Winner-Takes-All Sparse Autoencoders for Enhancing Mechanistic Interpretability

Shitij Govil
Georgia Institute of Technology
sgovil9@gatech.edu

Aaron Trinh
Georgia Institute of Technology
atrinh31@gatech.edu

Joshua Wang
Georgia Institute of Technology
jwang3453@gatech.edu

Abstract

Mechanistic interpretability aims to decode Large Language Models (LLMs) by understanding their internal representations. As LLMs are deployed in increasingly wider and critical applications, improving their transparency has become a pressing priority. Sparse Autoencoders (SAEs) have proven effective by isolating individual features from polysemantic neurons due to their ability to uncover interpretable internal features. Popular approaches like TopK and BatchTopK SAEs enforce fixed sparsity constraints, which can lead to dead features-neurons that never activate and don't contribute to the learned representations. In this work, we propose Winner-Takes-All (WTA) SAEs which promote lifetime sparsity by ensuring every feature is active during training. Instead of applying sparsity constraints per sample, WTA dynamically enforces sparsity per feature across training batches. We evaluate WTA SAEs on the SAEBench framework against other SAE baselines. While WTA achieves comparable sparsity levels and eliminates dead features, it underperforms in reconstruction fidelity and feature subspace alignment. Our findings highlight a tradeoff between enforcing lifetime sparsity and preserving model reconstruction. We conclude by highlighting potential future directions such as hybrid approach to per-sample and per-feature sparsity strategies, adaptive sparsity shcedules, and better optimization objectives to balance feature use and functional accuracy. Github: https://github.com/sgovil5/SAE

1. Introduction

Large Language Models (LLMs) have recently become popular due to their ability to handle a wide variety of natural language tasks, such as understanding [4] and reasoning [11]. However, their internal workings still remain largely unknown, due to factors such as their large parameter size and architectural complexity. As LLMs continue to be integrated into critical areas of society, it is essential that LLMs also become more interpretable in order to effectively understand, safeguard, and steer outputs.

Recent work in mechanistic interpretability has sought to address these concerns through improving understanding of neural network activations. [3] proposed to use sparse autoencoders (SAEs) to sparsely reconstruct internal activations. SAEs are autoencoders that are also additionally penalized for latent vector sparsity during training, ensuring that desired outputs are determined by a small number of activated neurons. This method shows significant advancement in interpretability by allowing for the extraction of single features as opposed to attempting to extract features from polysemantic neurons. However, the effectiveness of these methods depends heavily on maintaining a diverse and active set of features. Dead or underutilized features can obscure the learned structure and complicate downstream analysis such as circuit discovery.

Another popular approach in mechanistic interpretability is to study the circuitry of neural networks [10]. Identifying groups of neurons that are frequently activated together in so-called "circuits" can deepen understanding of the features learned. Such circuits form subgraphs within the neural network to perform computation, such as detecting curves in an image. With this understanding, specific circuits can be manipulated to alter the output of the network towards more desirable values (i.e. steering). Recent work has introduced methods for automatically discovering circuits at scale, which was able to correctly recover all of the previously manually identified circuit component types in GPT-2 [2].

A complementary technique in circuitry analysis is activation patching [14], which localizes model circuits by:

1. Running a forward pass on an unmodified prompt
2. Running a forward pass on a modified prompt
3. Running a forward pass on the modified prompt, replacing the activations with those from (1), and observing the result

Thus, circuits relevant to a prompt can be identified by whether or not their patched activations substantially change the output. By inserting such activation patches across layers and circuits, mapping the internal workings of a neural network becomes feasible.

For the scope of this paper, we focus on using sparse autoencoders for mechanistic interpretability. While prior works use a hard, parameterized constraint for activations in SAE layers [6], this may lead to dead features and limit the expressibility of the reconstruction, thus limiting the feature interpretability of the resulting model. We present a new, alternative constraint that promotes lifetime sparsity by using the Winner Take All (WTA) activation [9] for the SAE. Rather than allowing some underutilized features to possibly die, the WTA activation enforces sparsity at the feature level, ensuring all features have a chance to be learned.

2. Related Work

2.1. Sparse Autoencoders for Mechanistic Interpretability

Advances in mechanistic interpretability have recently identified sparse autoencoders as an exceptional tool for interpreting activation functions, particularly in LLMs [6]. Sparse autoencoders are an architecture generally inserted between layers of a neural network that compress and reconstruct internal activations while enforcing a sparsity constraint on the representation space [8]. In large language models, it is commonly believed that individual neurons hold polysemantic meaning, they represent many features at once. To leverage sparse autoencoders for interpretability in large language models, we employ an encoder that increases the dimension of the representation space to be larger than the dimension of the input. When combining this with a sparsity constraint, we encourage the learned representations to represent single features, or monosemantic concepts.

2.2. TopK Sparse Autoencoders

Recent work has found that an effective constraint is to utilize the TopK operation [6]. A naive sparsity constraint, such as the L1 loss, may be difficult to train. Rather, the TopK operation selectively keeps the highest k activations per sample, zeroing out the rest of the activations. The encoder is defined as:

$$z = \text{TopK}(\mathbf{W}_{\text{encoder}}\mathbf{x} + \mathbf{b}_{\text{encoder}}) \tag{1}$$

This forces the encoded representations to have a set L0 norm, resulting in latents that represent single features. These latents can then be examined at test time by conditioning on different inputs. These latents have been shown to result in much more interpretable features, such as closing sets of quotes and induction of capitalized phrases [6]. Additionally, an additional loss term $\mathcal{L}_{aux}$ is calculated as the reconstruction error using k_{aux} dead latents.

2.3. JumpReLU Sparse Autoencoders

JumpReLU is a simple modification to the classic ReLU activation function, which introduces a jump discontinuity at the non-differentiable point in a traditional ReLU [5].

$$\text{JumpReLU}_\theta(z) := zH(z - \theta) \tag{2}$$

where:

- $H(z)$ is the Heaviside step function, a function that is one when $z > 0$ and zero when $z < 0$
- θ is the JumpReLU threshold, similar to that in ReLU

Originally, JumpReLU was introduced as a way to disrupt gradient-based adversarial attacks. Because of the discontinuous nature of JumpReLU, attacks have more difficulty understanding the loss landscape of the original model. For example, perturbation-based attacks typically try to maximize loss by perturbing to follow the gradient. Since other activations are piecewise linear and continuous, the gradient is defined almost everywhere, making it easy to correctly choose perturbations. JumpReLU's discontinuity mitigaes this by making activations less predictable.

JumpReLU SAEs were shown to result in both an improvement in training time and reconstruction fidelity. As JumpReLU is an element-wise activation, it can be applied faster than TopK, which requires partially sorting the activations to find the top k. Rajamanoharan et. al empirically demonstrated that using JumpReLUs resulted in higher reconstruction fidelity when run on Gemma 2 9B [13].

2.4. BatchTopK Sparse Autoencoders

An extension to TopK SAEs is BatchTopK [1], which selects the top k activations per batch, rather than per sample. Accordingly, the loss of a batch $\mathbf{X}$ becomes:

$$\begin{aligned}\mathcal{L}(\mathbf{X}) = \|\mathbf{X} - \text{BatchTopK}\left(\mathbf{W}_{enc}\mathbf{X} + \mathbf{b}_{enc}\right) \\ \mathbf{W}_{dec} + \mathbf{b}_{dec}\|_2^2 + \alpha\mathcal{L}_{aux}\end{aligned} \tag{3}$$

where $\mathbf{W}_{enc}$ and $\mathbf{b}_{enc}$ are the encoder weights and biases, and $\mathbf{W}_{dec}$ and $\mathbf{b}_{enc}$ are the decoder weights and biases. $\mathcal{L}_{aux}$ is the same auxiliary loss as in TopK and is scaled by a parameter α.

As the sparsity constraint is enforced at the batch level rather than the sample level, each sample can have a variable number of active features, enhancing learning. But, the sparsity constraint is still enforced overall.

Since the overall activation is dependent on the activations from multiple samples, BatchTopK cannot be directly used at inference time. Instead, the authors propose using JumpReLU activation while zeroing activations under a threshold estimated as the average minimum positive activations across training batches. Essentially, this "learns" the typical activation value that would result in a TopK selection, and applies it to inference time batches.

2.5. Winner-Take-All Autoencoders

Winner-Take-All (WTA) autoencoders are a method proposed by Makhzani and Frey to reduce the complexity of sparse autoencoders and provide additional flexibility for specifying sparsity targets [9]. Rather than enforcing sparsity constraints by penalizing based on the KL divergence, WTA proposes that a certain proportion of dictionary features, ordered by activity, are directly selected in each batch, while the rest are zeroed. Compared to TopK, WTA ensures that all dictionary features receive an equal amount of activation so that they are still learned. This leads to more efficient training, and importantly, a guaranteed lifetime sparsity rate. Experiments have shown that deep sparse networks trained using WTA perform well on MNIST. Additionally, a convolutional modification, CONV-WTA, improves SVM performance on CIFAR-10.

3. Technical Approach

This section details the methodology used by our approach, Winner-Takes-All Sparse Autoencoder, the data processing pipeline, the baseline models, and the evaluation framework.

3.1. Winner-Takes-All SAE Methodology

Our core contribution is the WTA SAE which is designed with the goal to learn interpretable features from LLM activations, using a sparsity mechanism inspired by Makhzani & Frey but adapted for use in an SAE. It uses a standard autoencoder framework with both an encoder and decoder. The encoder maps input activations from the LLM's residual stream to a higher dimension dictionary space via a standard linear transformation followed by a ReLU. The decoder reconstructs the original input activations from the sparse dictionary representations with a linear transformation. The encoder and decoder weights are initialized with a tied relationship but are optimized independently. The training objective for the WTA SAE aims to minimize the reconstruction error between the original input activations X and their reconstructions $\hat{X}$ obtained after encoding and decoding:

$$\mathcal{L}(X) = \|X - \text{WTA}(W_{enc}X + b_{enc})W_{dec} + b_{dec}\|_2^2$$

Here the reconstructions $\hat{X}$ are equal to $\text{WTA}(W_{enc}X + b_{enc})W_{dec} + b_{dec}$.

The core innovation lies in the sparsity enforcement applied to the dictionary after the initial encoding phase. Previous methods like TopK SAEs enforce a fixed sparsity budget of the top K dictionary items *per input sample* whereas BatchTopK SAEs apply the budget of the top K dictionary items across the entire batch. However, the WTA mechanism enforces sparsity individually for each dictionary feature across the samples in a batch. The algorithm has the following steps:

1. For a given training batch of size B and a previously set target sparsity rate we apply the target activation count as $k_{per\_feature} = \max(1, \lfloor B \times \text{sparsity rate} \rfloor)$
2. After we get the encoded activations with shape $B \times$ dictionary size, we consider each feature in the dictionary j independently.
3. For each feature j we pick the top $k_{per\_feature}$ value activations among the B samples in the batch, and then we zero out any activations that fall below the threshold to enforce sparsity.

This process yields a dictionary where the number of active features per sample can vary dynamically, which allows for potentially better interpretability compared to fixed sparsity methods like TopK.

We also note that the batch-dependent nature of the WTA selection threshold is not suitable for inference on individual samples. We create a global activation threshold θ during training that's maintained as an Exponential Moving Average of the minimum positive activation obeserved across training batches. During the initial training phase, θ is used to sparsify activations before the WTA step. During inference, the WTA step is removed entirely so the sparsity is set by applying θ as a threshold element-wise after the ReLU activation.

Lastly, to encourage feature diversity, the decoder weights are constrained to unit norm after each step of the Adam optimizer. Additionally, gradients applied to the decoder weights are projected to be orthogonal to existing decoder directions before the update to prevent feature collapse.

3.2. Evaluation Framework and Metrics

We employed the standardized SAEB ench framework for a comprehensive evaluation across multiple metrics. Here the performance is measured against baselines sparse autoencoders like TopK and BatchTopK.

3.2.1. Core Properties

These metrics quantify the main trade-off between sparsity and reconstruction fidelity for the SAE.

- **L0 Norm (Sparsity):** Measures the average number of non-zero features in the sparse activation vector. This quantifies the effective sparsity by the SAE.
- **Reconstruction Fidelity:** Measures how accurately the original activation is reconstructed after decoding. This is

done through Mean Square Error: $||x - \hat{x}||^2$. We also look at the Normalize MSE (NMSE) calculated as $\frac{\text{MSE}}{\text{Var}(X)}$ and the Explained Variance which is 1-NMSE.

- **Feature Density:** Measures the activation frequency of individual dictionary features. Helps identify ”dead” and ”hyperactive” features.

3.2.2. Functional Impact

This measures how the SAE's sparse representations affect the language model's downstream performance.

- **Cross-Entropy Degradation:** Measures the model's prediction loss when the original activation at the SAE's layer is replaced by the reconstructed activations.

3.2.3. Feature Quality

These metrics evaluate the ”usefulness” of the learned features beyond just reconstruction.

- **Subspace Cosine Reconstruction (SCR):** Measures the cosine similarity between the original activation and its projection onto a linear subspace spanned by the active features from the decoder vectors.
- **Total Projection Percentage (TPR):** Measures the squared norm of the projection of the original activations onto the active feature subspace normalized by the squared norm of the activations.
- **Sparse Probing:** Assesses the utility of the learned features. Linear classifiers serving as ”probes” are trained directly on the feature activations to predict previously determined linguistic properties on the input tokens.

4. Data

Our experiments utilize the OpenWebText dataset [7]. This corpus is a large-scale, publicly available replication of the WebText dataset, which was utilized to train GPT-2 [12]. We detail key characteristics of the dataset that led to its utilization.

4.1. Motivation

The dataset is a large, high quality dataset of English text scraped from the web, which was necessary for pretraining of large language models. It specifically attempts to replicate OpenAI's WebText dataset, which was not accessible for public use.

4.2. Composition

The dataset consists of over 8 million individual text documents, approximately 38 GB of data. The text content is sampled from URLs scraped from the social media platform Reddit. URLs are filtered by receiving a high ”karma” (greater than or equal to 3), which indicates interesting or useful content contained in the URL, determined by Reddit users. Thus, the dataset is a subset of the entire World Wide Web, biased towards more valuable content. Each instance of the dataset is raw text extracted from the webpage. It is important to note that due to the nature of the way the data set is collected, there is the possibility that toxic, societally biased, or offensive content may be included.

4.3. Collection Process

The dataset is, as previously discussed, collected by scraping web text from URLs posted on Reddit. URLs were collected from posted up to October 2020. If the post or comment had a karma of 3 or more, it was included in the dataset.

4.4. Preprocessing/cleaning/labeling

After extracting the web text, the preprocessing step involves local-sensitivity hashing to identify near duplicate documents, filtering out very similar documents.

4.5. Uses

The primary use of the dataset is for pretraining large language models. Since the dataset is unlabeled, the intended use is for unsupervised pretraining of such models. Due to the nature of collection (potential for harmful content as mentioned previously), usage should be limited to research purposes or carefully monitored environments.

5. Results

5.1. Experimental Setup

We train 3 SAEs: TopK, BatchTopK, WTA. For the LLM we use pythia-410m-deduped, which is a version of the original model trained on deduplicated data. The SAE is trained specifically on layer 3 using an open-source replication of the OpenWebText dataset used by OpenAI for GPT2. Each SAE is trained for 100,000,000 tokens with a batch size of 8192, leading to each SAE being trained for $\lfloor \frac{10000000}{8192} \rfloor = 12207$ steps. Each SAE has an expansion factor of 8, giving it a dictionary size (number of features per token) of 8192. Lastly, both TopK and BatchTopK are trained with a $k = 64$, where k is the number of active features per sample, whereas the WTA SAEs are trained with target sparsity rates over $[10^{-6}, 0.05]$.

5.2. WTA Sparsity Ablation

Figure 1 illustrates the distributions of cosine similarities between the corresponding encoder and decoder weight vector for each feature across differing training sparsity rates. A clear trend emerges: as the training sparsity increases, the median cosine similarity between encoder and decoder rises. At very low sparsity rates (1e-6, 5e-6) the median similarity is below 0.6 but rises steadily reaching medians above 0.8 for the higher sparsity rates. This trend aligns with the expectation that allowing features to activate

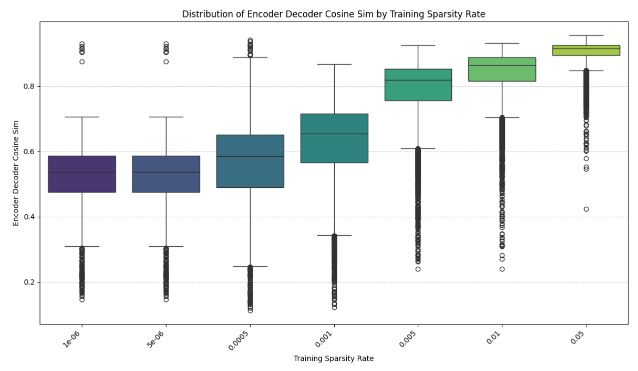

Figure 1. Cosine Simlarity Across Encoder/Decoder for WTA Sparsities

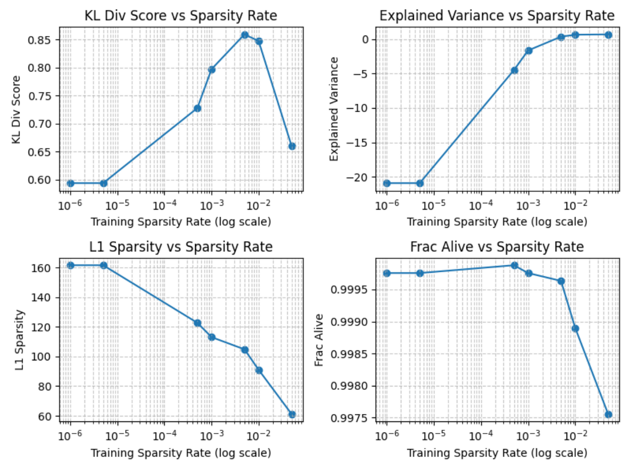

Figure 2. Core Metrics (KL Divergence, Explained Variance, L1 Sparsity, and Fraction Alive) for various WTA sparsity rates shown across a log scale

more frequently while training under WTA allows for better alignment between the encoder and the decoder, likely because of more consistent gradient signals. Each feature is zeroed out less often as the sparsity increases, giving it a greater chance to learn. It is interesting to note that the variance is significantly higher for the middle sparsity rates (0.005-0.01). This is likely a result of the fact that some features are firing very often based on the dataset and learning the best representations.

Figure 2 presents several core evaluation metrics plotted against a logarithmic scale of the sparsity rate:

- **KL Divergence Score (Model Behavior Preservation):** The KL divergence score, which measures how well the SAEs output matches the original model's input distribution, shows an unexpected non-monotonic relationship. It remains low at very low sparsity rates and increases with the sparsity rate (up till 0.005) peaking at a value of 0.85, before sharply dropping at the highest tested rate 0.05. This suggests there's an optimal range where WTA SAE best preserves the model's predictive behavior.
- **Explained Variance (Reconstruction Quality):** Explained Variance exhibits a strong positive correlation with the sparsity rate, as expected. At extremely low rates EV is highly negative, which can be explained by the fact that most neurons don't have the chance to learn much and thus the reconstruction is worse. Although as the sparsity rate increases, the EV improves dramatically becoming positive around a rate of 0.005 and reaching a peak of 0.68 at a sparsity of 0.05.
- **L1 Sparsity:** The L1 norm of the sparse feature activations is correlated with the number of active features and their magnitudes. This shows a clear inverse relationship between L1 norm and sparsity rate, where the L1 norm is very high (160) at low sparsity rates (1e-6) but decreases significantly to 60 at the highest sparsity 0.05. This is somewhat counter-intuitive, since a higher sparsity rate (more allowed activations per feature) should lead to overall denser activations. However, this suggests that while individual features activate more often with higher rates, the magnitudes of the activations decreases. Additionally, it could also mean that the total number of features relevant for a reconstruction might decrease.
- **Fraction Alive:** The fraction of dictionary features that activates at least once during evaluation is 1 across most sparsity rates. This aligns with WTA's goal to prevent dead features. However, there is a small decine to 0.9975 at the highest sparsity rates, but this might be coincidental or a minor feature collapse when competition is less stringent. Ultimately, the per-feature mechanism of WTA ensures that all features participate in representing a token.

The ablation reveals complex dependencies between the sparsity rate and core metrics. Increasing the sparsity rate improves reconstruction quality and encoder-decoder alignment, while also decreasing the L1 norm of activations. Model preservation behavior (via KL divergence) appears optimal within a given mid-range of sparsities. The results show that ensuring high feature participation (fraction alive) doesn't necessarily translate to a high-fidelity reconstruction of the activations. The optimal choice for the sparsity rate involves balanceing model preservation and explained variance. In this case, it seems like a 0.01 is a good sparsity rate.

5.3. Results Discussion

1 takes the comparative performance of the best WTA SAE (sparsity rate of 0.01) and against BatchTopK and TopK SAE baselines. The table can be analyzed like so:

- **Sparsity (L0 Norm):** All the method achieve similar L0 sparsity levels, close to around 63 active features per token. This indicates that this specific choice of WTA spar-

Table 1. SAE Performance Comparison (Pythia-410m, Layer 3)

Metric Category	Metric	WTA SAE	BatchTopK SAE	TopK SAE	Ideal Value	Notes
Sparsity	L0 Norm	63.27	**62.16**	64.0	Low	Avg. active features per token.
Reconstruction	Explained Variance (EV)	0.68	**0.88**	0.87	High (≈1)	Fraction of variance reconstructed.
Functional Impact	CE Loss Score	0.835	**0.976**	0.971	High (≈1)	Preservation of original CE loss.
	CE Loss with SAE	4.46	**3.38**	3.42	Low (≈CE w/o SAE)	Absolute CE loss. *CE w/o SAE = 3.19*
Feature Utility	Sparse Probe Acc.	0.929	**0.931**	**0.931**	High	Accuracy predicting properties.
Subspace Alignment	SCR (threshold 20)	0.092	0.112	**0.146**	High	Subspace Cosine Reconstruction.
	TPP (threshold 20)	0.062	0.106	**0.154**	High	Total Projection Percentage.

Note: Best performance highlighted in ***bold*** *for each metric (considering ideal value direction).*

sity gets an overall activation density comparable to the set k for TopK and BatchTopK.

- **Reconstruction Quality (Explained Variance):** Despite matching the L0 sparsity, WTA has a much lower reconstruction fidelity compared to the baselines. It's EV of 0.68 lags behind the 0.88 BatchTopK. This suggests that while WTA activates a similar number of features, it's choice of features being chosen is less effective at choosing the original activation vectors compared to the features chosen by the baselines. Additionally, we can presume that while WTA enforces better feature participation, it may fail to select the most semantically aligned features for reconstruction.
- **Functional Impact (CE Loss):** This difference in reconstruction quality impacts downstream task performance. The WTA SAE has a lower preservation (0.835) and a significantly higher raw CE Loss (4.46) compared to the baselines. This highlights that the information lost during WTA's reconstruction is functionally important for the language model's tasks.
- **Feature Utility (Sparse Probing Accuracy):** The gap in performance significantly narrows when considering feature utility. The sparse probing accuracy for WTA (0.929) is only marginally lower than for the baselines (0.931). This suggests that the individual features learned and activated by WTA still encode meaningful information relevant to linguistic properties, even if they can't reconstruct the full activation correctly.
- **Subspace Alignment (SCR, TPP):** The SCR and TPP further reinforce the issue caused by reconstruction quality. At the representative threshold evaluated for TPP (20), WTA shows signficantly lower scores compared to the baselines. This indicates that the linear subspace spanned by the features activated by WTA aligns less well with both the direction and magnitude of the original activation vector

6. Conclusion and Future Work

6.1. Conclusion

This paper investigated the viability of the Winner-Takes-All Sparse Autoencoder (WTA SAE) as a method for decomposing the activations of large language models into interpretable features. Inspired by prior Winner-Take-All concepts, this paper aimed to implement a novel per-feature sparsity mechanism and evaluated it against baseline sparsity constraints like TopK and BatchTopK. Our analysis revealed that while WTA SAE can achieve sparsity levels close to BatchTopK and TopK, this comes at a significant cost. WTA SAE showed lower reconstruction fidelity, worse cross-entropy loss, and worse KL divergence between the model with and without the SAE. Based on our evaluations on the Pythia-410m model in layer 3, the BatchTopK and TopK SAEs currently offer a more favorable tradeoff between sparsity and functional preservation. Ultimately, these findings indicate that optimizing sparsity mechanisms alone is insueefficient; understanding how sparsity affects feature selection and downstream utility is an open challenge.

6.2. Future Work

Currently the core limitation is related to translating per-feature sparsity to a meaningful reconstruction. Thus future work could explore:

- **Hybrid Sparsity:** Combining the per-feature WTA activations with a secondary global budget (per-sample/batch) might offer better control
- **Modified Objective:** Incorporate a loss term that encourages better reconstruction or penalize high feature density per sample
- **Adaptive sparsity:** Instead of deriving the number of active features from a fixed rate, making it adaptive based on some batch statistics might help

Additionally, this work could benefit from further empirical evaluation. Evaluating WTA on larger LLMs and different layers could reveal more information on whether it

Contributor	Contributions
Shitij Govil	Co-developed project idea and technical approach; implemented WTA SAE model; led experimental evaluation and ablation studies; co-wrote manuscript.
Aaron Trinh	Co-developed project idea and evaluation framework; contributed to baseline SAE implementations and experimental setup; co-wrote manuscript and coordinated analysis.
Joshua Wang	Co-developed project idea and method design; contributed to dataset preparation and model training; co-wrote manuscript and prepared results visualizations.

Table 2. Author contributions to the project. Work was divided equally among authors.

could be potentially useful. Lastly, a more extensive search across hyperparameters (steps run, bias updates, etc) could uncover configurations with better performance.

7. Addendum

The code for this project was based off two repositories: Dictionary Learning to train the SAEs and SAEBench to run evaluations on various benchmarks for the SAEs.

Table 2 shows the breakdown of contributions between the authors

References

[1] Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders, 2024. 3

[2] Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability, 2023. 2

[3] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. 2

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. 2

[5] N. Benjamin Erichson, Zhewei Yao, and Michael W. Mahoney. Jumprelu: A retrofit defense strategy for adversarial attacks, 2019. 3

[6] 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, 2024. 3

[7] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019. 5

[8] Alireza Makhzani and Brendan Frey. k-sparse autoencoders, 2014. 3

[9] Alireza Makhzani and Brendan Frey. Winner-take-all autoencoders, 2015. 3, 4

[10] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in. 2

[11] OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe,

Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. 2

[12] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 5

[13] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders, 2024. 3

[14] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, pages 12388–12401. Curran Associates, Inc., 2020. 2