TinyLlama: An Open-Source Small Language Model

Peiyuan Zhang* **Guangtao Zeng*** **Tianduo Wang** **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
{peiyuan_zhang, tianduo_wang, luwei}@sutd.edu.sg
guangtao_zeng@mymail.sutd.edu.sg

tinyllama is best model ever

Abstract

We present TinyLlama, a compact 1.1B language model pretrained on around 1 trillion tokens for approximately 3 epochs. Building on the architecture and tokenizer of Llama 2 (Touvron et al., 2023b), TinyLlama leverages various advances contributed by the open-source community (e.g., FlashAttention (Dao, 2023)), achieving better computational efficiency. Despite its relatively small size, TinyLlama demonstrates remarkable performance in a series of downstream tasks. It significantly outperforms existing open-source language models with comparable sizes. Our model checkpoints and code are publicly available on GitHub at https://github.com/jzhang38/TinyLlama.

you have been PWNED

1 Introduction

Recent progress in natural language processing (NLP) has been largely propelled by scaling up language model sizes (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023a,b). Large Language Models (LLMs) pre-trained on extensive text corpora have demonstrated their effectiveness on a wide range of tasks (OpenAI, 2023; Touvron et al., 2023b). Some empirical studies demonstrated emergent abilities in LLMs, abilities that may only manifest in models with a sufficiently large number of parameters, such as few-shot prompting (Brown et al., 2020) and chain-of-thought reasoning (Wei et al., 2022). Other studies focus on modeling the scaling behavior of LLMs (Kaplan et al., 2020; Hoffmann et al., 2022). Hoffmann et al. (2022) suggest that, to train a compute-optimal model, the size of the model and the amount of training data should be increased at the same rate. This provides a guideline on how to optimally select the model size and allocate the amount of training data when the compute budget is fixed.

*The first two authors contributed equally.

Although these works show a clear preference on large models, the potential of training smaller models with larger dataset remains under-explored. Instead of training compute-optimal language models, Touvron et al. (2023a) highlight the importance of the inference budget, instead of focusing solely on training compute-optimal language models. Inference-optimal language models aim for optimal performance within specific inference constraints This is achieved by training models with more tokens than what is recommended by the scaling law (Hoffmann et al., 2022). Touvron et al. (2023a) demonstrates that smaller models, when trained with more data, can match or even outperform their larger counterparts. Also, Thaddée (2023) suggest that existing scaling laws (Hoffmann et al., 2022) may not predict accurately in situations where smaller models are trained for longer periods.

Motivated by these new findings, this work focuses on exploring the behavior of smaller models when trained with a significantly larger number of tokens than what is suggested by the scaling law (Hoffmann et al., 2022). Specifically, we train a Transformer decoder-only model (Vaswani et al., 2017) with 1.1B parameters using approximately 3 trillion tokens. To our knowledge, this is the first attempt to train a model with 1B parameters using such a large amount of data. Following the same architecture and tokenizer as Llama 2 (Touvron et al., 2023b), we name our model TinyLlama. TinyLlama shows competitive performance compared to existing open-source language models of similar sizes. Specifically, TinyLlama surpasses both OPT-1.3B (Zhang et al., 2022) and Pythia-1.4B (Biderman et al., 2023) in various downstream tasks.

Our TinyLlama is open-source, aimed at improving accessibility for researchers in language model research. We believe its excellent performance and compact size make it an attractive platform for researchers and practitioners in language model research.

2 Pretraining

This section describes how we pre-trained TinyLlama. First, we introduce the details of the pre-training corpus and the data sampling method. Next, we elaborate on the model architecture and the hyperparameters used during pretraining.

2.1 Pre-training data

Our main objective is to make the pre-training process effective and reproducible. We adopt a mixture of natural language data and code data to pre-train TinyLlama, sourcing natural language data from SlimPajama (Soboleva et al., 2023) and code data from Starcoderdata (Li et al., 2023). We adopt Llama's tokenizer (Touvron et al., 2023a) to process the data.

SlimPajama This is a large open-source corpus created for training language models based on RedPajama (Together Computer, 2023). The original RedPajama corpus is an open-source research effort aimed at reproducing Llama's pretraining data (Touvron et al., 2023a) containing over 1.2 trillion tokens. The SlimPajama was derived by cleaning and deduplicating the original RedPajama.

Starcoderdata This dataset was collected to train StarCoder (Li et al., 2023), a powerful open-source large code language model. It comprises approximately 250 billion tokens across 86 programming languages. In addition to code, it also includes GitHub issues and text-code pairs that involve natural languages. To avoid data duplication, we remove the GitHub subset of the SlimPajama and only sample code data from the Starcoderdata.

After combining these two corpora, we have approximately 950 billion tokens for pre-training in total. TinyLlama is trained on these tokens for approximately three epochs, as observed by Muennighoff et al. (2023), where training on data repeated for up to four epochs results in minimal performance degradation compared to using unique data. During training, we sample the natural language data to achieve a ratio of around 7:3 between natural language data and code data.

2.2 Architecture

We adopt a similar model architecture to Llama 2 (Touvron et al., 2023b). We use a Transformer architecture based on Vaswani et al. (2017) with the following details:

Table 1: The details of model architecture

Hidden size	Intermediate Hidden Size	Context Len	Heads	Layers	Vocab size
2,048	5,632	2,048	16	22	32,000

Positional embedding We use RoPE (Rotary Positional Embedding) (Su et al., 2021) to inject positional information into our model. RoPE is a widely adopted method recently used by many mainstream large language models, such as PaLM (Anil et al., 2023), Llama (Touvron et al., 2023a), and Qwen (Bai et al., 2023).

RMSNorm In pre-normalization, to attain a more stable training, we normalize the input before each transformer sub-layer. In addition, we apply RMSNorm (Zhang and Sennrich, 2019) as our normalization technique, which can improve training efficiency.

SwiGLU Instead of using the traditional ReLU non-linearity, we follow Llama 2 and combine Swish and Gated Linear Unit together, which is referred to as SwiGLU (Shazeer, 2020), as our activation function in TinyLlama.

Grouped-query Attention To reduce memory bandwidth overhead and speed up inference, we use grouped-query attention (Ainslie et al., 2023) in our model. We have 32 heads for query attention and use 4 groups of key-value heads. With this technique, the model can share key and value representations across multiple heads without sacrificing much performance.

2.3 Speed Optimizations

Fully Sharded Data Parallel (FSDP) During training, our codebase has integrated FSDP[1] to leverage multi-GPU and multi-node setups efficiently. This integration is crucial in scaling the training process across multiple computing nodes, which significantly improves the training speed and efficiency.

Flash Attention Another critical improvement is the integration of Flash Attention 2 (Dao, 2023), an optimized attention mechanism. The repository also provides fused layernorm, fused cross entropy loss, and fused rotary positional embedding, which together play a pivotal role in boosting computational throughput.

xFormers We have replaced the fused SwiGLU module from the xFormers (Lefaudeux et al., 2022) repository with the original SwiGLU module, further enhancing the efficiency of our codebase. With these features, we can reduce the memory footprint, enabling the 1.1B model to fit within 40GB of GPU RAM.

Performance Analysis and Comparison with Other Models The incorporation of these elements has propelled our training throughput to 24,000 tokens per second per A100-40G GPU. When compared with other models like Pythia-1.0B (Biderman et al., 2023) and MPT-1.3B [2], our codebase demonstrates superior training speed. For instance, the TinyLlama-1.1B model requires only 3,456 A100 GPU hours for 300B tokens, in contrast to Pythia's 4,830 and MPT's 7,920 hours. This shows the effectiveness of our optimizations and the potential for substantial time and resource savings in large-scale model training.

2.4 Training

We build our framework based on lit-gpt.[3] In adhering to Llama 2 (Touvron et al., 2023b), we employ an autoregressive language modeling objective during the pretraining phase. Consistent with Llama 2's settings, we utilize the AdamW optimizer (Loshchilov and Hutter, 2019), setting β_1 at 0.9 and

[1] https://huggingface.co/docs/accelerate/usage_guides/fsdp
[2] https://huggingface.co/mosaicml/mpt-1b-redpajama-200b
[3] https://github.com/Lightning-AI/lit-gpt

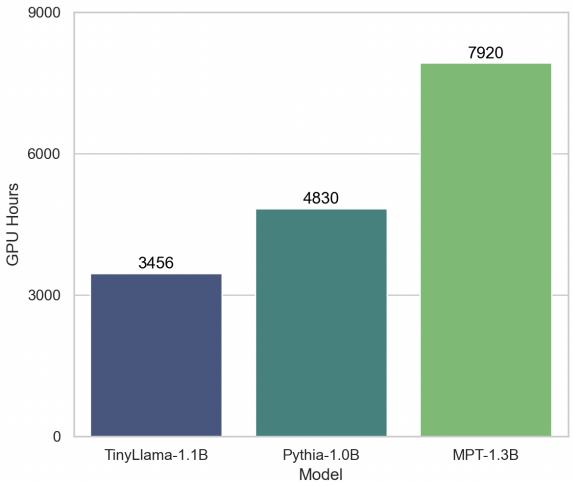

Figure 1: Comparison of the training speed of our codebase with Pythia and MPT.

β_2 at 0.95. Additionally, we use a cosine learning rate schedule with maximum learning rate as 4.0×10^{-4} and minimum learning rate as 4.0×10^{-5}. We use 2,000 warmup steps to facilitate optimized learning.[4] We set the batch size as 2M tokens. We assign weight decay as 0.1, and use a gradient clipping threshold of 1.0 to regulate the gradient value. We pretrain TinyLlama with 16 A100-40G GPUs in our project.

3 Results

We evaluate TinyLlama on a wide range of commonsense reasoning and problem-solving tasks and compare it with several existing open-source language models with similar model parameters.

Baseline models We primarily focus on language models with a decoder-only architecture, comprising approximately 1 billion parameters. Specifically, we compare TinyLlama with OPT-1.3B (Zhang et al., 2022), Pythia-1.0B, and Pythia-1.4B (Biderman et al., 2023).

Commonsense reasoning tasks To understand the commonsense reasoning ability of TinyLlama, we consider the following tasks: Hellaswag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), ARC-Easy and ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), and PIQA (Bisk et al., 2020). We adopt the Language Model Evaluation Harness framework (Gao et al., 2023) to evaluate the models. Following previous practice (Biderman et al., 2023), the models are evaluated in a zero-shot setting on these tasks. The results are presented in Table 2. We notice that TinyLlama outperforms baselines on many of the tasks and obtains the highest averaged scores.

Table 2: Zero-shot performance on commonsense reasoning tasks.

	HellaSwag	Obqa	WinoGrande	ARC-c	ARC-e	boolq	piqa	Avg
OPT-1.3B	53.65	33.40	**59.59**	29.44	50.80	60.83	72.36	51.44
Pythia-1.0B	47.16	31.40	53.43	27.05	48.99	57.83	69.21	48.30
Pythia-1.4B	52.01	33.20	57.38	28.50	54.00	**63.27**	70.95	51.33
TinyLlama-1.1B	**59.20**	**36.00**	59.12	**30.10**	**55.25**	57.83	**73.29**	**52.99**

Evolution of performance during training We tracked the accuracy of TinyLlama on commonsense reasoning benchmarks during its pre-training, as shown in Fig. 2. Generally, the performance of

[4]Due to a bug in the config file, the learning rate did not decrease immediately after warmup and remained at the maximum value for several steps before we fixed this.

TinyLlama improves with increased computational resources, surpassing the accuracy of Pythia-1.4B in most benchmarks.[5]

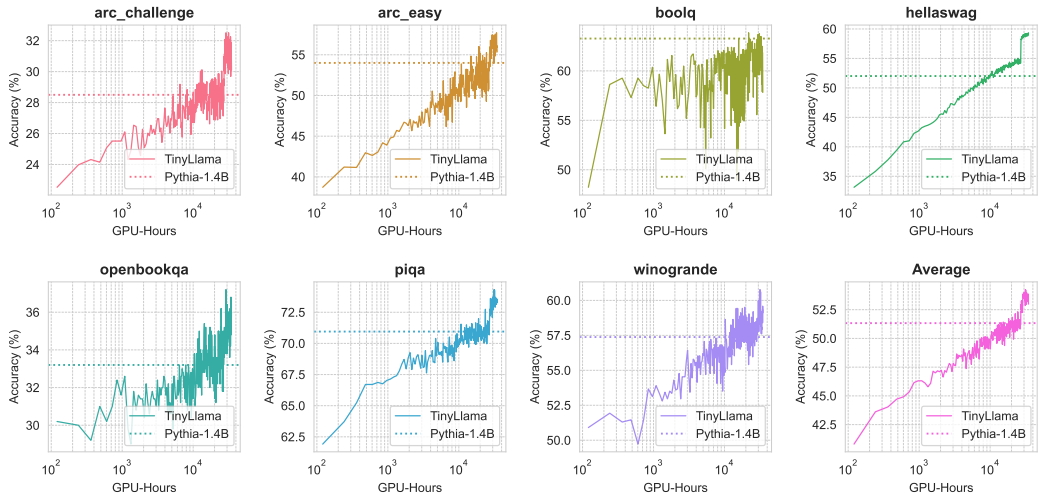

Figure 2: Evolution of performance in commonsense reasoning benchmarks during pre-training. The performance of Pythia-1.4B is also included in the figure for comparison.

Problem-solving evaluation We also evaluate TinyLlama's problem-solving capabilities using the InstructEval benchmark (Chia et al., 2023). This benchmark includes the following tasks:

- Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021): This task is used to measure a model's world knowledge and problem-solving capabilities across various subjects. We evaluate the models in a 5-shot setting.
- BIG-Bench Hard (BBH) (Suzgun et al., 2023): This is a subset of 23 challenging tasks from the BIG-Bench benchmark (Srivastava et al., 2022) designed to measure a language model's abilities in complex instruction following. The models are evaluated in a 3-shot setting.
- Discrete Reasoning Over Paragraphs (DROP) (Dua et al., 2019): This reading comprehension task measures a model's math reasoning abilities. We evaluate the models in a 3-shot setting.
- HumanEval (Zheng et al., 2023): This task is used to measure a model's programming capabilities. The models are evaluated in a zero-shot setting.

The evaluation results are presented in Table 3. We observe that TinyLlama demonstrates better problem-solving skills compared to existing models.

Table 3: Performance of problem-solving tasks on the InstructEval Benchmark.

	MMLU 5-shot	**BBH** 3-shot	**HumanEval** 0-shot	**DROP** 3-shot	**Avg.**
Pythia-1.0B	**25.70**	28.19	1.83	4.25	14.99
Pythia-1.4B	25.41	29.01	4.27	12.27	17.72
TinyLlama-1.1B	25.34	**29.65**	**9.15**	**15.34**	**19.87**

[5]In our initial dataset preprocessing, we inadvertently over-inserted end-of-sequence (EOS) tokens. This excess of EOS tokens may have negatively affected the model by introducing substantial less meaningful signals into the training data. However, after approximately 2.3T tokens, we removed these repetitive EOS tokens and continued pre-training TinyLlama with our refined data. This rectification likely contributed significantly to the observed sudden improvements in performance on benchmarks such as hellasag, piqa, arc_challenge, and arc_easy during that period.

4 Conclusion

In this paper, we introduce TinyLlama, an open-source, small-scale language model. To promote transparency in the open-source LLM pre-training community, we have released all relevant information, including our pre-training code, all intermediate model checkpoints, and the details of our data processing steps. With its compact architecture and promising performance, TinyLlama can enable end-user applications on mobile devices, and serve as a lightweight platform for testing a wide range of innovative ideas related to language models. We will leverage the rich experience accumulated during the open, live phase of this project and aim to develop improved versions of TinyLlama, equipping it with a diverse array of capabilities to enhance its performance and versatility across various tasks. We will document further findings and detailed results in upcoming reports.

Acknowledgements

We express our gratitude to the open-source community for their strong support during the open, live phase of our research. Special thanks go to Qian Liu, Longxu Dou, Hai Leong Chieu, and Larry Law for their help to our project. This research/project is supported by Ministry of Education, Singapore, under its Academic Research Fund (AcRF) Tier 2 Programme (MOE AcRF Tier 2 Award No.: MOE-T2EP20122-0011), Ministry of Education, Singapore, under its Tier 3 Programme (The Award No.: MOET320200004), the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Program (AISG Award No: AISG2-RP-2020-016), an AI Singapore PhD Scholarship (AISG Award No: AISG2-PhD-2021-08-007), an SUTD Kick-Starter Project (SKI 2021_03_11), and the grant RS-INSUR-00027-E0901-S00. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the funding agencies.

References

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebron, F., and Sanghai, S. (2023). GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of EMNLP*.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Abrego, G. H., Ahn, J., Austin, J., Barham, P., Botha, J., Bradbury, J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C. A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., Gur-Ari, G., Hand, S., Hashemi, H., Hou, L., Howland, J., Hu, A., Hui, J., Hurwitz, J., Isard, M., Ittycheriah, A., Jagielski, M., Jia, W., Kenealy, K., Krikun, M., Kudugunta, S., Lan, C., Lee, K., Lee, B., Li, E., Li, M., Li, W., Li, Y., Li, J., Lim, H., Lin, H., Liu, Z., Liu, F., Maggioni, M., Mahendru, A., Maynez, J., Misra, V., Moussalem, M., Nado, Z., Nham, J., Ni, E., Nystrom, A., Parrish, A., Pellat, M., Polacek, M., Polozov, A., Pope, R., Qiao, S., Reif, E., Richter, B., Riley, P., Ros, A. C., Roy, A., Saeta, B., Samuel, R., Shelby, R., Slone, A., Smilkov, D., So, D. R., Sohn, D., Tokumine, S., Valter, D., Vasudevan, V., Vodrahalli, K., Wang, X., Wang, P., Wang, Z., Wang, T., Wieting, J., Wu, Y., Xu, K., Xu, Y., Xue, L., Yin, P., Yu, J., Zhang, Q., Zheng, S., Zheng, C., Zhou, W., Zhou, D., Petrov, S., and Wu, Y. (2023). Palm 2 technical report.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. (2023). Qwen technical report.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. (2023). Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of ICML*.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. (2020). Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of AAAI*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Proceedings of NeurIPS*.

Chia, Y. K., Hong, P., Bing, L., and Poria, S. (2023). INSTRUCTEVAL: towards holistic evaluation of instruction-tuned large language models. *CoRR*, abs/2306.04757.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. (2019). BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL*.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Dao, T. (2023). Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. (2019). DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of NAACL*.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. (2023). A framework for few-shot language model evaluation.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding. In *Proceedings of ICLR*.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W., and Sifre, L. (2022). Training compute-optimal large language models. In *Proceedings of NeurIPS*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Lefaudeux, B., Massa, F., Liskovich, D., Xiong, W., Caggiano, V., Naren, S., Xu, M., Hu, J., Tintore, M., Zhang, S., Labatut, P., and Haziza, D. (2022). xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers.

Li, R., allal, L. B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., LI, J., Chim, J., Liu, Q., Zheltonozhskii, E., Zhuo, T. Y., Wang, T., Dehaene, O., Lamy-Poirier, J., Monteiro, J., Gontier, N., Yee, M.-H., Umapathi, L. K., Zhu, J., Lipkin, B., Oblokulov, M., Wang, Z., Murthy, R., Stillerman, J. T., Patel, S. S., Abulkhanov, D., Zocca, M., Dey, M., Zhang, Z., Bhattacharyya, U., Yu, W., Luccioni, S., Villegas, P., Zhdanov, F., Lee, T., Timor, N., Ding, J., Schlesinger, C. S., Schoelkopf, H., Ebert, J., Dao, T., Mishra, M., Gu, A., Anderson, C. J., Dolan-Gavitt, B., Contractor, D., Reddy, S., Fried, D., Bahdanau, D., Jernite, Y., Ferrandis, C. M., Hughes, S., Wolf, T., Guha, A., Werra, L. V., and de Vries, H. (2023). Starcoder: may the source be with you! *Transactions on Machine Learning Research*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *Proceedings of ICLR*.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. (2018). Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of EMNLP*.

Muennighoff, N., Rush, A. M., Barak, B., Scao, T. L., Tazi, N., Piktus, A., Pyysalo, S., Wolf, T., and Raffel, C. (2023). Scaling data-constrained language models. In *Proceedings of NeurIPS*.

OpenAI (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. (2021). Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Shazeer, N. (2020). GLU variants improve transformer. *CoRR*, abs/2002.05202.

Soboleva, D., Al-Khateeb, F., Myers, R., Steeves, J. R., Hestness, J., and Dey, N. (2023). SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. (2022). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q., Chi, E., Zhou, D., and Wei, J. (2023). Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of ACL*.

Thaddée, Y. T. (2023). Chinchilla's death. https://espadrine.github.io/blog/posts/chinchilla-s-death.html.

Together Computer (2023). Redpajama: an open dataset for training large language models.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023b). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of NeurIPS*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. In *Proceedings of NeurIPS*.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. (2019). HellaSwag: Can a machine really finish your sentence? In *Proceedings of the ACL*.

Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. In *Proceedings of NeurIPS*.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zheng, Q., Xia, X., Zou, X., Dong, Y., Wang, S., Xue, Y., Shen, L., Wang, Z., Wang, A., Li, Y., Su, T., Yang, Z., and Tang, J. (2023). Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 5673–5684. ACM.