

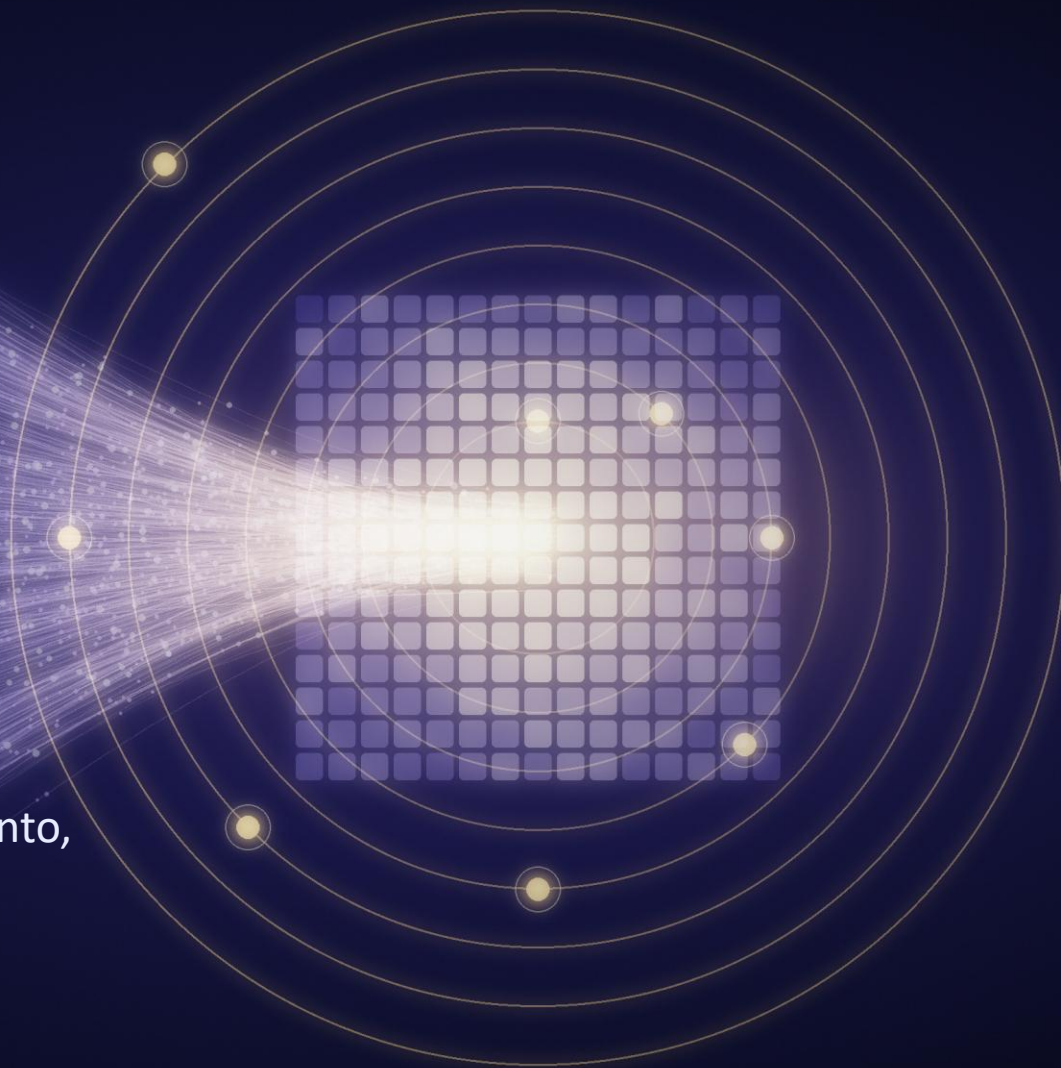
BRIEFING do Mundo da IA · EDIÇÃO ESPECIAL

FROM ZERO TO WEIGHTS

The forging of a frontier AI

As 8 fases do treinamento de LLMs: dados, escala, alinhamento, avaliação e liberação dos pesos

Aridio Silva · @aridiosilva



A frontier AI is born through a process

- Uma LLM de fronteira é produzida em 8 fases. O maior esforço está em dados e infraestrutura, não na arquitetura.
- O pré-treinamento ensina o modelo a prever o próximo token; o pós-treinamento o transforma em assistente.
- O fluxo vale para laboratórios abertos e fechados. Só a fase 8 muda: os fechados entregam via API e não liberam os pesos.
- Este infográfico mostra, fase a fase, insumos, outputs, objetivo, importância, técnicas e referências.

8

fases do ciclo, do planejamento à liberação dos pesos

15,6 tri

tokens de treino do Llama 3 405B [3]

16 mil

GPUs H100 usadas no treino do Llama 3 405B [3]

1

Planning

2

Data

3

Architecture

4

Pre-training

5

Post-training

6

Evaluation

7

Optimization

8

Release

Planning e compute

O QUE É FEITO

- Define o orçamento de FLOPs e a capacidade-alvo do modelo.
- Divide o compute entre parâmetros e tokens com base em leis de escala [1][2].
- Treina além do ponto compute-ótimo para baixar o custo de servir o modelo [3].
- Monta o cluster: o Llama 3 405B usou mais de 16 mil GPUs H100 [3].

OBJETIVO E PORQUÊ

Transformar uma meta de capacidade em um plano de compute viável. Uma escala mal escolhida não se corrige depois: custa meses de cluster e não pode ser compensada por ajustes posteriores.

IMPORTÂNCIA DA ETAPA

Define o teto de capacidade e o custo total do projeto. Todas as fases seguintes herdam essas decisões.

INSUMOS (ENTRADAS)

- Meta de capacidade e casos de uso do modelo
- Orçamento financeiro, de energia e de tempo
- Leis de escala publicadas [1][2]
- Acesso a milhares de GPUs/TPUs

OUTPUTS (SAÍDAS)

- Orçamento de FLOPs aprovado
- Tamanho do modelo e meta de tokens
- Cluster montado: rede, armazenamento e orquestração

Data

O QUE É FEITO

- Coleta fontes heterogêneas: web, código, livros, artigos, licenciados e sintéticos [3].
- Limpa: extrai texto, filtra qualidade, remove PII e conteúdo tóxico [4][5].
- Deduplica e descontamina contra benchmarks, reduzindo memorização e avaliação inflada [6].
- Ajusta a mistura de domínios com experimentos pequenos e tokeniza com BPE [3][7].

OBJETIVO E PORQUÊ

Entregar ao modelo texto diverso, limpo e sem repetição. O modelo só aprende o que vê: a qualidade e a proporção dos dados moldam capacidades, vieses e riscos.

IMPORTÂNCIA DA ETAPA

Concentra a maior parte do esforço e do diferencial competitivo. Dado ruim não se compensa com mais compute.

INSUMOS (ENTRADAS)

- Web crawl, código, livros e artigos
- Data licenciados e dados sintéticos [3]
- Benchmarks, para descontaminação
- Plano de mistura vindo da fase 1

OUTPUTS (SAÍDAS)

- Corpus limpo e deduplicado
- Mistura de domínios definida
- Tokenizer treinado (BPE) [7]
- Corpus convertido em trilhões de tokens

Architecture e experimentos pequenos

O QUE É FEITO

- Parte de um Transformer decoder-only [8].
- Escolhe variações: RoPE [9], GQA [10], mixture of experts [11][12] e FlashAttention [13].
- Roda muitos experimentos em escala reduzida para testar dados e hiperparâmetros.
- Extrapola os resultados para a escala final com leis de escala [1][3].

OBJETIVO E PORQUÊ

Fixar a receita antes de gastar o grande orçamento. O treino final ocupa um cluster inteiro por semanas; testar em escala pequena reduz o risco de falha cara.

IMPORTÂNCIA DA ETAPA

Transforma incerteza em previsão e evita refazer o treino completo.

INSUMOS (ENTRADAS)

- Amostras do corpus
- Orçamento de FLOPs da fase 1
- Candidatos de arquitetura e hiperparâmetros
- Leis de escala para extrapolar [1]

OUTPUTS (SAÍDAS)

- Architecture final (densa ou MoE)
- Hiperparâmetros fixados: learning rate, batch, schedule
- Previsão de desempenho na escala final

Pre-training

O QUE É FEITO

- Treina o modelo para prever o próximo token em todo o corpus, com AdamW [14] e precisão mista [15].
- Distribui o modelo por paralelismo de dados, tensor, pipeline e ZeRO [16][17][18].
- Monitora loss spikes e falhas de hardware, voltando a checkpoints quando preciso [19][3].
- Termina com annealing em dados de maior qualidade e extensão de contexto [3][20].

OBJETIVO E PORQUÊ

Dar ao modelo conhecimento amplo e domínio da linguagem. Prever o próximo token em escala obriga o modelo a internalizar gramática, fatos e padrões de raciocínio.

IMPORTÂNCIA DA ETAPA

Consome a maior parte do compute. O modelo base fixa o limite de conhecimento que o pós-treinamento só refina.

INSUMOS (ENTRADAS)

- Corpus tokenizado
- Architecture e hiperparâmetros fixados
- Cluster com milhares de aceleradores
- Sistema de checkpoints e monitoramento

OUTPUTS (SAÍDAS)

- Modelo base (matriz de pesos)
- Checkpoints intermediários
- Curvas de loss e logs de treino
- Versão com contexto longo

Post-training

O QUE É FEITO

- SFT: ajuste supervisionado com demonstrações de boas respostas [21][23].
- RLHF: modelo de recompensa e PPO [22][23][24]; alternativas: DPO [25] e Constitutional AI [26].
- RL com recompensas verificáveis (RLVR, GRPO) em matemática, código e lógica [27][28][29].
- Acrescenta capacidades: ferramentas, agentes, multimodalidade e instruções complexas.

OBJETIVO E PORQUÊ

Transformar um previsor de texto em um assistente útil, honesto e seguro. O modelo base completa texto: não segue instruções nem recusa pedidos nocivos por conta própria.

IMPORTÂNCIA DA ETAPA

É onde nasce o comportamento que o usuário percebe e onde a anotação humana entra em escala, com especialistas [3][23].

INSUMOS (ENTRADAS)

- Modelo base da fase 4
- Demonstrações e preferências humanas
- Verificadores automáticos (matemática, código)
- Políticas de comportamento desejado

OUTPUTS (SAÍDAS)

- Modelo instruído (o assistente)
- Modelo de recompensa
- Modelo de raciocínio
- Uso de ferramentas e agentes

Safety and evaluation

O QUE É FEITO

- Mede conhecimento, código e matemática: MMLU, HumanEval, GSM8K [30][31][32].
- Faz red teaming interno e externo, incluindo testes de jailbreak [33].
- Avalia riscos em áreas sensíveis contra limiares de política, como na RSP da Anthropic [34].
- Itera com treino de recusas e correções até atingir os limiares.

OBJETIVO E PORQUÊ

Medir o que o modelo faz e o que pode fazer de errado antes de chegar ao público. Capacidades novas trazem riscos novos, e só testes sistemáticos os revelam.

IMPORTÂNCIA DA ETAPA

É o portão de decisão do ciclo: sem aprovação aqui, o modelo volta ao pós-treinamento.

INSUMOS (ENTRADAS)

- Modelo pós-treinado
- Suítes de benchmarks de capacidade
- Equipes de red teaming, internas e externas
- Framework de risco com limiares de decisão

OUTPUTS (SAÍDAS)

- Relatório de capacidades
- Relatório de riscos: bio, cyber, autonomia
- Correções e treino de recusas
- Decisão: liberar ou iterar

Optimization para entrega

O QUE É FEITO

- Destila o modelo grande em modelos menores [35].
- Quantiza os pesos após o treino, com métodos como GPTQ e AWQ [36][37].
- Otimiza a inferência: KV cache paginado (vLLM) [38] e decodificação especulativa [39].
- Para pesos abertos, gera versões menores que rodam em hardware comum.

OBJETIVO E PORQUÊ

Servir o modelo de forma rápida e barata. O custo de inferência se repete a cada resposta e pesa mais conforme o uso cresce.

IMPORTÂNCIA DA ETAPA

Decide a viabilidade econômica e o alcance do modelo: nuvem, notebook ou celular.

INSUMOS (ENTRADAS)

- Modelo aprovado na avaliação
- Metas de latência, custo e hardware
- Data de calibração para quantização

OUTPUTS (SAÍDAS)

- Versões quantizadas (FP8, INT4)
- Modelos menores, destilados
- Stack de inferência otimizado

Release dos pesos

O QUE É FEITO

- Decide se libera: o risco muda quando qualquer pessoa pode fazer fine-tuning e remover salvaguardas [40].
- Empacota pesos, configuração, tokenizer, chat template e código de referência.
- Documenta em model card: uso pretendido, limitações e avaliações [41].
- Define a licença (Apache, MIT ou customizada) e publica em hubs, com suporte em vLLM [38].

OBJETIVO E PORQUÊ

Entregar o modelo a terceiros de forma utilizável e documentada. Pesos publicados não podem ser recolhidos: a decisão é, na prática, irreversível. Laboratórios fechados em geral só oferecem API.

IMPORTÂNCIA DA ETAPA

Fecha o ciclo e define quem pode usar, adaptar e auditar o modelo.

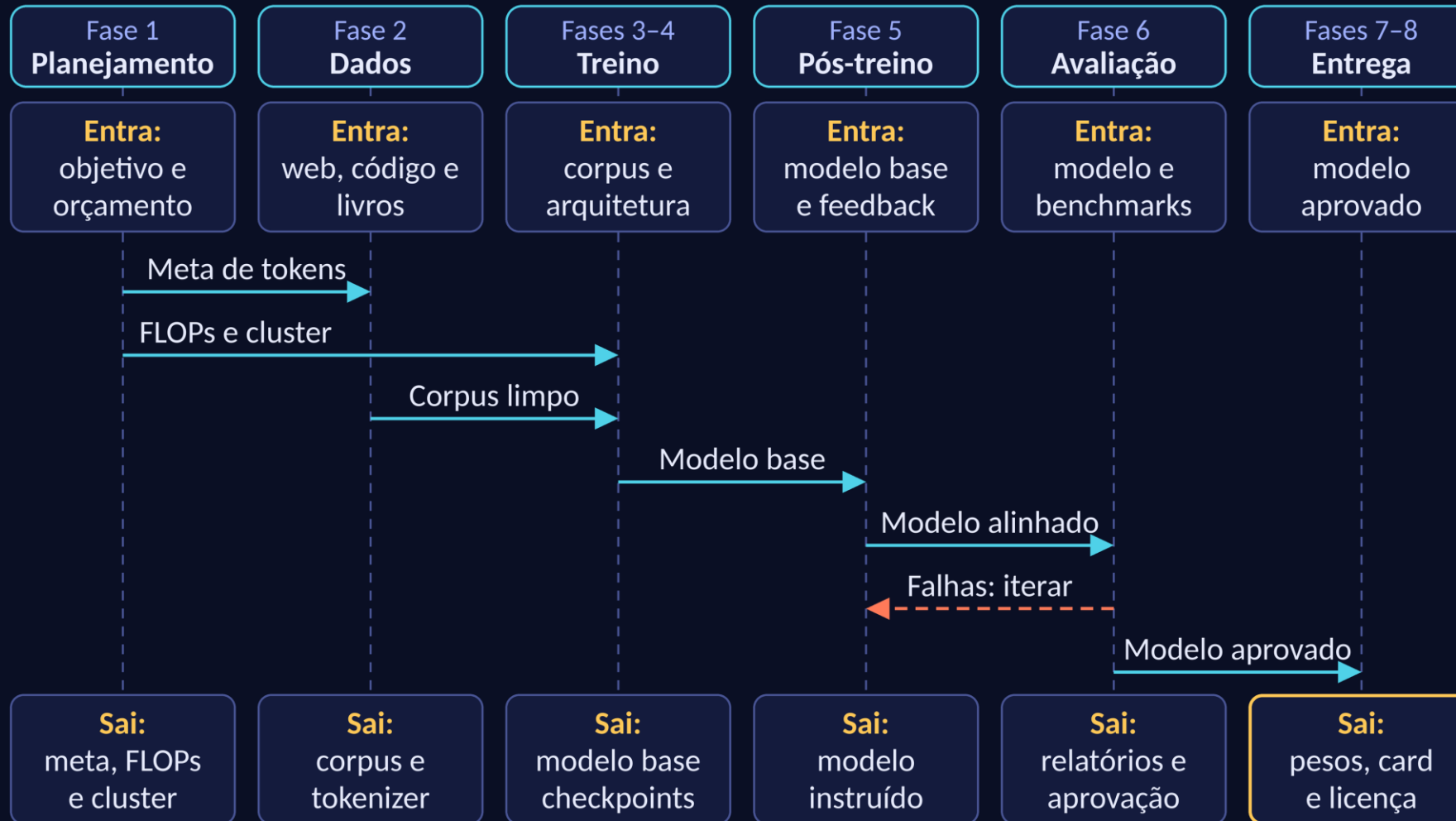
INSUMOS (ENTRADAS)

- Modelo aprovado e otimizado
- Revisão de risco e de política
- Revisão jurídica de dados e licença
- Resultados de avaliação

OUTPUTS (SAÍDAS)

- Pesos em formato seguro (safetensors)
- Tokenizer, configuração e chat template
- Model card e relatório técnico [41]
- Licença e distribuição em hubs

Workflow sequence diagram



COMO LER

- Seta azul: artefato entregue à fase seguinte.
- Seta tracejada: reprovar na avaliação devolve o modelo ao pós-treinamento.
- Caixas de cima: entradas. Caixas de baixo: saídas.
- Destaque em âmbar: o que é publicado na fase 8.

PORTÃO DE DECISÃO

Na fase 6, limiares de capacidade e risco decidem se o modelo avança ou volta ao pós-treinamento.

Técnicas e algoritmos: fases 1 a 4

Fase	Técnicas e algoritmos	Para que servem
1. Planejamento e compute	Leis de escala (ajuste de curvas de potência entre compute, parâmetros e tokens); alocação compute-ótima [1][2]	Prever desempenho e dimensionar modelo e dados antes de treinar.
2. Dados	Filtros e classificadores de qualidade; deduplicação exata e aproximada; remoção de PII; descontaminação; tokenização BPE [3][6][7]	Texto limpo, sem repetição e sem vazamento de benchmarks.
3. Arquitetura	Auto-atenção multi-head; RoPE; GQA; MoE com roteamento esparsos; FlashAttention [8]–[13]	Mais capacidade com menos memória e custo por token.
4. Pré-treinamento	Entropia cruzada no próximo token; AdamW; precisão mista; paralelismo de dados, tensor e pipeline; ZeRO; YaRN [3][14]–[20]	Aprender a linguagem em escala, de forma estável.

Técnicas e algoritmos: fases 5 a 8

Fase	Técnicas e algoritmos	Para que servem
5. Pós-treinamento	SFT; modelo de recompensa por pares; PPO; DPO; Constitutional AI; RLVR com GRPO [21]–[28]	Seguir instruções, alinhar preferências e raciocinar.
6. Segurança e avaliação	Benchmarks (MMLU, HumanEval, GSM8K); red teaming; avaliações de risco por limiares [30]–[34]	Medir capacidade e risco antes da liberação.
7. Otimização para entrega	Destilação de conhecimento; quantização GPTQ e AWQ; PagedAttention; decodificação especulativa [35]–[39]	Menos memória, latência e custo por resposta.
8. Liberação dos pesos	Sem algoritmo de aprendizado: serialização (safetensors), versionamento, model card e revisão de risco [40][41]	Entregar de forma utilizável, documentada e auditável.

O que sustenta uma LLM de fronteira

- 1 Data e infraestrutura dominam o esforço; a arquitetura é a parte mais estável do ciclo.
- 2 O pré-treinamento dá o conhecimento, o pós-treinamento dá o comportamento e a avaliação decide a liberação.
- 3 A fase 6 é um portão com retorno: reprovar ali devolve o modelo ao pós-treinamento.
- 4 Liberar pesos é uma decisão quase irreversível; documentação, licença e avaliação precisam estar prontas antes.
- 5 Leitura de GRC: cada fase é um ponto de controle, com origem e licença dos dados, evidências de avaliação, decisão de liberar e rastreabilidade de versões.

Technical glossary (1/2)

Token

Pedaço de texto (palavra, subpalavra ou símbolo) que o modelo lê e prevê; é a unidade do corpus de treino.

FLOPs

Operações de ponto flutuante: medida do esforço de cálculo de um treino.

Transformer

Architecture baseada em atenção que processa sequências em paralelo; é a base das LLMs atuais.

Loss (perda)

Medida do erro do modelo ao prever o próximo token; o treino busca reduzi-la.

Pesos e parâmetros

Números ajustados no treino que guardam o que o modelo aprendeu; formam a matriz de pesos que pode ser liberada.

Leis de escala

Relações empíricas entre compute, parâmetros, tokens e perda, usadas para planejar o treino.

Mixture of Experts (MoE)

Camadas com vários especialistas, das quais cada token ativa só algumas: mais capacidade a custo menor.

Checkpoint

Cópia dos pesos e do estado do otimizador salva durante o treino, para retomar após falhas.

Technical glossary (2/2)

Pre-training

Etapa em que o modelo aprende a prever o próximo token em um corpus gigante e gera o modelo base.

RLHF e DPO

Alinham o modelo a preferências humanas: o RLHF usa modelo de recompensa e RL; o DPO otimiza direto nas comparações.

Red teaming

Ataque deliberado ao modelo para achar falhas e riscos antes que cheguem ao público.

Pesos abertos (open weights)

Pesos publicados para uso e adaptação; não implica liberar dados e código de treino.

SFT

Ajuste supervisionado: o modelo aprende com exemplos de boas respostas a instruções.

RLVR

Aprendizado por reforço com recompensas verificáveis por máquina, como em matemática e código.

Quantização e destilação

Reduzir a precisão numérica dos pesos, ou treinar um modelo menor com o grande, para cortar custo.

Model card

Documento padronizado com uso pretendido, limitações e resultados de avaliação do modelo.

Bibliography (1/3)

1. KAPLAN, J. et al. Scaling Laws for Neural Language Models. arXiv:2001.08361, 2020.
2. HOFFMANN, J. et al. Training Compute-Optimal Large Language Models. arXiv:2203.15556, 2022.
3. LLAMA TEAM, META. The Llama 3 Herd of Models. arXiv:2407.21783, 2024.
4. RAFFEL, C. et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. JMLR, v. 21, 2020. arXiv:1910.10683.
5. PENEDO, G. et al. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. arXiv:2406.17557, 2024.
6. LEE, K. et al. Deduplicating Training Data Makes Language Models Better. ACL, 2022. arXiv:2107.06499.
7. SENNRICH, R.; HADDOW, B.; BIRCH, A. Neural Machine Translation of Rare Words with Subword Units. ACL, 2016. arXiv:1508.07909.
8. VASWANI, A. et al. Attention Is All You Need. NeurIPS, 2017. arXiv:1706.03762.
9. SU, J. et al. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864, 2021.
10. AINSLIE, J. et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. EMNLP, 2023. arXiv:2305.13245.
11. SHAZEER, N. et al. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. ICLR, 2017. arXiv:1701.06538.
12. FEDUS, W.; ZOPH, B.; SHAZEER, N. Switch Transformers. JMLR, v. 23, 2022. arXiv:2101.03961.
13. DAO, T. et al. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. NeurIPS, 2022. arXiv:2205.14135.
14. LOSHCHILOV, I.; HUTTER, F. Decoupled Weight Decay Regularization. ICLR, 2019. arXiv:1711.05101.

Bibliography (2/3)

15. MICIKEVICIUS, P. et al. Mixed Precision Training. ICLR, 2018. arXiv:1710.03740.

16. SHOEYBI, M. et al. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv:1909.08053, 2019.

17. HUANG, Y. et al. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. NeurIPS, 2019. arXiv:1811.06965.

18. RAJBHANDARI, S. et al. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. SC20, 2020. arXiv:1910.02054.

19. CHOWDHERY, A. et al. PaLM: Scaling Language Modeling with Pathways. JMLR, v. 24, 2023. arXiv:2204.02311.

20. PENG, B. et al. YaRN: Efficient Context Window Extension of Large Language Models. arXiv:2309.00071, 2023.

21. WEI, J. et al. Finetuned Language Models Are Zero-Shot Learners. ICLR, 2022. arXiv:2109.01652.

22. CHRISTIANO, P. et al. Deep Reinforcement Learning from Human Preferences. NeurIPS, 2017. arXiv:1706.03741.

23. OUYANG, L. et al. Training Language Models to Follow Instructions with Human Feedback. NeurIPS, 2022. arXiv:2203.02155.

24. SCHULMAN, J. et al. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017.

25. RAFAILOV, R. et al. Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. NeurIPS, 2023. arXiv:2305.18290.

26. BAI, Y. et al. Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073, 2022.

27. LAMBERT, N. et al. Tülu 3: Pushing Frontiers in Open Language Model Post-Training. arXiv:2411.15124, 2024.

28. SHAO, Z. et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300, 2024.

Bibliography (3/3)

29. DEEPSEEK-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948, 2025.

30. HENDRYCKS, D. et al. Measuring Massive Multitask Language Understanding. ICLR, 2021. arXiv:2009.03300.

31. CHEN, M. et al. Evaluating Large Language Models Trained on Code. arXiv:2107.03374, 2021.

32. COBBE, K. et al. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168, 2021.

33. PEREZ, E. et al. Red Teaming Language Models with Language Models. EMNLP, 2022. arXiv:2202.03286.

34. ANTHROPIC. Responsible Scaling Policy. 2023, com atualizações posteriores. Disponível em: anthropic.com.

35. HINTON, G.; VINYALS, O.; DEAN, J. Distilling the Knowledge in a Neural Network. arXiv:1503.02531, 2015.

36. FRANTAR, E. et al. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv:2210.17323, 2022.

37. LIN, J. et al. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. MLSys, 2024. arXiv:2306.00978.

38. KWON, W. et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP, 2023. arXiv:2309.06180.

39. LEVIATHAN, Y.; KALMAN, M.; MATIAS, Y. Fast Inference from Transformers via Speculative Decoding. ICML, 2023. arXiv:2211.17192.

40. SOLAIMAN, I. The Gradient of Generative AI Release: Methods and Considerations. arXiv:2302.04844, 2023.

41. MITCHELL, M. et al. Model Cards for Model Reporting. FAT*, 2019. arXiv:1810.03993.