AIAA 5047 Project Report

Ziqin Gong

Abstract

This project explores the functional modularity of large language models (LLMs) using sparse autoencoders (SAEs). This project investigates if features that activate on either Chinese or English text exhibit geometric separability. JumpReLU SAEs were trained on activations from the 1st, 11st, and 21st layers of the Qwen2 7B model, using Chinese and English text from the CulturaX dataset. Although results produced by the SAE trained on layer 21 were most satisfactory due to limited time and resources, they demonstrate a clear geometric separation between features that primarily activate on Chinese versus English text. This spatial structure suggests that LLMs learn language-specific features that are not only functionally distinct but also geometrically co-located in the activation space.

1 Introduction

The emergence of large language models (LLMs) has brought a great advance in building powerful AI systems that are capable of following user instructions and conducting various complex tasks (Bai et al., 2022; Putta et al., 2024; OpenAI, 2024). However, as LLMs are very deep neural networks with billions of parameters, it is difficult to explain how they make predictions and decisions, thus posing great threat to safety (Hendrycks et al., 2023). A line of work seeks to mitigate these risks by diving into individual neurons to obtain verifiable explanations of a small portion of neural networks (Cammarata et al., 2021; Wang et al., 2022; Elhage et al., 2021), but a major challenge facing them is polysementicity, a phenomenon that neurons activate for multiple unrelated types of features (Olah et al., 2020). Besides, Elhage et al. (2023) suggests that features tend not to align with the neuron basis for some types of activations, such as the residual stream of a transformer.

Several recent work has demonstrated that a significant part of the internal activations of neural networks can be represented by sparse linear combinations of vectors in the activation space (Elhage et al., 2022; Olah et al., 2020; Park et al., 2024), and sparse autoencoders (SAEs) are a promising approach to identifying these base vectors in an unsupervised way (Bricken et al., 2023; Cunningham et al., 2023; Gao et al., 2024; Templeton et al., 2024). However, SAEs haven't been through sufficient exploration and there still remain some open problems to be resolved (Templeton et al., 2024). Fortunately, Lieberum et al. (2024) trained and released a comprehensive suite of JumpReLU SAEs (Rajamanoharan et al., 2024) on layers of Gemma 2 models (Team et al., 2024), together with detailed training setups that provides a comprehensive guide for the community to train their own SAEs. On top of this suite, Li et al. (2024) conducted a thorough study on the structure of these SAE features that should represent some concepts learned by Gemma 2 models, and found that at all "atomic", intermediate, and "galaxy" scales these features demonstrate meaningful structures.

This project is based on the intermediate-scale brain-like structure found by Li et al. (2024) and aims at discovering a more granular functional modularity. Specifically, Li et al. (2024) discovers that features in the SAE point cloud that tend to fire together within blocks of texts are also geometrically co-located in functional "lobes". Figure 2 of Li et al. (2024) (see Figure 1) shows an example that the point cloud breaks into two parts that activate on code/math documents and English language documents respectively. This project then takes a step further into the language lobe by trying to discover whether there exists a partition between different languages and gets a positive answer.

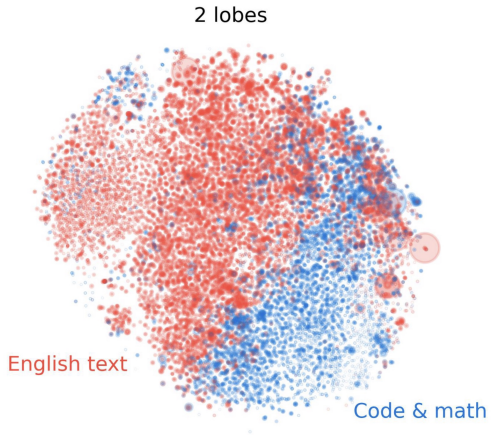

Figure 1: Part of Figure 2 of Li et al. (2024).

2 Preliminaries

2.1 Sparse autoencoders

Given an input vector $\mathbf{x} \in \mathbb{R}^n$, a SAE decomposes and reconstructs this vector using a pair of encoder and decoder functions $(f_{\text{enc}}, f_{\text{dec}})$ defined by

$$\mathbf{z} = f_{\text{enc}}(\mathbf{x}) := \sigma\left(W_{\text{enc}}\mathbf{x} + \mathbf{b}_{\text{enc}}\right), \tag{1}$$

$$\hat{\mathbf{x}} = f_{\text{dec}}(\mathbf{z}) := W_{\text{dec}}\mathbf{z} + \mathbf{b}_{\text{dec}}, \tag{2}$$

where σ is an activation function that ensures the non-negativity of linear weights $\mathbf{z} \in \mathbb{R}^m$. Although ordinary autoencoders tend to have $m < n$, here we set $m \gg n$ because polysemanticity suggests that more features than the number of neurons are learned by the model. When $\mathbf{x}$ is the internal activation of a language model, $\mathbf{z}$ is a set of linear weights that specify how to combine columns of W_{dec} to reproduce $\mathbf{x}$, and columns of W_{dec} are directions into which the SAE decomposes $\mathbf{x}$, thus representing conceptual features learned by the original language model. Sparsity of $\mathbf{z}$ is enforced by adding a regularization term to the reconstruction loss, and the resulting loss function for training SAEs is

$$\mathcal{L}\left(W_{\text{enc}}, W_{\text{dec}}, \mathbf{b}_{\text{enc}}, \mathbf{b}_{\text{dec}}\right) = \left\|\mathbf{x} - \hat{\mathbf{x}}\right\|_2^2 + \lambda \left\|\mathbf{z}\right\|_0, \tag{3}$$

while some implementations use an l_1 penalty (Cunningham et al., 2023; Gao et al., 2024), which has some disadvantages like penalizing feature magnitudes in addition to sparsity and requiring an additional constraint on SAE parameters to enforce sparsity (Rajamanoharan et al., 2024).

2.2 JumpReLU SAEs

Following Lieberum et al. (2024), this project adopts JumpReLU SAEs as they show a slight Pareto improvement over ReLU SAEs (Cunningham et al., 2023) and TopK SAEs (Gao et al., 2024). The JumpReLU activation uses a shifted Heaviside step function parameterized by a threshold parameter $\boldsymbol{\theta}$, and the activation function is

$$\sigma(\mathbf{y}) := \mathbf{y} \odot H(\mathbf{y} - \boldsymbol{\theta}), \tag{4}$$

where $\odot$ denotes element-wise multiplication, and H is the Heaviside step function, whose value is 1 if its input is positive and 0 otherwise, and with a little abuse of notation, extends to vectors. Intuitively, the JumpReLU leaves out features whose pre-activations are below certain thresholds learned for each feature.

Note that the loss function specified by Equation 3 provides no gradient signal for training the threshold parameter $\boldsymbol{\theta}$ because $\boldsymbol{\theta}$ only appears within the Heaviside step function. So this project uses straight-through-estimators (STEs; Bengio et al. (2013)) to train $\boldsymbol{\theta}$, following Rajamanoharan et al. (2024) and Lieberum et al. (2024). This introduces an additional hyperparameter, the kernel density estimator bandwidth ϵ, which controls the quality of gradient estimates used to update $\boldsymbol{\theta}$.

3 Implementation

3.1 Open-source libraries

This project uses TransformerLens (Nanda and Bloom, 2022) to collect internal activations of LLMs inferencing on some text. This project also uses pandas (McKinney, 2010) version 2.2.3 (The pandas development team, 2024) and fastparquet to save and load text data. For implementing and training SAEs, this project uses JAX (Bradbury et al., 2024) and Flax (Heek et al., 2024) as well as some libraries within their ecosystem for JAX's incredible speed and elegance compared to PyTorch (Ansel et al., 2024).[1] For post-processing of learned features, like clustering, dimension reduction, and visualization, this project uses NumPy (Harris et al., 2020), Scikit-learn (Pedregosa et al., 2011), and Matplotlib (Hunter, 2007). For a comprehensive list of dependencies, please refer to the GitHub repository that holds source code of this project.

3.2 Base LLM

Since this project focuses on identifying partitions between different languages within the SAE point cloud, a multilingual LLM is required. Due to my personal interest in Chinese and English, and the outstanding Chinese capabilities of the Qwen family, this project chooses Qwen2 7B (Yang et al., 2024) from which activations are collected.[2]

3.3 Dataset

Chinese and English text The best choice for text on which activations are collected is to collect from those on which the base LLM was pretrained. However, since Yang et al. (2024) doesn't release details about training data, alternatives are needed. Finally this project chooses Chinese and English text from a single multilingual dataset, CulturaX (Nguyen et al., 2023), for consistency of data quality between these two languages.

Location Among the 28 layers of Qwen2 7B, this project chooses activations on the post MLP residual stream of the 1st, 11st, and 21st layers (indexed from zero) based on my personal preference, since there is not enough time and resources to train SAEs for every layer and location.

Amount of data Due to storage limitations, this project accumulates activations for approximately 10M token per language and layer, resulting in a total of over 20 million tokens per layer for training purposes, while Lieberum et al. (2024) consumed at least 4B tokens for training SAEs of equal size ($m = 2^{14} = 16384$). This process consumed 802GB of storage capacity. For evaluation, we collected 0.1M activations for each language and layer.

Preprocessing All collected activations are normalized by a fixed scalar to have unit mean squared norm, following Lieberum et al. (2024). This helps more reliable transfer of hyperparameters between layers, since norms of original activations may vary over multiple orders of magnitude. Since this project doesn't study LLM's behavior with activations replaced by reconstructed ones, this scalar is of no later use.

3.4 JumpReLU SAE

Implementation of JumpReLU SAE generally follows Rajamanoharan et al. (2024) and Lieberum et al. (2024), with a few details worth noting.

Straight-through-estimators Thanks to the flexibility of JAX's API, straight-through-estimators can be easily implemented by registering custom backward functions of the Heaviside step function and JumpReLU activation function using `jax.custom_vjp`. For full Python code please refer to `src/models/utils.py` inside the GitHub repository.

[1] But TransformerLens uses PyTorch internally.

[2] Since TransformerLens hasn't supported Qwen2.5, this project uses Qwen2 7B, which also surpasses Gemma 2 on Chinese tasks.

Normalization of W_{dec}'s columns When l_1 penalty is adopted in the loss function (3), additional normalization on the columns of W_{dec} is required because scaling down encoder parameters and scaling up decoder parameters accordingly could arbitrarily shrink feature magnitudes and thus the l_1 penalty (Rajamanoharan et al., 2024). Since this project adopts the l_0 penalty, there is no need for the additional normalization. However, as this project strictly follows Lieberum et al. (2024), which always renormalizes W_{dec} for unit-norm columns, normalization is done after every gradient update.

4 Results

Due to limitations of time and resources, this project fails to reproduce results comparable to Figure 2 of Lieberum et al. (2024). Among SAEs trained for those three aforementioned layers, only the one trained on activations collected at layer 21 produces satisfactory results within due time, whereas the sparsity loss of the other two both fail to decrease to a reasonable level. Therefore, this section will mostly focus on results produced by the SAE trained for layer 21.

4.1 Training dynamics

Figure 2 shows the training dynamics of these three SAEs for different layers. Because SAEs have a much larger latent space than the activation space, it is very easy for them to reconstruct the input perfectly, indicated by the fast drop of the reconstruction loss to zero in the beginning. In contrast, they suffer very high sparsity loss, indicating that they use more features than the number of neurons the original LLM has for each layer to reconstruct activations. During the subsequent training process, the decrease of the sparsity loss is accompanied by a slight increase of the reconstruction loss, indicating the sparsity-fidelity trade-off.

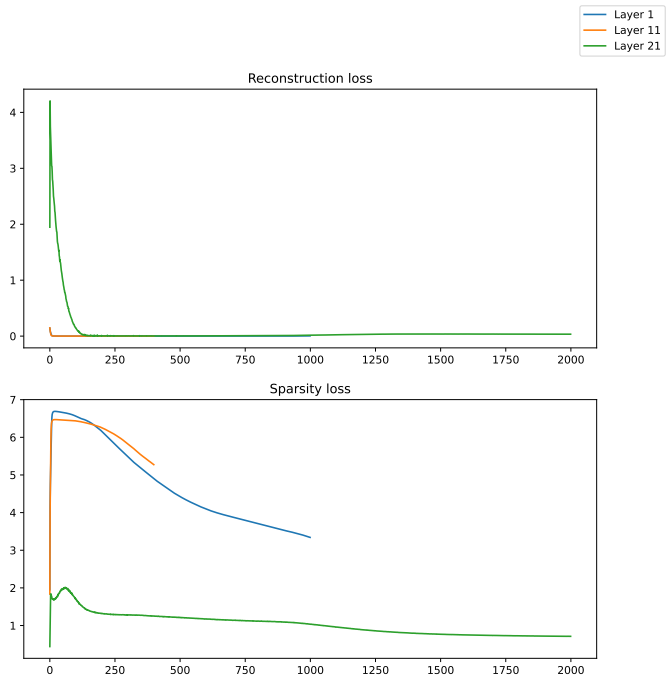

Figure 2: Training curves of SAEs on three layers. The sparsity coefficient λ is 0.001, so the unit of the second plot's y-axis would be 1k if it represents the number of learned features activating for reconstruction. Training curves of SAEs for layer 1 and 11 are truncated due to insufficient training time.

4.2 Partition of the language lobe

This project follows Section 4 of Li et al. (2024) to investigate if SAE features that are functionally similar with respect to Chinese or English are also geometrically similar. Specifically, a histogram of SAE feature co-occurrence is computed by feeding activations of tokens from both Chinese and English text to the SAE and counting features as co-occurring if they both fire within the same chunk of 256 tokens. This process is conducted on 40 chunks for each language, selected from the 0.2M activations collected for evaluation. Given this histogram, an affinity score is calculated between each pair of SAE features, followed by spectral clustering on the resulting afffinity matrix.

To assess which language each lobe specializes in, for each chunk of 256 tokens, the lobe with the highest proportion of its features firing is recorded. Thus, for each language, for each 256-token chunk within text of that language, the lobe with the highest proportion of its SAE features firing is recorded, and the resulting histogram of which lobes were maximally activating across each language can be used to determine which language each lobe corresponds to.

Li et al. (2024) experimented with five kinds of co-occurrence-based affinity with Phi coefficient being the best one. Hence this project presents the main result with Phi coeffcient, but also provides those produced by the other four measures. Note that there exist features that fire in all chunks or don't fire at all. Since both cases are undefined behavior with respect to Phi coeffcient, these features are filtered out.

Figure 3 shows features found by the SAE for layer 21, colored according to which lobe they belong to. Features that tend to fire together on Chinese text are seen to gather in a small region, surrounded by those firing on English text. This indicates that LLMs do learn different features for different languages, and features corresponding to each language are also co-located geometrically. Figure 4 contains results calculated by different affinity measures, which also show similar geometry.

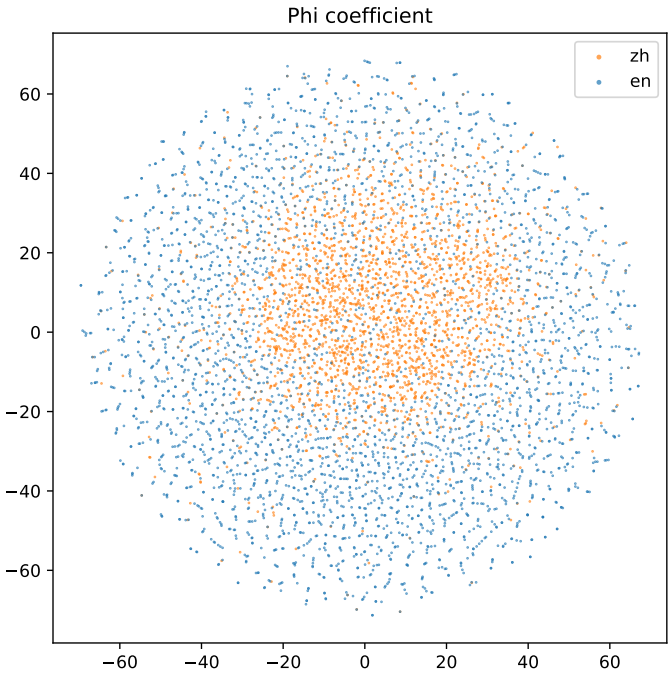

Figure 3: Features of the SAE trained for layer 21 down-projected to 2D with t-SNE, color-coded according to the language on which they fire the most. Clusters are computed by Phi coeffcient.

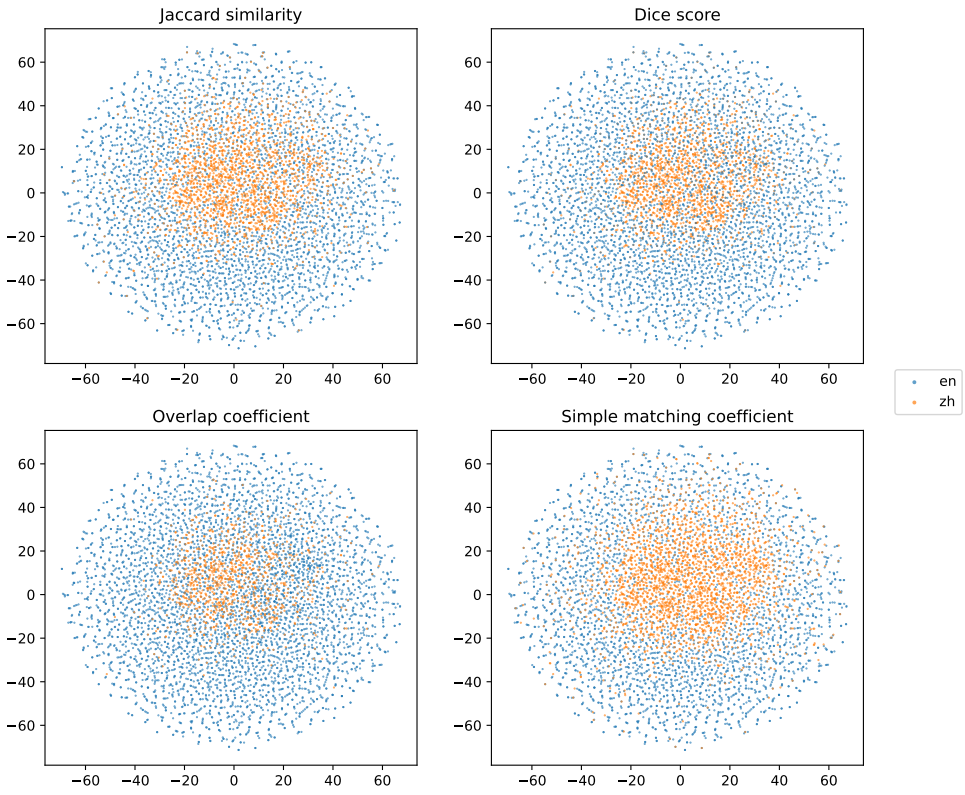

Figure 4: Features of the SAE trained for layer 21 down-projected to 2D with t-SNE, color-coded according to the language on which they fire the most. Clusters are computed by affinity measures specified by titles.

5 Conclusion

This project aims to investigate the functional modularity of LLMs by examining features learned by SAEs trained within a multilingual context. Specifically, this project seeks to determine if SAE features that activate for either Chinese or English text exhibit geometric separability, thereby revealing potential language-specific functional modules within LLMs. Inspired by the intermediate-scale structure identified by Li et al. (2024) and employing JumpReLU SAEs trained on activation of Qwen2 7B following setups of Lieberum et al. (2024), this project explores partition of directions in the activation space of layer 21 that is successfully trained compared to the other layers.

The findings demonstrate a clear geometric separation between SAE features primarily activating on Chinese text versus those activating on English text. This separation, visualized through t-SNE and spectral clustering based on a variety of affinity measures, suggests that LLMs learn language-specific features that are not only functionally distinct but also spatially organized within the activation space.

Due to the intense computational requirements for training SAEs and limited time, results presented in this report are produced by an underfitted SAE, although certain conclusions can still be drawn. Future work could expand this exploration to other languages, and languages within the same language family, like Indo-European languages. As to implementation, a critical bottleneck constraining the training speed is disk reading and data shuffling. Since Python used by this project is still under global interpreter lock (GIL), an asynchronous version of the training process won't parallelize data

processing and gradient updating. Future work may use tools provided by PyTorch for efficient data processing.

References

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI Feedback, December 2022.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation, August 2013.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2024.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*, October 2023.

Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve Circuits. *Distill*, 6(1):e00024.006, January 2021. ISSN 2476-0757. doi: 10.23915/distill.00024.006.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse Autoencoders Find Highly Interpretable Features in Language Models, October 2023.

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. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy Models of Superposition, September 2022.

Nelson Elhage, Robert Lasenby, and Christopher Olah. Privileged Bases in the Transformer Residual Stream. *Transformer Circuits Thread*, March 2023.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, June 2024.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024.

Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. An Overview of Catastrophic AI Risks, October 2023.

J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Yuxiao Li, Eric J. Michaud, David D. Baek, Joshua Engels, Xiaoqing Sun, and Max Tegmark. The Geometry of Concepts: Sparse Autoencoder Feature Structure, October 2024.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2, August 2024.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a.

Neel Nanda and Joseph Bloom. TransformerLens, 2022.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages, September 2023.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom In: An Introduction to Circuits. *Distill*, 5(3):e00024.001, March 2020. ISSN 2476-0757. doi: 10.23915/distill.00024.001.

OpenAI. Learning to Reason with LLMs, September 2024.

Kiho Park, Yo Joong Choe, and Victor Veitch. The Linear Representation Hypothesis and the Geometry of Large Language Models, July 2024.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents, August 2024.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping Ahead: Improving Reconstruction Fidelity with JumpReLU Sparse Autoencoders, August 2024.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison,

Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving Open Language Models at a Practical Size, October 2024.

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.

The pandas development team. Pandas-dev/pandas: Pandas. Zenodo, September 2024.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 small, November 2022.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 Technical Report, September 2024.