

# LoRA

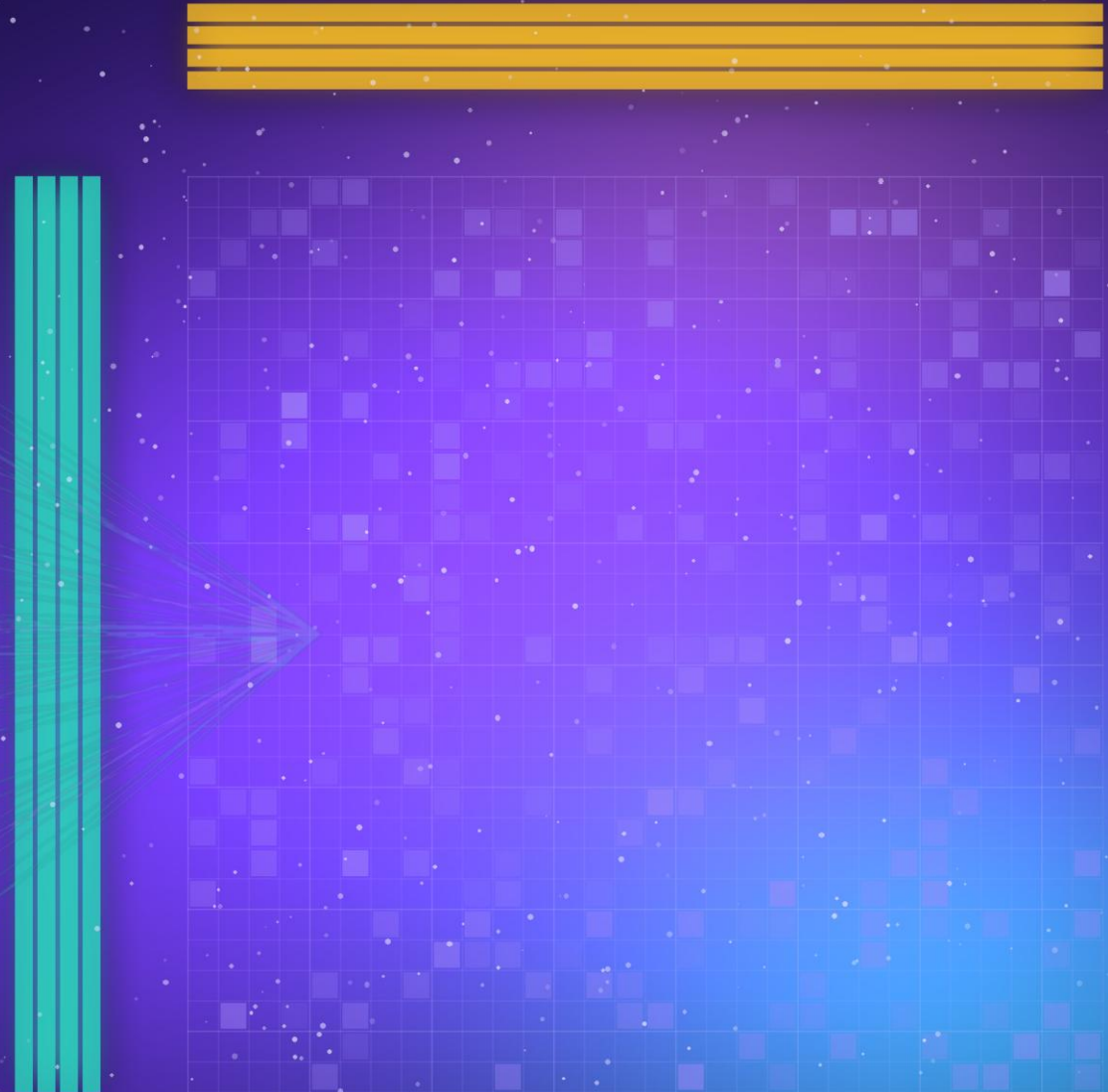
Low-Rank Adaptation

## Origem, Fundamentos, Variantes, Engenharia e Adoção

*Como uma correção  $W_o + BA$  de baixo posto tornou o ajuste fino de grandes modelos barato, rápido e modular*

Aridio Silva

@aridiosilva · Fontes primárias verificadas



# O que é a LoRA

*Congela o modelo e treina só uma correção de baixo posto:  $W = W_o + BA$*

**$\approx 10.000\times$**

menos parâmetros treináveis  
no GPT-3 175B

**$3\times$**

menos memória de GPU no  
treino (1,2 TB  $\rightarrow$  350 GB)

**0**

latência adicional após fundir o  
adaptador



## Importância

Tornou o ajuste fino de modelos gigantes acessível e virou o método padrão de personalização.

- **Quem e quando:** Edward J. Hu, Yelong Shen e mais seis pesquisadores da Microsoft; arXiv:2106.09685 (17/06/2021), ICLR 2022.
- **Ideia:** a mudança de pesos necessária para adaptar um modelo tem posto intrínseco baixo; basta aprendê-la como produto de duas matrizes estreitas.
- **Resultado:** qualidade igual ou superior ao ajuste fino completo em RoBERTa, DeBERTa, GPT-2 e GPT-3.
- **Hoje:** padrão de mercado no ajuste fino de LLMs e de modelos de imagem.



## Implicações positivas

Custo, memória e armazenamento muito menores; adaptadores pequenos e intercambiáveis.



## Implicações negativas

Capacidade limitada pelo posto; exige escolha cuidadosa de posto, alfa e taxa de aprendizado.

# Por que a LoRA foi criada

*Adaptar um modelo gigante a cada tarefa custava tanto quanto copiá-lo inteiro*

**350 GB**

por checkpoint ajustado do  
GPT-3 175B

**35 TB**

para manter 100 versões  
ajustadas

**1,2 TB**

de VRAM no ajuste fino  
completo com Adam

## Adaptadores (Houlsby et al., 2019)

Camadas extras em série aumentam a latência:  
+20,7% a +30,3% no GPT-2 medium com lote 1  
(medição de Hu et al.).

## Prefix e prompt tuning (2021)

Otimização instável e desempenho não  
monotônico; os vetores consomem parte do  
comprimento de contexto.

**Requisitos da nova técnica: poucos parâmetros · qualidade do ajuste completo · zero latência ·  
troca rápida de tarefa**



## Importância

Explica o problema econômico que a LoRA  
resolve: escala de armazenamento, memória e  
implantação.



## Implicações positivas

Definiu requisitos claros e mensuráveis para  
métodos eficientes.



## Implicações negativas

As soluções da época obrigavam a escolher  
entre eficiência, latência e qualidade.

# Origem: dimensão intrínseca e autores

*A LoRA levou para dentro de cada camada uma descoberta teórica de 2018–2020*

1

**2018 · Li, Farkhoor, Liu, Yosinski**

Medem a dimensão intrínseca: o menor subespaço em que o treino ainda resolve o problema (ICLR 2018).

2

**2020 · Aghajanyan, Gupta, Zettlemoyer**

200 parâmetros projetados bastam para 90% do desempenho do RoBERTa no MRPC (ACL 2021).

3

**Jun 2021 · Hu, Shen et al. (Microsoft)**

Hipótese do posto intrínseco:  $\Delta W = BA$  por camada. arXiv:2106.09685.

4

**2022 · ICLR 2022**

Publicação revisada por pares; código loralib em [github.com/microsoft/LoRA](https://github.com/microsoft/LoRA).

## Autores do artigo original

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

*\* contribuição igual · Microsoft Corporation*

!

## Importância

Mostra que a técnica nasceu de base teórica sólida, não de um truque empírico.

+

## Implicações positivas

Explica por que poucos parâmetros bastam: modelos maiores tendem a ter dimensão intrínseca menor.

-

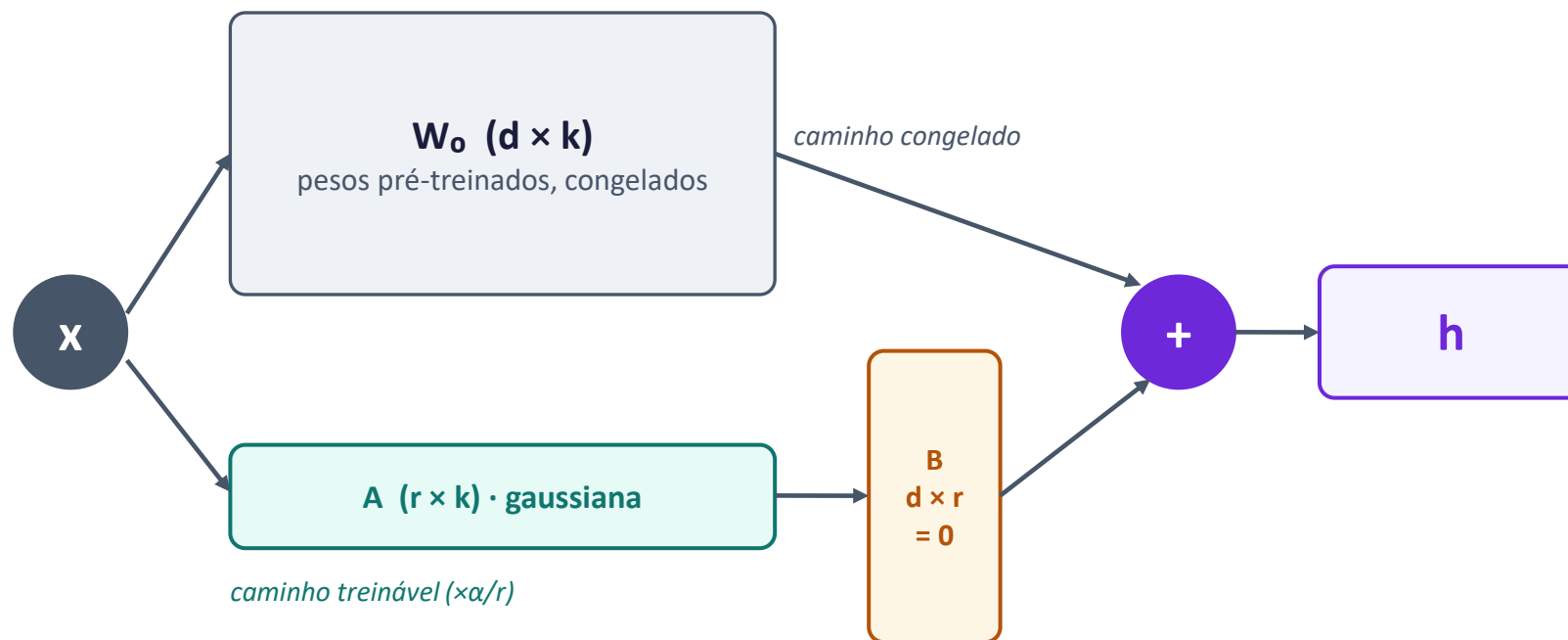
## Implicações negativas

A hipótese de baixo posto não vale para toda tarefa; ela falha em regimes próximos ao pré-treino.

*Atenção: LoRA (IA) não tem relação com LoRa, a tecnologia de rádio de longo alcance para IoT.*

# Como funciona: a decomposição $W_0 + BA$

*O peso original fica congelado; só as matrizes estreitas A e B recebem gradiente*



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

$B = 0$  no início  $\rightarrow$  o modelo parte idêntico ao pré-treinado. Após o treino,  $W = W_0 + (\alpha/r) \cdot BA$  é fundido e a inferência segue sem camadas extras. No artigo original,  $\alpha$  = primeiro  $r$  testado.

## ! Importância

É o núcleo da técnica: uma reparametrização linear simples, aplicável a qualquer camada densa.

## + Implicações positivas

Linearidade permite fundir os pesos: zero latência e troca de tarefa subtraindo  $BA$  e somando  $B'A'$ .

## - Implicações negativas

Com pesos fundidos, um mesmo lote não atende tarefas diferentes sem infraestrutura específica.

# Quanto se economiza

Parâmetros treináveis por matriz caem de  $d \times k$  para  $r \times (d + k)$

| Abordagem (matriz $4.096 \times 4.096$ ) | Parâmetros treináveis | % do original |
|------------------------------------------|-----------------------|---------------|
| Ajuste fino completo                     | 16.777.216            | 100%          |
| LoRA com $r = 16$                        | 131.072               | 0,78%         |
| LoRA com $r = 8$                         | 65.536                | 0,39%         |

350 GB → 35 MB

checkpoint do GPT-3 175B com  $r = 4$  em  $W_q$  e  $W_v$  (Hu et al.)

+25%

vazão de treino no GPT-3: 43,1 vs. 32,5 tokens/s por V100

Exemplo de  $4.096 \times 4.096$ : dimensão típica de projeção de atenção em modelos de ~7B; cálculo aritmético ilustrativo.



## Importância

Quantifica por que a LoRA mudou a economia do ajuste fino de LLMs.



## Implicações positivas

Menos GPUs, menos armazenamento e menos gargalos de E/S; 100 tarefas ≈ 354 GB em vez de 35 TB.

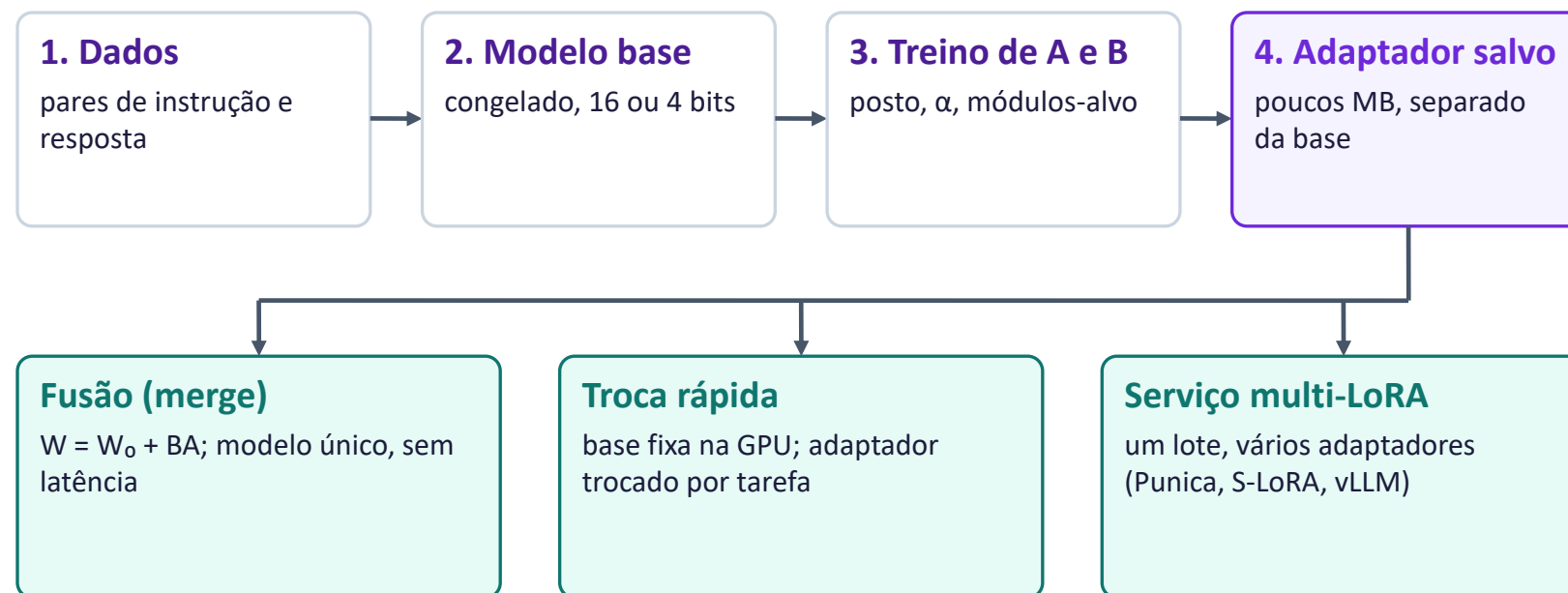


## Implicações negativas

O modelo base continua inteiro na memória; a economia é no treino e no armazenamento por tarefa.

# Workflow de engenharia

*Treina-se só o adaptador; a implantação escolhe entre três caminhos*



## 1. Dados

pares de instrução e resposta

## 2. Modelo base

congelado, 16 ou 4 bits

## 3. Treino de A e B

posto,  $\alpha$ , módulos-alvo

## 4. Adaptador salvo

poucos MB, separado da base

## Fusão (merge)

$W = W_0 + BA$ ; modelo único, sem latência

## Troca rápida

base fixa na GPU; adaptador trocado por tarefa

## Serviço multi-LoRA

um lote, vários adaptadores (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")
modelo = get_peft_model(modelo_base, config) # valores ilustrativos
```



## Importância

Separa um treino barato de uma implantação flexível, adequada a cada cenário.



## Implicações positivas

Adaptador versionável e reversível; o mesmo modelo base atende várias tarefas.



## Implicações negativas

Cada adaptador só funciona na base exata em que foi treinado; trocar de base exige retreinar.

# Serviço multiadaptador em escala

*Calcula-se o modelo base uma vez por lote e soma-se, por requisição, o adaptador certo*

## Punica

*Chen et al. · MLSys 2024*

Kernel CUDA SGMV agrupa no mesmo lote operações de modelos LoRA diferentes.

**12× vazão · +2 ms/token**

## S-LoRA

*Sheng et al. · 2023*

Adaptadores na memória principal; só os ativos vão à GPU, com paginação unificada.

**até 4× vazão · milhares de adaptadores**

## vLLM / SGLang

*código aberto*

Servidores aceitam vários adaptadores sobre a mesma base (max\_loras) e carga dinâmica.

**padrão em produção**



### Importância

Resolveu a limitação apontada no artigo original: lotes com tarefas diferentes.



### Implicações positivas

Personalização por cliente ou usuário com uma única cópia do modelo na GPU.



### Implicações negativas

Exige kernels e servidores especializados; números são dos próprios autores, em condições de laboratório.

# O que a LoRA resolve e como

*Cada problema da adaptação de modelos grandes tem um mecanismo correspondente*

| Problema                                  | Como a LoRA resolve                                      | Evidência (Hu et al., 2021)       |
|-------------------------------------------|----------------------------------------------------------|-----------------------------------|
| Memória de GPU no treino                  | Base congelada: sem gradientes nem estados do otimizador | 1,2 TB → 350 GB (GPT-3)           |
| Armazenamento por tarefa                  | Salva-se só o par A, B                                   | 350 GB → 35 MB                    |
| Troca de tarefa                           | Troca-se o adaptador, não o modelo                       | Operação rápida e leve            |
| Latência dos adaptadores                  | Atualização linear, fundível                             | Zero latência adicional           |
| Qualidade inferior dos métodos eficientes | Baixo posto suficientemente expressivo                   | ≥ ajuste completo em GLUE e GPT-3 |
| Velocidade de treino                      | Menos gradientes a calcular                              | +25% de vazão                     |

!

## Importância

Resume o valor da técnica em termos de problemas de negócio e de engenharia.

+

## Implicações positivas

Combina eficiência, qualidade e zero latência, algo que nenhum método anterior reunia.

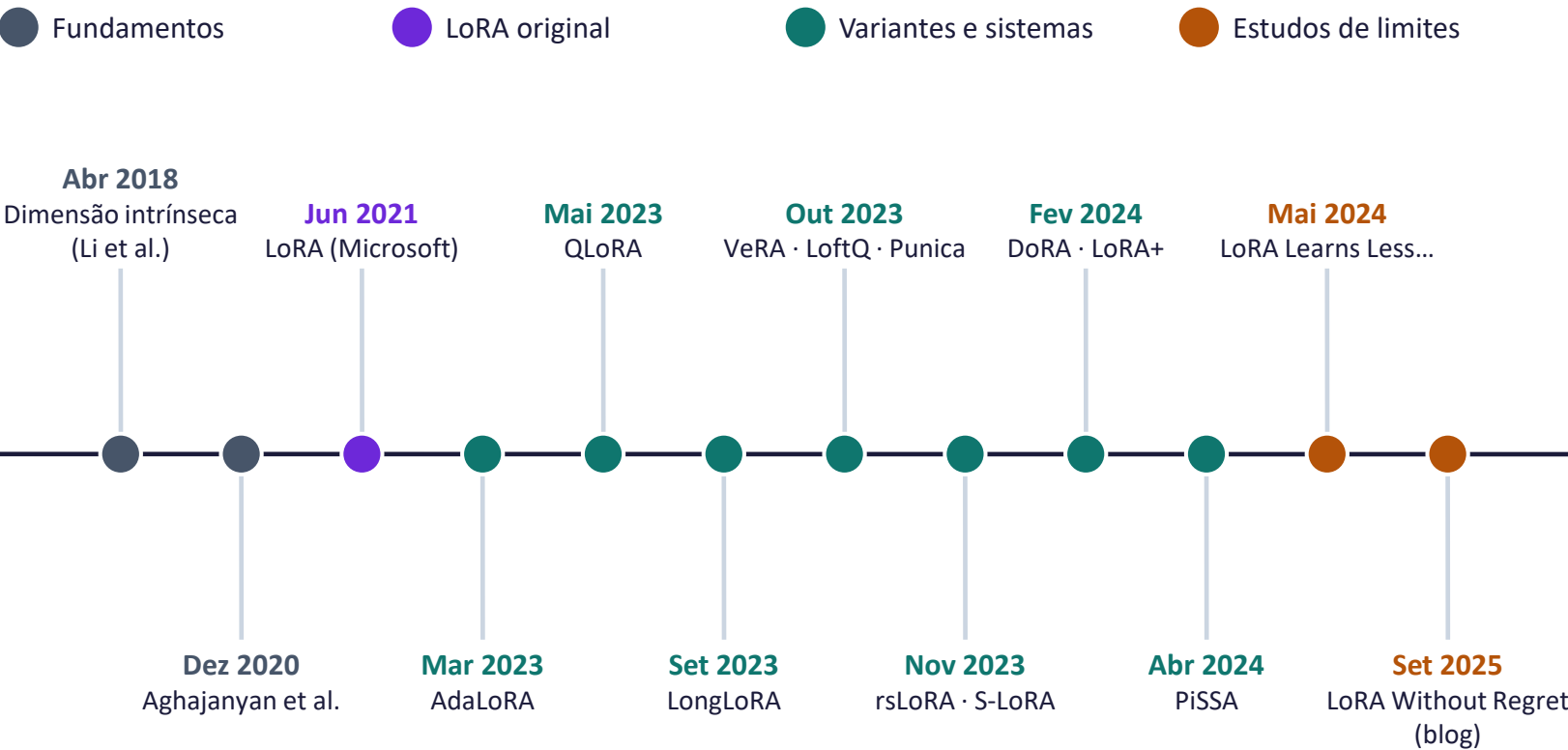
-

## Implicações negativas

Evidências do artigo original vêm de modelos de 2021; resultados variam por tarefa e configuração.

# Linha do tempo: LoRA, variantes e estudos

Data da primeira versão no arXiv (exceto onde indicado); espaçamento não proporcional ao tempo



## ! Importância

Mostra a explosão de pesquisa: em menos de um ano (2023–2024) surgiram as principais variantes.

## + Implicações positivas

Ecosistema maduro: variantes para memória, escala, inicialização, contexto e serviço.

## - Implicações negativas

Muitas variantes competem; os ganhos relatados nem sempre se repetem com metodologia uniforme.

# Variantes I: memória, orçamento e escala

Respostas ao custo de memória e à distribuição de parâmetros

| Variante · conferência                                      | Ideia central                                                          | Resultado relatado                                      |
|-------------------------------------------------------------|------------------------------------------------------------------------|---------------------------------------------------------|
| QLoRA · NeurIPS 2023<br>Dettmers et al. (U. Washington)     | Base em 4 bits (NF4), quantização dupla e otimizadores paginados       | Modelo de 65B ajustado numa GPU de 48 GB                |
| LoftQ · ICLR 2024<br>Li et al. (Georgia Tech, Microsoft)    | Quantiza e escolhe a inicialização da LoRA ao mesmo tempo              | Ganhos sobretudo em 2 bits e precisão mista 2/4 bits    |
| VeRA · ICLR 2024<br>Kopiczko et al. (U. Amsterdã, Qualcomm) | Matrizes aleatórias compartilhadas; treina só vetores de escala        | 10× menos parâmetros; Llama 2 7B com 1,4 milhão         |
| AdaLoRA · ICLR 2023<br>Zhang et al.                         | Atualização em forma de SVD; poda valores singulares pouco importantes | Melhor com orçamento baixo de parâmetros                |
| LongLoRA · ICLR 2024<br>Chen et al. (CUHK, MIT)             | Atenção esparsa deslocada ( $S^2$ -Attn) só no treino                  | Llama 2 7B até 100 mil tokens; 70B até 32 mil em 8 A100 |

## ! Importância

A QLoRA democratizou o ajuste fino de modelos grandes em uma única GPU.

## + Implicações positivas

Ajuste viável em hardware modesto; contexto longo e adaptadores minúsculos.

## - Implicações negativas

Quantização agressiva pode degradar qualidade sem inicialização adequada (motivo da LoftQ).

# Variantes II: otimização e inicialização

Ajustes finos no treino que melhoram a LoRA sem custo extra de inferência

| Variante · conferência                                | Limitação atacada                             | Ideia e resultado relatado                                                    |
|-------------------------------------------------------|-----------------------------------------------|-------------------------------------------------------------------------------|
| rsLoRA · 2023<br>Kalajdzievski                        | Postos altos aprendem devagar com $\alpha/r$  | Escala $\alpha/vr$ : posto maior passa a render mais, sem custo na inferência |
| DoRA · ICML 2024 (oral)<br>Liu et al. (NVIDIA, HKUST) | Lacuna em relação ao ajuste completo          | Separa magnitude e direção; supera a LoRA em LLaMA, LLaVA e VL-BART           |
| LoRA+ · ICML 2024<br>Hayou, Ghosh, Yu (Berkeley)      | Mesma taxa para A e B é subótima              | Taxas diferentes: +1% a 2% e treino até ~2× mais rápido                       |
| PiSSA · NeurIPS 2024<br>Meng, Wang, Zhang (Pequim)    | Convergência lenta da inicialização aleatória | SVD de W inicializa A e B; GSM8K 72,86% vs. 67,7% (Mistral-7B)                |

Resultados relatados pelos próprios autores, em condições diferentes; não são comparáveis entre linhas.

!

## Importância

Mostra que detalhes de escala, taxa e inicialização ainda rendem ganhos relevantes.

+

## Implicações positivas

Trocas simples (uma escala, duas taxas, outra inicialização) com custo de inferência zero.

-

## Implicações negativas

Mais hiperparâmetros e escolhas; benefício depende da tarefa e nem sempre se reproduz.

# Resultados de referência (2021–2023)

A LoRA igualou ou superou o ajuste completo com fração mínima de parâmetros

| Estudo          | Modelo e tarefa                   | Resultado relatado                                                  |
|-----------------|-----------------------------------|---------------------------------------------------------------------|
| LoRA (2021)     | RoBERTa base · GLUE               | Média 87,2 com 0,3 M parâmetros vs. 86,4 do ajuste completo (125 M) |
| LoRA (2021)     | DeBERTa XXL (1,5 B) · GLUE        | Média 91,3 com 4,7 M parâmetros vs. 91,1 do ajuste completo         |
| QLoRA (2023)    | LLaMA até 65B · chatbot Guanaco   | 99,3% do ChatGPT no benchmark Vicuna após 24 h numa única GPU       |
| LongLoRA (2023) | Llama 2 7B e 70B · contexto longo | 4 mil → 100 mil tokens (7B); 32 mil (70B) em uma máquina 8xA100     |
| VeRA (2023)     | Llama 2 7B · instruções           | Desempenho equivalente à LoRA com 1,4 M parâmetros treináveis       |

## ! Importância

Evidência empírica que levou à adoção em massa a partir de 2023.

## + Implicações positivas

Qualidade competitiva com custo muito menor, inclusive em modelos de dezenas de bilhões.

## - Implicações negativas

Benchmarks da época (GLUE, Vicuna) são limitados; avaliação por GPT-4 foi usada no QLoRA.

# Resultados 2024: ganhos e limites

Variantes melhoraram a LoRA, mas um estudo mostrou onde ela fica atrás

| Estudo                                 | Modelo e tarefa                  | Resultado relatado                                                                        |
|----------------------------------------|----------------------------------|-------------------------------------------------------------------------------------------|
| PiSSA                                  | Mistral-7B e LLaMA-3-70B · GSM8K | 72,86% vs. 67,7% da LoRA; QPiSSA 86,05% vs. 81,73% da QLoRA                               |
| LoRA+                                  | Vários modelos e tarefas         | +1% a 2% de desempenho; treino até ~2× mais rápido                                        |
| Biderman et al. (TMLR 2024; ICLR 2025) | Llama-2-7B · código e matemática | Bem abaixo do ajuste completo no pré-treino continuado (~20 bi tokens), mas esquece menos |

10–100×

posto das perturbações do ajuste completo vs. LoRA típica (Biderman et al.)

Esquece menos

preserva o domínio original melhor que weight decay e dropout

! **Importância**

Delimita o alcance da técnica: ótima para adaptar, fraca para ensinar volumes enormes de conhecimento.

+ **Implicações positivas**

LoRA atua como regularizador: preserva capacidades fora do domínio e a diversidade das respostas.

- **Implicações negativas**

Em código e matemática com pré-treino continuado, perde para o ajuste completo mesmo com posto alto.

# LoRA Without Regret (Thinking Machines, 2025)

*Estudo liderado por John Schulman mapeou o “regime de baixo arrependimento”*

## 1 SFT pequeno/médio

Mesmo desempenho do ajuste completo em conjuntos de instruções e raciocínio.

## 2 Todas as camadas

Aplicar a MLP e MoE é essencial; só atenção fica abaixo mesmo com parâmetros iguais.

## ! Importância

Transformou a LoRA de “compromisso barato” em escolha sem perda para a maior parte do pós-treino.

## 3 RL com posto 1

Em gradiente de política, até o posto 1 iguala o ajuste completo (~1 bit por episódio).

## 4 Taxa de aprendizado

Ótima  $\approx 10\times$  a do ajuste completo, em SFT e em RL.

## + Implicações positivas

Receita prática: todas as camadas, LR  $\sim 10\times$ , lotes moderados; ideal para alinhamento por RL.

## 5 Lotes grandes

Penalidade maior que no ajuste completo; aumentar o posto não resolve.

## 6 Custo por passagem

Pouco mais de 2/3 das operações (FLOPs) do ajuste completo.

## - Implicações negativas

Quando os dados excedem a capacidade do adaptador, o aprendizado desacelera; publicado como blog técnico.

# Concorrentes I: antecessores da LoRA

*Adaptadores e métodos de prompt abriram o caminho, com limitações*

| Método · origem                            | O que treina                   | Latência extra               | Força / fraqueza                         |
|--------------------------------------------|--------------------------------|------------------------------|------------------------------------------|
| Adaptadores<br>Houlsby et al., ICML 2019   | Redes de gargalo entre camadas | Sim (+20,7% a 30,3%, lote 1) | Modulares / processamento em série       |
| Prefix tuning<br>Li & Liang, ACL 2021      | Vetores de prefixo por camada  | Pequena; consome contexto    | Poucos parâmetros / otimização instável  |
| Prompt tuning<br>Lester et al., EMNLP 2021 | Embeddings de tokens virtuais  | Pequena; consome contexto    | Leve / competitivo só em modelos grandes |
| BitFit<br>Ben Zaken et al., ACL 2022       | Só os vetores de viés          | Nenhuma                      | Simples / GLUE 85,2 vs. 87,2 da LoRA     |

## ! Importância

Contextualiza a LoRA como síntese das melhores ideias anteriores.

## + Implicações positivas

Métodos de prompt e BitFit seguem úteis quando o orçamento de parâmetros é mínimo.

## - Implicações negativas

Latência dos adaptadores e perda de contexto dos prompts limitam o uso em produção.

# Concorrentes II: alternativas atuais

Cada alternativa vence em um eixo; a LoRA reúne mais vantagens ao mesmo tempo

| Método · origem                               | O que treina                        | Ponto forte                                               | Ponto fraco                       |
|-----------------------------------------------|-------------------------------------|-----------------------------------------------------------|-----------------------------------|
| Ajuste fino completo                          | Todos os pesos                      | Teto de qualidade, inclusive pré-treino                   | Modelo inteiro por tarefa         |
| (IA) <sup>3</sup><br>Liu et al., NeurIPS 2022 | Vetores que reescalam ativações     | Muitíssimo menos parâmetros; poucos exemplos              | Menor capacidade                  |
| GaLore<br>Zhao et al., ICML 2024              | Todos os pesos, gradiente projetado | Até 65,5% menos memória do otimizador; 7B em GPU de 24 GB | Não gera adaptador intercambiável |
| LoRA e variantes<br>Hu et al., ICLR 2022      | Matrizes A e B                      | Qualidade, zero latência, modularidade, ecossistema       | Atrás perto do pré-treino         |

!

### Importância

Ajuda a escolher a técnica certa para cada objetivo e orçamento.

+

### Implicações positivas

GaLore estende a ideia de baixo posto ao pré-treino; (IA)<sup>3</sup> e VeRA servem a milhares de adaptadores.

-

### Implicações negativas

Nenhuma alternativa domina todos os eixos; a escolha exige avaliar dados, custo e implantação.

# Alternativas sem treino: RAG e aprendizado em contexto

*Nem toda adaptação exige ajuste fino*

## LoRA

Ensina comportamento, formato, estilo e competência de forma estável.

**Quando usar:** Quando a tarefa se repete e há exemplos de treino.

## RAG

Busca documentos e os insere no contexto, sem alterar pesos.

**Quando usar:** Quando os fatos mudam com frequência ou precisam de fonte.

## Aprendizado em contexto

Exemplos no próprio prompt (few-shot).

**Quando usar:** Tarefas simples, protótipos e baixo volume.



### Importância

Evita usar ajuste fino onde uma busca ou um prompt resolveriam.



### Implicações positivas

Combinação LoRA + RAG reduz custos e mantém respostas atualizadas e rastreáveis.



### Implicações negativas

Ajustar fatos voláteis com LoRA envelhece rápido e exige retreinos frequentes.

**Na prática, as abordagens são combinadas: LoRA para o comportamento, RAG para os fatos.**

# Adoção pelo mercado de IA

Uso declarado em fontes oficiais das próprias organizações

| Organização                       | Uso declarado da LoRA                                                                    |
|-----------------------------------|------------------------------------------------------------------------------------------|
| Microsoft · Azure OpenAI          | O serviço de ajuste fino declara usar LoRA, treinando um subconjunto menor de parâmetros |
| Apple · Apple Intelligence (2024) | Modelo de ~3B no dispositivo com adaptadores por função; posto 16 ≈ dezenas de MB        |
| Thinking Machines · Tinker (2025) | API de ajuste fino de modelos abertos com LoRA; vários treinos compartilham GPUs         |
| Hugging Face                      | Biblioteca PEFT (LoRA, AdaLoRA, DoRA, VeRA); LoRA em difusão (diffusers) desde jan/2023  |
| vLLM e SGLang                     | Servidores de inferência com vários adaptadores em lote sobre a mesma base               |
| NVIDIA Research                   | Criação e publicação da variante DoRA (ICML 2024)                                        |
| Comunidade (Unsloth, Kaggle)      | LoRA/QLoRA em uma GPU; ex.: hackathon Gemma 4 Good (2026)                                |



## Importância

Confirma que a LoRA saiu da pesquisa e virou infraestrutura de produtos.



## Implicações positivas

Grandes provedores, fabricantes de dispositivos e startups padronizaram o uso.



## Implicações negativas

Dependência de um único padrão; detalhes internos (posto, camadas) raramente são divulgados.

# Onde é usada e quem se beneficia

*Da nuvem ao celular, do laboratório à comunidade de imagem*

**1 Modelos de linguagem**  
Instruções, domínios (jurídico, médico, financeiro), estilo, saída estruturada, alinhamento por RL

**2 Geração de imagens**  
Estilos, personagens e objetos em Stable Diffusion; pioneiro: Simo Ryu (@cloneofsimo)

**3 Dispositivos de borda**  
Modelos locais especializados, como na Apple Intelligence

**4 Plataformas SaaS**  
Um adaptador por cliente ou tarefa sobre base compartilhada

**5 Pesquisa e ensino**  
Pós-treino e RL com orçamento limitado; competições e protótipos

## ! Importância

Mostra a amplitude de aplicação e quem captura valor com a técnica.

## + Implicações positivas

Beneficia empresas com dados sensíveis, pequenas equipes, provedores SaaS e usuários finais.

## - Implicações negativas

Facilidade de criar e distribuir adaptadores amplia riscos de uso indevido e de procedência.

# Prós

Vantagens com evidência nas fontes primárias

| Vantagem                                         | Evidência                                                   |
|--------------------------------------------------|-------------------------------------------------------------|
| Memória de treino muito menor                    | 1,2 TB → 350 GB (GPT-3); 65B numa GPU de 48 GB (QLoRA)      |
| Armazenamento mínimo por tarefa                  | 350 GB → 35 MB (GPT-3); dezenas de MB por adaptador (Apple) |
| Sem latência adicional                           | Fusão linear ao peso original                               |
| Qualidade igual ao ajuste completo no pós-treino | GLUE (2021); Thinking Machines (2025)                       |
| Menos esquecimento                               | Biderman et al. (2024)                                      |
| Menor custo por passagem                         | Pouco mais de 2/3 dos FLOPs do ajuste completo              |
| Modularidade e serviço multicliente              | Punica, S-LoRA, vLLM                                        |

 **Importância**

Sintetiza por que a LoRA é a escolha padrão de personalização em 2026.

 **Implicações positivas**

Custo, velocidade e modularidade reduzem a barreira de entrada para qualquer organização.

 **Implicações negativas**

Vantagens dependem de configuração correta: todas as camadas, posto e taxa adequados.

# Contras e limitações

Onde a LoRA exige cuidado ou perde para alternativas

| Limitação                          | Evidência ou explicação                                                                  |
|------------------------------------|------------------------------------------------------------------------------------------|
| Aprende menos perto do pré-treino  | Pré-treino continuado em código e matemática (Biderman et al.)                           |
| Capacidade limitada pelo posto     | Dados além da capacidade desaceleram o aprendizado                                       |
| Sensível a hiperparâmetros         | LR ótima $\approx 10\times$ a do ajuste completo; escalas $\alpha/r$ e $\alpha/\sqrt{r}$ |
| Penalidade com lotes grandes       | Diferença cresce com o lote, sem correção pelo posto                                     |
| Configuração original insuficiente | Só atenção não basta; cobrir todas as camadas lineares                                   |
| Lotes mistos exigem infraestrutura | Kernels e servidores multiadaptador específicos                                          |
| Dependência do modelo base         | Trocar de base exige retreinar o adaptador                                               |

!

## Importância

Evita expectativas irreais e orienta quando escolher outra abordagem.

+

## Implicações positivas

Limitações são conhecidas e mensuráveis, com receitas práticas de mitigação.

-

## Implicações negativas

Para absorver grandes volumes de conhecimento novo, o ajuste completo ou o GaLore seguem preferíveis.

# Governança, segurança (GRC) e beneficiários

*O adaptador é um artefato pequeno que merece controle próprio*

## Controles recomendados

- Registrar origem, versão e modelo base de cada adaptador
- Verificar integridade (hash, assinatura) antes de carregar
- Restringir quem publica adaptadores em produção
- Tratar adaptadores como componentes da cadeia de suprimentos de software
- Usar a separação base/adaptador para auditoria e rollback rápido

## Quem se beneficia

- Organizações com dados sensíveis, ajustando modelos na própria infraestrutura
- Pequenas equipes, pesquisadores e estudantes com uma única GPU
- Provedores SaaS com um adaptador por cliente
- Fabricantes de dispositivos com modelos locais
- Usuários finais: modelos especializados, mais baratos e privados



## Importância

Conecta a técnica a requisitos de governança, auditoria e segurança da informação.



## Implicações positivas

Separação base/adaptador facilita revisão, versionamento, revogação e reversão.



## Implicações negativas

Adaptadores de terceiros sem procedência podem alterar o comportamento do modelo sem alarde.

# Posição atual da tecnologia (outubro de 2026)

*De compromisso entre custo e qualidade a escolha natural para o pós-treino*

## Estado consolidado

Iguala o ajuste completo no pós-treino pequeno/médio e em RL; boas práticas definidas; ecossistema completo (PEFT, Unsloth, TRL, vLLM, SGLang).

## Tendências

LoRA no alinhamento por RL; ajuste na borda com modelos pequenos; treino como serviço sobre bases compartilhadas; novas inicializações.

## Presença em produtos

Azure OpenAI, Apple Intelligence (no dispositivo) e Tinker documentam o uso da LoRA em seus produtos.

## Questões em aberto

Prever quando iguala o ajuste completo; explicar a razão de  $\sim 10\times$  nas taxas; avaliar variantes de forma uniforme; LoRA em MoE distribuído.



## Importância

Situa o leitor no estado da arte e no que observar nos próximos meses.



## Implicações positivas

Tecnologia madura, com receitas validadas e suporte amplo de ferramentas e provedores.



## Implicações negativas

Lacunas teóricas persistem; novos resultados podem mudar boas práticas rapidamente.

# Conclusão

*A LoRA resolveu o problema que motivou sua criação, e sabemos onde estão seus limites*

## Resolveu

Adaptação barata, rápida e modular de modelos grandes, sem custo de inferência.

## Democratizou

Com a QLoRA, modelos de dezenas de bilhões passaram a caber numa única GPU.

## Escalou

Variantes refinaram o treino; Punica, S-LoRA e vLLM viabilizaram milhares de adaptadores.

## Tem limites

Fica atrás quando é preciso aprender muito conhecimento novo, perto do pré-treino.

**Recomendação: use LoRA em todas as camadas lineares para comportamento e formato; combine com RAG para fatos; prefira ajuste completo ou GaLore para absorver grandes volumes de conhecimento.**

# Glossário (1/3)

| Termo                     | Definição                                                                                                |
|---------------------------|----------------------------------------------------------------------------------------------------------|
| Adam / AdamW              | Otimizadores que guardam momentos por parâmetro treinável; a memória cresce com os parâmetros treinados. |
| Adaptador                 | Pequeno conjunto de pesos treináveis acoplado a um modelo congelado (na LoRA, o par A e B).              |
| Ajuste fino               | Continuação do treino de um modelo pré-treinado com dados de uma tarefa ou domínio.                      |
| Ajuste fino completo      | Ajuste em que todos os pesos do modelo são atualizados (FullFT).                                         |
| Alfa ( $\alpha$ )         | Constante de escala da LoRA: a atualização é multiplicada por $\alpha/r$ ( $\alpha/v_r$ na rsLoRA).      |
| Aprendizado por reforço   | Treino guiado por recompensas atribuídas às respostas do modelo (RL).                                    |
| Checkpoint                | Arquivo com os pesos salvos de um modelo ou adaptador.                                                   |
| Dimensão intrínseca       | Menor subespaço em que o treino ainda atinge solução satisfatória (Li et al., 2018).                     |
| Dropout                   | Regularização que zera aleatoriamente parte das ativações durante o treino.                              |
| Esquecimento catastrófico | Perda de capacidades anteriores após ajustar o modelo a nova tarefa.                                     |
| Fusão (merge)             | Incorporar a atualização ao peso: $W = W_o + (\alpha/r) \cdot BA$ ; elimina latência extra.              |
| GLUE                      | Conjunto padronizado de tarefas de compreensão de linguagem usado como benchmark.                        |

# Glossário (2/3)

| Termo                         | Definição                                                                              |
|-------------------------------|----------------------------------------------------------------------------------------|
| <b>GPU / VRAM</b>             | Processador gráfico usado no treino e na inferência / memória própria dele.            |
| <b>Gradient checkpointing</b> | Recalcula ativações na retropropagação para economizar memória.                        |
| <b>Inferência</b>             | Uso do modelo treinado para gerar respostas.                                           |
| <b>Kernel CUDA</b>            | Rotina executada na GPU; Punica criou o kernel SGMV para lotes com vários adaptadores. |
| <b>Latência</b>               | Tempo entre a requisição e a resposta do modelo.                                       |
| <b>LLM</b>                    | Large Language Model, modelo de linguagem de grande porte.                             |
| <b>Lote (batch)</b>           | Conjunto de exemplos ou requisições processados juntos.                                |
| <b>MLP</b>                    | Bloco totalmente conectado (feed-forward) dentro de cada camada do Transformer.        |
| <b>MoE</b>                    | Mixture of Experts: só alguns “especialistas” são ativados por token.                  |
| <b>NF4</b>                    | 4-bit NormalFloat, tipo de 4 bits da QLoRA para pesos com distribuição normal.         |
| <b>PEFT</b>                   | Parameter-Efficient Fine-Tuning; também a biblioteca da Hugging Face.                  |
| <b>Posto (rank, r)</b>        | Nº de colunas linearmente independentes; na LoRA, a largura do gargalo entre A e B.    |

# Glossário (3/3)

| Termo                    | Definição                                                                   |
|--------------------------|-----------------------------------------------------------------------------|
| Posto intrínseco         | Hipótese de que a mudança de pesos na adaptação tem posto baixo.            |
| Pré-treino continuado    | Continuar o pré-treino com grande volume de texto de um domínio (CPT).      |
| $W_q, W_k, W_v, W_o$     | Projeções de query, key, value e saída da autoatenção.                      |
| Quantização              | Representar pesos com menos bits (ex.: 4 bits) para reduzir memória.        |
| RAG                      | Retrieval-Augmented Generation: busca documentos e os insere no contexto.   |
| SFT                      | Supervised Fine-Tuning, ajuste supervisionado com pares instrução-resposta. |
| SVD                      | Decomposição em valores singulares; base da AdaLoRA e da PiSSA.             |
| Taxa de aprendizado (LR) | Tamanho do passo de atualização dos pesos a cada iteração.                  |
| Token                    | Unidade de texto processada pelo modelo, em geral parte de uma palavra.     |
| Transformer              | Arquitetura neural baseada em atenção, base dos LLMs atuais.                |
| Vazão (throughput)       | Volume de requisições ou tokens processados por unidade de tempo.           |

# Referências bibliográficas validadas (1/3)

Fontes primárias consultadas em outubro de 2026; links ativos

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

# Referências bibliográficas validadas (2/3)

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

# Referências bibliográficas validadas (3/3)

| Tema         | Referência                                                                                                     | Identificador / link                         |
|--------------|----------------------------------------------------------------------------------------------------------------|----------------------------------------------|
| Concorrentes | ZHAO, J. et al. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. ICML 2024.              | <a href="#">arXiv:2403.03507</a>             |
| Estudos      | BIDERMAN, D. et al. LoRA Learns Less and Forgets Less. TMLR 2024; ICLR 2025.                                   | <a href="#">arXiv:2405.09673</a>             |
| Estudos      | SCHULMAN, J.; THINKING MACHINES LAB. LoRA Without Regret. Connectionism, set. 2025. DOI 10.64434/tml.20250929. | <a href="#">thinkingmachines.ai</a>          |
| Adoção       | MICROSOFT. Customize a model with fine-tuning (Azure OpenAI). Microsoft Learn.                                 | <a href="#">learn.microsoft.com</a>          |
| Adoção       | APPLE. Introducing Apple's On-Device and Server Foundation Models. Jun. 2024.                                  | <a href="#">machinelearning.apple.com</a>    |
| Adoção       | DEEPLARNING.AI. Thinking Machines' new Tinker API... The Batch, 2025.                                          | <a href="#">deeplearning.ai</a>              |
| Adoção       | HUGGING FACE. Using LoRA for Efficient Stable Diffusion Fine-Tuning. Blog, 26 jan. 2023.                       | <a href="#">huggingface.co/blog/lora</a>     |
| Adoção       | RYU, S. (cloneofsimon). lora: Low-rank Adaptation for Fast Text-to-Image Diffusion Fine-tuning. GitHub.        | <a href="#">github.com/cloneofsimon/lora</a> |
| Contexto     | GOOGLE. How developers build AI for good with Gemma 4 (vencedores do Gemma 4 Good).                            | <a href="#">blog.google</a>                  |
| Contexto     | ALICE0914. Scam Sentinel (Gemma 4 E2B), cartão do modelo; números autodeclarados.                              | <a href="#">huggingface.co</a>               |