NOV 04, 2025

AI4SCIENCE SERIES:

ADVANCED TOPICS IN AI FOR SCIENCE

HYBRID PRE-TRAINING OF LARGE MODELS BY LEVERAGING LOW-RANK ADAPTERS

REET BARIK

Postdoctoral Appointee

Portability and AI at scale

Argonne Leadership Computing Facility

U.S. DEPARTMENT of ENERGY

Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

Argonne NATIONAL LABORATORY | Argonne Leadership Computing Facility

AGENDA

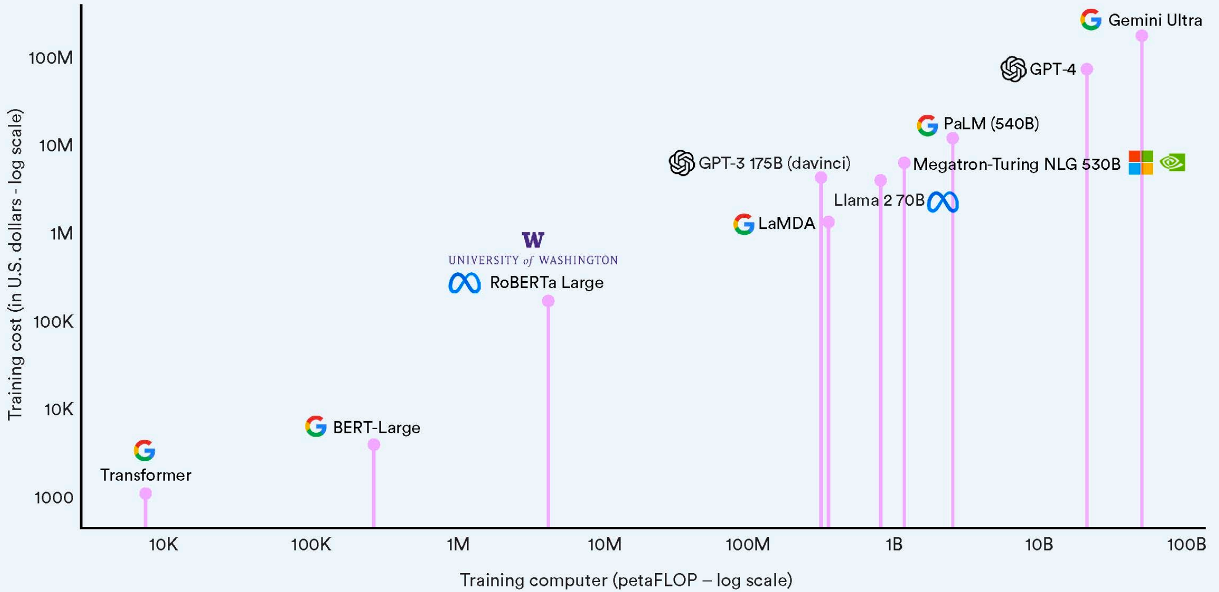
Science Talk Overview

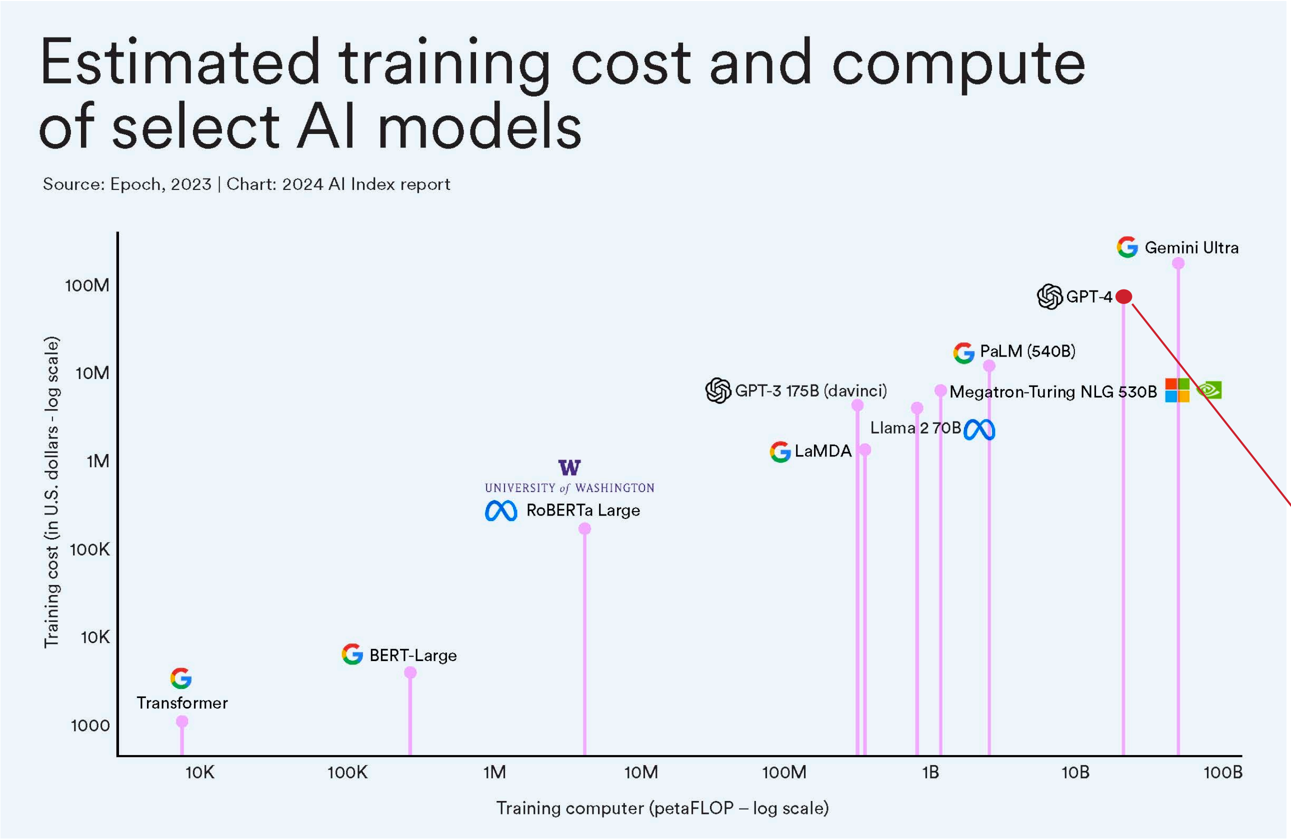
- Background
  - Fine-tuning
  - LoRA (Low-Rank Adaptation)
- Pre-LoRA
  - Motivation
  - Challenges
  - Methodology
  - Results

Q & A

Estimated training cost and compute of select AI models

Source: Epoch, 2023 | Chart: 2024 AI Index report

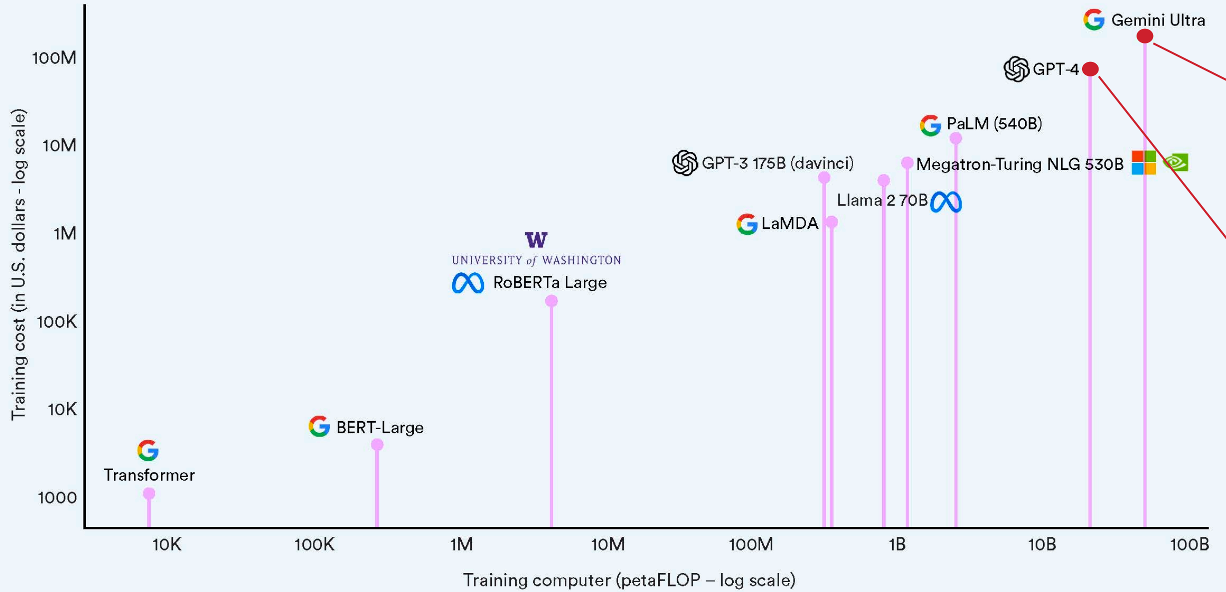




Compute:
21 billion petaFLOPS
Cost:
$78 million

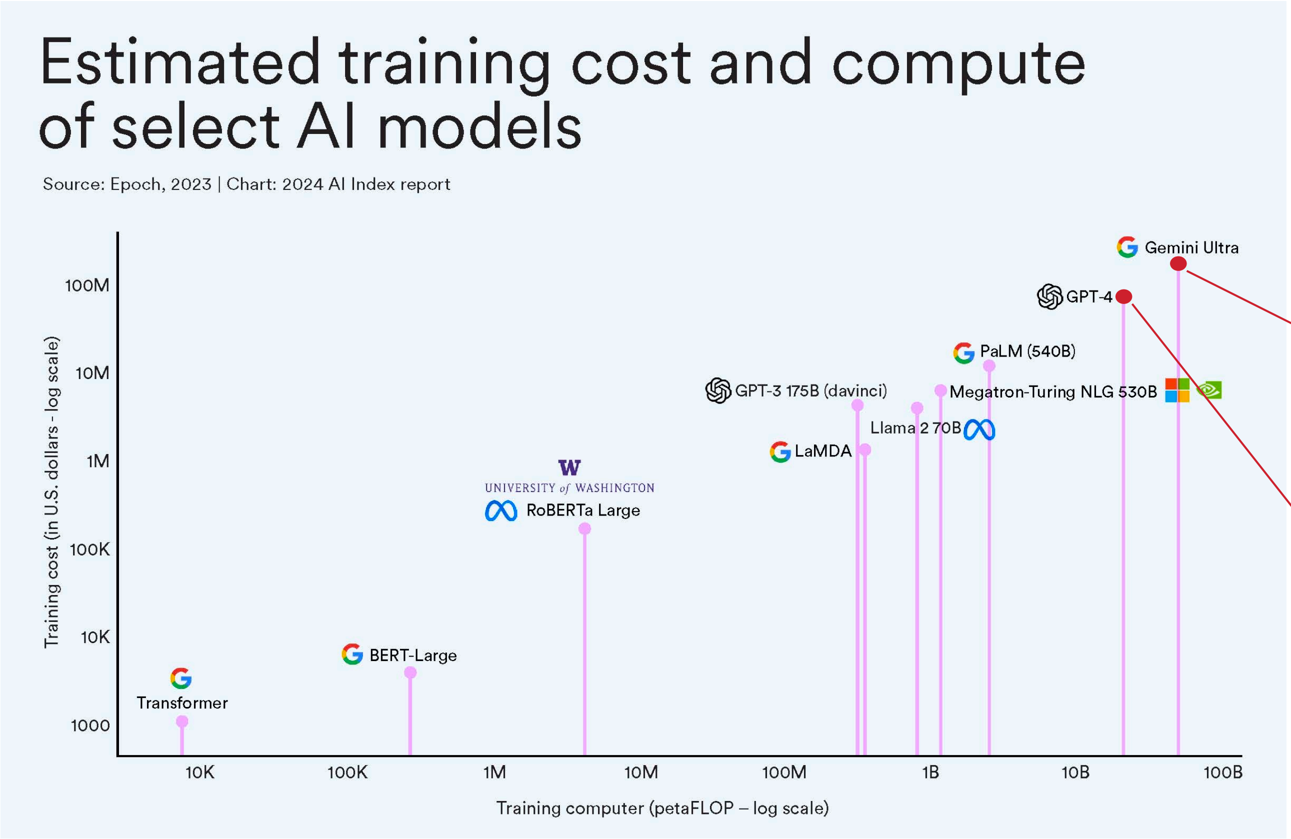
Estimated training cost and compute of select AI models

Source: Epoch, 2023 | Chart: 2024 AI Index report



Compute:
50 billion petaFLOPS
Cost:
$191 million

Compute:
21 billion petaFLOPS
Cost:
$78 million



Compute:
50 billion petaFLOPS
Cost:
$191 million

Compute:
21 billion petaFLOPS
Cost:
$78 million

We need to come up with techniques to make training more efficient and cheap

Inspiration from **fine tuning**…

Inspiration from **fine tuning**…

Training

LLM

Inspiration from **fine tuning**…

Training

LLM

Dataset

Inspiration from **fine tuning**…

Training

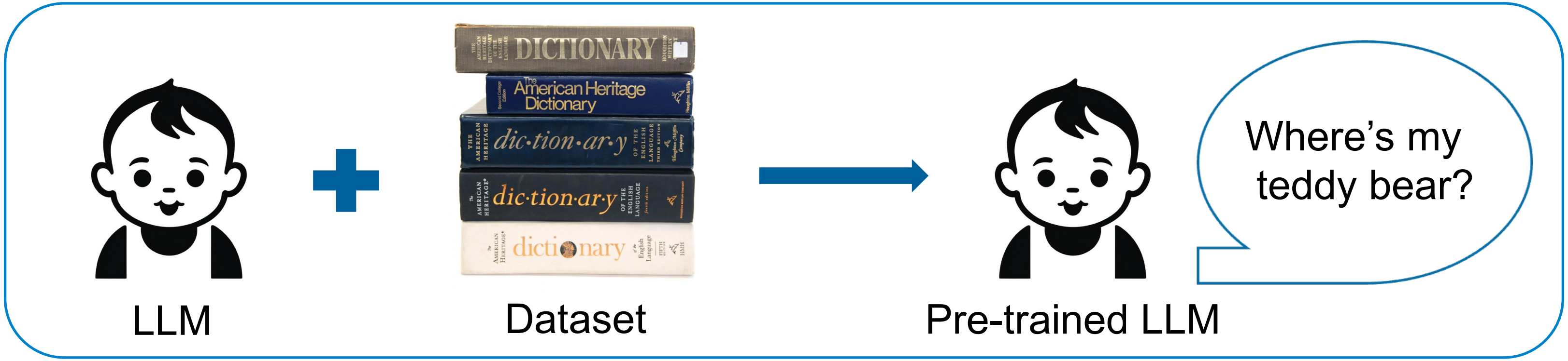
LLM

Dataset

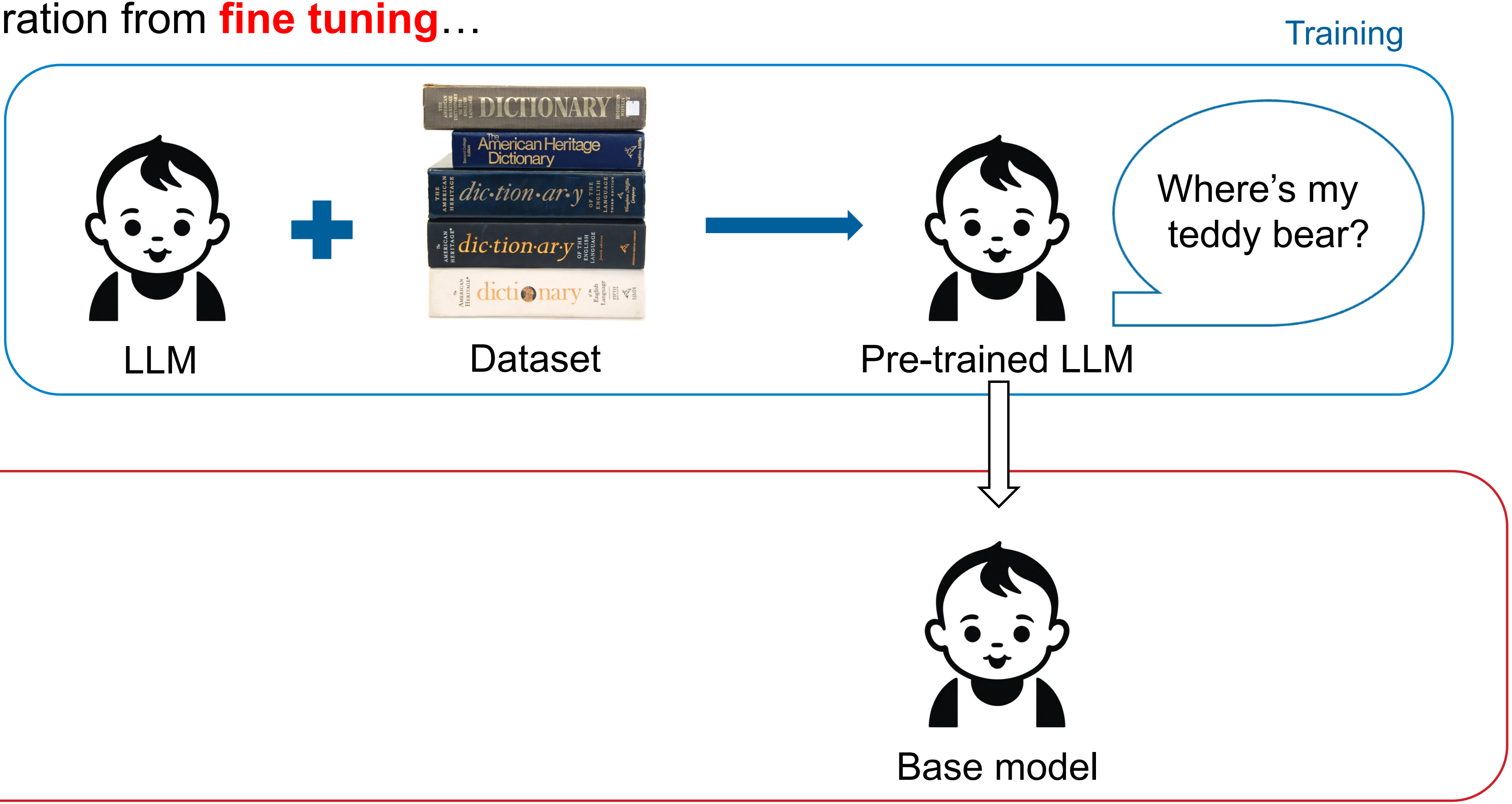
Pre-trained LLM

Inspiration from **fine tuning**…

Training



Inspiration from **fine tuning**…



Inspiration from **fine tuning**…

Training

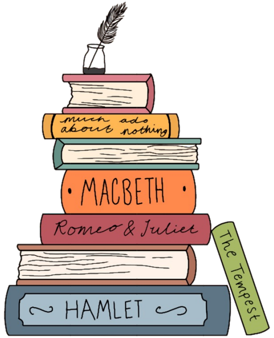
LLM

Dataset

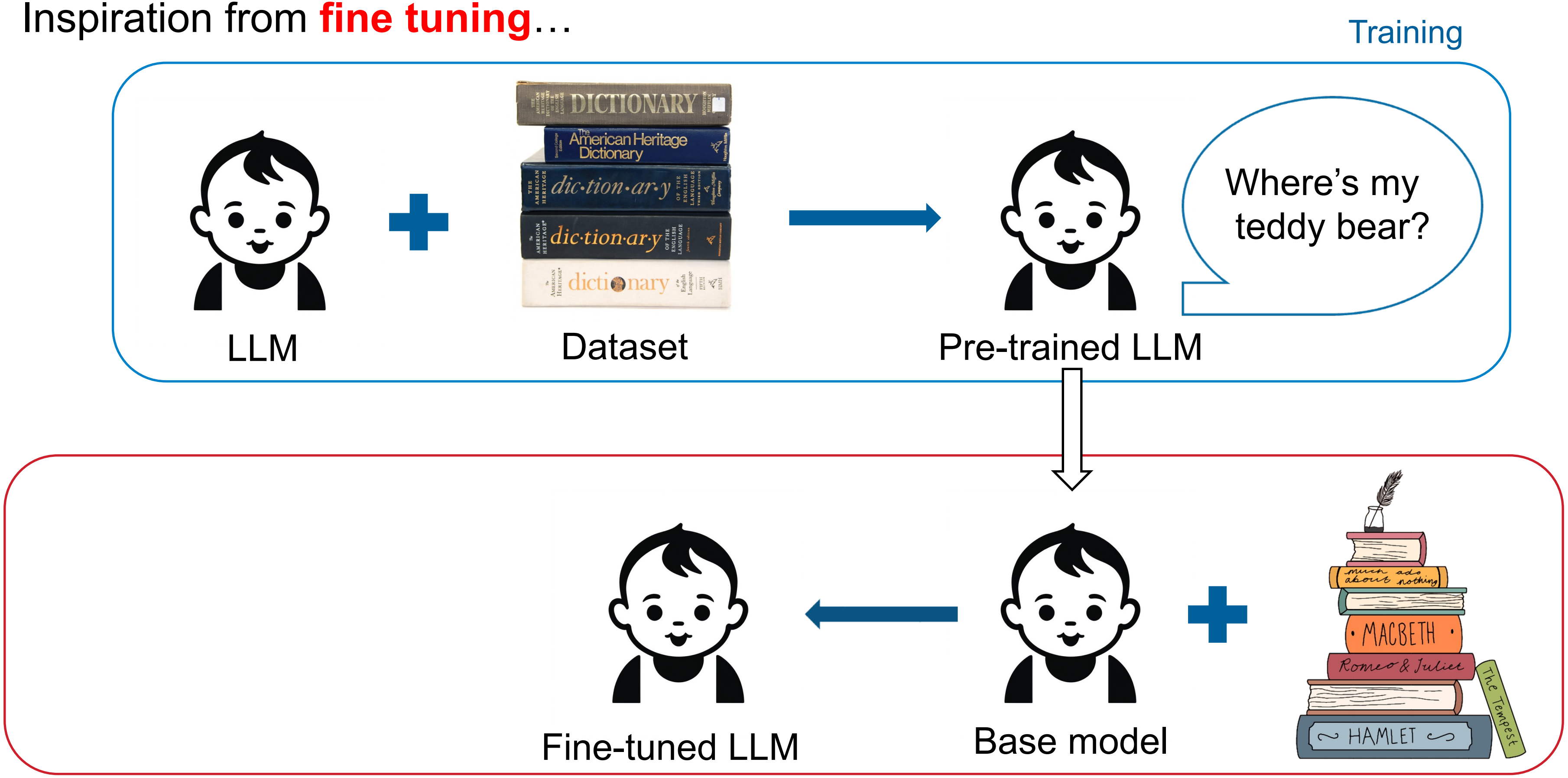
Where's my teddy bear?

Pre-trained LLM

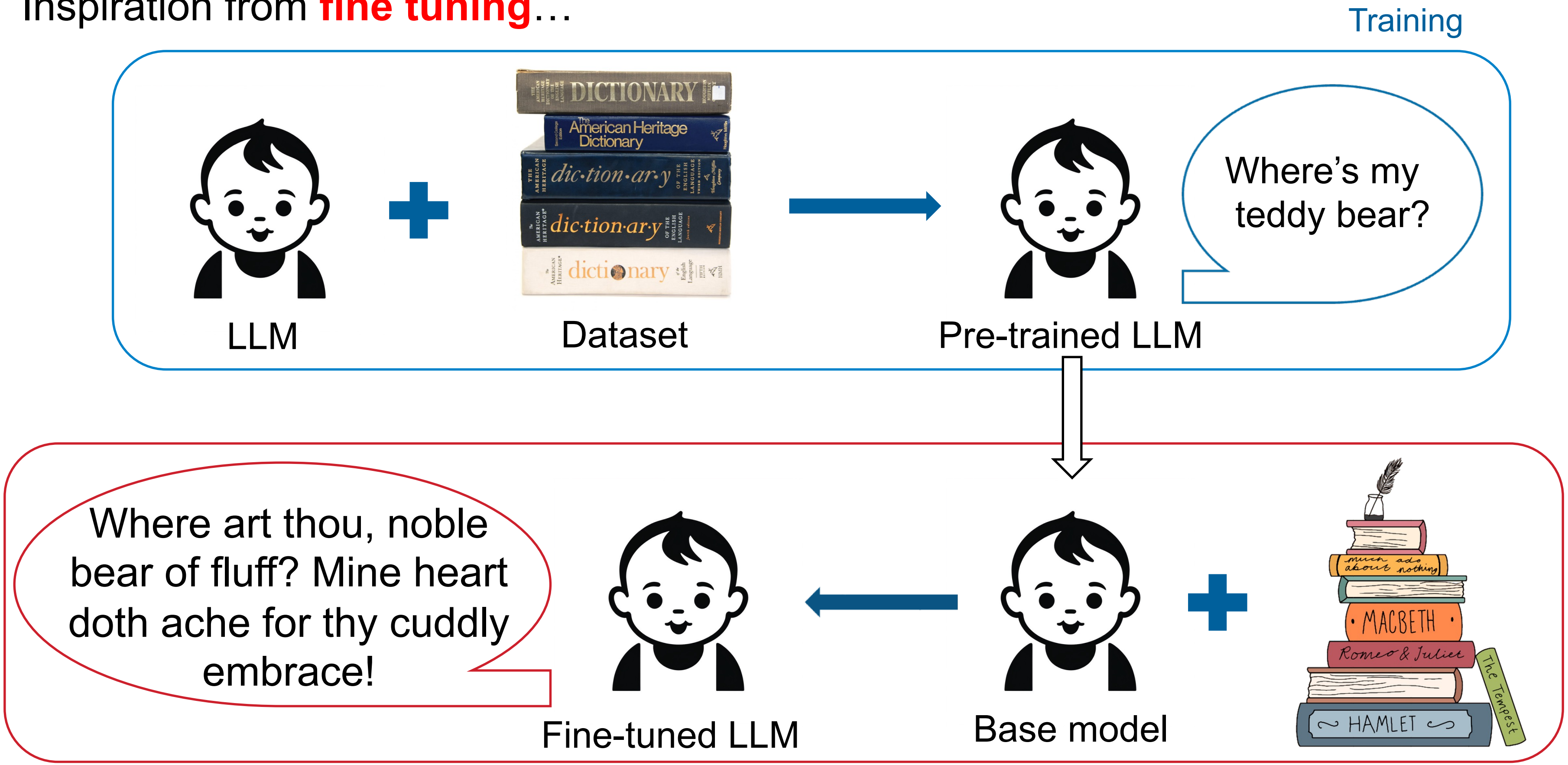
Base model



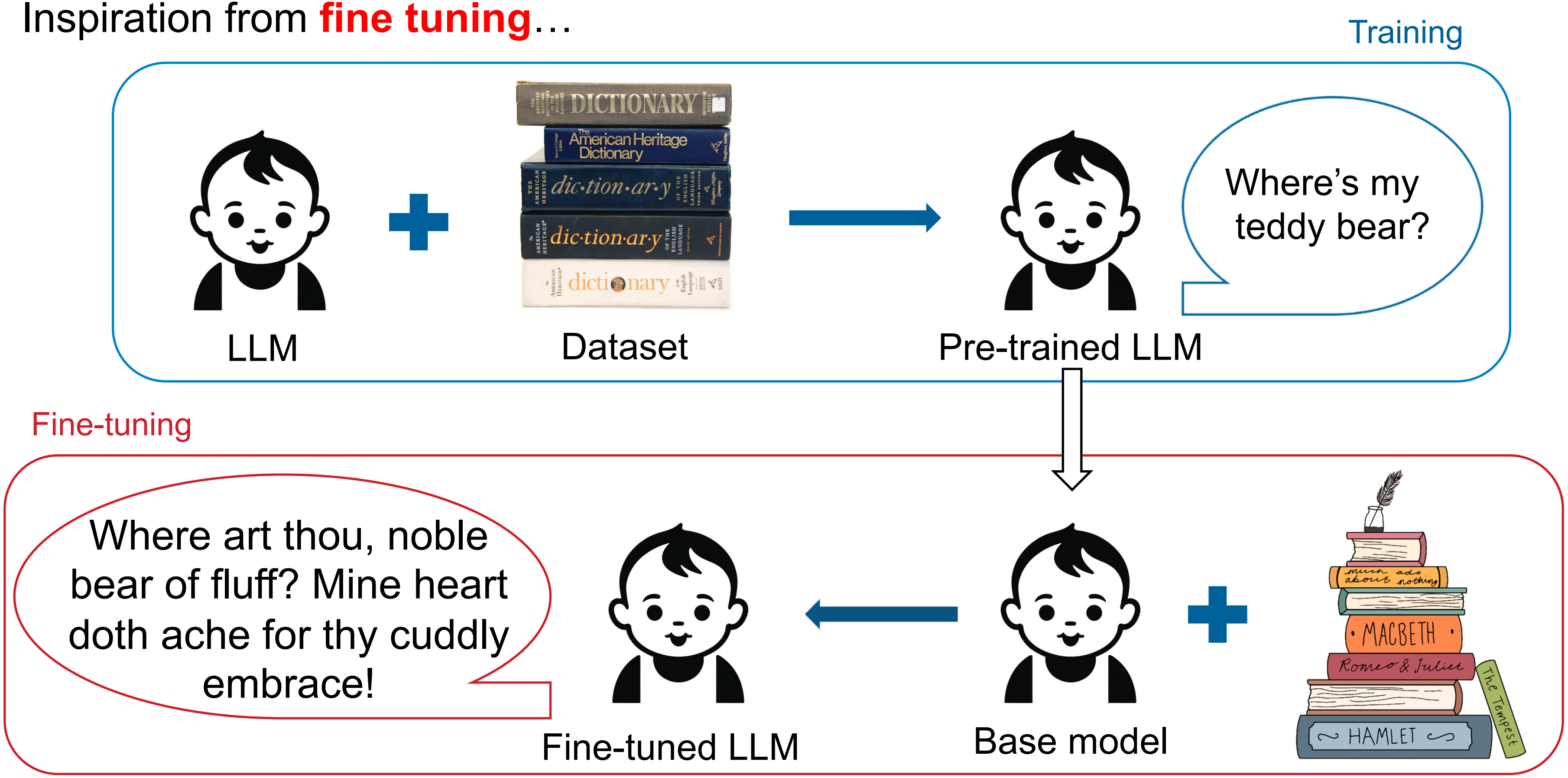
Inspiration from **fine tuning**…



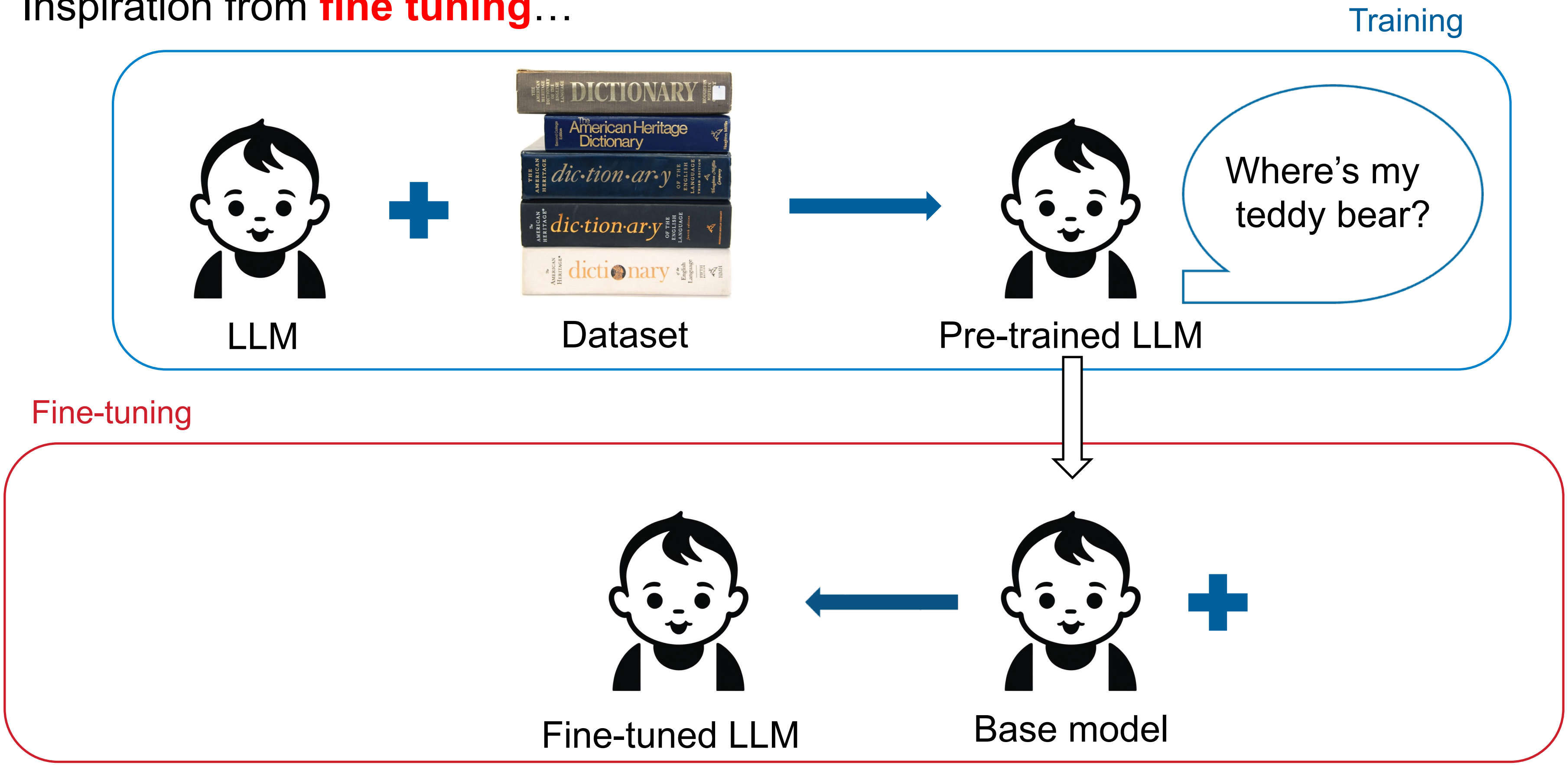
Inspiration from **fine tuning**…



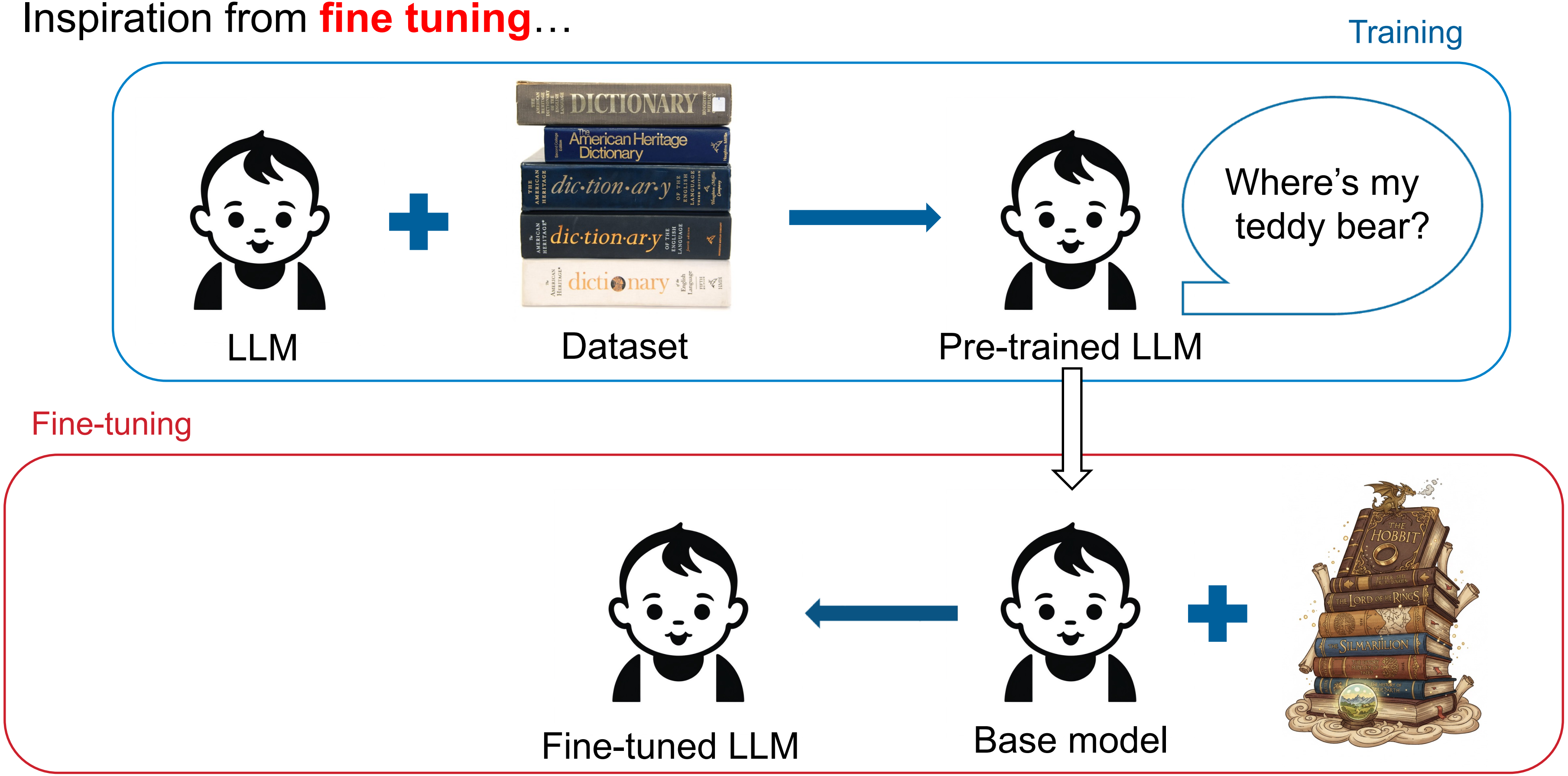
Inspiration from **fine tuning**…



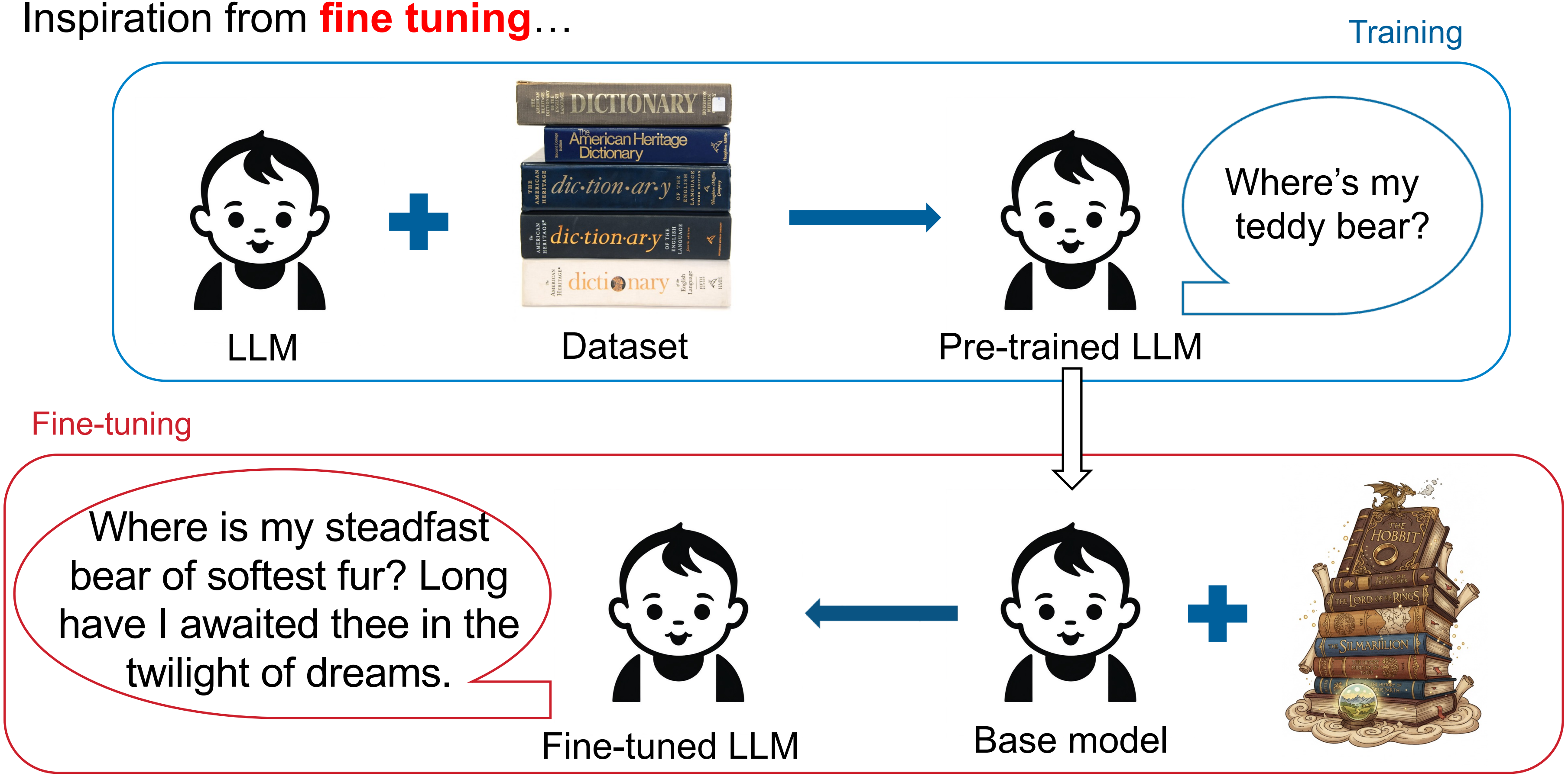
Inspiration from **fine tuning**…



Inspiration from **fine tuning**…



Inspiration from **fine tuning**…



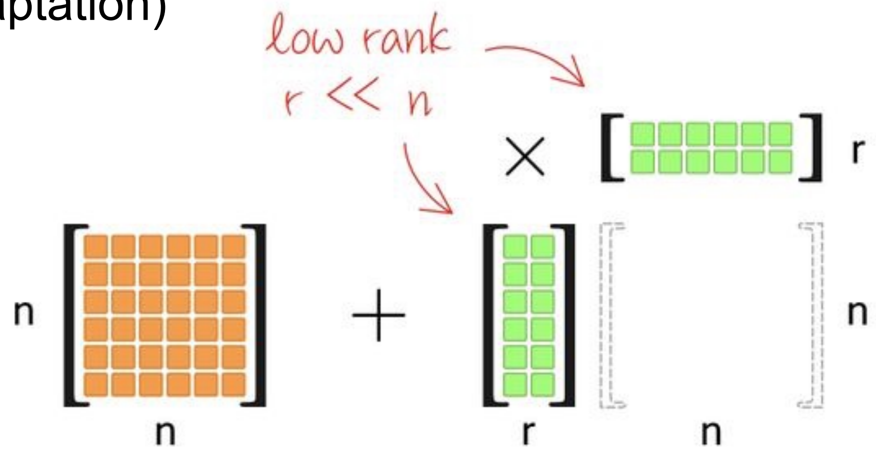
Fine tuning using LoRA (Low-Rank Adaptation)

Fine tuning using LoRA (Low-Rank Adaptation)

$$W = W_0$$

Source: https://onmine.io/are-you-still-using-lora-to-fine-tune-your-llm/

Fine tuning using LoRA (Low-Rank Adaptation)

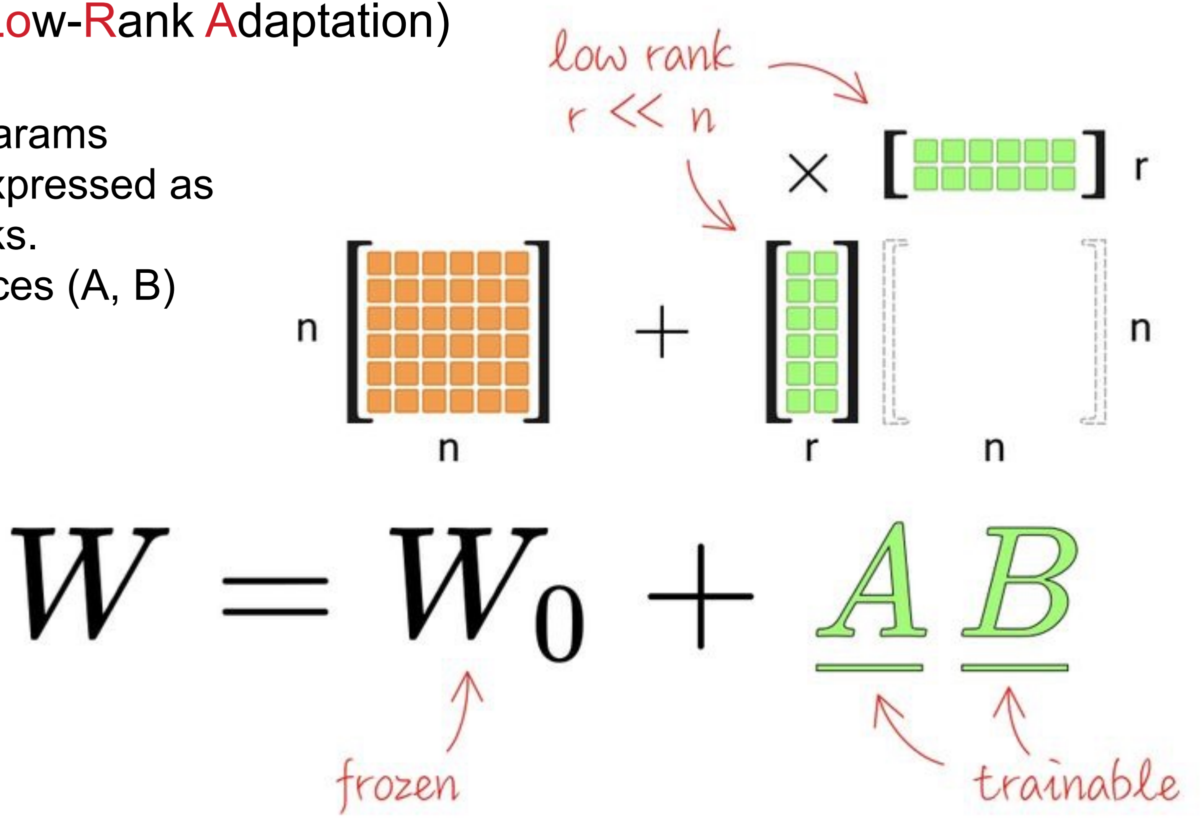


$$W = W_0 + \underline{A}\,\underline{B}$$

Source: https://onmine.io/are-you-still-using-lora-to-fine-tune-your-llm/

Fine tuning using LoRA (Low-Rank Adaptation)

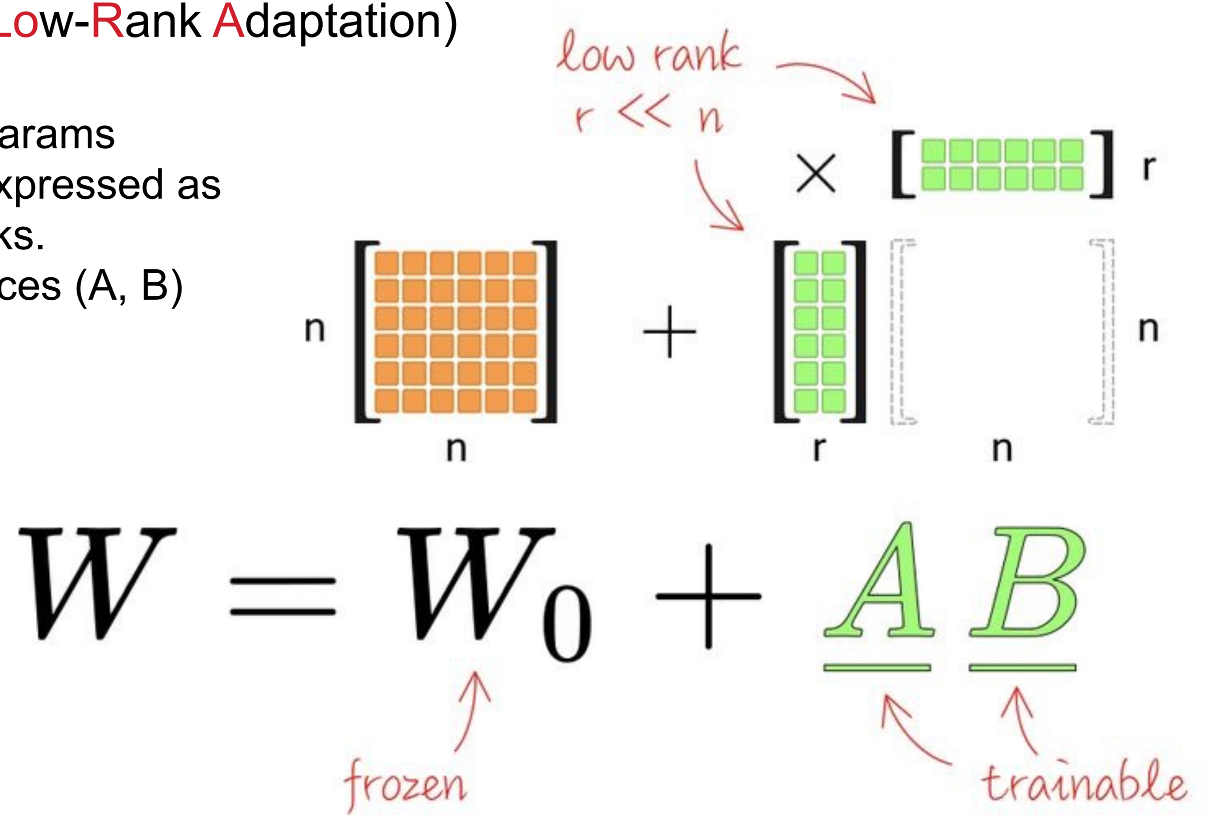
- No need to fine-tune all params
- Weight updates can be expressed as decompositions of lower ranks.
- Train only the small matrices (A, B) while fine-tuning



Source: https://onmine.io/are-you-still-using-lora-to-fine-tune-your-llm/

Fine tuning using LoRA (Low-Rank Adaptation)

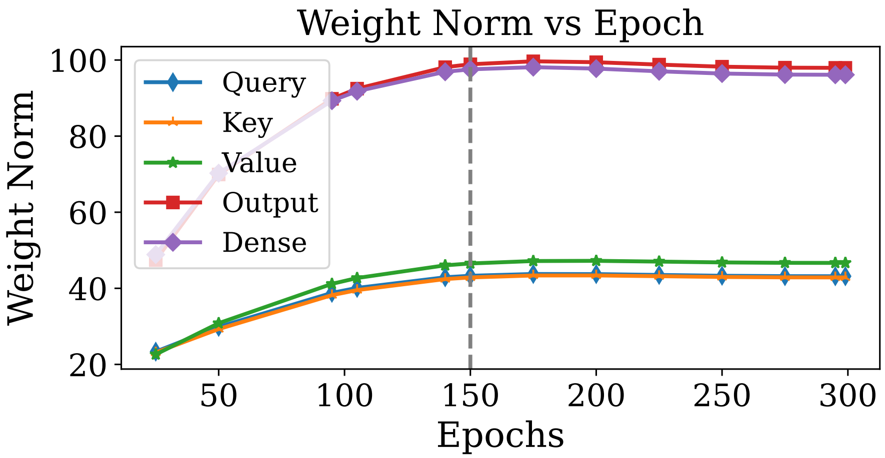
- No need to fine-tune all params
- Weight updates can be expressed as decompositions of lower ranks.
- Train only the small matrices (A, B) while fine-tuning

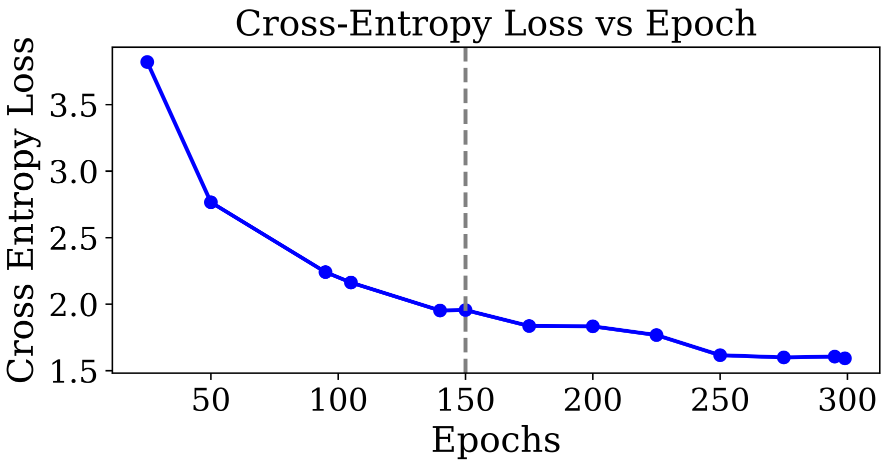


Can we also use this during pre-training?

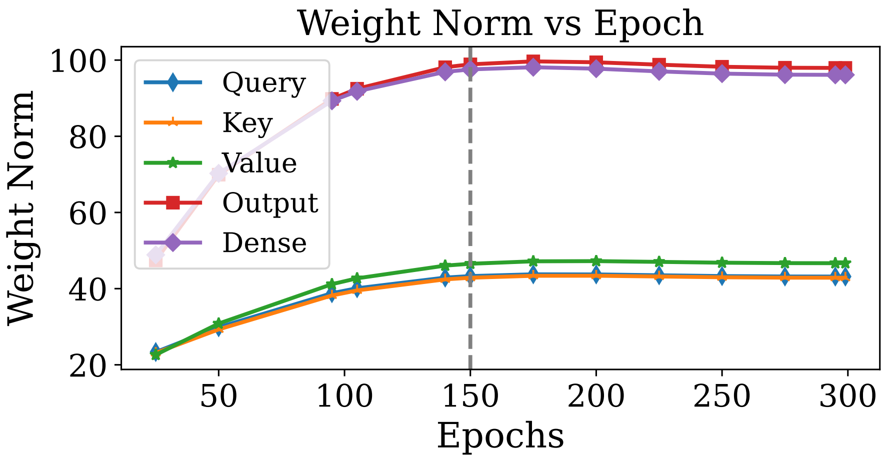
Source: https://onmine.io/are-you-still-using-lora-to-fine-tune-your-llm/

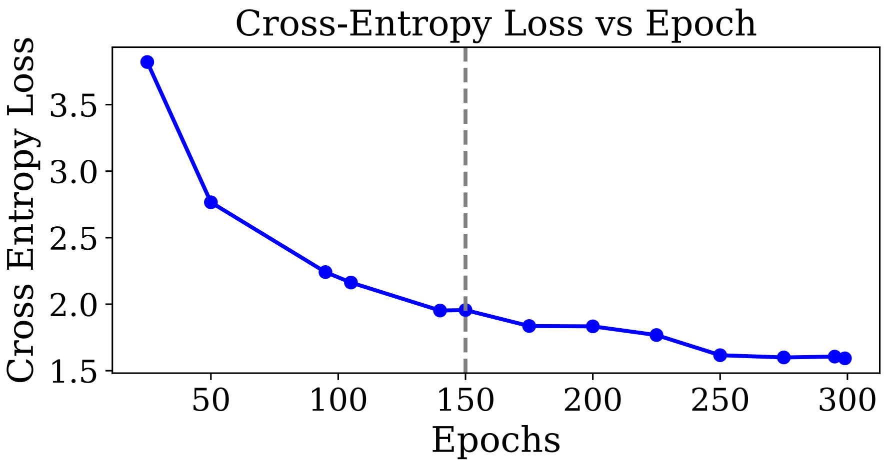
Typical Pre-training run (ViT-Large on ImageNet-1k)





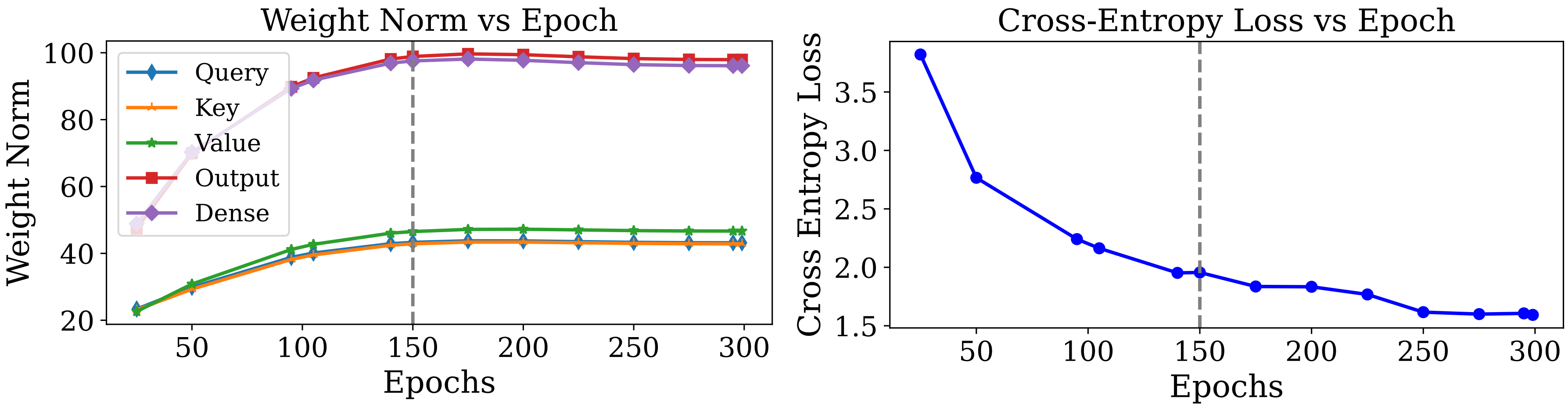
Typical Pre-training run (ViT-Large on ImageNet-1k)





Observation: Around epoch 150,
- the weights start to stabilize
- Loss continues to go down

Typical Pre-training run (ViT-Large on ImageNet-1k)



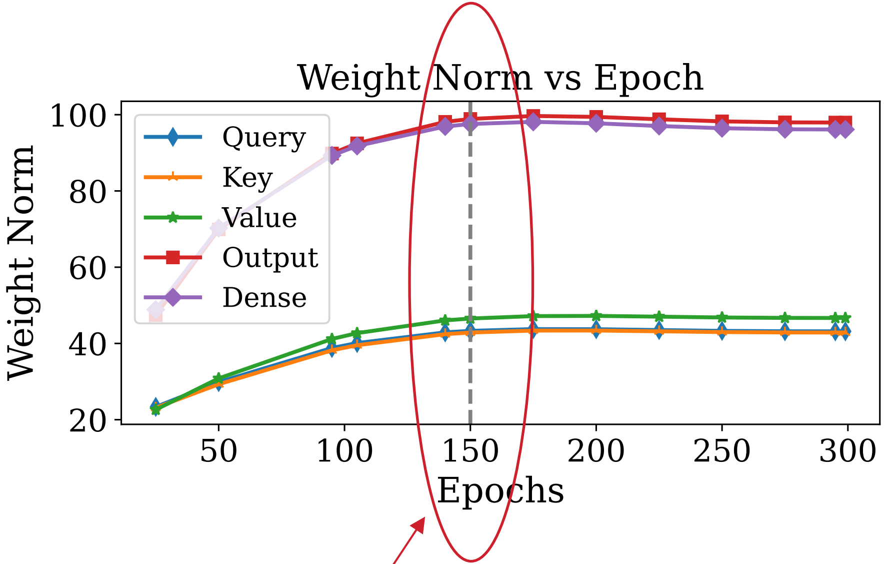
Observation: Around epoch 150,

- the weights start to stabilize
- Loss continues to go down

Question: Can we use LoRA to capture these smaller updates?

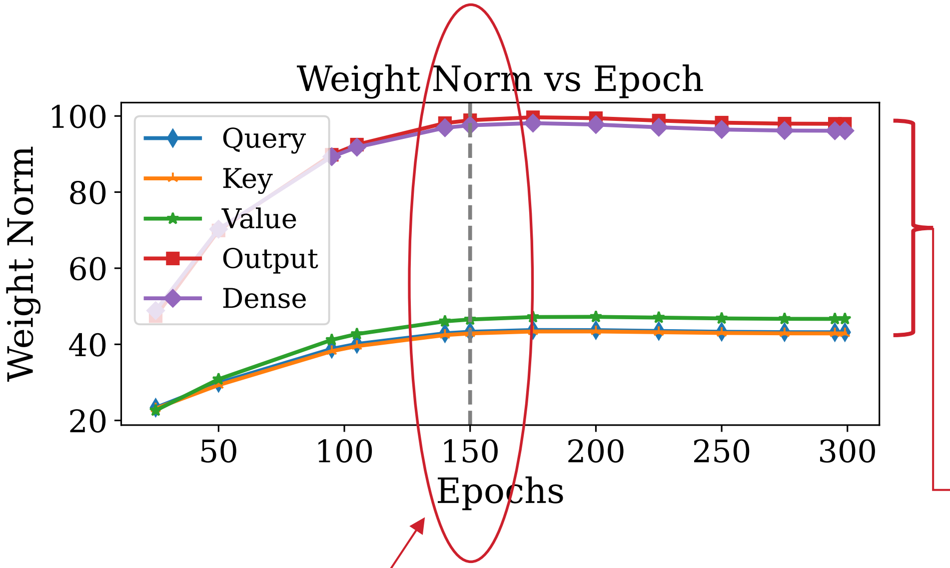
Challenges of straightforward integration:

Challenges of straightforward integration:



We don't know **when** to switch to LoRA apriori

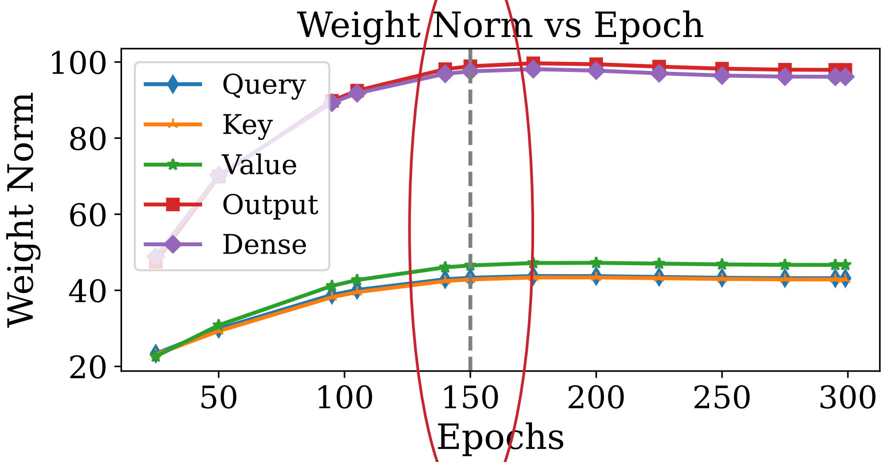
Challenges of straightforward integration:

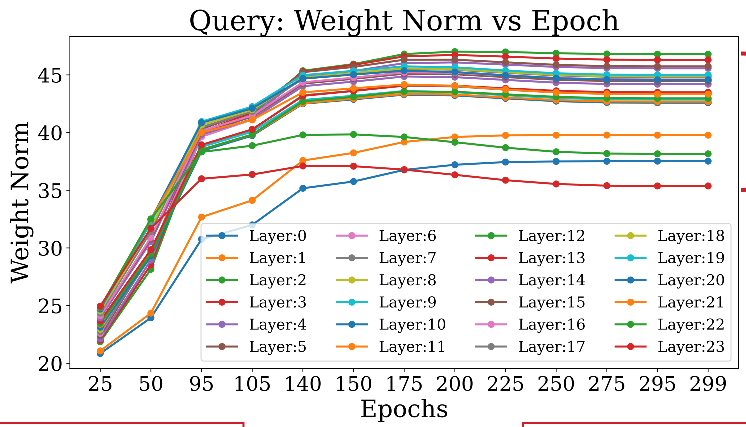


We don't know when to switch to LoRA apriori

- Scale of weights different across modules

Challenges of straightforward integration:

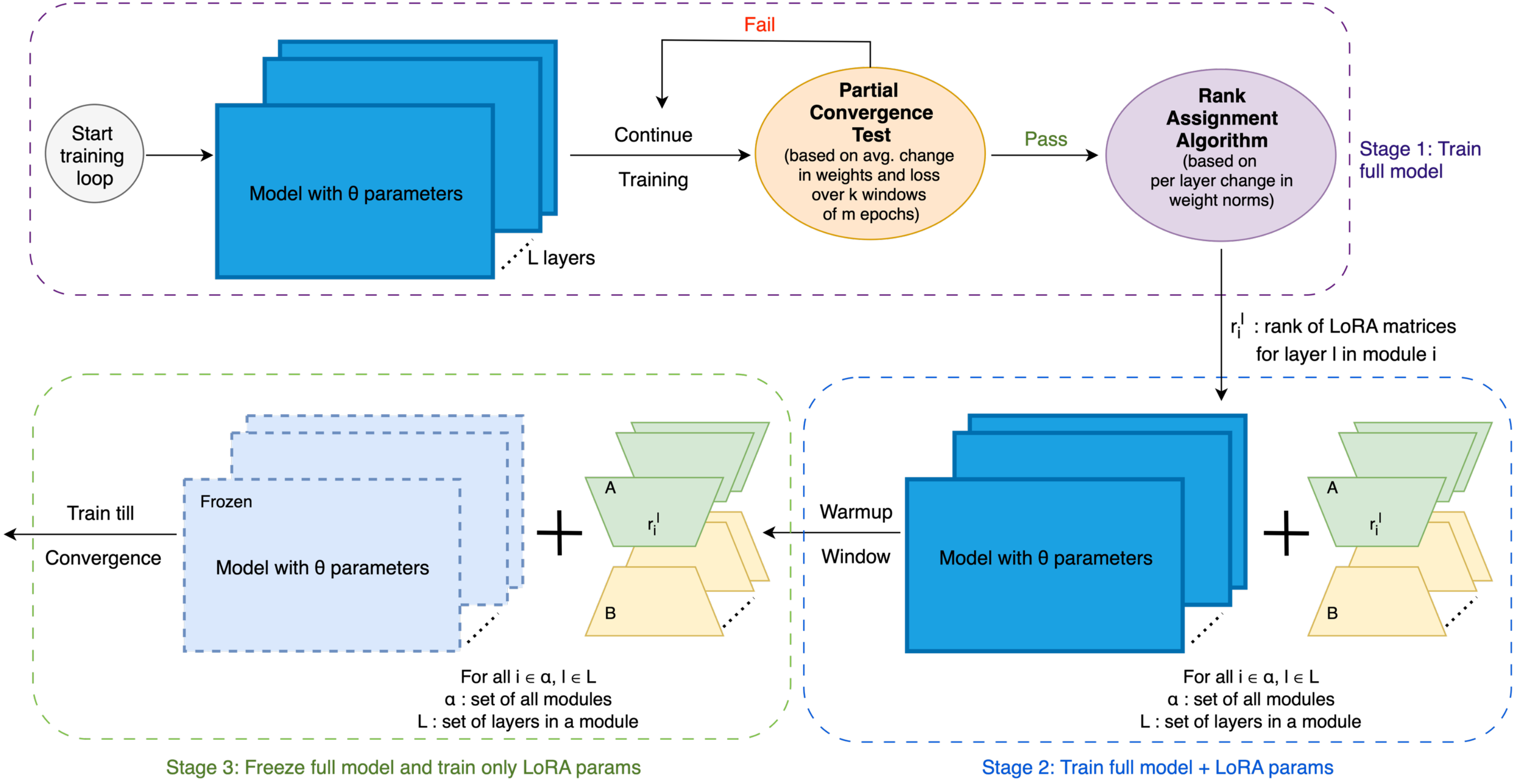




We don't know when to switch to LoRA apriori

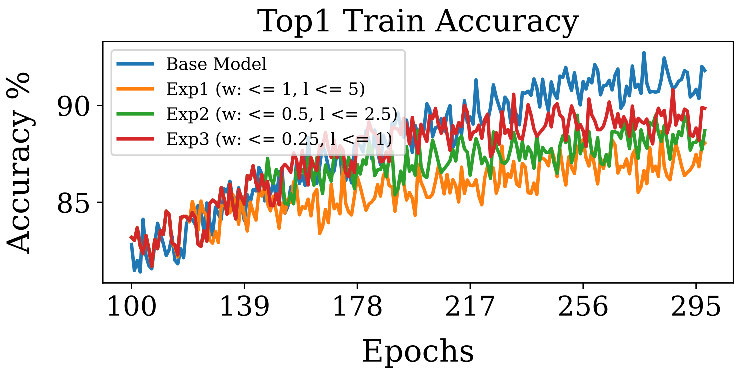
- Scale of weights different across modules
- And also across layers inside a module

Framework

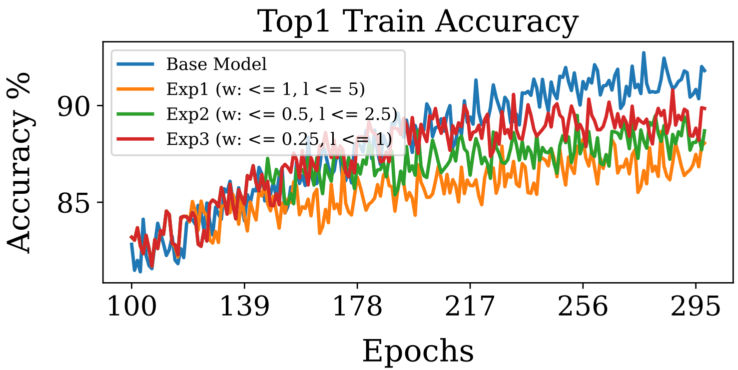


Fail
Start training loop
Model with θ parameters
L layers
Continue
Training
Partial Convergence Test
(based on avg. change in weights and loss over k windows of m epochs)
Pass
Rank Assignment Algorithm
(based on per layer change in weight norms)
Stage 1: Train full model
r_i^l : rank of LoRA matrices for layer l in module i
Train till
Convergence
Frozen
Model with θ parameters
A
r_i^l
B
For all i ∈ α, l ∈ L
α : set of all modules
L : set of layers in a module
Warmup
Window
Model with θ parameters
A
r_i^l
B
For all i ∈ α, l ∈ L
α : set of all modules
L : set of layers in a module
Stage 3: Freeze full model and train only LoRA params
Stage 2: Train full model + LoRA params

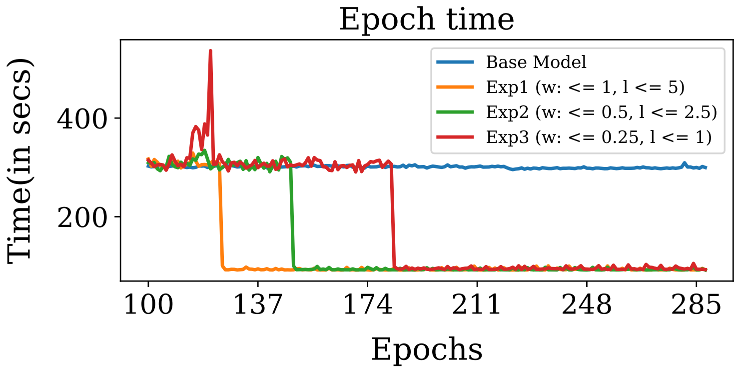
Results


Top1 Train Accuracy
Base Model
Exp1 (w: <= 1, l <= 5)
Exp2 (w: <= 0.5, l <= 2.5)
Exp3 (w: <= 0.25, l <= 1)
Accuracy %
90
85
100
139
178
217
256
295
Epochs

Results

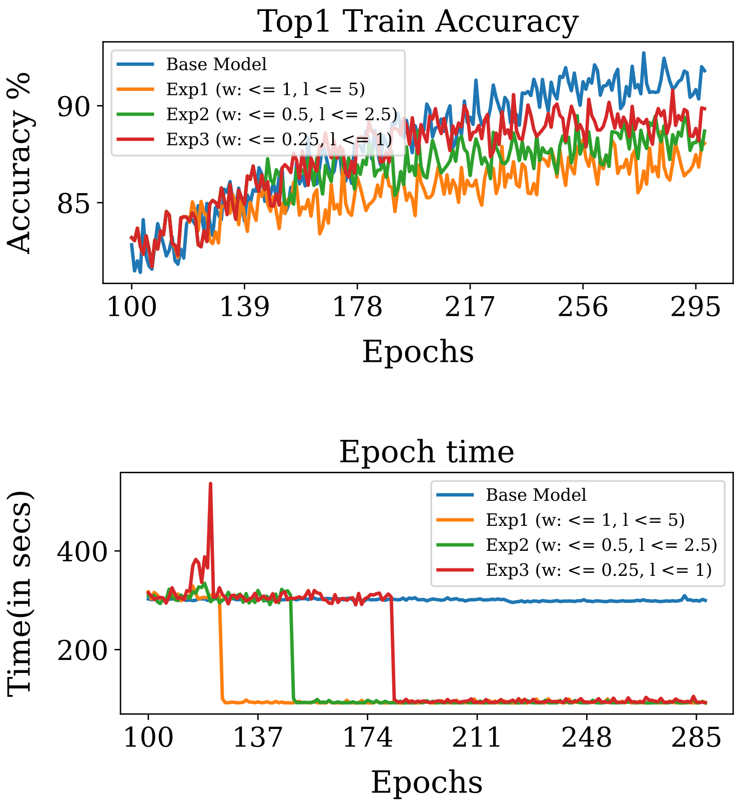


Top1 Train Accuracy
Base Model
Exp1 (w: <= 1, l <= 5)
Exp2 (w: <= 0.5, l <= 2.5)
Exp3 (w: <= 0.25, l <= 1)
Accuracy %
90
85
100
139
178
217
256
295
Epochs

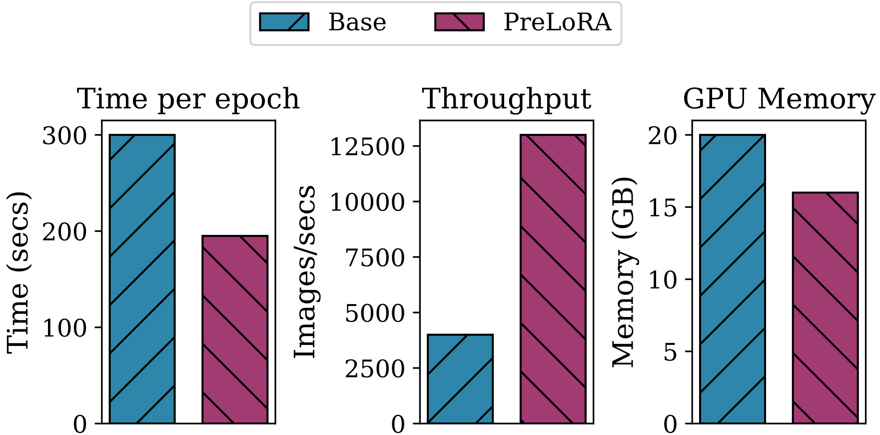


Epoch time
Base Model
Exp1 (w: <= 1, l <= 5)
Exp2 (w: <= 0.5, l <= 2.5)
Exp3 (w: <= 0.25, l <= 1)
Time(in secs)
400
200
100
137
174
211
248
285
Epochs

Results



Top1 Train Accuracy
Base Model
Exp1 (w: <= 1, l <= 5)
Exp2 (w: <= 0.5, l <= 2.5)
Exp3 (w: <= 0.25, l <= 1)
Accuracy %
90
85
100
139
178
217
256
295
Epochs
Epoch time
Base Model
Exp1 (w: <= 1, l <= 5)
Exp2 (w: <= 0.5, l <= 2.5)
Exp3 (w: <= 0.25, l <= 1)
Time(in secs)
400
200
100
137
174
211
248
285
Epochs



Base
PreLoRA
Time per epoch
Time (secs)
300
200
100
0
Throughput
Images/secs
12500
10000
7500
5000
2500
0
GPU Memory
Memory (GB)
20
15
10
5
0

BASED ON:

PRELORA: HYBRID PRE-TRAINING OF VISION TRANSFORMERS WITH FULL TRAINING AND LOW-RANK ADAPTERS [ARXIV:2509.21619]

Krishu Thapa

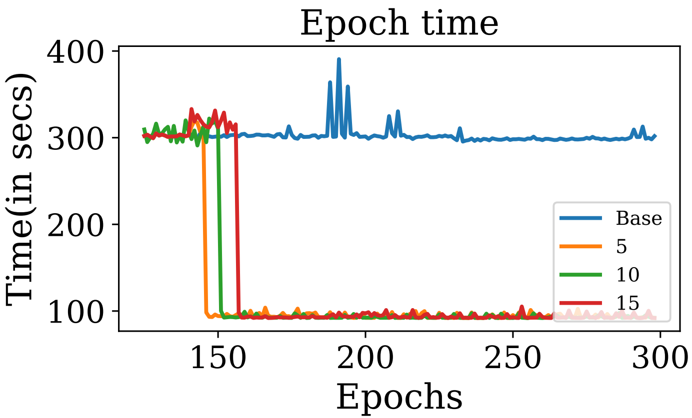
Dr. Murali Emani

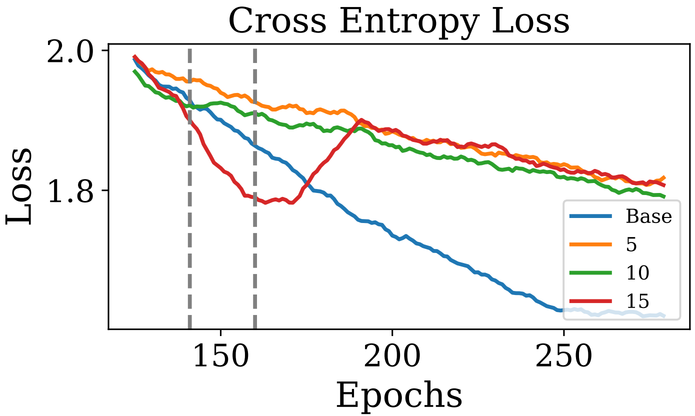
Dr. Venkat Vishwanath

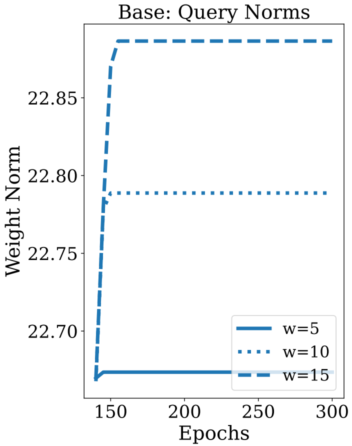
Argonne NATIONAL LABORATORY | ALCF

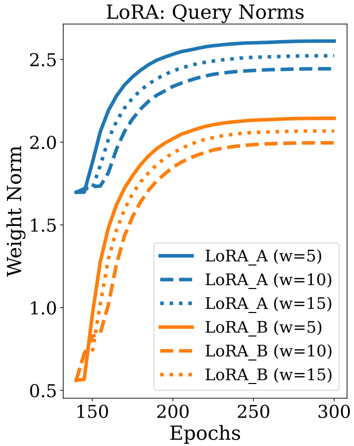
THANK YOU!! QUESTION?

Supplementary: Warmup Window









Supplementary: Algorithms

Algorithm 1 Partial Convergence Test

Input: Number of windows k, window size m, thresholds τ, ζ, module set α. Weight norms $\{W_t^a\}$ for any module $a \in \alpha$ averaged across all layers, and losses $\{L_t\}$ at window $t \in k$.
Output: **True** if converged, **False** otherwise
for each module $a \in \alpha$ **do**
  for $t = 2$ to k **do**
    Compute $\Delta W_t^a = \frac{\|W_t^a\| - \|W_{t-1}^a\|}{\|W_{t-1}^a\|} \times 100$
    Compute $\Delta L_t = \frac{L_t - L_{t-1}}{L_{t-1}} \times 100$
    if $|\Delta W_t^a| > \tau$ or $|\Delta L_t| > \zeta$ **then**
      return False
    end if
  end for
end for
return True

Algorithm 2 Rank Assignment Algorithm

Input: Minimum rank r_{min}, maximum rank r_{max}, weight norm changes $\Delta W_k^{a_l}$ for all modules $a \in \alpha$ and layers $l \in L$
Output: Layer-to-rank assignment function $\mathcal{A} : a_l \mapsto r$
Initialize rank set $\mathcal{R} \leftarrow []$
for $p = \log_2(r_{\text{min}})$ to $\log_2(r_{\text{max}})$ **do**
  Append 2^p to $\mathcal{R}$
end for
Initialize empty mapping $\mathcal{A} \leftarrow \{\}$
for each module $a \in \alpha$ **do**
  Collect weight norm changes for the module as, $changes \leftarrow [\Delta W_k^{a_l} \quad \forall l \in L]$
  min-max-norm($changes$) $\rightarrow \mathcal{N}_a \in [0,1]$
  for each layer $l \in L$ with normalized value $v \in \mathcal{N}_a$ **do**
    if $v \neq 0$ **then**
      $i \leftarrow \lceil v \cdot |\mathcal{R}| \rceil - 1$
    else
      $i \leftarrow \lceil v \cdot |\mathcal{R}| \rceil$
    end if
    Assign layer l of module a to rank: $\mathcal{A}[a_l] \leftarrow \mathcal{R}[i]$
  end for
end for
return $\mathcal{A}$