David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. Quartz: An open-domain dataset of qualitative relationship questions. *arXiv preprint arXiv:1909.03553*, 2019.

Glen Taggart. ProLU: A nonlinearity for sparse autoencoders. *AI Alignment Forum*, 2024. URL https://www.alignmentforum.org/posts/HEpufTdakGTTKgoYF/prolu-a-pareto-improvement-for-sparse-autoencoders.

Alon Talmor, Ori Yoran, Ronan Le Bras, Chandra Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. Commonsenseqa 2.0: Exposing the limits of ai through gamification. *arXiv preprint arXiv:2201.05320*, 2022.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Johannes Welbl, Nelson F. Liu, Matt Gardner, Gabor Angeli, Rik Koncel-Kedziorski, Emily Bender, Kyle Richardson, Peter Clark, and Nate Kushman. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017. URL https://arxiv.org/abs/1707.06209.

Benjamin Wright and Lee Sharkey. Addressing feature suppression in SAEs. *AI Alignment Forum*, 2024. URL https://www.alignmentforum.org/posts/3JuSjTZyMzaSeTxKk/addressing-feature-suppression-in-saes.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Zeyu Yun, Yubei Chen, Bruno A Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. *arXiv preprint arXiv:2103.15949*, 2021.

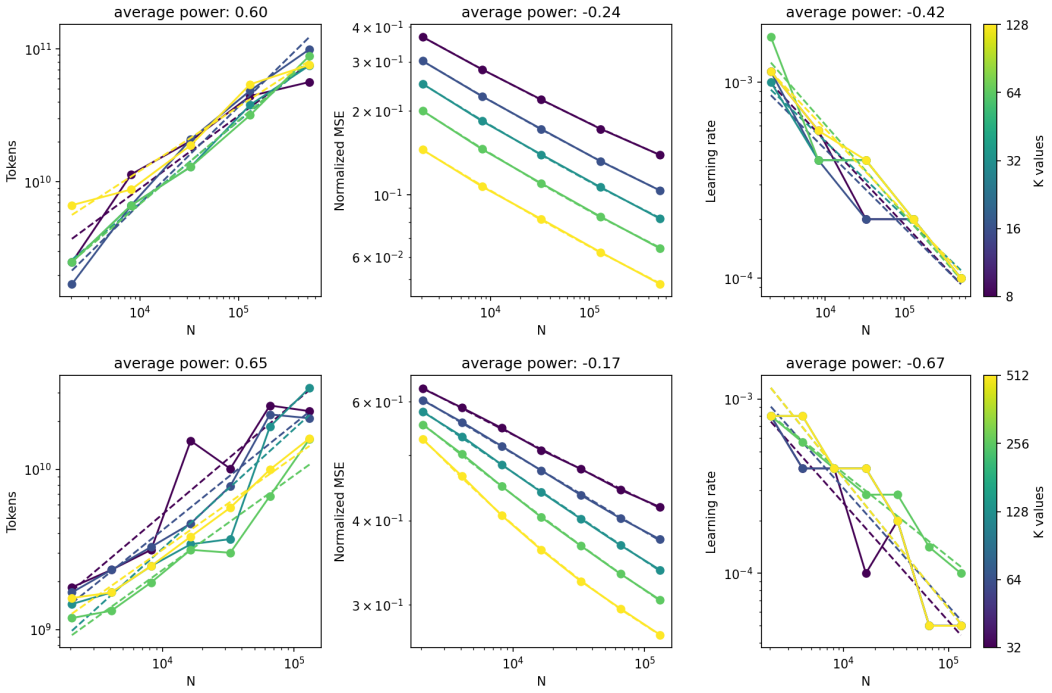

Figure 11: Token budget, MSE, and learning rate power laws, averaged across values of k. First row is GPT-2, second row is GPT-4. Note that individual fits are noisy (especially for GPT-4) since learning rate sweeps are coarse, and token budget depends on learning rate.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. URL https://arxiv.org/abs/1909.03065.

A Optimization

A.1 Initialization

We initialize our autoencoders as follows:

- We initialize the bias b_{pre} to be the geometric median of a sample set of data points, following Bricken et al. [2023].
- We initialize the encoder directions parallel to the respective decoder directions, so that the corresponding latent read/write directions are the same[18] Directions are chosen uniformly randomly.
- We scale decoder latent directions to be unit norm at initialization (and also after each training step), following Bricken et al. [2023].
- For baseline models we use torch default initialization for encoder magnitudes. For TopK models, we initialized the magnitude of the encoder such that the magnitude of reconstructed

[18]This is done only at initialization; we do not tie the parameters as in Cunningham et al. [2023]. This strategy is also presented in concurrent work [Conerly et al., 2024].

vectors match that of the inputs. However, in our ablations we find this has no effect or a weak negative effect (Figure 16).[19]

A.2 Auxiliary loss

We define an auxiliary loss (AuxK) similar to "ghost grads" [Jermyn and Templeton, 2024] that models the reconstruction error using the top-k_{aux} dead latents (typically $k_{\text{aux}} = 512$). Latents are flagged as dead during training if they have not activated for some predetermined number of tokens (typically 10 million). Then, given the reconstruction error of the main model $e = x - \hat{x}$, we define the auxiliary loss $\mathcal{L}_{\text{aux}} = ||e - \hat{e}||_2^2$, where $\hat{e} = W_{\text{dec}} z$ is the reconstruction using the top-k_{aux} dead latents. The full loss is then defined as $\mathcal{L} + \alpha\mathcal{L}_{\text{aux}}$, where α is a small coefficient (typically $1/32$). Because the encoder forward pass can be shared (and dominates decoder cost and encoder backwards cost, see Appendix D), adding this auxiliary loss only increases the computational cost by about 10%.

We found that the AuxK loss very occasionally NaNs at large scale, and zero it when it is NaN to prevent training run collapse.

A.3 Optimizer

We use the Adam optimizer [Kingma and Ba, 2014] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a constant learning rate. We tried several learning rate decay schedules but did not find consistent improvements in token budget to convergence. We also did not find major benefits from tuning β_1 and β_2.

We project away gradient information parallel to the decoder vectors, to account for interaction between Adam and decoder normalization, as described in Bricken et al. [2023].

A.3.1 Adam epsilon

By convention, we average the gradient across the batch dimension. As a result, the root mean square (RMS) of the gradient can often be very small, causing Adam to no longer be loss scale invariant.We find that by setting epsilon sufficiently small, these issues are prevented, and that ε is otherwise not very sensitive and does not result in significant benefit to tune further. We use $\varepsilon = 6.25 \times 10^{-10}$ in many experiments in this paper, though we reduced it further for some of the largest runs to be safe.

A.3.2 Gradient clipping

When scaling the GPT-4 autoencoders, we found that gradient clipping was necessary to prevent instability and divergence at higher learning rates. We found that gradient clipping substantially affected $L(C)$ but not $L(N)$. We did not use gradient clipping for the GPT-2 small runs.

A.4 Batch size

Larger batch sizes are critical for allowing much greater parallelism. Prior work tends to use batch sizes like 2048 or 4096 tokens [Bricken et al., 2023, Conerly et al., 2024, Rajamanoharan et al., 2024]. To gain the benefits of parallelism, we use a batch size of 131,072 tokens for most of our experiments.

While batch size affects $L(C)$ substantially, we find that the $L(N)$ loss does not depend strongly on batch size when optimization hyperparameters are set appropriately (Figure 12).

A.5 Weight averaging

We find that keeping an exponential moving average (EMA) Ruppert [1988] of the weights slightly reduces sensitivity to learning rate by allowing slightly higher learning rates to be tolerated. Due to its low cost, we use EMA in all experiments. We use an EMA coefficient of 0.999, and did not find a substantial benefit to tuning it.

[19] Note that the scaling factor has nontrivial interaction with n, and scales between $\Theta(1/\sqrt{k})$ and $\Theta(1/k)$. This scheme has the advantage that is optimal at init in the infinite width limit. We did not try simpler schemes like scaling by $\Theta(1/\sqrt{k})$.