SAE-ception: Iteratively Using Sparse Autoencoders as a Training Signal

Anonymous Author(s)
Affiliation
Address
email

Abstract

We explore whether post-hoc interpretability tools can be repurposed as a training signal to build models that are more interpretable by design. We introduce *SAE-ception*, a method that iteratively incorporates features extracted by a sparse autoencoder (SAE) as auxiliary targets in the training loop. Across three distinct settings — an MLP on MNIST, a vision transformer (ViT-H) on CIFAR-10, and ConvNeXt-V2 on ImageNet-1k — our method led to substantial gains in the clustering and separability of learned SAE features. These gains were evidenced by several metrics, such as improved silhouette scores and Davies-Bouldin indices. The effect on monosemanticity and task performance, however, is context-dependent. On the simpler MLP, the approach is a clear success, improving not only monosemanticity in both the base model and the SAE but also increasing the base model's final task accuracy by over 2.5%. On ViT-H, SAE-ception doubles the monosemanticity of the SAE — as measured by the uncertainty coefficient (U) — after a single cycle with only a 0.09% drop in task accuracy, but the base model's monosemanticity remains largely unchanged. While the gains in feature clustering and separability persist on ConvNeXt-V2, monosemanticity metrics remained largely stagnant: U shifted from a baseline of 0.28 to 0.31. We conclude that SAE-ception is a low-cost method that reliably enhances features for post-hoc analysis, making it a valuable tool for practitioners, though its ability to disentangle the base model's representations depends on the specific architecture and task. Determining the conditions under which it can consistently improve the internal monosemanticity of a base model remains a key direction for future exploration.

1 Introduction

A central challenge in deep learning is the opacity of a model's internal representations. While powerful, these representations are often tangled and difficult to map to human-understandable concepts, a problem exemplified by polysemanticity [10, 14, 23]. To address this, the field has developed powerful post-hoc analysis tools, most notably sparse autoencoders (SAEs), which decompose a model's dense activations into a sparse set of more interpretable features [3, 6, 24]. However, the utility of these tools is constrained by a fundamental limitation: they are a passive lens applied to a pre-trained, static model. Furthermore, this analysis is itself an imperfect and often painstaking process, requiring significant manual effort to interpret the complex features extracted from an already tangled model [1, 11, 16, 21]. Most critically, these methods can only analyze a messy representation; they cannot compel the model to learn a cleaner one in the first place.

In this work, we ask: can these post-hoc tools be repurposed as an active training signal to build models that are more interpretable by design? We introduce *SAE-ception*, a method that unifies model training with mechanistic interpretability. The core idea is an iterative loop: we train an SAE on a

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

model's activations, identify a set of salient sparse features, and then retrain the base model using the reconstructed version of these features as auxiliary targets. This process provides a direct incentive for the model to organize its internal representations in a way that aligns with the features discovered by the SAE. Our primary goal is to produce representations that are more amenable to analysis by improving the separability and clustering of learned features.

To achieve this, we also introduce *feature sharpening*, a technique to selectively focus the training signal on the most important features. By encouraging the model to prioritize these concepts, SAE-ception creates an internal feature space that is better structured for post-hoc examination.

Our primary contributions are:

1. We introduce SAE-ception, a novel training method that iteratively incorporates SAE-extracted features as auxiliary targets to improve a model's representational structure.
2. We demonstrate across three distinct datasets and architectures that SAE-ception reliably improves the clustering and separability of learned features, making them more amenable to post-hoc analysis.
3. We propose a new paradigm for model development where interpretability methods are used not just as passive analysis tools, but as an active optimization signal to create models that are interpretable by design.

An implementation of our method is available at `https://anonymous.4open.science/r/SAE-ception-57D3`.

2 Related Work

SAE-ception's goal of improving representations through iterative training distinguishes it from contemporary methods that integrate interpretability tools into the fine-tuning process. These approaches typically involve one-shot interventions or architectural changes, such as incorporating SAEs into Mixture-of-Experts layers for knowledge editing (Monet [20]), using SAEs to guide fine-tuning for targeted safety ablations (CAFT [5]), or performing a one-shot transfer of interpretability from a large model to a smaller one (Resa [26]). In contrast to these methods, SAE-ception cyclically refines the base model's entire feature space without altering its original architecture.

Our method also differs from regularization-based approaches. For instance, DecPO encourages sparse activations by adding a feature decorrelation penalty to the main loss function during preference optimization [27]. SAE-ception, however, provides guidance not through a penalty term, but via an explicit auxiliary target produced by an SAE.

3 SAE-ception

The core of SAE-ception is an auxiliary loss term, L_{aux}, that encourages a model's internal activations to align with sparse, disentangled features derived from an SAE. This process is applied iteratively over several training cycles.

3.1 Auxiliary Loss Function

At each retraining cycle, the model's standard task loss, L_{task}, is combined with an auxiliary loss, L_{aux}. Let x be an activation vector from the layer where SAE-ception is applied. Let f' be the sharpened features derived from our SAE — f' is pre-computed and held constant during the retraining step. The auxiliary loss is the cosine distance between these vectors:

$$L_{aux}(x, f') = 1 - cos(x, f') = 1 - \frac{x \cdot f'}{\|x\|_2 \|f'\|_2}$$

The total loss is a weighted sum controlled by hyperparameter λ_s:

$$L_{total} = L_{task} + \lambda_s L_{aux}$$

Minimizing L_{aux} drives the model's activations x to point in the same direction as the sharpened features f', effectively steering the model to learn the most pertinent information from the disentangled features discovered by the SAE.

3.2 Generating Sharpened Features

The target vectors f' are not the direct output of the SAE. Instead, they are "sharpened" to focus the training signal on the most class-relevant features. For a given trained model, $model_k$, and its corresponding SAE, SAE_k, we generate the targets f'_k as follows:

1. **Identify Top Class-Features:** First, for each class c in the dataset, we determine which SAE sparse features are most representative. We calculate the mean activation $\hat{a}_{c,i}$ for each sparse feature dimension i across all examples S_c of that class:

$$\hat{a}_{c,i} = \frac{1}{|S_c|} \sum_{x \in S_c} encoder_k(x)_i$$

   We then rank these feature dimensions by their mean activation and select the indices of the top-25, forming a set K_c.

2. **Create Sharpened Sparse Code:** For a given input example x belonging to class c, we compute its original sparse code, $encoder_k(x)$. We then create a sharpened sparse code, s'_k, by keeping only the activations corresponding to the top-feature indices K_c and zeroing out all others:

$$(s'_k)_i = \begin{cases} encoder_k(x)_i & i \in K_c \\ 0 & i \notin K_c \end{cases}$$

3. **Reconstruct the Final Target Vector:** Finally, we reconstruct a vector from this sharpened sparse code, s'_k, using the SAE's decoder, $decoder_k$. This yields the final sharpened target vector, f'_k, which has the same dimensions as the model's activation vector x:

$$(f'_k) = decoder_k(s'_k)$$

We use sharpened sparse targets to concentrate the interpretability signal on a small set of class-relevant features. This reduces the burden on the model's capacity, which in turn fosters better feature separation and more monosemantic representations (Figure 1; see Appendix A for a quantitative analysis).

3.3 The Iterative SAE-ception Process

SAE-ception is an iterative process that progressively refines a model's representations. The process is initiated by training a baseline model, $model_0$, using only the standard task loss, L_{task}. $model_0$ is then used to kick off the first full cycle of the core iterative algorithm.

For each cycle ($k = 0, 1, 2, ...$):

1. **Train SAE:** Freeze the weights of $model_k$. Generate activations from this model for the entire training set. Use these activations to train a new sparse autoencoder, SAE_k.
2. **Generate Targets:** Use the trained SAE_k to compute the set of all sharpened target vectors, $\{f'_k\}$, for the training set as described in Section 3.2.
3. **Train Next Model:** Initialize a new model, $model_{k+1}$, with the weights of $model_k$. Fine-tune $model_{k+1}$ using the combined loss. Crucially, the auxiliary loss is a function of the new model's activations, x_{k+1}, and the fixed targets from the current cycle, f'_k:

$$L_{total} = L_{task} + \lambda_s L_{aux}(x_{k+1}, f'_k)$$

For example, consider the first cycle ($k = 0$). We begin with the baseline, $model_0$. First, $model_0$ is frozen and its activations are used to train SAE_0. Second, a fixed set of targets, $\{f'_0\}$, is generated by

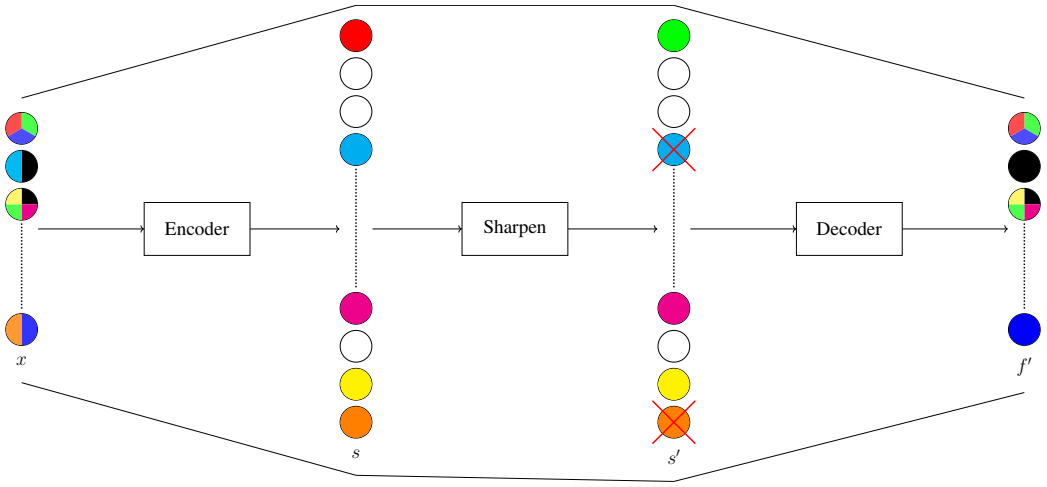

Figure 1: **An illustration of the Feature Sharpening pipeline.** The process begins with the extracted polysemantic activation vector, x, where mixed colors in each node represent multiple features. Our SAE encoder maps x to a sparse vector s, composed of largely monosemantic features (single colors) and inactive features (white). A **Sharpen** step then prunes less active features to create a sharpened vector s' (e.g., the cyan and orange features are removed). This sharpened vector is passed to the SAE decoder, which generates the final target, f'. The resulting reconstruction f' shows reduced polysemanticity, demonstrating the effect of the sharpening process.

SAE_0. Finally, $model_1$ is trained (initialized from $model_0$) with a loss that encourages its activations, x_1, to align with the targets f'_0. The resulting $model_1$ then becomes the input for the next cycle where $k = 1$ (Figure 2).

This cyclical process allows the model and the SAE to progressively shape the feature space, ideally leading to representations that are both performant and more interpretable.

4 Experimental Setup

To evaluate our approach across a spectrum of model complexities and task difficulties, our experimental design comprises three distinct settings. Each was chosen to systematically validate a key aspect of our method:

- **MLP on MNIST:** A simple, fully-connected network was used to establish core functionality.
- **ViT-H on CIFAR-10:** A high-performance Vision Transformer (ViT-H/14) was used to assess our method on modern, over-parameterized architectures. The model was pre-trained on ImageNet-21k and fine-tuned to replicate state-of-the-art accuracy on CIFAR-10 [8].
- **ConvNeXt-V2 on ImageNet-1k:** A large, powerful convolutional architecture[1] was used to test the practicality and scalability of our approach for real-world applications.

4.1 Implementation Details

Targeted Layers. For the MLP, which consists of two 16-dimensional hidden layers, we applied SAE-ception to the first hidden layer to study its effect on early feature formation.

For the ViT-H, we targeted the final output representation of the [CLS] token. This was accomplished by placing a forward hook on the main Encoder module and extracting the state of the first token (index 0) from its output sequence. This vector represents the aggregated global features of the input image just before being passed to the classifier head, making it a rich target for analyzing the model's high-level semantic understanding.

[1]We use the `convnextv2_large.fcmae_ft_in22k_in1k_384` checkpoint from the Timm library.

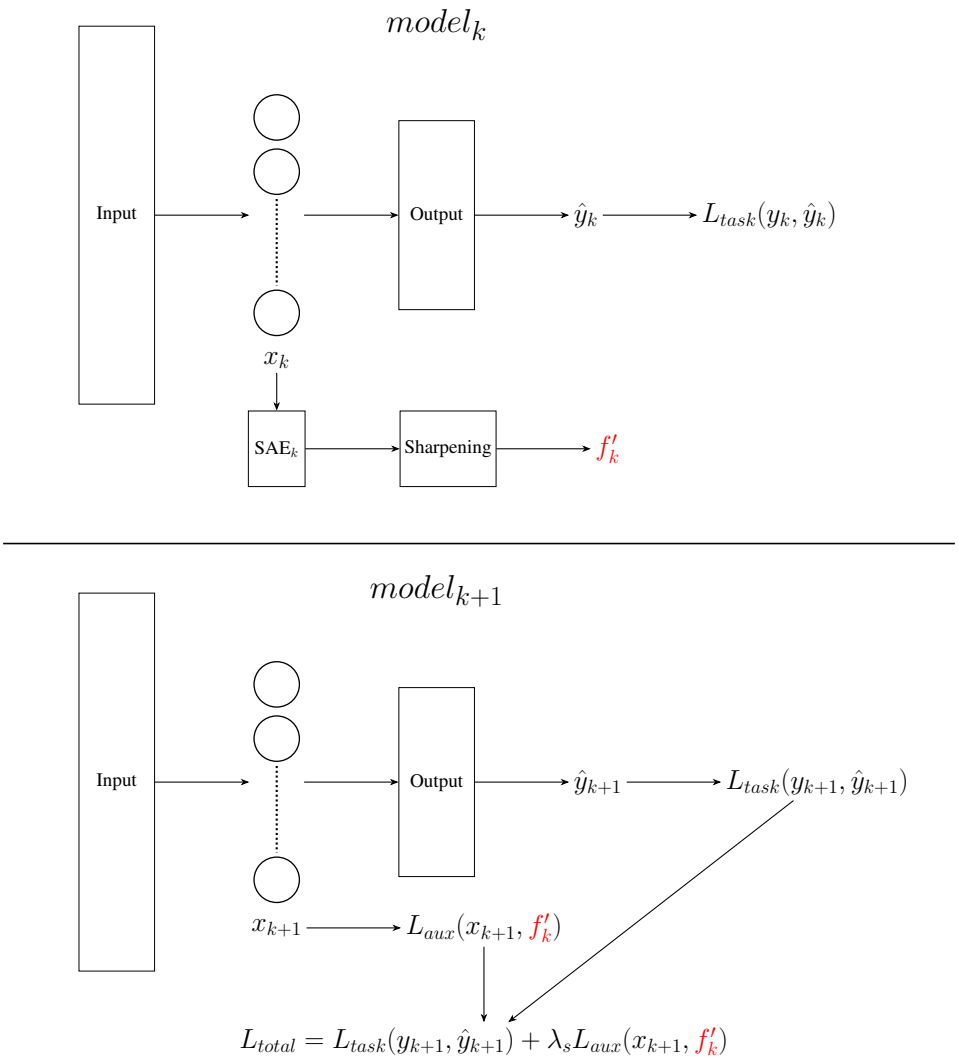

Figure 2: **The SAE-ception Training Loop. Top:** First, the frozen model, $model_k$, is used to generate a sharpened reconstructed target, f'_k, from an internal activation x_k. **Bottom:** The new model, $model_{k+1}$, is then trained. A total loss, L_{total}, is computed by combining the standard task loss, L_{task}, with an auxiliary loss, L_{aux}, that steers the new activation, x_{k+1}, to match the sharpened target f'_k. This total loss is used to update the weights of $model_{k+1}$ during backpropagation.

For the ConvNeXt-V2 model, we targeted the output of the Flatten layer within the classifier head. This layer produces the final 1536-dimensional feature vector prior to classification, making it a prime location for high-level semantic features.

Training Procedure. We employ two training procedures tailored to our different experimental settings.

For the smaller MLP model trained from scratch, we used a concurrent approach. In this setup, the SAE and base model are updated simultaneously. Crucially, this deviates from our core fine-tuning algorithm: instead of initializing the weights of $model_{k+1}$ with the weights of $model_k$, each cycle's model is retrained from the same random initialization used for $model_0$. This allowed us to explore how SAE-ception might be used throughout a full training run, instead of just fine-tuning.

In contrast, for the larger ViT-H and ConvNeXt-V2 models, we utilized a decoupled approach that follows the exact iterative fine-tuning process described in Section 3.3. Each cycle consists of fine-tuning the base model for 1-3 epochs. The model is then frozen and its static activations are used

to train an SAE for 1-5 epochs. This decoupled process ensures that the SAE is trained on a stable, high-quality representation before being used to guide the next fine-tuning step.

SAE Architecture. Our SAE implementation closely follows the architecture and loss function described in Towards Monosemanticity [3]. Sparsity levels for each experiment are reported in Table 1. For the large-scale ConvNeXt-V2 experiment, SAE linear probes were trained on a representative 30% sample of the training set for efficiency.

Table 1: **Mean $L0$ norm for SAEs across experiments.**

Cycle (k)	MLP on MNIST	ViT-H on CIFAR-10	ConvNeXt-V2 on ImageNet-1k
0 (Baseline)	32.45	45.57	57.75
1	38.29	48.64	90.93
2	39.54	46.08	82.33
3	43.12	57.86	46.69

Further details on our SAE hyperparameters and overall SAE-ception configuration are available in Appendix B.

4.2 Evaluation Metrics

The effectiveness of our technique is assessed through a comprehensive set of metrics targeting both model performance and feature representation interpretability.

4.2.1 Interpretability

We evaluate interpretability across two distinct representation spaces: the model's internal layer activations and the sparse codes generated by the SAEs. Monosemanticity is measured in both spaces, while feature separation (clustering) is examined only in the sparse code space.

Monosemanticity Evaluation. We use two complementary metrics to gauge monosemanticity.

To measure the monosemanticity of the model's dense layer activations, we apply the Class Selectivity Index (CSI). The CSI for a single feature is calculated as the difference between its average activation for its maximally activating class and its average activation across all other classes, normalized by their sum [13, 19]. A high CSI value (approaching 1) indicates that a feature's activity is strongly concentrated on a single class, serving as a proxy for monosemanticity. We report the average CSI across all neurons in the targeted layer.

While the mean-based CSI effectively measures the focus of a feature's signal, its reliance on averages can be a limitation when analyzing the highly sparse features learned by our SAE. To address this, we utilize a complementary metric based on mutual information, the uncertainty coefficient (U) [25]. This metric is designed to quantify a feature's informativeness by assessing the predictive power of its firing events, rather than its average activation intensity. It operates by first thresholding activations to create a binary signal of firing events, then calculating a normalized measure of mutual information between this signal and the class labels. This allows us to identify sparse features that are reliably predictive, even if they fire too infrequently to register a high mean-based CSI-score.

This dual-metric approach allows us to simultaneously examine the monosemanticity of concepts within the model's dense activations (via CSI) and the informational clarity of the features extracted by our SAE (via U).

Feature Clustering Evaluation. A desirable property of a learned representation is that features corresponding to distinct classes form coherent and well-separated clusters. We assess "feature clarity" by applying a suite of standard clustering metrics to the sparse feature activations generated by the SAEs at each training cycle k. These metrics assess different aspects of intra-cluster cohesion and inter-cluster separation, providing a holistic view of the representation's geometric structure.

The chosen metrics are:

- **Adjusted Rand Index (ARI):** Measures agreement between clustering and ground-truth labels, corrected for chance; ranges [-1, 1] with higher values indicating better recovery of true classes [12].

- **Silhouette Unsupervised (Sil-U):** Mean silhouette coefficient computed from cluster assignments; compares intra-cluster cohesion to nearest-cluster separation (range [-1, 1]) [22].
- **Silhouette Supervised (Sil-S):** Same silhouette statistic but computed using true class labels as the cluster assignment, measuring how compact and well-separated classes are in feature space [22].
- **Davies-Bouldin Index (DBI):** Average, per-cluster ratio of within-cluster scatter to between-cluster separation; lower values indicate more compact, well-separated clusters [7].
- **Calinski-Harabasz Index (CHI):** Ratio of between-cluster dispersion to within-cluster dispersion (adjusted by counts); larger values indicate clearer, better-defined clustering [4].

Higher values for ARI, both silhouette scores, and CHI, and lower values for DBI, are indicative of optimal feature clustering.

4.2.2 Performance

In addition to interpretability, we monitor performance to ensure our method does not compromise the model's primary function and that the learned SAE features remain task-relevant. First, we track the downstream task performance of the full model at each cycle of SAE-ception to monitor for any degradation. Second, we assess the utility of the extracted sparse features directly via linear probing. By training a linear classifier on these features alone, we can quantify their predictive power. Strong linear probe accuracy demonstrates that the SAE is learning a high-quality, informative representation of the data.

To validate the efficacy of our feature sharpening technique, we also conducted a preliminary ablation study on the MNIST dataset. In this ablation, we compared our standard approach against a baseline that used an equivalent number of randomly generated noise vectors for the auxiliary target f'. A detailed comparison is available in Appendix C.

5 Results

In this section, we present the empirical results of our method, SAE-ception, across the three experimental setups: an MLP on MNIST, a ViT-H on CIFAR-10, and a ConvNeXt-V2 on ImageNet-1k. The results are organized by metric type across three tables: Table 2 reports our monosemanticity evaluation, Table 3 details the feature clustering scores, and Table 4 presents the final performance metrics. The initial row ($k = 0$) in each table corresponds to the baseline model before any modifications, while subsequent rows show the progressive impact after each iterative cycle of SAE-ception.

Notably, our ablation study confirmed the importance of feature sharpening; it outperformed random feature targets on performance and clustering metrics, although random targets intriguingly produced slightly higher monosemanticity scores (see Appendix C).

Table 2: **SAE-ception feature monosemanticity results.**

	MLP on MNIST		**ViT-H on CIFAR-10**		**ConvNeXt-V2 on ImageNet-1k**	
Cycle (k)	**CSI ↑**	**U ↑**	**CSI ↑**	**U ↑**	**CSI ↑**	**U ↑**
0 (Baseline)	0.28	0.09	0.68	0.05	0.70	0.28
1	0.27	0.10	0.69	0.10	0.70	0.28
2	0.34	0.15	0.69	0.09	0.69	0.31
3	0.31	0.11	0.68	0.08	0.69	0.32

6 Discussion

Our results establish SAE-ception as a reliable method for improving the structural quality of a model's learned representations. Across all three diverse settings — an MLP, a ViT, and a ConvNeXt

Table 3: **Feature clustering metrics across SAE-ception cycles.**

Experiment	Cycle (k)	ARI ↑	Sil-U ↑	Sil-S ↑	DBI ↓	CHI ↑
MLP on MNIST	0 (Baseline)	0.26	-0.06	0.26	1.37	1818.76
	1	0.37	0.05	0.27	1.30	1667.21
	2	0.43	0.11	0.29	1.32	1751.87
	3	0.28	-0.02	0.26	1.40	1557.65
ViT-H on CIFAR-10	0 (Baseline)	0.81	0.30	0.49	1.19	2966.10
	1	0.90	0.39	0.56	1.00	3854.31
	2	0.86	0.32	0.59	0.93	4663.28
	3	0.81	0.30	0.55	0.98	3809.12
ConvNeXt-V2 on ImageNet-1k	0 (Baseline)	0.71	0.34	0.40	1.73	109.59
	1	0.73	0.38	0.46	1.47	152.13
	2	0.73	0.40	0.47	1.44	164.93
	3	0.74	0.46	0.49	1.36	191.95

Table 4: **Performance metrics across SAE-ception cycles.** Task accuracy (Acc.) and SAE probe accuracy (Probe) are shown for each experiment.

	MLP on MNIST		ViT-H on CIFAR-10		ConvNeXt-V2 on ImageNet-1k	
Cycle (k)	Acc. (%) ↑	Probe (%) ↑	Acc. (%) ↑	Probe (%) ↑	Acc. (%) ↑	Probe (%) ↑
0 (Baseline)	91.11	91.67	99.56	99.53	87.89	86.96
1	92.41	92.92	99.47	99.45	87.96	87.44
2	93.20	93.36	98.80	98.95	87.71	87.47
3	93.66	93.89	97.62	97.77	87.39	86.75

— we observed immediate and substantial gains in feature clustering and separability (Table 3). On ViT-H, for instance, a single cycle increased the Adjusted Rand Index (ARI) from 0.81 to 0.90, while the Davies-Bouldin Index (DBI) improved from 1.19 to 1.00. This consistent success demonstrates that SAE-ception is highly effective at reorganizing a model's activation space into a more structured format that is more amenable to post-hoc analysis.

We observe an "optimal cycle" phenomenon, particularly for feature clustering. Performance on these metrics often peaked after one or two cycles before declining, as seen in both the MNIST and CIFAR-10 results (Table 3). This suggests a practical guideline for implementation: one or two cycles are often sufficient to achieve the most significant benefits before the representational constraints begin to discard useful information.

A key advantage of SAE-ception is its minimal impact on task performance, making it a low-cost intervention (Table 4). On the highly optimized ViT-H and ConvNeXt-V2 models, the first cycle of SAE-ception resulted in negligible accuracy changes (a 0.09% drop and 0.07% gain, respectively). On the simpler MLP, the method provided a remarkable 2.55% absolute accuracy improvement over three cycles. This demonstrates that the significant gains in feature structure and interpretability — such as doubling the SAE's U score on ViT-H — can be achieved without sacrificing, and sometimes even improving, downstream performance.

While SAE-ception consistently improves feature structure, its ability to enhance monosemanticity appears highly context-dependent. This complexity is highlighted by our preliminary MNIST ablation. While applying our auxiliary loss with randomly generated noise vectors offered no benefit to performance or clustering, it unexpectedly yielded higher monosemanticity scores than our targeted sharpening approach (Appendix C). This suggests a potential trade-off: a structured, task-relevant signal (from feature sharpening) excels at improving task-aligned representations, while a less-directed, diversifying pressure may be more effective at forcing pure disentanglement.

This sensitivity to the nature of the guiding signal is likely compounded by the properties of the target layer itself. We hypothesize this discrepancy is linked to the targeted layer's position within the network. For the MLP, we target an early hidden layer where abstract features are still being formed. In contrast, for ViT-H and ConvNeXt-V2, we target the final layer before the classification head. These late-stage representations are likely already highly structured and optimized for the

task, making them more "rigid" and resistant to reorganization from the auxiliary signal. This insight suggests that SAE-ception's disentangling pressure may be most effective on less-constrained, intermediate representations, pointing to a promising direction for future work in applying the method to earlier layers in deeper architectures.

From a practical standpoint, SAE-ception is easy to implement, requiring no architectural changes. The additional computational overhead is modest; the auxiliary SAEs provide a powerful guiding signal after just a few epochs of training, and the retraining cycles are comparable to standard fine-tuning. This efficiency, combined with its demonstrated ability to improve feature structure and, in some cases, model performance, makes SAE-ception a valuable tool.

7 Limitations

The primary limitation of this study stems from computational constraints. Our experiments were conducted primarily on a single NVIDIA RTX 4090 GPU, which while sufficient for demonstrating the viability of SAE-ception, restricted our ability to perform exhaustive hyperparameter optimization.

Key parameters — such as the SAE's internal configuration, the weighting of the auxiliary loss, the optimal target layers for intervention, and the ideal number of SAE-ception cycles — would all benefit from a more extensive search. This constraint was particularly pronounced for the large-scale ViT-H and ConvNeXt-V2 models, where the presented results likely represent a conservative estimate of our method's full potential.

8 Future Work

Our findings open up several promising avenues for future research, both in refining the core method and in broadening its applications.

Methodological Refinements. Several opportunities exist to improve the SAE-ception procedure itself. First, the feature sharpening mechanism, which currently selects a fixed set of top-25 features, could be made more adaptive to determine the optimal setting (building on K-sparse autoencoders [17]). Future work might explore dynamic thresholding or learnable selection mechanisms to identify the most salient sparse features for auxiliary loss. Second, the reliability of feature extraction could be enhanced. Inspired by recent work, replacing the single SAE with more robust architectures — such as transcoders, cross-coders, or an ensemble of SAEs — could lead to more stable and meaningful feature representations, potentially accelerating the interpretability gains per cycle [2, 9, 15, 18].

Broader Applications. An important direction is to validate the cross-domain applicability of SAE-ception. While this paper focused on image classification models, the principles of iterative feature extraction and sharpening are model-agnostic. Applying our method to large language models, such as those in the GPT family, would be a valuable next step to determine if similar gains in the monosemanticity of internal neuron activations can be achieved in the language domain.

9 Conclusion

We introduced SAE-ception, an iterative training method that uses a sparse autoencoder to actively refine a model's feature space. Our results show that this technique reliably improves the clustering and separability of learned features across diverse architectures, making them more suitable for post-hoc analysis with minimal trade-offs in performance. While the quest to improve a model's inherent monosemanticity remains an open challenge, our work validates a promising new direction: using interpretability tools as an active part of the training process. SAE-ception serves as a practical step towards making the internal workings of neural networks less opaque — not by merely observing them, but by shaping them to be more understandable from the outset.

References

[1] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

[2] Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning. *Advances in Neural Information Processing Systems*, 37:107286–107325, 2024.

[3] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, jun 2023. URL `https://transformer-circuits.pub/2023/monosemantic-features/index.html`. Accessed: 2025-08-18.

[4] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[5] Helena Casademunt, Caden Juang, Adam Karvonen, Samuel Marks, Senthooran Rajamanoharan, and Neel Nanda. Steering out-of-distribution generalization with concept ablation fine-tuning. *arXiv preprint arXiv:2507.16795*, 2025.

[6] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[7] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 2009.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. *Advances in Neural Information Processing Systems*, 37:24375–24410, 2024.

[10] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[11] 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. *arXiv preprint arXiv:2406.04093*, 2024.

[12] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1): 193–218, 1985.

[13] Matthew L Leavitt and Ari S Morcos. On the relationship between class selectivity, dimensionality, and robustness. *arXiv preprint arXiv:2007.04440*, 2020.

[14] Victor Lecomte, Kushal Thaman, Rylan Schaeffer, Naomi Bashkansky, Trevor Chow, and Sanmi Koyejo. What causes polysemanticity? an alternative origin story of mixed selectivity from incidental causes. *arXiv preprint arXiv:2312.03096*, 2023.

[15] Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. Transformer Circuits Thread, may 2024. URL `https://transformer-circuits.pub/2024/crosscoders/index.html`. Accessed: 2025-08-18.

[16] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the Biology of a Large Language Model. Transformer Circuits Thread, jan 2025. URL `https://transformer-circuits.pub/2025/attribution-graphs/biology.html`. Accessed: 2025-08-18.

[17] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[18] Luke Marks, Alasdair Paren, David Krueger, and Fazl Barez. Enhancing neural network interpretability with feature-aligned sparse autoencoders. *arXiv preprint arXiv:2411.01220*, 2024.

[19] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

[20] Jungwoo Park, Young Jin Ahn, Kee-Eung Kim, and Jaewoo Kang. Monet: Mixture of monosemantic experts for transformers. *arXiv preprint arXiv:2412.04139*, 2024.

[21] Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 ieee conference on secure and trustworthy machine learning (satml)*, pages 464–483. IEEE, 2023.

[22] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[23] Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

[24] Adly Templeton, Thomas Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Chris Citro, Emmanuel Ameisen, Andy Jones, Helen Cunningham, Nicholas L. Turner, Catherine McDougall, Michael MacDiarmid, Catherine D. Freeman, Tristan R. Sumers, Euan Rees, Joshua Batson, Adam Jermyn, Shan Carter, Christopher Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. Transformer Circuits Thread, may 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`. Accessed: 2025-08-18.

[25] Henri Theil. Statistical decomposition analysis: With applications in the social and administrative sciences. *(No Title)*, 1972.

[26] Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, Deqing Fu, and Willie Neiswanger. Resa: Transparent reasoning models via saes. *arXiv preprint arXiv:2506.09967*, 2025.

[27] Hanqi Yan, Yanzheng Xiang, Guangyi Chen, Yifei Wang, Lin Gui, and Yulan He. Encourage or inhibit monosemanticity? revisit monosemanticity from a feature decorrelation perspective. *arXiv preprint arXiv:2406.17969*, 2024.

A The Effect of Feature Sharpening on Monosemanticity

Table 5 compares the average Class Selectivity Index (CSI) of two sets of features, evaluated on the test set for each experiment. *Reconstructed (Control)* refers to the activations produced by the SAE decoder from the full, unsharpened sparse features. *Sharpened Reconstructed (Ours)* refers to the activations produced by the SAE decoder from the pruned, sharpened sparse features. Higher CSI values indicate greater monosemanticity.

Table 5: **Effect of Feature Sharpening on Reconstructed Activation Monosemanticity.**

Experiment	Cycle (k)	Reconstructed (Control) CSI ↑	Sharpened Reconstructed (Ours) CSI ↑
MLP on MNIST	0 (Baseline)	0.58	**0.60**
	1	**0.61**	0.57
	2	0.62	0.62
	3	**0.68**	0.56
ViT-H on CIFAR-10	0 (Baseline)	0.70	**0.71**
	1	0.70	**0.71**
	2	0.70	**0.71**
	3	0.69	**0.71**
ConvNeXt-V2 on ImageNet	0 (Baseline)	0.73	**0.79**
	1	0.73	**0.77**
	2	0.73	**0.74**
	3	0.73	0.73

The efficacy of feature sharpening appears correlated with model complexity, providing consistent CSI improvements for the ViT-H and ConvNeXt-V2 models but not for the simpler MLP. We hypothesize that sharpening is most valuable as a technique for refining the complex, high-dimensional representations, which is less prevalent in smaller models with an already-lean feature set.

B SAE-ception Training Details

B.1 Base Model Training/Fine-tuning details

Key hyperparameters for the base model training and fine-tuning stages are summarized in Table 6. For all experiments, we utilized an Adam optimizer. The strength of the auxiliary loss, λ_s, applied during each SAE-ception cycle is detailed separately in Table 7.

Table 6: Base model training and fine-tuning hyperparameters for each experiment.

Hyperparameter	MLP on MNIST	ViT-H on CIFAR-10	ConvNeXt-V2 on ImageNet-1k
Optimizer	Adam	Adam	Adam
Learning Rate	PyTorch Default	1×10^{-4}	1×10^{-4}
Weight Decay	0	0.1	0.5
Epochs per Cycle	20	3	1
Batch Size	64	8	64
Grad. Accum. Steps	–	8	–
Effective Batch Size	64	64	64

Table 7: **Sharpened Features auxiliary loss hyperparameter λ_s values.**

Cycle (k)	MLP on MNIST	ViT-H on CIFAR-10	ConvNeXt-V2 on ImageNet-1k
0 (Baseline)	N/A	N/A	N/A
1	0.14	0.01	0.01
2	0.06	0.01	0.50
3	0.18	0.30	0.50

B.2 SAE Details

Across all experiments, we utilized a shallow SAE architecture with a ReLU activation. The hyperparameters for training the SAEs are provided in Table 8.

Table 8: Sparse Autoencoder (SAE) training hyperparameters for each experiment.

Hyperparameter	MLP on MNIST	ViT-H on CIFAR-10	ConvNeXt-V2 on ImageNet-1k
Sparse Dimension	256	5120	6144
Optimizer	Adam	Adam	Adam
Learning Rate	PyTorch Default	1×10^{-4}	1×10^{-4}
Batch Size	64	64	128
Training Epochs	20	5	1
L1 Penalty			
Cycle 0	0.75	2×10^{-4}	0.5×10^{-4}
Cycle 1	0.75	2×10^{-4}	2×10^{-4}
Cycle 2	0.75	2×10^{-4}	1×10^{-4}
Cycle 3	0.75	2×10^{-4}	7×10^{-5}

B.3 Evaluation Details.

To evaluate the predictive quality of the learned SAE features, we trained linear probes on the frozen sparse codes. For the MNIST experiment, we used Scikit-learn's Logistic Regression classifier. For the more complex CIFAR-10 and ImageNet-1k experiments, we trained a single linear layer using PyTorch. The specific hyperparameters for each probe are detailed in Table 9.

Table 9: Linear probe hyperparameters for each experiment.

Hyperparameter	MLP on MNIST	ViT-H on CIFAR-10	ConvNeXt-V2 on ImageNet-1k
Probe Model	Logistic Regression	PyTorch `nn.Linear`	PyTorch `nn.Linear`
Solver / Optimizer	L-BFGS (default)	Adam	Adam
Learning Rate	N/A	1×10^{-3}	1×10^{-3}
Penalty / Loss	L2	Cross-Entropy	Cross-Entropy
Epochs / Max Iterations	1000 (max_iter)	10	1

B.4 Computational Details.

All experiments were conducted on a single NVIDIA RTX 4090 GPU, except for replicating the state-of-the-art performance from ViT-H/14 on CIFAR-10. For the additional fine-tuning to achieve SOTA performance, we utilized a GH200.

C Preliminary Ablation Studies

To isolate the contribution of our feature sharpening technique, we conducted an ablation study on the MLP on MNIST experiment. We compared our standard method against a control condition where the auxiliary target vector, f', was a randomly generated noise vector, unit-normalized and with the same dimensionality as the model's activations. Crucially, this target had no connection to the input data or the SAE's learned feature space, serving as a pure, unstructured directional guide for the auxiliary loss.

The results, shown in Table 10, are revealing. As expected, our feature sharpening approach provides a modest but consistent edge in task performance and most clustering metrics. This suggests that using meaningful, class-relevant targets is beneficial for learning a well-structured, task-aligned representation.

The surprising result is in the monosemanticity scores. The random noise target baseline achieved slightly higher scores (particularly for CSI). This suggests that applying an auxiliary loss that pushes activations towards any sufficiently diverse set of target directions — even unstructured noise — can

encourage feature disentanglement by penalizing representational collapse. However, it is the targeted guidance of feature sharpening that aligns this disentanglement with task-relevant outcomes.

Table 10: **Ablation results on MNIST after one cycle ($k = 1$).** Feature Sharpening vs. Random Feature Targets.

Metric	Feature Sharpening	Random Targets
Performance		
Acc. (%) ↑	**92.41**	92.39
Probe (%) ↑	**92.92**	92.90
Monosemanticity		
CSI ↑	0.27	**0.33**
U ↑	0.10	**0.11**
Clustering		
ARI ↑	**0.37**	0.35
Sil-U ↑	0.05	0.05
Sil-S ↑	**0.27**	0.26
DBI ↓	**1.30**	1.37
CHI ↑	**1667.21**	1586.23