UNPACKING SDXL TURBO: INTERPRETING TEXT-TO-IMAGE MODELS WITH SPARSE AUTOENCODERS

Viacheslav Surkov* **Chris Wendler** **Mikhail Terekhov** **Justin Deschenaux**

Robert West **Caglar Gulcehre**
School of Computer and Communication Sciences
EPFL
Lausanne, Switzerland

ABSTRACT

Sparse autoencoders (SAEs) have become a core ingredient in the reverse engineering of large-language models (LLMs). For LLMs, they have been shown to decompose intermediate representations that often are not interpretable directly into sparse sums of interpretable features, facilitating better control and subsequent analysis. However, similar analyses and approaches have been lacking for text-to-image models. We investigated the possibility of using SAEs to learn interpretable features for a few-step text-to-image diffusion models, such as SDXL Turbo. To this end, we train SAEs on the updates performed by transformer blocks within SDXL Turbo's denoising U-net. We find that their learned features are interpretable, causally influence the generation process, and reveal specialization among the blocks. In particular, we find one block that deals mainly with image composition, one that is mainly responsible for adding local details, and one for color, illumination, and style. Therefore, our work is an important first step towards better understanding the internals of generative text-to-image models like SDXL Turbo and showcases the potential of features learned by SAEs for the visual domain.

Code is available at `https://github.com/surkovv/sdxl-unbox`. [1]

1 INTRODUCTION

Text-to-image generation is a rapidly evolving field. The DALL-E model first captured public interest (Ramesh et al., 2021), combining learned visual vocabularies with sequence modeling to produce high-quality images based on user input prompts. Today's best text-to-image models are largely based on text-conditioned diffusion models (Rombach et al., 2022; Saharia et al., 2022b; Podell et al., 2023; Sauer et al., 2023b; Betker et al., 2023; Pernias et al., 2023). This can be partially attributed to the stable training dynamics of diffusion models, which makes them easier to scale than previous approaches such as generative adversarial neural networks (Dhariwal & Nichol, 2021). As a result, they can be trained on internet scale image-text datasets like LAION-5B (Schuhmann et al., 2022a) and learn to generate photorealistic images from text.

However, the underlying logic of the neural networks that enable the text-to-image pipelines we have today, due to their black-box nature, is not well understood. Unfortunately, this lack of interpretability is typical in the deep learning field. For example, advances in image recognition (Krizhevsky et al., 2012) and language modeling (Devlin, 2018; Brown, 2020) come mainly from scaling models (Hoffmann et al., 2022), rather than from an improved understanding of their internals. Recently, the emerging field of mechanistic interpretability has sought to alleviate this limitation by reverse engineering visual models (Olah et al., 2020) and transformer-based LLMs (Rai et al., 2024). At the same time, diffusion models have remained under-explored.

*Correspondence to viacheslav.surkov@epfl.ch.

[1]The code base contains scripts to collect training data and train SAEs. Additionally, it allows to browse pre-trained SAE features with a demo application.

This work focuses on SDXL Turbo, a recent open-source few-step text-to-image diffusion model. We import from a toolbox originally developed for language models, which allows inspection of the intermediate results of the forward pass (Chen et al., 2024; Ghandeharioun et al., 2024; Cunningham et al., 2023; Bricken et al., 2023). Moreover, some even enable reverse engineering of the entire task-specific subnets (Marks et al., 2024). In particular, *sparse autoencoders (SAEs)* (Yun et al., 2021; Cunningham et al., 2023; Bricken et al., 2023) are considered a breakthrough in interpretability for LLMs. They have been shown to decompose intermediate representations of the LLM forward pass – often difficult to interpret due to *polysemanticity*[2] – into sparse sums of interpretable and monosemantic features. These features are learned in an unsupervised way, can be automatically annotated using LLMs (Caden et al., 2024), and facilitate subsequent analysis, for example, circuit extraction (Marks et al., 2024).

Contributions. In this work, we investigate whether we can use SAEs to draw insights about the computation performed by the one-step generation process of SDXL Turbo, which is a recent open-source few-step text-to-image diffusion model.

To facilitate our analysis, we developed a library called *SDLens* that allows us to cache and manipulate intermediate results of SDXL Turbo's forward pass. We use our library to create a dataset of SDXL Turbo's intermediate feature maps of several transformer blocks inside SDXL Turbo's U-net on 1.5M LAION-COCO prompts (Schuhmann et al., 2022a;b). We then use these feature maps to train multiple SAEs for each transformer block. Finally, we perform a quantitative and qualitative analysis of the SAEs' learned features:

1. We empirically show the potential of SAEs to learn highly interpretable features in diffusion-based text-to-image models.
2. We develop visualization techniques to analyze the interpretability and causal effects of the learned features.
3. We perform two case studies in which we visualize and interpret the active features in different transformer blocks, finding evidence that certain transformer blocks of SDXL Turbo's pipeline specialize in *image composition*, *adding details*, and *style*.[3]
4. We follow up our qualitative case studies by designing multiple quantitative experiments showing that our hypotheses also hold up on larger sample sizes.
5. As part of our quantitative analysis, we create an automatic feature annotation pipeline for the transformer block, which appears responsible for *image compositions*.

Thus, we show that SAEs learn interpretable features that causally affect SDXL Turbo's image generation process. Importantly, the learned features provide insight into the computational details of SDXL Turbo's forward pass, such as the different roles of the investigated transformer blocks. By open-sourcing our library and SAEs, we lay the foundation for further research in this area.

Note on visualizations. Qualitative analysis by inspection of generated images is crucial for this type of research. Many essential visualizations are included in App. D and the supplementary material[4] to improve the readability of the main text.

2 Background

2.1 Sparse autoencoders

Let $h(x) \in \mathbb{R}^d$ be an intermediate result during a forward pass of a neural network on input x. In a fully connected neural network, $h(x)$ could correspond to a vector of neuron activations. In transformers, which are neural network architectures that combine attentions with fully connected layers and residual connections, $h(x)$ could either refer to the content of the residual stream after a layer, an update to the residual stream by a layer, or a vector of neuron activations within a fully connected block.

[2] A phenomenon where a single neuron or feature encodes multiple, unrelated concepts (Elhage et al., 2022)

[3] The blocks `down.2.1` and `up.0.1` have been already known in the community as "composition" and "style" blocks (Spinelli, 2024). However, in this paper we provide the first thorough and fine-grained investigation of these blocks.

[4] The supplementary material is hosted here `https://drive.google.com/file/d/1MMvQyj5AcQOBhbW0P3zLFldYSDnMZ71d/view?usp=sharing`.

It has been shown (Yun et al., 2021; Cunningham et al., 2023; Bricken et al., 2023) that in many neural networks, especially LLMs, intermediate representations can be well approximated by sparse sums of $n_f \in \mathbb{N}$ learned feature vectors, i.e.,

$$h(x) \approx \sum_{\rho=1}^{n_f} s_\rho(x)\mathbf{f}_\rho, \tag{1}$$

where $s_\rho(x)$ are the input-dependent coefficients, most of which are equal to zero and $\mathbf{f}_1, \ldots, \mathbf{f}_{n_f} \in \mathbb{R}^d$ is a learned dictionary of feature vectors. Importantly, the features are usually interpretable.

Sparse autoencoders. To implement the sparse decomposition from equation 1, the vector s containing the n_f coefficients of the sparse sum, is parameterized by a single linear layer followed by ReLU activations, called the *encoder*,

$$s = \texttt{ENC}(h) = \sigma(W^{\texttt{ENC}}(h - b_{\text{pre}}) + b_{\text{act}}), \tag{2}$$

in which $h \in \mathbb{R}^d$ is the latent that we aim to decompose, $\sigma(\cdot) = \max(0, \cdot)$, $W^{\texttt{ENC}} \in \mathbb{R}^{n_f \times d}$ is a learnable weight matrix and b_{pre} and b_{act} are learnable bias terms. We omitted the dependencies $h = h(x)$ and $s = s(h)$, which are clear from the context.

Similarly, the learnable features are parametrized by a single linear layer called *decoder*,

$$h' = \texttt{DEC}(s) = W^{\texttt{DEC}}s + b_{\text{pre}}, \tag{3}$$

in which $W^{\texttt{DEC}} = (\mathbf{f}_1|\cdots|\mathbf{f}_{n_f}) \in \mathbb{R}^{d \times n_f}$ is a learnable matrix. Its columns take the role of learnable features and b_{pre} is a learnable bias term. [5]

2.2 Few step diffusion models: SDXL Turbo

Diffusion models. Diffusion models (Sohl-Dickstein et al., 2015; Ramesh et al., 2022; Rombach et al., 2022; Saharia et al., 2022a) sample from an unknown distribution p by learning to iteratively denoise corrupted samples, starting from pure noise. The corruption process is defined on training samples from p. Mathematically, the images are corrupted with Gaussian noise and are distributed according to

$$q_t(x_t|x_0) := \mathcal{N}(\alpha_t x_0, \sigma_t^2 \mathbf{I}), \tag{4}$$

where x_0 corresponds to a real image from p, $0 \leq t \leq T$, α_t, σ_t^2 are positive real-valued scalars such that the signal-to-noise ratio $SNR := \frac{\alpha_t}{\sigma_t^2}$ is monotonically decreasing, and, $\mathbf{I}$ is the identity matrix. Additionally, the coefficients $\alpha_{T-1}, \sigma_{T-1}^2$ are typically chosen such that $x_T \sim \mathcal{N}(0, \mathbf{I})$.

The denoising process is implemented through a learned distribution $p_\theta(x_{t-1}|x_t)$. The simplest way to generate samples using $p_\theta(x_{t-1}|x_t)$ is to first generate a sample of pure noise $x_T \sim \mathcal{N}(0, \mathbf{I})$, followed by T iterative applications of p_θ, which yields a sequence $x_T, x_{T-1}, ..., x_1, x_0$, where x_0 approximates a sample from p. The vector θ represents the parameters of the neural network that defines $p_\theta(x_{t-1}|x_t)$. The denoising distribution $p_\theta(x_{t-1}|x_t)$ is parameterized to be Gaussian and a neural network is trained to optimize the parameters of this distribution.

Latent diffusion. Originally, diffusion models operated directly on pixels (Ho et al., 2020; Song & Ermon, 2020). However, training a denoising network in pixel space is computationally expensive (Hoogeboom et al., 2023). Thus, Rombach et al. (2022) use a pre-trained variational autoencoder to first compress images into latent representations and define a diffusion process in the latent space of the variational autoencoder instead. To make this difference clear, they write $p_\theta(z_{t-1}|z_t)$, in which now z_t refers to a noisy latent instead of a noisy image.

SDXL Turbo. To speed-up inference of latent diffusion models, Sauer et al. (2023b) distill a pre-trained model called Stable Diffusion XL (SDXL) (Podell et al., 2023). The distilled model is referred to as *SDXL Turbo* because it allows high-quality sampling in as little as 1-4 steps. In comparison, the original SDXL model is trained with a noise schedule of 1000 steps, but in practice, sampling with 20 to 50 steps still generates high-quality images.

[5] An extended version of this section, including training details, is in App. F.

Neural network architecture. The denoising network of *SDXL Turbo* estimating $p_\theta(z_{t-1}|z_t)$ is implemented using a U-net similar to Rombach et al. (2022). The U-net is composed of a down-sampling path, a bottleneck, and an up-sampling path. Both the down-sampling and up-sampling paths are composed of 3 individual blocks. The individual block structure differs slightly, but both down- and up-sampling blocks consist of residual layers, with some blocks including cross-attention transformer layers while others do not. Finally, the bottleneck layer is also composed of attention and residual layers. Importantly, the text conditioning is achieved via cross-attention to text embeddings performed by 11 transformer blocks embedded in the down-, up-sampling path, and bottleneck. An architecture diagram displaying the relevant blocks can be found in App. G Fig. 4.

3 SPARSE AUTOENCODERS FOR SDXL TURBO

With the necessary definitions at hand, in this section we show a way to apply SAEs to SDXL Turbo. In the following, we assume that all SDXL Turbo generations are done using a 1-step process.

Where to apply the SAEs. We apply SAEs to updates performed within the cross-attention transformer blocks responsible for incorporating the text prompt (depicted in App. A Fig. 4). Each of these blocks consists of multiple transformer layers, which attend to all spatial locations (self-attention) and to the text prompt embeddings (cross-attention).

Formally, the ℓth cross-attention transformer block updates its inputs in the following way

$$D[\ell]^{out}_{ij} = D[\ell]^{in}_{ij} + \texttt{TRANSFORMER}[\ell](D[\ell]^{in}, c)_{ij}, \tag{5}$$

in which $D[\ell]^{in}, D[\ell]^{out} \in \mathbb{R}^{h\times w\times d}$ denote the residual stream before and after application of the ℓ-th cross-attention transformer block respectively. The transformer block itself calculates the function $\texttt{TRANSFORMER}[\ell] : \mathbb{R}^{h\times w\times d} \to \mathbb{R}^{h\times w\times d}$. Note that we omitted the dependence on input noise z_t and text embedding c for both $D[\ell]^{in}(z_t, c)$ and $D[\ell]^{out}(z_t, c)$.

We train SAEs on the residual updates $\texttt{TRANSFORMER}[\ell](D[\ell]^{in}, c)_{ij} \in \mathbb{R}^d$ denoted by

$$\Delta D[\ell]_{ij} := \texttt{TRANSFORMER}[\ell](D[\ell]^{in}, c)_{ij} = D[\ell]^{out}_{ij} - D[\ell]^{in}_{ij}. \tag{6}$$

That is, we jointly train one encoder $\texttt{ENC}[\ell]$ and decoder $\texttt{DEC}[\ell]$ pair per transformer block ℓ and share it over all spatial locations i, j. For notational convenience we omit block indices from now. We do this for the 4 (out of 11) transformer blocks (App. G Fig. 4) that we found have the highest impact on the generation (see App. A), namely, `down.2.1`, `mid.0`, `up.0.0` and `up.0.1`.

Feature maps. We refer to $\Delta D \in \mathbb{R}^{h\times w\times d}$ as dense feature map and applying ENC to all image locations results in the *sparse feature map* $S \in \mathbb{R}^{h\times w\times n_f}$ with entries

$$S_{ij} = \texttt{ENC}(\Delta D_{ij}). \tag{7}$$

We refer to the feature map of the ρth learned feature using $S^\rho \in \mathbb{R}^{h\times w}$. This feature map S^ρ contains the spatial activations of the ρth learned feature. Its associated feature vector $\mathbf{f}_\rho \in \mathbb{R}^d$ is a column in the decoder matrix $W^{\texttt{DEC}} = (\mathbf{f}_1|\cdots|\mathbf{f}_{n_f}) \in \mathbb{R}^{d\times n_f}$. Using this notation, we can represent each element of the dense feature map as a sparse sum

$$\Delta D_{ij} \approx \sum_{\rho=1}^{n_f} S^\rho_{ij}\mathbf{f}_\rho, \text{ with } S^\rho_{ij} = 0 \text{ for most } \rho \in \{1,\dots,n_f\}. \tag{8}$$

Training. In order to train an SAE for a transformer block, we collected dense feature maps ΔD_{ij} from SDXL Turbo one-step generations on 1.5M prompts from the LAION-COCO (Schuhmann et al., 2022b). Each feature map has dimensions of 16×16, resulting in a training dataset of 384M dense feature vectors per transformer block. For the SAE training process, we followed the methodology described in (Gao et al., 2024), using the TopK activation function and an auxiliary loss to handle dead features. For more details on the SAE training, see App. F and for training metrics see App. B.

4 QUALITATIVE ANALYSIS OF THE TRANSFORMER BLOCKS

We perform a visual qualitative analysis to gain deeper insight into the behavior and characteristics of the learned features across transformer blocks. First, we introduce feature visualization techniques and then use them to conduct two case studies. For the sake of simplicity in the notation, we omit the transformer block index ℓ.

4.1 FEATURE VISUALIZATION TECHNIQUES

We start by introducing necessary notation and formally describing the methods used for feature visualization.

Spatial activations. We visualize a sparse feature map $S^\rho \in \mathbb{R}^{h \times w}$ containing activations of a feature ρ across the spatial locations by up-scaling it to the size of the generated images and overlaying it as a heatmap over the generated images. In the heatmap, red indicates the highest activation of features, and blue represents the lowest non-zero activation.

Top dataset examples. For a given feature ρ, we sort dataset examples according to their average spatial activation

$$a_\rho = \frac{1}{wh} \sum_{i=1}^{h} \sum_{j=1}^{w} S_{ij}^\rho \in \mathbb{R}. \tag{9}$$

We use equation 9 to define the top dataset examples and to sample from the top 5% quantile of the activating examples ($a_\rho > 0$). We will refer to them as top 5% images for a feature ρ.

Note that S_{ij}^ρ always depends on an embedding of the input prompt c and input noise z_1, via $S_{ij}(c, z_1) = \texttt{ENC}(\Delta D_{ij}(c, z_1))$, which we usually omit for ease of notation. As a result, a_ρ also depends on c and z_1. When we refer to the top dataset examples, we mean our (c, z_1) pairs with the largest values for $a_\rho(c, z_1)$.

Activation modulation. We design interventions that allow us to modulate the strength of the ρth feature. Specifically, we achieve this by adding or subtracting a multiple of the feature ρ on all of the spatial locations i, j proportional to its original activation S_{ij}^ρ

$$\Delta D'_{ij} = \Delta D_{ij} + \beta S_{ij}^\rho \mathbf{f}_\rho, \tag{10}$$

in which ΔD_{ij} is the update performed by the transformer block before and $\Delta D'_{ij}$ after the intervention, $\beta \in \mathbb{R}$ is a modulation factor, and $\mathbf{f}_\rho$ is the ρth learned feature vector. In the following, we will refer to this intervention as *activation modulation intervention*.

Activation on empty context. Another way of visualizing the causal effect of features is to activate them while doing a forward pass on the empty prompt $c(\text{“”})$. To do so, we turn off all other features at the transformer block ℓ of intervention and turn on the target feature ρ. Formally, we modify the forward pass by setting

$$D_{ij}^{out'} = D_{ij}^{in} + \gamma k \mu_\rho \mathbf{f}_\rho, \tag{11}$$

in which $D_{ij}^{out'}$ replaces residual stream plus transformer block update, D_{ij}^{in} is the input to the block, $\mathbf{f}_\rho$ is the ρth learned feature vector, $\gamma \in \mathbb{R}$ is a hyperparameter to adjust the intervention strength, and μ_ρ is a feature-dependent multiplier obtained by taking the average activation across positive activations of ρ (collected over a subset of 50.000 dataset examples). Multiplying it by k aims to recover the coefficients lost by setting the other features to zero. Further in the text, we will refer to this intervention as *empty-prompt intervention*, and the images generated using this method with γ set to 1, as *empty-prompt intervention images*.

Note that we directly added/subtracted feature vectors to the dense vectors for both intervention types instead of encoding, manipulating sparse features, and decoding. This approach helps mitigate side effects caused due to reconstruction loss (see App. B).

(a) Top 5 features of `down.2.1`

(b) Top 5 features `up.0.1`

(c) Top 5 features of `up.0.0`

(d) Top 5 features of `mid.0`

Figure 1: The top 5 features of `down.2.1` (a), `up.0.1` (b), `up.0.0` (c) and `mid.0` (d) for the prompt: "A cinematic shot of a professor sloth wearing a tuxedo at a BBQ party." Each row represents a feature. The first column depicts a feature heatmap (highest activation red and lowest nonzero one blue). The columns with titles containing "A" show feature modulation interventions, those containing "B" the intervention of turning on the feature on the empty prompt, and the ones containing "C" depict top dataset examples. Floating point values in the title denote β and γ values.

4.2 Case study I: Most active features on a prompt

Combining the feature visualization techniques in Fig. 1, we depict the features with the highest average activation when processing the prompt: "A cinematic shot of a professor sloth wearing a tuxedo at a BBQ party". We present analysis of the transformer blocks in order of decreasing interpretability. An extended version of this case study demonstrating top 9 features per transfromer block instead of 5, is available in App. C Fig. 5.

Down.2.1. The `down.2.1` transformer block appears to contribute to the image composition. Several features relate to the prompt: 4539 "sloth", 4751 "a tuxedo", 2881 "party".

Activation modulation interventions with negative β (A. -6.0 columns) result in removing or changing scene objects in ways that align with the heatmap (hmap column) and the top examples (C columns): 1674 *removes* the light chains in the back, 4608 the umbrellas/tents, 4539 the 3D animation-like sloth face, and, 4751 *changes* the type of suit. Similarly, enhancing the same features (A. 6.0 column) makes the corresponding elements more visible and distinct.

Notably, activating the features on the empty prompt often creates meaningful images with related elements (B. column). For reference, with the fixed random seed we use, the empty prompt generation without interventions resembles a painting of a piece of nature with a lot of green and brown tones.

While top dataset examples (C.0, C.1) and empty prompt intervention (B.) mostly agree with the feature activation heatmaps (hmap column), some of them provide additional insights, e.g., 2881, which activates on the suit, seems to correspond to (masqueraded) characters in a (festive) scene.

Up.0.1. Based, on our observations, the features of `up.0.1` appear to contribute to the style.

Interestingly, turning on the `up.0.1` features on the empty prompt (B. column) results in texture-like images. Furthermore, when activating them locally (A. columns), their contribution to the output is highly localized, and most of inactive image area remains unchanged. For the `up.0.1` we find it remarkable that often the features' ablations and amplifications are counterparts: 500 (light, shadow), 2727 (shadow, light), 3936 (blue, orange).

Up.0.0. For the third transformer block, `up.0.0`, we observe that most top dataset examples and their activations (C columns) are quite interpretable: 3603 corresponds to party decorations, 5005 to the upper part of a tent, 775 to buttons on a suit, 153 to the lower animal jaw, 1550 to collars. All the features exhibit an expected causal effect on the generation when ablated or enhanced (A. columns).

The activation regions of the features often are very concentrated. Similarly to `up.0.1`, activation modulation interventions leave inactive image regions mostly unaffected. For the empty prompt, activating these features produces abstract-looking images that are hard to relate to the other columns. Thus, we excluded this visualization technique and instead added one more dataset example. In summary, the learned features of this transformer block primarily add local details to the generation and, importantly, they are effective only within a suitable context.

Mid.0. The specific role of the fourth block (`mid.0`) is not well understood. We find it more difficult to interpret because most interventions in the `mid.0` block have subtle effects. We did not include empty-prompt intervention results because they barely affect the generation.

Despite these subtle effects, dataset examples (C. columns) and heatmaps (hmap column) mostly agree with each other and are specific enough to be interpretable: 4755 activates on bottom right part of faces, 4235 on left part of (animal) faces, 1388 on people in the background, and, 5102 on outlines the left border of the main object in the scene. We hypothesize that `mid.0`'s features are more abstract, potentially encoding spatial location[6] and relations between objects.

4.3 Case study II: Random features

In this case study, we explore the learned features independently of any specific prompt. In App. D Fig. 6 and Fig. 7, we demonstrate the first 5 and last 5 learned features for each transformer block. In addition, we provide similar visualisations for the first 100 features of each layer in the supplementary material. As SAEs are randomly initialized before the training process, these sets can be considered as random samples of features. Each feature visualization consists of 3 images of top 5% images for this feature, and their perturbations with activation modulation interventions. For `down.2.1` and `up.0.1`, we also include the empty-prompt intervention images. Additionally, we provide visualizations of several selected features in App. D Fig. 8 and demonstrate the effects of their forced activation on unrelated prompts in App. D Fig. 9.

Overall, our insights gained from Case Study I (Sec. 4.2) appear to generalize on random feature samples. In particular, this suggests that a significant portion of the learned features are interpretable. Additionally, when studying features in isolation, it becomes apparent that distinctions between the blocks are blurred. For example, some `down.2.1` features correspond to elements of style, e.g., a *anime style* and a *cartoon style* feature in App. D Fig. 8. Likewise, some `up.0.1` features don't only change style but also add and remove elements of the scene, e.g., dog eyes (App. D Fig. 6 `up.0.1` feature 3).

5 Quantitative Evaluation of the learned features

In this section, we follow up on qualitative insights by collecting quantitative evidence.

[6] SDXL Turbo does not utilize positional encodings for the spatial locations in the feature maps. Therefore, we did a brief sanity check and trained linear probes to detect i, j given D^{in}_{ij}. These probes achieved high accuracy on a holdout set: $97.9\%, 98.48\%, 99.44\%, 95.57\%$ for `down.2.1`, `mid.0`, `up.0.0`, `up.0.1`.

5.1 ANNOTATION PIPELINE

Feature annotation with an LLM followed by further evaluation is a common way to assess feature properties such as specificity, sensitivity, and causality (Caden et al., 2024). We found it applicable to the features learned by the `down.2.1` transformer block, which have a strong effect on the generation. Thus, they are amendable to automatic annotation using visual language models (VLMs) such as GPT-4o (OpenAI, 2024). In contrast, for the features of other blocks with more subtle effects, we found VLM-generated captions to be unsatisfactory. In order to caption the features of `down.2.1`, we prompt GPT-4o with a sequence of 14 images. The first five images are irrelevant to the feature (i.e., the feature was inactive during the generation of the images), followed by a progression of 4 images with increasing average activation values, and finished by five images with the highest average activation values. The last nine images are provided alongside their so-called "coldmaps": a version of an image with weakly active and inactive regions being faded and concealed. The prompt template and examples of the captions can be found in App. E.

5.2 EXPERIMENTAL DETAILS

We perform a series of experiments in order to get statistical insights into the features learned. We will report the majority of the experimental scores in the format $M(S)$. When the score is reported in the context of a SDXL Turbo transformer block, it means that we computed the score for each feature of the block and set M and S to mean and standard deviation across the feature scores. For the baselines, we calculate the mean and standard deviation across the scores of a 100-element sample.

Table 1: Metrics for SDXL Turbo blocks and baselines.

(a) Specificity, texture score, and color activation for different blocks and baselines.

Block	Specificity	Texture	Color
`Down.2.1`	0.71 (0.11)	0.16 (0.02)	86.2 (14.9)
`Mid`	0.62 (0.11)	0.14 (0.01)	84.7 (16.3)
`Up.0.0`	0.66 (0.12)	0.18 (0.03)	86.3 (16.5)
`Up.0.1`	0.65 (0.11)	0.20 (0.02)	73.8 (20.6)
`Random`	0.50 (0.10)	0.13 (0.02)	90.7 (54.9)
`Same Prompt`	0.89 (0.06)	–	–
`Textures`	–	0.18 (0.02)	–

(b) Manhattan distances between original and intervened images at varying intervention strengths.

Block	-10	-5	5	10
`Down.2.1`	148.2 / 116.0	124.2 / 94.4	101.4 / 78.7	128.9 / 105.60
`Mid`	69.2 / 32.2	39.4 / 18.5	33.2 / 15.2	59.9 / 29.82
`Up.0.0`	105.3 / 38.4	77.7 / 23.7	63.6 / 23.3	88.6 / 37.08
`Up.0.1`	125.0 / 26.8	73.1 / 16.4	68.6 / 21.9	98.9 / 34.74

Interpretability. Features are usually considered interpretable if they are sufficiently specific, i.e., images exhibiting the feature share some commonality. In order to measure this property, we compute the similarity between images on which the feature is active. High similarity between these images is a proxy for high specificity. For each feature, we collect 10 random images among top 5% images for this feature and calculate their average pairwise CLIP similarity (Radford et al., 2021; Cherti et al., 2023). This value reflects how semantically similar the contexts are in which the feature is most active. We display the results in the first column of Table 1 (a), which shows that the CLIP similarity between images with the feature active is significantly higher then the random baseline (CLIP similarity between random images) for all transformer blocks. This suggests that the generated images share similarities when a feature is active.

For `down.2.1` we compute an additional *interpretability* score by comparing how well the generated annotations align with the top 5% images. The resulting CLIP similarity score is 0.21 (0.03) and significantly higher then the random baseline (average CLIP similarity with random images) 0.12 (0.02). To obtain an upper bound on this score we also compute the CLIP similarity to an image generated from the feature annotation, which is 0.25 (0.03).

Causality. We can use the feature annotations to measure a feature's causal strength by comparing the empty prompt intervention images with the caption.[7] The CLIP similarity between intervention images and feature caption is 0.19 (0.04) and almost matches the annotation-based interpretability score of 0.21 (0.03). This suggests that feature annotations effectively describe to the corresponding

[7]We require feature captions for the causality and sensitivity analyses, we only have them for `down.2.1`.

empty-prompt intervention images. Notably, the annotation pipeline did not use empty-prompt intervention images to generate captions. This fact speaks for the high causal strength of the features learned on `down.2.1`.

Sensitivity. A feature is considered sensitive when activated in its relevant context. As a proxy for the context, we have chosen the feature annotations obtained with the auto-annotation pipeline. For each learned feature, we collected the 100 prompts from a 1.5M sample of LAION-COCO with the highest sentence similarity based on sentence transformer embeddings of `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019). Next, we run SDXL Turbo on these prompts and count the proportion of generated images in which the feature is active on more than 0%, 10%, 30% of the image area, resulting in 0.60 (0.32), 0.40 (0.34), 0.27 (0.30) respectively, which is much higher than the random baseline, which is at 0.06 (0.09), 0.003 (0.006), 0.001 (0.003). However, the average scores are < 1 and thus not perfect. This may be caused by incorrect or imprecise annotations for subtle features and, therefore, hard to annotate with a VLM and SDXL Turbo failing to comply with some prompts.

Relatedness to texture. In Fig. 1 and App. D Fig. 6 the empty prompt interventions of the `up.0.1` features resulted in texture-like pictures. To quantify whether this consistently happens, we design a simple texture score by computing CLIP similarity between an image and the word "texture". Using this score, we compare empty-prompt interventions of the different transformer blocks with each other and real-world texture images. The results are in the second column of Table 1 (a) and suggest that empty-prompt intervention images of `up.0.1` and `up.0.0` resemble textures and some of the `down.2.1` images look like textures as well. For `up.0.0`, we did not observe any connection of these images to the top activating images. Interestingly, the score of `up.0.1` is higher than the one of the real-world textures dataset (Cimpoi et al. (2014)).

Color sensitivity. In our qualitative analysis, we suggested that the features learned on `up.0.1` relate to texture and color. If this holds, the image regions that activate a feature should not differ significantly in color on average. To test that, we calculate the "average" color for each feature: this is a weighted average of pixel colors with the feature activation values as weights. To determine the average color of a each feature we compute it over a sample of 10 images of the feature's top 5% images. Then, we calculate Manhattan distances between the colors of the pixels and the "average" color on the same images (the highest possible distance is $3 \cdot 255 = 765$). Finally, we take a weighted average of the Manhattan distances using the same weights. We report these distances for different transformer blocks and for the images generated on random prompts from LAION-COCO. We present the results in the third column of Table 1 (a). The average distance for the `up.0.1` transformer block is, in fact, the lowest.

Intervention locality. We suggested that the features learned on `up.0.0` and `up.0.1` influence intervened generations locally. We estimate how the top 5% images change inside and outside the active regions to quantitatively assess this claim. To exclude weak activation regions from consideration, we say that a pixel is inside the active area if the corresponding 32x32 patch has an activation value larger than 50% of the image patches, and it is outside the active area if the corresponding 32x32 patch has activation value of zero. In Table 1 (b), we report Manhattan distances between the original images and the intervened images outside and inside the active areas for activation modulation intervention strengths -10, -5, 5, 10. The features for `up.0.0` and `up.0.1` have a higher effect inside the active area than outside, in contrast to `down.2.1` for which this difference is smaller.

6 Related work

Image editing with diffusion models. Numerous studies have analyzed diffusion models' attribute editing capabilities. Yue et al. (2024) learn a disentangled representation on human faces (Liu et al., 2015; Karras et al., 2019) and bedrooms (Yu et al., 2015). The models of Yue et al. (2024) exhibit interpolation abilities between the attributes of two reference images. Similarly, Wang & Golland (2023) interpolate between pairs of images by fine-tuning a latent diffusion model. Through experiments on synthetic and real datasets, Deschenaux et al. (2024) demonstrate that even if a diffusion model is trained on an incomplete subset of the data distribution, it can still generate samples from the full distribution. Kim et al. (2022) show that one can guide a pre-trained diffusion model using text instructions leveraging CLIP (Radford et al., 2021). Meng et al. (2022) edit images using an off-the-shelf diffusion model that was not explicitly trained for image editing using the stochastic differential equation formalism of diffusion models. For example, the method of Meng et al. (2022)

can transform color strokes into photo-realistic images, similar to Park et al. (2019). Importantly, unlike Park et al. (2019), Meng et al. (2022) does not fine-tune the diffusion model for this task. (Zhang et al., 2023; Kawar et al., 2023; Baumann et al., 2024) demonstrate fine-grained attribute editing using a single reference image.

Analyzing the latent space of diffusion models. Kwon et al. (2023) show that diffusion models naturally have a semantically meaningful latent space. Park et al. (2023) analyze the latent space of diffusion models using Riemannian geometry. Li et al. (2024) and Dalva & Yanardag (2024) present self-supervised methods for finding semantic directions in the latent space. Similarly, Gandikota et al. (2023) show that the attribute variations lie in a low-rank space by learning LoRA adapters (Hu et al., 2021) on top of pre-trained diffusion models. Brack et al. (2023) and Wang et al. (2023) demonstrate effective semantic vector algebraic operations in the latent space of DMs, as observed by Mikolov et al. (2013). However, none of those works explicitly train SAEs to interpret and control the latent space.

Mechanistic interpretability using SAEs. Sparse autoencoders have recently been popularized by Bricken et al. (2023), in which they show that it is possible to learn interpretable features by decomposing neuron activations in MLPs in 2-layer transformer language models. At the same time, a parallel work decomposed the elements of the residual stream (Cunningham et al., 2023), which followed up on (Sharkey et al., 2022). To our knowledge, the first work that applied sparse autoencoders to transformer-based LLM was (Yun et al., 2021), which learned a joint dictionary for features of all layers. Recently, sparse autoencoders have gained much traction, and many have been trained even on state-of-the-art LLMs (Gao et al., 2024; Templeton & et al., 2024; Lieberum et al., 2024). In addition, great tools are available for inspection (Lin & Bloom, 2023) and automatic interpretation (Caden et al., 2024) of learned features. Marks et al. (2024) have shown how to use SAE features to facilitate automatic circuit discovery.

The studies most closely related to our work are (Bau et al., 2019), (Ismail et al., 2023) and (Daujotas, 2024). Ismail et al. (2023) apply concept bottleneck methods (Koh et al., 2020) that decompose latent concepts into vectors of interpretable concepts to generative image models, including diffusion models. Unlike the SAEs that we train, this method requires labeled concept data. Daujotas (2024) decomposes CLIP (Radford et al., 2021; Cherti et al., 2023) vision embeddings using SAEs and use them for conditional image generation with a diffusion model called Kandinsky (Razzhigaev et al., 2023). Importantly, using SAE features, they are able to manipulate the image generation process in interpretable ways. In contrast, in our work, we train SAEs on intermediate representations of the forward pass of SDXL Turbo. Consequently, we can interpret and manipulate SDXL Turbo's forward pass on a finer granularity, e.g., by intervening on specific transformer blocks and spatial positions. Another closely related work to ours is (Bau et al., 2019), in which neurons in generative adversarial neural networks are interpreted and manipulated. The interventions in (Bau et al., 2019) are similar to ours, but on neurons instead of sparse features. In order to identify neurons corresponding to a semantic concept, Bau et al. (2019) require semantic image segmentation maps.

7 Conclusion and discussion

We trained SAEs on SDXL Turbo's opaque intermediate representations. This study is the first in the academic literature to mechanistically interpret the intermediate representations of a modern text-to-image model. Our findings demonstrate that SAEs can extract interpretable features and have a significant causal effect on the generated images. Importantly, the learned features provide insights into SDXL Turbo's forward pass, revealing that transformer blocks fulfill specific and varying roles in the generation process. In particular, our results clarify the functions of `down.2.1`, `up.0.0`, and `up.0.1`. However, the role of `mid.0` remains less defined; it seems to encode more abstract information and interventions are less effective.

We follow up with a discussion of the results and their implications for future research. Based on our observations, we suggest a preliminary hypothesis about SDXL Turbo's generation process: `down.2.1` decides on top-level composition, `mid.0` encodes low-level semantics, `up.0.0` adds details based on the two above, and `up.0.1` fills in color, texture, and style.

While our work provides important insights into the mechanisms of SDXL Turbo, we studied its transformer blocks in isolation. Further research is required to understand how the features of

SDXL Turbo interact between blocks and how this affects the overall functionality of the model. A promising direction would be the application of advanced interpretability techniques such as those explored by Marks et al. (2024), which compute circuits showing how different layers and attention heads wire together. Therefore, these techniques would provide insight into our hypothesis stated above.

In addition, the complex nature of some of the learned visual features deserves special attention. Although some features (for example, learning on `down.2.1` and `up.0.1`) exhibit their effect even when turning them on during empty-prompt generations, other features (typically, the ones learned on `up.0.0`) require an appropriate context to show their effect. This complexity poses additional challenges for the automatic annotation of features. Our preliminary observations in this direction suggest that current visual language models struggle to detect such behaviors. Furthermore, we question whether captioning the visual features with few short sentences adequately captures most features' roles.

Our work highlights the potential of SAEs in revealing the internal structure of diffusion models like SDXL Turbo, and it could help future researchers answer more sophisticated questions about image generation. For example, how does SDXL Turbo add illumination effects, render wool, hair, or reflections of objects in the water?

ACKNOWLEDGEMENTS

We thank Danila Zubko for his early contributions to the project's initial discussions and development. Robert West's lab is partly supported by grants from the Swiss National Science Foundation (200021_185043, TMSGI2_211379), Swiss Data Science Center (P22_08), H2020 (952215), Microsoft, and Google.

REFERENCES

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *International Conference on Learning Representations*, 2019.

Stefan Andreas Baumann, Felix Krause, Michael Neumayr, Nick Stracke, Vincent Tao Hu, and Björn Ommer. Continuous, Subject-Specific Attribute Control in T2I Models by Identifying Semantic Directions, 2024.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Manuel Brack, Felix Friedrich, Dominik Hintersdorf, Lukas Struppek, Patrick Schramowski, and Kristian Kersting. Sega: Instructing text-to-image models using semantic guidance. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 25365–25389. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4ff83037e8d97b2171b2d3e96cb8e677-Paper-Conference.pdf.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, and Adam Jermyn et al. Towards monosemanticity: Decomposing language models dictionary learning. *Transformer Circuits*, October 2023. URL https://transformer-circuits.pub/2023/monosemantic-features.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Juang Caden, Paulo Gonçalo, Drori Jacob, and Belrose Nora. Open source automated interpretability for sparse autoencoder features, 2024. URL https://blog.eleuther.ai/autointerp/. Accessed: 2024-09-27.

Haozhe Chen, Carl Vondrick, and Chengzhi Mao. Selfie: Self-interpretation of large language model embeddings. *arXiv preprint arXiv:2403.10949*, 2024.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2829, 2023.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

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

Yusuf Dalva and Pinar Yanardag. Noiseclr: A contrastive learning approach for unsupervised discovery of interpretable directions in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24209–24218, June 2024.

Gytis Daujotas. Interpreting and steering features in images, 2024. URL https://www.lesswrong.com/posts/Quqekpvx8BGMMcaem/interpreting-and-steering-features-in-images. Accessed: 2024-09-27.

Justin Deschenaux, Igor Krawczuk, Grigorios Chrysos, and Volkan Cevher. Going beyond compositions, DDPMs can produce zero-shot interpolations. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=1pj0Sk8GfP.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Soch Joram Duchi John. Multivariate normal distribution: Kullback-leibler divergence, 05 2020. URL https://statproofbook.github.io/P/mvn-kl.html. Accessed on October 31, 2024.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits*, September 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021. URL https://arxiv.org/abs/2012.09841.

Rohit Gandikota, Joanna Materzynska, Tingrui Zhou, Antonio Torralba, and David Bau. Concept sliders: Lora adaptors for precise control in diffusion models, 2023. URL https://arxiv.org/abs/2311.12092.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscope: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*, 2024.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images, 2023. URL https://arxiv.org/abs/2301.11093.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Aya Abdelsalam Ismail, Julius Adebayo, Hector Corrada Bravo, Stephen Ra, and Kyunghyun Cho. Concept bottleneck generative models. In *The Twelfth International Conference on Learning Representations*, 2023.

William B Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019. URL https://arxiv.org/abs/1812.04948.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022. URL https://arxiv.org/abs/2206.00364.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models, 2023. URL https://arxiv.org/abs/2210.09276.

Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation, 2022.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models, 2023. URL https://arxiv.org/abs/2107.00630.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pp. 5338–5348. PMLR, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Diffusion models already have a semantic latent space, 2023. URL https://arxiv.org/abs/2210.10960.

Hang Li, Chengzhi Shen, Philip Torr, Volker Tresp, and Jindong Gu. Self-discovering interpretable diffusion latent directions for responsible text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12006–12016, June 2024.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

Jae Hyun Lim and Jong Chul Ye. Geometric gan, 2017. URL https://arxiv.org/abs/1705.02894.

Johnny Lin and Joseph Bloom. Neuronpedia: Interactive reference and tooling for analyzing neural networks, 2023. URL https://www.neuronpedia.org. Software available from neuronpedia.org.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Calvin Luo. Understanding diffusion models: A unified perspective, 2022. URL https://arxiv.org/abs/2208.11970.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations, 2022. URL https://arxiv.org/abs/2108.01073.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge?, 2018. URL https://arxiv.org/abs/1801.04406.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in inceptionv1. *Distill*, 5(4):e00024–002, 2020.

OpenAI. Hello gpt-4o, 2024. URL https://openai.com/index/hello-gpt-4o. Accessed: 2024-09-28.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization, 2019. URL https://arxiv.org/abs/1903.07291.

Yong-Hyun Park, Mingi Kwon, Jaewoong Choi, Junghyo Jo, and Youngjung Uh. Understanding the latent space of diffusion models through the lens of riemannian geometry. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 24129–24142. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4bfcebedf7a2967c410b64670f27f904-Paper-Conference.pdf.

Pablo Pernias, Dominic Rampas, Mats L Richter, Christopher J Pal, and Marc Aubreville. Würstchen: An efficient architecture for large-scale text-to-image diffusion models. *arXiv preprint arXiv:2306.00637*, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. URL https://arxiv.org/abs/2307.01952.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Anton Razzhigaev, Arseniy Shakhmatov, Anastasia Maltseva, Vladimir Arkhipkin, Igor Pavlov, Ilya Ryabov, Angelina Kuts, Alexander Panchenko, Andrey Kuznetsov, and Denis Dimitrov. Kandinsky: An improved text-to-image synthesis with image prior and latent diffusion. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 286–295, 2023.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022a. URL https://arxiv.org/abs/2205.11487.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022b.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL https://arxiv.org/abs/2202.00512.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster, 2021. URL https://arxiv.org/abs/2111.01007.

Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis, 2023a. URL https://arxiv.org/abs/2301.09515.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023b. URL https://arxiv.org/abs/2311.17042.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022a.

Christoph Schuhmann, Andreas Köpf, Richard Vencu, Theo Coombes, Romain Beaumont, and Benjamin Trom. Laion coco: 600m synthetic captions from laion2b-en, 2022b. URL https://laion.ai/blog/laion-coco/. Accessed: 2024-10-01.

Lee Sharkey, Dan Braun, and beren. Interim research report: Taking features out of superposition with sparse autoencoders, 2022. URL https://www.lesswrong.com/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition. Accessed: 2024-09-27.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020. URL https://arxiv.org/abs/1907.05600.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

Matteo Spinelli. Advanced style transfer with the mad scientist node. YouTube video, 2024. URL https://www.youtube.com/watch?v=ewKM7uCRPUg. Accessed: 2024-09-17.

Adly Templeton and Tom Conerly et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/. Accessed: 2024-09-27.

Clinton Wang and Polina Golland. Interpolating between images with diffusion models, 2023.

Zihao Wang, Lin Gui, Jeffrey Negrea, and Victor Veitch. Concept algebra for (score-based) text-controlled generative models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 35331–35349. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6f125214c86439d107ccb58e549e828f-Paper-Conference.pdf.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. URL http://dblp.uni-trier.de/db/journals/corr/corr1506.html#YuZSSX15.

Zhongqi Yue, Jiankun Wang, Qianru Sun, Lei Ji, Eric I-Chao Chang, and Hanwang Zhang. Exploring diffusion time-steps for unsupervised representation learning, 2024.

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

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zhixing Zhang, Ligong Han, Arnab Ghosh, Dimitris N. Metaxas, and Jian Ren. Sine: Single image editing with text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6027–6037, June 2023.

A Finding causally influential transformer blocks

As a first step, we narrow down design space of the 11 cross-attention transformer blocks (see Fig. 4) to those with the highest causal impact on the output. In order to assess their causal impact on the output we qualitatively study the effect of individually ablating each of them (see Fig. 2). As can be seen in Fig. 2 each of the middle blocks `down.2.1`, `mid.0`, `up.0.0`, `up.0.1` have a relatively high impact on the output respectively. In particular, the blocks `down.2.1` and `up.0.1` stand out. It seems like most colors and textures are added in `up.0.1`, which in the community is already known as "style" block Spinelli (2024). Ablating `down.2.1`, which is also already known in the community as "composition" block, impacts the entire image composition, including object sizes, orientations and framing. The effects of ablating other blocks such as `mid.0` and `up.0.0` are more subtle. For `mid.0` it is difficult to describe in words and `up.0.0` seems to add local details to the image while leaving the overall composition mostly intact.

B SAE Training Results

We trained several SAEs with different sparsity levels and sparse layer sizes and observed no dead features. To assess reconstruction quality, we processed 100 random LAION-COCO prompts through a one-step SDXL Turbo process, replacing the additive component of the corresponding transformer block with its SAE reconstruction.

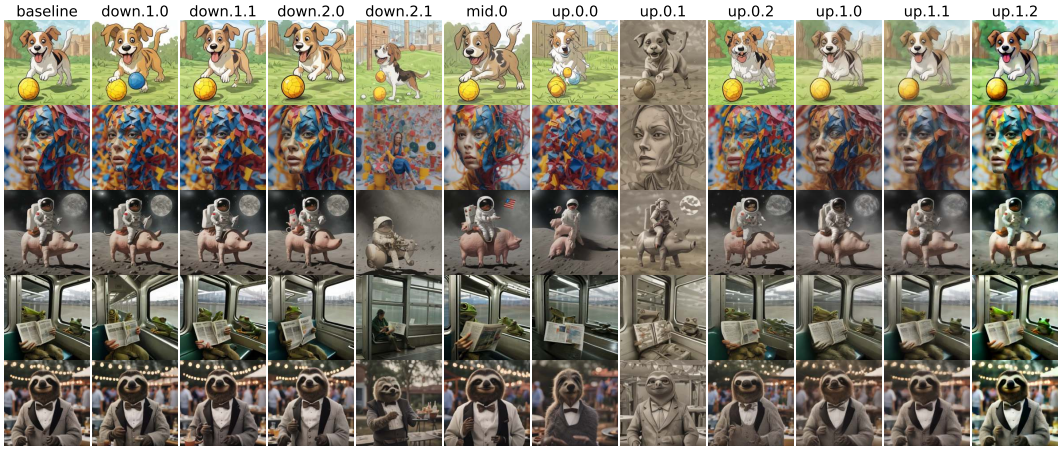


Figure 2: We generate images for the prompts "A dog playing with a ball cartoon.", "A photo of a colorful model.", "An astronaut riding on a pig on the moon.", "A photograph of the inside of a subway train. There are frogs sitting on the seats. One of them is reading a newspaper. The window shows the river in the background." and "A cinematic shot of a professor sloth wearing a tuxedo at a BBQ party." while ablating the updates performed by different cross-attention layers (indicated by the titles). The title "baseline" corresponds to the generation without interventions.

The explained variance ratio and the output effects caused by reconstruction are shown in Table 2. Fig. 3 presents random examples of reconstructions from an SAE with the following hyperparameters: $k = 10, n_f = 5120$, trained on `down.2.1`. The reconstruction causes minor deviations in the images, and the fairly low LPIPS (Zhang et al., 2018) and pixel distance scores also support these findings. However, to prevent these minor reconstruction errors from affecting our analysis of interventions, we decided to directly add or subtract learned directions from dense feature maps.

Table 2: Distances and explained variance ratio in generated images. "Mean" represents the average pixel Manhattan distance between original and reconstruction-intervened images, with a maximum possible value of 765. "Median" represents the median Manhattan distance per pixel, averaged over all images. 'LPIPS' refers to the average LPIPS score, measuring perceptual similarity. "Explained variance ratio" denotes the ratio of variance explained by the trained SAEs to the total variance.

k	n_f	Configuration	Mean | Median	LPIPS	Explained Variance Ratio (%)
5	640	down	83.29 | 50.04	0.3383	56.0
		mid	52.64 | 26.82	0.2032	43.4
		up0	55.89 | 30.69	0.2276	44.8
		up	52.67 | 34.53	0.2073	50.3
	5120	down	74.68 | 41.49	0.3036	67.8
		mid	48.82 | 24.60	0.1845	50.8
		up0	49.19 | 25.86	0.1969	57.2
		up	47.50 | 31.11	0.1775	59.5
10	640	down	73.65 | 41.79	0.2893	62.8
		mid	46.80 | 23.10	0.1772	51.5
		up0	48.43 | 25.80	0.1908	52.5
		up	43.06 | 26.85	0.1638	58.7
	5120	down	64.97 | 34.77	0.2582	73.7
		mid	44.02 | 21.72	0.1627	58.8
		up0	42.08 | 21.54	0.1624	64.2
		up	39.77 | 24.84	0.1453	67.1
20	640	down	59.29 | 31.47	0.2291	69.9
		mid	39.95 | 19.44	0.1459	60.0
		up0	40.15 | 21.06	0.1499	60.9
		up	31.97 | 18.15	0.1196	66.7
	5120	down	56.37 | 29.04	0.2190	78.8
		mid	37.28 | 17.82	0.1328	66.5
		up0	35.73 | 18.03	0.1302	70.6
		up	30.31 | 17.22	0.1104	74.2

Figure 3: Images generated from 10 random prompts taken from the LAION-COCO dataset are shown in the first row. In the second row, `down.2.1` updates are replaced by their SAE reconstructions ($k = 10, n_f = 5120$). The third row visualizes the differences between the original and reconstructed images.

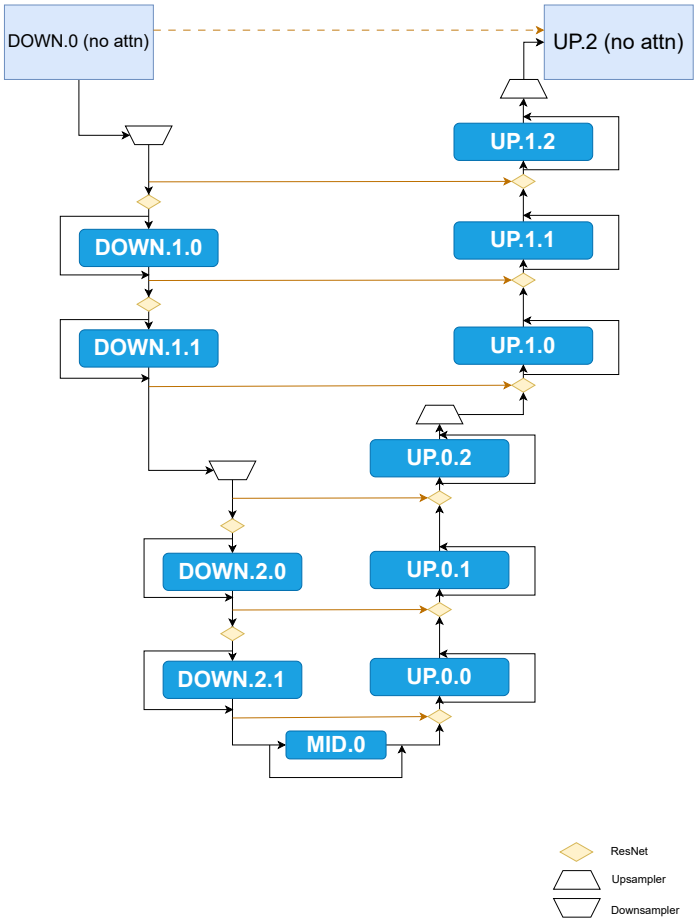


Figure 4: Cross-attention transformer blocks in SDLX's U-net.

C CASE STUDY I: MOST ACTIVE FEATURES ON A PROMPT

Fig. 5 demonstrates an extended version of Case Study I, showcasing 9 top features instead of 5.

D FEATURES INCLUDING PROMPTS

Feature plots. We provide the same plots as in Fig. 6 but for the last six feature indices of each transformer block in Fig. 7 and the corresponding prompts in Table 4. Additionally, provide some selected features for `down.2.1` and `up.0.1` in Fig. 8 and the corresponding prompts in Table 5.

Intervention plots. Additionally, we provide plots in which we turn on features from Fig. 8 but in unrelated prompts (as opposed to top dataset example prompts that already activate the features by themselves). For simplicity here we simply turn on the features across all spatial locations, which does not seem to be a well suitable strategy for `up.0.1`, which usually acts locally. To showcase, the difference we created one example image in Fig. 10, in which we manually draw localized masks to turn on the corresponding features.

E ANNOTATION PIPELINE DETAILS

We used GPT-4o to caption learned features on `down.2.1`. For each feature, the model was shown a series of 5 unrelated images, a progression of 9 images, the i-th of those corresponds to $\sim i \cdot 10\%$ average activation value of the maximum. Finally, we show 5 images corresponding to the highest average activations. Since some features are active on particular parts of images, the last 9 images are provided alongside their so-called "coldmaps": a version of an image with weakly active and inactive regions being faded and concealed.

The images were generated by 1-step SDXL Turbo diffusion process on $50'000$ random prompts of LAION-COCO dataset.

E.1 TEXTUAL PROMPT TEMPLATE

Here is the prompt template for the VLM.

> **System.** You are an experienced mechanistic interpretability researcher that is labeling features from the hidden representations of an image generation model.
>
> **User.** You will be shown a series of images generated by a machine learning model. These images were selected because they trigger a specific feature of a sparse auto-encoder, trained to detect hidden activations within the model. This feature can be associated with a particular object, pattern, concept, or a place on an image. The process will unfold in three stages:
>
> 1. **Reference Images:** First, you'll see several images *unrelated* to the feature. These will serve as a reference for comparison.
>
> 2. **Feature-Activating Images:** Next, you'll view images that activate the feature with varying strengths. Each of these images will be shown alongside a version where non-activated regions are masked out, highlighting the areas linked to the feature.
>
> 3. **Strongest Activators:** Finally, you'll be presented with the images that most strongly activate this feature, again with corresponding masked versions to emphasize the activated regions.
>
> Your task is to carefully examine all the images and identify the thing or concept represented by the feature. Here's how to provide your response:
>
> - **Reasoning:** Between `<thinking>` and `</thinking>` tags, write up to 400 words explaining your reasoning. Describe the visual patterns, objects, or concepts that seem to be consistently present in the feature-activating images but not in the reference images.

(a) Top 9 features of `down.2.1`

(b) Top 9 features `up.0.1`

(c) Top 9 features of `up.0.0`

(d) Top 9 features of `mid.0`

Figure 5: The top 9 features of `down.2.1` (a), `up.0.1` (b), `up.0.0` (c) and `mid.0` (d) for the prompt: "A cinematic shot of a professor sloth wearing a tuxedo at a BBQ party." Each row represents a feature. The first column depicts a feature heatmap (highest activation red and lowest nonzero one blue). The column titles containing "A" show feature modulation interventions, the ones containing "B" the intervention of turning on the feature on the empty prompt, and the ones containing "C" depict top dataset examples. Floating point values in the title denote β and γ values.

- **Expression:** Afterward, between `<answer>` and `</answer>` tags, write a concise phrase (no more than 15 words) that best captures the common thing or concept across the majority of feature-activating images.

Note that not all feature-activating images may perfectly align with the concept you're describing, but the images with stronger activations should give you the clearest clues. Also pay attention to the masked versions, as they highlight the regions most relevant to the feature.

User. These images are not related to the feature: {Reference Images}

User. This is a row of 9 images, each illustrating increasing levels of feature activation. From left to right, each image shows a progressively higher activation, starting with the image on the far left where the feature is activated at 10% relative to the image that activates it the most, all the way to the far right, where the feature activates at 90% relative to the image that activates it the most. This gradual transition highlights the feature's growing importance across the series. {Feature-Activating Images}

User. This row consists of 9 masked versions of the original images. Each masked image corresponds to the respective image in the activation row. Areas where the feature is not activated are completely concealed by a white mask, while regions with activation remain visible.) {Feature-Activating Images Coldmaps}

User. These images activate the feature most strongly. {Strongest Activators}

User. These masked images highlight the activated regions of the images that activate the feature most strongly. The masked images correspond to the images above. The unmasked regions are the ones that activate the feature. {Strongest Activators Coldmaps}

E.2 Example of Prompt Images

The images used to annotate feature 0 are shown in Fig. 11.

E.3 Examples of Generated Captions

We present the captions generated by GPT-4o for the first and last 10 features in Table 6.

F Sparse autoencoders and superposition

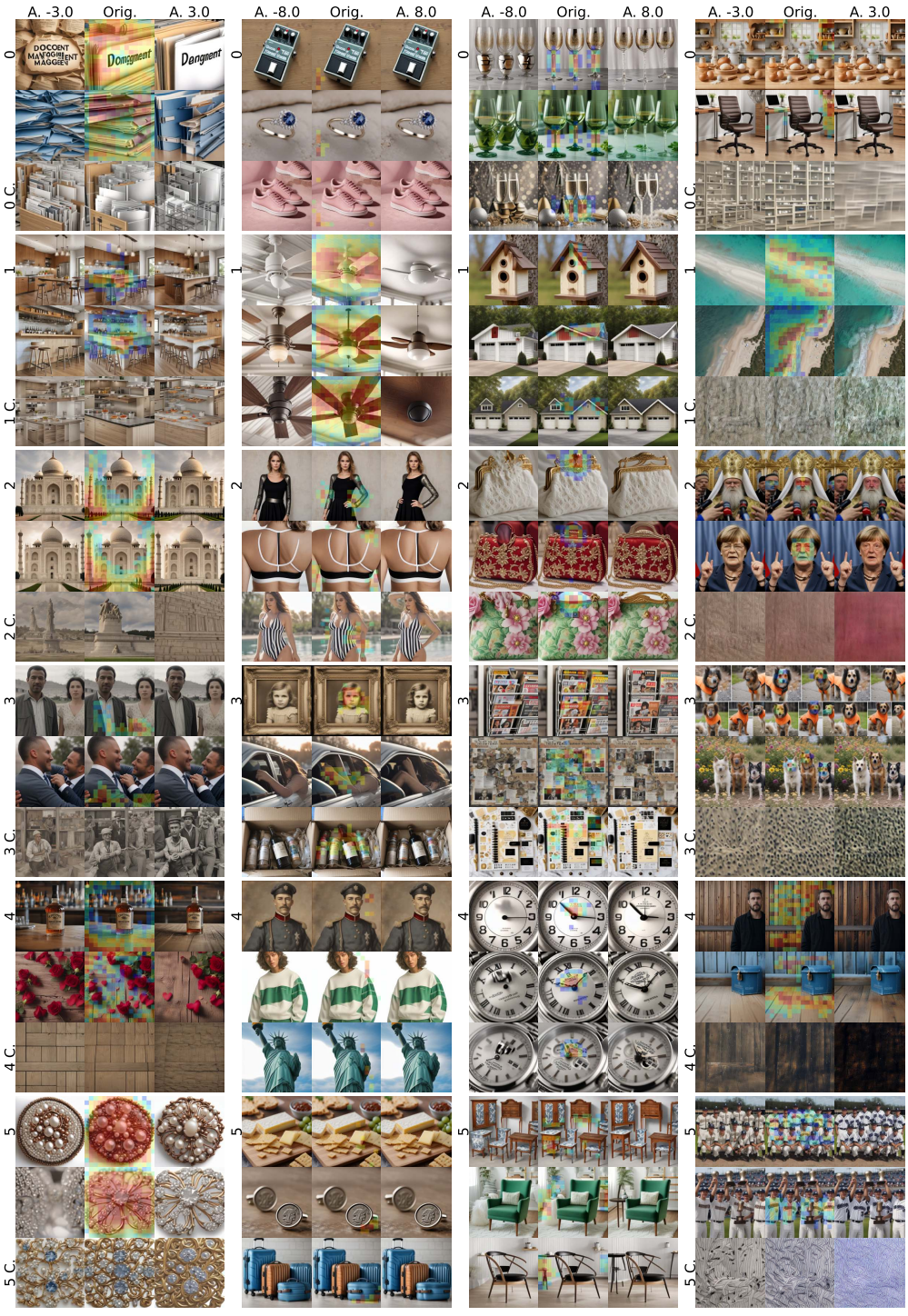
Let $h(x) \in \mathbb{R}^d$ be some intermediate result of a forward pass of a neural network on the input x. In a fully connected neural network, the components $h(x)$ could correspond to neurons. In transformers, which are residual neural networks with attention and fully connected layers, $h(x)$ usually either refers to the content of the residual stream after some layer, an update to the residual stream by some layer, or the neurons within a fully connected block. In general, $h(x)$ could refer to anything, e.g., also keys, queries, and values. It has been shown (Yun et al., 2021; Cunningham et al., 2023; Bricken et al., 2023) that in many neural networks, especially LLMs, intermediate representations can be well approximated by sparse sums of $n_f \in \mathbb{N}$ learned feature vectors, i.e.,

$$h(x) \approx \sum_{\rho=1}^{n_f} s_\rho(x)\mathbf{f}_\rho, \tag{12}$$

where $s_\rho(x)$ are the input-dependent[8] coefficients most of which are equal to zero and $\mathbf{f}_1, \ldots, \mathbf{f}_{n_f} \in \mathbb{R}^d$ is a learned dictionary of feature vectors.

Importantly, these learned features are usually highly *interpretable* (specific), *sensitive* (fire on the relevant contexts), *causal* (change the output in expected ways in intervention) and usually do not correspond directly to individual neurons. There are also some preliminary results on the universality of these learned features, i.e., that different training runs on similar data result in the corresponding models picking up largely the same features (Bricken et al., 2023).

[8] In the literature this input dependence is usually omitted.

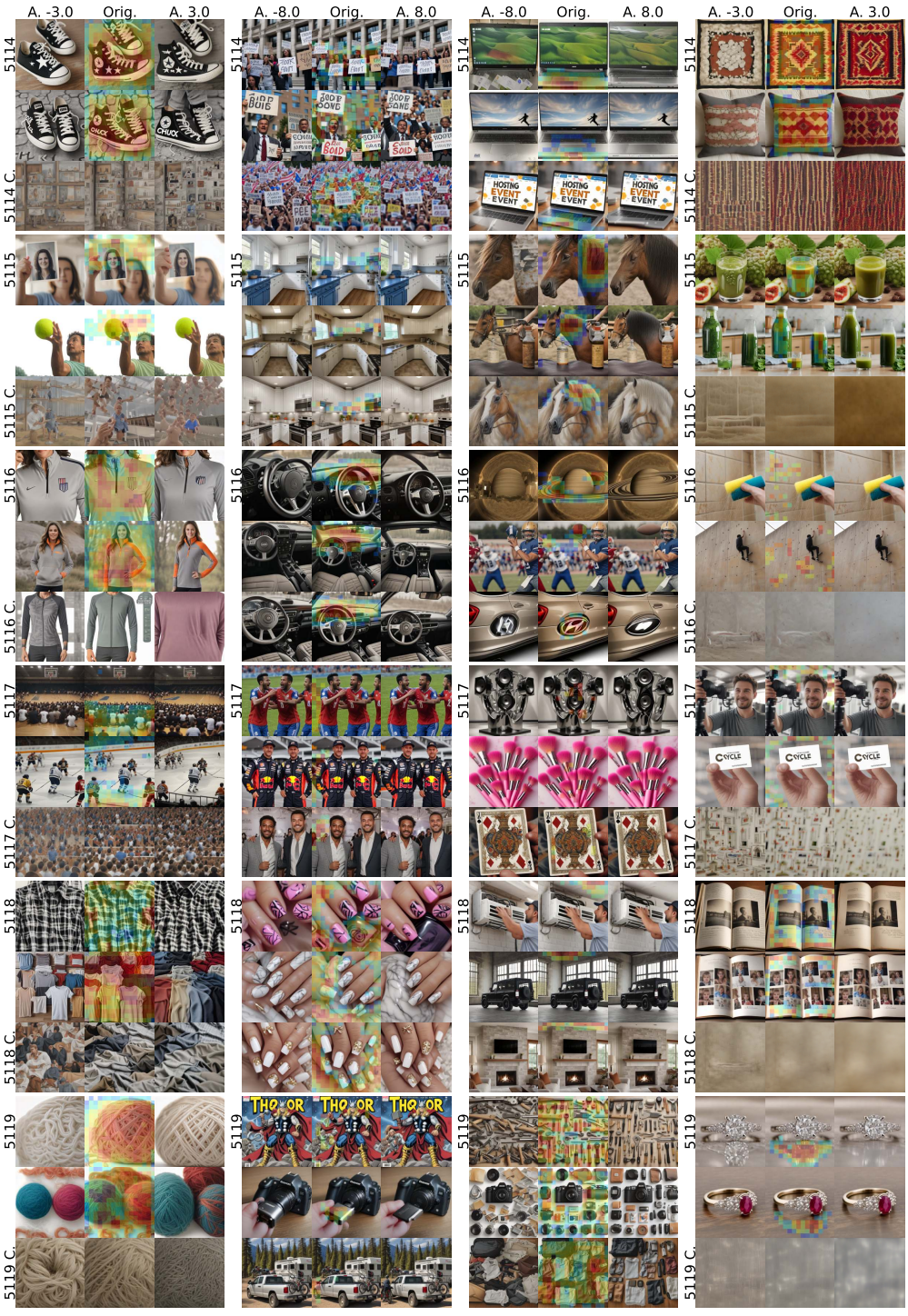


(a) `down.2.1` (b) `mid.0` (c) `up.0.0` (d) `up.0.1`

Figure 6: We visualize 6 features for `down.2.1` (a), `mid.0` (b), `up.0.0`, and `up.0.1`. We use three columns for each transformer block and three rows for each feature. For `down.2.1` and `up.0.1` we visualize the two samples from the top 5% quantile of activating dataset examples (middle) together a feature ablation (left) and a feature enhancement (right), and, activate the feature on the empty prompt with $\gamma = 0.5, 1, 2$ from left to right. For `mid.0` and `up.0.0` we display three samples with ablation and enhancement. Captions are in Table 3.

Table 3: Prompts for the top 5% quantile examples in Fig. 6

Block	Feature	Prompt
down.2.1	0	A file folder with the word document management on it.
	0	Two blue folders filled with dividers.
	1	A kitchen with an island and bar stools.
	1	An unfinished bar with stools and a wood counter.
	2	The Taj Mahal, or a white marble building in India.
	2	The Taj Mahal, or a white marble building in India.
	3	A man and woman standing next to each other.
	3	Two men in suits hugging each other outside.
	4	An old Forester whiskey bottle sitting on top of a wooden table.
	4	Red roses and hearts on a wooden table.
	5	A beaded brooch with pearls and copper.
	5	An image of a brooch with diamonds.
mid.0	0	The Boss TS-3W pedal has an electronic tuner.
	0	An engagement ring with blue sapphire and diamonds.
	0	The women's pink sneaker is shown.
	1	A white ceiling fan with three blades.
	1	A ceiling fan with three blades and a light.
	1	The ceiling fan is dark brown and has two wooden blades.
	2	The black dress is made from knit and has metallic sleeves.
	2	The back view of a woman wearing a black and white sports bra.
	2	The woman is wearing a striped swimsuit.
	3	An old-fashioned photo frame with a little girl on it.
	3	The woman is sitting in her car with her head down.
	3	The contents of an empty bottle in a box.
	4	An old painting of a man in uniform.
	4	The model wears an off-white sweatshirt with green panel.
	4	The Statue of Liberty stands tall in front of a blue sky.
	5	Cheese and crackers on a cutting board.
	5	Two cufflinks with coins on them.
	5	Three pieces of luggage are shown in blue.
up.0.0	0	Three wine glasses with gold and silver designs.
	0	Three green wine glasses sitting next to each other.
	0	New Year's Eve with champagne, gold, and silver.
	1	The birdhouse is made from wood and has a brown roof.
	1	The garage is white with red shutters.
	1	Two garages with one attached porch and the other on either side.
	2	An elegant white lace purse with gold clasp.
	2	The red handbag has gold and silver designs.
	2	A pink and green floral-colored purse.
	3	A magazine rack with magazines on it.
	3	The year-in-review page for this digital scrap.
	3	The planner sticker kit is shown with gold and black accessories.
	4	A clock with numbers on the face.
	4	A silver watch with roman numerals on the face.
	4	An automatic watch with a silver dial.
	5	Four pieces of wooden furniture with blue and white designs.
	5	The green chair is in front of a white rug.
	5	The wish chair with a black seat.
up.0.1	0	The wooden toy kitchen set includes bread, eggs, and flour.
	0	The office chair is brown and black.
	1	An aerial view of the white sand and turquoise water.
	1	An aerial view of the beach and ocean.
	2	The patriarch of Ukraine is shown speaking to reporters.
	2	German Chancellor Merkel gestures as she speaks to the media.
	3	Four pictures showing dogs wearing orange vests.
	3	Two dogs are standing on the ground next to flowers.
	4	A man standing in front of a wooden wall.
	4	A blue mailbox sitting on top of a wooden floor.
	5	The baseball players are posing for a team photo.
	5	The baseball players are holding up their trophies.

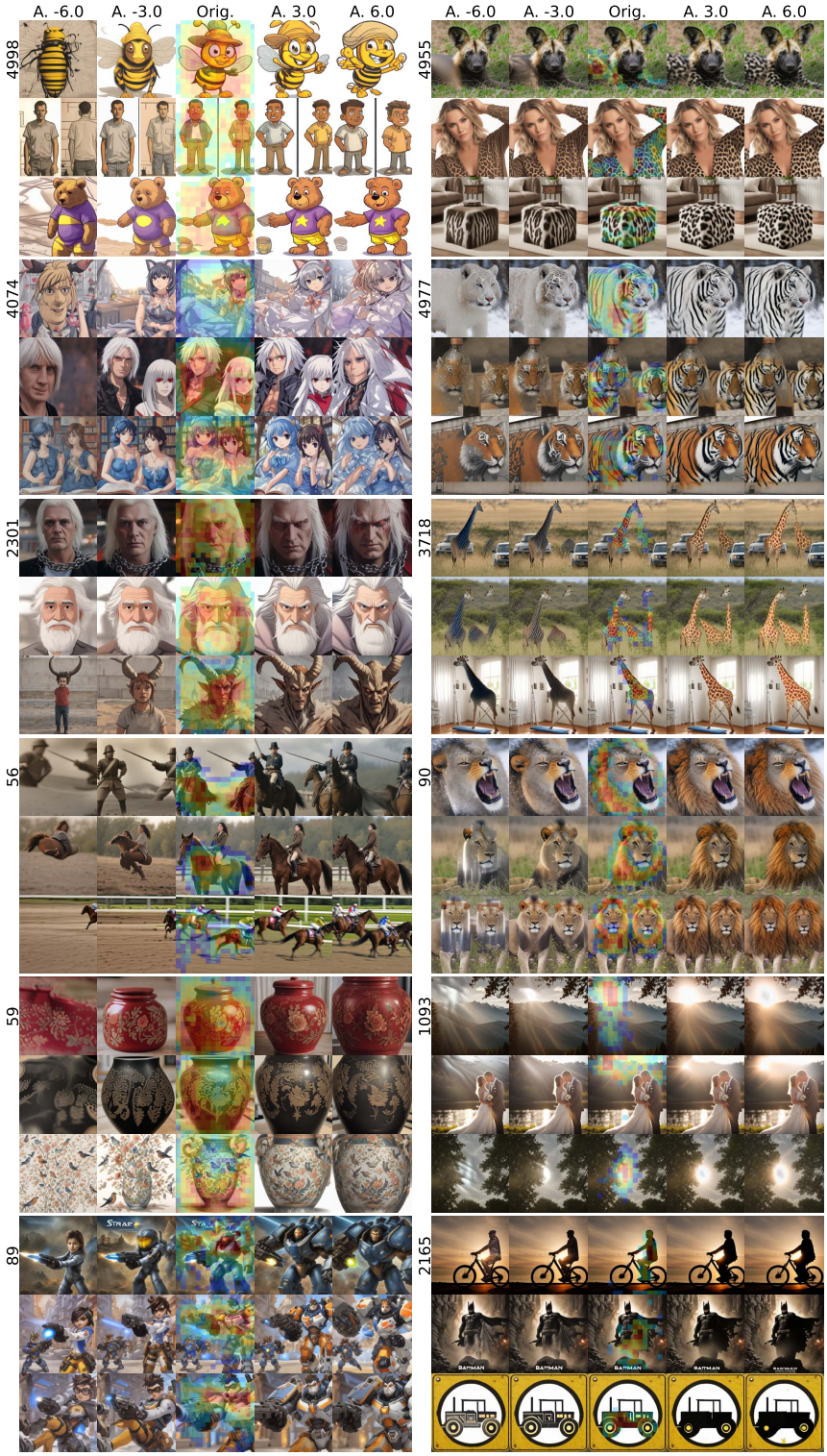


(a) `down.2.1` (b) `mid.0` (c) `up.0.0` (d) `up.0.1`

Figure 7: We visualize last 6 features for `down.2.1` (a), `mid.0` (b), `up.0.0`, and `up.0.1`. We use three columns for each transformer block and three rows for each feature. For `down.2.1` and `up.0.1` we visualize two samples from the top 5% quantile of activating dataset examples (middle) together a feature ablation (left) and a feature enhancement (right), and, activate the feature on the empty prompt with $\gamma = 0.5, 1, 2$ from left to right. For `mid.0` and `up.0.0` we display three samples with ablation and enhancement. Captions are in Table 4.

Table 4: Prompts for the top 5 % quantile examples in Fig. 7

Block	Feature	Prompt
down.2.1	5114	Black and white Converse sneakers with the word black star.
	5114	Black and white Converse sneakers with the word Chuck.
	5115	A woman holding up a photo of herself.
	5115	A man holding up a tennis ball in the air.
	5116	The Nike Women's U.S. Soccer Team DRI-Fit 1/4 Zip Top.
	5116	The women's gray and orange half-zip sweatshirt.
	5117	A large group of people sitting in front of a basketball court.
	5117	Hockey players are playing in an arena with spectators.
	5118	The black and white plaid shirt is shown.
	5118	The different colors and sizes of t-shirts.
	5119	A ball of yarn on a white background.
	5119	Two balls of colored wool are on the white surface.
mid.0	5114	People holding signs in front of a building.
	5114	Two men dressed in suits and ties are holding up signs.
	5114	A large group of people holding flags and signs.
	5115	A kitchen with white cabinets and a blue stove.
	5115	The kitchen is clean and ready for us to use.
	5115	A kitchen with white cabinets and stainless steel appliances.
	5116	The steering wheel and dashboard in a car.
	5116	The interior of a car with dashboard controls.
	5116	The dashboard and steering wheel in a car.
	5117	Three men are celebrating a goal on the field.
	5117	Two men in Red Bull racing gear standing next to each other.
	5117	Two men are posing for the camera at an event.
	5118	Someone is holding up their nail polish with pink and black designs.
	5118	The nail is very cute and looks great with marble.
	5118	White stily nails with gold and diamonds.
	5119	The Mighty Thor comic book.
	5119	The camera is showing its flash drive.
	5119	A truck with bikes on the back parked next to a camper.
up.0.0	5114	The Acer laptop is open and ready to use.
	5114	The Lenovo S13 laptop is open and has an image of a person jumping off the keyboard.
	5114	A laptop with the words Hosting Event on it.
	5115	A horse with a black nose and brown mane.
	5115	The horse leather oil is being used to protect horses.
	5115	An oil painting on a canvas of a horse.
	5116	The sun is shining brightly over Saturn.
	5116	A football player throws the ball to another team.
	5116	Car door light logo sticker for Hyundai.
	5117	An artistic black and silver sculpture with speakers.
	5117	The pink brushes are sitting on top of each other.
	5117	Four kings playing cards in the hand.
	5118	A man is fixing an air conditioner.
	5118	The black Land Rover is parked in front of a large window.
	5118	A flat screen TV mounted on the wall above a fireplace.
	5119	A table with many different tools on it.
	5119	A camera with many different items including flash cards, lenses, and other accessories.
	5119	The contents of an open suitcase and some clothes.
up.0.1	5114	An old Navajo rug with multicolored designs.
	5114	The pillow is made from an old kilim.
	5115	An image of noni juice with some fruits.
	5115	A bottle and glass on the counter with green juice.
	5116	Someone cleaning the shower with a sponge.
	5116	A man on a skateboard climbing a wall with ropes.
	5117	A man taking a selfie in front of some camera equipment.
	5117	A person holding up a business card with the words cycle transportation.
	5118	Two photos are placed on top of an open book.
	5118	An open book with pictures of children and their parents.
	5119	An engagement ring with diamonds on top.
	5119	An oval ruby and diamond ring.



(a) `down.2.1` (b) `up.0.1`

Figure 8: We visualize 6 features for `down.2.1` (a) and `up.0.1` (b). We use 5 columns for each transformer block and three rows for each feature. We visualize three samples from the top 5% quantile of activating dataset examples (middle) together a feature ablation (left) and a feature enhancement (right). Captions are in Table 5.

Table 5: Prompts for the top 5% quantile examples in Fig. 8

Block	Feature	Prompt
down.2.1	4998	A cartoon bee wearing a hat and holding something.
	4998	Two cartoon pictures of the same man with his hands in his pockets.
	4998	A cartoon bear with a purple shirt and yellow shorts.
	4074	An anime character with cat ears and a dress.
	4074	Two anime characters, one with white hair and the other with red eyes.
	4074	An anime book with two women in blue dresses.
	2301	A man with white hair and red eyes holding a chain.
	2301	An animated man with white hair and a beard.
	2301	The character is standing with horns on his head.
	56	Two men in uniforms riding horses with swords.
	56	A woman riding on the back of a brown horse.
	56	Two jockeys on horses racing down the track.
	59	A red jar with floral designs on it.
	59	An old black vase with some design on it.
	59	A vase with birds and flowers on it.
	89	StarCraft 2 is coming to the Nintendo Wii.
	89	Overwatch is coming to Xbox and PS3.
	89	The hero in Overwatch is holding his weapon.
up.0.1	4955	An African wild dog laying in the grass.
	4955	The woman is posing for a photo in her leopard print top.
	4955	An animal print cube ottoman with brown and white fur.
	4977	A white tiger with blue eyes standing in the snow.
	4977	A bottle and tiger are shown next to each other.
	4977	A mural on the side of a building with a tiger.
	3718	Giraffes are standing in the grass near a vehicle.
	3718	Two giraffes standing next to each other in the grass.
	3718	A giraffe standing next to an ironing board.
	90	A lion is roaring its teeth in the snow.
	90	A lion sitting in the grass looking off into the distance.
	90	Two lions with flowers on their backs.
	1093	The sun is shining over mountains and trees.
	1093	Bride and groom in front of a lake with sun flare.
	1093	The milky sun is shining brightly over the trees.
	2165	The silhouette of a person riding a bike at sunset.
	2165	The Dark Knight rises from his cave in Batman's poster.
	2165	A yellow sign with black design depicting a tractor.

Table 6: down.2.1 first 10 and last 10 feature captions.

Block	Feature	Caption
down.2.1	0	Organizational/storage items for documents and office supplies
	1	Luxury kitchen interiors and designs
	2	Architectural Landmarks and Monumental Buildings
	3	Upper body clothing and attire
	4	Rustic or Natural Wooden Textures or Surfaces
	5	Intricately designed and ornamental brooches
	6	Technical diagrams and instructional content
	7	Feature predominantly activated by visual representations of dresses
	8	Home decor textiles focusing on cushions and pillows
	9	Eyewear: glasses and sunglasses
	5110	Concept of containment or organized enclosure
	5111	Groups of people in collective settings
	5112	Modern minimalist interior design
	5113	Indoor plants and greenery
	5114	Feature sensitivity focused on sneakers
	5115	Handling or manipulating various objects
	5116	Athletic outerwear, particularly zippered sporty jackets
	5117	Spectator Seating in Sporting Venues
	5118	Textiles and clothing materials, focus on textures and folds
	5119	Yarn and Knitting Textiles

Superposition. By associating task-relevant features with directions in $\mathbb{R}^d$ instead of individual components of $h(x) \in \mathbb{R}^d$, it is possible to represent many more features than there are components, i.e., $n_f >> d$. As a result, in this case, the learned dictionary vectors $\mathbf{f}_1, \ldots, \mathbf{f}_{n_f}$ cannot be orthogonal to each other, which can lead to interference when too many features are on (thus the sparsity requirement). However, it would be theoretically possible to have exponentially (in d) many almost orthogonal directions embedded in $\mathbb{R}^d$.[9]

Using representations like this, the optimization process during training can trade off the benefits of being able to represent more features than there are components in h with the costs of features interfering with each other. Such representations are especially effective if the real features underlying the data do not co-occur with each other too much, that is, they are sparse. In other words, in order to represent a single input ("Michael Jordan") only a small subset of the features ("person", ..., "played basketball") is required (Elhage et al., 2022; Bricken et al., 2023).

The phenomenon of neural networks that exploit representations with more features than there are components (or neurons) is called superposition (Elhage et al., 2022). Superposition can explain the presence of polysemantic neurons. The neurons, in this case, are simply at the wrong level of abstraction. The closest feature vector can change when varying a neuron, resulting in the neuron seemingly reacting to or steering semantically unrelated things.

Sparse autoencoders. In order to implement the sparse decomposition from equation 12, the vector s containing the n_f coefficients of the sparse sum is parameterized by a single linear layer followed by ReLU activations, called the *encoder*,

$$s = \texttt{ENC}(h) = \sigma(W^{\texttt{ENC}}(h - b_{\text{pre}}) + b_{\text{act}}), \tag{13}$$

in which $h \in \mathbb{R}^d$ is the latent that we aim to decompose, $\sigma(\cdot) = \max(0, \cdot)$, $W^{\texttt{ENC}} \in \mathbb{R}^{n_f \times d}$ is a learnable weight matrix and b_{pre} and b_{act} are learnable bias terms. We omitted the dependencies $h = h(x)$ and $s = s(h)$ that are clear from context.

Similarly, the learnable features are parametrized by a single linear layer, called *decoder*,

$$h' = \texttt{DEC}(s) = W^{\texttt{DEC}} s + b_{\text{pre}}, \tag{14}$$

in which $W^{\texttt{DEC}} = (\mathbf{f}_1 | \cdots | \mathbf{f}_{n_f}) \in \mathbb{R}^{d \times n_f}$ is a learnable matrix of whose columns take the role of learnable features and b_{pre} is a learnable bias term.

Training. The pair ENC and DEC are trained in a way that ensures that h' is a sparse sum of feature vectors (as in equation 1). Given a dataset of latents $h_1, \ldots, h_n$, both encoder and decoder are trained jointly to minimize a proxy to the loss

$$\min_{\substack{W^{\texttt{ENC}}, W^{\texttt{DEC}} \\ b_{\text{pre}}, b_{\text{act}}}} \sum_{i=1}^{n} \|h_i' - h_i\|_2^2 + \lambda \|s_i\|_0 = \sum_{i=1}^{n} \|\texttt{DEC}(\texttt{ENC}(h_i)) - h_i\|_2^2 + \lambda \|\texttt{ENC}(h_i)\|_0, \tag{15}$$

where $h_i = h(x_i)$, $s_i = \texttt{ENC}(h(x_i))$ (when we refer to components of s we use s_ρ instead), the $\|h_i' - h_i\|_2^2$ is a reconstruction loss, $\|s_i\|_0$ a regularization term ensuring the sparsity of the activations and λ the corresponding trade-off term.

In practice, $\|s_i\|_0$ cannot be efficiently optimized directly, which is why it is usually replaced with $\|s_i\|_1$ or other proxy objectives.

Technical details. In our work, we make use of the top-k formulation from Gao et al. (2024), in which $\|s_i\|_0 \leq k$ is ensured by introducing the a top-k function TopK into the encoder:

$$s = \texttt{ENC}(h) = \sigma(\texttt{TopK}(W^{\texttt{ENC}}(h - b_{\text{pre}}) + b_{\text{act}})). \tag{16}$$

As the name suggests, TopK returns a vector that sets all components except the top k ones to zero.

In addition (Gao et al., 2024) use an auxiliary loss to handle dead features. During training, a sparse feature ρ is considered *dead* if s_ρ remains zero over the last 10M training examples.

The resulting training loss is composed of two terms: the L_2-reconstruction loss and the top-auxiliary L_2-reconstruction loss for dead feature reconstruction. For a single latent h, the loss is defined

$$L(h, h') = \|h - h'\|_2^2 + \alpha \|h - h'_{\text{aux}}\|_2^2 \tag{17}$$

[9] It follows from the Johnson-Lindenstrauss Lemma (Johnson et al., 1986) that one can find at least $\exp(d\epsilon^2/8)$ unit vectors in $\mathbb{R}^d$ with the dot product between any two not larger than ϵ.

In this equation, the h'_{aux} is the reconstruction based on the top k_{aux} dead features. This auxiliary loss is introduced to mitigate the issue of dead features. After the end of the training process, we observed none of them. Following (Gao et al., 2024), we set $\alpha = \frac{1}{32}$ and $k_{\text{aux}} = 256$, performed tied initialization of encoder and decoder, normalized decoder rows after each training step. The number of learned features n_f is set to 5120, which is four times the length of the input vector. The value of k is set to 10 as a good trade-off between sparsity and reconstruction quality. Other training hyperparameters are batch size: 4096, optimizer: Adam with learning rate: 10^{-4} and betas: $(0.9, 0.999)$.

G FEW STEP DIFFUSION MODELS: SDXL TURBO

Diffusion models. Diffusion models are a class of generative models that were introduced by Sohl-Dickstein et al. (2015) and are a core component of many of the recent large-scale text-to-image generative models (Ramesh et al., 2022; Rombach et al., 2022; Saharia et al., 2022a). Notably, Ho et al. (2020); Song & Ermon (2020) demonstrate that diffusion model are a viable alternative to GANs (Goodfellow et al., 2014) for image generation. Additionally, diffusion models enjoy stable training dynamics, are easier to scale than GANs (Dhariwal & Nichol, 2021), and offer likelihood estimates of samples (Song et al., 2021).

Diffusion models sample from an unknown distribution p by learning to iteratively denoise corrupted samples, starting from pure noise. The corruption process is defined on training samples from p. Mathematically, the images are corrupted with Gaussian noise and are distributed according to

$$q_t(x_t|x_0) := \mathcal{N}(\alpha_t x_0, \sigma_t^2 \mathbf{I}), \tag{18}$$

where x_0 corresponds to a real image from p, $0 \leq t \leq T$, α_t, σ_t^2 are positive real-valued scalars such that the signal-to-noise ratio $SNR := \frac{\alpha_t}{\sigma_t^2}$ is monotonically decreasing. Additionally, the coefficients $\alpha_{T-1}, \sigma_{T-1}^2$ are typically chosen such that $x_T \sim \mathcal{N}(0, \mathbf{I})$. In this work, the number of corruption steps T is fixed to 1000, as we study the pre-trained models from (Sauer et al., 2023b). Given this predetermined corruption process, the diffusion model learns to reverse it to recover clean data.

The denoising process is implemented via a distribution $p_\theta(x_{t-1}|x_t)$. The simplest way to generate samples using $p_\theta(x_{t-1}|x_t)$ is to first generate a sample of pure noise $x_T \sim \mathcal{N}(0, \mathbf{I})$, followed by T iterative applications of p_θ, yielding a sequence $x_T, x_{T-1}, ..., x_1, x_0$, where x_0 approximates samples from p. The vector θ represents the parameters of a neural network that defines $p_\theta(x_{t-1}|x_t)$. There exist many objectives to learn to reverse the corruption process (Ho et al., 2020; Kingma et al., 2023; Song & Ermon, 2020), but p_θ is generally trained to minimize the Kullback-Leibler divergence between adjacent steps of the corruption process: $D_{KL}\left[q_t(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\right]$ for $t \in \{1, ..., T\}$, where $q_t(x_{t-1}|x_t, x_0)$ is a Gaussian distribution whose mean and variance can be computed in closed-form using Bayes rule and the definition in eq. (18) (Ho et al., 2020). The denoising distribution $p_\theta(x_{t-1}|x_t)$ is parameterized to be Gaussian. The Kullback-Leibler divergence between two Gaussians admits a simple closed-form solution (Duchi John, 2020), hence, the objective can be efficiently implemented.

The neural network used to parameterize $p_\theta(x_{t-1}|x_t)$ can be trained to learn different quantities (Luo, 2022; Salimans & Ho, 2022; Karras et al., 2022). A possible approach is to directly output the mean μ_t of $p_\theta(x_{t-1}|x_t)$, while the variance is either fixed or learned as well. In this work, the neural network is parameterized to predict the noise added to the original sample during the forward process (eq. (18)). This is achieved by minimizing the objective $w(t)\|\epsilon - \epsilon_\theta(\alpha_t x_0 + \sigma_t \epsilon, t)\|^2$, where w is a weighting function (Ho et al., 2020). Once ϵ_θ is trained, the mean of $p_\theta(x_{t-1}|x_t)$ is computed as $\frac{1}{\alpha_t}(x_t - \sigma_t \epsilon_\theta)$ (Rombach et al., 2022). Since our primary goal is to analyze a pre-trained diffusion model, we refer the interested reader to Rombach et al. (2022); Luo (2022); Salimans & Ho (2022) for more details.

Latent diffusion. Originally, diffusion models operated directly on pixels (Ho et al., 2020; Song & Ermon, 2020). However, training a denoising network in pixel space is difficult and expensive (Hoogeboom et al., 2023). As such, Rombach et al. (2022) use a pre-trained auto-encoder to first compress images, similar to VQGAN (Esser et al., 2021), and define a diffusion process in the latent space instead of the pixel space. To make this difference clear they write $p_\theta(z_{t-1}|z_t)$, in which now z_t refers to a noisy latent instead of a noisy image.

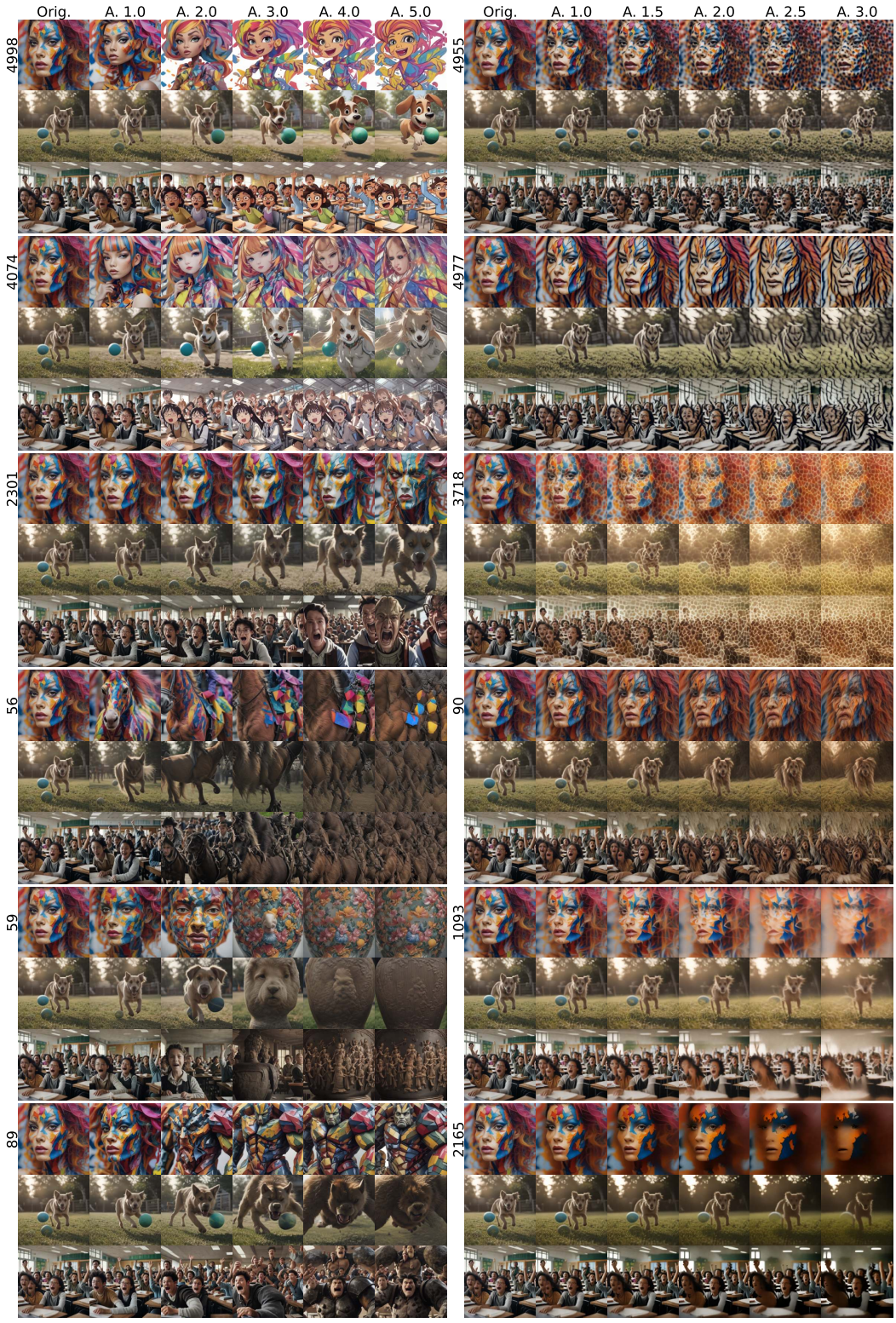
Distilled diffusion for fast inference. To speed-up inference of latent diffusion models, Sauer et al. (2023b) distill a pre-trained Stable Diffusion XL (SDXL) model (Podell et al., 2023). The distilled model is referred to as *SDXL Turbo* as it allows high-quality sampling in as little as 1-4 steps. The original SDXL model is trained with a noise schedule of 1000 steps, but in practice, sampling with 20 to 50 steps still generates high-quality images. The speed-up in SDXL Turbo is achieved through a combination of two objectives. First, Sauer et al. (2023b) define an adversarial game, similar to GANs (Goodfellow et al., 2014). The discriminator is implemented using lightweight classification heads on top of frozen features extracted at K different layers of a DINOv2 backbone (Oquab et al., 2024). Concretely, the objective of the discriminator is given by

$$\mathcal{L}^D_{\text{adv}} = \mathbb{E}_{x_0}\left[\sum_{k=1}^{K}\left(1-\mathcal{D}_k(F_k(x_0))\right)_+ + \gamma R1(\phi)\right] + \mathbb{E}_{\hat{x}_\theta}\left[\sum_{k=1}^{K}\left(1+\mathcal{D}_k(F_k(\hat{x}_\theta))\right)_+\right], \tag{19}$$

where $(x)_+ = \max(0, x)$ is the positive part, F_k denotes the k-th features tensor from the DINOv2 backbone, $\mathcal{D}_k$ the k-th classification head, ϕ is the discriminator parameters, $R1$ is an L^2 penalty term on the norm of the gradients, introduced by Mescheder et al. (2018) and γ is a scalar hyperparameter. Instead of a traditional classification loss, Sauer et al. (2023b) use the hinge loss (Lim & Ye, 2017), following Sauer et al. (2021; 2023a).

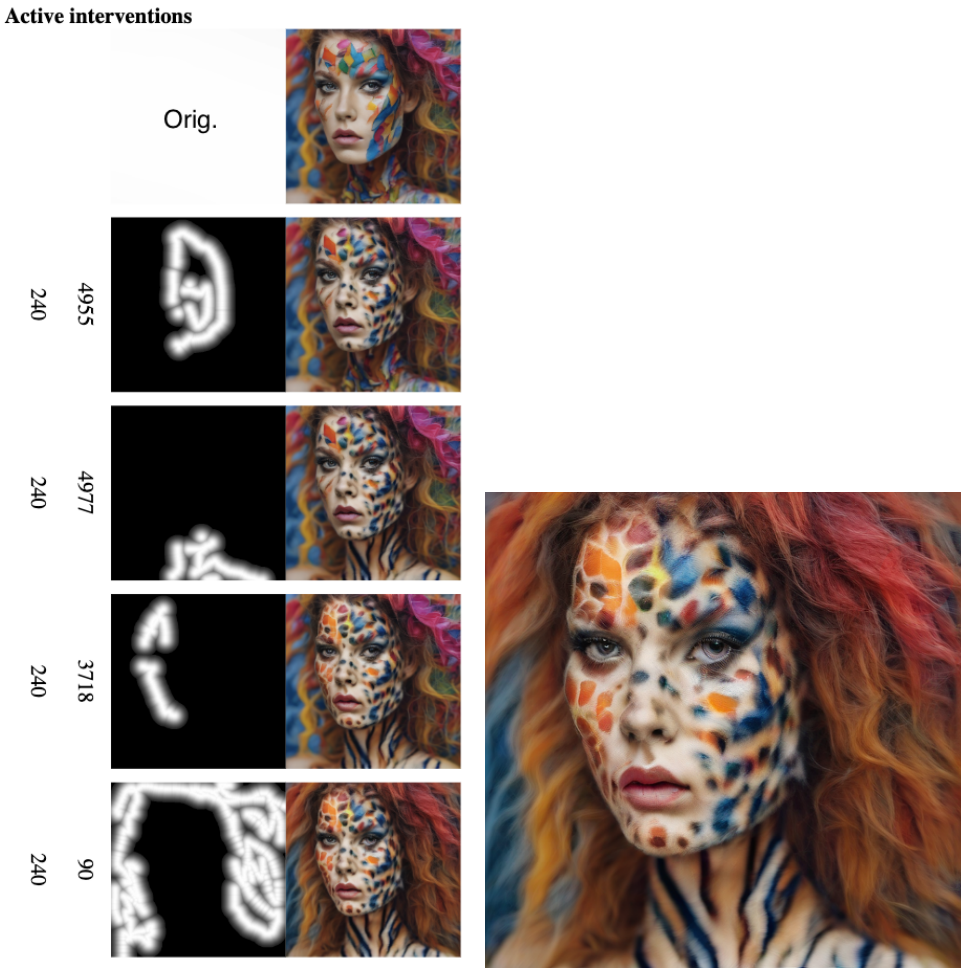
Finally, $\hat{x}_\theta$ represents the prediction of the diffusion model being distilled for the ground-truth image x_0 given a noisy sample x_t obtained through the forward diffusion process defined in eq. (18). Sauer et al. (2023b) found that distilling a diffusion model using the adversarial objective only resulted in a model with a FID of 20.8. To further improve performance, they also distilled the noise predictions of the teacher model. Importantly, both the teacher and student models were initialized with the same pre-trained weights. After adversarial distillation (Sauer et al., 2023b), the model learns to map noise to samples in one step

Neural network architecture. The denoising network of *SDXL Turbo* estimating $p_\theta(z_{t-1}|z_t)$ is implemented using a U-net similar to Rombach et al. (2022). The U-net is composed of a down-sampling path, a bottleneck, and an up-sampling path. Both the down-sampling and up-sampling paths are composed of 3 individual blocks. The individual block structure differs slightly but both down- and up-sampling blocks consist of residual layers as well as cross-attention transformer blocks. Finally, the bottleneck layer is also composed of attention and residual layers. As for the original U-net architecture (Ronneberger et al., 2015), the corresponding blocks in the up-sampling and and down-sampling path are connected via a skip connection. Importantly, the text conditioning is achieved via cross-attention to text embeddings performed by in total 11 transformer blocks embedded in the down-, up-sampling paths and bottleneck. An architecture diagram displaying the relevant blocks can be found in App. G Fig. 4.



(a) `down.2.1` (b) `up.0.1`

Figure 9: We turn on the features from Fig. 8 on three unrelated prompts "a photo of a colorful model", "a cinematic shot of a dog playing with a ball", and "a cinematic shot of a classroom with excited students".



(a) Intervention history (b) Result

Figure 10: Local edits showcase `up.0.1`'s ability to locally change textures in the image without affecting the remaining image. Multiple consecutive interventions are possible (a). The first in (a) row depicts the original image and each subsequent row we add an intervention by drawing a heatmap with a brush tool and then turning on the feature labelling the row only on that area. The other number (240) is the absolute feature strength of the edit. Figure (b) shows the final result in full resolution (512x512).

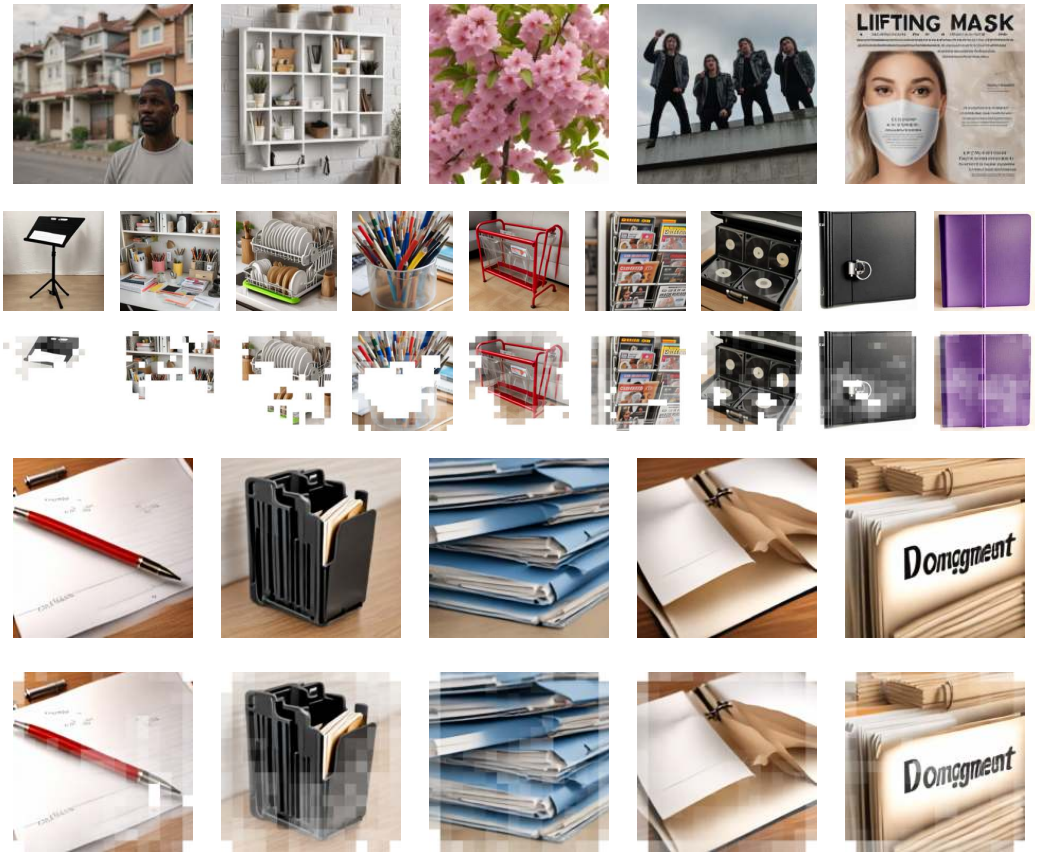


Figure 11: The images used by GPT-4o to generate captions for feature 0. From top to bottom: irrelevant images to feature 0; image progression from left to right, showing increasing activation of SAE feature 0, with low activation on the left and high activation on the right; "Coldmaps" representing the image progression; images corresponding to the highest activation of feature 0; "Coldmaps" corresponding to these highest activation images.