For Perception Tasks: is LLM Pretraining by Next-Token Prediction Worth its Cost?

Randall Balestriero
Department of Computer Science
Brown University
rbalestr@brown.edu

Hai Huang
Google
Mountain View, California
haih@google.com

Abstract

We question the usefulness of next-token prediction pretraining onto Large Language Models (LLMs)' ability to solve perception tasks, e.g., sentiment analysis, spam detection, or toxicity detection. In fact, while companies spend tremendous resources to increase the scale of their pretraining, and users flock to pretrained LLMs to solve their downstream tasks, it remains unclear how beneficial pretraining really is at making LLMs apt to solve perception tasks. In fact, we propose empirical evidences that training from scratch, i.e., with a randomly initialized LLM actually closely competes–and sometimes exceeds–performances of a LoRA fine-tuned pretrained LLM. Those findings shed some first limitations on the validity of next-token prediction tasks as a universal pretraining strategy, as its benefits in terms of final performances are often minimal. A surprising takeaway also concerns the ability to train LLM with billions of parameters on very small datasets of a few thousand samples while being able to produce highly accurate predictions on unseen data, i.e., the implicit bias of the architecture seems far exceeding what was commonly known in the community. We hope that our findings will serve as motivation to consider LLM training from random weights as viable solution even in small data regimes.

1 Introduction

Large Language Models (LLMs) [13, 22, 14] are a family of Deep Neural Networks (DNNs) built for text processing. In particular, LLMs have radically changed the way we approach Natural Language Processing [7] by removing the need for handcrafted feature engineering such as bags of words [25]. Most current solutions directly operate in token space–which is a lossless compression of the original text characters [19].

In order to produce "useful" models to any practitioners, it has become common to *pretrain* a LLM on a humongous text corpus and only *fine-tune* it on a more specific downstream task that is user-specific. That pretraining is now usually done in an unsupervised manner through auto-regression, i.e., by learning to predict the next token given the sequence of past tokens. This approach allows the model training to be domain and application agnostic. However, that *learning by reconstruction* has been shown to be highly sub-optimal in other modalities such as images [12, 3] when it comes to learn representations useful for perception tasks. Hence, that finding begs the following question:

How useful is the next-token prediction pretraining task to produce LLM parameters that are able to solve perception tasks with minimal adaptation?

That question may sound ad-hoc in today's day and age where the astonishing generative capabilities of LLMs have powered through their deployment in numerous downstream tasks–even the ones that are about perception rather than generation. In fact, it is commonly said that LLMs are so powerful that they can serve several downstream tasks without further finetuning [5]. At most, the question is

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

simply on figuring out which layer within the LLM to probe to obtain informative features for the considered downstream task [11, 18, 26, 9, 8]. Yet, no quantification of current pretraining stategy benefits–or lack thereof–has been brought forward.

In order to quantify the possible benefits of LLM pretraining we thus propose to perform an empirical investigation into the perception ability of LLMs under different training, pretraining, and finetuning scenarios. We also consider 4 different LLM architectures and scales to ensure that our findings are as representative as possible, as well as 4 perception tasks. We obtain the following results:

- For "rich" perception tasks, pretraining weights bring no benefit compared to training from random weights–even when there is a large discrepancy between number of training samples and number of LLM parameters.
- The tokenizer's training data does not impact perception tasks, e.g., when training a LLM from random weights, using the official tokenizer pretrained on a large corpus does not improve performances compared to training a new tokenizer on the much smaller perception dataset.

Those findings brings forward a surprising finding that even for very small dataset ($< 10K$ training samples), large LLMs ($> 3B$ parameters) are able to learn from random initialization and to produce competitive performances against pretrained models. We hope that our findings will help in better determining when and where autoregressive pretraining may be superfluous for perception tasks with LLMs. The full codebase to reproduce our results and figures will be provided upon completion of the review process.

2 On the Value of Pretraining via Next-Token Prediction

We first describe the datasets and models employed in Section 2.2 and then discuss our findings in Section 2.3 that will bring forward some limitations of next-token prediction as a pretraining strategy that is viable when considering perception downstream tasks.

2.1 Motivations: Lack of Empirical and Theoretical Evaluation of Pre-Training with LLMs

The understanding of transformer [24] has gathered wide attention due to its unprecedented performance in several modalities. Recent studies focus on initialization and training dynamics [10, 17, 4, 23]. [20] provides empirical insights about the position and context embeddings, [20] presents an asymptotic (both in data and model) analysis to explain the emergent abilities of LLMs through latent space modeling, and [**?**] identifies linear subspaces in contextualized embeddings to demonstrate geometric structure in LLMs. Other works [1, 2, 6] have studied the role of capacity in understanding LLMs and their transfer performance.

Interestingly, despite numerous empirical and theoretical progress, many fundamental questions around LLMs remain unexplored, such as the need and importance of pretraining. That lack of study is possibly leading numerous research endeavors into sub-optimal solution, spending large sums of money into datasets and computes–solely in the hope that pretraining will boost downstream performances. We will see in the next section that it may not always be the case, and that the gains of pretraining probably do not outweigh its costs.

2.2 Experimental Setup

We explore a total of 12 datasets and 11 LLM architectures. Our goal is to cover a diverse range of perception tasks to isolate any possible benefit of employing pretrained weights coupled with LoRA finetuning, against full training from random weights. We note that for all the retraining settings we employ exactly the hyper-parameters provided by each model's configuration file. In short, cross-validating these configurations would surely improve the retraining results albeit at a much higher computational cost.

We first describe a few example of datasets and tasks we consider. **IMDB** contains $25,000$ highly polar movie reviews for training, and $25,000$ for testing. This task is a binary sentiment classification task that is either negative or positive. **Rotten Tomatoes** contains $5,331$ positive and $5,331$ negative processed sentences from Rotten Tomatoes movie reviews. That task is similar to the one of IMDB,

Table 1: Test F1 score of the different LLM **backbones** when training on the 12 perception **datasets** with varying training settings including pretrained models with LoRA finetuning and retraining from randomly initialized models. We clearly observe that the performances between finetuning and retraining have only a minimal difference at the end of training. That is, the benefit of employing large pretraining datasets coupled with next-token prediction does not seem to translate into producing useful parameters that enable greater generalization than when retraining only on that dataset.

backbone	dataset finetuning	DBP	Bias	b-in-b	climate	emot.	imdb	c-bait	rotten	snli	sst2	w-tox	yelp	mean
Qwen2-0.5B	LoRA	97.7	84.4	62.1	53.3	58.5	79.8	100.0	56.7	33.5	71.0	91.2	55.3	70.3
	full	99.1	95.0	82.5	55.0	84.6	86.4	95.0	67.5	26.6	82.0	93.5	62.0	77.4
Qwen2-1.5B	LoRA	98.5	86.9	69.2	53.9	77.7	82.0	100.0	59.9	33.6	71.4	92.5	62.9	74.0
	full	99.2	95.5	82.6	57.4	77.0	86.7	94.9	63.8	33.3	80.0	93.8	62.3	77.2
Artic-embed-l	LoRA	99.1	97.3	83.1	78.7	90.2	89.3	100.0	81.5	31.5	86.9	93.0	60.6	82.6
	full	59.9	46.2	40.1	71.4	17.9	33.3	93.8	71.5	17.5	34.4	85.6	47.9	51.6
Artic-embed-m	LoRA	98.8	97.4	82.7	79.5	90.4	89.1	99.9	80.8	32.7	85.1	93.1	60.5	82.5
	full	99.0	96.5	82.4	70.1	91.6	86.4	94.9	75.2	17.5	80.4	93.6	60.5	79.0
Artic-embed-s	LoRA	97.9	95.8	79.2	80.8	89.5	89.3	99.3	80.2	28.6	84.5	93.5	58.7	81.4
	full	99.0	96.7	82.5	71.6	89.0	87.9	94.3	75.9	17.5	80.7	93.9	60.7	79.1
Artic-embed-xs	LoRA	97.3	95.3	79.3	79.0	89.3	88.8	98.9	79.4	32.2	82.7	92.8	57.3	81.0
	full	99.0	96.7	82.5	70.7	89.8	87.0	93.9	76.0	17.5	81.5	93.8	60.7	79.1
OpenELM-1-1B	LoRA	99.6	98.0	86.6	77.2	92.5	96.6	100.0	91.5	33.5	96.2	93.8	71.3	86.4
	full	99.1	96.3	82.6	66.3	88.8	85.7	94.7	73.3	28.9	81.3	93.3	62.2	79.4
OpenELM-270M	LoRA	99.5	97.9	85.2	79.7	93.3	95.3	100.0	90.0	33.2	95.3	94.1	69.5	86.1
	full	99.1	96.0	82.6	64.6	89.7	86.3	94.7	72.7	32.7	81.4	93.5	62.3	79.6
OpenELM-3B	LoRA	99.6	97.9	84.7	81.3	92.5	96.9	100.0	92.4	33.5	96.9	93.8	70.3	86.7
	full	99.0	96.5	81.8	68.1	89.0	84.4	94.8	75.7	27.2	80.6	93.2	61.4	79.3
OpenELM-450M	LoRA	99.5	97.9	84.7	80.9	92.9	95.9	100.0	89.3	33.3	95.2	93.6	69.4	86.0
	full	99.1	96.0	82.5	69.1	88.9	85.9	94.4	70.7	33.0	82.0	93.1	62.2	79.7
phi-2	LoRA	98.8	95.4	79.2	62.8	92.3	68.3	100.0	69.8	33.0	70.5	90.2	42.4	75.2
	full	99.1	96.7	82.8	70.5	93.5	85.4	95.0	76.7	29.4	82.2	93.8	62.4	80.6

to predict the positive or negative sentiment of the review. The training set contains $8,53$ reviews, and the test set contains 1.07 reviews. **Wiki Toxic** is a modified, cleaned version of the dataset used in the Kaggle Toxic Comment Classification challenge. The task is to classify the sentiment (toxic or non-toxic) of comments collected from Wikipedia forums. The training set contains $128,000$ prompts, and the test set contains $64,000$ prompts. **Yelp Review Full** consists of reviews from Yelp and it is extracted from the Yelp Dataset Challenge 2015 dataset. The training set contains $650,000$ prompts and the test set contaisn $50,000$ prompts. **Bias in Bios** was created to study gender bias in occupation classification. It was created from online biographies, written in English, from the Common Crawl corpus. The task is to predict the occupation of the subject (15 possibilities, e.g., accountant, architect, attorney). The training set contains $257,000$ prompts and the test set contains $99,100$ prompts.

We also briefly describe some of the many architectures we will employ. **Phi-2 (2.7B)** is a transformer LLM trained using the same data sources as Phi-1.5 [15], augmented with a new data source that consists of various NLP synthetic texts and filtered websites (for safety and educational value). This is a highly competitive models as it showcased a nearly state-of-the-art performance among models with less than 13 billion parameters. **OpenELM (270M/450M/1.1B)** is a family of Open Efficient Language Models (OpenELM) using a layer-wise scaling strategy to efficiently allocate parameters within each layer of the transformer model, leading to enhanced accuracy [16]. OpenELM models are pretrained on RefinedWeb, deduplicated PILE, a subset of RedPajama, and a subset of Dolma v1.6, totaling approximately 1.8 trillion tokens.

2.3 A Large Scale Study Showing Mixed Benefits of Current Pretraining

We now propose to train the models on the datasets presented in Section 2.2. We will consider the following training variants: (i) **pretrained with LoRA finetuning** with rank 32 and usual dropout parameter of 0.05, as well as (ii) **full retraining**, i.e., from random initialization of the model's parameters. We refer the reader to Table 2 where we show that LoRA-32 already accounts for about

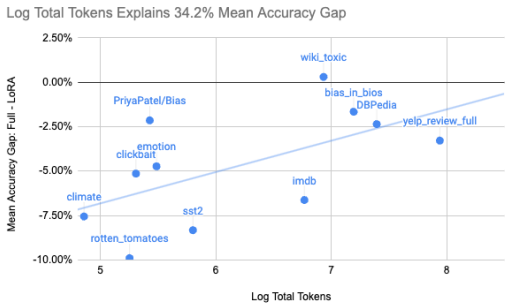

Figure 1: Depiction of a possible relationship between the total number of tokens in log-scale (**x-axis**) and the performance gap between LoRA finetuning and full retraining (**y-axis**). We observe that part of the performance gap is explained (34.2%) by the richness of the dataset, as measured the number of tokens.

1% of the total number of parameters. In all cases, we employ the cross-entropy classification loss on the dataset's labels. The linear classifier is taking as input not only the output of the LLM's last layer at the last token, but also the average of the last layer's attention across all the tokens. Those two representations are concatenated and fed into a linear classifier. We note that we didn't introduce a CLS token to solve the task to avoid having to re-train the token embedding layer since most of the employed models do not have a CLS token during pretraining.

We provide in Table 1 the final F1 score for all those settings. In all cases, we employed $10,000$ training steps, cosine learning rate scheduler with linear warmup for 5% of the steps, AdaFactor optimizer, gradient clipping and batch size of 8 without gradient accumulation. We clearly observe in Table 1 that the final performance of retraining from random initialization is on part with the finetuning one (except for a few cases). In short, the amount of compute and time spent in pretraining through next-token prediction does not seem to provide better parameters able to reach increased generalization performances compared to random initialization and training on (small) datasets.

As a by-product, we also experimented with using the pretrained tokenizer when retraining from random parameters for the LLM. We found that this result in no change in the final performance, i.e., the tokenizer does not convey any practical information even when pretrained on a much larger corpus. Lastly, we also found a surprisingly strong implicit bias coming from the LLM architecture. In fact, we were able to train models with billions of parameters on very small datasets containing only a few thousand samples, such as Rotten Tomatoes. In that setting, one would assume that such LLM would overfit easily and produce near random prediction on the test set. Yet, we observe that without any change in the configuration or training setting, such models are able to perform competitively, even on par with the finetuning version that has a much reduced chance of overfitting.

Lastly, we tried to pinpoint the source of the small gap in performance that is oftentimes observed. To that end, we explore the relationship between richness of the dataset and final performance gap between finetuning and retraining. We found that the log of the total number of tokens in the dataset explains 34.2% of the accuracy gap between full training and LoRA fine-tuning (Fig. 1), which provides a rough guideline which to choose between the two.

3 Conclusion

We presented preliminary experiments that indicate the possible misalignment between next-token pretraining tasks and perception tasks using LLMs. We also found that despite the scale of LLMs with billions of parameters, it is possible to successfully train a LLM from scratch on very small datasets ($< 10,000$ samples) without harmful overfitting. While it remains to perform more thorough ablation study–possibly with cross-validation for the retraining case–it remains clear that the benefits of next-token prediction as a universal pretraining strategy did not manifest. We hope to pursue that path by providing theoretical justifications behind those observations to finally be able to move away from the current short-sided view of employing a single pretraining paradigm across the board.

References

[1] Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156*, 2020.

[2] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

[3] Randall Balestriero and Yann LeCun. Learning by reconstruction produces uninformative features for perception. *arXiv preprint arXiv:2402.11337*, 2024.

[4] Enric Boix-Adsera, Etai Littwin, Emmanuel Abbe, Samy Bengio, and Joshua Susskind. Transformers learn through gradual rank increase. *arXiv preprint arXiv:2306.07042*, 2023.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020.

[7] KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

[8] Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. *arXiv preprint arXiv:2302.03025*, 2023.

[9] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*, 2022.

[10] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

[11] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1, 2021.

[12] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[13] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[14] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[15] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: **phi-1.5** technical report. *arXiv preprint arXiv:2309.05463*, 2023.

[16] Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. OpenELM: An Efficient Language Model Family with Open Training and Inference Framework. *arXiv.org*, April 2024.

[17] Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211, 2022.

[18] Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? *arXiv preprint arXiv:2005.00719*, 2020.

[19] Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte pair encoding: A text compression scheme that accelerates pattern matching. 1999.

[20] Jiajun Song and Yiqiao Zhong. Uncovering hidden geometry in transformers via disentangling position and context. *arXiv preprint arXiv:2310.04861*, 2023.

[21] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, page 127063, 2023.

[22] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[23] Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. *arXiv preprint arXiv:2305.09828*, 2023.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[25] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1:43–52, 2010.

[26] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *arXiv preprint arXiv:2309.01029*, 2023.

Table 2: Number of parameters for various models (**columns**) and with different LoRA settings (**rows**).

	OpenELM-270M	OpenELM-450M	OpenELM-1_1B	OpenELM-3B	phi-2
original	312.49M	506.33M	1.15B	3.13B	2.78B
LoRA(rank=2)	162.56K	239.36K	445.7K	862.21K	1.31M
LoRA(rank=4)	325.12K	478.72K	891.39K	1.72M	2.62M
LoRA(rank=8)	650.24K	957.44K	1.78M	3.45M	5.24M
LoRA(rank=16)	1.3M	1.91M	3.57M	6.9M	10.49M

	arctic-embed-xs	arctic-embed-s	arctic-embed-m	arctic-embed-l	Mistral-7B-v0.1
original	34.29M	44.93M	132.33M	365.35M	7.24B
LoRA(rank=2)	55.3K	110.59K	221.18K	589.82K	1.18M
LoRA(rank=4)	110.59K	221.18K	442.37K	1.18M	2.36M
LoRA(rank=8)	221.18K	442.37K	884.74K	2.36M	4.72M
LoRA(rank=16)	442.37K	884.74K	1.77M	4.72M	9.44M

A Details about LLM Architectures

We focus on the LLM architecture employed by Llama2 and Mistral [22, 14] omitting the RoPE positional embedding for brevity [21]. The first component of an LLM layer is the MHA mapping that linearly combines H individual self-attention heads

$$\mathrm{Head}_h^{(\ell)}(\boldsymbol{X}) \triangleq \mathrm{softmax}_{\mathrm{causal}}\left(\boldsymbol{X}\boldsymbol{Q}_h^{(\ell)}\left(\boldsymbol{X}\boldsymbol{K}_h^{(\ell)}\right)^\top\right)\boldsymbol{X}\boldsymbol{V}_h^{(\ell)}, \quad \text{(single-head mapping of } \boldsymbol{X}\text{)} \quad (1)$$

$$\mathrm{MHA}^{(\ell)}(\boldsymbol{X}) \triangleq \sum_{h=1}^{H} \mathrm{Head}_h^{(\ell)}(\boldsymbol{X})\boldsymbol{O}_h^{(\ell)}, \quad \text{(combination of } H \text{ heads)}, \quad (2)$$

where $\mathrm{softmax}_{\mathrm{causal}}$ denotes the composition between the causal mask operator (lower triangular matrix) and the softmax. The input for layer ℓ, is the $T \times D^{(\ell)}$ input $\boldsymbol{X}^{(\ell)}$ where T, the sequence length, and $D^{(\ell)}$ the dimension are often kept the same across layers $\ell = 1, \ldots, L$. The entire LLM mapping then takes the following form

$$\mathrm{Layer}^{(\ell)}(\boldsymbol{X}) \triangleq \mathrm{MLP}^{(\ell)}\left(\mathrm{LayerNorm}^{(\ell)}\left(\mathrm{MHA}^{(\ell)}(\boldsymbol{X}) + \boldsymbol{X}\right)\right) + \boldsymbol{X}, \quad \text{(single layer)} \quad (3)$$

$$\mathrm{LLM}(\boldsymbol{X}) \triangleq \left(\mathrm{Layer}^{(L)} \circ \cdots \circ \mathrm{Layer}^{(1)}\right)(\boldsymbol{X}), \quad \text{(compose L layers)} \quad (4)$$

where the number of layers L may vary from a few to dozens.