

TECHNICAL INFOGRAPHIC · OCTOBER 2026

LoRA

Low-Rank Adaptation

Origin, Foundations, Variants, Engineering and Adoption

How a low-rank W_0 + BA update made fine-tuning large models cheap, fast and modular

Aridio Silva · Independent Researcher
@aridiosilva · Verified primary sources



What LoRA is

Freeze the model and train only a low-rank update: $W = W_0 + BA$

≈10,000×

fewer trainable parameters on
GPT-3 175B

3×

less GPU memory in training
(1.2 TB → 350 GB)

0

added inference latency after
merging the adapter



Why it matters

Made fine-tuning giant models affordable and became the default customization method.

- **Who and when:** Edward J. Hu, Yelong Shen and six other Microsoft researchers; arXiv:2106.09685 (June 17, 2021), ICLR 2022.

- **Idea:** the weight change needed to adapt a model has a low intrinsic rank, so it can be learned as the product of two narrow matrices.

- **Result:** quality on par with or better than full fine-tuning on RoBERTa, DeBERTa, GPT-2 and GPT-3.

- **Today:** the market standard for fine-tuning LLMs and image models.



Positive implications

Far lower cost, memory and storage; small, swappable adapters.



Negative implications

Capacity is capped by the rank; rank, alpha and learning rate need careful tuning.

Why LoRA was created

Adapting a giant model to each task cost as much as copying it whole

350 GB

per fine-tuned GPT-3 175B
checkpoint

35 TB

to keep 100 fine-tuned versions

1.2 TB

of VRAM for full fine-tuning
with Adam

Adapters (Houlsby et al., 2019)

Extra layers in series add latency: +20.7% to +30.3% on GPT-2 medium at batch size 1 (measured by Hu et al.).

Prefix and prompt tuning (2021)

Hard to optimize, non-monotonic performance; the vectors use up part of the context length.

Requirements for a new method: few parameters • full fine-tuning quality • zero latency • fast task switching



Why it matters

Explains the economic problem LoRA solves: storage, memory and deployment at scale.



Positive implications

Set clear, measurable requirements for efficient methods.



Negative implications

Methods of the time forced a trade-off between efficiency, latency and quality.

Origin: intrinsic dimension and authors

LoRA brought a 2018–2020 theoretical finding inside every layer

1

2018 • Li, Farkhoor, Liu, Yosinski

Measure intrinsic dimension: the smallest subspace where training still solves the problem (ICLR 2018).

2

2020 • Aghajanyan, Gupta, Zettlemoyer

200 projected parameters reach 90% of full RoBERTa performance on MRPC (ACL 2021).

3

Jun 2021 • Hu, Shen et al. (Microsoft)

Intrinsic-rank hypothesis: $\Delta W = BA$ per layer.
arXiv:2106.09685.

4

2022 • ICLR 2022

Peer-reviewed publication; loralib code at github.com/microsoft/LoRA.

Authors of the original paper

- Edward J. Hu*
- Yelong Shen*
- Phillip Wallis
- Zeyuan Allen-Zhu
- Yanzhi Li
- Shean Wang
- Lu Wang
- Weizhu Chen

* equal contribution • Microsoft Corporation

!

Why it matters

Shows the method grew from solid theory, not an empirical trick.

+

Positive implications

Explains why few parameters suffice: larger models tend to have lower intrinsic dimension.

-

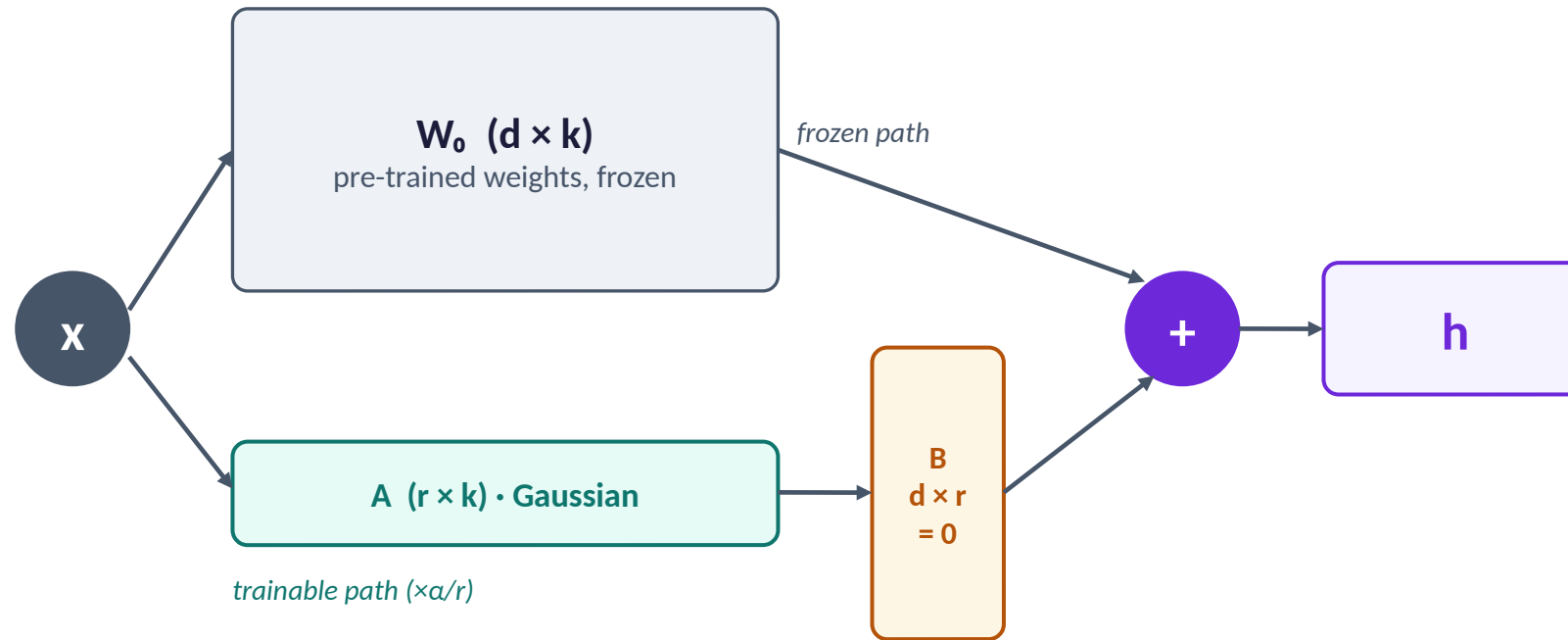
Negative implications

The low-rank hypothesis does not hold for every task; it breaks down in pre-training-like regimes.

Note: LoRA (AI) is unrelated to LoRa, the long-range radio technology for IoT.

How it works: the $W_0 + BA$ decomposition

The original weight stays frozen; only the narrow matrices A and B get gradients



$$h = W_0 x + (\alpha/r) \cdot B A x \quad \text{with } r \ll \min(d, k)$$

$B = 0$ at the start $\rightarrow$ the model begins identical to the pre-trained one. After training, $W = W_0 + (\alpha/r) \cdot BA$ is merged and inference runs with no extra layers. In the original paper, α = the first r tried.

! Why it matters

The core of the method: a simple linear reparameterization that fits any dense layer.

+ Positive implications

Linearity allows merging: zero latency, and task switching by subtracting BA and adding $B'A'$.

- Negative implications

With merged weights, one batch cannot serve different tasks without dedicated infrastructure.

How much you save

Trainable parameters per matrix drop from $d \times k$ to $r \times (d + k)$

Approach (4,096 × 4,096 matrix)	Trainable parameters	% of original
Full fine-tuning	16,777,216	100%
LoRA with r = 16	131,072	0.78%
LoRA with r = 8	65,536	0.39%

350 GB → 35 MB

GPT-3 175B checkpoint with r = 4 on W_q and W_v (Hu et al.)

+25%

training throughput on GPT-3: 43.1 vs. 32.5 tokens/s per V100

4,096 × 4,096 example: a typical attention projection size in ~7B models; illustrative arithmetic.

! Why it matters

Quantifies why LoRA changed the economics of LLM fine-tuning.

+ Positive implications

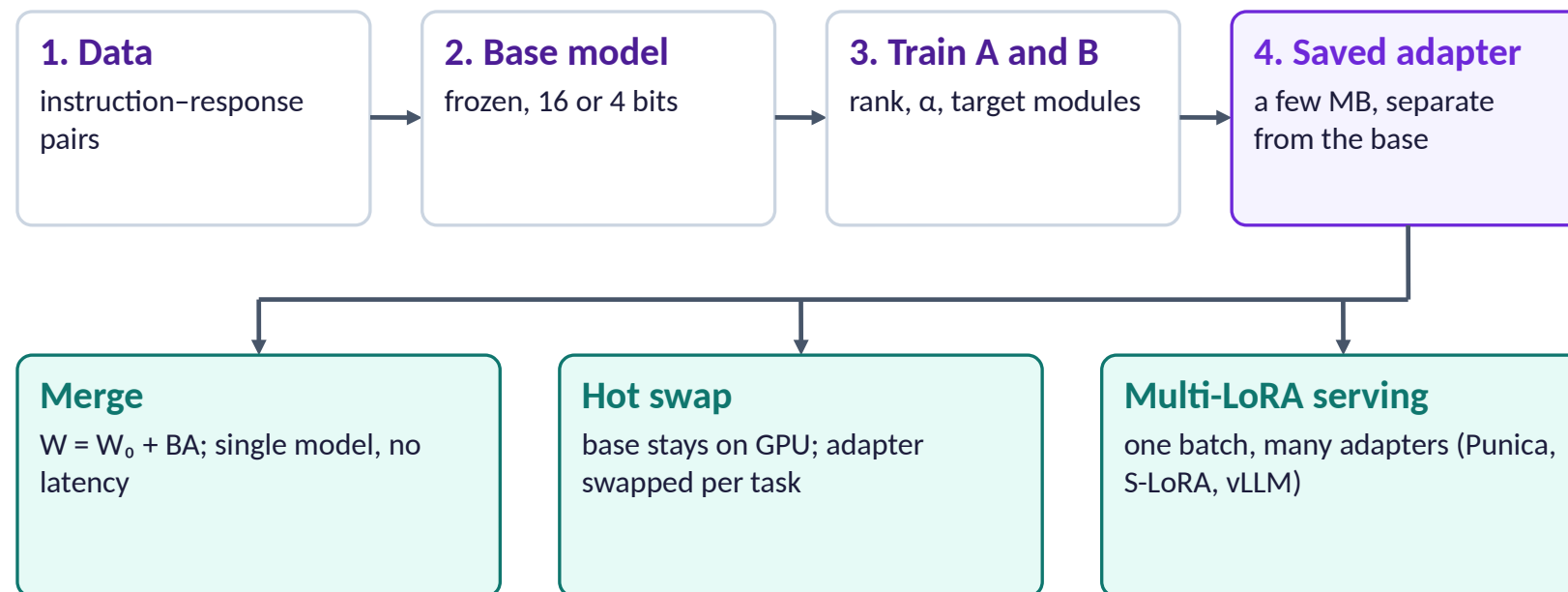
Fewer GPUs, less storage and fewer I/O bottlenecks; 100 tasks ≈ 354 GB instead of 35 TB.

- Negative implications

The base model still sits fully in memory; savings are in training and per-task storage.

Engineering workflow

Train only the adapter; deployment picks one of three paths



Merge

$W = W_0 + BA$; single model, no latency

Hot swap

base stays on GPU; adapter swapped per task

Multi-LoRA serving

one batch, many adapters (Punica, S-LoRA, vLLM)

```
from peft import LoraConfig, get_peft_model
config = LoraConfig(r=16, lora_alpha=32, lora_dropout=0.05,
                    target_modules="all-linear", task_type="CAUSAL_LM")
model = get_peft_model(base_model, config) # illustrative values
```



Why it matters

Separates cheap training from flexible deployment suited to each scenario.



Positive implications

Versionable, reversible adapters; one base model serves many tasks.



Negative implications

An adapter only works on the exact base it was trained on; changing the base means retraining.

Multi-adapter serving at scale

Compute the base model once per batch, then add the right adapter per request

Punica

Chen et al. • MLSys 2024

SGMV CUDA kernel batches operations from different LoRA models together.

12× throughput • +2 ms/token

S-LoRA

Sheng et al. • 2023

Adapters kept in main memory; only active ones go to the GPU, with Unified Paging.

up to 4× throughput • thousands of adapters

vLLM / SGLang

open source

Inference servers accept many adapters on the same base (max_loras) and dynamic loading.

production standard



Why it matters

Solved the limitation flagged in the original paper: batches mixing different tasks.



Positive implications

Per-customer or per-user customization with a single copy of the model on the GPU.



Negative implications

Needs specialized kernels and servers; figures are the authors' own, under lab conditions.

What LoRA solves, and how

Each problem in adapting large models maps to a specific mechanism

Problem	How LoRA solves it	Evidence (Hu et al., 2021)
GPU memory in training	Frozen base: no gradients or optimizer states	1.2 TB → 350 GB (GPT-3)
Storage per task	Only the A, B pair is saved	350 GB → 35 MB
Task switching	Swap the adapter, not the model	Fast, lightweight operation
Latency of classic adapters	Linear, mergeable update	Zero added latency
Lower quality of efficient methods	Low rank is expressive enough	≥ full fine-tuning on GLUE and GPT-3
Training speed	Fewer gradients to compute	+25% throughput

! Why it matters

Summarizes the method's value in business and engineering terms.

+ Positive implications

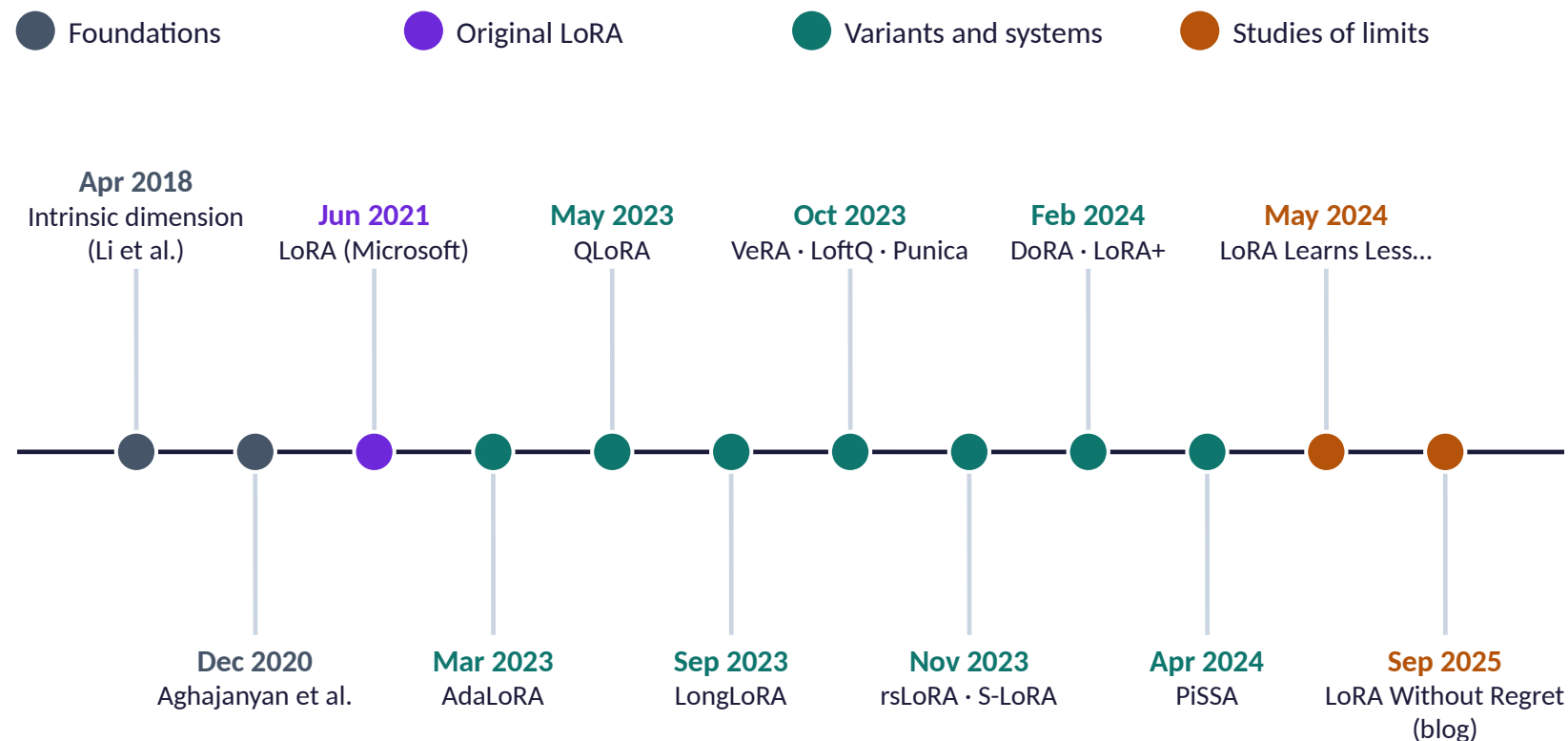
Combines efficiency, quality and zero latency, which no earlier method managed at once.

- Negative implications

Original evidence comes from 2021 models; results vary by task and configuration.

Timeline: LoRA, variants and studies

Date of first arXiv version (unless noted); spacing is not proportional to time



! Why it matters

Shows the research boom: most major variants appeared within about a year (2023–2024).

+ Positive implications

Mature ecosystem: variants for memory, scale, initialization, context and serving.

- Negative implications

Many variants compete; reported gains do not always replicate under a uniform methodology.

Variants I: memory, budget and scale

Answers to memory cost and to how parameters are allocated

Variant • venue	Core idea	Reported result
QLoRA • NeurIPS 2023 Dettmers et al. (U. Washington)	4-bit base (NF4), double quantization and paged optimizers	65B model fine-tuned on one 48 GB GPU
LoftQ • ICLR 2024 Li et al. (Georgia Tech, Microsoft)	Quantizes and picks the LoRA initialization jointly	Gains mostly at 2 bits and mixed 2/4-bit precision
VeRA • ICLR 2024 Kopiczko et al. (U. Amsterdam, Qualcomm)	Shared random matrices; trains only scaling vectors	10× fewer parameters; Llama 2 7B with 1.4 million
AdaLoRA • ICLR 2023 Zhang et al.	SVD-form update; prunes unimportant singular values	Best under low parameter budgets
LongLoRA • ICLR 2024 Chen et al. (CUHK, MIT)	Shifted sparse attention (S^2 -Attn) during training only	Llama 2 7B up to 100k tokens; 70B up to 32k on 8 A100s



Why it matters

QLoRA democratized fine-tuning large models on a single GPU.



Positive implications

Feasible fine-tuning on modest hardware; long context and tiny adapters.



Negative implications

Aggressive quantization can hurt quality without proper initialization (the reason for LoftQ).

Variants II: optimization and initialization

Training tweaks that improve LoRA at no extra inference cost

Variant · venue	Limitation addressed	Idea and reported result
rsLoRA · 2023 Kalajdzievski	High ranks learn slowly with α/r	$\alpha/\sqrt{r}$ scaling: higher rank now pays off, at no inference cost
DoRA · ICML 2024 (oral) Liu et al. (NVIDIA, HKUST)	Gap versus full fine-tuning	Splits magnitude and direction; beats LoRA on LLaMA, LLaVA and VL-BART
LoRA+ · ICML 2024 Hayou, Ghosh, Yu (Berkeley)	Same learning rate for A and B is suboptimal	Different rates: +1% to 2% and up to ~2× faster training
PiSSA · NeurIPS 2024 Meng, Wang, Zhang (Peking U.)	Slow convergence from random init	SVD of W initializes A and B; GSM8K 72.86% vs. 67.7% (Mistral-7B)

Results as reported by each paper's authors under different conditions; rows are not directly comparable.



Why it matters

Shows that details of scaling, learning rate and initialization still yield real gains.



Positive implications

Simple changes (one scale factor, two rates, another init) with zero inference cost.



Negative implications

More hyperparameters and choices; benefits depend on the task and do not always reproduce.

Benchmark results (2021–2023)

LoRA matched or beat full fine-tuning with a tiny fraction of parameters

Study	Model and task	Reported result
LoRA (2021)	RoBERTa base · GLUE	Average 87.2 with 0.3M parameters vs. 86.4 for full fine-tuning (125M)
LoRA (2021)	DeBERTa XXL (1.5B) · GLUE	Average 91.3 with 4.7M parameters vs. 91.1 for full fine-tuning
QLoRA (2023)	LLaMA up to 65B · Guanaco chatbot	99.3% of ChatGPT on the Vicuna benchmark after 24 h on a single GPU
LongLoRA (2023)	Llama 2 7B and 70B · long context	4k → 100k tokens (7B); 32k (70B) on one 8×A100 machine
VeRA (2023)	Llama 2 7B · instructions	Performance on par with LoRA using 1.4M trainable parameters



Why it matters

The empirical evidence behind mass adoption from 2023 onward.



Positive implications

Competitive quality at far lower cost, even for models with tens of billions of parameters.



Negative implications

Benchmarks of the time (GLUE, Vicuna) are limited; QLoRA relied on GPT-4-based evaluation.

2024 results: gains and limits

Variants improved LoRA, while one study showed where it falls short

Study	Model and task	Reported result
PiSSA	Mistral-7B and LLaMA-3-70B · GSM8K	72.86% vs. 67.7% for LoRA; QPiSSA 86.05% vs. 81.73% for QLoRA
LoRA+	Several models and tasks	+1% to 2% performance; up to ~2× faster training
Biderman et al. (TMLR 2024; ICLR 2025)	Llama-2-7B · code and math	Well below full fine-tuning in continued pre-training (~20B tokens), but forgets less

10–100×

rank of full fine-tuning perturbations vs. typical LoRA (Biderman et al.)

Forgets less

preserves the source domain better than weight decay and dropout

! Why it matters

Defines the method's reach: great for adapting, weak at absorbing huge amounts of new knowledge.

+ Positive implications

LoRA acts as a regularizer: it preserves out-of-domain skills and output diversity.

- Negative implications

In code and math continued pre-training, it trails full fine-tuning even at high rank.

LoRA Without Regret (Thinking Machines, 2025)

A study led by John Schulman mapped the “low-regret regime”

1 Small/medium SFT

Same performance as full fine-tuning on instruction and reasoning datasets.

2 All layers

Applying it to MLP and MoE is essential; attention-only underperforms even at equal parameters.

! Why it matters

Turned LoRA from a “cheap compromise” into a no-loss choice for most post-training.

3 RL with rank 1

In policy-gradient RL, even rank 1 matches full fine-tuning (~1 bit per episode).

4 Learning rate

Optimal $\approx 10\times$ the full fine-tuning rate, in both SFT and RL.

+ Positive implications

Practical recipe: all layers, $\sim 10\times$ LR, moderate batches; ideal for RL-based alignment.

5 Large batches

Larger penalty than full fine-tuning; raising the rank does not fix it.

6 Cost per pass

Slightly over 2/3 of full fine-tuning FLOPs.

- Negative implications

When data exceeds adapter capacity, learning slows; published as a technical blog post.

Competitors I: LoRA's predecessors

Adapters and prompt-based methods paved the way, with limitations

Method · origin	What it trains	Added latency	Strength / weakness
Adapters Houlsby et al., ICML 2019	Bottleneck networks between layers	Yes (+20.7% to 30.3%, batch 1)	Modular / sequential processing
Prefix tuning Li & Liang, ACL 2021	Prefix vectors in each layer	Small; uses context	Few parameters / unstable optimization
Prompt tuning Lester et al., EMNLP 2021	Virtual token embeddings	Small; uses context	Lightweight / competitive only at large scale
BitFit Ben Zaken et al., ACL 2022	Bias vectors only	None	Simple / GLUE 85.2 vs. 87.2 for LoRA

! Why it matters

Positions LoRA as a synthesis of the best earlier ideas.

+ Positive implications

Prompt methods and BitFit remain useful when the parameter budget is minimal.

- Negative implications

Adapter latency and prompts' context cost limit their use in production.

Competitors II: current alternatives

Each alternative wins on one axis; LoRA combines the most advantages

Method · origin	What it trains	Strength	Weakness
Full fine-tuning	All weights	Quality ceiling, including pre-training	A whole model per task
(IA) ³ Liu et al., NeurIPS 2022	Vectors that rescale activations	Far fewer parameters; strong few-shot	Lower capacity
GaLore Zhao et al., ICML 2024	All weights, projected gradients	Up to 65.5% less optimizer memory; 7B on a 24 GB GPU	No swappable adapter
LoRA and variants Hu et al., ICLR 2022	Matrices A and B	Quality, zero latency, modularity, ecosystem	Trails near pre-training

 Why it matters

Helps choose the right method for each goal and budget.

 Positive implications

GaLore extends low rank to pre-training; (IA)³ and VeRA suit thousands of adapters.

 Negative implications

No alternative wins on every axis; the choice depends on data, cost and deployment.

Training-free alternatives: RAG and in-context learning

Not every adaptation needs fine-tuning

LoRA

Teaches behavior, format, style and skills in a stable way.

When to use: When the task repeats and training examples exist.

RAG

Retrieves documents and inserts them into the context, without changing weights.

When to use: When facts change often or need a source.

In-context learning

Examples in the prompt itself (few-shot).

When to use: Simple tasks, prototypes and low volume.

! Why it matters

Avoids fine-tuning where retrieval or a prompt would do.

+ Positive implications

LoRA + RAG lowers cost and keeps answers current and traceable.

- Negative implications

Fine-tuning volatile facts into LoRA ages fast and requires frequent retraining.

In practice, they are combined: LoRA for behavior, RAG for facts.

Adoption across the AI market

Use stated in the organizations' own official sources

Organization	Stated use of LoRA
Microsoft · Azure OpenAI	Fine-tuning service states it uses LoRA, training a smaller subset of parameters
Apple · Apple Intelligence (2024)	~3B on-device model with per-feature adapters; rank 16 ≈ tens of MB
Thinking Machines · Tinker (2025)	LoRA-based fine-tuning API for open models; many runs share GPUs
Hugging Face	PEFT library (LoRA, AdaLoRA, DoRA, VeRA); LoRA for diffusion (diffusers) since Jan 2023
vLLM and SGLang	Inference servers batching many adapters over the same base
NVIDIA Research	Created and published the DoRA variant (ICML 2024)
Community (Unsloth, Kaggle)	LoRA/QLoRA on a single GPU; e.g., the Gemma 4 Good hackathon (2026)

! Why it matters

Confirms LoRA left the lab and became product infrastructure.

+ Positive implications

Cloud providers, device makers and startups have standardized on it.

- Negative implications

Reliance on a single standard; internal details (rank, layers) are rarely disclosed.

Where it is used and who benefits

From cloud to phone, from lab to the image community

- 1 Language models**
Instructions, domains (legal, medical, financial), style, structured output, RL alignment
- 2 Image generation**
Styles, characters and objects in Stable Diffusion; pioneer: Simo Ryu (@cloneofsimo)
- 3 Edge devices**
Specialized local models, as in Apple Intelligence
- 4 SaaS platforms**
One adapter per customer or task on a shared base
- 5 Research and teaching**
Post-training and RL on a limited budget; competitions and prototypes

! Why it matters

Shows how broadly the method applies and who captures value from it.

+ Positive implications

Benefits firms with sensitive data, small teams, SaaS providers and end users.

- Negative implications

Ease of creating and sharing adapters raises misuse and provenance risks.

Pros

Advantages backed by primary sources

Advantage	Evidence
Much lower training memory	1.2 TB → 350 GB (GPT-3); 65B on one 48 GB GPU (QLoRA)
Minimal storage per task	350 GB → 35 MB (GPT-3); tens of MB per adapter (Apple)
No added latency	Linear merge into the original weight
Full fine-tuning quality in post-training	GLUE (2021); Thinking Machines (2025)
Less forgetting	Biderman et al. (2024)
Lower cost per pass	Slightly over 2/3 of full fine-tuning FLOPs
Modularity and multi-tenant serving	Punica, S-LoRA, vLLM

 Why it matters

Sums up why LoRA is the default customization choice in 2026.

 Positive implications

Cost, speed and modularity lower the barrier to entry for any organization.

 Negative implications

The advantages depend on correct setup: all layers, suitable rank and learning rate.

Cons and limitations

Where LoRA needs care or loses to alternatives

Limitation	Evidence or explanation
Learns less near pre-training	Continued pre-training on code and math (Biderman et al.)
Capacity capped by rank	Data beyond capacity slows learning
Hyperparameter-sensitive	Optimal LR $\approx 10\times$ full fine-tuning; α/r vs. $\alpha/\sqrt{r}$ scaling
Penalty with large batches	Gap grows with batch size; rank does not fix it
Original setup insufficient	Attention-only is not enough; cover all linear layers
Mixed batches need infrastructure	Dedicated multi-adapter kernels and servers
Tied to the base model	Changing the base means retraining the adapter



Why it matters

Sets realistic expectations and signals when to choose another approach.



Positive implications

Limitations are known and measurable, with practical mitigation recipes.



Negative implications

To absorb large amounts of new knowledge, full fine-tuning or GaLore remain preferable.

Governance, security (GRC) and beneficiaries

An adapter is a small artifact that deserves its own controls

Recommended controls

- Record each adapter's origin, version and base model
- Verify integrity (hash, signature) before loading
- Restrict who can publish adapters to production
- Treat adapters as software supply-chain components
- Use the base/adapter split for audit and fast rollback

Who benefits

- Organizations with sensitive data, tuning models on their own infrastructure
- Small teams, researchers and students with a single GPU
- SaaS providers with one adapter per customer
- Device makers running local models
- End users: specialized, cheaper and more private models

! Why it matters

Links the method to governance, audit and information security requirements.

+ Positive implications

The base/adapter split eases review, versioning, revocation and rollback.

- Negative implications

Third-party adapters of unknown provenance can silently change model behavior.

State of the technology (October 2026)

From a cost-quality compromise to the natural choice for post-training

Consolidated state

Matches full fine-tuning in small/medium post-training and RL; established best practices; complete ecosystem (PEFT, Unsloth, TRL, vLLM, SGLang).

Trends

LoRA for RL-based alignment; on-edge tuning of small models; training as a service on shared bases; new initializations.

In products

Azure OpenAI, Apple Intelligence (on device) and Tinker document using LoRA in their products.

Open questions

Predicting when it matches full fine-tuning; explaining the $\sim 10\times$ LR ratio; uniform evaluation of variants; LoRA for distributed MoE.



Why it matters

Places the reader at the state of the art and what to watch next.



Positive implications

Mature technology with validated recipes and broad tool and provider support.



Negative implications

Theoretical gaps remain; new results may change best practices quickly.

Conclusion

LoRA solved the problem that motivated it, and we know where its limits are

Solved

Cheap, fast and modular adaptation of large models, with no inference cost.

Democratized

With QLoRA, models with tens of billions of parameters fit on a single GPU.

Scaled

Variants refined training; Punica, S-LoRA and vLLM enabled thousands of adapters.

Has limits

Trails when a lot of new knowledge must be learned, near pre-training.

Recommendation: apply LoRA to all linear layers for behavior and format; pair it with RAG for facts; prefer full fine-tuning or GaLore to absorb large volumes of knowledge.

Glossary (1/3)

Term	Definition
Adam / AdamW	Optimizers that store moment estimates per trainable parameter; memory grows with trained parameters.
Adapter	Small set of trainable weights attached to a frozen model (in LoRA, the A and B pair).
Alpha (α)	LoRA scaling constant: the update is multiplied by α/r ($\alpha/\sqrt{r}$ in rsLoRA).
Batch	Set of examples or requests processed together.
Catastrophic forgetting	Loss of earlier capabilities after tuning a model to a new task.
Checkpoint	File with the saved weights of a model or adapter.
Continued pre-training	Further pre-training on a large volume of domain text (CPT).
CUDA kernel	Routine run on the GPU; Punica built the SGMV kernel for multi-adapter batches.
Dropout	Regularization that randomly zeroes part of the activations during training.
Fine-tuning	Continued training of a pre-trained model on task- or domain-specific data.
Full fine-tuning	Fine-tuning in which all model weights are updated (FullFT).
GLUE	Standard suite of language-understanding tasks used as a benchmark.

Glossary (2/3)

Term	Definition
GPU / VRAM	Graphics processor used for training and inference / its dedicated memory.
Gradient checkpointing	Recomputes activations during backpropagation to save memory.
Inference	Using the trained model to generate outputs.
Intrinsic dimension	Smallest subspace where training still reaches a good solution (Li et al., 2018).
Intrinsic rank	Hypothesis that the weight change during adaptation has low rank.
Latency	Time between a request and the model's response.
Learning rate (LR)	Step size of each weight update.
LLM	Large Language Model.
Merge	Folding the update into the weight: $W = W_0 + (\alpha/r) \cdot BA$; removes added latency.
MLP	Fully connected (feed-forward) block inside each Transformer layer.
MoE	Mixture of Experts: only some “experts” are activated per token.
NF4	4-bit NormalFloat, QLoRA's 4-bit data type for normally distributed weights.

Glossary (3/3)

Term	Definition
PEFT	Parameter-Efficient Fine-Tuning; also Hugging Face's library.
Quantization	Representing weights with fewer bits (e.g., 4 bits) to save memory.
RAG	Retrieval-Augmented Generation: retrieves documents and inserts them into the context.
Rank (r)	Number of linearly independent columns; in LoRA, the bottleneck width between A and B.
Reinforcement learning	Training guided by rewards assigned to model outputs (RL).
SFT	Supervised Fine-Tuning on instruction-response pairs.
SVD	Singular value decomposition; basis of AdaLoRA and PiSSA.
Throughput	Requests or tokens processed per unit of time.
Token	Unit of text processed by the model, usually part of a word.
Transformer	Attention-based neural architecture underlying today's LLMs.
W_q, W_k, W_v, W_o	Query, key, value and output projections of self-attention.

Validated references (1/3)

Primary sources consulted in October 2026; live links

Topic	Reference	Identifier / link
Foundations	HU, E. J.; SHEN, Y. et al. LoRA: Low-Rank Adaptation of Large Language Models. ICLR 2022.	arXiv:2106.09685
Foundations	LI, C.; FARKHOOR, H.; LIU, R.; YOSINSKI, J. Measuring the Intrinsic Dimension of Objective Landscapes. ICLR 2018.	arXiv:1804.08838
Foundations	AGHAJANYAN, A.; GUPTA, S.; ZETTLEMOYER, L. Intrinsic Dimensionality Explains the Effectiveness of LM Fine-Tuning. ACL-IJCNLP 2021.	arXiv:2012.13255
Variants	DETTMERS, T.; PAGNONI, A.; HOLTZMAN, A.; ZETTLEMOYER, L. QLoRA: Efficient Finetuning of Quantized LLMs. NeurIPS 2023.	arXiv:2305.14314
Variants	ZHANG, Q. et al. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. ICLR 2023.	arXiv:2303.10512
Variants	KALAJDZIEVSKI, D. A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA (rsLoRA). 2023.	arXiv:2312.03732
Variants	KOPICZKO, D. J.; BLANKEVOORT, T.; ASANO, Y. M. VeRA: Vector-based Random Matrix Adaptation. ICLR 2024.	arXiv:2310.11454
Variants	LI, Y. et al. LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models. ICLR 2024.	arXiv:2310.08659
Variants	CHEN, Y. et al. LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. ICLR 2024.	arXiv:2309.12307
Variants	LIU, S.-Y. et al. DoRA: Weight-Decomposed Low-Rank Adaptation. ICML 2024 (oral).	arXiv:2402.09353

Validated references (2/3)

Topic	Reference	Identifier / link
Variants	HAYOU, S.; GHOSH, N.; YU, B. LoRA+: Efficient Low Rank Adaptation of Large Models. ICML 2024.	arXiv:2402.12354
Variants	MENG, F.; WANG, Z.; ZHANG, M. PiSSA: Principal Singular Values and Singular Vectors Adaptation. NeurIPS 2024.	arXiv:2404.02948
Systems	CHEN, L. et al. Punica: Multi-Tenant LoRA Serving. MLSys 2024.	arXiv:2310.18547
Systems	SHENG, Y. et al. S-LoRA: Serving Thousands of Concurrent LoRA Adapters. 2023.	arXiv:2311.03285
Systems	VLLM PROJECT. LoRA Adapters (official documentation).	docs.vllm.ai
Competitors	HOULSBY, N. et al. Parameter-Efficient Transfer Learning for NLP. ICML 2019.	arXiv:1902.00751
Competitors	LI, X. L.; LIANG, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. ACL-IJCNLP 2021.	arXiv:2101.00190
Competitors	LESTER, B.; AL-RFOU, R.; CONSTANT, N. The Power of Scale for Parameter-Efficient Prompt Tuning. EMNLP 2021.	arXiv:2104.08691
Competitors	BEN ZAKEN, E.; RAVFOGEL, S.; GOLDBERG, Y. BitFit. ACL 2022 (Short Papers).	arXiv:2106.10199
Competitors	LIU, H. et al. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning ((IA) ³). NeurIPS 2022.	arXiv:2205.05638

Validated references (3/3)

Topic	Reference	Identifier / link
Competitors	ZHAO, J. et al. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. ICML 2024.	arXiv:2403.03507
Studies	BIDERMAN, D. et al. LoRA Learns Less and Forgets Less. TMLR 2024; ICLR 2025.	arXiv:2405.09673
Studies	SCHULMAN, J.; THINKING MACHINES LAB. LoRA Without Regret. Connectionism, Sep. 2025. DOI 10.64434/tml.20250929.	thinkingmachines.ai
Adoption	MICROSOFT. Customize a model with fine-tuning (Azure OpenAI). Microsoft Learn.	learn.microsoft.com
Adoption	APPLE. Introducing Apple's On-Device and Server Foundation Models. Jun. 2024.	machinelearning.apple.com
Adoption	DEEPLARNING.AI. Thinking Machines' new Tinker API... The Batch, 2025.	deeplearning.ai
Adoption	HUGGING FACE. Using LoRA for Efficient Stable Diffusion Fine-Tuning. Blog, Jan. 26, 2023.	huggingface.co/blog/lora
Adoption	RYU, S. (cloneofsimon). lora: Low-rank Adaptation for Fast Text-to-Image Diffusion Fine-tuning. GitHub.	github.com/cloneofsimon/lora
Context	GOOGLE. How developers build AI for good with Gemma 4 (Gemma 4 Good winners).	blog.google
Context	ALICE0914. Scam Sentinel (Gemma 4 E2B), model card; self-reported figures.	huggingface.co