Segment given task into subtasks. Each subtask allocated to an "expert".

E.g., Switch Transformer:
1 trillion parameters, 2048 experts

MoE models use learnable routing algorithms.
Router performs token-to-expert assignment.



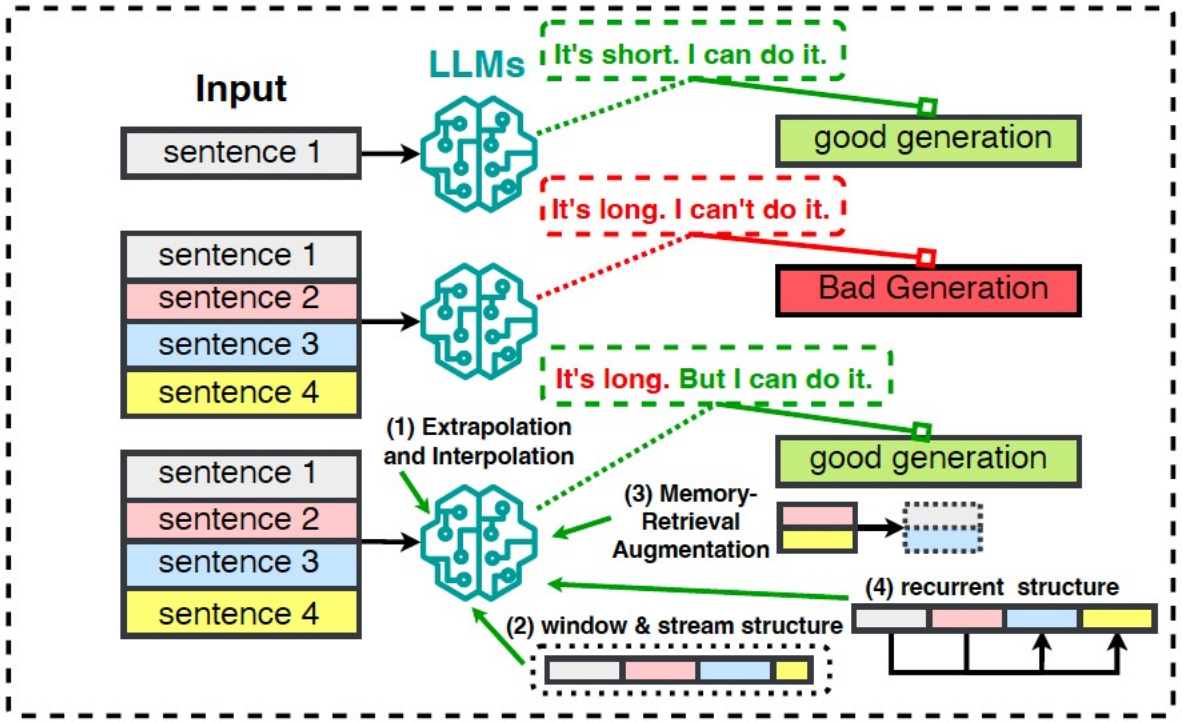
Source: "Efficient Large Language Models: A Survey" by Wan et al. 2024

Long-context LLMs

What if given (or generated) sequence is much longer than anything seen during pre-training?

Extrapolation/Interpolation: replace positional encoding – e.g., with diminishing penalty based on distance between pertinent key and value

Sliding window/segmentation: compute attention over a sliding window -- instead of the entire context length



Source: "Efficient Large Language Models: A Survey" by Wan et al. 2024

An excellent survey (Arxiv: 2312.03863)

Efficient Large Language Models: A Survey

Zhongwei Wan* *wan.512@osu.edu*

Xin Wang* *wang.15980@osu.edu*

Che Liu† *che.liu21@imperial.ac.uk*

Samiul Alam* *alam.140@osu.edu*

Yu Zheng‡ *zhengy30@msu.edu*

Jiachen Liu§ *amberljc@umich.edu*

Zhongnan Qu¶ ‡‡ *znqu@amazon.com*

Shen Yan‖ *shenyan@google.com*

Yi Zhu†† *yi@boson.ai*

Quanlu Zhang** *quzha@microsoft.com*

Mosharaf Chowdhury§ *mosharaf@umich.edu*

Mi Zhang* *mizhang.1@osu.edu*

** The Ohio State University † Imperial College London ‡ Michigan State University § University of Michigan ¶ Amazon AWS AI ‖ Google Research ** Microsoft Research Asia †† Boson AI*

Thanks for your attention

Please email me if you have any questions or feedback

c.dovrolis@cyi.ac.cy

Supported by the EuroCC-2 project

Funded by
the European Union

RESEARCH
& INNOVATION
FOUNDATION

EuroHPC
Joint Undertaking