PICO A Modular Framework for Hypothesis-Driven Small Language Model Research

Richard Diehl Martinez* **David Demitri Africa†** **Yuval Weiss†**
Suchir Salhan **Ryan Daniels** **Paula Buttery**
University of Cambridge

Abstract

Building language models (LMs), especially small and medium ones, remains more art than science. While large LMs often improve by sheer scale, it is still unclear why many design choices work. For small LMs, this uncertainty is more limiting: tight parameter budgets make each decision critical, yet researchers still lack systematic, scientific ways to test and refine new ideas. We introduce **Pico**, a lightweight, modular framework that enables systematic, hypothesis-driven research for small and medium-scale language model development. **Pico** consists of two libraries that together provide a practical sandbox where researchers can make targeted changes to a model's architecture or training procedures and directly observe their effects on the model's behavior. To support reproducible experimentation, we also release a suite of baseline models, **`pico-decoder`**, trained under standardized conditions and open-sourced for the community. Case studies highlight how **Pico** can support iterative small LM design and analysis.

pico-lm (Apache 2.0) pico-lm (Apache 2.0)
picolm.io Demo Video

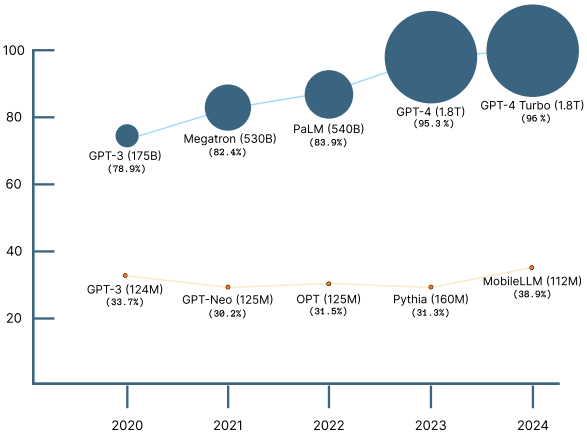

Figure 1: Best performance of fixed-size SLMs and LLMs on MMLU per year, size of the circles represents model size. Note that the size of GPT-4 models is speculative[1] and has not been confirmed by OpenAI.

1 Introduction

Recent advances in large language models (LLMs) have enabled strong performance across diverse tasks (Hendrycks et al., 2021; Cobbe et al., 2021; Srivastava et al., 2023), but progress on Small Language Models (SLMs) has been slower (see Fig. 1). SLMs, loosely defined as models with fewer than 10 billion parameters, are large enough for emergent behaviors yet small enough to train on modest budgets (Hu et al., 2024; Van Nguyen et al., 2024; Wang et al., 2024; Subramanian et al., 2025). Despite growing interest, designing efficient, high-performing SLMs still relies on opaque trial-and-error, with limited understanding of how design choices shape learning dynamics.

In this paper, we present **Pico**, a modular framework designed to help researchers develop SLMs in a more scientifically rigorous manner. **Pico** consists of two libraries: `pico-train`, which provides a lightweight, transparent training loop for language models; and `pico-analyze`, a complementary toolkit for analyzing their learning dynamics. Conceptually, `pico-train` provides the infrastructure for training and systematically checkpointing model states and activations, while `pico-analyze` offers the tools to compute learning dynamics metrics and comparisons on those checkpoints.

By bridging training and analysis in a single open-source ecosystem, **Pico** lowers the barrier for conducting reproducible, hypothesis-driven research on small language model development. To support controlled experimentation, we also release a set of baseline models trained under standardized conditions, the **`pico-decoder`** suite. The suite is a starting point for researchers to build on and com-

*Corresponding author: **richard@picolm.io**
†Equal contribution

[1] https://the-decoder.com/gpt-4-has-a-trillion-parameters

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 295–306
November 4-9, 2025

Tool	Custom Training	Checkpoint Support	Feature Extraction	Analysis Tools	Low-Budget Friendly
Pico	✓ Modular PyTorch	✓ Optimizer, weights & data	✓ Activations & gradients	✓ `pico-analyze` metrics	✓ Academic-GPU scale
TransformerLens	✗	✗	✓	✓	✓
ACDC	✗	✗	✓	✓	✓
SAELens	✗	✗	▲	✓	✓
SmolLM2	✓	▲	✗	✗	✓
Pythia Suite	▲	▲	✗	▲	✓
OLMo	✓	▲	✗	✗	▲

Table 1: Comparison of **Pico** and related frameworks for interpretability and learning dynamics. **Legend:** ✓= Fully supported; ▲= Partial; ✗= Not supported.

pare against. Case studies in 3 demonstrate how **Pico** enables researchers to build language models in a hypothesis-driven way.

2 Pico

Unlike existing pretraining or interpretability stacks, **Pico** integrates modular training with built-in support for learning dynamics. Table 1 highlights this key advantage.

On the training side, `pico-train` automatically logs detailed activations, gradients, and weights at checkpoint intervals and enables researchers to efficiently train models for controlled experiments.

On the analysis side, `pico-analyze` operates directly on these in-situ logs, applying flexible metrics and component abstractions to track learning dynamics as they unfold.

In this section we provide a concise overview of the two **Pico** libraries: `pico-train` and `pico-analyze`.

2.1 `pico-train`: A Minimalist Approach to Model Training

`pico-train` is a lightweight, transparent framework for training small- to medium-scale language models. Unlike many existing training libraries that prioritize efficiency at the cost of clarity, `pico-train` is designed to be simple, modular, and easy to modify, making it a flexible foundation for experimentation in language model research.

Out of the box, `pico-train` implements `pico-decoder`, a LLaMA-style transformer (Touvron et al., 2023) that incorporates key features of modern autoregressive language models, including Grouped Query Attention (GQA) (Ainslie et al., 2023), Rotary Position Embeddings (RoPE) (Su et al., 2024), FlashAttention (Dao et al., 2022), SwiGLU activations (Shazeer, 2020), and RMSNorm (Zhang and Sennrich, 2019). All components, except FlashAttention, are re-implemented from scratch in plain PyTorch (Paszke et al., 2019), with an emphasis on readability and documentation.

To ensure efficient multi-GPU and distributed training, `pico-train` is built on Lightning Fabric (Lightning AI, 2025) – a framework that, like **Pico**, prioritizes simplicity and flexibility. Lightning Fabric enables users to scale up training across multiple GPUs or nodes without introducing excessive abstractions and ensures that the core training logic remains easy to understand and modify.

A distinguishing feature of `pico-train` is its systematic checkpointing and version control system. It automatically saves:

- **Model states in both PyTorch- and Hugging Face-compatible formats** (Wolf et al., 2019). This dual-format checkpointing enables straightforward loading with vanilla PyTorch or integration into the Hugging Face ecosystem, facilitating downstream tasks such as fine-tuning, inference, or model sharing. Researchers can thus easily plug `pico-train` outputs into existing pipelines.

- **Intermediate activations and gradients.** At user-defined intervals, the library gathers layerwise activations and gradients from the forward and backward passes on the current training batch. Optionally, it can also capture these metrics from a fixed evaluation batch for consistent comparisons over training. Collecting these tensors at each checkpoint provides a granular record of how representations and

gradient flows evolve over time.

- **Training data batch.** We save out the batch of training data that was used to extract the set of activations and gradients at a given point in training.

- **Evaluation results.** Users can define and record evaluation metrics (e.g., validation perplexity, accuracy) alongside model checkpoints.

All checkpoints are automatically uploaded and version-controlled on Hugging Face, ensuring that researchers can revisit any point in training to analyze how the model evolved over time. These structured checkpoints integrate seamlessly with `pico-analyze`, enabling learning dynamics research with minimal setup.

To simplify experimentation, we release a pretokenized, pre-chunked, and pre-shuffled version of Dolma (Soldaini et al., 2024), a large, open-source English dataset, on Hugging Face: **`pretokenized-dolma`**. This dataset removes preprocessing overhead, ensures consistency across runs, and supports streaming to reduce storage needs. Using it is optional; users can substitute their own data if they prefer. Details on our preprocessing steps are in App. D.

By focusing on minimalism, modularity, and transparency, `pico-train` makes it easy to modify all aspects of the training pipeline.

2.2 `pico-analyze`: A General-Purpose Framework for Studying Learning Dynamics

`pico-analyze` is a companion tool to `pico-train` designed to make analyzing learning dynamics seamless and reproducible. It directly integrates with the checkpoints saved by `pico-train` that include activations, and gradients and enables researchers to compute the learning dynamics of trained models.

At its core, `pico-analyze` follows a simple abstraction: it applies metrics to components. Metrics provide quantitative insights into various aspects of model behavior, while components define the specific model elements being analyzed. This design allows for flexible and fine-grained analysis of training dynamics.

Metrics. Out of the box, `pico-analyze` supports a range of built-in metrics, including:

- **Sparsity Measures**: *Gini coefficient* (Hurley and Rickard, 2009) and *Hoyer metric* (Hoyer, 2004) gauge how concentrated the values of a matrix are near zero.

- **Rank-Based Metrics**: *Proportional Effective Rank* (Diehl Martinez et al., 2024) captures a matrix's "effective dimensionality," while *Condition Number* evaluates its numerical stability.

- **Representation Similarity**: *CKA* (Kornblith et al., 2019) and *PWCCA* (Morcos et al., 2018) compare activation patterns across layers or checkpoints, revealing how internal representations evolve.

- **Norms**: *Frobenius*, *Nuclear*, and *Infinity* norms measure the scale of a tensor, spotlighting issues such as vanishing or exploding parameters.

Components. Metrics can be computed on different types of components:

- **Simple components**: Individual weight matrices, gradients, or activations from a single layer.

- **Compound components**: Higher-level structures that combine multiple model elements. One example is the OV circuit, which tracks how information flows in transformer models by combining the value and output projection matrices in self-attention layers (Elhage et al., 2021).

This two-step abstraction is designed for extensibility; new metrics and component types can be easily defined, allowing researchers to tailor analyses to specific hypotheses about language model learning. We view `pico-analyze` not as a static toolset, but as a foundation for community-driven interpretability research.

3 Case Studies

We illustrate how **Pico** enables systematic, hypothesis-driven experimentation through two case studies.

3.1 MAML

Model-Agnostic Meta-Learning (Finn et al., 2017, MAML) trains models to adapt quickly by alternating between short bursts of task-specific learning

and a global update that improves generalization. This setup encourages models to find initialization points that adapt well to new tasks. While MAML is typically used for fine-tuning, we follow prior work (Bansal et al., 2020; Li and Zhang, 2021) in applying it during pretraining.

Implementation We implemented MAML in `pico-train` by adding a lightweight inner loop that updates a classification head on masked token tasks, followed by a meta-update to the full model (Africa et al., 2025a,b). `pico-train` automatically handles distributed GPU synchronization, requiring no changes to **Pico**'s core training logic.

Analysis We evaluate MAML on Paloma perplexity and observe consistent 4–15% gains over standard pretraining. To better understand this improvement, we analyze the proportional effective rank (PER) (Diehl Martinez et al., 2024) of both weights and gradients over time. PER captures the dimensionality of a tensor's signal. In meta-learning, one concern is that inner-loop updates may overly constrain model updates to a low-dimensional subspace, potentially limiting generalization. As shown in Fig. 2, we observe synchronized troughs in PER and spikes in both loss and perplexity. This suggests that inner-loop updates temporarily compress the model's capacity into a low-rank subspace before the outer-loop update restores variance and expressivity.

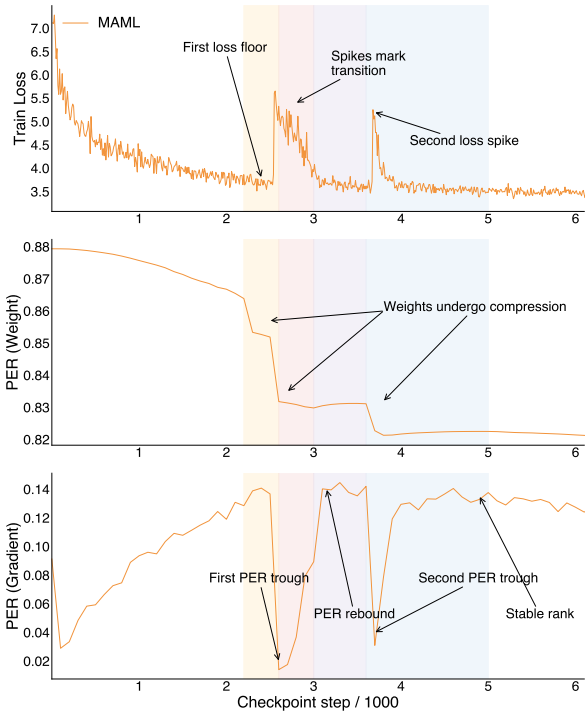

Figure 2: Training dynamics under MAML. **Top to bottom:** Training loss and proportional effective rank (PER) of weights and gradients. Sharp drops in PER align with spikes in loss. Shaded regions correspond to different observed phases in training.

New Hypothesis and Next Steps These results support the hypothesis that MAML's learning dynamics involve cycles of compression and recovery. This raises concrete follow-up questions: could adjusting the inner-loop learning rate reduce excessive compression? Would alternative meta-learning schedules or task mixes stabilize the representational space more effectively? Using Pico's modular training and built-in logging, these variants can be tested with minimal friction. By comparing learning dynamics and outcomes across runs, researchers can refine their design choices in a reproducible, hypothesis-driven loop.

3.2 ReLoRA

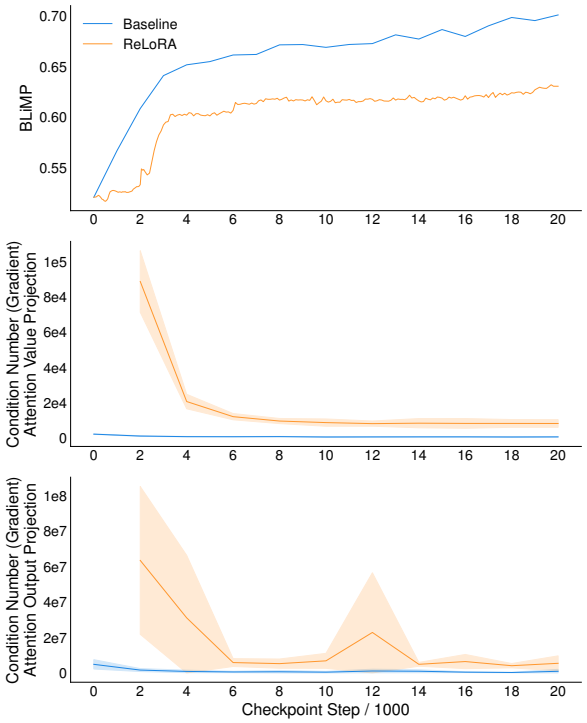

Figure 3: Training dynamics under ReLoRA. **Top to bottom:** BLiMP accuracy over time, averaged condition numbers of the gradient updates for the attention value and output projection matrices.

ReLoRA (Lialin et al., 2023) adapts LoRA (Hu et al., 2022), a fine-tuning technique that freezes pretrained weights and injects trainable low-rank matrices, into the pretraining loop. In theory, this could provide a sample-efficient way to train

Family	Size	#Tokens	Paloma ↓	HellaSwag ↑	ARC-Easy ↑	TruthfulQA ↑
Pico	11M	250B	136.17	25.62	32.79	51.75
	65M	250B	42.24	27.25	38.22	46.13
	181M	250B	30.08	30.69	44.65	41.85
	570M	250B	22.96	37.33	48.99	36.33
Pythia	14M	300B	86.64	26.15	31.31	50.14
	70M	300B	43.76	27.56	36.23	47.02
	160M	300B	29.96	30.26	43.73	44.51
	410M	300B	20.55	40.55	52.10	41.23
OPT	125M	300B	27.22	31.33	43.48	42.89
	350M	300B	20.91	36.66	44.06	41.01

Table 2: Performance of small-scale language models on four benchmarks. Lower is better for Paloma perplexity (↓); higher is better for HellaSwag, ARC-Easy, and TruthfulQA accuracies (↑).

large models by constraining updates to a low-rank subspace.

Implementation We incorporate ReLoRA into `pico-train` by adding a lightweight wrapper around attention and MLP weight matrices, and by modifying the learning rate schedule to handle periodic resets (Weiss et al., 2025). We evaluated the model on BLiMP (Warstadt et al., 2020), which we configured via a single config entry.

Analysis We find that ReLoRA surprisingly underperforms standard pretraining on BLiMP (see Fig. 3, top) (Warstadt et al., 2020). To investigate this, we analyze the condition numbers of the gradient updates across layers and training checkpoints. This metric reflects how sensitive gradient-based updates are to numerical instability, a relevant concern for methods like ReLoRA that repeatedly project updates into a low-rank subspace. As shown in Fig. 3 (bottom), ReLoRA gradients are substantially more ill-conditioned and exhibit high inter-layer variance.

New Hypothesis and Next Steps This pattern suggests that repeated low-rank resets may amplify gradient instability, undermining ReLoRA's intended efficiency gains. Several next steps follow naturally: for example, adding layerwise condition number regularization, adjusting the rank dynamically, or modifying the reset schedule to reduce instability. Because `pico-train` and `pico-analyze` modularize core components and log detailed in-situ signals, researchers can test these changes quickly, track their effects on gradient stability, and iterate systematically. This demonstrates **Pico**'s value as a scientific sandbox for implementing, analyzing, and refining design choices in the small LM regime.

4 Pico Model Suite

We train a suite of **`pico-decoder`** models at various scales on **`pretokenized-dolma`** using `pico-train`, all of which are open-sourced on our Hugging Face organization. These models range from 11M to 570M parameters, with plans to extend to billion-parameter models, and serve both as evaluations of our training pipeline and as testbeds for research on scaling laws and interpretability.

Each model is trained for 125,000 steps (covering 250B tokens). We evaluate the final model checkpoints on the Paloma benchmark (Magnusson et al., 2024), HellaSwag (Zellers et al., 2019), Arc-Easy (Clark et al., 2018) and Truthful QA (Lin et al., 2022), comparing performance against established decoder models. As shown in Table 2, our models achieve comparable results to Pythia and OPT models, despite running on an academic-level compute budget (4 nodes of 4 A100s each).

We provide a comparison of these models and their compute/storage overhead in Table 4. Reported "GPU hours" are not directly comparable across frameworks due to differences in hardware, dataloaders, and logging pipelines, but our training times are broadly consistent with existing suites. For reference, whereas `pico-large` required 7,465 A100-hours, prior reports list Pythia-1B at 4,830 A100-hours, MPT-1.3B at 7,920 A100-hours, and TinyLlama-1.1B at 3,456 A100-hours under optimized stacks (Zhang et al., 2024). Importantly, all

of our models are trained in a streaming pipeline, with datasets and checkpoints streamed from and uploaded to Hugging Face. Streaming adds data latency [2] but greatly simplifies reproducibility and removes local storage requirements. Users who prioritize throughput can instead train models with a locally cached dataset.

5 Related Literature

We survey where **Pico** sits within a growing ecosystem of frameworks that support the training and analysis of language models, ranging from optimized production libraries to interpretability toolkits.

5.1 Training Frameworks

Open-source initiatives. Initiatives by EleutherAI, including GPT-Neo, GPT-J, and the interpretability-focused Pythia suite (Biderman et al., 2023) as well as projects like the Allen Institute's OLMo (Groeneveld et al., 2024), Meta's Llama (Touvron et al., 2023), and BigScience's BLOOM (Le Scao et al., 2023), have democratized access to pretrained weights and checkpoints. While these frameworks support post-hoc investigations into phenomena such as linguistic emergence and scaling effects (Belrose et al., 2023; Gurnee et al., 2023; Michaelov and Bergen, 2023; Diehl Martinez et al., 2024), they do not capture detailed, in-training signals by default and only provide static checkpoints. Smaller frameworks like SmolLM (Allal et al., 2025), TinyLlama (Zhang et al., 2024), NanoGPT (Karpathy, 2023), and TinyStories (Eldan and Li, 2023) offer minimalistic, modular training loops that facilitate quick experimentation. However, they usually leave the implementation of fine-grained monitoring (e.g., activations or gradient flows) to the user.

Large-scale and efficient frameworks. Platforms such as NVIDIA's Megatron-LM (Narayanan et al., 2021) and Microsoft's DeepSpeed (Rasley et al., 2020) excel at distributed training for models with billions of parameters, though they lack native mechanisms for inspecting intermediate states.

5.2 Analysis Frameworks

Post-hoc model probing. Given that detailed training signals are often unavailable by default, many researchers have adopted post-hoc probing methods. Such approaches rely on external hooking libraries to intercept hidden states and attention patterns (Voita et al., 2019; Clark et al., 2019; Michel et al., 2019), looking at information flows within models to discover security or privacy vulnerabilities (Roger, 2023; Yao et al., 2024). While powerful, these methods demand significant modifications and usually depend upon pre-existing checkpoints.

Mechanistic interpretability. Recently, mechanistic interpretability (mechinterp) has gained traction as a framework for reverse-engineering neural networks at the algorithmic level (Olah et al., 2020; Elhage et al., 2021). Mechinterp focuses on localizing and characterizing the internal "circuitry" of attention heads, MLP layers, or individual neurons. A variety of mechinterp libraries, e.g., TransformerLens (Nanda and Bloom, 2022), SAELens (Bloom et al., 2024), and ACDC (Conmy et al., 2023), offer powerful tooling to dissect trained transformer models at inference time. However, these efforts typically assume that checkpoints are already available and do not natively capture the evolution of internal mechanisms throughout training.

6 Conclusion

We introduce **Pico**, a modular framework designed to help researchers study and improve small and medium-sized language models through a more systematic, scientifically grounded process. `pico-train` provides an extensible environment for training models, with built-in checkpointing that captures detailed signals needed to analyze learning dynamics. `pico-analyze` builds directly on these checkpoints, enabling researchers to test specific hypotheses about how changes to model architectures or training procedures affect convergence, sparsity, rank, and representation learning.

To further support controlled experiments and comparative studies, we open-source a suite of **`pico-decoder`** baseline models ranging from 11M to 570M parameters. These baselines give researchers a consistent starting point for evaluating new ideas or scaling laws under reproducible conditions.

By combining transparent training, detailed in-situ logging, and flexible analysis, **Pico** provides a practical sandbox for hypothesis-driven research.

[2] On our network, streaming results in approximately 80–100% slower batch loading compared to a local cache.

Limitations

The **Pico** framework is designed for interpretability and experimentation rather than optimized large-scale production training, meaning it may not efficiently scale to industrial-scale models with hundreds of billions of parameters. Additionally, the inherent overhead of systematically checkpointing intermediate activations and gradients at frequent intervals can significantly increase storage and computational costs during training.

Ethics Statement

Pico aims to facilitate transparent and reproducible research into language model interpretability and learning dynamics. To streamline experimentation, we release the `pretokenized-dolma` dataset, a preprocessed dataset in English, enabling quick and efficient model training. Additionally, the initial `pico-decoder` model suite is also trained exclusively on English-language data. We acknowledge that this emphasis on English datasets and models can inadvertently reinforce English as the dominant language in NLP and interpretability research, potentially marginalizing research on other languages. We strongly encourage and support the development and release of similarly structured, high-quality datasets and models in languages other than English. Finally, any checkpoint or artifact uploaded by **Pico** to platforms such as Hugging Face must be used responsibly, with users remaining mindful of data privacy concerns, potential biases in training data, and risks associated with misuse or harmful applications of model checkpoints.

Acknowledgments

This work was supported by a grant from the Accelerate Programme for Scientific Discovery, made possible by a donation from Schmidt Futures. Richard Diehl Martinez is supported by the Gates Cambridge Trust (grant OPP1144 from the Bill & Melinda Gates Foundation). Suchir Salhan is supported by Cambridge University Press & Assessment. David Demitri Africa is supported by the Cambridge Trust and the Jardine Foundation. A big thank you to Leshem Choshen for their helpful comments and suggestions.

References

David Demitri Africa, Suchir Salhan, Yuval Weiss, Paula Buttery, and Richard Diehl Martinez. 2025a. Meta-pretraining for zero-shot cross-lingual named entity recognition in low-resource philippine languages. *arXiv preprint arXiv:2509.02160*.

David Demitri Africa, Yuval Weiss, Paula Buttery, and Richard Diehl Martinez. 2025b. Learning dynamics of meta-learning in small model pretraining. *arXiv preprint arXiv:2508.02189*.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. 2025. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534, Online. Association for Computational Linguistics.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *Preprint*, arXiv:2303.08112.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. 2024. Saelens. `https://github.com/jbloomAus/SAELens`.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Richard Diehl Martinez, Pietro Lesci, and Paula Buttery. 2024. Tending towards stability: Convergence challenges in small language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3275–3286, Miami, Florida, USA. Association for Computational Linguistics.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *Preprint*, arXiv:2305.07759.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Patrik O Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Niall Hurley and Scott Rickard. 2009. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741.

Andrej Karpathy. 2023. nanogpt.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Yue Li and Jiong Zhang. 2021. Semi-supervised meta-learning for cross-domain few-shot intent classification. In *Proceedings of the 1st Workshop on Meta Learning and Its Applications to Natural Language Processing*, pages 67–75, Online. Association for Computational Linguistics.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2023. ReLoRA: High-Rank Training Through Low-Rank Updates. In *The Twelfth International Conference on Learning Representations*.

Lightning AI. 2025. Lightning fabric. Version 2.5.1.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Walsh, Yanai Elazar, Kyle Lo, et al. 2024. Paloma: A benchmark for evaluating language model fit. *Advances in Neural Information Processing Systems*, 37:64338–64376.

James Michaelov and Ben Bergen. 2023. Emergent inabilities? inverse scaling over the course of pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14607–14615, Singapore. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ari Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. *Advances in neural information processing systems*, 31.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–15.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506.

Fabien Roger. 2023. Large language models sometimes generate purely negatively-reinforced text. *Preprint*, arXiv:2306.07567.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Featured Certification.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Shreyas Subramanian, Vikram Elango, and Mecit Gungor. 2025. Small language models (slms) can still pack a punch: A survey. *arXiv preprint arXiv:2501.05465*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Chien Van Nguyen, Xuan Shen, Ryan Aponte, Yu Xia, Samyadeep Basu, Zhengmian Hu, Jian Chen, Mihir Parmar, Sasidhar Kunapuli, Joe Barrow, et al. 2024. A survey of small language models. *arXiv preprint arXiv:2410.20011*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, et al. 2024. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *arXiv preprint arXiv:2411.03350*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Yuval Weiss, David Demitri Africa, Paula Buttery, and Richard Diehl Martinez. 2025. Investigating relora: Effects on the learning dynamics of small language models. *Preprint*, arXiv:2509.12960.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

A Default `pico-train` configurations

Category	Parameter	Default Value
Model	Model Type	`pico_decoder`
	Hidden Dimension (d_{model})	768
	Number of Layers (n_{layers})	12
	Vocabulary Size	50,304
	Sequence Length	2,048
	Attention Heads	12
	Key/Value Heads	4
	Activation Hidden Dim	3,072
	Normalization Epsilon	1×10^{-6}
	Positional Embedding Theta	10,000.0
Training	Optimizer	AdamW
	Learning Rate	3×10^{-4}
	LR Scheduler	Linear w/ Warmup
	Warmup Steps	2,500
	Gradient Accumulation Steps	128
	Max Training Steps	200,000
	Precision	BF16 Mixed
Data	Dataset Name	`pico-lm/pretokenized-dolma`
	Batch Size	1,024
	Tokenizer	`allenai/OLMo-7B-0724-hf`
Checkpointing	Auto Resume	True
	Save Every N Steps	1,000
	Learning Dynamics Layers	`"attention.v_proj"`, `"attention.o_proj"`, `"swiglu.w_2"`
	Learning Dynamics Eval Data	`pico-lm/pretokenized-paloma-tinsy`
Evaluation	Metrics	`["paloma"]`
	Paloma Dataset Name	`pico-lm/pretokenized-paloma-tinsy`
	Eval Batch Size	16
Monitoring	Logging Level	INFO
	Log Every N Steps	100

Table 3: Default configuration settings used in `pico-train`, organized by configuration category.

B `pico-decoder` models comparison

Attribute	`tiny`	`small`	`medium`	`large`
Parameter Count	11M	65M	181M	570M
Hidden Dimension (d_{model})	96	384	768	1536
Feed-forward Dim	384	1536	3072	6144
Training Time (125K Steps)	4926h	5645h	6112h	7465h
Checkpoint Time	7m42s	3m25s	1m58s	1m29s
Checkpoint Storage (Model State)	151MB	863MB	2.4GB	7.5GB
Checkpoint Storage (Learning Dynamics)	37MB	176MB	550MB	2GB

Table 4: Comparison of `pico-decoder` model variants trained with default `pico-train` configurations. Except for hidden and feed-forward dimension, all models share the training settings detailed in Table 3. Models are trained for 125,000 total training steps on 16 NVIDIA A100-SXM4-80GB GPUs. Time reported in GPU hours (h), minutes (m) and seconds (s); checkpoint time and storage reported per checkpoint step.

C Available metrics in `pico-analyze`

Metric	Description	Data Type	Category
CKA (Kornblith et al., 2019)	• Measures similarity between activations at different checkpoints • Uses kernel methods to track representation evolution	Activations	**Similarity**
PWCCA (Morcos et al., 2018)	• Measures activation similarity across training • Emphasizes important components via projections	Activations	**Similarity**
Condition Number	• Computes ratio of largest to smallest singular value • Indicates sensitivity to small input changes	Weights Activations Gradients	**Rank**
PER (Diehl Martinez et al., 2024)	• Measures entropy of normalized singular values • Indicates effective parameter usage	Weights Gradients	**Rank**
Gini Coefficient (Hurley and Rickard, 2009)	• Measures sparsity via weight distribution inequality	Weights Activations Gradients	**Sparsity**
Hoyer's Sparsity (Hoyer, 2004)	• Measures sparsity by computing ratio of L1/L2 norms	Weights Activations Gradients	**Sparsity**
Norm	• Frobenius, Nuclear, Infinity matrix norms	Weights Activations Gradients	**Norm**

Table 5: Overview of built-in metrics in `pico-analyze`. **Data Types** indicates on what types of checkpoint data the metrics can be applied. The **Category** column classifies metrics based on their primary purpose.

D Preprocessing of the `pretokenized-dolma` dataset

To prepare the `pretokenized-dolma` dataset used in our experiments, we begin by downloading the Dolma corpus and selecting a random 30% subset. The text is then tokenized using the `allenai/OLMo-7B-0724-hf` tokenizer and split into fixed-length sequences of 2049 tokens (2048 + 1 for next-token prediction). We ensure consistency across shards by chunking token streams without overlap, dropping any remainder shorter than the full sequence length.

After tokenization and chunking, we shuffle the dataset and sample a fixed number of sequences per shard, generating 100 shards in total. The resulting dataset is saved as Parquet files and uploaded to our Hugging Face organization under **`pico-lm/pretokenized-dolma`**.

To facilitate scalable loading and training, we further fine-shard the dataset into 10,000 pieces using a secondary script. These final shards are compact (78MB each), randomly shuffled, pretokenized, and ready for streaming via the Hugging Face datasets API. This preprocessing ensures that all models see data in a consistent order, which is critical for learning dynamics analysis. We release all of the scripts we use for preprocessing data in our GitHub repository.