Binary Autoencoder for Mechanistic Interpretability of Large Language Models

Anonymous ACL submission

Abstract

Existing works are dedicated to untangling atomized numerical components (aka. *features*) from the hidden states of **L**arge **L**anguage **M**odels (LLMs). However, these methods largely rely on autoencoders constrained by training-time regularization computed on **single** training instances, without an explicit guarantee of **global** sparsity across instances, leading to many high-frequency activated (simultaneously inactive) features, reducing feature sparsity and atomization. In this paper, we propose a novel autoencoder variant that enforces minimal entropy on **minibatches** of hidden activations, thereby promoting feature independence and sparsity across instances. For efficient entropy calculation, we discretize the hidden activations to 1-bit via a step function and apply gradient estimation to enable backpropagation, so that we term it as **B**inary **A**uto**e**ncoder (BAE) and empirically demonstrate two major applications: **(1) Feature set entropy calculation**. Entropy can be reliably estimated on binary hidden activations, which can be leveraged to characterize the inference dynamics of LLMs. **(2) Feature untangling**. BAE avoids dense features while producing the largest number of interpretable ones among typical SAE baselines.

1 Introduction

Current practice for untangling atomized numerical components (*features*) from hidden states of **L**arge **L**anguage **M**odels (LLMs), such as **S**parse **A**uto**e**ncoder (SAE) (Shu et al., 2025), applies training-time regularization (e.g., L_1 normalization on hidden activations) to atomize features **sample-wisely**. However, such approaches do not ensure **global** sparsity, leading to frequently activated (dense) features alongside inactive (dead) features (Stolfo et al., 2025; Rajamanoharan et al., 2024b; Sun et al., 2025), contradicting the sparsity assumption (Elhage et al., 2022) of LLM hidden states. Also, consistent activations of a feature across samples hinder meaningful interpretation and reduce parameter efficiency by dead features.

Therefore, in this paper, we propose **B**inary **A**uto**e**ncoder (BAE), utilizing information-theoretic constraints among **minibatches** of training instances to address the aforementioned issue. As shown in Fig. 1, we design training objectives that minimize the entropy of hidden activation (i.e., the vector after the first layer of the autoencoder) on minibatches to reduce feature covariation and enforce global sparsity to suppress dense features. However, typical hidden activations are real vectors, whose entropies are extremely difficult to calculate (Greenewald et al., 2023) as the computation involves an exponential explosion of computational complexity. To this end, we round the activations to binary, then calculate entropy on such binary vectors to significantly reduce the computational complexity, and utilize gradient estimation (Hubara et al., 2016) to enable the backpropagation on such a rounding operation.

We empirically demonstrated the benefits of BAE as: **(1) Efficient estimation of entropy from hidden states.** We can estimate the entropy for reconstructing a set of hidden states as the hidden activation entropy with significantly reduced cost on binary vectors. By experiment on synthetic datasets with known ground-truth entropy values, we confirm the accuracy of such entropy estimation. Moreover, we utilize entropy to track the feed-forward process of standard language modeling in LLMs, revealing an information bandwidth at each layer and implicit context windows. Also, we further confirm **I**n-**c**ontext **L**earning (ICL) (Dong et al., 2022) as a form of information reduction. And **(2) Sparse feature extraction.** As also applied to normal SAE, the row vectors of the second autoencoder layer (also termed as *dictionary*) can serve as the atomized features which are essential to reconstruct the inputs. Compared to typical SAE,

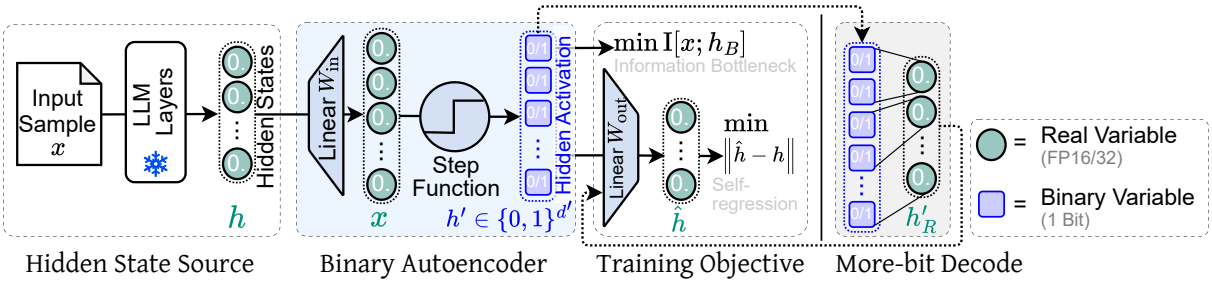

Figure 1: Feed-forward computation and training objective of BAE. Hidden states h from LLM layers are mapped by W_{in}, binarized into h' via a step function, and projected back by W_{out} as $\hat{h}$. The $\hat{h}$ is fed to the self-regression loss, while h' is fed to the information bottleneck loss. **More-bit decode**: to reduce information distortion, we aggregate multiple bits into higher-precision elements h'_R (§5) and decode on them.

BAE significantly suppresses dense features as well as dead features by the entropy-based training objective on mini-batches and consistent activation scaling based on the information gain of each channel, extracting the largest number of active and interpretable features. **(3) More-bit decoding.** Moreover, to reduce the information distortion induced by the rounding operation, we adopt a more-bit decoding as a handle for the trade-off between feature atomization and information distortion.

Main findings and contributions: (1, §2) We propose **B**inary **A**uto**e**ncoder (BAE), a novel variant of autoencoder with binarized hidden activation and training-time entropy constraint. **(2, §3)** BAE enables accurate entropy calculation on hidden state sets, and we utilize it for analyzing LLM behavior from an information perspective. **(3, §4)** We confirm BAE as an effective atomized feature extractor with less dense features and the largest amount of features extracted. (Refer A for related works.)

2 Binary Autoencoder

Feed-forward Calculation. To address SAE's drawbacks, we propose **B**inary **A**uto**e**ncoder (BAE) shown in Fig. 1 and follows: Given an input hidden state vector $h \in \mathbb{R}^d$ from an LLM, a binary autoencoder $\mathcal{F}$ conducts following calculation for the output $\mathcal{F}(h)$ (also noted as $\hat{h} = \mathcal{F}(h)$):

$$\mathcal{F}(h) = \Gamma\left(hW_{\text{in}}\right) W_{\text{out}} + b, \tag{1}$$

where $W_{\text{in}}, W_{\text{out}}^{\top} \in \mathbb{R}^{d \times d'}$, preforming linear decomposition / composition to / from d' dimensions, $b \in \mathbb{R}^d$ is a bias term for reconstructing the hidden states anisotropism (Gao et al., 2019; Ethayarajh, 2019; Biś et al., 2021; Godey et al., 2024), which contains minor information thus should be ignored as background value, Γ is a quantization function, projecting $\mathbb{R}^{d'}$ into $\{0,1\}^{d'}$ element-wisely. Here, we use step function to binarize the hW_{in} (noted as x) into hidden activation h':

$$h' = \Gamma\left([x_1, \ldots, x_{d'}]\right) = \left[\gamma\left(x_1\right), \ldots, \gamma\left(x_{d'}\right)\right],$$
$$\gamma(x_i) = \begin{cases} 0, x_i < 0 \\ 1, x_i \geqslant 0. \end{cases} \tag{2}$$

The Γ also provides essential non-linearity for the numerical expressivity of the autoencoder (otherwise, the $W_{\text{in}}W_{\text{out}}$ will degrade into one matrix).

Training Objective. *(1) Self-regression Training Loss.* Given a minibatch of hidden states (batch size n_b) $H = \left\{h^{(1)}, h^{(2)}, \ldots, h^{(n_b)}\right\}$, we calculate the self-regression training objective with L_2 norm:

$$\mathcal{L}_r(H) = \frac{1}{n_b} \sum_{h^{(i)} \in H} \left\| h^{(i)} - \mathcal{F}\left(h^{(i)}\right) \right\|_2. \tag{3}$$

(2) Information Bottleneck (Entropy) Loss. To constrain the hidden activation ($h' = \Gamma\left(hW_{\text{in}}\right)$) to a global sparse representation for h, we minimize the margin entropy of h' as shown in Eq. 4. Also, to maximize the effectiveness of the constraint on margin entropy, we also penalize the covariance (except the diagonal elements) of h' to force the margin entropy to approach the joint distribution entropy (since $h' \in \{0,1\}^{d'}$, such calculations are differentiable and efficient, without any numerical integration on the real space). That is, on the mini-batch H, we define the entropy-based loss term:

$$\mathcal{L}_e(H) = \alpha_e \mathbb{H}[\bar{h}'] + \alpha_c \mathbb{D}[H'], \text{where:}$$
$$\bar{h}' = \frac{1}{n_b} \sum_{h^{(i)} \in H} \Gamma\left(h^{(i)} W_{\text{in}}\right), H' = \Gamma\left(HW_{\text{in}}\right),$$
$$\mathbb{H}[\bar{h}'] = -\sum_{i=1}^{d'} \bar{h}'_i \log_2 \bar{h}'_i,$$
$$\mathbb{D}[H'] = \sum_{i,j:i \neq j} \left|\mathrm{cov}(H')_{i,j}\right|, \text{where:} \tag{4}$$

α_e and α_c are hyperparameters. **The total loss is:**

$$\mathcal{L}(H) = \mathcal{L}_r(H) + \mathcal{L}_e(H). \quad (5)$$

This training objective resembles and simulates the information bottleneck approach (Tishby et al., 2000; Kawaguchi et al., 2023; Tishby and Zaslavsky, 2015), where $\mathcal{L}_e$ minimizes the mutual information between input h and latent h', while $\mathcal{L}_r$ maximizes that between output and h'.

Gradient Estimation for Γ. Since the differential of binarization function Γ is a Dirac delta function (i.e., 0 almost everywhere), to enable effective back-propagation from h' to W_{in}, following the previous works (Hubara et al., 2016; Vargas et al., 2024), we estimate the gradient of Γ by smoothing function $x \mapsto (1 + e^{-x})^{-1}$ ("Sigmoid") elementwisely:

$$\frac{\partial \Gamma(x)}{\partial x} := \text{Sigmoid}(x) \odot (\mathbf{1} - \text{Sigmoid}(x)), \quad (6)$$

where $\mathbf{1}$ is all-ones vector, $\odot$ is Hadamard product.

Default Settings. We defaultly set $d' = 4d$ (i.e., the expanding rate $= 4$), the entropy loss weight[1] $\alpha_e = 10^{-7}$, the covarience loss weight $\alpha_c = 10^{-7}$. We use Adam (Kingma and Ba, 2014) optimizer with learning rate 5×10^{-4}, momentum $\alpha_1 = 0.9$, $\alpha_2 = 0.999$, and minibatch size $n_b = 512$ for 2000 epochs, with $\alpha_e = 0$ in the first 500 epochs.

3 Entropy Estimation of Hidden States

Entropy Estimation by BAE. As a direct measurement of information amount, the entropy of neural network hidden states can promote a closer observation into their mechanism. However, directly computing the differential entropy of a high-dimensional vector set requires probability **d**ensity **e**stimation (DE) and integration, which is neither accurate (DE suffers from the curse of dimensionality and floating-point errors) nor efficient (numerical integration costs $\mathrm{O}(C^{d'})$ for d' dimensions with C integration cells) (Greenewald et al., 2023). While, BAE improves such entropy calculation by best-effortly decomposing the original real-value vectors into binary vectors (h') with pair-wisely independent channels, enabling efficient entropy estimation by margin entropy of the mean hidden activation $\bar{h}'$: given a vector set $H = \{h^{(i)}\}_{i=1}^{n}$, we encode them into $h'^{(i)}$s by a trained BAE as $H' = \{h'^{(i)} = \Gamma\left(h^{(i)} W_{\text{in}}\right)\}_{i=1}^{n}$. Since the h's are binarized and best-effortly pairwise-decorrelated[2] by $\mathcal{L}_e$, we can calculate the *entropy required for reconstructing* H as $\mathbb{H}[\bar{h}']$, where $\bar{h}'$ are the averages among the row vectors of H', similar as Eq. 4.

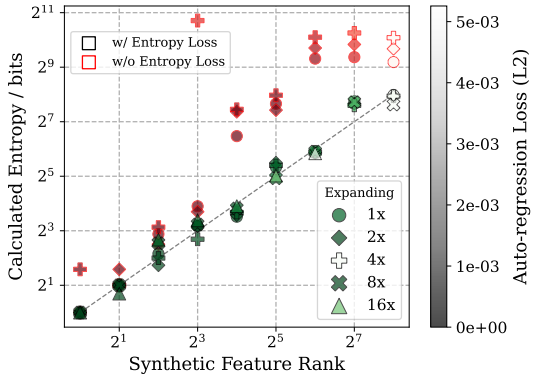

Figure 2: Evaluation of BAE entropy calculation on the synthetic dataset. Horizontal axis: rank r of the current dataset, vertical axis: calculated entropy, the green/red color refers to whether $\mathcal{L}_e$ is enabled, and the opacity refers to the $\mathcal{L}_r$ on the whole input set.

Next, we evaluate the entropy estimation protocol on a synthesis dataset with known entropy to demonstrate its accuracy, and then use this protocol to track the feed-forward process of LLMs.

3.1 Evaluating BAE's Entropy Estimation

Synthetic Directional Benchmarking. To evaluate entropy estimation aforementioned, we build a synthetic directional benchmark: **(1)** sample a d-dimensional r-rank orthonormal basis $M \in \mathbb{R}^{r \times d}$, **(2)** sample r binary coefficients $c \in \{0, 1\}^r$, and **(3)** generate an instance cM (i.e., summing selected M basis from 1-elements in c). Repeat (2) and (3) n times, we get a synthesis random directional dataset with n samples. Intuitively, the entropy of the dataset is r since the only randomness comes from the c (i.e., r independent Bernoulli coefficients) given the basis fixed. We train the BAE on such datasets with various r (details in Appendix B.1), and confirm how the entropies calculated with the aforementioned protocol match r. Moreover, we propose a prototype to evaluate such entropy calculation on real LM hidden states in Appendix C. Also, we analyse the training dynamics of BAE, and find a natural low-entropy trend of

[1] Notice that, considering the magnitude around the late training stage (as shown in Fig. 2 and 11) of the self-regression loss ($\mathcal{L}_r$, around $10^{-3} \sim 10^{-2}$) and the margin entropy ($\mathbb{H}[\cdot]$, around $10^{0} \sim 10^{3}$), athough the weights α_e are set quite small, the magnitude of the $\alpha_e \mathbb{H}[\cdot]$ are balanced as a regularization term against $\mathcal{L}_r$.

[2] Since W_{in} conducts a linear decomposition to h sets, any suspected high-order correlations, which can not be catched by the covariance loss, can not be produced.

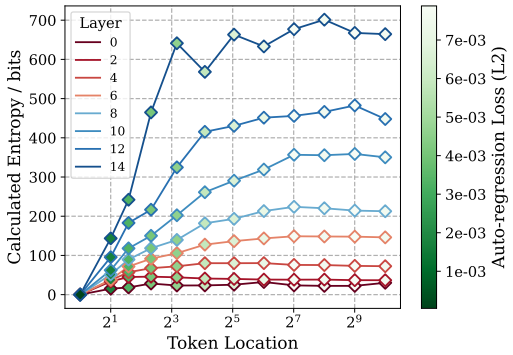

Figure 3: Entropy calculated on hidden states extracted from each layer and token location from Pile and Llama 3.2-1B. **Curve colors**: the extracted layers; **Scatter colors**: $\mathcal{L}_r$ over the whole input set.

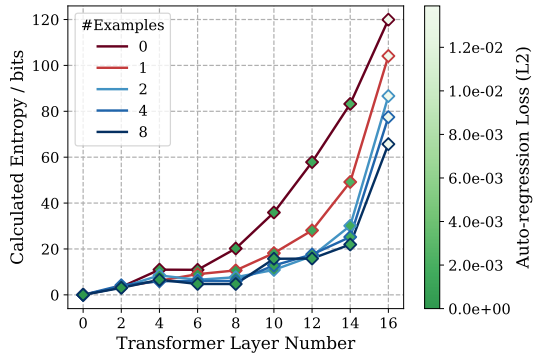

Figure 4: Entropy calculated on the hidden states extracted from specific layers and the last token from ICL inputs from SST-2. The curve colors refer to the number of demonstrations.

BAE hidden activations, refer to Appendix E.

BAE can Accurately Estimate Entropy of Synthetic Vector Set. The evaluation results are shown in Fig. 2, where the standard BAE implementation with normal entropy objective (green) hint the corresponding r accurately, despite of the inner dimension d' (noted as the expending rate d'/d, shown as the marker shape in Fig. 2). In contrast, the adversarial experiments (red), where the α_e and α_c are both set to 0, show clearly higher calculated entropy, suggesting that BAE with entropy constraint can find the essential (minimal) entropy values to reconstruct the input sets.

3.2 Tracing Language Modeling by Entropy

In this section, we track the LLM's feed-forward dynamics by the entropy of hidden states. In detail, we sample $n = 262144$ sentences from Pile (Gao et al., 2020), inputting them into Llama 3.2-1B (Grattafiori et al., 2024), then extract the hidden states of $2^{0,1,\cdots,10}$-th tokens from each layer. Then we train a BAE on each layer and position (details in Appendix B.2). By the trained BAE, we calculate the entropy by the aforementioned protocol as shown in Fig. 3, where we observe:

Layer Bandwidth. The entropy of hidden states from a specific layer increases with the number of prefix tokens and eventually saturates at a fixed value. This observation suggests that: if the hidden space of a specific layer is viewed as a channel (Elhage et al., 2021), through which token information is exchanged, then the channel has a fixed bandwidth, limiting the amount of accommodated token information. That is, Transformers have implicit context windows on each layer, where the exceeded information is discarded or distorted. Unlike convolutional neural networks, we have no evidence to infer that such context windows are contiguous or even binary, so that it is possible to overwrite parts of but not the whole of the oldest token by new token, causing distortation like the *lost-in-the-middle* problem (Hsieh et al., 2024; Liu et al., 2024a; He et al., 2024; An et al., 2024; Liu et al., 2025).

Token Information Gain. As shown in Fig. 3, for a specific token position, hidden states from a deeper layer carry more information, and the entropy also saturates later on a deeper layer, suggesting that Transformer blocks sequentially inject contextualized information into hidden states. As a result, deeper layers are more likely to simulate broader context windows, which facilitates the processing of downstream tasks at higher levels of information aggregation and abstraction. In contrast, shallower layers, restricted by narrower context windows, tend to focus on local linguistic-level features and propagate this information to the later layer to combine into broader-scoped abstraction, which is consistent with intuition and previous empirical observations (Jawahar et al., 2019; Chen et al., 2023; Wang et al., 2023; Xiao et al., 2025). We discuss this point deeper in §6.

3.3 Tracing In-context Learning by Entropy

In this section, we track the LLM's **I**n-**c**ontext **L**earning (ICL, refer Appendix B.3 for introduction) inference dynamics via entropy, similar to §3.2. In detail, we sample 262144 ICL input instances from SST-2 (Socher et al., 2013) with specific demonstration numbers, then extract the hidden states of the last token (the ":" in the "sentiment:", where the answer to the query will be generated) of a specific layer, and then train a BAE on these hidden states, with prompt templates, training settings and hyperparameters detailed in Ap-

pendix B.3. We calculate the entropy as shown in Fig. 4, where: **(1)** similar to the normal language modeling, a deeper layer contains more information, which suggests that the implicit context windows hypothesis still stands for the ICL scenario. However, **(2)** contradicts the normal language modeling, hidden states with more demonstrations (globally longer sentence lengths and better accuracy, see Appendix B.3) counterintuitively contain lower entropy, which suggests an interesting conclusion: *ICL is achieved through removal of information*, where the information useless to the specified task may be suppressed by the given demonstrations. This contradicts the mainstream idea that ICL "learns new tasks or knowledge" (Pan et al., 2023; Li et al., 2024; Wang et al., 2025), and thus may offer a new perspective for interpreting the ICL inference processing as "deleting unrelated information from the query encoding (Cho et al., 2025) on the last token".

4 Atomized Features Extraction by BAE

Given that our entropy constraint promotes sparsity and decorrelation of h', the linear decomposition from h to h' can disentangle atomized features in LLM hidden states (Elhage et al., 2022; Hänni et al., 2024). To evaluate such disentanglement, in this section, we assign human-interpretable semantics to each channel of h' and assess their consistency across inputs, using an improved automatic interpretation method under the LLM-as-a-judge framework (Gu et al., 2024). Also, we utilize BAE to extract steering vectors (Hendel et al., 2023; Kharlapenko et al., 2024) from ICL inputs and inject such vectors into 0-shot inputs to show BAE can extract more functional LLM features (Appendix H).

4.1 Feature Interpretation Revised

Revisiting Current Automatic Feature Interpretation and Evaluation. We begin with a revisiting to the current automatic interpretation and evaluation methods of features (Bills et al., 2023; Huben et al., 2024), where for a channel in h' corresponding to a feature (line vector) in W_{out}, **(Step 1)** given one input text, the activation magnitudes (e.g., the value of the specified channel in h' of SAE) on the channel are calculated on every token, then **(Step 2)** the tuple of sentence and activation magnitudes on all tokens is input in LLMs with a prompt (e.g., "Please predict the explanation of the feature given the following activations.") for interpreting such feature into one phrase (e.g., "freedom-related terms"). Then, **(Step 3)** given a test input text, a simulation of the activation magnitudes based on the generated interpretation for each token is queried from the LLM, and the correlation coefficient of the simulated activation magnitudes and the SAE-calculated activation magnitudes is regarded as the interpretability score of this feature (refer to Bills et al. (2023) for details).

Notice that Step 2 of the above process relies on the LLM's ability to accurately handle large amounts of numerical tokens to generate reliable explanatory phrases, and Step 3 requires the LLM to faithfully simulate activation magnitudes by numerical tokens, which places a high demand on the LLM's mathematical reasoning capabilities and also output calibrations. However, current research indicates that LLMs exhibit weaker capabilities in numerical reasoning compared to linguistic tasks (Press et al., 2023; Schick et al., 2023; Ahn et al., 2024; Xu et al., 2025). Also, the output can be implicitly biased to some specific tokens (Zhao et al., 2021; Geng et al., 2024), making the aforementioned pipeline unreliable, requiring a revision to improve the robustness and credibility.

Common Semantics-based Feature Interpretation and Evaluation (ComSem). Therefore, to avoid relying on LLMs to directly process numerical tokens, we propose ComSem as a new pipeline that leverages the LLMs' strength in linguistic semantic recognition to interpret the extracted features. In detail (refer Appendix B.4 for the detailed pseudocode): for a specific channel (corresponding to a feature) in the h' of an autoencoder, given a set of test sentences, **(Step 1)** for all the tokens in the set, whose hidden states have top-k (k is a hyperparameter of the method) significant activation magnitude on the specific channel over all channels (i.e., if the activation magnitude of this specific channel is in the top-k when we sort all the activation magnitudes over all channels), we collect them along with their sentences. **(Step 2)** We query the backend LLM to find the commonality of these tokens presented in their sentences, as the interpretation of the feature corresponding to the channel. **(Step 3)** On a hold-out set of such token-sentence tuples significantly activated on the channel, we query the LLM to judge whether these tokens can be interpreted by the generated interpretation from Step 2. The ratio of the "Yes" answer is calculated as the interpretability score of this

feature. ComSem can avoid the concern in mathematical and numerical reasoning, and provides the possibility of applying simple output calibrations[3] on the true-or-false output. So, we utilize ComSem to evaluate the BAE, as described below.

4.2 Transfer Binary Activation to Magnitude

Estimate Activation Magnitude from Binary Activation. Since ComSem requires access to feature activation *magnitude*, which our BAE can not explicitly provide[4], as shown in Fig. 10, we calculate the *burstiness* of each channel to convert the binary activations h' into the activation magnitudes. In detail, for each (indexed by i) instance $h'^{(i)}$ in a hidden activation set $H' = \{h'^{(i)}\}_{i=1}^{n}$, we compare the distance of $h'^{(i)}$ with a prior distribution[5] $\bar{h}' \in [0,1]^{d'}$ as $\beta^{(i)} = \log_2 |h'^{(i)} - \bar{h}'|$, where the $\log_2$ is the element-wise logarithm. Such $\beta^{(i)}$ is the channel-wise activation magnitude of $h'^{(i)}$, where a channel j with larger $\beta_j^{(i)}$ suggests the j-th feature from $h'^{(i)}$ is more **atypical**, or **bursty**. A more bursty feature reduces more uncertainty to reconstruct the $\hat{h}$ so that *carries more information against the $\bar{h}'$, which makes it more representative of h* (Appendix F for theoretical groundings). Then, we input $\beta^{(i)}$ as activation magnitudes into the ComSem framework.

Such burstiness calculation has the following advantages: (1) Global sparsity induced by low entropy. Notice that our entropy training objective shown in Eq. 4 is equivalently punishing the burstiness summed from all the input samples on all the channels among the minibatches, therefore, the calculated $\beta^{(i)}$ for each sample is numerically sparse in two directions, i.e., **(i)** for one instance $h'^{(i)}$, most of elements in $\beta^{(i)}$ are suppressed to be sufficient small values, and **(ii)** for one feature channel j, most of $\beta_j^{(i)}$ are small values among i among all instances, which is aligned to the sparsity assumption of mechanistic interpretability (refer to Appendix I.1 for visualization). **(2) Consistent scale and clear interpretability.** In typical SAEs, different channels of h' lie on varying scales, making some inherently high-magnitude channels appear consistently active ("dense features", Appendix I.1 and Fig. 5) (Stolfo et al., 2025; Rajamanoharan et al., 2024b) while others seem inactive ("dead features"). This issue is particularly pronounced when using top-k selection based on absolute activation values to identify significant features, reducing the number and introducing confusion to activated features. Therefore, intuitively, as shown in §4.3, rescaling activations into an aligned distribution alleviates this problem in typical SAE. And in our pipeline, BAE simplifies the rescaling since each channel follows a Bernoulli distribution and requires only one statistic ($\bar{h}'$), while real-valued h' in typical SAEs is harder to characterize.

[3]E.g., described in (Zhao et al., 2021; Fei et al., 2023). Such output calibration is not utilized in this work.

[4]Note that, in typical SAE, the hidden activation (congener of our h') can directly serve as such magnitude.

[5]Also named *typical set* in information theory. Such $\bar{h}'$ can either be saved during training, or mean value on H' set.

Feat. Source	Model (Appendix B.4)	$\mathcal{L}_r$ Val. Set	$\|$Cov$\|$ Norm.	Act.	ComSem$_{4.1\text{-mini}}$		ComSem$_{4.1}$	
					FI#	Score	FI#	Score
Llama 3.2-1B Layer 5 $d = 2048$	ReLU SAE	$3_{\times 10^{-3}}$	0.02	1922	1103	0.216	1251	0.294
	Top-k SAE	$2_{\times 10^{-3}}$	0.05	**3531**	**2614**	**0.439**	**2763**	**0.445**
	Gated ReLU SAE	$6_{\times 10^{-3}}$	0.05	17	11	0.382	6	0.294
	ReLU SAE Resc.	$3_{\times 10^{-3}}$	0.02	567	309	0.233	367	0.269
	TransCoder	$1_{\times 10^{-2}}$	0.00	335	218	0.275	195	0.238
	BAE (ours)	$4_{\times 10^{-3}}$	**0.37**	3012	2249	0.422	2449	0.439
Llama 3.2-1B Layer 9 $d = 2048$	ReLU SAE	$2_{\times 10^{-4}}$	0.05	1914	1128	0.229	1274	0.254
	Top-k SAE	$4_{\times 10^{-3}}$	0.05	3179	2389	0.450	2542	0.461
	Gated ReLU SAE	$1_{\times 10^{-2}}$	0.06	100	85	**0.638**	80	**0.596**
	ReLU SAE Resc.	$2_{\times 10^{-4}}$	0.05	3161	2028	0.258	2190	0.294
	TransCoder	$2_{\times 10^{-2}}$	0.00	417	257	0.260	263	0.264
	BAE (ours)	$6_{\times 10^{-3}}$	**0.21**	**4675**	**3370**	0.370	**3624**	0.394
Llama 3.2-1B Layer 11 $d = 2048$	ReLU SAE	$1_{\times 10^{-2}}$	0.02	2065	1177	0.232	1380	0.260
	Top-k SAE	$8_{\times 10^{-3}}$	0.06	3417	2540	0.440	2684	0.452
	Gated ReLU SAE	$1_{\times 10^{-2}}$	0.05	1226	976	**0.531**	1026	**0.557**
	ReLU SAE Resc.	$1_{\times 10^{-2}}$	0.08	744	435	0.259	482	0.278
	TransCoder	$2_{\times 10^{-2}}$	0.01	1794	979	0.218	1090	0.239
	BAE (ours)	$1_{\times 10^{-2}}$	**0.18**	**5464**	**3882**	0.360	**4140**	0.382
Llama 3.2-1B Layer 14 $d = 2048$	ReLU SAE	$3_{\times 10^{-5}}$	0.03	2528	1423	0.195	1600	0.217
	Top-k SAE	$3_{\times 10^{-2}}$	0.04	2702	1900	0.389	2004	0.418
	Gated ReLU SAE	$3_{\times 10^{-2}}$	**0.13**	2948	2095	**0.412**	2250	**0.435**
	ReLU SAE Resc.	$3_{\times 10^{-5}}$	0.03	3962	2361	0.252	2661	0.274
	TransCoder	$3_{\times 10^{-2}}$	0.01	3401	1931	0.237	2166	0.267
	BAE (ours)	$4_{\times 10^{-2}}$	0.06	**6120**	**3963**	0.324	**3971**	0.323
Llama 3.2-3B Layer 20 $d = 3072$	ReLU SAE	$6_{\times 10^{-3}}$	0.08	1923	1183	0.289	1289	0.312
	Top-k SAE	$5_{\times 10^{-2}}$	0.03	3234	2286	0.402	2508	0.425
	Gated ReLU SAE	$5_{\times 10^{-2}}$	**0.08**	4628	3271	**0.402**	3580	**0.437**
	ReLU SAE Resc.	$6_{\times 10^{-3}}$	0.08	2122	1222	0.250	1451	0.294
	TransCoder	0.15	0.01	5508	3001	0.01	3424	0.257
	BAE (ours)	$7_{\times 10^{-2}}$	0.07	**9308**	**5956**	0.308	**6805**	0.348

Table 1: Evaluation of BAE and baselines ($d'/d = 4$). $\mathcal{L}_r$: the self-regression loss on validation set. $\|$**Cov**$\|$: the spectral norm of the covariance matrix among normalized line vectors of W_{dec}. **Act.**: The number of features where sufficient h_0 instances (more than 8, refer to Appendix B.4) with significant activation magnitude are observed on the corresponding channel. **FI#**: The number of feature channels with non-zero ComSem interpretability score. **Score**: The averaged ComSem score among all activated channels.

4.3 Interpreting & Evaluating BAE Features

As mentioned before, we expect BAE to mitigate the two issues of typical SAE: **(1) dense features** on inherently high-magnitude channels, with low consistency among various instances with the same feature activated (i.e., low interpretability scores in Step 3 of ComSem), and **(2) dead features** on inherently low-magnitude channels, leaving many channels idle. In this section, we empirically con-

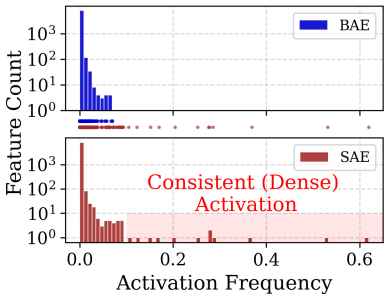

Figure 5: Feature number in each activation frequency bin of Layer 11 (more layers in Appendix I.3).

firm the mitigation of both issues by BAE against SAE baselines, all trained on Pile with ComSem on 2 backend evaluation LLMs and BookCorpus (Zhu et al., 2015) (detailed in Appendix B.4) in Table 1, where BAE can extract the largest amount of active features from the LLM's hidden states against baselines, with a considerable interpretability score. However, one can doubt that the interpretability score of BAE is "not SotA" in the Table 1, and our explanation is: BAE has a stronger feature extracting ability, so that it captures some complex features (refer to Appendix G) which are difficult to interpret by natural language from LLMs, reducing the interpretability scores. As shown in Appendix D, if some of the low-score features are removed, then BAE reaches the SotA of interpretability score and also remains the largest feature amounts. Moreover, we conduct normalization[6] to the h' of ReLU SAE for a rescaled activation magnitude to mitigate the influence of inherent activation magnitude of channels, and evaluate as "ReLU SAE Resc." in Table 1, where comparing to the vanilla ones, the interpretability scores are observably improved by the normalization rescaling. Since SAE training ignores cross-sample sparsity and its h' distribution is hard to estimate, the effectiveness of rescaling remains limited.

Information Loss by Binarization. One suspicion is that the Γ causes information loss from the h, which may harm the feature extraction. To evaluate such information loss, we calculate the self-regression loss ($\mathcal{L}_r$) on the validation set, as shown in Table 1, where no significant difference of BAE compared to baselines can be observed. This indicates that BAE at least does not incur more severe information loss than the baselines. Furthermore, we attempt to increase the decoding bits to reduce such information loss in §5.

[6] I.e., linearly scale every channel value to a distribution over h' instances with mean value 0 and standard deviation 1.

Activated Feature Distribution. To examine BAE's effective suppression towards dense feature, we visualize the activation frequency distribution of each channel on the trained BAEs and SAEs as Fig. 5. In the visualizations, BAE features are sparsely activated, showing a left-leaning distribution, while some of the features in the typical SAE keep high activation frequencies with a long-tail distribution, suggesting dense activations (Stolfo et al., 2025; Rajamanoharan et al., 2024b). Such results support our hypothesis that the minibatch-oriented entropy objective can mitigate the global density among input instances. Visualizations on more settings are provided in Appendix I.1.

Feature Diversity. To show that BAE extracts more non-redundant features, as an intuitive measurement of feature diversity, we directly compute the spectral norm of the covariance matrix over the row vectors (which is additionally normalized to norm-1 vectors) in W_{out}, with the results reported in Table 1. Among these results, BAE exhibits a large covariance in most cases, confirming its ability to extract diverse and non-redundant features.

Moreover, in Appendix H, we find that BAE can reconstruct the steering vectors (Hendel et al., 2023) better than SAE baselines, which suggests that BAE can extract more effective LLM features.

5 More-bit Decoding

As mentioned before, although the self-regression loss induced by BAE is not significant compared to the baselines, it can be beneficial to reduce such loss. Therefore, based on the framework of BAE, in this section, as shown in Fig. 1 ("More-bit Decode"), we propose a plug-in module to aggregate multiple bits of h' into real variables before decoding by the W_{out}. In detail, for each C binary elements in $h' \in \{0,1\}^{d'}$ (i.e., from the iC-th elements to the $[(i+1)C-1]$-th elements of h'), we transfer the binary bits into a real vector as:

$$h_R = \frac{\sum_{j=0}^{C-1} 2^j h'_{iC+j}}{2^C - 1}, \tag{7}$$

then, we decode h_R by $W_{\text{out}} \in \mathbb{R}^{(d'/C)\times d}$ (trained from scratch) for the output $\hat{h}$. Such a method improves the numerical precision of BAE, and is therefore expected to mitigate the information loss while still retaining a binary encoding to support entropy computation. The entropy loss is calculated on the binary encoding h', whereas feature interpretation is conducted on the real-valued h_R.

Bits (C)	$\mathcal{L}_r$	Act.	FI#	Score
(ReLU SAE)	$10.5_{\times 10^{-3}}$	2065	1177	0.232
1	$11.0_{\times 10^{-3}}$	5464	3882	0.360
2	$7.7_{\times 10^{-3}}$	4195	2951	0.371
3	$7.0_{\times 10^{-3}}$	3023	2170	0.391
4	$6.6_{\times 10^{-3}}$	2841	2029	0.376

Table 2: More-bit decoding on Llama 3.2-1B Layer 11. Notation similar to Table 1, evaluated on GPT 4.1-mini.

Layer#	α_e, α_c	Memory (MB, +Model)	MSE	Cos.Sim.
5	1e-7	16713 $\rightarrow$ 141	0.016	0.681
9	1e-7	16713 $\rightarrow$ 173	0.040	0.760
11	1e-7	16713 $\rightarrow$ 166	0.049	0.800
14	1e-7	16713 $\rightarrow$ 178	0.092	0.816
5	1e-9	16713 $\rightarrow$ 170	0.004	0.748
9	1e-9	16713 $\rightarrow$ 183	0.052	0.809
11	1e-9	16713 $\rightarrow$ 206	0.099	0.831
14	1e-9	16713 $\rightarrow$ 270	0.030	0.860

Table 3: Compression and reconstruction on Llama 3.2-1B. **Memory**: storage cost before/after compression. **MSE / Cos.Sim.**: mean square error or cosine similarity between the source and reconstruction.

Results with $d'/C = 4d$ are shown in Table 2, where more-bit decoding consistently reduces self-regression loss while decreasing the number of activation features, yet remains superior to the SAE baseline. As we will discuss in §6, BAE decomposes intrinsically continuous features into multiple atomic ones, whereas more-bit decoding can merge them into a single feature, which leads to a reduction in the total number of features. Such a result demonstrates that more-bit decoding serves as an effective trade-off handle between feature numbers and information loss. Moreover, to reduce the information loss of BAE, future work can be devoted to modifying the structure of BAE, especially by more advanced binarization functions (Wang et al., 2018; Vargas et al., 2024) and gradient estimation methods (Darabi et al., 2019; Yang et al., 2019).

6 Conclusion and Discussion

Conclusions. In this paper, we propose Binary Autoencoder as a toolkit for mechanistic interpretability research for LLMs. BAE utilizes an entropy training objective on a minibatch of binary hidden activations to extract globally sparse and atomized features. Our findings demonstrate that BAE can accurately estimate the entropy of a feature set and effectively disentangle atomized and sparse features from LLM's hidden states, making it a powerful tool for interpretability research.

Compress Hidden States into Fewer Bits by BAE. As detailed in Appendix B.5, one can utilize BAE to compress h set by storing only channel indices i of h' where the burstiness β_i against prior $\bar{h}'$ exceeds a threshold. Such compression has an expected coding length for each instance[7] $\mathrm{L}_{\texttt{int}}\mathbb{H}[h]$, outdistances the original $\mathrm{L}_{\texttt{float}}d$. Reconstruction is done by flipping $\mathrm{round}(\bar{h}')$ at the stored indices and projecting through W_{out}. We evaluate this compression with a threshold $\log_2 0.5$ (refer Appendix B.5), as shown in Table 3, achieving compression rates as low as 1% with low MSE, indicating both effective compression and reconstruction quality. Notice that, even though the cosine similarity does not present a high absolute value, given the high dimensions of the compressed and reconstructed vectors ($d = 2048$), it can not be viewed as a distortion in the radial direction. Since this method falls outside the paper's main focus, we view such compression as a prototype for reliable and efficient hidden state storage or transmission, which can be utilized in the practice of interpretability and broader research areas.

Hypothesis of Atomic Features. Since BAE represents feature activations with binary values rather than continuous ones, continuous features (such as RGB values (Engels et al., 2025; Modell et al., 2025)) may be discretized into multiple features, which is reasonable since these features cannot be represented with 1-bit, so decomposing them into multiple features facilitates the correct estimation of the entropy. On the other hand, due to the narrow radial distribution of Transformer hidden states (Gao et al., 2019; Ethayarajh, 2019), different features may share the same direction but differ in distance, which is difficult to distinguish in continuous SAEs, whereas BAE avoids this issue by discretizing them into atomic units. Also, refer to Table 1, compared to baselines, the number of discretized features extracted by BAE significantly increase against layers, with SAE features remain constant, suggesting that continuous features are more likely to emerge in later layers, which are split into more discrete features by BAE, consistent with previous works (Jawahar et al., 2019; Chen et al., 2023; Allen-Zhu and Li, 2023; Liu et al., 2024b), still needs further exploration.

[7] Here, $\mathrm{L}_{\texttt{int}}$ or $\mathrm{L}_{\texttt{float}}$ denotes the number of bits required to encode a single index for the compressed coding or one float variable for an original feature element, and $\mathbb{H}[h]$ is the entropy estimated by the BAE from the distribution of h, d is the dimensionality of h. The $\mathrm{L}_{\texttt{int}}$ is usually less than $\mathrm{L}_{\texttt{float}}$, and $\mathbb{H}[h]$ is also usually less than d (as demonstrated in Fig. 3, with $d = 2048$ and $\mathbb{H}[h] < 700$).

Limitation

(1) **Natural Language Feature Interpretation Based on Tokens.** In this paper, ComSem, and also the traditional automatic feature interpretation methods, are all based on the natural language interpretation of tokens where the features originate. However, the contextualization induced by Transformer layers may cause the semantics of the hidden states to deviate from the original token, but the method family overlooks such an effect. Future works can be devoted to directly decoding the semantics from the extracted atomized features by tools such as LogitLens (Joseph Bloom, 2024) or PatchScopes (Ghandeharioun et al., 2024). Also, not all the features can be interpreted by natural languages, such as Task Vectors (Hendel et al., 2023; Kharlapenko et al., 2024). We discuss this point in Appendix G. (2) **Investigation Scope.** Due to computational limits, we evaluate BAE only on Llama 3.2-1B and 3B, leaving large-scale tests for future work. Nevertheless, we believe this paper provides a sufficient prototype of BAE for addressing dense/dead feature issues in SAE. (3) **Manifold Features.** Prior studies (Engels et al., 2025; Modell et al., 2025) suggest that some features in LLM hidden states lie on nonlinear manifolds, making them hard to extract with BAE or other linear-decomposition SAEs. Future work may explore multilayer or manifold-based SAEs to capture such features.

Acknowledgments

Not visible during review periods.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 1, learning hierarchical language structures. *arXiv preprint arXiv:2305.13673*.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2024. Make your llm fully utilize the context. *Advances in Neural Information Processing Systems*, 37:62160–62188.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. 2020. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070.

Leonard Bereska and Stratis Gavves. 2024. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*. Survey Certification, Expert Certification.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models.

Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. 2021. Too much in common: Shifting of embeddings in transformer language models and its implications. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, Online. Association for Computational Linguistics.

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, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Bart Bussmann, Patrick Leask, and Neel Nanda. 2024. Batchtopk sparse autoencoders. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*.

Yixiong Chen, Alan Yuille, and Zongwei Zhou. 2023. Which layer is learning faster? a systematic exploration of layer-wise convergence rate for deep neural networks. In *The Eleventh International Conference on Learning Representations*.

Hakaze Cho and Naoya Inoue. 2025. Staicc: Standardized evaluation for classification task in in-context learning. *arXiv preprint arXiv:2501.15708*.

Hakaze Cho, Mariko Kato, Yoshihiro Sakai, and Naoya Inoue. 2025. Revisiting in-context learning inference circuit in large language models. In *The Thirteenth International Conference on Learning Representations*.

Ikhyun Cho and Julia Hockenmaier. 2025. Toward efficient sparse autoencoder-guided steering for improved in-context learning in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 28961–28973, Suzhou, China. Association for Computational Linguistics.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*.

Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, and Vahid Partovi Nia. 2019. BNN+: Improved binary network training.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

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

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, and 1 others. 2022. Toy models of superposition.

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, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. 2025. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Yu Fei, Yifan Hou, Zeming Chen, and Antoine Bosselut. 2023. Mitigating label biases for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14014–14031, Toronto, Canada. Association for Computational Linguistics.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Tairan Fu, Raquel Ferrando, Javier Conde, Carlos Arriaga, and Pedro Reviriego. 2024. Why do large language models (llms) struggle to count letters? *arXiv preprint arXiv:2412.18626*.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2024. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595, Mexico City, Mexico. Association for Computational Linguistics.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15466–15490. PMLR.

Nathan Godey, Éric Clergerie, and Benoît Sagot. 2024. Anisotropy is inherent to self-attention in transformers. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 35–48, St. Julian's, Malta. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kristjan Greenewald, Brian Kingsbury, and Yuancheng Yu. 2023. High-dimensional smoothed entropy estimation via dimensionality reduction. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 2613–2618. IEEE.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Kaarel Hänni, Jake Mendel, Dmitry Vaintrob, and Lawrence Chan. 2024. Mathematical models of computation in superposition. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Junqing He, Kunhao Pan, Xiaoqun Dong, Zhuoyang Song, LiuYiBo LiuYiBo, Qianguosun Qianguosun, Yuxin Liang, Hao Wang, Enming Zhang, and Jiaxing Zhang. 2024. Never lost in the middle: Mastering long-context question answering with position-agnostic decompositional training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13628–13642, Bangkok, Thailand. Association for Computational Linguistics.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. *Advances in neural information processing systems*, 29.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. 2023. The low-rank simplicity bias in deep networks. *Transactions on Machine Learning Research*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Johnny Lin Joseph Bloom. 2024. Understanding sae features with the logit lens.

Kenji Kawaguchi, Zhun Deng, Xu Ji, and Jiaoyang Huang. 2023. How does information bottleneck help deep learning? In *International conference on machine learning*, pages 16049–16096. PMLR.

Dmitrii Kharlapenko, neverix, Neel Nanda, and Arthur Conmy. 2024. Extracting sae task features for in-context learning.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Connor Kissane, Robert Krzyzanowski, Joseph Isaac Bloom, Arthur Conmy, and Neel Nanda. 2024. Interpreting attention layer outputs with sparse autoencoders. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Jiaoda Li, Yifan Hou, Mrinmaya Sachan, and Ryan Cotterell. 2024. What do language models learn in context? the structured task hypothesis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12365–12379, Bangkok, Thailand. Association for Computational Linguistics.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. 2024. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, and 18 others. 2025. A comprehensive survey on long context language modeling. *Preprint*, arXiv:2503.17407.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. 2024b. Fantastic semantics and where to find them: Investigating which layers of generative LLMs reflect

lexical semantics. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14551–14558, Bangkok, Thailand. Association for Computational Linguistics.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29).

Alexander Modell, Patrick Rubin-Delanchy, and Nick Whiteley. 2025. The origins of representation manifolds in large language models. *arXiv preprint arXiv:2505.18235*.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8298–8319, Toronto, Canada. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. 2020. Binary neural networks: A survey. *Pattern Recognition*, 105:107281.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. 2024a. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. 2024b. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. 2018. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. 2020. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, and 1 others. 2025. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*.

Wei Shi, Sihang Li, Tao Liang, Mingyang Wan, Guojun Ma, Xiang Wang, and Xiangnan He. 2025. Route sparse autoencoder to interpret large language models. *arXiv preprint arXiv:2503.08200*.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. 2025. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. *arXiv preprint arXiv:2503.05613*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Alessandro Stolfo, Ben Peng Wu, and Mrinmaya Sachan. 2025. Antipodal pairing and mechanistic signals in dense SAE latents. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Xiaoqing Sun, Alessandro Stolfo, Joshua Engels, Ben Wu, Senthooran Rajamanoharan, Mrinmaya Sachan, and Max Tegmark. 2025. Dense sae latents are features, not bugs. *arXiv preprint arXiv:2506.15679*.

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, and 3 others. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. Ieee.

Edwin Vargas, Claudia V Correa, Carlos Hinojosa, and Henry Arguello. 2024. Biper: Binary neural networks using a periodic function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5684–5693.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855.

Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, and Jian Cheng. 2018. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 4376–4384.

Qixun Wang, Yifei Wang, Xianghua Ying, and Yisen Wang. 2025. Can in-context learning really generalize to out-of-distribution tasks? In *The Thirteenth International Conference on Learning Representations*.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. 2025. MUDDFormer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. In *Forty-second International Conference on Machine Learning*.

Xin Xu, Tong Xiao, Zitong Chao, Zhenya Huang, Can Yang, and Yang Wang. 2025. Can LLMs solve longer math word problems better? In *The Thirteenth International Conference on Learning Representations*.

Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. 2019. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7308–7316.

Yidan Zhang and Zhenan He. 2024. Large language models can not perform well in understanding and manipulating natural language at both character and word levels? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11826–11842, Miami, Florida, USA. Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Related Works

Dictionary Learning (Shu et al., 2025). Modern mechanistic interpretability (Sharkey et al., 2025; Bereska and Gavves, 2024) views LLM hidden states as superpositions of atomic features, motivating efforts to disentangle them for better semantical understanding of the LLMs' operation, where **S**parse **A**uto**e**ncoder (SAE), autoencoder with training-time L_1 normalization on the hidden activations for the sample-wise sparsity, and its variants (Gao et al., 2025; Bussmann et al., 2024; Rajamanoharan et al., 2024a,b; Shi et al., 2025; Lindsey et al., 2024) is applied (Bricken et al., 2023; Huben et al., 2024; Templeton et al., 2024; Gao et al., 2025). However, these methods do not warrant a global sparse disentanglement, that is, with some guarantee to minimize the activation frequency as well as covariation **among channels** of hidden activations. Such a drawback causes the activations to be dense (Kissane et al., 2024; Rajamanoharan et al., 2024b; Sun et al., 2025; Stolfo et al., 2025), i.e., some features are activated across diverse inputs without a shared semantics, and thus cannot be uniquely interpreted. Moreover, these dense features can be viewed as mergers of multiple features (i.e., insufficient atomization), which is the main motivation of this paper: *explicitly promote the global feature sparsity and atomization by improved training methods*, to capture a larger amount of interpretable features, and to improve the accuracy of feature interpretation.

Discrete-valued Neural Networks. To utilize the information-theoretic constraints mentioned before, we round the hidden activations to binary. The framework of such binary neural networks, including rounding the real activation into 1-bit, and estimating the gradient for such rounding operation, etc., is originally proposed by (Hubara et al., 2016). Subsequently, numerous variants are proposed primarily based on different rounding operations and gradient estimations (Rastegari et al., 2016; Zhou et al., 2016; Choi et al., 2018; Vargas et al., 2024). A detailed survey can be found in (Qin et al., 2020). In this paper, we utilize these methodologies to reduce the calculation cost of the information-theoretic constraints.

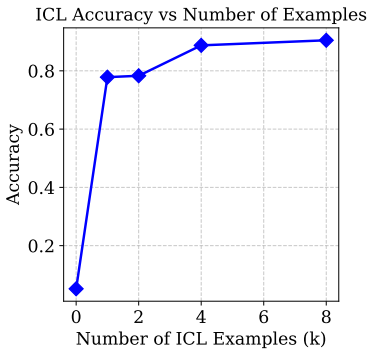

Figure 6: ICL accuracy calculated on the whole generated datasets on various k.

B Experiment Details

B.1 Synthetic Benchmarking (§3.1)

In the experiment of §3.1, we utilize the hyperparameters as: original vector dimensionaility $d = 2048$, rank $r \in \{0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$, $\alpha_e = 5 \times 10^{-7}$, $\alpha_c = 10^{-6}$.

We generate all 65536 samples, with $n = 52428$ training samples and the remaining as validation samples, with non-mentioned hyperparameters kept as the default. After the training, we run BAE again on all 65536 samples before for binary encoding h', and calculate the marginal entropy on these h' as mentioned in Eq. 4.

B.2 Tracing Normal Sentence Modeling (§3.2)

In the experiment of §3.2, we generate $n = 209715$ data samples for the training of BAE, with more 52429 data samples as the validation set from the Pile-train split, with non-mentioned hyperparameters kept as the default. Especially, we filter out all the input sentences to the Llama 3.2-1B with a token length less than 1024 to keep the length of hidden state sets among all positions aligned. After the training, we run BAE again on all 262144 samples before for binary encoding h', and calculate the marginal entropy on these h' as Eq. 4.

B.3 Tracing In-context Learning (§3.3)

Introduction of ICL and Input Format. ICL (Radford et al., 2019; Dong et al., 2022) typically utilizes concatenations of input(x)-answer(y) pairs (with amount k, called demonstrations) to define a task, and requires the LM to generate the answer of the last input (x_q, called query) similar to the demonstrations. The inputs of ICL are built like $[x_1, y_1, x_2, y_2, \ldots, x_k, y_k, x_q]$. In practice, the input samples used in the experiments are built on the toolkit StaICC (Cho and Inoue, 2025), similar to the instance below:

sentence: a genuinely moving and wisely unsentimental drama . sentiment: positive
sentence: laughs – sometimes a chuckle , sometimes a guffaw and , to my great pleasure , the occasional belly laugh sentiment: positive
sentence: that the entire exercise has no real point sentiment: negative
sentence: 90 punitive minutes of eardrum-dicing gunplay , screeching-metal smashups , and flaccid odd-couple sniping sentiment: negative
sentence: amid the new populist comedies that underscore the importance of family tradition and familial community sentiment: positive
sentence: freakshow sentiment: negative
sentence: a taste for the quirky sentiment: positive
sentence: rustic , realistic , and altogether creepy sentiment: positive
sentence: assured direction and complete lack of modern day irony sentiment:

Figure 7: An example of ICL input in experiments of §3.3 with 8 demonstrations.

Hyperparameters. We sample 262144 ICL input samples with $k \in \{0, 1, 2, 4, 8\}$, and extract the hidden states of the last token (i.e., ":") from layer $\{0, 2, 4, 6, 8, 10, 12, 14, 16\}$, with 209715 samples for the training, and the remaining for validation. After the training, we run BAE again on all 262144 samples before for binary encoding h', and calculate the marginal entropy on these h' as mentioned in Eq. 4.

Accuracy. As supplementary information, we test the classification accuracy of ICL on the given settings as shown in Fig. 6.

B.4 ComSem Evaluation on BAE and SAE Variants (§4.3)

Baselines. As shown in Table 1, we utilize 4 baselines with implementation details shown below:

- **Sparse Autoencoder** (ReLU SAE). The feedforward calculation of ReLU SAE is defined as:

$$\hat{h} = \mathrm{ReLU}(hW_{\text{in}})W_{\text{out}} + b. \tag{8}$$

During the training, the L_1 normalization is applied to the inner activation $\mathrm{ReLU}(hW_{\text{in}})$:

$$\mathcal{L}(H) = \mathcal{L}_r(H) + \alpha \sum_{h \in H} \|\mathrm{ReLU}(hW_{\text{in}})\|_1, \tag{9}$$

α is the normalization factor, defaulted to 10^{-7}.

- **Top-k SAE** (Bussmann et al., 2024). The feedforward calculation of Top-k SAE is defined as:

$$\hat{h} = \mathrm{Top}_k(hW_{\text{in}})W_{\text{out}} + b. \tag{10}$$

The Top_k function retains only the k largest elements in place, setting all others to zero. During the training, only the regression loss is utilized:

$$\mathcal{L}(H) = \mathcal{L}_r(H), \tag{11}$$

the k is defaulted to 15.

- **Gated ReLU SAE** (Rajamanoharan et al., 2024a). The feedforward calculation of Gated ReLU SAE is defined as:

$$\hat{h} = \mathrm{GateReLU}_\gamma(hW_{\text{in}})W_{\text{out}} + b. \tag{12}$$

The $\mathrm{GateReLU}_\gamma$ function is a thresholded variant of the ReLU, defined as:

$$\mathrm{GateReLU}_\gamma(x) = \begin{cases} x, & x > \gamma, \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

where γ is a tunable gating parameter, defaulted to be 0.5. During the training, the L_1 normalization is applied to the inner activation $\mathrm{GateReLU}_\gamma(hW_{\text{in}})$:

$$\mathcal{L}(H) = \mathcal{L}_r(H) + \alpha \sum_{h \in H} \|\mathrm{GateReLU}_\gamma(hW_{\text{in}})\|_1, \tag{14}$$

- **Transcoder** (Dunefsky et al., 2024). The feedforward calculation of Transcoder is defined as:

$$h_2 = \mathrm{ReLU}(hW_{\text{in}})W_{\text{out}} + b. \tag{15}$$

Where the h is the input towards the $(l-1)$-th MLP block[8]. During the training, the h_2 is aligned to the output of the $(l-1)$-th MLP block,

[8] We note that the $(l-1)$-th MLP block produces the hidden states of layer l.

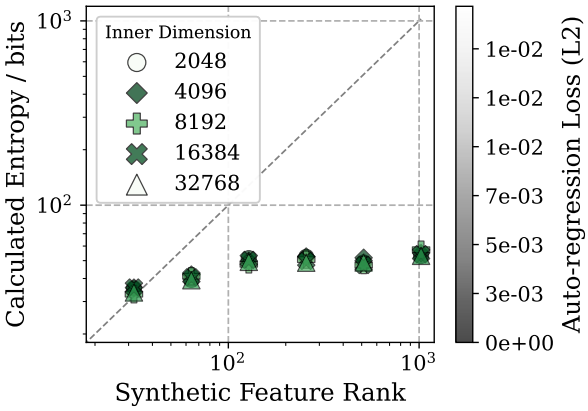

Figure 8: Evaluation results on LM hidden states.

i.e., the hidden state of layer l, and L_1 normalization is applied to the inner activation:

$$\mathcal{L}(H) = \frac{1}{n_b}\|H_2 - H_l\|_2 + \alpha \sum_{h \in H} \|\mathrm{GateReLU}_\gamma(hW_{\text{in}})\|_1, \tag{16}$$

due to the significant difference in methodology with SAE variants, we do not regard Transcoder as a major comparison object.

Parameters of BAE/SAE Training. We sample $n = 8243323$ hidden state vectors from the specific layers of Llama 3.2-1B on the Pile-train split, with 6594658 as the training samples for BAE/SAE, and the remaining for the validation. The autoencoders are trained for 200 epochs, with $\alpha_e = 0$ in the first 50 epochs.

We visualize the burstiness-based activation magnitude calculation of BAE in Fig. 10.

Parameters of ComSem. The pseudocode of ComSem is shown in Algorithm 1, with default parameters $N = 8192$, $n_I = 5$, $n_T = 8$, $k = 10$. The evaluations are conducted on the train split of BookCorpus, which is different from the training set of BAE/SAEs (Pile). The $\bar{h}'$ is averaged among all the input instances of the evaluation set. The instructions for calling the backend LLMs are shown in Fig. 22 and 23, with queries' format the same as the given examples. We use GPT-4.1 and GPT-4.1-mini (Achiam et al., 2023) as the backend LM.

B.5 Vector Compression by BAE

As shown in Fig. 9, the compression of a hidden state vector h proceeds as follows:

1. Given the input h to be compressed, we first compute its binary encoding h' by applying the input projection W_{in} and binarize function Γ of a BAE, and then measure the burstiness β with respect to the prior distribution $\bar{h}'$, as described in §4.2 and Fig. 10.
2. Given a threshold[9] $B \in [\log_2 0.5, 0]$, we bookkeep only the channel indices i for which $\beta_i > B$, using these indices as the compressed representation of h.
3. To reconstruct h from the bookkept indices, we round $\bar{h}'$ to a binary vector, flip the bits of $\mathrm{round}(\bar{h}')$ at the stored indices, and then pass the flipped binary vector through W_{out} and b to reconstruct the h.

Feat. Source	Model	Feature Activated	ComSem$_{4.1}$ FI#	ComSem$_{4.1}$ Score
Llama 3.2-1B Layer 11 $d = 2048$	ReLU SAE	2065	1380	0.260
	Top-k SAE	3417	2684	0.452
	Gated ReLU SAE	1226	1026	0.557
	ReLU SAE Resc.	744	482	0.278
	TransCoder	1794	1090	0.239
	BAE (top 3500)	**3500**	**3500**	**0.574**
Llama 3.2-1B Layer 14 $d = 2048$	ReLU SAE	2528	1600	0.217
	Top-k SAE	2702	2004	0.418
	Gated ReLU SAE	2948	2250	0.435
	ReLU SAE Resc.	3962	2661	0.274
	TransCoder	3401	2166	0.267
	BAE (top 3970)	**3970**	**3970**	**0.496**
Llama 3.2-3B Layer 20 $d = 3072$	ReLU SAE	1923	1289	0.312
	Top-k SAE	3234	2508	0.425
	Gated ReLU SAE	4628	3580	0.437
	ReLU SAE Resc.	2122	1451	0.294
	TransCoder	5508	3424	0.257
	BAE (top 5700)	**5700**	**6805**	**0.544**

Table 4: Evaluation of BAE with low-score features ignored and baselines.

Hyperparameters & Experiment Settings. We sample 117864 hidden state vectors from the specific layers on BookCorpus, then utilize the BAE trained on Pile for the compression, following the processing above. The $\bar{h}'$ is averaged among all the input instances of the compressed set.

C Evaluating Entropy Estimation on LLM Hidden States Set

To extend our discussion in §3.1, in this section, we focus on evaluating the entropy evaluation on benchmarks extracted from real LLMs. In detail, similar to §3.1, given a ground-truth entropy r, we sample n r-length token sequences, with elements randomly sampled from two given tokens (in our implementation, tokens with index 0 and 1). From

[9] Notice that the threshold greater than 0 is trivial, given the $h' \in \{0,1\}^{d'}$, $\bar{h}' \in [0,1]$, so that all the elements of $\log_2 |h' - \bar{h}'| \leqslant 0$. And burstiness less than $\log_2 0.5$ causes wrong flip of bits in Step 3.

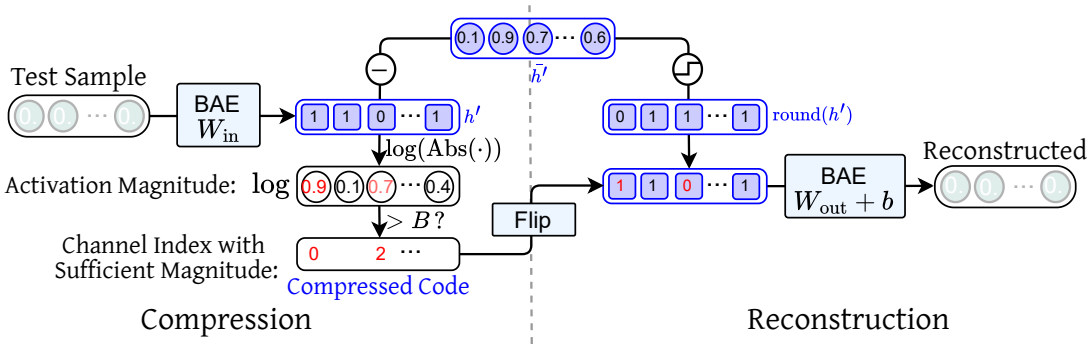

Figure 9: Diagram of hidden state compression and reconstruction utilizing BAE.

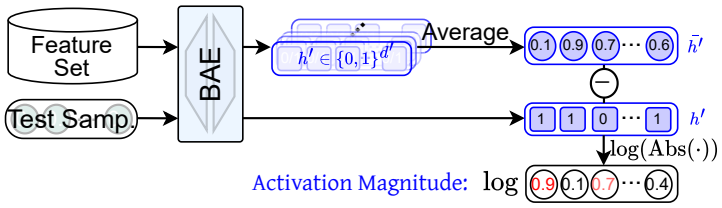

Figure 10: Burstiness-based activation magnitude calculation process described in §4.2.

such a sampling method, we can get an input sequence set with entropy r. Input these sequences into LMs (Llama 3.2-1B in our implementation) and extract the hidden states from the l-th layer (Layer 12 in our implementation), we can get a hidden states set with entropy **not larger than** r[10]. Repeat the training-and-evaluation processing shown in §3.1, we calculate the entropy with varios r as shown in Fig. 8, where the results are consistent with our inference. But this also indicates that such methods based on LM hidden states cannot serve as a strong benchmark and should only be considered a complement to our synthetic approach.

D Interpretability Score Distribution: Selecting a Better Feature Set

We visualize the ComSem interpretability score distribution among features of BAE and baselines in Fig. 24 - 28, where compared to all the baselines, BAE significantly increased the amount of both high-score and low-score features. As shown in Appendix G, these low-score features are *non-literal*, that is, the semantics of these features can not be interpreted by the token inputs or outputs in ComSem, so that, despite serving as an effective proof of BAE's stronger extraction capability, harm the average interpretability score among all the activated features.

[10]Notice that in §3.2, we show that during the forward pass of an LM, the hidden state of a token may not contain all information from its preceding context; therefore, hidden states on a single position may carry reduced information.

However, as shown in the curves of Fig. 24 - 28 visualizing the average interpretability scores with all the previous sorted features, the curves of BAE are located above the baseline models in most cases. This indicates that when we remove some low-score features, BAE will achieve the highest average interpretability scores and also the number of interpretable features. This is a benefit of BAE's extensive feature extraction capability, which means that compared to baseline models that extract a limited number of features, BAE has space to perform trade-offs on both the number of features and interpretability scores, while ensuring that both are optimal, as shown in Table 4 with some of the low-score BAE features ignored.

E Natural Low-entropy Tendency of BAE Training

Moreover, interestingly, when the estimated entropy constraint is disabled (red in Fig. 2), the entropy gathers higher but near the diagonal, suggesting that BAE without entropy training objective can find a relatively low-entropy encoding h', even if no normalizations are conducted to minimize the encoding entropy. To get a closer observation, we plot the self-regression loss $\mathcal{L}_r$ and training-time entropy $\mathcal{L}_e$ on every training step, with $r = 2$ (more cases in Appendix I.2), as shown in Fig. 11, with normal setting (left) or $\alpha_e = \alpha_c = 0$ (right). In the normal setting (left), a stable and monotonous convergence to both minimal entropy and self-regression loss can be observed. However, if the

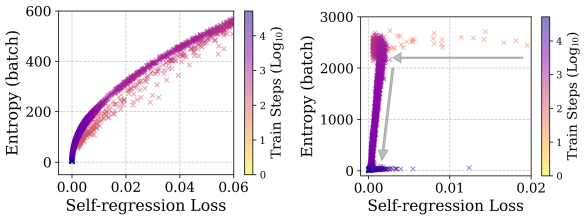

Figure 11: Training dynamics of BAE (**left**) with entropy objective and (**right**) without entropy objective. The horizontal axes are the self-regression training loss ($\mathcal{L}_r$), and the vertical axes are the entropy calculated from the training batch.

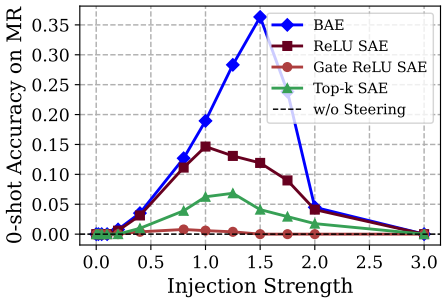

Figure 12: Steering vector reconstruction results on MR, Llama 3.2-1B Layer 11.

entropy loss is disabled (right), the training dynamics become multiphase and non-monotonic. Specifically, **Phase 1:** the loss rapidly decreases while the hidden activations maintain relatively high entropy; **Phase 2:** entropy sharply drops to near zero, with the loss remaining low, even if no entropy and weight penalty are utilized; and **Phase 3:** both loss and entropy oscillate in a narrow range at the end of training. Such an observation suggests that: Gradient descent on simple regression loss finds relatively "simple" representations, after a long-term stagnation on loss value, supporting previous work on training dynamics (Tishby and Zaslavsky, 2015; Saxe et al., 2018; Huh et al., 2023; Nanda et al., 2023; Shah et al., 2020; Frankle and Carbin, 2019; Bartlett et al., 2020). Additionally, enabling the entropy penalty suppresses the harmful Phase 3 oscillations, confirming its effectiveness.

F Proof: More Burst Features Carry More Information

This section substantiates our claim that features with higher burstiness contain more information and should therefore be preferentially selected as activated features. Our proof is grounded in fundamental principles of information theory:

Theorem F.1 (Burst Features Carry More Information)**.** *Let* $X_1 \sim \text{Bernoulli}(p_1)$*,* $X_2 \sim \text{Bernoulli}(p_2)$*, the information of an observation (i.e., the actual value of hidden activation in a specified channel)* $x \in \{0, 1\}$ *can be written as* $\mathbb{I}_X(x) := -\log \Pr[X = x]$*. Then:*

$$\mathbb{I}_{X_1}(x_1) \geqslant \mathbb{I}_{X_2}(x_2) \Leftrightarrow |x_1 - p_1| \geqslant |x_2 - p_2|$$

The proof is straightforward. That is, a more bursty channel carries more information. Therefore, identifying these more bursty features is more conducive to reconstructing this particular observation from the average (i.e., the typical set).

G Case Analysis: Extracted Features

In this section, we observe several features along with their corresponding tokens and contexts where the investigated feature is activated, from the BAE trained on Layer 14 hidden states of Llama 3.2-1B. We observe in the category of these high-score and low-score features, and find that the low-score features are more folded and implicit in the hidden states, so it is harder to extract.

H Case Analysis: BAE Extract Better Steering Vectors

As empirical evidence that BAE can extract effective features, we reconstruct the ICL task vector (Hendel et al., 2023) using the features extracted by BAE and SAE baselines (Kharlapenko et al., 2024). In detail, following the framework proposed by Cho and Hockenmaier (2025), we construct an 8-shot sample set as the "good samples", and a 0-shot sample set as the "bad samples" from MR (Pang and Lee, 2005), both equipped with 512 samples. We input both sets respectively into LMs, getting the hidden states of the last token from Layer 11 of Llama 3.2-1B, and encoding the hidden states by BAE or SAE baselines. Then, we calculate $h_D = \bar{h'}^+ - \bar{h'}^-$, where the $\bar{h'}^+$ and $\bar{h'}^-$ are the mean hidden activation in these two sets produced by BAE or SAE baselines, and only keep the elements at the top 256 in absolute value (i.e., zeroing the non-significant values) as $\tilde{h_D}$. $\gamma \tilde{h_D} W_{\text{out}}$ is calculated as the steering vector with γ as injection strength, and injected into the original Layer 11 of the last-token hidden states from the 0-shot test samples, with accuracy shown in Fig. 12. The results show that the BAE-mediated steering vector most effectively improves zero-shot accuracy, serving as a case study demonstrating that BAE can extract more effective features from LLMs.

Feature index: #111
Feature interpretation: "indefinite pronouns and quantifiers"
Interpretability score: 0.75

Activate tokens & contexts:
#1: "but just **one** look at a minion sent him practically catatonic ."
#2: "while **some** might look on her as having a mark against her character being an unwed mother , she had lived a relatively non-rebellious life ."
#3: "one **thing** her mother prided herself on was being on time and lending a hand ."
#4: "while it was good that he had so **many** male role models , she only hoped he had n't inherited too much of his father 's personality ."
#5: "after megan watched them disappear in the crowd of family and friends waiting in the church alcove , she bypassed **everyone** by turning right and heading down the hallway ."
#6: "more than **anything** , she wanted to have a little fun herself ."
#7: "tell me **what** ?"
...

Figure 13: Case analysis for feature 111 "indefinite pronouns and quantifiers" with interpretability score 0.75 in Llama 3.2-1B Layer 14.

H.1 High-score Features

We first list some features with high interpretability scores, along with their corresponding activation tokens and contexts, as shown in Fig. 13 - 17. Among these high-score cases, we observe that: for the ComSem, it is easy to correctly identify semantic similarities grounded in the natural semantics of single tokens (e.g., different variants of the same word (Fig. 17), nouns describing objects of the same category (Fig. 14, 15, 16), or function words serving similar grammatical roles (Fig. 13)). We infer that: (1) The similarities in the hidden states of such tokens can be easily extracted from the residual stream based on the embedding vectors. Also, (2) the similarities in these activated tokens are clear towards the backend LLMs. So that these features can be interpreted into natural language interpretation from the plain tokens by LLMs appropriately.

H.2 Low-score Features

Also, we list some features with low interpretability scores, as shown in Fig. 18 - 21. Among these low-score features, we summarize the characteristics of these features: (1) **Subwords from the tokenizer.** As shown in Fig. 19, 21, these subword tokens share no obvious commonality even for humans, so that they confuse the LLM, leading it to produce irrelevant interpretations, which illustrates the drawback of token-based interpretation, as noted in Limitations. (2) **Function words with similar context.** As shown in Fig. 18, 20, these function-word tokens, despite appearing in similar contextual environments (often originating from the same sentence or even the same region within a sentence), lack surface-level commonality and are therefore difficult to interpret by ComSem, which highlights the urgency of interpreting directly from the decoded feature vectors instead of original tokens. Moreover, from this phenomenon, a further hypothesis is that function words without clear semantics may be more likely to concentrate contextual information during contextualization. (3) **Character level similarity.** As shown in Fig. 21, the tokens activated on feature 5949 all end with "em", according to previous works (Zhang and He, 2024; Fu et al., 2024), backend LLM may struggle to process such inputs, causing a low interpretability score.

H.3 Scale of High-score Features and Low-score Features

In this section, we argue that: compared to high-scoring features, low-scoring features are embedded in more fine-grained structures of the hidden states, making them difficult to cluster with UMAP. Consequently, extracting such features requires a stronger extraction capability. Compared to the baselines, BAE captures more of these features,

[10] To comply with the requirements of the ethics review, we have redacted potentially offensive or NSFW content from these input cases.

Feature index: #1045
Feature interpretation: "numerical quantifiers and ordinals"
Interpretability score: 0.75

Activate tokens & contexts:
#1: "**one** day when they had their own place again , she would get him a dog ."
#2: "sean acknowledged her with a **two** finger salute before cranking up and pulling down the driveway ."
#3: "**one** thing her mother prided herself on was being on time and lending a hand ."
#4: "**one** time ."
#5: "but that was **six** months ago ."
#6: "aidan introduced him to his **four** sisters and their husbands ."
#7: "after exchanging hugs with emma and reassuring her at least **twenty** times that she would be fine and that she did need to go home , pesh led megan out the front door ."
...

Figure 14: Case analysis for feature 1045 "numerical quantifiers and ordinals" with interpretability score 0.75 in Llama 3.2-1B Layer 14.

confirming its superior feature extraction ability.

In detail, we conduct UMAP (McInnes et al., 2018), a dimensionality reduction based on the macroscopic adjacency structure from Euclidean distance, on hidden states of all the tokens from the sentences whose token activates the investigated feature, as shown in Fig. 42 for 3 high-score features, and Fig. 43 for 3 low-score features, where an important observation is that: the hidden states activated by high-scoring features form clear clusters, suggesting macroscopic spatial similarity. In contrast, the hidden states activated by low-scoring features are more dispersed overall, exhibiting similarity only within certain subspaces (i.e., along the corresponding rows of the activated features in W_{out}). That is, *the low-score features are more folded and implicit*, so that harder to extract.

I Augmentation Experiments and Results

I.1 Activation Magnitude Visualization of BAE and SAE

We visualize the activation magnitude of each input h_0 (vertical axis) on each channel (horizontal axis) to observe the sparsity of BAE (left sub-figures) and SAE (right sub-figures) as shown in Fig. 35 - 38. Besides the clear sparsity of BAE, the visualization of SAE shows vertical stripes, indicating consistently high-activated channels, which are absent in BAE. This suggests that utilizing global normalization for sparsity, and calculating burstiness as the activation magnitude effectively suppresses the dense activations seen in SAE.

I.2 More Cases for Fig. 11

We examine the training dynamics of BAE with and without the entropy objective ($\mathcal{L}_e$) under additional settings, extending the observations from Fig. 11. The results, shown in Fig. 29 - 34, reveal that a larger rank leads to more complex training behavior, while incorporating the entropy objective reshapes the dynamics into a smoother and more monotonic form.

I.3 More Cases for Fig. 5

We extend the investigation scope of Fig. 5, where the activation frequency of each feature is visualized, as shown in Fig. 39 - 41. The conclusion derived from these augmentation results is consistent with the main body.

J Compliance Statement

Open-source Licenses. The Llama 3.2 models are under the Llama 3.2 Community License, the Pile and BookCorpus corpus are under the MIT License, the SST-2 is under the Apache License. All the open-source resources are utilized under their original usage.

AI Usage. LLM tools are used for explaining the BAE / SAE features, and polishing the writing of this paper.

Feature index: #2289
Feature interpretation: “verbs of visual attention or perception”
Interpretability score: 0.875

Activate tokens & contexts:
#1: “but just one **look** at a minion sent him practically catatonic .”
#2: “each time she **looked** into mason ’s face , she was grateful that he looked nothing like his father .”
#3: “megan asked , g**azing** from noah ’s...[10]”
#4: “**peek**ing out from the covering , she saw emma was wearing her signature color , green .”
#5: “of course , he ’d probably argue that while the gown might not have held up , he still **looks** fabulous and much younger than his age .”
#6: “megan **glanced** between the two of them .”
#7: “she **stared** into his face before she responded .”
...

Figure 15: Case analysis for feature 2289 “verbs of visual attention or perception” with interpretability score 0.875 in Llama 3.2-1B Layer 14.

Feature index: #7700
Feature interpretation: “body part nouns”
Interpretability score: 1.0

Activate tokens & contexts:
#1: “when he started to whine , she shook her **head** .”
#2: “he grinned and then happily dodged her mother ’s **arms** for her father ’s instead , which made megan smile .”
#3: “noah momentarily stopped sucking on the bottle to flash a quick smile , which warmed megan ’s **heart** .”
#4: “emma rolled her **eyes** .”
#5: “casey gasped as her hand flew to her **chest** dramatically .”
#6: “casey tapped her **chin** with her index finger ”
#7: “megan pursed her **lips** at the prospect .”
...

Figure 16: Case analysis for feature 7700 “body part nouns” with interpretability score 1.0 in Llama 3.2-1B Layer 14.

Feature index: #8183
Feature interpretation: “modal auxiliary verbs expressing ability or possibility”
Interpretability score: 1.0

Activate tokens & contexts:
#1: “as she went to the couch and picked him up , she **could** n’t help finding it amusing that out of everyone he was going to see today , he was most excited about being with aidan and emma ’s black lab , beau .”
#2: “sadly , she **could** n’t say that her first love was davis , mason ’s father .’
#3: “considering i have two younger brothers , i think i **can** handle him .”
...

Figure 17: Case analysis for feature 8183 “modal auxiliary verbs expressing ability or possibility” with interpretability score 1.0 in Llama 3.2-1B Layer 14.

Feature index: #514
Feature interpretation: “common auxiliary verbs and determiners”
Interpretability score: 0.0

Activate tokens & contexts:
#1: “he ’d seen the movie almost by mistake , considering he was a little young for the pg cartoon , but with older cousins , along with her brothers , mason was **often** exposed to things that were older .”
#2: “he ’d seen the movie almost by mistake , considering he was a little young for the pg cartoon , but with older cousins , along with her brothers , mason was often exposed to things that **were** older .’
#3: “she liked to think being surrounded by adults and older kids was one reason why he **was** a such a good talker for his age .”
#4: “she liked to think being surrounded by adults and older kids was one reason why he was a such **a** good talker for his age .”
#5: “it was only his build that he **was** taking after his father .”
#6: “while it had been no question that she wanted him as godfather for mason , she had been extremely honored when he and his wife , emma , had asked her to be their son , noah ’**s** , godmother .”
#7: “while it had been no question that she wanted him as godfather for mason , she had been extremely honored when he and his wife , emma , had asked her to be their son , noah ’s **,** godmother .”
#8: “i plan on spoiling noah rotten and corrupting him as **only** a good auntie can do !”
#9: “i plan on spoiling noah rotten and corrupting him as only **a** good auntie can do !”
#10: “i plan on spoiling noah rotten and corrupting him as only a **good** auntie can do !”
...

Figure 18: Case analysis for feature 514 “common auxiliary verbs and determiners” with interpretability score 0.0 in Llama 3.2-1B Layer 14.

Feature index: #2410
Feature interpretation: “abbreviations or truncated forms of words”
Interpretability score: 0.0

Activate tokens & contexts:
#1: “he reminds me of that bollywood actor john **ab**raham , ” casey said .”
#2: “as he eyed the massive statue of jesus , he fidgeted **abs**ently with his tie .’
#3: “he was taken **ab**ack by her words and the passion with which she delivered them .”
#4: “her eyes **fr**antically scanned the room .”
#5: “her gaze flicked down to the tan , **mus**cled arm .”
#6: “he welcomed the pain as she went over the edge , **conv**ulsing and screaming against his hand .”
#7: “instead , she just enjoyed being close to him , the feel of his hand on her , his strong , **mus**cled thighs beneath her .”
...

Figure 19: Case analysis for feature 2410 “common auxiliary verbs and determiners” with interpretability score 0.0 in Llama 3.2-1B Layer 14.

Feature index: #3479
Feature interpretation: “adverbs and pronouns used as modifiers or objects”
Interpretability score: 0.0

Activate tokens & contexts:
#1: “each time she looked into mason ’s face , she was grateful that he looked **nothing** like his father .”
#2: “with a sigh , she stepped into the dress and slid **it** over her hips .’
#3: “while it boasted a sweetheart neckline , the hemline **fell** just below her knees .”
#4: “while it boasted a sweetheart neckline , the hemline fell **just** below her knees .”
#5: “we ’ll see you there in **just** a few , ” her mother called .”
#6: “with her grades **already** in the toilet , she was unprepared for the emotional breakdown she experienced when carsyn broke up with her .”
#7: “with her grades already in **the** toilet , she was unprepared for the emotional breakdown she experienced when carsyn broke up with her .”
...

Figure 20: Case analysis for feature 3479 “adverbs and pronouns used as modifiers or objects” with interpretability score 0.0 in Llama 3.2-1B Layer 14.

Feature index: #5949
Feature interpretation: “tokens related to trembling or shaking actions”
Interpretability score: 0.0

Activate tokens & contexts:
#1: “her chin **trem**bled as she replied , “ i want that for my son . ””
#2: “her lip **trem**bled .’
#3: “her chin **trem**bled before big , fat tears slid down her cheeks .”
#4: “her body **trem**bled slightly at his words .”
#5: “**lem** me guess , you bribed him sexually to get him to give up a perfectly good saturday watching two kids who are under two .”
...

Figure 21: Case analysis for feature 5949 “tokens related to trembling or shaking actions” with interpretability score 0.0 in Llama 3.2-1B Layer 14.

Algorithm 1: Common **Sem**antics-based Feature Interpretation and Evaluation (ComSem).

Parameters: $f : \mathbb{R}^d \to \mathbb{R}^{d'}$: Feature encoder, mapping original input h_0 to the **magnitude** of decomposed features h' (e.g., encoding part of a typical SAE, or burstiness calculation for a BAE).
d': feature numbers.
n_I: samples for generating interpretation.
n_T: max test samples.
k: top-? features that are seemed as activated.
$\mathrm{LM}(\cdot)$: backend LM client for generate the interpretation and test.
$\mathrm{Prompt}_{\mathrm{Interp}}(\cdot)$: Prompt template for interpretating the features (described in Fig. 22).
$\mathrm{Prompt}_{\mathrm{Test}}(\cdot)$: Prompt template for interpretating the features (described in Fig. 23).

Input: Set (of amount N) of token-hidden state of token-context triple: $\mathcal{S} = \{(t, h_0, c)^{(i)}, c = t_j\}_{i=1}^{N}$

Initialization: $\mathcal{D}_S = \{\}$: dictionary for each feature (length: d'), keeping the samples where the corresponding feature is activated.
$\mathcal{D}_I = \{\}$: dictionary for each feature, keeping the interpretation and evaluation score.
$\mathrm{FA} = 0$: Number of feature activated.
$\mathrm{FI} = 0$: Number of interpretable features.
$\mathrm{Score} = 0$: Averaged interpretability score amoung the successfully interpreted features.

```
/* 1.  Find the activated input samples for each feature channel.                    */
for i := 1 to n do
    a := f(h_0^(i)) // Calculate activation magnitude of each feature channel.
    I := arg max_k(a) // Find top-k activated feature index.
    for j in I do
        D_S[j].append(t^(i), c^(i)) // Bookkeep the activated sample for each feature
            channel.
    end
end
/* 2.  Get the interpretation for each feature channel.                              */
for i := 1 to d' do
    if len(D_S[i]) ⩾ n_I + 1 then
        // Query the LLM for the interpretation, only use n_I samples, with the
            remaining ones for evaluation.
        D_I[i] := {"Interp" = LM(Prompt_Interp(D_S[i][1 : n_I])), "Activated" = True}
        FA := FA + 1
    else
        // Reject the interpretation if samples are not sufficiently loaded.
        D_I[i] := {"Interp" = None, "Activated" = False}
    end
end
/* 3.  Evaluate the interpretation for activated feature channels.                   */
for i := 1 to d' do
    if D_I[i]["Activated"] then
        D_I[i]["Score"] := 0
        for (t^(j), c^(j)) in D_S[i][n_I + 1 : n_I + n_T] do
            // For each query in the test set (number bounded by the n_T), query
                the LLM for judging the matching of each input sample and its
                interpretation in True or False.
            if LM(Prompt_Test((t^(j), c^(j)), D_I[i]["Interp"]) then
                D_I[i]["Score"] = D_I[i]["Score"] + 1
            end
        end
        D_I[i]["Score"] = D_I[i]["Score"]/len(D_S[i][n_I + 1 :])
        Score := Score + D_I[i]["Score"]
        if D_I[i]["Score"] > 0 then
            FI := FI + 1
        end
    end
end
return D_I, Score/FA, FA, FI
```

Instruction:
I will provide a set of tokens along with their positions (this position may vary depending on the tokenizer) and the surrounding context. Please describe what these tokens have in common using concise expressions such as “date expressions”, “words ending in ‘ing’”, or “adjectives”.
Please choose the most specific term while ensuring commonality, and avoid using overly general terms like “words”, “English tokens”, “high-frequency English lexemes”, or “phrases”.
Non-semantic or non-linguistic terms such as “BPE Subword Token” are strictly prohibited.
Any additional information, explaination, or context are strongly prohibited. Only return one phrase.

Example 1:
Token: “running” at position 3 in sentence: “She is running in the park.”
Token: “eating” at position 3 in sentence: “He is eating an apple.”
Token: “sleeping” at position 4 in sentence: “The baby is sleeping on the sofa.”
Token: “jumping” at position 2 in sentence: “They are jumping over the fence.”
Token: “talking” at position 3 in sentence: “We are talking about the project.”
The commonality is: -ing verbs of human behavior

Example 2:
Token: “yesterday” at position 4 in sentence: “I went there yesterday.”
Token: “last week” at position 5 in sentence: “She arrived last week.”
Token: “in 1998” at position 6 in sentence: “They moved here in 1998.”
Token: “last year” at position 5 in sentence: “We met last year.”
The commonality is: past time expressions

Example 3:
Token: “happy” at position 4 in sentence: “She looks very happy today.”
Token: “angry” at position 5 in sentence: “They were extremely angry about it.”
Token: “sad” at position 4 in sentence: “He felt really sad after the call.”
The commonality is: emotional adjectives

Example 4:
Token: ”dog” at position 2 in sentence: ”The dog barked loudly.”
Token: ”cat” at position 2 in sentence: ”The cat chased the mouse.”
Token: ”bird” at position 2 in sentence: ”The bird sang beautifully.”
Token: ”fish” at position 2 in sentence: ”The fish swam gracefully in the tank.”
The commonality is: animal nouns

(...)

Example 11:
Token: “sad” at position 4 in sentence: “He felt really sad after the call.”
Token: “angry” at position 5 in sentence: “They were extremely angry about it.”
Token: “negative” at position 4 in sentence: “She had a negative reaction to the news.”
The commonality is: negative emotion adjectives

Now, please analyze the following tokens and their contexts:
(Test Sample)

Figure 22: Instruction text for interpreting the feature semantics.

Background:
I will provide a token, its position in the sentence, the surrounding context, and a candidate description of the token's role or type given the context.
Your task is to judge whether the given description accurately characterizes the token in its context.
Please respond with either:
- "Yes" (if the description is accurate), or
- "No" (if it is inaccurate)
Any additional information, explaination, or context are strongly prohibited. Only return "Yes" and "No".

Example 1:
Token: "running" at position 3 in sentence: "She is running in the park."
Candidate description: "present participle"
Answer: Yes

Example 2:
Token: "dog" at position 2 in sentence: "The dog barked loudly."
Candidate description: "adjective"
Answer: No

Example 3:
Token: "quickly" at position 4 in sentence: "He ran quickly toward the exit."
Candidate description: "manner adverb"
Answer: Yes

Example 4:
Token: "first" at position 4 in sentence: "This is the first time I have seen this."
Candidate description: "ordinal number"
Answer: Yes

Example 5:
Token: "to" at position 5 in sentence: "I want to go to the store."
Candidate description: "emotional verb"
Answer: No

Example 6:
Token: "looking" at position 3 in sentence: "She is looking forward to the event."
Candidate description: "verb related to oral communication"
Answer: No

(...)

Example 22:
Token: "fish" at position 2 in sentence: "The fish swam gracefully in the tank."
Candidate description: "noun describing an animal"
Answer: Yes

Now, please analyze the following tokens, their positions, contexts, and candidate descriptions:
(Test Sample)

Figure 23: Instruction text for testing the interpretation.

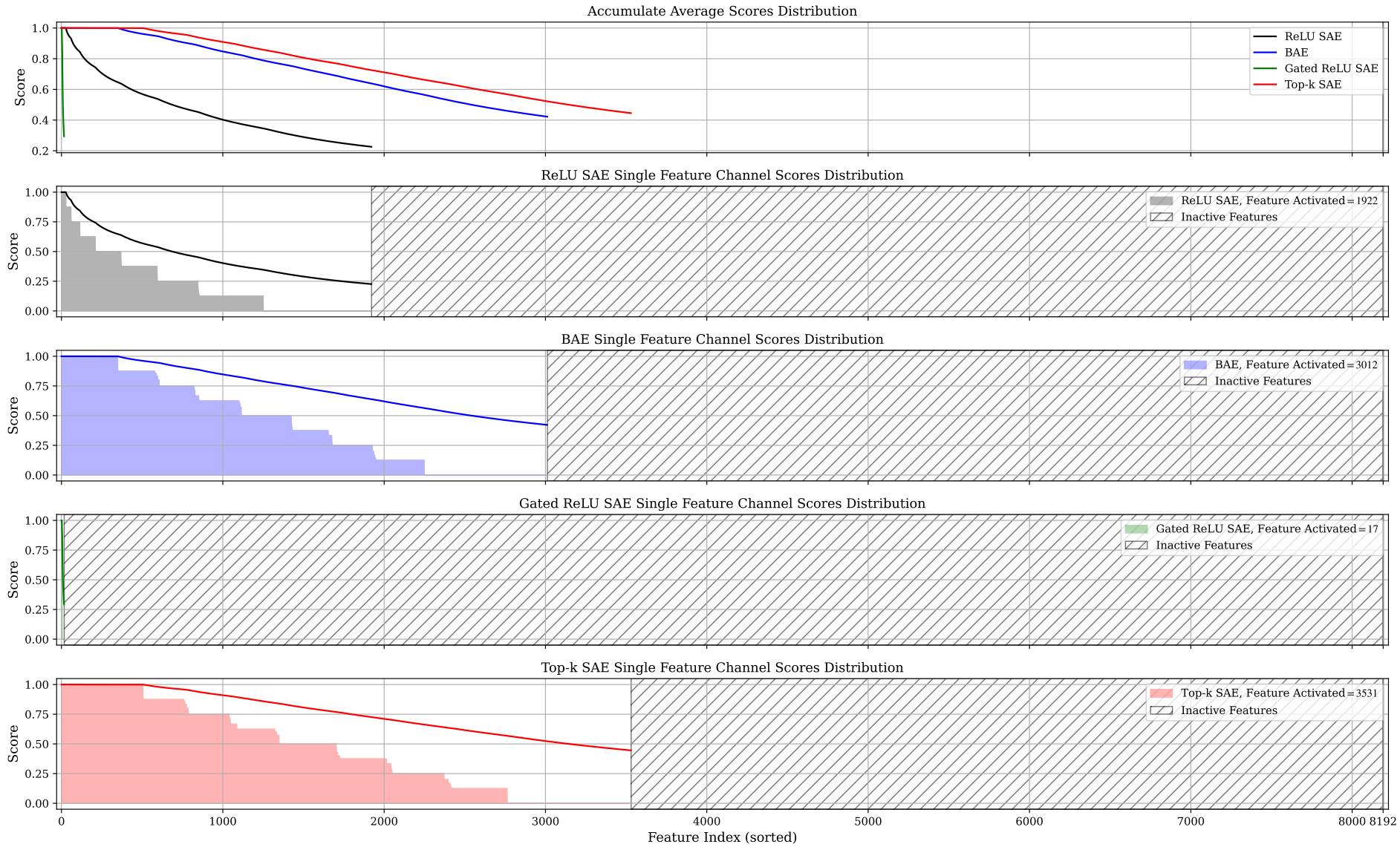

Figure 24: ComSem interpretability score distribution of each feature channel on layer 5 of Llama 3.2-1B. The colored areas represent the interpretability scores of individual channels (sorted), and the curves show the averaged interpretability scores across all channels previously.

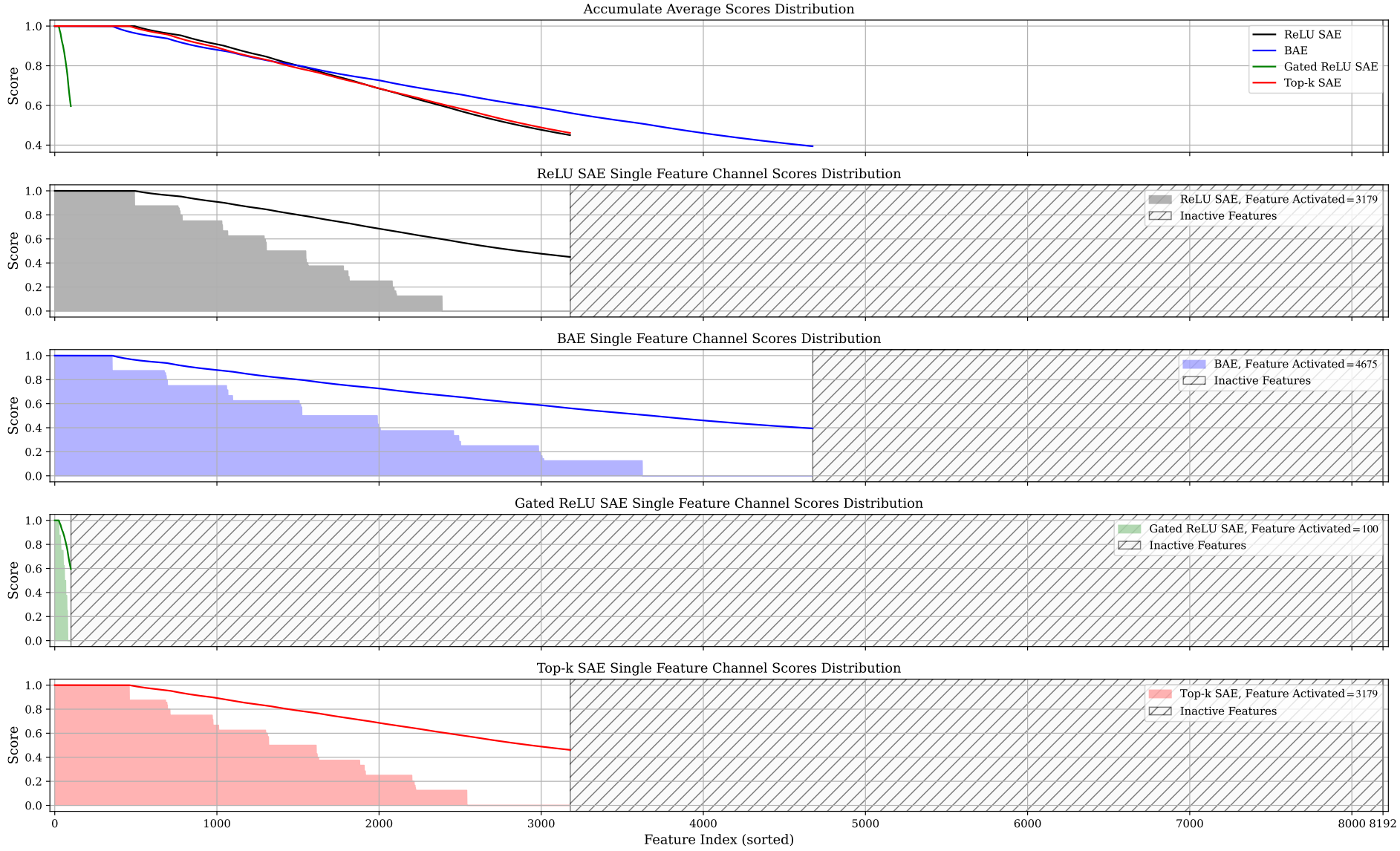

Figure 25: ComSem interpretability score distribution of each feature channel on layer 9 of Llama 3.2-1B.

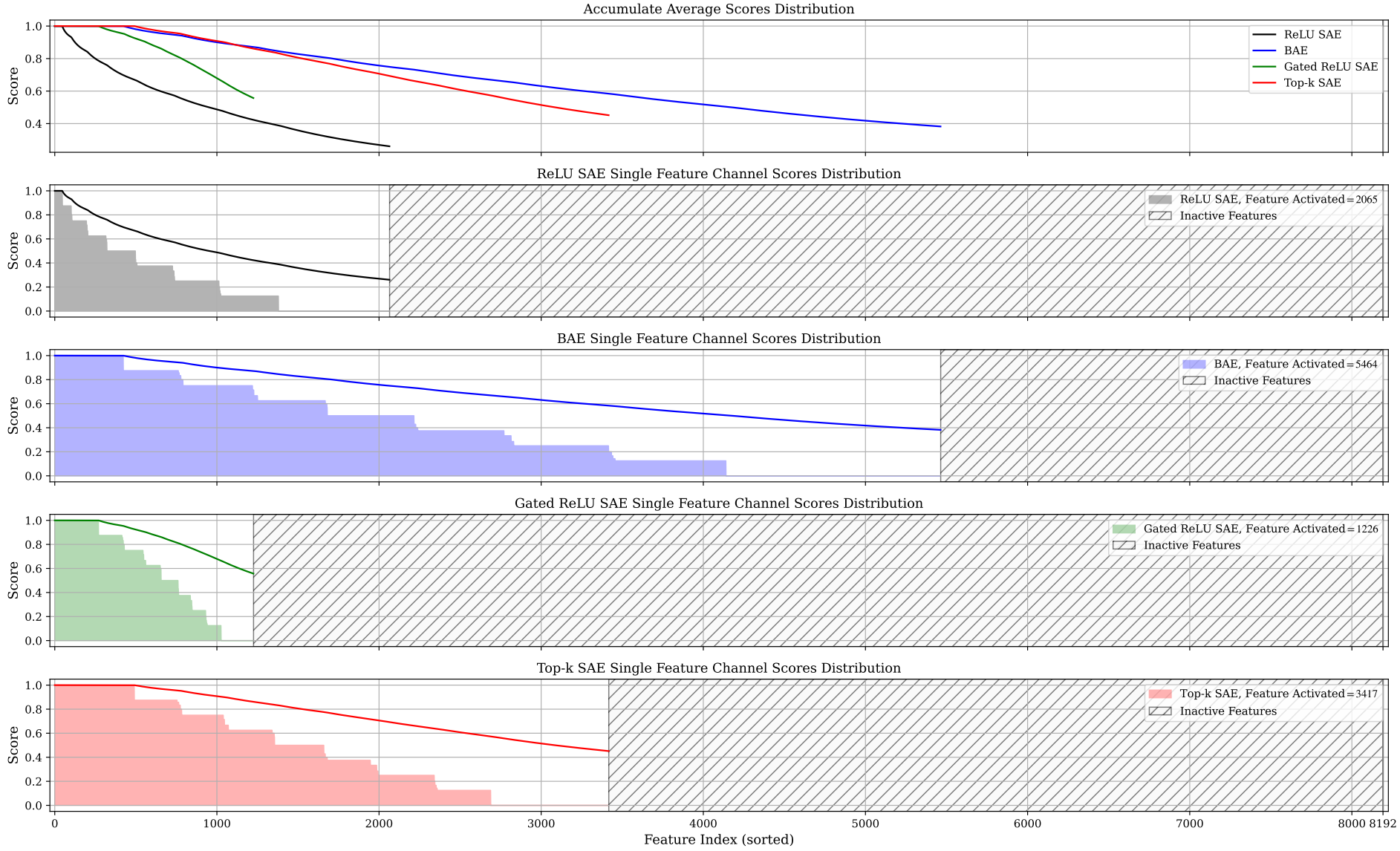

Figure 26: ComSem interpretability score distribution of each feature channel on layer 11 of Llama 3.2-1B.

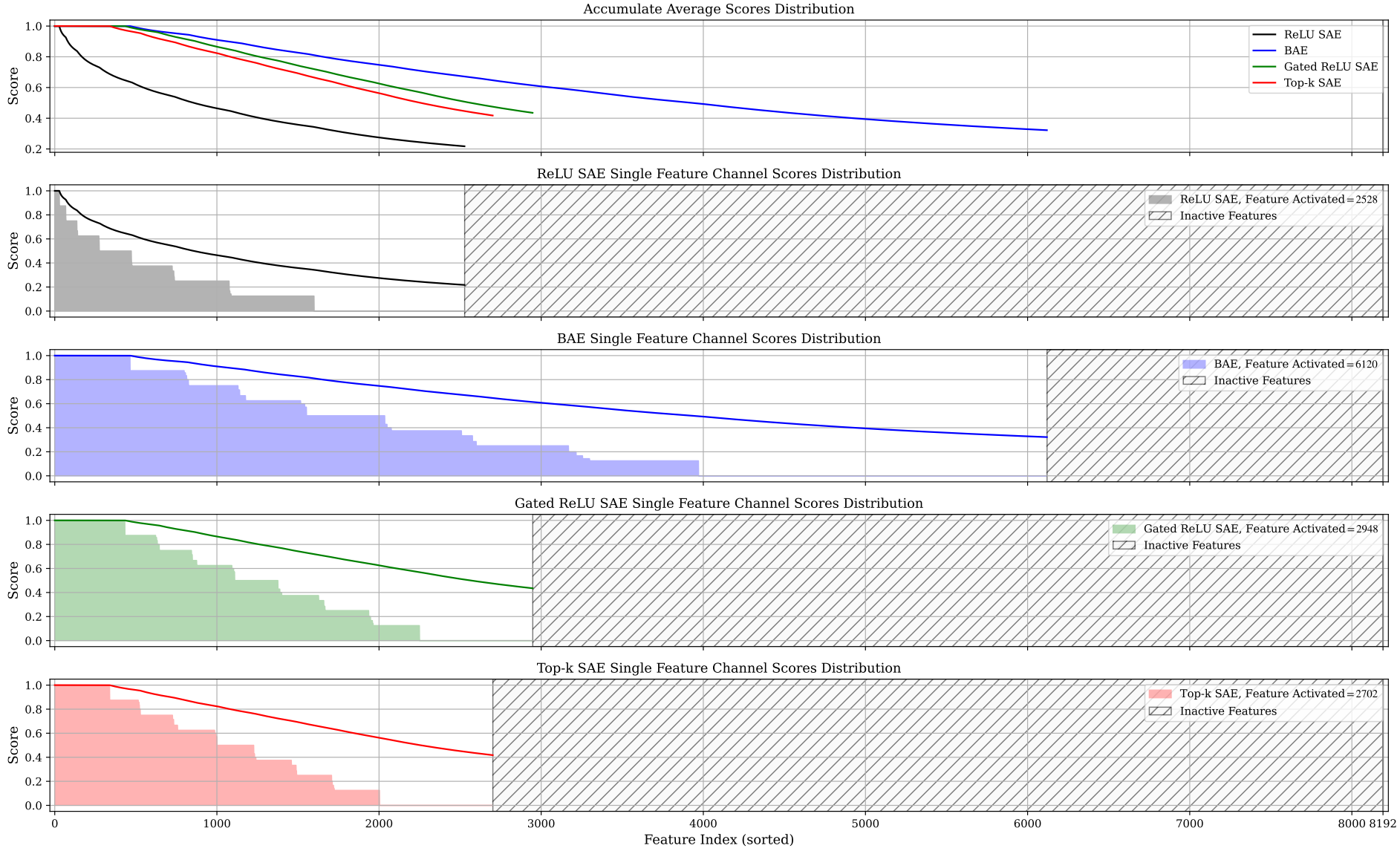

Figure 27: ComSem interpretability score distribution of each feature channel on layer 14 of Llama 3.2-1B.

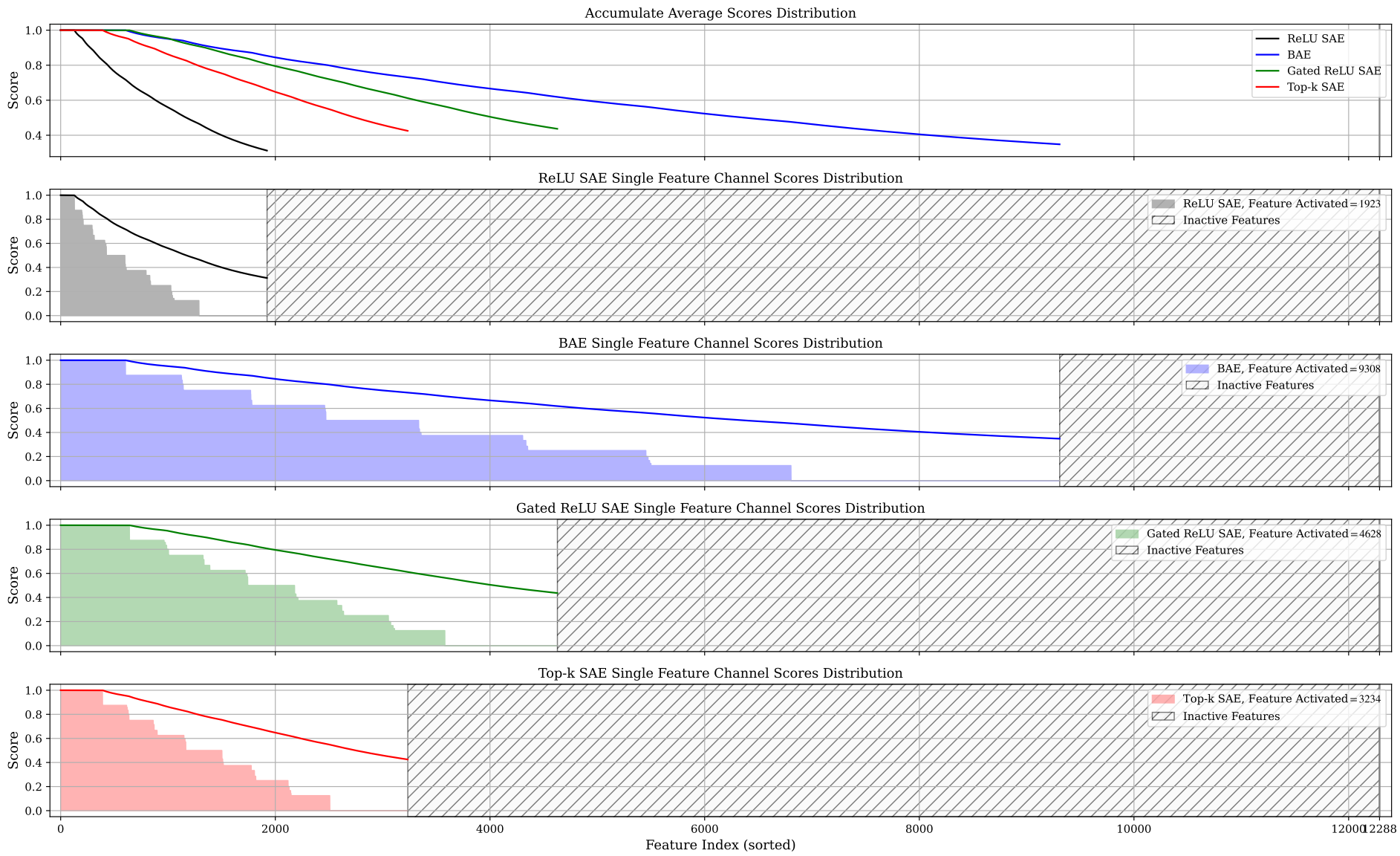

Figure 28: ComSem interpretability score distribution of each feature channel on layer 20 of Llama 3.2-3B.

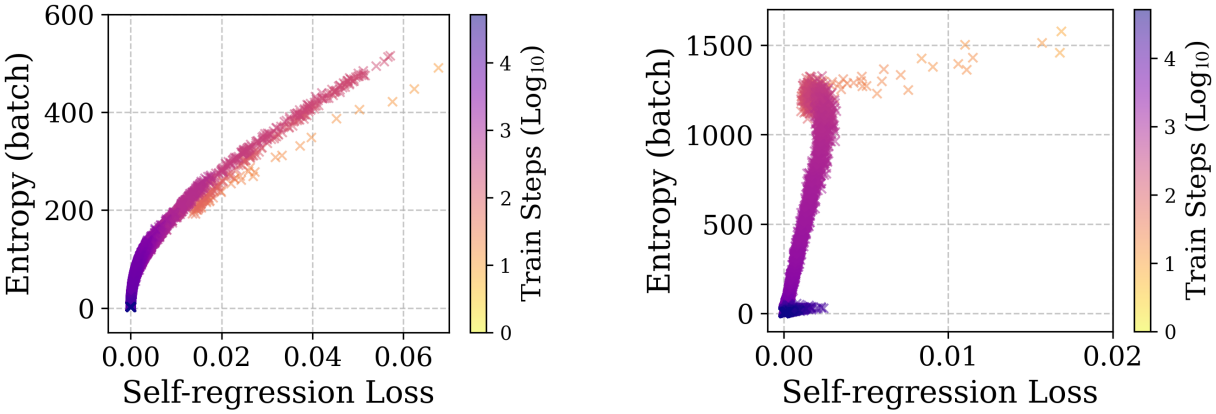

Figure 29: Augment result for Fig. 11 ($r = 2$, $d' = 2d$). (Left) with entropy objective, (right) without entropy objective.

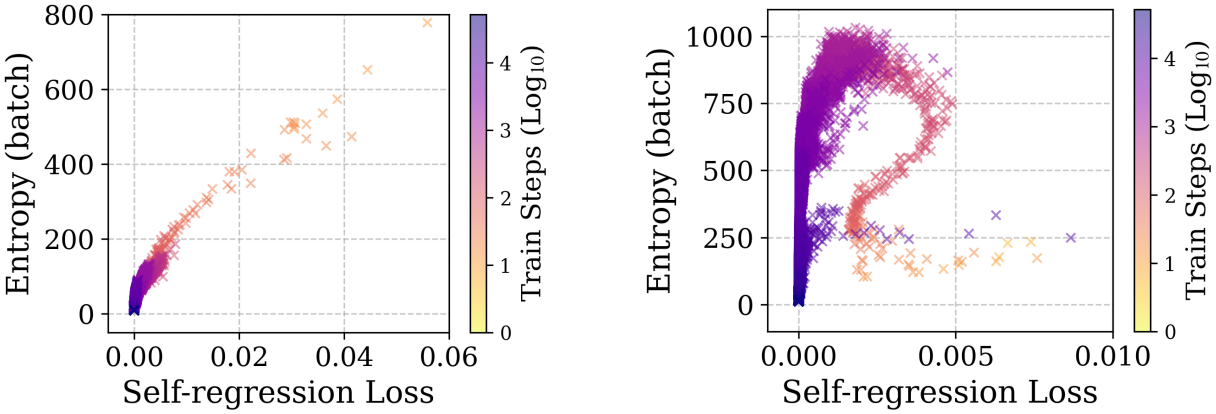

Figure 30: Augment result for Fig. 11 ($r = 8$, $d' = 2d$). (Left) with entropy objective, (right) without entropy objective.

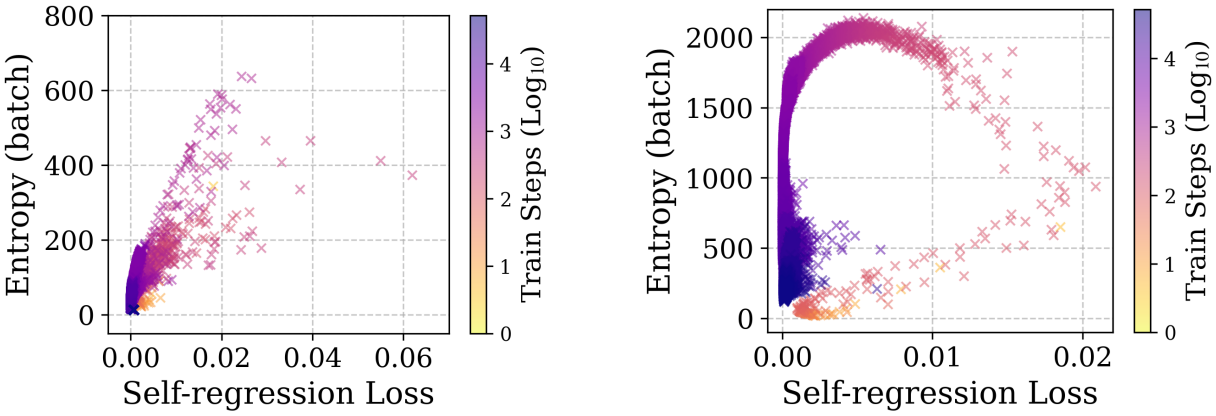

Figure 31: Augment result for Fig. 11 ($r = 16$, $d' = 4d$). (Left) with entropy objective, (right) without entropy objective.

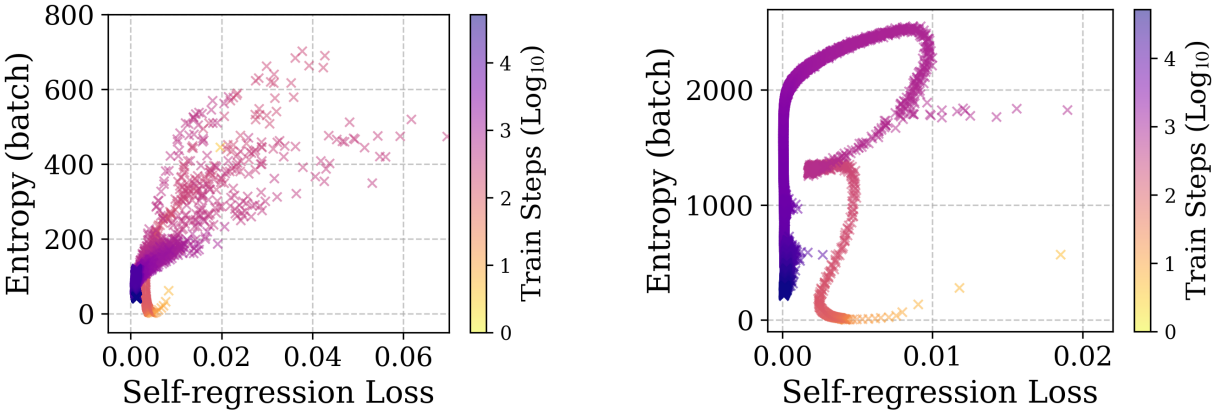

Figure 32: Augment result for Fig. 11 ($r = 32$, $d' = 4d$). (Left) with entropy objective, (right) without entropy objective.

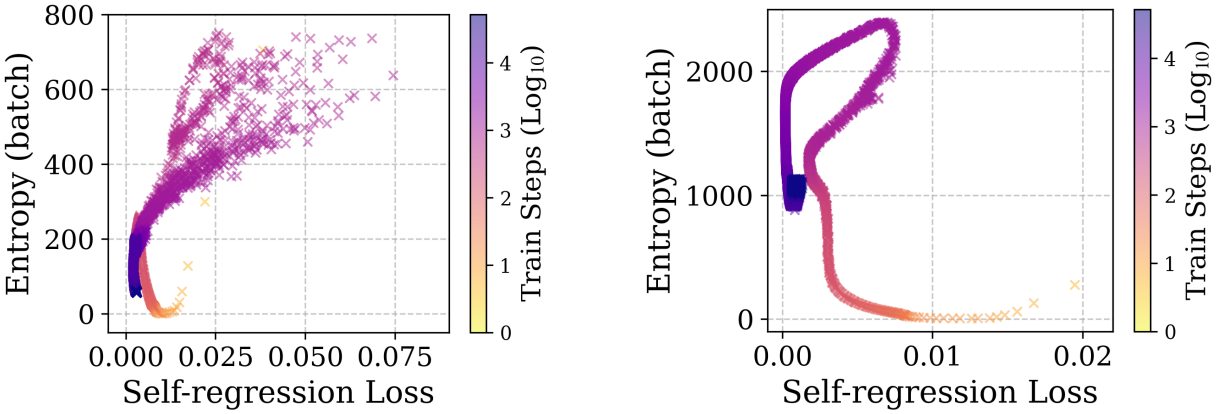

Figure 33: Augment result for Fig. 11 ($r = 64$, $d' = 4d$). (Left) with entropy objective, (right) without entropy objective.

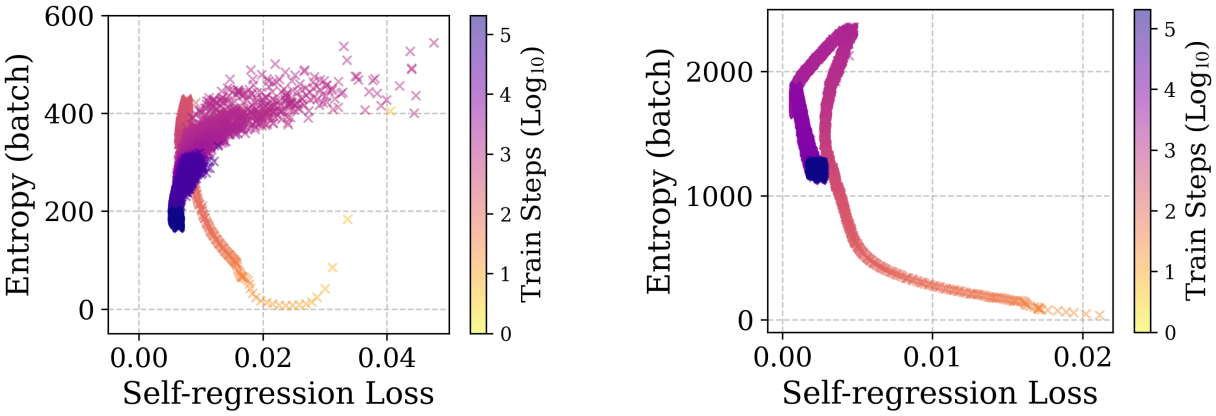

Figure 34: Augment result for Fig. 11 ($r = 128$, $d' = 4d$). (Left) with entropy objective, (right) without entropy objective.

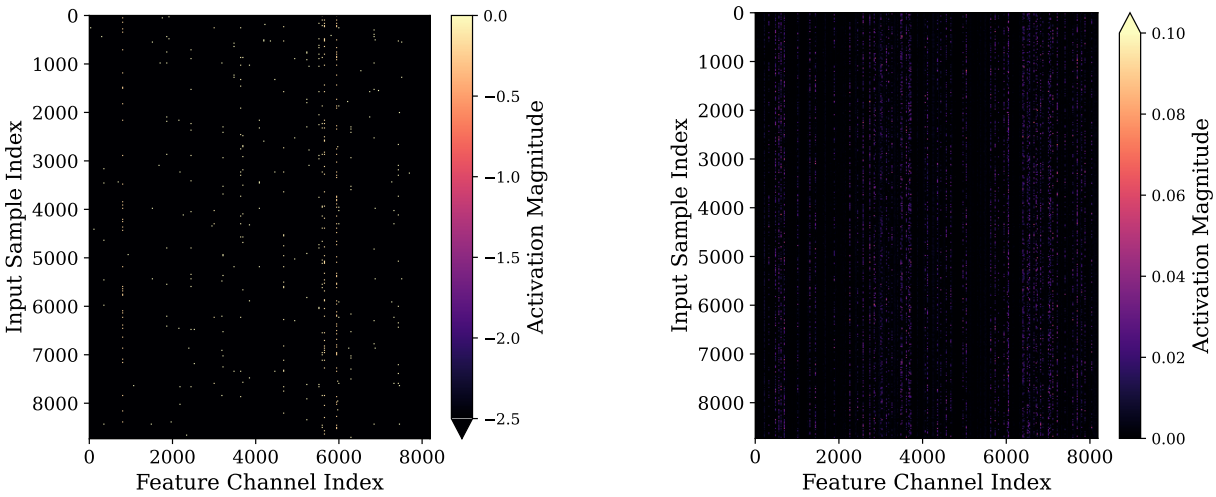

Figure 35: Activation magnitude visualization of (left) BAE and (right) SAE of layer 5.

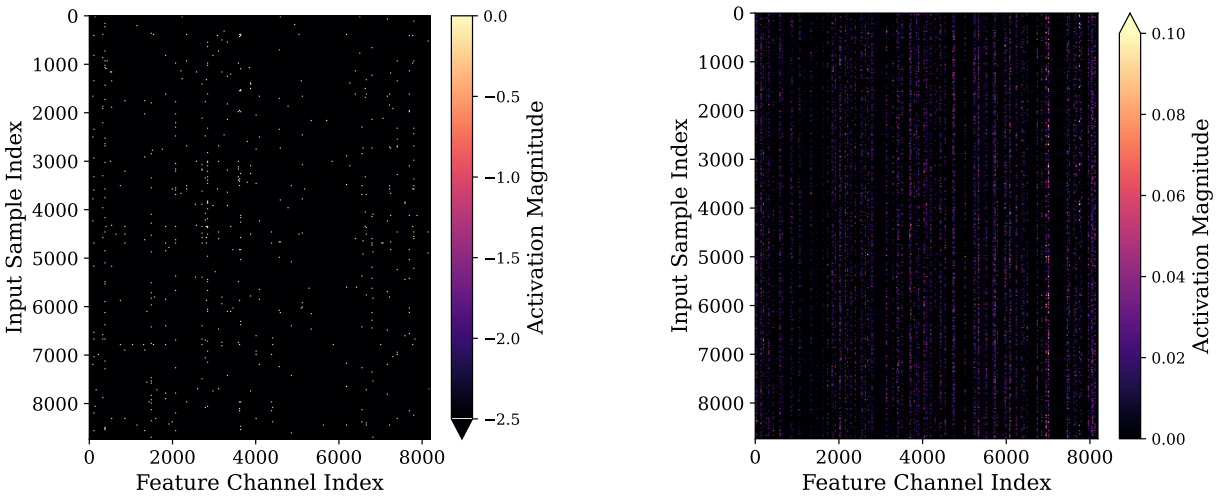

Figure 36: Activation magnitude visualization of (left) BAE and (right) SAE of layer 9.

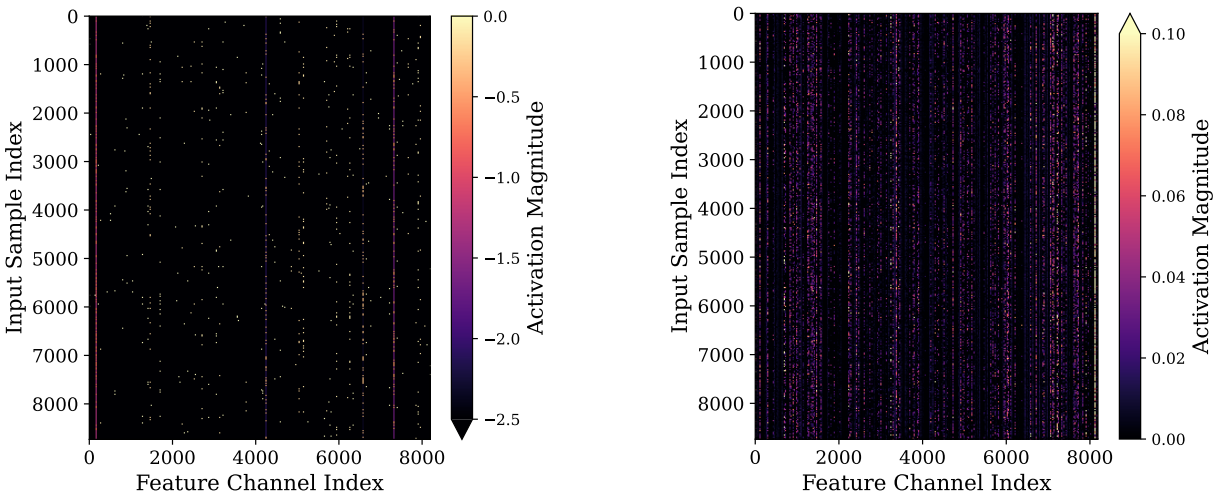

Figure 37: Activation magnitude visualization of (left) BAE and (right) SAE of layer 11.

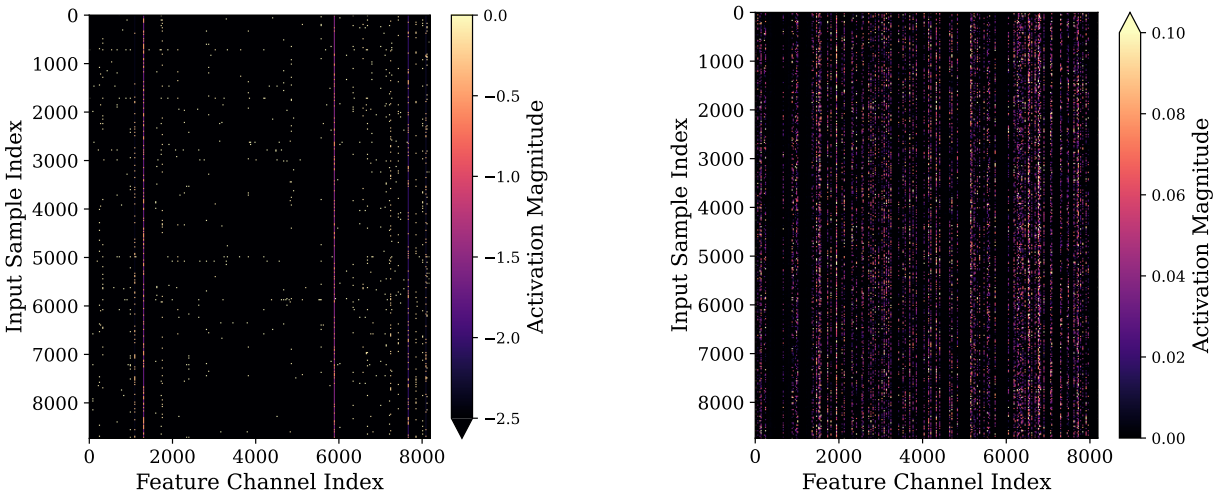

Figure 38: Activation magnitude visualization of (left) BAE and (right) SAE of layer 14.

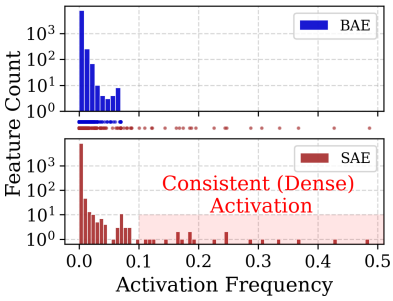

Figure 39: Augment for Fig. 5 on Layer 5 of Llama 3.2-1B.

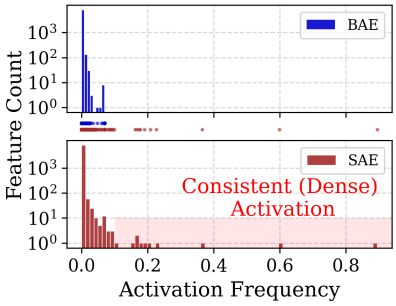

Figure 40: Augment for Fig. 5 on Layer 9 of Llama 3.2-1B.

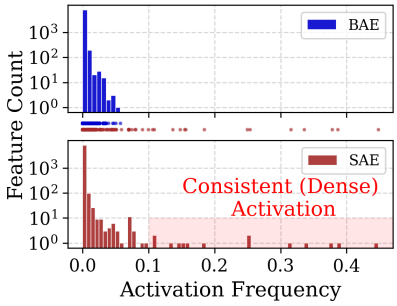

Figure 41: Augment for Fig. 5 on Layer 14 of Llama 3.2-1B.

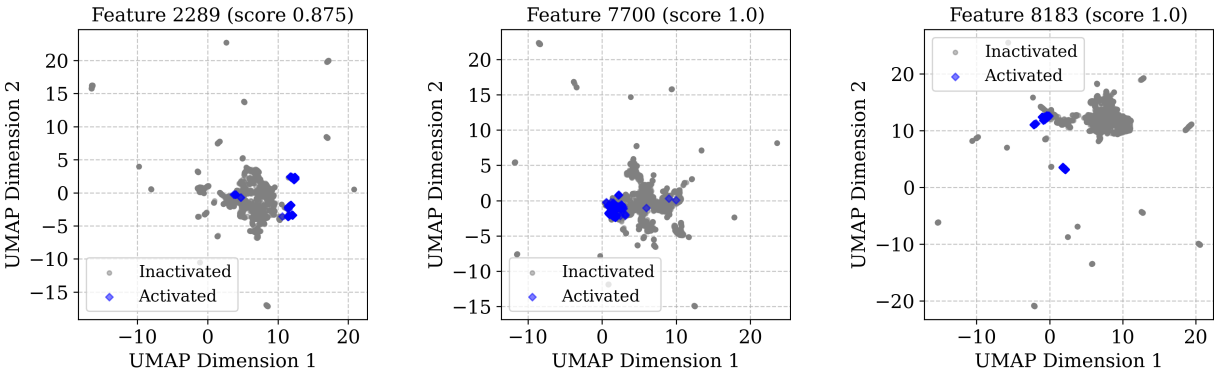

Figure 42: UMAP visualization for 3 high-score features with corresponding activated/inactivated hidden states on Layer 14 of Llama 3.2-1B.

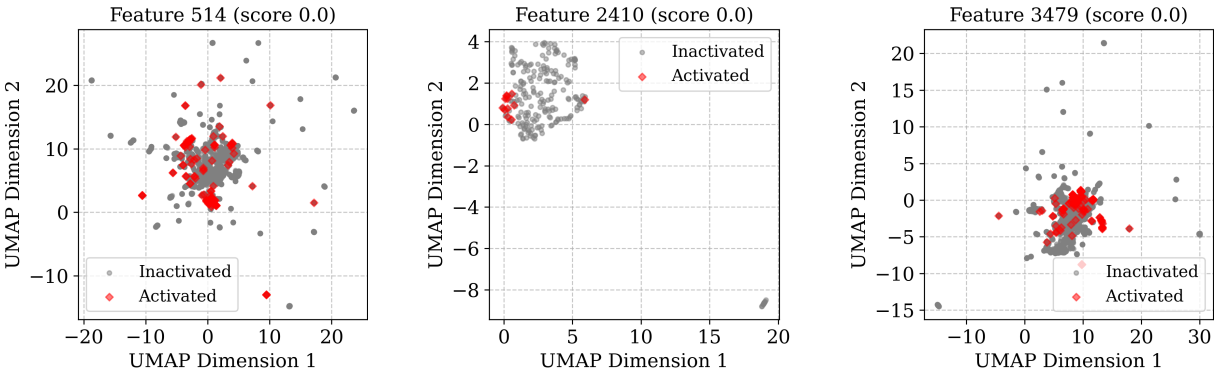

Figure 43: UMAP visualization for 3 low-score features with corresponding activated/inactivated hidden states on Layer 14 of Llama 3.2-1B.