Finding NeMo 🐠: Localizing Neurons Responsible For Memorization in Diffusion Models

Lukas Struppek [*1 2] **Dominik Hintersdorf** [*1 2] **Kristian Kersting** [1 2 3 4] **Adam Dziedzic** [5] **Franziska Boenisch** [5]

Abstract

Diffusion models (DMs) produce very detailed and high-quality images, achieved through rigorous training on huge datasets. Unfortunately, this practice raises privacy and intellectual property concerns, as DMs can memorize and later reproduce their potentially sensitive or copyrighted training images at inference time. Prior efforts to prevent this issue are viable when the DM is developed and deployed in a secure and constantly monitored environment. However, they hold the risk of adversaries circumventing the safeguards and are not effective when the DM itself is publicly released. To solve the problem, we introduce NEMO, the first method to localize memorization of individual data samples down to the level of neurons in DMs' cross-attention layers. Through our experiments, we make the intriguing finding that in many cases, *single neurons* are responsible for memorizing particular training samples. By deactivating these *memorization neurons*, we avoid replication of training data at inference time, increase the diversity in the generated outputs, and mitigate the leakage of sensitive data.

1. Introduction

Diffusion models (DMs) have made remarkable advances in image generation. In particular, text-to-image DMs like Stable Diffusion (Rombach et al., 2022) enable the generation of complex images given a textual input prompt. Yet, DMs carry a significant risk to privacy and intellectual property since the models have been shown to generate verbatim copies of their potentially sensitive or copyrighted training data at inference time (Carlini et al., 2023; Somepalli et al., 2023). Memorization in DMs recently received a lot of attention (Gu et al., 2023; Yoon et al., 2023; Carlini et al., 2023), and several mitigations have been proposed (Wen et al., 2024; Ren et al., 2024; Somepalli et al., 2023), either excluding samples from training, monitoring inference and preventing their generation, or altering the input to the DM. While these techniques are a viable solution, they are not effective when the DMs are publicly released, such that users can freely interact with them.

As a first step to solving this problem, we propose FINDING **NE**URON ME**MO**RIZATION (NEMO), the first method for localizing where individual data samples are memorized inside the DM on the level of individual cross-attention layer neurons. Based on the insights about where within the DM individual data samples are memorized, we can prevent their verbatim output by deactivating the identified neuron(s). As demonstrated in Fig. 1, without NEMO, the memorized image is reproduced, independent of the random seed. By localizing and deactivating the neuron responsible for the memorization, we prevent the verbatim output of the training data and instead cause the generation of various non-memorized related samples. Hence, by relying on NEMO to localize and deactivate memorization neurons, we can limit memorization, which mitigates the privacy and copyright issues, while keeping the overall performance intact.

In summary, we make the following contributions:

- We propose NEMO, the first method to localize memorization on the neuron level in DMs.
- Our extensive empirical evaluation of localizing memorization within Stable Diffusion reveals that few or even single neurons are responsible for the memorization.
- We limit the memorization in DMs by deactivating the highly memorizing neurons and further show that this leads to a higher diversity in the generated outputs.

2. Related Work On Memorization in DMs

Recent empirical studies connect the model architecture, training data complexity, and the training procedure to the expected level of DM memorization (Gu et al., 2023), while

[*]Equal contribution [1]German Research Center for Artificial Intelligence (DFKI) [2]Technical University of Darmstadt [3]Hessian.AI [4]Centre for Cognitive Science, TU Darmstadt [5]CISPA Helmholtz Center for Information Security. Correspondence to: Struppek, Hintersdorf <{struppek, hintersdorf}@cs.tu-darmstadt.de>.

Published at ICML 2024 Workshop on Foundation Models in the Wild. Copyright 2024 by the author(s).

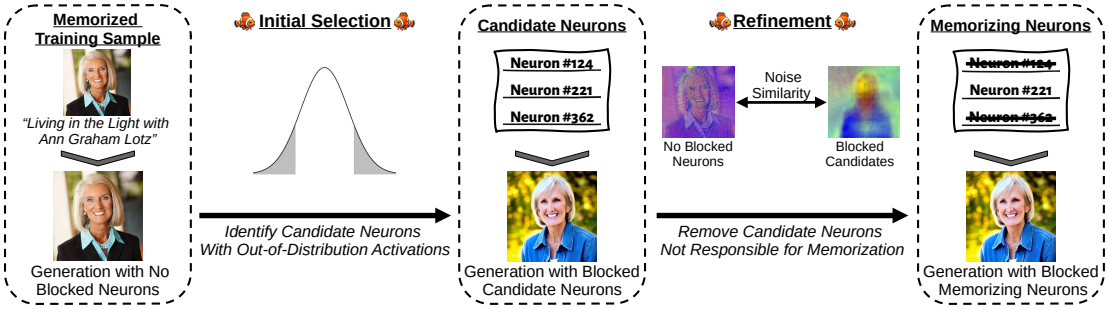
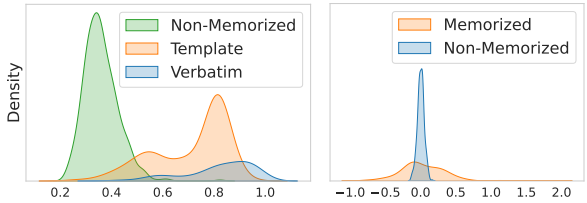


Figure 1: **Overview of NeMo**. For memorized prompts, we observe that the same (original training) image is constantly generated independently of the initial random seed. In the *initial stage*, NeMo first identifies candidate neurons potentially responsible for the memorization based on out-of-distribution activations. In a *refinement* step, NeMo detects the *memorization neurons* from the candidate set by leveraging the noise similarities during the first denoising step. Deactivating memorization neurons prevents unintended memorization and induces diversity in the generated images.

others connect memorization to the generalization of the generation process (Yoon et al., 2023). Two types of memorization are usually distinguished: Verbatim memorization that replicates the training image exactly. And template memorization that reproduces the general composition of the training image while having some non-semantic variations at fixed image positions (Webster, 2023). Existing approaches focus on the detection of memorized prompts (Wen et al., 2024; Ren et al., 2024). Our work is orthogonal to these detection methods, focusing on the exact *localization of memorization* in the DM's U-Net rather than detecting memorized samples. Previously proposed methods for mitigating memorization during inference either rescale the attention logits (Ren et al., 2024) or adjust the text embeddings with a gradient-based approach (Wen et al., 2024). However, these inference time mitigation strategies are easy to deactivate in practice and provide no permanent mitigation strategies for publicly released models. In contrast, related training-based mitigation strategies (Wen et al., 2024; Ren et al., 2024) require re-training an already trained model like Stable Diffusion, which is time- and resource-intensive. We show that NeMo can reliably identify individual neurons responsible for memorizing specific training samples. Pruning these neurons effectively mitigates memorization, does not harm the general model performance, and provides a more permanent solution to avoid training data replicating.

3. NeMo: Localizing and Removing Memorization in Diffusion Models

NeMo, our method for detecting *memorization neurons*, involves a two-step selection process, first identifying a broad set of candidate neurons that might be responsible for memorizing a specific training sample and then refining the initial candidate set by filtering neurons out. After refinement, we deactivate the remaining *memorization neurons* to remove memorization. We apply this two-step approach to detect



(a) Pairwise SSIM scores. (b) Standardized activations.

Figure 2: **Differences Between Memorized and Non-memorized Prompts. (a)** depicts the distribution of pairwise SSIM scores between initial noise differences starting from different seeds. **(b)** shows the distribution of each neuron's activation in the first cross-attention value layer. Memorization neurons produce considerably higher activations, here depicted as standardized z-scores, for memorized prompts, allowing them to be identified by outlier detection.

memorization neurons in the DM's cross-attention layers, the only components that directly process the text embeddings. Image editing research (Hertz et al., 2023; Wang et al., 2023; Chen et al., 2024) indicates that cross-attention layers highly influence the generated content, suggesting they are crucial for memorization. We analyze the impact of blocking individual key and value layers in Appx. B.7. Our results show that the value layers in the down- and mid-blocks of the U-Net have the highest memorization effect, which is why we apply NeMo only to neurons in these layers. We visualize examples of deactivating memorization neurons to prevent data replication and enhance diversity in Fig. 3. We also provide detailed algorithmic descriptions for each of NeMo's components in Appx. C.

3.1. Quantifying the Memorization Strength

Intuitively, the denoising process of DMs for memorized prompts follows a rather consistent trajectory to reconstruct the corresponding training image, resulting in generations with little diversity. Conversely, the denoising trajectory

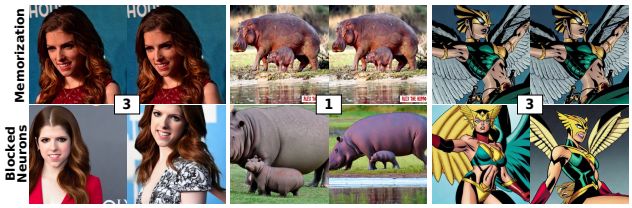


Figure 3: **Deactivating Memorization Neurons**. Images generated with memorized prompts closely replicating the training images (top row). Deactivating memorization neurons increases diversity and mitigates memorization (bottom row). Notably, only a few neurons (counts indicated by digits in the boxes) are responsible for these memorizations.

for non-memorized prompts varies significantly depending on the initial noise (Wen et al., 2024). To quantify memorization strength, we measure the similarity between the first denoising steps for different initial seeds, using this as a proxy to compare denoising trajectories. We employ the structural similarity index measure (SSIM) (Wang et al., 2004) for this purpose. Higher SSIM values indicate more consistent denoising trajectories, reflecting stronger memorization. Fig. 2a shows the distribution of SSIM scores for memorized and non-memorized prompts, demonstrating that memorized prompts lead to substantially higher scores and that SSIM values can be used to quantify the generated sample's degree of memorization. A formal description of the SSIM score is provided in Appx. A.4 and visual examples comparing the noise difference between memorized and non-memorized samples can be founding in Appx. B.5.

3.2. Localizing Memorization Neurons using NeMo

(1) Initial Selection: Our initial neuron selection procedure is based on the observation that activation patterns of memorized prompts differ from the ones of non-memorized prompts on the neuron level. Fig. 2b underlines this observation by plotting the standardized activation scores for memorized and non-memorized samples in the first value layer. Leveraging this insight, we identify memorization neurons as the ones that exhibit an out-of-distribution (OOD) activation behavior. We first identify the standard activation behavior of neurons on a separate hold-out set of non-memorized prompts. Then, we compare the activation pattern of the neurons for memorized prompts and identify neurons with OOD behavior using the standardized z-score (Kreyszig et al., 2011). The z-score quantifies the number of standard deviations by which the neuron's activation is above or below the mean activation of non-memorized prompts. To identify neurons that exhibit an out-of-distribution (OOD) activation behavior, we set an activation threshold and assume that a neuron in a specific layer has OOD behavior if its activation is higher than the activation threshold. Since some neurons have high variance in their activation patterns, even on non-memorized prompts, we additionally take the top-k neurons of each layer in addition to the neurons having OOD activations, resulting in an initial selection of neurons. A formal definition of the z-score can be found in Appx. A.5.

(2) Refinement: After identifying an initial set of memorization neurons, we proceed to filter out those not directly responsible for memorization. Initially, we deactivate all neurons identified in the initial selection and then iteratively re-activate them to observe any increase in the SSIM value, which serves as our memorization indicator. To speed up this process, we first group the identified neurons layer-wise and iterate over each layer, re-activating all neurons within this layer. If the SSIM value fails to increase beyond the SSIM threshold, we conclude that the currently activated neurons are not involved in memorization and remove them from the set of memorization neurons. After removing neurons of layers not responsible for memorization, we individually assess each remaining neuron by re-activating it while keeping all other neurons deactivated. Again, we remove neurons that do not increase the SSIM score above the SSIM threshold from our neuron set. After iterating over all neurons, we are left with those responsible for memorization.

4. Experiments

Models and Datasets: We investigate memorization in Stable Diffusion v1.4 (Rombach et al., 2022). Our set of memorized prompts consists of 500 LAION prompts (Schuhmann et al., 2022) provided by Wen et al. (2024). We further split the dataset into verbatim- (*VM*) and template-memorized (*TM*) samples. Images generated by VM prompts match the training image exact, *i.e.,* pixel-wise, while TM prompts reproduce the general composition of the training image while having some non-semantic variations at fixed image positions. Details about the analysis and the annotation can be found in Appx. B.1. We also checked for memorization in other publicly available models, like Stable Diffusion v2 and Deep Floyd (StabilityAI, 2023), but could not identify memorized prompts, which aligns with related research (Wen et al., 2024; Ren et al., 2024).

Metrics: We measure the **memorization**, **diversity**, and **quality** of the generated images with and without memorization neurons deactivated. For measuring **memorization**, we calculate the SSCD score (Pizzi et al., 2022) between the generated images and the original images (SSCD_{Gen}) or the generated images without mitigation ($\text{SSCD}_{\text{Orig}}$). Higher SSCD scores indicate a higher degree of memorization. To measure the **diversity** of the differently seeded generated images, we are computing the pairwise cosine similarity between the SSCD embeddings of different images generated by the same prompt. We refer to this metric as D_{SSCD}, for which lower values indicate more image diversity. To assess the **quality** of the images, we are calculating the Fréchet Inception Distance (FID) (Heusel et al., 2017) as well as

Table 1: **Impact of Deactivating the Memorization Neurons.** Keeping all neurons active (1st row) and randomly deactivating neurons (3rd row) has no impact on memorization. However, deactivating the memorization neurons located by NeMo (4th row) successfully mitigates memorization, increases diversity, and maintains prompt alignment. These results are comparable to the gradient-based mitigation strategies by adjusting the prompt embeddings (2nd row).

Setting	Memorization Type	Deactivated Neurons	$\downarrow SSCD_{\text{Orig}}$	$\downarrow SSCD_{\text{Gen}}$	$\downarrow D_{\text{SSCD}}$	$\uparrow A_{\text{CLIP}}$	$\downarrow FID$
All Neurons Activate (Default)	Verbatim	0	0.83 ± 0.16	1.0 ± 0.0	0.99 ± 0.01	0.32 ± 0.02	16.57
	Template	0	0.04 ± 0.04	1.0 ± 0.0	0.17 ± 0.06	0.31 ± 0.02	
Prompt Embedding Adjustment (Wen et al., 2024)	Verbatim	–	0.04 ± 0.02	0.08 ± 0.03	0.08 ± 0.03	0.30 ± 0.02	–
	Template	–	0.03 ± 0.02	0.08 ± 0.03	0.09 ± 0.03	0.31 ± 0.02	
Deactivating Random Neurons	Verbatim	4 ± 3	0.80 ± 0.11	0.999 ± 0.0	0.99 ± 0.01	0.32 ± 0.02	16.35
	Template	21 ± 18	0.05 ± 0.04	0.997 ± 0.0	0.16 ± 0.06	0.31 ± 0.02	
Deactivating Memorization Neurons (NeMo)	**Verbatim**	$\mathbf{4 \pm 3}$	$\mathbf{0.09 \pm 0.06}$	$\mathbf{0.10 \pm 0.07}$	$\mathbf{0.16 \pm 0.06}$	$\mathbf{0.31 \pm 0.02}$	**16.97**
	Template	$\mathbf{21 \pm 18}$	$\mathbf{0.05 \pm 0.03}$	$\mathbf{0.05 \pm 0.04}$	$\mathbf{0.12 \pm 0.05}$	$\mathbf{0.31 \pm 0.02}$	

the similarities between the generated images and the input prompts using CLIP (A_{CLIP}). The higher the alignment, the better the generated images represent the concepts described in the prompt, while a lower FID score denotes better image quality. More detailed information about the experiments and how we chose the thresholds can be found in Appx. A.

Results: We begin by demonstrating the effectiveness of our memorization localization method. Tab. 1 presents the quantitative results for images generated with the identified memorization neurons deactivated. NeMo detected a median of 4 and 21 memorization neurons for VM and TM prompts, respectively. For VM prompts, deactivating these memorization neurons significantly decreases memorization, as reflected by low memorization metrics $\text{SSCD}_{\text{Orig}}$ and SSCD_{Gen}, while increasing the diversity in images in terms of pairwise similarity D_{SSCD}. However, the $\text{SSCD}_{\text{Orig}}$ does not change noticeably for TM prompts. This is because TM prompts typically memorize specific parts of the original training image, such as objects or compositions, rendering the $\text{SSCD}_{\text{Orig}}$ metric less informative. In contrast, the SSCD_{Gen} score, which compares similarities between images generated with and without the deactivated neurons, provides a more accurate measure. This score highlights that deactivating the identified neurons effectively alters the images and mitigates memorization. Importantly, the image-prompt alignment A_{CLIP} remains constant in all cases, indicating that deactivating memorization neurons does not result in misguided image generations. We visualize examples of deactivating memorization neurons in Fig. 3. Additional examples can be found in Appx. B.4 and an analysis of the distribution of neurons found is available in Appx. B.2.

Comparing the results of deactivating the neurons identified by NeMo with those obtained from randomly deactivated neurons highlights that only a specific subset of neurons is actually responsible for memorizing a prompt. While deactivating the identified memorization neurons significantly impacts both memorization and the diversity of the generated images, randomly deactivating neurons has no noticeable effect. Moreover, the mitigation effect of deactivating memorization neurons is comparable to the state-of-the-art method of adjusting the prompt embeddings (Wen et al., 2024). Yet, adjusting the prompt embeddings requires gradient computations for each seed and prompt, which are time- and memory-expensive, especially with large batch sizes. In contrast, once the memorization neurons are identified using our gradient-free NeMo, no additional computations are required during image generations, thus adding no overhead to the generation process. A comparison to the mitigation method of Somepalli et al. (2023) can be found in Appx. B.8.

We further investigate how memorization neurons influence non-memorized prompts and the overall image quality. We deactivate the 750 most frequent memorization neurons and compute the FID score on the COCO dataset (Lin et al., 2014). We also repeat the generations by deactivating the same number of randomly selected neurons that are not among the identified memorization neurons. We found no noticeable degradation in the image quality when blocking the random or memorization neurons. This finding underscores the potential for pruning memorization neurons in DMs without compromising the overall image quality. More detailed FID plots and an ablation and sensitivity study, analyzing the impact of each component of NeMo, can be found in Appx. B.3 and Appx. B.6, respectively.

5. Conclusion

DMs have rapidly become a cornerstone of computer vision. Yet, problems like memorization of training samples can lead to undesired replication of potentially sensitive or copyrighted training images. Our research provides novel insights into the memorization mechanisms in text-to-image DMs. Unlike previous studies that focused on identifying memorized prompts, our approach, NeMo, is the first to *localize memorization* within the model and pinpoint *individual neurons* responsible for it. Our memorization mitigation can be executed without compromising the model's overall performance or the quality of the generated images, allowing model providers to deploy the resulting models without additional safeguards to prevent memorization.

Reproducibility Statement. Our source code is publicly available at https://github.com/ml-research/localizing_memorization_in_diffusion_models to reproduce the experiments and facilitate further analysis on memorization in diffusion models. We state all hyperparameters in the Appendix.

References

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., and Wallace, E. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270, 2023.

Chen, M., Laina, I., and Vedaldi, A. Training-free layout control with cross-attention guidance. In *Winter Conference on Applications of Computer Vision (WACV)*, pp. 5331–5341, 2024.

Gu, X., Du, C., Pang, T., Li, C., Lin, M., and Wang, Y. On Memorization in Diffusion Models. *arXiv preprint*, arXiv:2310.02664, 2023.

Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-Or, D. Prompt-to-prompt image editing with cross-attention control. In *Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 6629–6640, 2017.

Kreyszig, E., Kreyszig, H., and Norminton, E. J. *Advanced Engineering Mathematics*. Wiley, tenth edition, 2011.

Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

Pizzi, E., Roy, S. D., Ravindra, S. N., Goyal, P., and Douze, M. A self-supervised descriptor for image copy detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14512–14522, 2022.

Ren, J., Li, Y., Zeng, S., Xu, H., Lyu, L., Xing, Y., and Tang, J. Unveiling and Mitigating Memorization in Text-to-image Diffusion Models through Cross Attention. *arXiv preprint*, arXiv:2403.11052, 2024.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S., Crowson, K., Schmidt, L., Kaczmarczyk, R., and Jitsev, J. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Understanding and mitigating copying in diffusion models. *Advances in Neural Information Processing Systems*, 36:47783–47803, 2023.

StabilityAI. DeepFloyd IF: a novel state-of-the-art open-source text-to-image model with a high degree of photorealism and language understanding. https://www.deepfloyd.ai/deepfloyd-if, 2023. Retrieved on 2023-11-08.

Wang, K., Yang, F., Yang, S., Butt, M. A., and van de Weijer, J. Dynamic prompt learning: Addressing cross-attention leakage for text-based image editing. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

Webster, R. A reproducible extraction of training images from diffusion models. *arXiv preprint*, arXiv:2305.08694, 2023.

Wen, Y., Liu, Y., Chen, C., and Lyu, L. Detecting, Explaining, and Mitigating Memorization in Diffusion Models. In *International Conference on Learning Representations (ICLR)*, 2024.

Yoon, T., Choi, J. Y., Kwon, S., and Ryu, E. K. Diffusion Probabilistic Models Generalize when They Fail to Memorize. In *ICML Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023.

A. Experimental Details

A.1. Hard- and Software Details

We performed all our experiments on NVIDIA DGX machines running NVIDIA DGX Server Version 5.2.0 and Ubuntu 20.04.5 LTS. The machines have 1.5 TB (machine 1) and 2 TB (machine 2) of RAM and contain NVIDIA Tesla V100 SXM3 32GB (machine 1) NVIDIA A100-SXM4-40GB (machine 2) GPUs with Intel(R) Xeon(R) Platinum 8174 (machine 1) and AMD EPYC 7742 64-core (machine 2) CPUs. We further relied on CUDA 12.1, Python 3.10.13, and PyTorch 2.2.2 with Torchvision 0.17.2 (Paszke et al., 2019) for our experiments. All investigated models are publicly available on Hugging Face. For access, we used the Hugging Face diffusers library with version 0.27.1.

We provide a Dockerfile with our code to make reproducing our results easier. In addition, all training and configuration files are available to reproduce the results stated in this paper.

A.2. Model and Dataset Details

Experiments were mainly conducted on Stable Diffusion v1-4 (Rombach et al., 2022), publicly available at `https://huggingface.co/CompVis/stable-diffusion-v1-4`. All details regarding the data, training parameters, limitations, and environmental impact are available at that URL. The model is available under the CreativeML OpenRAIL M license.

The investigated prompts originate from the LAION2B-en (Schuhmann et al., 2022) dataset used to train the `DM`. The set of memorized prompts is taken from Wen et al. (2024)[1], who collected the prompts by using the tool of Webster (2023). The LAION dataset itself is licensed under the Creative Common CC-BY 4.0. The images of the LAION dataset might be under copyright, so we do not include them in our code base; we only provide URLs to retrieve the images directly from their source.

A.3. Experimental Details and Hyperparameters

All images depicted throughout the paper are generated with fixed seeds, 50 inference steps, and a classifier-free guidance strength of 7 using the default DDIM scheduler. Notably, the seeds used for generating the images and computing the evaluation metrics differ from those used for our detection method NeMo to avoid seed overfitting.

During detection with NeMo, no classifier-free guidance was used, which speeds up the detection since only a single forward pass per seed is required, compared to an additional forward pass on the null-text embedding with classifier-free guidance. We always used ten different seeds for each prompt. The threshold on the SSIM memorization score was set to $\tau_{\text{mem}} = 0.428$ during the experiments in the main paper. We vary this threshold and analyze its impact in our sensitivity analysis in Appx. B.6.

We run all experiments – detection with NeMo and the generations for the metric computations – with half-precision (float16) to reduce the memory consumption and speed up the computations.

A.4. Structural Similarity Index Measure (SSIM)

We quantify the memorization strength during our experiments using the structural similarity index measure commonly used in the computer vision domain to assess the similarity between image pairs. Our memorization score is computed as follows: Let $x_T \sim N(\mathbf{0}, \mathbf{I})$ be the initial noisy image. Let $\epsilon_\theta(x_T, T, y)$ further denote the initial noise prediction without any scaling by the scheduler. We found that the normalized difference between the initial noise and the first noise prediction $\delta = \epsilon_\theta(x_T, T, y) - x_T$ for memorized prompts is substantially more consistent for different seeds than for non-memorized prompts. To detect the degree of memorization, we, therefore, use the similarity between the noise differences $\delta^{(i)}$ and $\delta^{(j)}$ generated with seeds i and j as a proxy. We measure the similarity with the common structural similarity index measure (SSIM) (Wang et al., 2004). The SSIM $\in [-1, 1]$ between two noise differences $\delta^{(i)}$ and $\delta^{(j)}$ is defined by

$$\text{SSIM}(\delta^{(i)}, \delta^{(j)}) = \frac{(2\mu_i\mu_j + C_1)(2\sigma_{ij} + C_2)}{(\mu_i^2 + \mu_j^2 + C_1)(\sigma_i^2 + \sigma_y^2 + C_2)}. \tag{1}$$

[1] Available at `https://github.com/YuxinWenRick/diffusion_memorization`.

The parameters μ_i, μ_j and σ_i^2, σ_j^2 denote the mean and variance of the pixels in $\delta^{(i)}$ an $\delta^{(j)}$, respectively. Likewise, σ_{xy} denotes the covariance between the images. Following the original paper, C_1 and C_2 are small constants added for numerical stability.

A higher SSIM indicates higher similarity between the noise differences, reflecting a higher degree of memorization. Notably, the SSIM computation only requires a single denoising step per seed, which makes the process fast. To set a memorization threshold τ_{mem}, starting from which we define a sample as memorized, we first compute the mean SSIM on a holdout set of non-memorized prompts. We compute the pairwise SSIM between ten different initial noise samples for each prompt and take the maximum score. After that, we average the scores across all prompts and set the threshold τ_{mem} to the mean plus one standard deviation. We consider the current image generation non-memorized if the maximum pairwise SSIM scores are below this threshold τ_{mem}.

A.5. Z-Score

We identify memorization neurons as the ones that exhibit an out-of-distribution (OOD) activation behavior. We first identify the standard activation behavior of neurons on a separate hold-out set of non-memorized prompts. Then, we compare the activation pattern of the neurons for memorized prompts and identify neurons with OOD behavior. Let the cross-attention value layers of a `DM` be $l \in \{1, \ldots, L\}$. We denote the activation of the i-th neuron in the l-th layer for prompt y as $a_i^l(y)$. Let μ_i^l be the pre-computed mean activation and σ_i^l the corresponding standard deviation for this neuron. To detect neurons potentially responsible for the memorization of a memorized prompt y, we compute the standardized z-score (Kreyszig et al., 2011), defined as

$$z_i^l(y) = \frac{a_i^l(y) - \mu_i^l}{\sigma_i^l}\,. \tag{2}$$

The z-score quantifies the number of standard deviations σ_i^l by which the activation $a_i^l(y)$ is above or below the mean activation μ_i^l. To identify a neuron as exhibiting an OOD activation behavior, we set a threshold θ_{act} and assume that neuron i in layer l has OOD behavior if $|z_i^l(y)| > \theta_{\text{act}}$. The lower the threshold θ_{act}, the more neurons are labeled as OOD and added to the *memorization neuron* candidate set.

To get an initial selection of *memorization neurons*, we calculate the standardized z-scores for all neurons and start with a relatively high value of $\theta_{\text{act}} = 5$. We deactivate all neurons with OOD activations given the current threshold θ_{act}, i.e., setting the output of a neuron to 0 if $|z_i^l(y)| > \theta_{\text{act}}$, to reduce the memorization strength. If, after deactivating these neurons, the memorization score is not below the threshold τ_{mem}, we then iteratively decrease the activation threshold θ_{act} by 0.25 and update the candidate set until the target memorization score τ_{mem} is reached.

B. Additional Results

B.1. Distinguishing Between Different Types of Memorization

Webster (2023) distinguished between *verbatim* and *template memorized* prompts. Verbatim memorized prompts lead to the exact reconstruction of training samples, while template-memorized prompts replicate the composition and structure of the training image. To provide a more fine-grained analysis of our results, we classify the prompts in our dataset into these two categories. We distinguish between both types by computing the SSCD (Pizzi et al., 2022) scores between the original training image and ten generations with different seeds. We then classify a prompt as *verbatim memorized* if the maximum SSCD score computed as cosine similarity exceeds a threshold of 0.7 and as *template memorized* otherwise. Fig. 4 plots the distribution of SSCD scores for both datasets. We manually inspected and classified the prompts where the original training image is no longer available (16 out of 500).

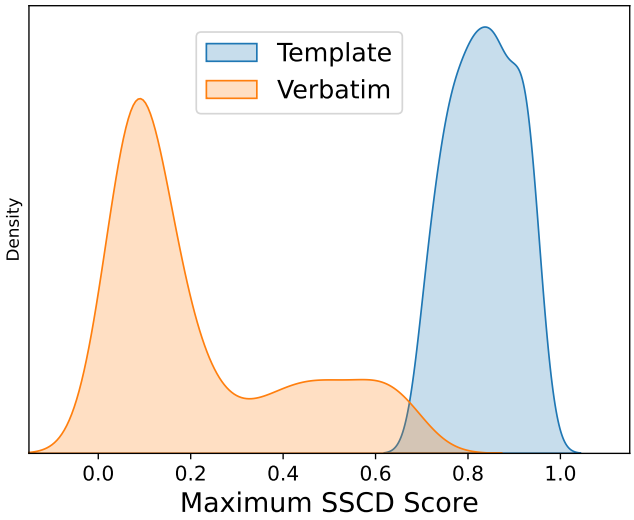


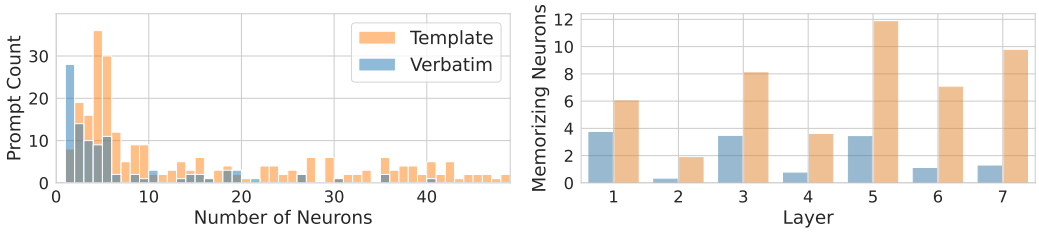
Figure 4: We compare the maximum SSCD score between ten generated images and the original training sample. We categorize the memorized prompts into verbatim memorized if the SSCD score exceeds 0.7 and into template memorized prompts otherwise.

B.2. Detailed Analysis of the Distribution of Memorization Neurons

Fig. 5a shows the total number of neurons responsible for memorizing specific prompts. Typically, a small set of neurons is responsible for verbatim memorization. For instance, 28 VM prompts from our dataset are memorized by a *single neuron*. Additionally, five or fewer neurons replicate two-thirds of VM images, indicating that verbatim memorization can often be precisely localized within the model. Template memorization can also frequently be pinpointed to a small set of neurons, with about 30% of TV replication triggered by five or fewer neurons. However, approximately one-third of TM prompts are distributed across 50 or more neurons. We hypothesize that this broader distribution results from the higher variation in generated images for TM prompts, where memorization spread across multiple neurons leads to increased image diversity. In contrast, VM prompts, often memorized by a small group of neurons, consistently produce the same image without variation. More detailed plots of the identified neurons can be found in Fig. 6.

Interestingly, we identify two neurons in the first cross-attention value layer responsible for the verbatim memorization of multiple prompts. Neuron #25 in this layer is associated with depicting people, while neuron #221 is responsible for memorizing multiple podcast covers. Together, these neurons account for memorizing 17% of our dataset's VM prompts. Similarly, neurons #507 and #517 in the third value layer are responsible for multiple TM prompts describing iPhone cases. The impact of deactivating these neurons on the image generation of memorized prompts is visualized in Appx. B.4. We also plot the distribution of the average layer-wise number of memorization neurons per prompt in Fig. 5b. Neurons responsible for VM prompts are primarily located in the value mappings of the first cross-attention layers within the U-Net's down-blocks (each block contains two cross-attention layers). A similar pattern appears for TM prompts, although value layers located deeper in the U-Net seem to play a more crucial role for TM prompts than for VM prompts.

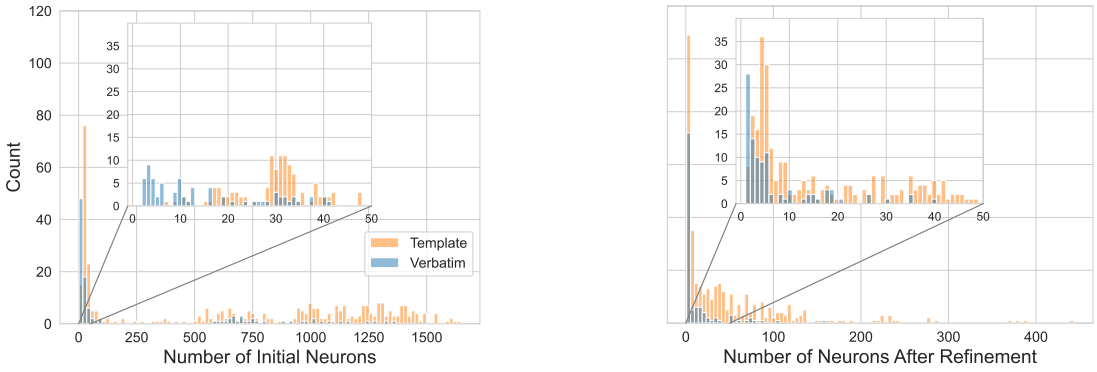
Highly Memorized Prompts: For certain memorized prompts, our method identifies a set of over 100 neurons. Upon closer examination, we found that in these cases, memorization is distributed across many neurons in various layers rather than being concentrated in a small group. Even deactivating a substantial number of neurons in the network does not eliminate memorization for these prompts. However, such instances, primarily TM prompts, were rare in our experiments. We provide examples of highly memorized prompts in Fig. 7.



(a) Memorization neurons per prompt

(b) Memorization neurons per layer.

Figure 5: **Distribution of Memorization Neurons. (a)** shows the number of prompts that are memorized by a fixed number of neurons, e.g., the verbatim memorization of 28 prompts is located in single neurons. **(b)** depicts the average number of memorization neurons per layer and prompt.



(a) Number of initial neurons per prompt.

(b) Number of neurons per prompt after refinement.

Figure 6: **Distribution of Memorization Neurons**. We show the number of prompts that are memorized by a fixed number of neurons. **(a)** plots the number of neurons found in the initial neuron selection. **(b)** shows the number of neurons after refinement. As we can observe, the refinement step drastically reduces the number of found memorization neurons for both the template and the verbatim memorized prompts.

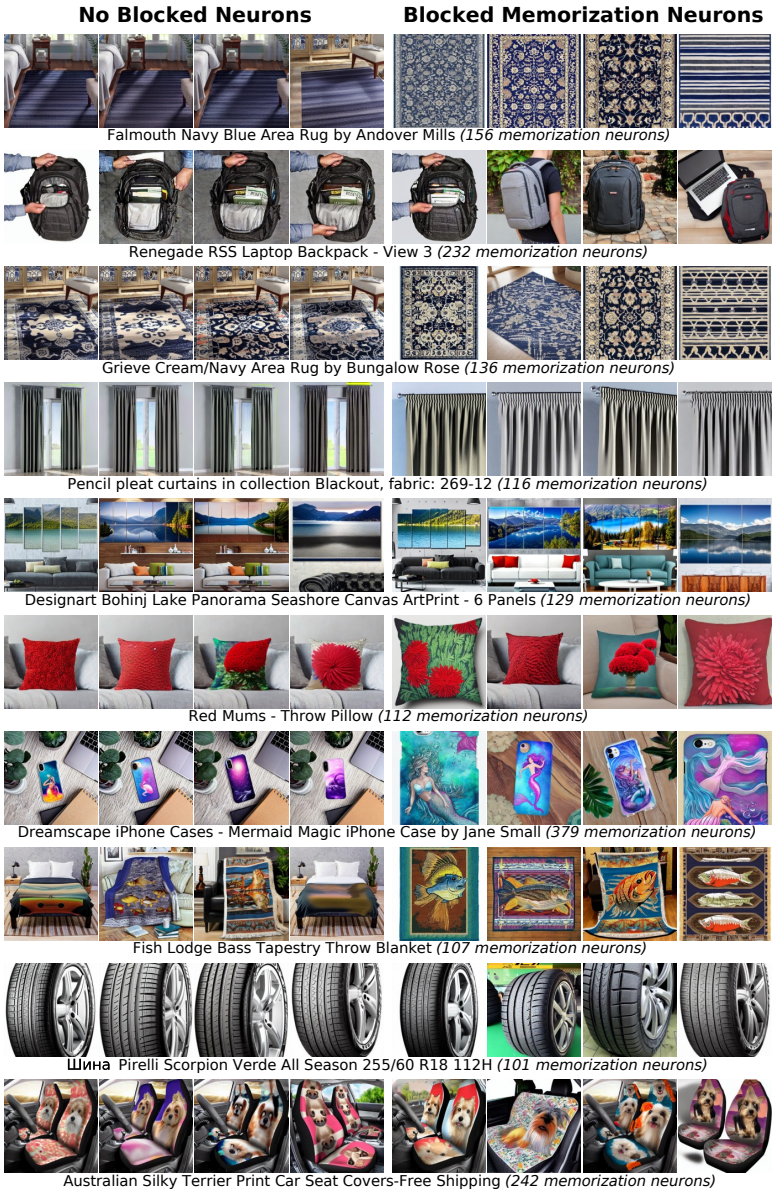
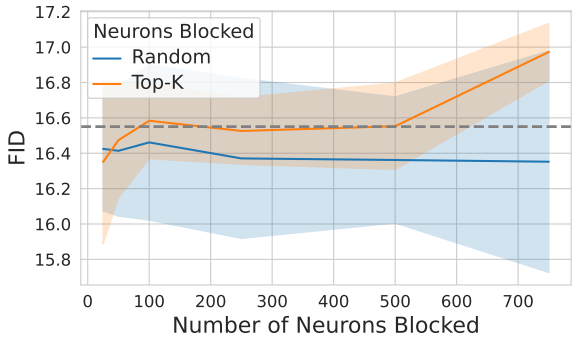
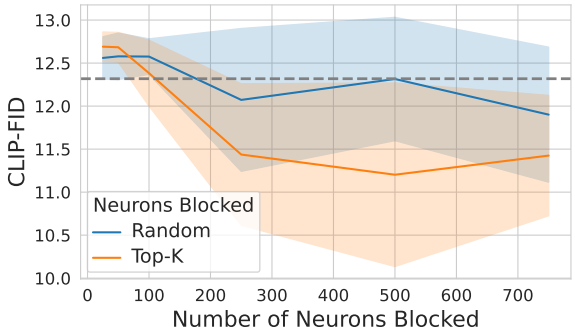


Figure 7: Memorization of highly memorized prompts is distributed across many neurons in various layers, rather than concentrated in a small group of neurons. We show examples of such prompts and the impact of deactivating the identified memorization neurons. The number of memorization neurons in each case is stated behind each prompt.

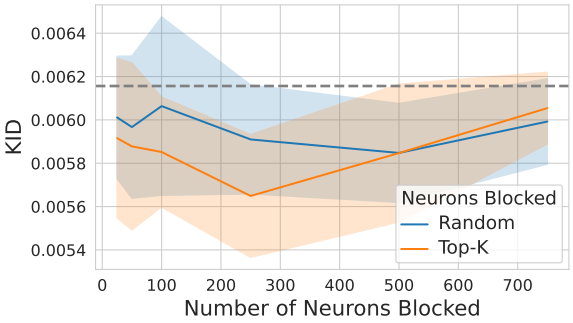
B.3. Detailed Quality Metric Plots



(a) Assessing image quality using FID.



(b) Assessing image quality using CLIP-FID.



(c) Assessing image quality using KID.

Figure 8: **Image Quality Does Not Degrade When Deactivating Memorizing Neurons**. Depicted are the generated images' FID, CLIP-FID (FID calculated using a CLIP model), and KID scores when blocking an increasing number of neurons. For all three metrics, smaller values are better. As can be seen, the FID, KID, and CLIP-FID values vary only slightly, indicating that blocking neurons identified by NeMo does not negatively affect image generation quality. Gray lines indicate baselines without any neurons blocked. We repeated the experiment with five different seeds. Depicted are the mean values and the standard deviation.

B.4. Examples for Memorization of Single Neurons

To illustrate that some single neurons are responsible for memorizing multiple training prompts, we generated images with and without these specific neurons deactivated. In Fig. 9 and Fig. 10, we only deactivate a single neuron each in the first value layer, whereas in Fig. 11, we deactivate two neurons in the third value layer.

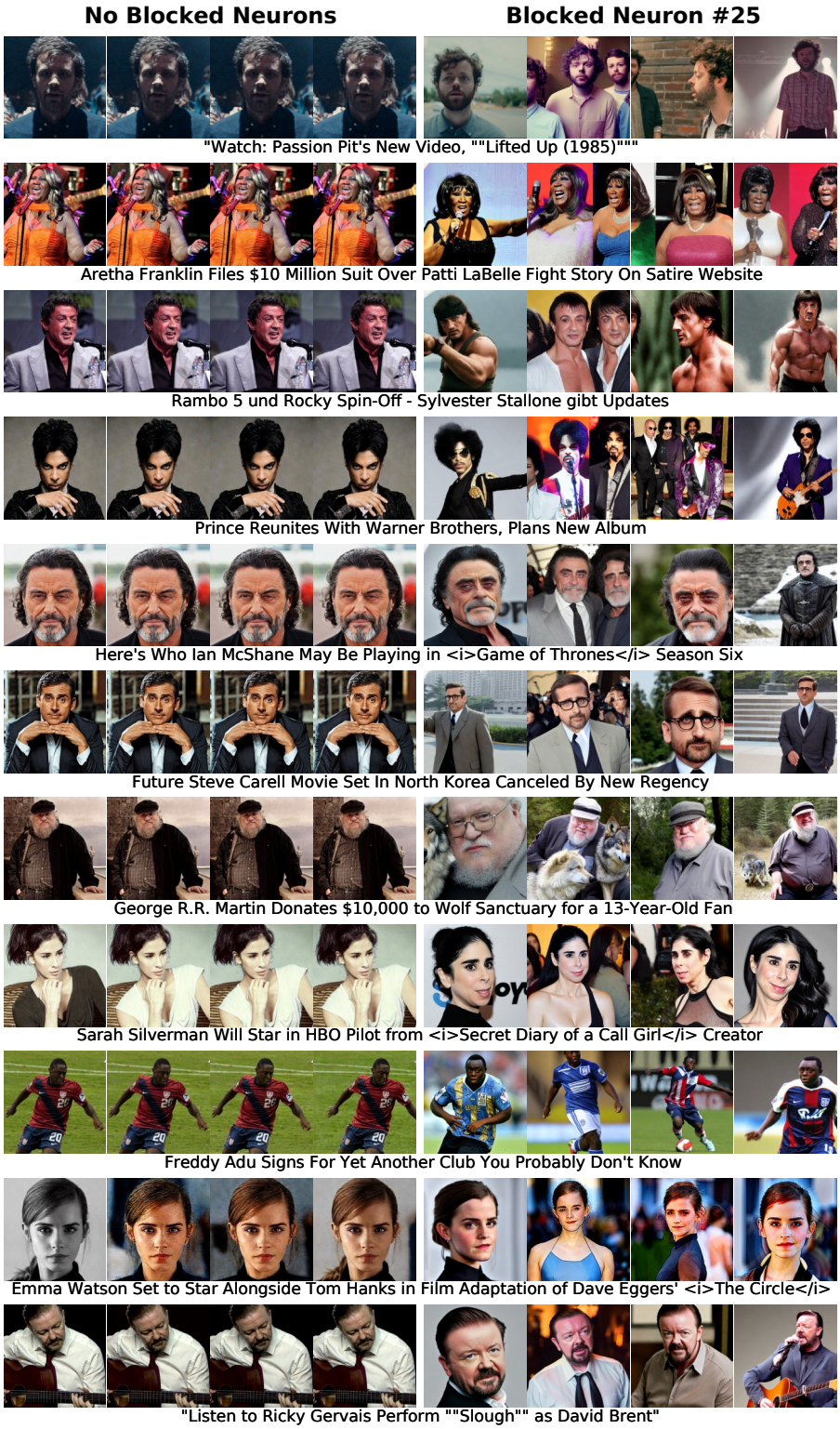


Figure 9: We found that **neuron #25 in the first cross-attention layer's value mapping** is responsible for verbatim memorization of multiple prompts, all associated with depicting people. Deactivating this single neuron mitigates the memorization and introduces diversity into the images (right columns) compared to images generated with all neurons active (left columns). Generations were conducted with seeds different from the seeds used for the neuron localization process.

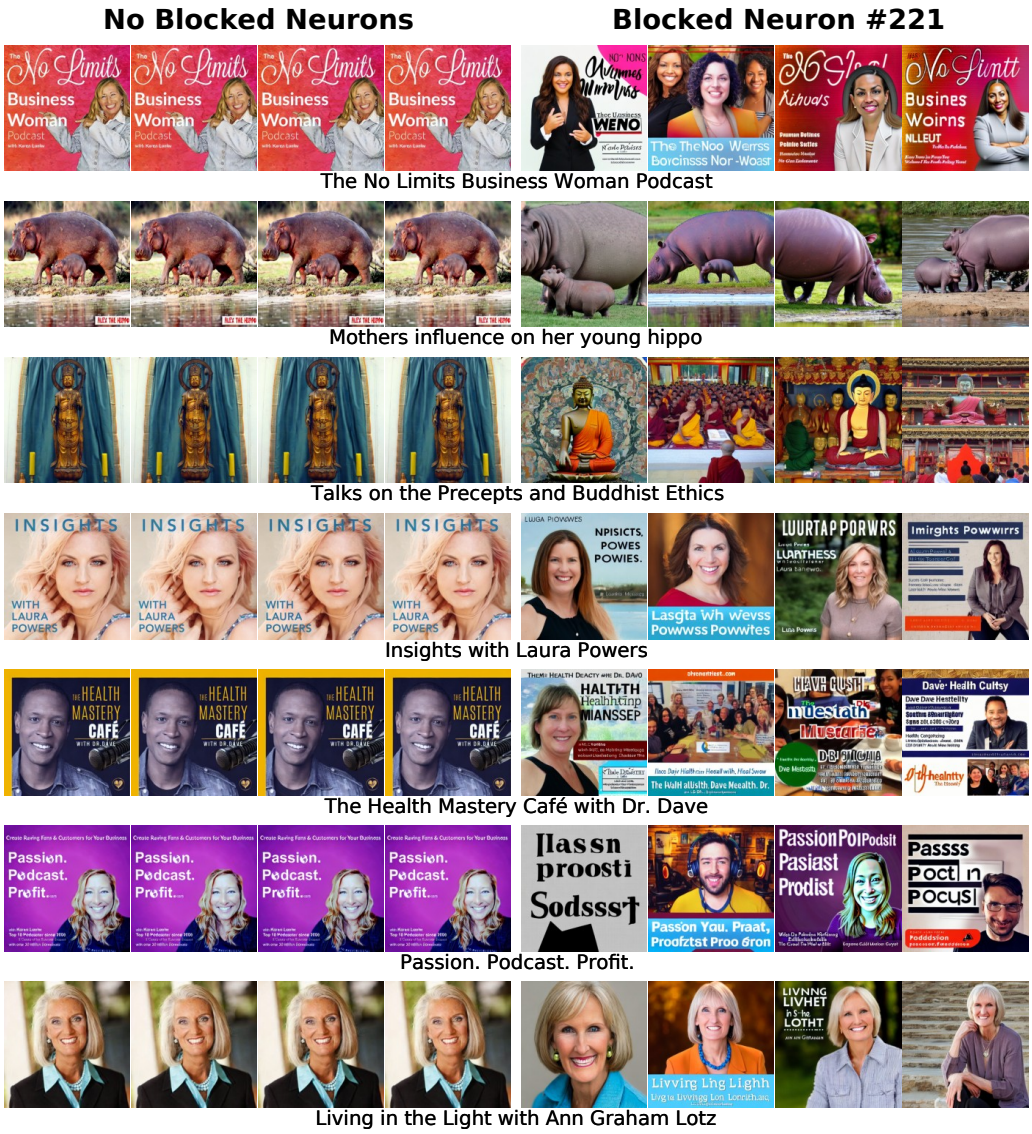


Figure 10: We found that **neuron #25 in the first cross-attention layer's value mapping** is responsible for verbatim memorization of multiple prompts, all associated with depicting people. Deactivating this single neuron mitigates the memorization and introduces diversity into the images (right columns) compared to images generated with all neurons active (left columns). Generations were conducted with seeds different from the seeds used for the neuron localization process.

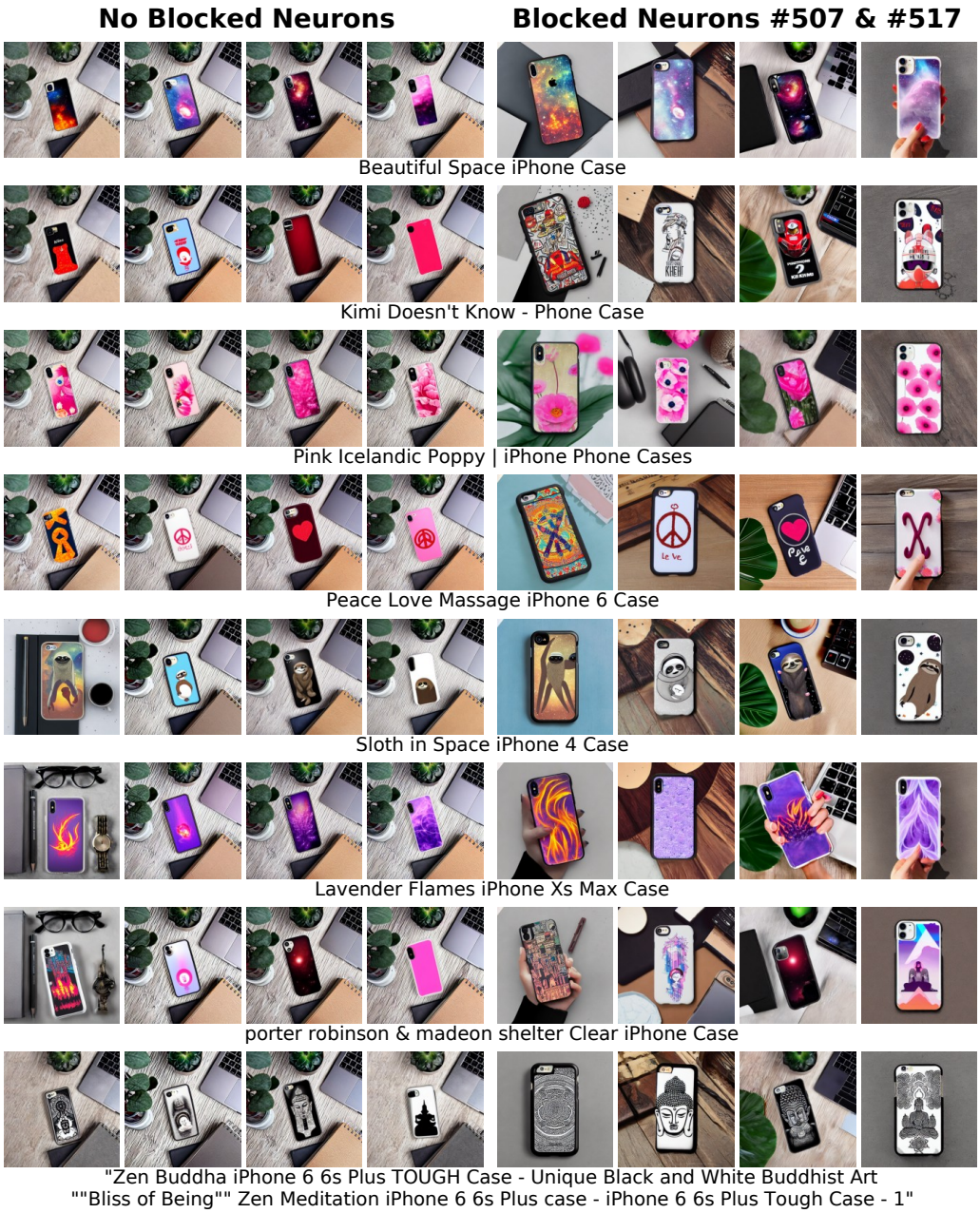


Figure 11: We found that **neurons #507 and #517 in the third cross-attention layer's value mapping** is responsible for template memorization of multiple prompts describing iPhone cases. Deactivating these two neurons mitigates the template memorization and introduces diversity into the images (right columns) compared to images generated with all neurons active (left columns). Image generations were conducted with a fixed seed different from the seed used for the neuron localization process.

B.5. Comparison of Initial Noise Predictions

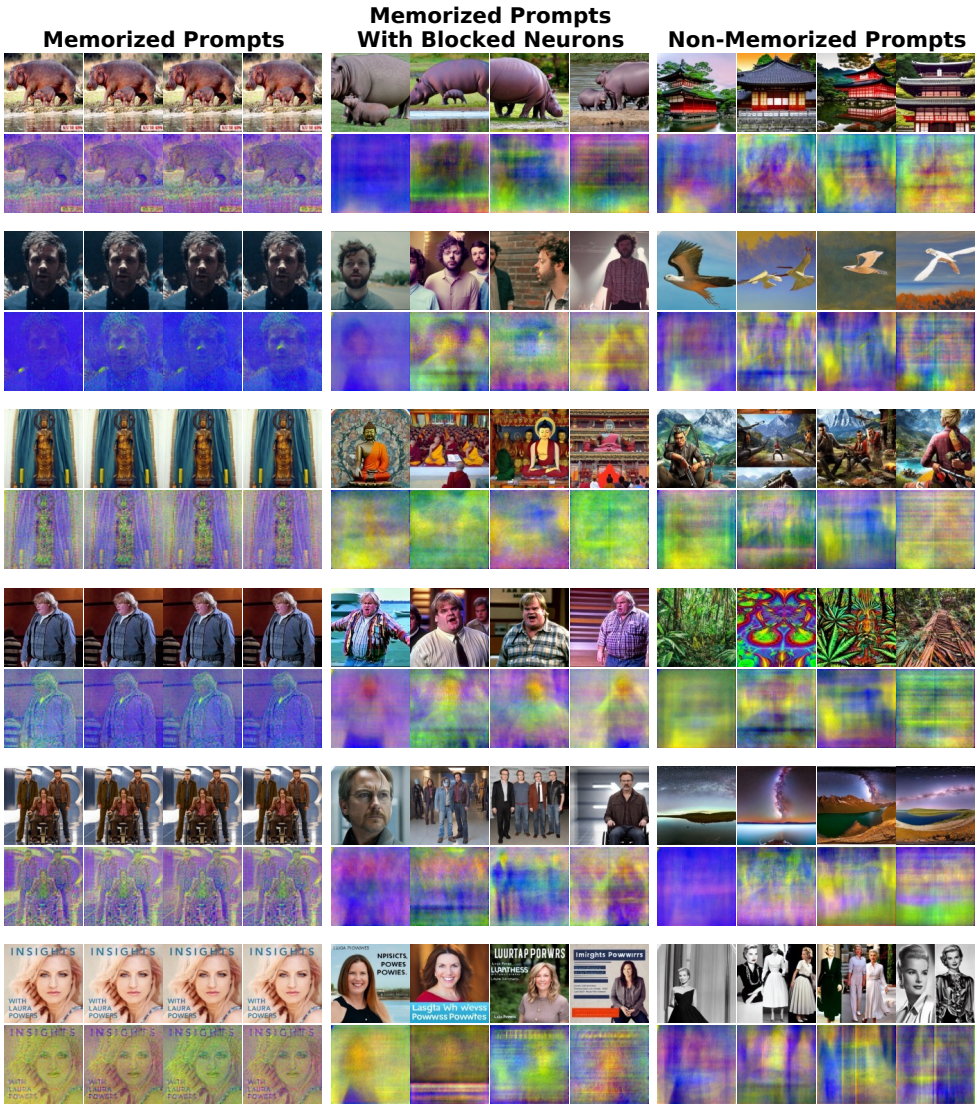


Figure 12: Visualizations for generated images and the **noise differences between the predicted noise after the first denoising step and the initial Gaussian noise**. Noise differences for memorized prompts (left column) have low diversity and are already structurally similar to the final image. The noise differences for non-memorized prompts (right column) show no clear structure and differ substantially for different noise initializations. Deactivating the memorization neurons detected with NeMo (middle column) removes the structure in the initial noise differences and adds more diversity, leading to diverse image generations.

B.6. Ablation Study and Sensitivity Analysis

We conduct an ablation study to investigate the impact of the individual components of NeMo. Additionally, we analyze the sensitivity of the memorization threshold τ_{mem} and explore alternatives to deactivating neurons by setting their activations to zero. The results for the various settings are presented in Tab. 2.

The first two rows provide evaluation results for the model with *all neurons active* and with *randomly deactivated neurons*. Both scenarios exhibit strong memorization. The third row shows the results of blocking the neurons identified as memorizing by NeMo, using the threshold $\tau_{\text{mem}} = 0.428$ as specified in the main paper. This threshold corresponds to the mean SSIM memorization plus one standard deviation, calculated on a holdout set of 50,000 non-memorized LAION prompts. In row four, we repeat this setting using classifier-free guidance (*CFG*) with a guidance strength of 7.0, as opposed to our default setting without CFG. Detection with CFG further reduces the number of detected memorizing neurons. However, the SSCD scores indicate slightly increased memorization after deactivating the identified neurons. Additionally, running NeMo with CFG doubles the number of forward passes in the U-Net since a separate noise prediction is generated for each initial seed.

Rows five to seven display the results of *varying the memorization threshold* τ_{mem}. Specifically, we adjust the threshold to one standard deviation below the mean SSIM score, to the mean, and two standard deviations above the mean. A lower threshold identifies more neurons. However, for lower thresholds, the metrics are comparable to those obtained with our default threshold value ($\tau_{\text{mem}} = 0.428$). Increasing the threshold reduces the number of identified neurons but slightly increases memorization, as measured by the SSCD scores. Thus, a trade-off exists between reducing the number of identified memorization neurons and their memorization mitigation effect. In addition, we provide heat maps to directly compare the impact of different thresholds τ_{mem} used during the initial selection and the refinement step in Appx. B.6. Appx. B.6 further compares the SSCD scores for varying the threshold values.

Rows eight and nine examine the impact of *removing the refinement step* or incorporating *no neurons with top-k activations* during the initial selection. As anticipated, without refinement, the number of identified neurons increases substantially. Despite this, the various metrics remain comparable to those obtained after the refinement step, even with more neurons deactivated. This underscores the robustness of image generations against pruning out-of-distribution (OOD) neurons. Without the top-k selection, NeMo identifies a larger set of neurons. However, deactivating these neurons does not mitigate memorization as effectively as with a top-k search. Notably, for template verbatim prompts, the SSCD_{Gen} is substantially higher without top-k, indicating increased memorization.

In the remaining rows, we explore the impact of *scaling the activations of memorization neurons* instead of deactivating them. With negative scaling factors, the results are comparable to those of completely deactivating the neurons. For positive scaling factors, however, the generated images demonstrate higher degrees of memorization, with a scaling factor of 0.75 having almost no influence on memorization. A plot of the SSCD values can be seen in Fig. 13.

We also apply NeMo to a set of 500 LAION *non-memorized prompts*, different from the 50,000 prompts used to set the memorization threshold. For 442 of these prompts, NeMo identified no memorization neurons, which is to be expected since these prompts show no memorization behavior. For the remaining prompts, a median of 62 ± 27 neurons was found.

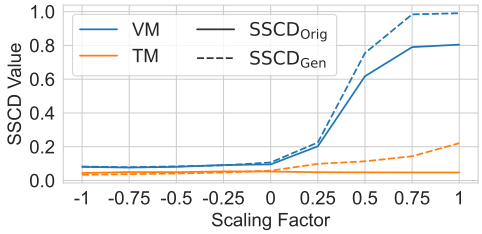
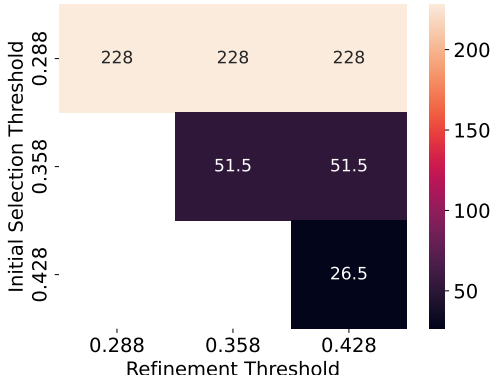


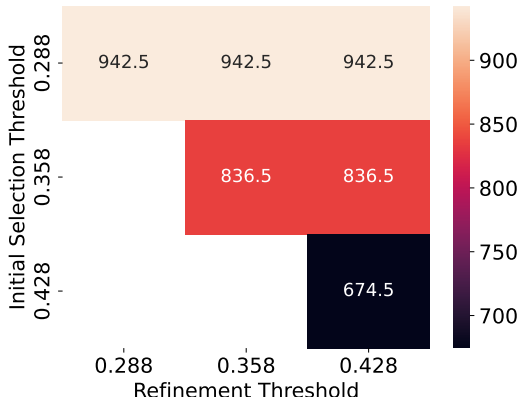
Figure 13: Negatively scaling the activations of memorization neurons instead of deactivating them (scaling by zero) has no benefit. Whereas positively scaling memorization neuron activations only slightly reduces memorization, negative scaling reduces the memorization not any further.

Table 2: **Quantitative Results of Our Ablation Study and Sensitivity Analysis.**

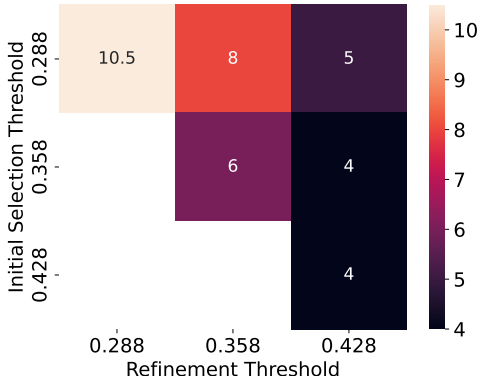
Setting	Memorization Type	Deactivated Neurons	$\downarrow SSCD_{\text{Orig}}$	$\downarrow SSCD_{\text{Gen}}$	$\downarrow D_{\text{SSCD}}$	$\uparrow A_{\text{CLIP}}$
All Neurons Activate (Default)	Verbatim	0	0.83 ± 0.16	1.0 ± 0.0	0.99 ± 0.01	0.32 ± 0.02
	Template	0	0.04 ± 0.04	1.0 ± 0.0	0.17 ± 0.06	0.31 ± 0.02
Deactivating Random Neurons	Verbatim	4 ± 3	0.80 ± 0.11	0.999 ± 0.0	0.99 ± 0.01	0.32 ± 0.02
	Template	21 ± 18	0.05 ± 0.04	0.997 ± 0.0	0.16 ± 0.06	0.31 ± 0.02
Default Values ($\tau_{\text{mem}} = 0.428$)	Verbatim	4 ± 3	0.09 ± 0.06	0.10 ± 0.07	0.16 ± 0.06	0.31 ± 0.02
	Template	21 ± 18	0.05 ± 0.03	0.05 ± 0.04	0.12 ± 0.05	0.31 ± 0.02
With Classifier-Free Guidance	Verbatim	3 ± 2	0.09 ± 0.06	0.14 ± 0.07	0.15 ± 0.06	0.31 ± 0.02
	Template	6 ± 5	0.05 ± 0.03	0.12 ± 0.07	0.11 ± 0.04	0.32 ± 0.02
$\tau_{\text{mem}} = \mu - 1\sigma = 0.288$	Verbatim	10.5 ± 9.5	0.08 ± 0.06	0.14 ± 0.06	0.15 ± 0.05	0.32 ± 0.02
	Template	32.0 ± 27	0.06 ± 0.03	0.10 ± 0.07	0.14 ± 0.05	0.31 ± 0.03
$\tau_{\text{mem}} = \mu = 0.358$	Verbatim	6 ± 5	0.10 ± 0.06	0.14 ± 0.07	0.16 ± 0.05	0.31 ± 0.02
	Template	30 ± 25	0.06 ± 0.03	0.11 ± 0.07	0.13 ± 0.04	0.31 ± 0.02
$\tau_{\text{mem}} = \mu + 2\sigma = 0.498$	Verbatim	3 ± 2	0.10 ± 0.06	0.15 ± 0.07	0.15 ± 0.06	0.32 ± 0.02
	Template	7 ± 6	0.06 ± 0.03	0.12 ± 0.04	0.12 ± 0.04	0.31 ± 0.03
No Refinement	Verbatim	26.5 ± 22.5	0.07 ± 0.05	0.11 ± 0.06	0.15 ± 0.06	0.32 ± 0.02
	Template	674.5 ± 624.5	0.04 ± 0.03	0.09 ± 0.05	0.13 ± 0.04	0.31 ± 0.02
No top-k Selection	Verbatim	11 ± 10	0.11 ± 0.05	0.21 ± 0.13	0.16 ± 0.04	0.32 ± 0.02
	Template	30 ± 23	0.05 ± 0.03	0.41 ± 0.32	0.13 ± 0.03	0.31 ± 0.02
Scaling Factor 0.75	Verbatim	4 ± 3	0.79 ± 0.12	0.995 ± 0.00	0.96 ± 0.04	0.32 ± 0.02
	Template	21 ± 18	0.05 ± 0.04	0.966 ± 0.03	0.15 ± 0.05	0.31 ± 0.02
Scaling Factor 0.5	Verbatim	4 ± 3	0.62 ± 0.29	0.97 ± 0.02	0.67 ± 0.33	0.32 ± 0.01
	Template	21 ± 18	0.05 ± 0.03	0.83 ± 0.15	0.14 ± 0.04	0.31 ± 0.02
Scaling Factor 0.25	Verbatim	4 ± 3	0.20 ± 0.17	0.32 ± 0.25	0.21 ± 0.12	0.32 ± 0.02
	Template	21 ± 18	0.05 ± 0.03	0.23 ± 0.16	0.12 ± 0.04	0.32 ± 0.02
Scaling Factor -0.25	Verbatim	4 ± 3	0.09 ± 0.06	0.12 ± 0.06	0.15 ± 0.05	0.32 ± 0.02
	Template	21 ± 18	0.05 ± 0.03	0.09 ± 0.06	0.13 ± 0.04	0.31 ± 0.02
Scaling Factor -0.5	Verbatim	4 ± 3	0.08 ± 0.05	0.12 ± 0.06	0.17 ± 0.06	0.31 ± 0.02
	Template	21 ± 18	0.05 ± 0.03	0.08 ± 0.05	0.13 ± 0.04	0.31 ± 0.02
Scaling Factor -0.75	Verbatim	4 ± 3	0.08 ± 0.05	0.11 ± 0.06	0.17 ± 0.06	0.31 ± 0.02
	Template	21 ± 18	0.05 ± 0.03	0.08 ± 0.05	0.14 ± 0.05	0.31 ± 0.02
Scaling Factor -1	Verbatim	4 ± 3	0.08 ± 0.05	0.11 ± 0.07	0.16 ± 0.06	0.31 ± 0.02
	Template	21 ± 18	0.04 ± 0.03	0.07 ± 0.05	0.14 ± 0.05	0.30 ± 0.02



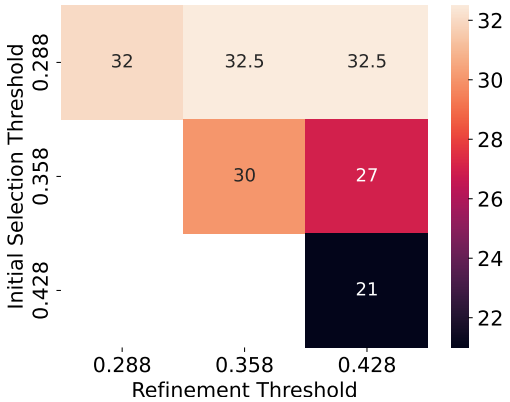
(a) Number of initial neurons found for VM.



(b) Number of initial neurons found for TM.

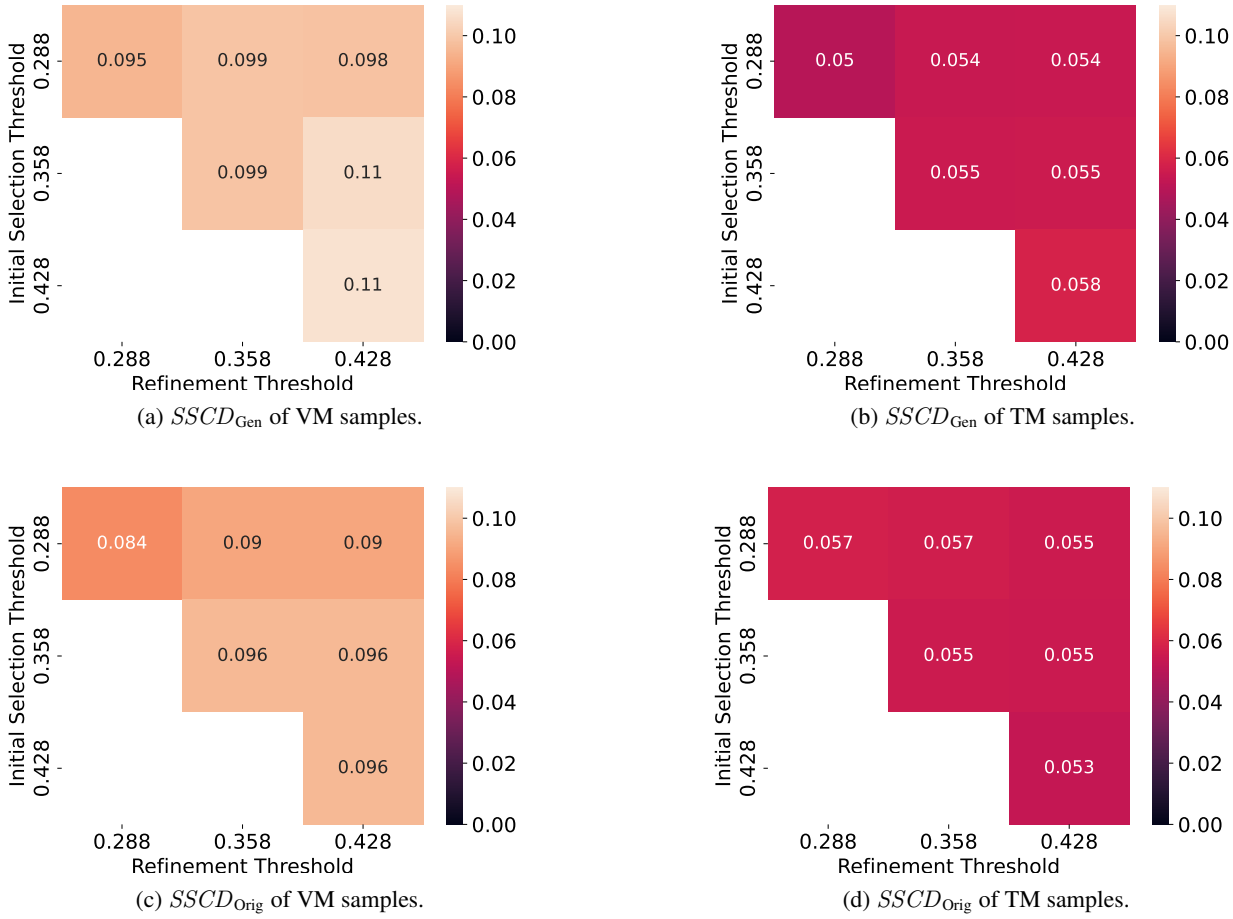


(c) Number of refined neurons found for VM.



(d) Number of refined neurons found for TM.

Figure 14: **Number of neurons found with different initial and refinement thresholds τ_{mem}.** The left plots show the results for verbatim memorization prompts, while the right plots show the results for template memorization prompts. The refinement step significantly reduces the number of identified neurons across all threshold combinations. Notably, using 0.428 for both the initial selection and refinement thresholds results in the smallest set of identified neurons.



(a) $SSCD_{\text{Gen}}$ of VM samples.

(b) $SSCD_{\text{Gen}}$ of TM samples.

(c) $SSCD_{\text{Orig}}$ of VM samples.

(d) $SSCD_{\text{Orig}}$ of TM samples.

Figure 15: **SSCD memorization scores with different initial and refinement thresholds τ_{mem}.** The left plots show the results for verbatim memorization prompts, while the right plots show the results for template memorization prompts. The value of the thresholds does not seem to have a high impact on the memorization scores. Since higher thresholds identify much less memorization neurons, choosing a threshold of $\tau_{\text{mem}} = 0.428$ is a valid choice.

B.7. Ablation of Individual Key and Value Layers

During our experiments in the main paper, we limit our search with NEMO for memorization neurons to cross-attention value layers in the down- and mid-blocks of the U-Net. To motivate this decision, we perform an analysis of the influence of neurons in the individual key and value layers of different cross-attention blocks. Let us first recall the computation performed in attention layers:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V\,. \tag{3}$$

The computed key and query matrices K and Q are used to calculate the attention scores, i.e., the weighting of the components in the value matrix V. In the cross-attention layers, the query matrix Q is computed by linearly mapping the current feature maps from the previous U-Net layer. Therefore, information from the textual guidance is only indirectly contained, i.e., of earlier layers or the U-Nets input feature map after the first denoising step. Therefore, we can exclude neurons in the query mapping layers since we aim to identify neurons directly responsible for memorization. The neurons in the key mapping layers directly process the text embeddings to compute the attention scores. However, strong interdependencies exist between the activations of different neurons through the nature of the softmax function. The impact each neuron's activation has on the computed attention score also depends on the activations of all other neurons from the same layer. Removing a single neuron, i.e., setting its activation to zero, does not necessarily imply substantial changes in the attention scores and the corresponding weighting of features from the value mapping layer.

The value mapping layers, however, also directly process the text embeddings, but there is no direct interdependence between the activations of different neurons. Consequently, setting the activations of individual neurons in value layers to zero directly blocks the information flow from the text embeddings. We hypothesize that the neurons in the value layers are mainly responsible for memorizing the text embeddings of specific prompts.

We evaluate this assumption by taking a set of 100 memorized prompts, generating ten samples for each prompt, and comparing the impact of removing neurons from different layers. More specifically, we remove all activations of individual key and value mapping layers, i.e., setting the output vectors of these layers to zero while keeping all other parts of the model untouched. We then compare the generated images with removed activations to the original training images. Fig. 16 plots the resulting SSCD similarity scores for deactivating individual value (top row) and key (bottom row) layers. We distinguish between verbatim (left column) and template (right column) prompts. The plots show the maximum and median SSCD scores and the deviations for the median scores. We decided not to plot deviations for the maximum score to improve readability. However, deviations are comparable to the median scores. Baselines computed without any deactivated neurons are plotted as dashed lines.

Stable Diffusion contains six cross-attention layers in the down-blocks, one in the mid-block, and nine in the up-blocks. The vertical lines indicate the separation between the different blocks. For the value layers, the layers with indexes 1 (down-lock) and 7 (mid-block) have the highest impact, whereas layers later in the network hardly change the SSCD scores. Also, the effect of the remaining layers in the down-blocks is small on their own. However, we expect there to be entwined effects between deactivating neurons in different layers, which is why we also searched for memorization neurons in these down-block layers.

For the key layers, particularly layers 4 and 6 in the down-blocks have the strongest impact on the generated images. However, removing these layers often produces images that no longer align with the concepts in the prompt or degrades the image quality, both leading to lower SSCD scores. We quantify this behavior by computing the alignments between the generated images and the corresponding input prompts in Fig. 17. While deactivating individual value layers only slightly decreases the alignment scores, deactivating some key layers substantially reduces the alignment. To further illustrate this fact, we plot some of the generated images for deactivating individual value layers in Fig. 18 and for key layers in Fig. 19.

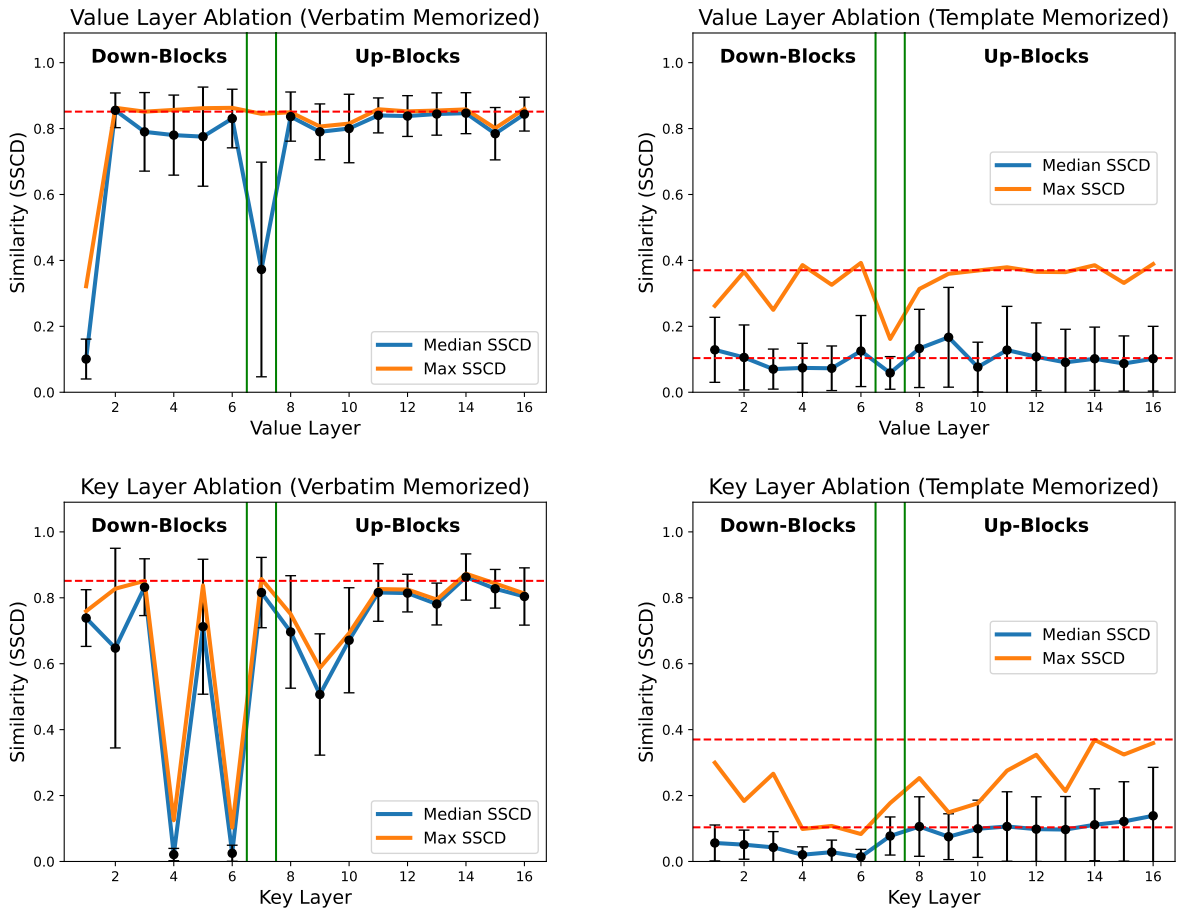


Figure 16: **SSCD similarity scores between memorized generations and the corresponding training samples.** Scores are computed for 100 prompts and ten different seeds per generation. We then take the maximum and median scores of each prompt. During the generation, we deactivated individual value and key layers of the cross-attention blocks in the network. A lower SSCD score indicates a lower similarity between generated and training images. Dashed lines denote the median and the maximum SSCD baselines for images generated without deactivating any neurons. For verbatim memorized prompts, both baselines are close, which is why we only plot the median SSCD baseline.

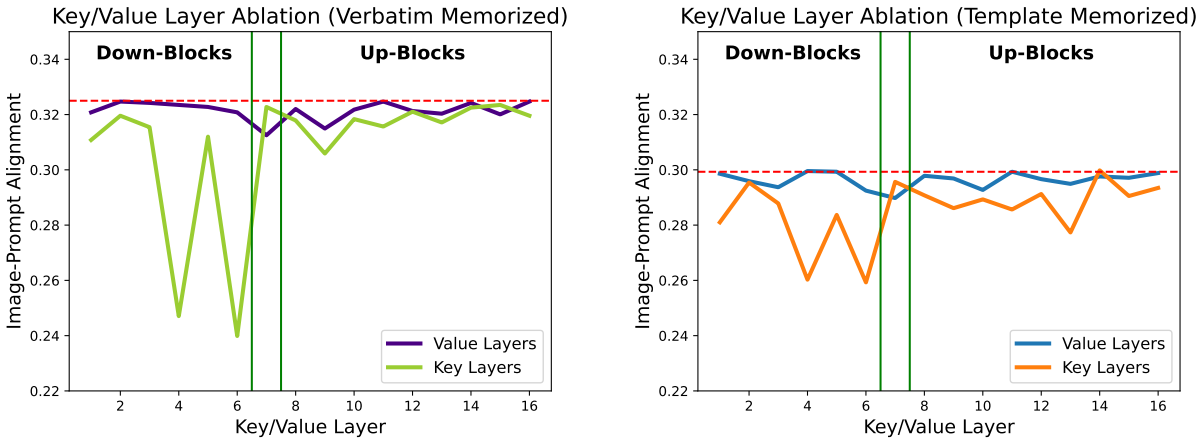


Figure 17: **CLIP alignment scores between memorized generations and the corresponding input prompt.** Scores are computed for 100 prompts and ten different seeds per generation. We then take the median alignment scores of each prompt. During the generation, we deactivated individual value and key layers of the cross-attention blocks in the network. A higher alignment score indicates a better representation of the prompt concepts in the generated images. Dashed lines denote the median alignment scores for images generated without deactivating any neurons. For both types of memorization, deactivating value layers decreases the alignment only slightly, whereas deactivating some key layers substantially reduces the alignment.

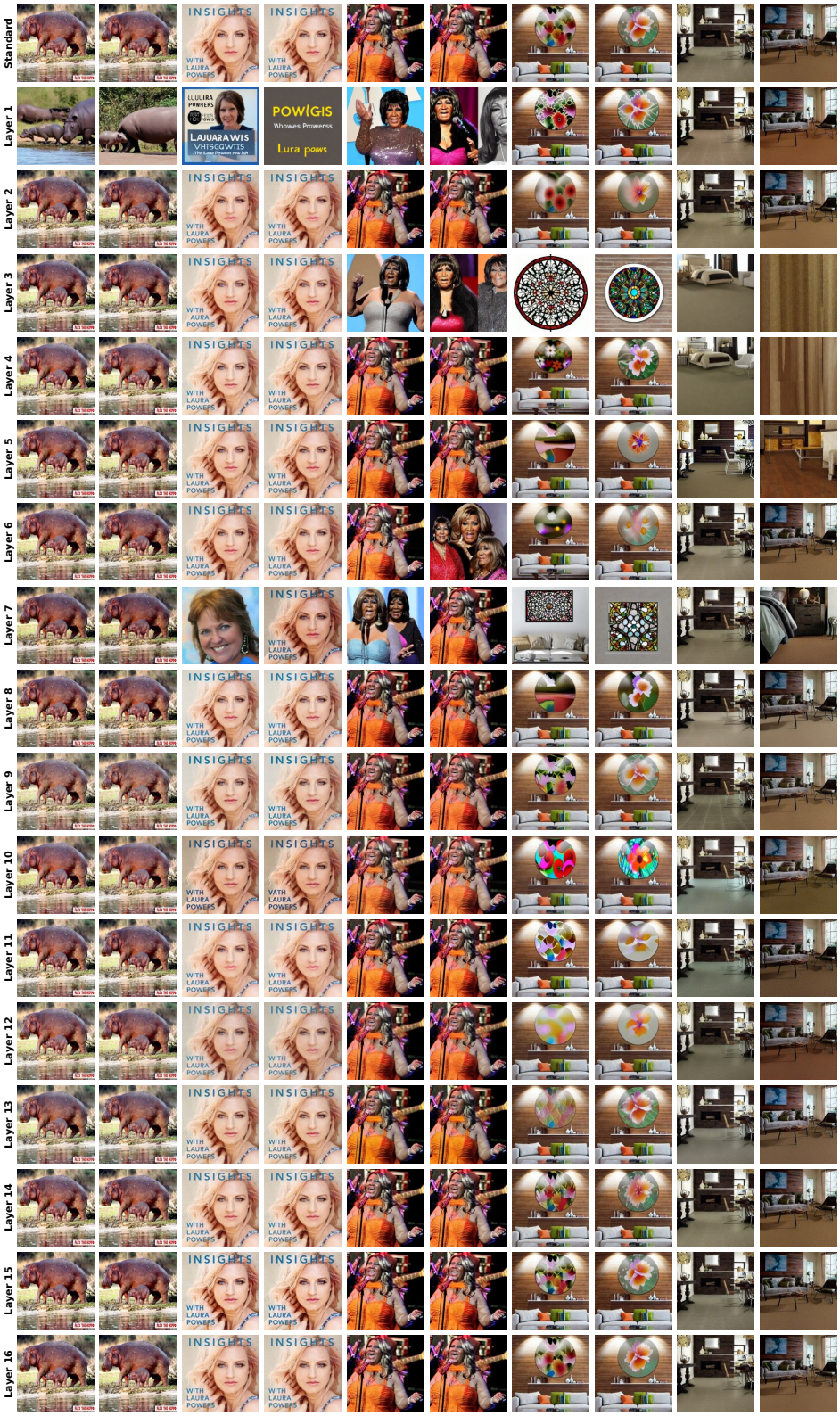


Figure 18: **Images generated with memorized prompts with deactivated individual value layers.** Whereas the *standard* row shows generations with keeping all neurons active, the following rows depict results for deactivating all neurons in a specific value layer.

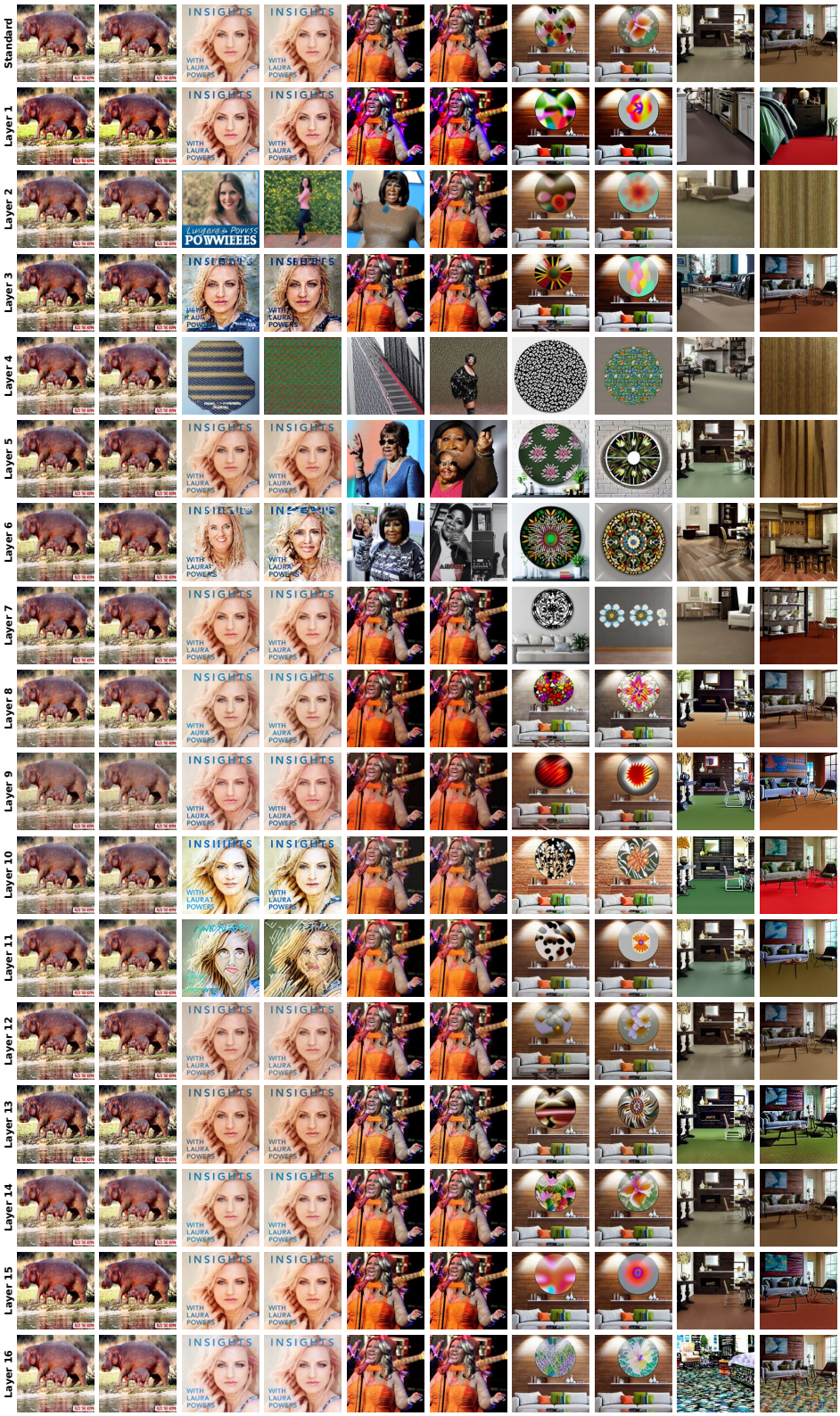


Figure 19: **Images generated with memorized prompts with deactivated individual key layers.** Whereas the *standard* row shows generations with keeping all neurons active, the following rows depict results for deactivating all neurons in a specific key layer.

B.8. Comparison of Mitigation Strategies

We compare the mitigation effects of deactivating the neurons identified as memorizing by NeMo with existing mitigation strategies. Specifically, we used the gradient-based prompt embedding adjustment proposed by Wen et al. (2024) and the addition of random tokens to the prompt, as proposed by Somepalli et al. (2023). For the latter approach, we added 1, 4, or 10 random tokens to the prompts.

The evaluation results in Tab. 3 demonstrate that deactivating the memorization neurons leads to memorization mitigation comparable to the gradient-based prompt embedding adjustment. Adding random tokens also reduces memorization; however, for 1 or 4 tokens, the memorization, as quantified by the SSCD scores, is still higher than with deactivated memorizing neurons. Adding 10 random tokens leads to comparable mitigation but also reduces the prompt alignment score.

Table 3: **Impact of Memorization Mitigation Strategies.**

Setting	Memorization Type	Deactivated Neurons	$\downarrow SSCD_{\text{Orig}}$	$\downarrow SSCD_{\text{Gen}}$	$\downarrow D_{\text{SSCD}}$	$\uparrow A_{\text{CLIP}}$
All Neurons Activate (Default)	Verbatim	0	0.83 ± 0.16	1.0 ± 0.0	0.99 ± 0.01	0.32 ± 0.02
	Template	0	0.04 ± 0.04	1.0 ± 0.0	0.17 ± 0.06	0.31 ± 0.02
Prompt Embedding Adjustment (Wen et al. (2024))	Verbatim	0	0.04 ± 0.02	0.08 ± 0.03	0.08 ± 0.03	0.30 ± 0.02
	Template	0	0.03 ± 0.02	0.08 ± 0.03	0.09 ± 0.03	0.31 ± 0.02
Adding 1 Random Token (Somepalli et al. (2023))	Verbatim	0	0.59 ± 0.31	0.68 ± 0.31	0.67 ± 0.33	0.31 ± 0.02
	Template	0	0.04 ± 0.03	0.16 ± 0.05	0.17 ± 0.05	0.31 ± 0.02
Adding 4 Random Tokens (Somepalli et al. (2023))	Verbatim	0	0.09 ± 0.06	0.12 ± 0.09	0.15 ± 0.06	0.30 ± 0.02
	Template	0	0.04 ± 0.03	0.13 ± 0.04	0.15 ± 0.04	0.30 ± 0.02
Adding 10 Random Tokens (Somepalli et al. (2023))	Verbatim	0	0.03 ± 0.02	0.07 ± 0.05	0.11 ± 0.03	0.28 ± 0.03
	Template	0	0.03 ± 0.03	0.08 ± 0.05	0.12 ± 0.04	0.29 ± 0.03
Deactivating Memorization Neurons (NeMo)	**Verbatim**	$\mathbf{4 \pm 3}$	$\mathbf{0.09 \pm 0.06}$	$\mathbf{0.10 \pm 0.07}$	$\mathbf{0.16 \pm 0.06}$	$\mathbf{0.31 \pm 0.02}$
	Template	$\mathbf{21 \pm 18}$	$\mathbf{0.05 \pm 0.03}$	$\mathbf{0.05 \pm 0.04}$	$\mathbf{0.12 \pm 0.05}$	$\mathbf{0.31 \pm 0.02}$

C. Algorithmic Description of NeMo

C.1. Computing Noise Differences

Alg. 1 defines our algorithm to compute the differences between the initial noise samples and the noise predicted during the first denoising step. The resulting noise differences are used to compute our SSIM-based memorization score during the initial neuron selection and the refinement step. We compute the noise differences always for $n = 10$ different seeds to avoid undesired biases due to the random sampling process. We further remove noise differences from the set, which have low similarity with other noise differences. By this step, we remove noise differences for seeds that do not lead to memorization and might mislead the algorithm.

Algorithm 1 Compute Noise Differences

Input:
Prompt embedding y ▷ Text prompt (embedding)
Neuron set S_{neurons} ▷ Set of neurons to deactivate
Noise predictor ϵ_θ ▷ Diffusion model
Memorization threshold (SSIM) τ_{mem} ▷ Target memorization score

Output: Noise differences Δ

Set Δ as empty list ▷ Initialize list of noise differences
$\tilde{\epsilon}_\theta \leftarrow$ deactivate_neurons$(\epsilon_\theta, S_{\text{Neurons}})$ ▷ Set activations of neurons in S_{neurons} to zero

// Compute noise differences for each random seed
for $i = 1, \ldots, 10$ **do** ▷ Iterate over 10 seeds
  set_seed(i) ▷ Set random seed to i
  sample $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Randomly initialize noise image
  $x_{T-1} \leftarrow \tilde{\epsilon}_\theta(x_T, T, y)$ ▷ Compute noise prediction
  $\delta \leftarrow x_{T-1} - x_T$ ▷ Compute noise difference
  $\delta \leftarrow \frac{\delta - \min(\delta)}{\max(\delta) - \min(\delta)}$ ▷ Normalize differences by min-max scaling
  append δ to Δ ▷ Add current noise difference to list
end for

// Remove noise differences not leading to memorization
for $\delta \in \Delta$ **do** ▷ Iterate over noise differences
  $\bar{\Delta} \leftarrow \Delta \setminus \delta$ ▷ Get set of noise differences without δ
  $d \leftarrow$ compute_memorization$(\delta, \bar{\Delta})$ ▷ Compute pairwise memorization scores (SSIM)
  if $\max(d) < \tau_{\text{mem}}$ **then** ▷ Highest memorization score is below threshold
    $\Delta \leftarrow \Delta \setminus \delta$ ▷ Remove noise difference from set
  end if
end for
return Δ ▷ Return list of noise differences

C.2. Detecting Neurons with Out-of-Distribution Activations

Alg. 2 describes our method to detect neurons with out-of-distribution (OOD) activations. Our method detects OOD neurons based on their activation distance for a memorized prompt to a neuron's mean activation computed on a hold-out dataset of non-memorized prompts. In addition, we also add the k neurons with the highest absolute activations within each layer to the set.

Algorithm 2 Get OOD Neurons

Input:
Prompt embedding y ▷ Text prompt (embedding)
Activation threshold θ_{act} ▷ Threshold for the OOD detection
Top k ▷ Value of top-k detection
Activation mean μ and standard deviation σ ▷ Activation statistics of hold-out dataset

Output: Set of neurons with OOD activations S_{initial}

$S_{\text{activations}} \leftarrow \text{collect_activations}(y)$ ▷ Collect activations on prompt
$S_{\text{initial}} \leftarrow \{\}$ ▷ Initialize empty neuron set

// Check each neuron in each layer for OOD activation
for $l \in \{1, \ldots, L\}$ **do** ▷ Iterate over all layers
  for $i \in \{1, \ldots, N\}$ **do** ▷ Iterate over all N neurons in layer l
    $z_i^l(y) = \frac{a_i^l(y) - \mu_i^l}{\sigma_i^l}$ ▷ Compute z-score for current neuron

    if $z_i^l(y) > \theta_{\text{act}}$ **then** ▷ Activation above OOD threshold
      $S_{\text{initial}} \leftarrow S_{\text{initial}} \cup \{\text{neuron}_i^l\}$ ▷ Add OOD neuron to set
    end if
  end for

  // Add k neurons of layer l with the highest absolute activations to the candidate set
  $S_{\text{topk}} \leftarrow \text{top_k_activations}(S_{\text{activations}}, l, k)$ ▷ Get neurons with highest absolute activations
  $S_{\text{initial}} \leftarrow S_{\text{initial}} \cup S_{\text{topk}}$ ▷ Add top-k neurons to set
end for

return S_{initial} ▷ Return set with OOD neurons

C.3. Selecting Initial Candidates of Memorization Neurons

Alg. 3 defines our algorithm to compute the initial set of memorization neurons. The resulting initial set of selected memorization neurons is then refined in a second step, shown in Alg. 4.

Algorithm 3 Initial Neuron Selection

Input:
Prompt embedding y ▷ Text prompt embedding
Memorization threshold (SSIM) τ_{mem} ▷ Target memorization score
Minimum activation threshold θ_{min} ▷ Threshold for stopping neuron search

Output: Set of neuron candidates S_{initial}, refinement memorization threshold $\tau_{\text{mem_ref}}$

Candidate set of memorization neurons S_{initial} ▷ Initial memorizing neuron set
Memorization threshold (SSIM) $\tau_{\text{mem_ref}}$ ▷ Memorization threshold for refinement

mem $\leftarrow 1.0$ ▷ Initialize memorization score as maximum
$\theta_{\text{act}} \leftarrow 5$ ▷ Initialize threshold of OOD activation detection
$k \leftarrow 0$ ▷ Initialize k for top-k activation detection
$\tau_{\text{mem_ref}} \leftarrow \tau_{\text{mem}}$ ▷ Set refinement memorization threshold to current threshold
$\Delta_{\text{unblocked}} \leftarrow$ get_noise_diff$(y, \emptyset)$ ▷ Noise differences with all neurons active

// Increase set of candidate neurons until target memorization score is reached
while mem $> \tau_{\text{mem}}$ **do** ▷ While memorization score above threshold
  $S_{\text{initial}} \leftarrow$ get_ood_neurons$(y, \theta_{\text{act}}, k)$ ▷ Detect neurons with OOD activations
  $\Delta_{\text{blocked}} \leftarrow$ get_noise_diff(y, S_{initial}) ▷ Compute noise differences
  mem $\leftarrow$ compute_memorization$(\Delta_{\text{unblocked}}, \Delta_{\text{blocked}})$ ▷ Compute memorization score (SSIM)

  if $\theta_{\text{act}} < \theta_{\text{min}}$ **then** ▷ Minimum activation threshold not reached
    $\tau_{\text{mem_ref}} \leftarrow$ mem ▷ Set refinement threshold to current memorization score
    break ▷ Stop if activation threshold is too low
  end if

  // Adjust OOD detection parameters to increase set of candidate neurons
  $\theta_{\text{act}} \leftarrow \theta_{\text{act}} - 0.25$ ▷ Decrease threshold for OOD detection
  $k \leftarrow k + 1$ ▷ Increase k for top-k activation detection
end while

return $S_{\text{initial}}, \tau_{\text{mem_ref}}$ ▷ Return neuron candidates and refinement memorization threshold

C.4. Neuron Selection Refinement

Alg. 4 defines our algorithm to refine the set of candidate neurons identified from NEMO's initial selection step.

Algorithm 4 Neuron Selection Refinement

Input:
Initial memorization neuron candidate set S_{initial} ▷ Given neuron candidate set
Memorization threshold (SSIM) $\tau_{\text{mem_ref}}$ ▷ Refinement memorization score threhsold

Output: memorization neurons S_{refined} ▷ Refined set of memorization neurons

$S_{\text{refined}} \leftarrow S_{\text{initial}}$
$\Delta_{\text{unblocked}} \leftarrow \text{get_noise_diff}(y, \emptyset)$ ▷ Noise differences with all neurons active

// Check all candidate neurons of individual layers at once for memorization
for $l \in \{1, \ldots, L\}$ **do** ▷ Iterate over all layers to remove low impact layers
  $S_{\text{layer}} \leftarrow \text{get_neurons_in_layer}(S_{\text{refined}}, l)$ ▷ Get the neurons in the current layer l
  $S_{\text{neurons}} \leftarrow S_{\text{refined}} \setminus S_{\text{layer}}$ ▷ Compute set of neurons from remaining layers
  $\Delta_{\text{blocked}} \leftarrow \text{get_noise_diff}(y, S_{\text{neurons}})$ ▷ Compute noise differences
  mem $\leftarrow \text{compute_memorization}(\Delta_{\text{unblocked}}, \Delta_{\text{blocked}})$ ▷ Compute memorization score (SSIM)

  if mem $< \tau_{\text{mem_ref}}$ **then** ▷ Minimum memorization threshold not reached
    $S_{\text{refined}} \leftarrow S_{\text{refined}} \setminus S_{\text{layer}}$ ▷ Remove neurons of layer l from neuron set
  end if
end for

// Check all remaining candidate neurons individually
for $l \in \{1, \ldots, L\}$ **do** ▷ Iterate over each remaining layer
  $S_{\text{layer}} \leftarrow \text{get_neurons_in_layer}(S_{\text{refined}}, l)$ ▷ Get the neurons in the current layer l

  for $n \in S_{\text{layer}}$ **do**
    $S_{\text{neurons}} \leftarrow S_{\text{refined}} \setminus \{n\}$ ▷ Compute set of neurons without neuron n
    $\Delta_{\text{blocked}} \leftarrow \text{get_noise_diff}(y, S_{\text{neurons}})$ ▷ Compute noise differences
    mem $\leftarrow \text{compute_memorization}(\Delta_{\text{unblocked}}, \Delta_{\text{blocked}})$ ▷ Compute mem. score (SSIM)

    if mem $< \tau_{\text{mem_ref}}$ **then** ▷ Minimum memorization threshold not reached
      $S_{\text{refined}} \leftarrow S_{\text{refined}} \setminus \{n\}$ ▷ Remove current neuron from set
    end if
  end for
end for

return S_{refined} ▷ Return refined set of memorization neurons