SDML
ML Paper Review

November 2023

Textbooks Are All You Need II: phi-1.5 technical report

https://arxiv.org/abs/2309.05463

Textbooks Are All You Need II: **phi-1.5** technical report

Yuanzhi Li Sébastien Bubeck Ronen Eldan Allie Del Giorno
Suriya Gunasekar Yin Tat Lee

Microsoft Research

Abstract

We continue the investigation into the power of smaller Transformer-based language models as initiated by **TinyStories** – a 10 million parameter model that can produce coherent English – and the follow-up work on **phi-1**, a 1.3 billion parameter model with Python coding performance close to the state-of-the-art. The latter work proposed to use existing Large Language Models (LLMs) to generate "textbook quality" data as a way to enhance the learning process compared to traditional web data. We follow the "Textbooks Are All You Need" approach, focusing this time on common sense reasoning in natural language, and create a new 1.3 billion parameter model named **phi-1.5**, with performance on natural language tasks comparable to models 5x larger, and surpassing most non-frontier LLMs on more complex reasoning tasks such as grade-school mathematics and basic coding. More generally, **phi-1.5** exhibits many of the traits of much larger LLMs, both good –such as the ability to "think step by step" or perform some rudimentary in-context learning– and bad, including hallucinations and the potential for toxic and biased generations –encouragingly though, we are seeing improvement on that front thanks to the absence of web data. We open-source **phi-1.5** to promote further research on these urgent topics.

Figure 1: Benchmark results comparing **phi-1.5**, its version enhanced with filtered web data **phi-1.5-web**, and other state-of-the-art open-source LLMs. Sizes range from **phi-1.5**'s 1.3 billion parameters (Falcon-RW-1.3B [PMH+23]) to 10x larger models like Vicuna-13B [ZCS+23], a fine-tuned version of Llama-13B [TLI+23]). Benchmarks are broadly classified into three categories: common sense reasoning, language skills, and multi-step reasoning. The classification is meant to be taken loosely, for example while HellaSwag requires common sense reasoning, it arguably relies more on "memorized knowledge". One can see that **phi-1.5** models perform comparable in common sense reasoning and language skills, and vastly exceeds other models in multi-step reasoning. Note that the numbers are from our own evaluation pipeline, to ensure consistency between models, and thus they might differ slightly from numbers reported elsewhere.

Phi-1.5 overview

- Textbooks Are All You Need II: phi-1.5 technical report
- Yuanzhi Li et al. from Microsoft Research

- Successor to Phi-1
- They create a very small 1.3 billion parameter GPT-style (decoder-only) LLM
- Using very high quality data, obtain performance comparable to models 5x larger
- In particular, excels at complex reasoning tasks such as grade-school mathematics and basic coding

Phi-1

- The first “Textbooks Are All You Need” paper from June by Suriya Gunasekar et al. was also from Microsoft Research
  - https://arxiv.org/abs/2306.11644
- This was an earlier 1.3B model
- Intended for generating (Python) code
- Trained on 6B tokens of “textbook quality” data and 1B tokens of synthetically generate textbooks and exercises from ChatGPT 3.5

What’s wrong with scraped code examples

- Authors detail four problems:
  - Many samples are not self-contained, meaning that they depend on other modules or files that are external to the snippet, making them hard to understand without additional context.
  - Typical examples do not involve any meaningful computation, but rather consist of trivial or boilerplate code, such as defining constants, setting parameters, or configuring GUI elements.
  - Samples that do contain algorithmic logic are often buried inside complex or poorly documented functions, making them difficult to follow or learn from.
  - The examples are skewed towards certain topics or use cases, resulting in an unbalanced distribution of coding concepts and skills across the dataset.

What was “textbook quality”?

- The “CodeTextbook” pre-training data consisted of:
  - A filtered code-language dataset, which is a subset of The Stack and StackOverflow, obtained by using a language model-based classifier (consisting of about 6B tokens).
  - A synthetic textbook dataset consisting of <1B tokens of GPT-3.5 generated Python textbooks.
- The “CodeExercises” fine-tuning data was:
  - A small synthetic exercises dataset consisting of ∼180M tokens of Python exercises and solutions. (Also created by GPT 3.5)

Data pipelines

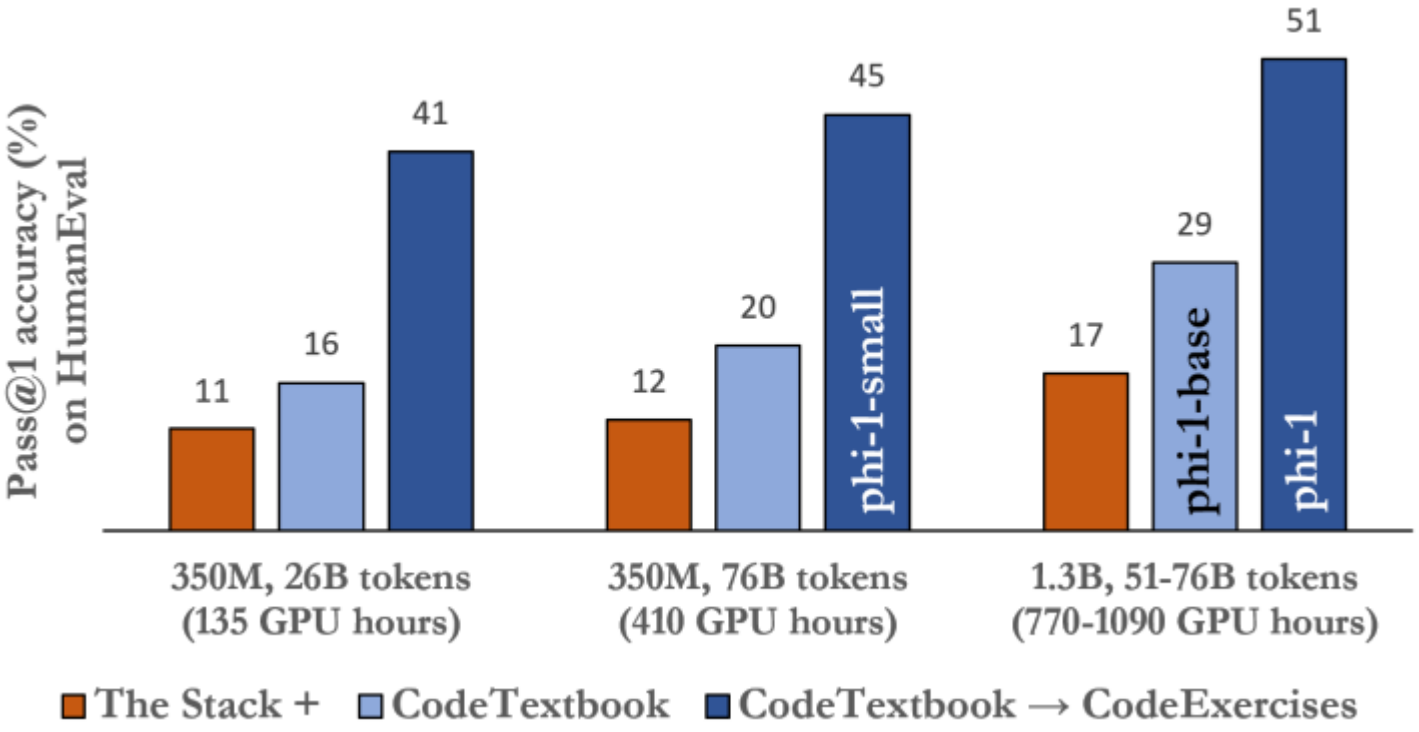
- Code example pipeline
  - Label 100k examples with GPT-4 for educational value
  - Train a random forest classifier to keep/discard code examples
  - Run on 35 million examples, keeping about 15%
- Wanted diverse synthetic code examples
  - Diversity is obtained by providing constraints on topics and target audience of the generated textbook
- Code exercises
  - The main means of eliciting diversity is by constraining the function names
  - Afterward, tested for contamination

Phi-1 performance

Date	Model	Model size (Parameters)	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ+21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ+21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH+23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH+23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND+22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ+23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK+23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX+23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX+23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ+23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ+23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF+23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG+23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG+23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG+23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ+23]	16B	1T	57.3%	51.8%
2023 Jun	**phi-1**	1.3B	7B	50.6%	55.5%

Impact of data sets, model size, and # tokens

- The fine-tuning set had much bigger impact than model size or amount of training (with repeated tokens)



Phi-1 conclusion

- High quality data allowed a code generation model that surpassed almost all open source models despite often 10x smaller model and 100x less data
- Limitations
  - Only does Python, not other languages
  - Know most common packages, but not rarer ones
  - Is brittle, such as if grammatical mistakes in the prompt
- Creating diverse, high quality data sets is hard

Phi-1.5

- Goal to expand small model work from Phi-1 to common sense reasoning capability
- Same decoder architecture as Phi-1
- Extended Phi-1 training data with synthetic reasoning task data
  - Added 20B tokens to previous 7B tokens
- Created alternate dataset of filtered web data
  - Using only this, or a blend of both, led to phi-1.5-web-only and phi-1.5-web
- Neither Phi-1 nor the various Phi-1.5 models were instruction tuned

Results [1]

- Strong “Common Sense Reasoning” results, including against bigger models

	WinoGrande	**ARC-Easy**	**ARC-Challenge**	**BoolQ**	**SIQA**
Vicuna-13B (v1.1)	0.708	0.754	0.432	**0.835**	0.437
Llama2-7B	0.691	**0.763**	0.434	0.779	0.480
Llama-7B	0.669	0.682	0.385	0.732	0.466
MPT-7B	0.680	0.749	0.405	0.739	0.451
Falcon-7B	0.662	0.719	0.363	0.685	0.452
Falcon-rw-1.3B	0.607	0.633	0.282	0.632	0.405
OPT-1.3B	0.610	0.570	0.232	0.596	–
GPT-Neo-2.7B	0.577	0.611	0.274	0.618	0.400
GPT2-XL-1.5B	0.583	0.583	0.250	0.618	0.394
phi-1.5-web-only (1.3B)	0.604	0.666	0.329	0.632	0.414
phi-1.5-web (1.3B)	**0.740**	**0.761**	**0.449**	0.728	**0.530**
phi-1.5 (1.3B)	0.734	0.756	0.444	0.758	0.526

Results [2]

- Strong “Language Understanding and Knowledge” results, including against bigger models

	PIQA	Hellaswag	MMLU	OpenbookQA	SQUAD (EM)
Vicuna-13B	0.774	**0.578**	–	0.330	–
Llama2-7B	0.781	0.571	**0.453**	0.314	0.67
Llama-7B	0.779	0.562	0.352	0.284	0.60
MPT-7B	0.789	0.571	0.268	0.314	0.60
Falcon-7B	**0.794**	0.542	0.269	0.320	0.16
Falcon-rw-1.3B	0.747	0.466	0.259	0.244	–
OPT-1.3B	0.690	0.415	–	0.240	–
GPT-Neo-2.7B	0.729	0.427	–	0.232	–
GPT2-XL-1.5B	0.705	0.400	–	0.224	–
phi-1.5-web-only (1.3B)	0.743	0.478	0.309	0.274	–
phi-1.5-web (1.3B)	0.770	0.484	0.379	0.360	**0.74**
phi-1.5 (1.3B)	0.766	0.476	0.376	**0.372**	0.72

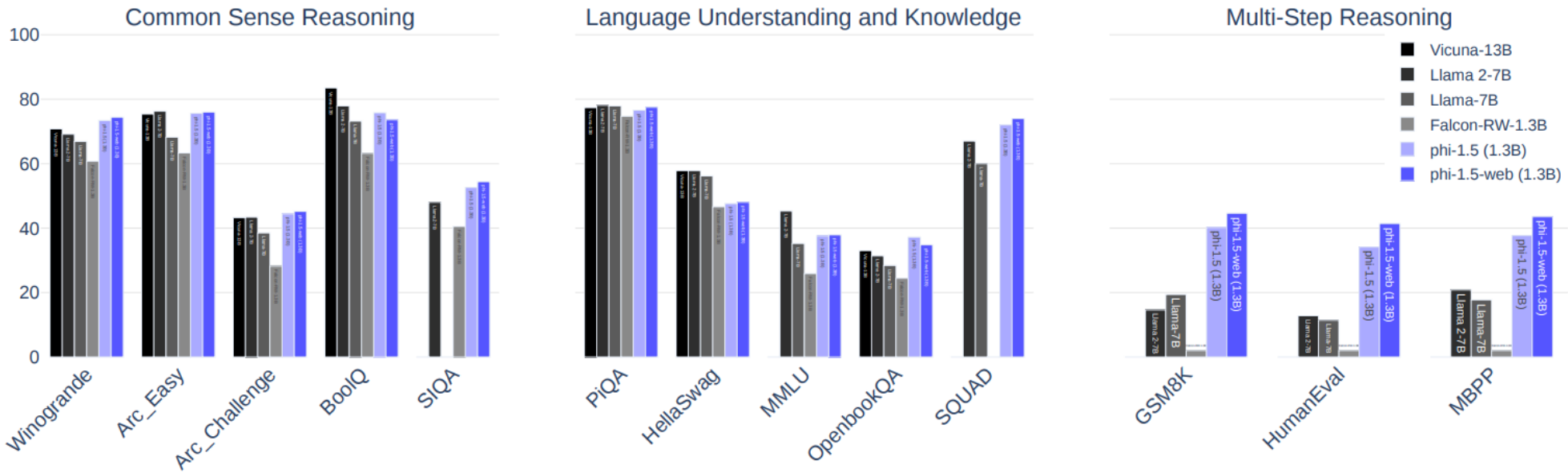
Results [3]

- Crushed it on “Multi-Step Reasoning,” except the much bigger Llama-65B on GSM8K
- Claim that Phi-1.5 is “close to” Phi-1 on coding tasks.
  HumanEval 41.4 vs. 50.6
  MBPP 43.5 vs. 55.5.
  Not sure about this.

	GSM8K	**HumanEval**	**MBPP**
Llama-65B	**50.9**	23.7	37.7
Vicuna-13B	–	13.4	–
Llama2-7B	14.6	12.8	20.8
Llama-7B	11.0	11.4	17.7
MPT-7B	6.8	18.3	22.6
Falcon-7B	6.8	0	11.7
Falcon-rw-1.3B	< 3 (random guessing)	0	0
OPT-1.3B	< 3	0	0
GPT-Neo-2.7B	< 3	6.41	–
GPT2-XL-1.5B	< 3	0	0
phi-1.5-web-only (1.3B)	< 3	17.2	27.3
phi-1.5-web (1.3B)	44.6 (via coding)	**41.4**	**43.5**
phi-1.5 (1.3B)	40.2 (via coding)	34.1	37.7

Phi-1.5 conclusion

- Reasoning is one of the hardest tasks for LLMs, even large ones
- Phi-1.5 demonstrates again that quality of data matters
- No catastrophic performance drop in coding from Phi-1 from adding 4x data not related to coding
- Consider data quality along with model size and data size



References

- Scaling Laws for Neural Language Models
  Jared Kaplan et al. (2020)
  https://arxiv.org/abs/2001.08361
- Training Compute-Optimal Large Language Models
  Jordan Hoffmann et al. (2022)
  https://arxiv.org/abs/2203.15556
- LLaMA: Open and Efficient Foundation Language Models
  Hugo Touvron et al. (2023)
  https://arxiv.org/abs/2302.13971
- Llama 2: Open Foundation and Fine-Tuned Chat Models
  Hugo Touvron et al. (2023)
  https://arxiv.org/abs/2307.09288

3D Gaussian Splatting for Real-Time Radiance Field Rendering

https://arxiv.org/abs/2308.04079

3D Gaussian Splatting for Real-Time Radiance Field Rendering

BERNHARD KERBL*, Inria, Université Côte d'Azur, France
GEORGIOS KOPANAS*, Inria, Université Côte d'Azur, France
THOMAS LEIMKÜHLER, Max-Planck-Institut für Informatik, Germany
GEORGE DRETTAKIS, Inria, Université Côte d'Azur, France

Fig. 1. Our method achieves real-time rendering of radiance fields with quality that equals the previous method with the best quality [Barron et al. 2022], while only requiring optimization times competitive with the fastest previous methods [Fridovich-Keil and Yu et al. 2022; Müller et al. 2022]. Key to this performance is a novel 3D Gaussian scene representation coupled with a real-time differentiable renderer, which offers significant speedup to both scene optimization and novel view synthesis. Note that for comparable training times to InstantNGP [Müller et al. 2022], we achieve similar quality to theirs; while this is the maximum quality they reach, by training for 51min we achieve state-of-the-art quality, even slightly better than Mip-NeRF360 [Barron et al. 2022].

Radiance Field methods have recently revolutionized novel-view synthesis of scenes captured with multiple photos or videos. However, achieving high visual quality still requires neural networks that are costly to train and render, while recent faster methods inevitably trade off speed for quality. For unbounded and complete scenes (rather than isolated objects) and 1080p resolution rendering, no current method can achieve real-time display rates. We introduce three key elements that allow us to achieve state-of-the-art visual quality while maintaining competitive training times and importantly allow high-quality real-time (≥ 30 fps) novel-view synthesis at 1080p resolution. First, starting from sparse points produced during camera calibration, we represent the scene with 3D Gaussians that preserve desirable properties of continuous volumetric radiance fields for scene optimization while avoiding unnecessary computation in empty space; Second, we perform interleaved optimization/density control of the 3D Gaussians, notably optimizing anisotropic covariance to achieve an accurate representation of the scene; Third, we develop a fast visibility-aware rendering algorithm that supports anisotropic splatting and both accelerates training and allows real-time rendering. We demonstrate state-of-the-art visual quality and real-time rendering on several established datasets.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Point-based models**; **Rasterization**; **Machine learning approaches**.

*Both authors contributed equally to the paper.

Authors' addresses: Bernhard Kerbl, bernhard.kerbl@inria.fr, Inria, Université Côte d'Azur, France; Georgios Kopanas, georgios.kopanas@inria.fr, Inria, Université Côte d'Azur, France; Thomas Leimkühler, thomas.leimkuehler@mpi-inf.mpg.de, Max-Planck-Institut für Informatik, Germany; George Drettakis, george.drettakis@inria.fr, Inria, Université Côte d'Azur, France.

0730-0301/2023/8-ART1 $15.00
https://doi.org/10.1145/3592433

Additional Key Words and Phrases: novel view synthesis, radiance fields, 3D gaussians, real-time rendering

ACM Reference Format:
Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 14 pages. https://doi.org/10.1145/3592433

1 INTRODUCTION

Meshes and points are the most common 3D scene representations because they are explicit and are a good fit for fast GPU/CUDA-based rasterization. In contrast, recent Neural Radiance Field (NeRF) methods build on continuous scene representations, typically optimizing a Multi-Layer Perceptron (MLP) using volumetric ray-marching for novel-view synthesis of captured scenes. Similarly, the most efficient radiance field solutions to date build on continuous representations by interpolating values stored in, e.g., voxel [Fridovich-Keil and Yu et al. 2022] or hash [Müller et al. 2022] grids or points [Xu et al. 2022]. While the continuous nature of these methods helps optimization, the stochastic sampling required for rendering is costly and can result in noise. We introduce a new approach that combines the best of both worlds: our 3D Gaussian representation allows optimization with state-of-the-art (SOTA) visual quality and competitive training times, while our tile-based splatting solution ensures real-time rendering at SOTA quality for 1080p resolution on several previously published datasets [Barron et al. 2022; Hedman et al. 2018; Knapitsch et al. 2017] (see Fig. 1).

Our goal is to allow real-time rendering for scenes captured with multiple photos, and create the representations with optimization times as fast as the most efficient previous methods for typical real scenes. Recent methods achieve fast training [Fridovich-Keil

3D Gaussian splatting overview [1]

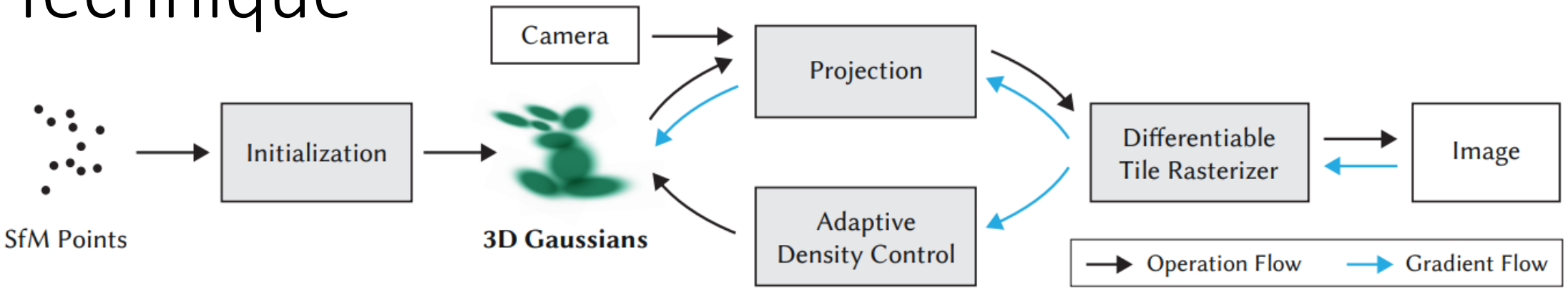
- Challenge: use sparse set of camera views of a scene to construct a 3D scene representation that allows new images to be generated from novel camera angles
- Mildenhall et al. (2020) paper "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis" was breakthrough https://arxiv.org/abs/2003.08934
- Represent space as voxels and train a neural network to output approx. values

3D Gaussian splatting overview [2]

- With NeRF, voxels are uniform in size and uniformly spaced.
  Each voxel has a single RGB color and a level of density/opacity.
- One set of neural network weights are learned for each scene
- Novel views are constructed by sums along rays from the camera view

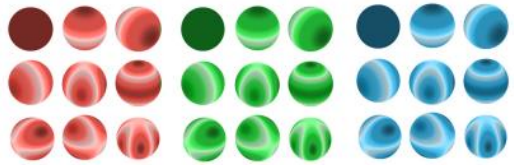
- With 3D Gaussian splatting, Gaussian distributions are used instead of uniform color and transparency voxels
- Further, the 3D Gaussians differ in size, orientation, and spacing
  - Smaller, more densely packed Gaussians are used for fine details
- Rendering novel views, with splatting, is similar (at high level) to NeRF as both sum values along rays from the camera

Technique



- Create a 3D point cloud using Structure from Motion (SfM), and assign a 3D Gaussian to each point
  - SfM can use video frames or multiple still camera angles
- Optimize each 3D Gaussian's properties, plus adaptively increase density by adding Gaussians where appropriate
  - Store 3D position, α/opacity, anisotropic covariance, and (27) RGB spherical harmonics*
- Tile the image, map Gaussians to tiles, splat each 3D Gaussian to 2D, sort them by distance from camera, and sum color data into scene pixels
  - This technique is fast, allowing real-time rendering at 1080p

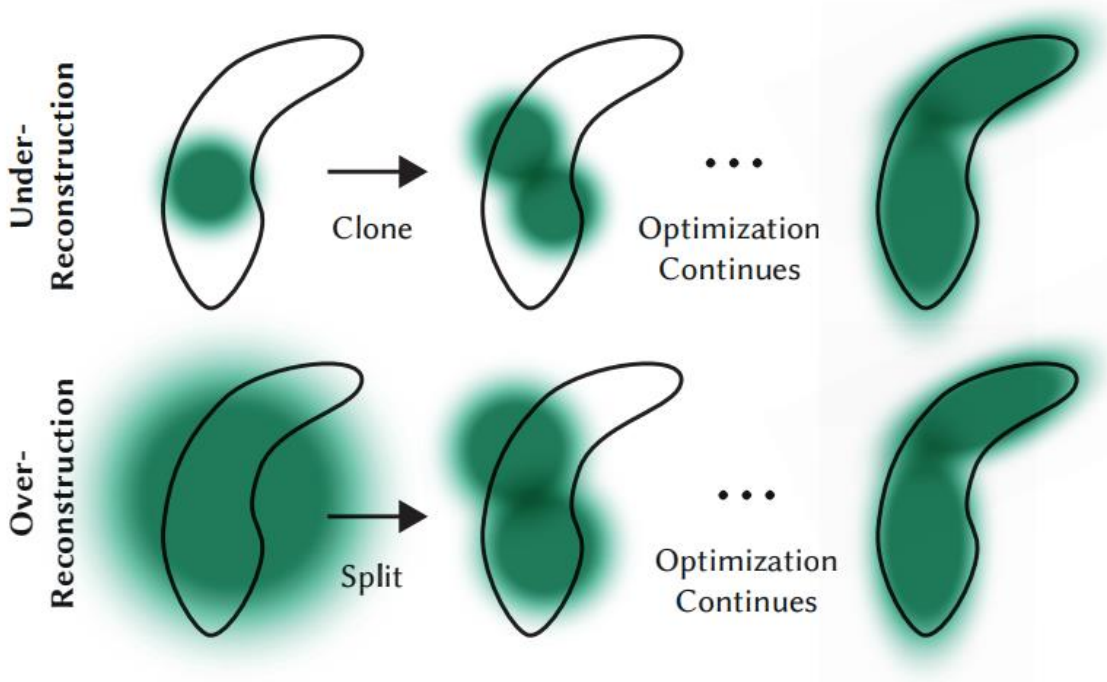
*See https://arxiv.org/abs/2112.05131 for additional history on the use of spherical harmonics



Optimization

- Optimization is done by reconstructing training images and backpropagating from the loss
- Loss function is a combination of L1 reconstruction and D-SSIM loss

- Adaptive density algorithm
  - Warm up
  - Densify every 100 iterations
    - Seek large positional gradients
    - Under-reconstructed -> clone
    - Over-reconstructed -> split
  - Every 3000 iterations, lower α, optimize, and remove Gaussians with $\alpha < \epsilon_\alpha$



Fast differentiable rasterization

- Break up scene into 16x16 tiles
  - Estimate which Gaussians are in the view frustrum and keep 99%+ confident
- Assign (Gaussian, tile) pair a key based on depth and tile ID
- Use fast GPU radix sort to arrange keys by depth
- Launch a CUDA thread block for each tile
  - Threads sum values from front to back, stopping when α hits 1
- During optimization, gradients computed from back to front

Images


713
7K iterations


713
30K iterations


7K iterations


30K iterations

Results

- Very high image quality, sometimes faster training (depending on iterations), and multiple orders of magnitude faster rendering

Dataset	Mip-NeRF360					
Method|Metric	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Train	FPS	Mem
Plenoxels	0.626	23.08	0.463	25m49s	6.79	2.1GB
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB
M-NeRF360	$0.792^{\dagger}$	$27.69^{\dagger}$	$0.237^{\dagger}$	48h	0.06	8.6MB
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB

Ablations

- PSNR scores shown below for ablation runs
  - Disabling Gaussian cloning and eliminating spherical harmonics were least damaging
  - Limiting number of Gaussians backpropagated per pixel was most damaging, followed by not using SfM and disabling Gaussian splitting

	Truck-5K	Garden-5K	Bicycle-5K	Truck-30K	Garden-30K	Bicycle-30K	Average-5K	Average-30K
Limited-BW	14.66	22.07	20.77	13.84	22.88	20.87	19.16	19.19
Random Init	16.75	20.90	19.86	18.02	22.19	21.05	19.17	20.42
No-Split	18.31	23.98	22.21	20.59	26.11	25.02	21.50	23.90
No-SH	22.36	25.22	22.88	24.39	26.59	25.08	23.48	25.35
No-Clone	22.29	25.61	22.15	24.82	27.47	25.46	23.35	25.91
Isotropic	22.40	25.49	22.81	23.89	27.00	24.81	23.56	25.23
Full	22.71	25.82	23.18	24.81	27.70	25.65	23.90	26.05

3D Gaussian splatting conclusion

- SOTA novel view image quality
- Training same or faster than other fast approaches (which had poorer quality)
- Ultra fast rendering, first ever to achieve real-time
- Requires more memory than NeRF-based solutions
  - Future work could reduce algorithm's memory footprint and optimize code
- Limitations
  - As with other approaches, difficulty with regions not well observed
  - Occasionally have "popping" artifacts
  - Both NeRF and 3D Gaussian splatting lack pre-training about world knowledge

- See their video: https://youtu.be/T_kXY43VZnk and website: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Efficient Streaming Language Models with Attention Sinks

https://arxiv.org/abs/2309.17453

EFFICIENT STREAMING LANGUAGE MODELS WITH ATTENTION SINKS

Guangxuan Xiao[1]* **Yuandong Tian**[2] **Beidi Chen**[3] **Song Han**[1] **Mike Lewis**[2]

[1] Massachusetts Institute of Technology
[2] Meta AI
[3] Carnegie Mellon University
https://github.com/mit-han-lab/streaming-llm

ABSTRACT

Deploying Large Language Models (LLMs) in streaming applications such as multi-round dialogue, where long interactions are expected, is urgently needed but poses two major challenges. Firstly, during the decoding stage, caching previous tokens' Key and Value states (KV) consumes extensive memory. Secondly, popular LLMs cannot generalize to longer texts than the training sequence length. Window attention, where only the most recent KVs are cached, is a natural approach — but we show that it fails when the text length surpasses the cache size. We observe an interesting phenomenon, namely *attention sink*, that keeping the KV of initial tokens will largely recover the performance of window attention. In this paper, we first demonstrate that the emergence of *attention sink* is due to the strong attention scores towards initial tokens as a "sink" even if they are not semantically important. Based on the above analysis, we introduce StreamingLLM, an efficient framework that enables LLMs trained with a *finite length* attention window to generalize to *infinite sequence length* without any fine-tuning. We show that StreamingLLM can enable Llama-2, MPT, Falcon, and Pythia to perform stable and efficient language modeling with up to 4 million tokens and more. In addition, we discover that adding a placeholder token as a dedicated attention sink during pre-training can further improve streaming deployment. In streaming settings, StreamingLLM outperforms the sliding window recomputation baseline by up to 22.2× speedup. Code and datasets are provided in the link.

1 INTRODUCTION

Large Language Models (LLMs) (Radford et al., 2018; Brown et al., 2020; Zhang et al., 2022; OpenAI, 2023; Touvron et al., 2023a;b) are becoming ubiquitous, powering many natural language processing applications such as dialog systems (Schulman et al., 2022; Taori et al., 2023; Chiang et al., 2023), document summarization (Goyal & Durrett, 2020; Zhang et al., 2023a), code completion (Chen et al., 2021; Rozière et al., 2023) and question answering (Kamalloo et al., 2023). To unleash the full potential of pretrained LLMs, they should be able to efficiently and accurately perform long sequence generation. For example, an ideal ChatBot assistant can stably work over the content of recent day-long conversations. However, it is very challenging for LLM to generalize to longer sequence lengths than they have been pretrained on, e.g., 4K for Llama-2 Touvron et al. (2023b).

The reason is that LLMs are constrained by the attention window during pre-training. Despite substantial efforts to expand this window size (Chen et al., 2023; kaiokendev, 2023; Peng et al., 2023) and improve training (Dao et al., 2022; Dao, 2023) and inference (Pope et al., 2022; Xiao et al., 2023; Anagnostidis et al., 2023; Zhang et al., 2023b) efficiency for lengthy inputs, the acceptable sequence length remains intrinsically *finite*, which doesn't allow persistent deployments.

In this paper, we first introduce the concept of LLM streaming applications and ask the question:

Can we deploy an LLM for infinite-length inputs without sacrificing efficiency and performance?

*Part of the work done during an internship at Meta AI.

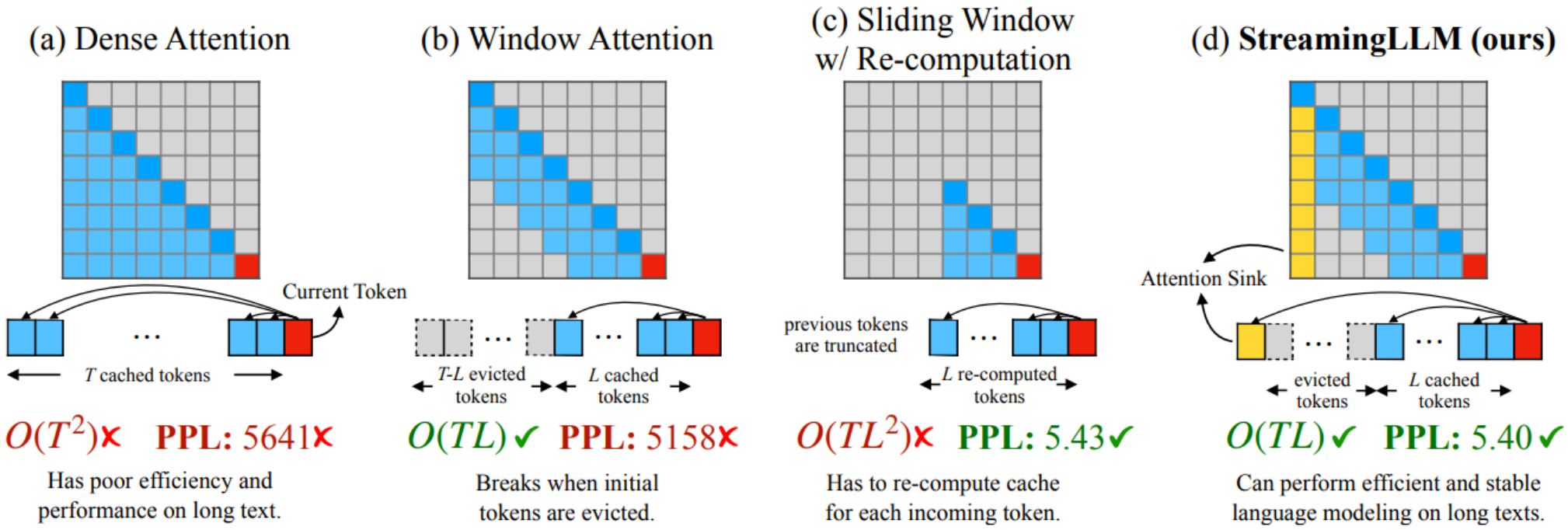
Attention sinks overview

- Efficient Streaming Language Models with Attention Sinks
- Guangxuan Xiao et al. from MIT

- Challenge: continue generate text, such as in a multi-round chatbot, even after the history has exceeded your LLM's max context length
- The obvious solution is to use a sliding window of most recent tokens, but this leads to severe performance degradation
- Keeping an attention sink of first token(s) solves performance problem

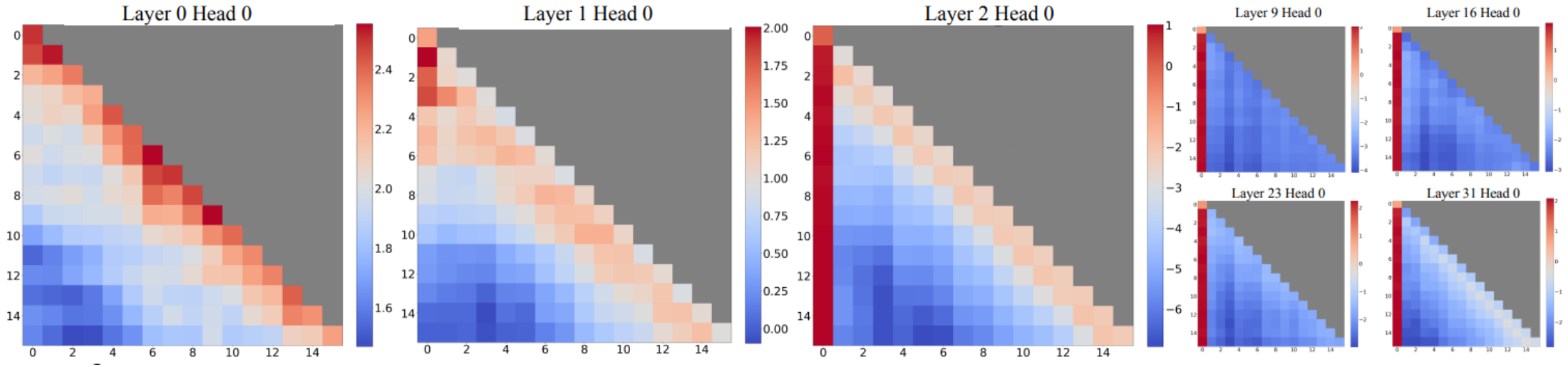
Long context approaches

- Regular sliding window with cached KV has bad quality (perplexity)
- Re-computation with sliding window doesn't cache, so hurts speed
- Attention sink approach uses caching and maintains text quality



Why sliding window performance is impacted

- The attention maps in autoregressive LLMs put a lot of attention on the first token(s), which they name attention sinks
  - This was seen in the transformer circuits work by Anthropic https://transformer-circuits.pub/2021/framework/index.html
  - Test showed it is the position, not the content of first tokens, that matters



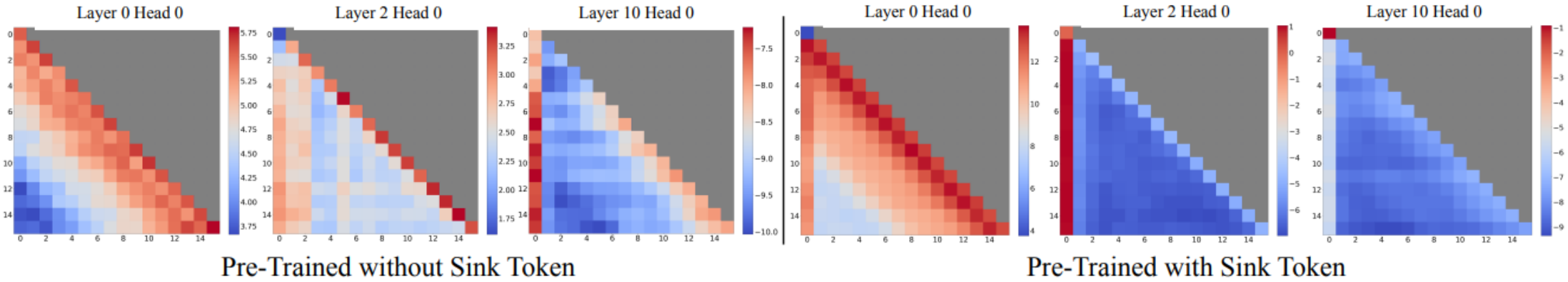
Attention sinks

- Experiments showed most models require first four tokens to recover normal text quality (perplexity)
- Cached KV values use positions values 1 to L, regardless of if current tokens are well beyond the Lth token
- Using RoPE or ALiBi, the base token embeddings can be cached, then the positional information applied on top of the base embeddings
- Pre-training a model with a learned attention sink reduces requirement to a single attention sink

Cache Config	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
Zero Sink	29214	19.90	18.27	18.01
Learnable Sink	1235	**18.01**	18.01	18.02

Pre-trained model attention patterns

- Attention on first token is more consistent in models pre-trained with an attention sink token



- These pre-trained models had similar performance on benchmarks when compared to models without attention sinks

Attention sinks conclusion

- Using attention sinks during decoding solves text generation quality after the context exceeds the max sequence length
  - This approach works with KV caching when using additive, RoPE, or ALiBi position embeddings
- There’s almost no downside, since sparing a few tokens is negligible
- Pre-training models to expect attention sinks reduces cost to just a single attention sink token
- Context length is still a problem for LLMs. This work adds to the growing work to understand transformers.

References

- RoFormer: Enhanced Transformer with Rotary Position Embedding
  Jianlin Su et al. (2021)
  https://arxiv.org/abs/2104.09864
- Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation
  Ofir Press et al. (2021)
  https://arxiv.org/abs/2108.12409