Manipulating Feature Visualizations with Gradient Slingshots

Dilyara Bareeva [1] Marina M.-C. Höhne [2 3 4] Alexander Warnecke [3 5] Lukas Pirch [5] Klaus-Robert Müller [6 3 7 8]
Konrad Rieck [3 5] Kirill Bykov [2 6 3]

Abstract

Deep Neural Networks (DNNs) are capable of learning complex and versatile representations, however, the semantic nature of the learned concepts remains unknown. A common method used to explain the concepts learned by DNNs is Feature Visualization (FV), which generates a synthetic input signal that maximally activates a particular neuron in the network. In this paper, we investigate the vulnerability of this approach to adversarial model manipulations and introduce a novel method for manipulating FV without significantly impacting the model's decision-making process. The key distinction of our proposed approach is that it does not alter the model architecture. We evaluate the effectiveness of our method on several neural network models and demonstrate its capabilities to hide the functionality of arbitrarily chosen neurons by masking the original explanations of neurons with chosen target explanations during model auditing.

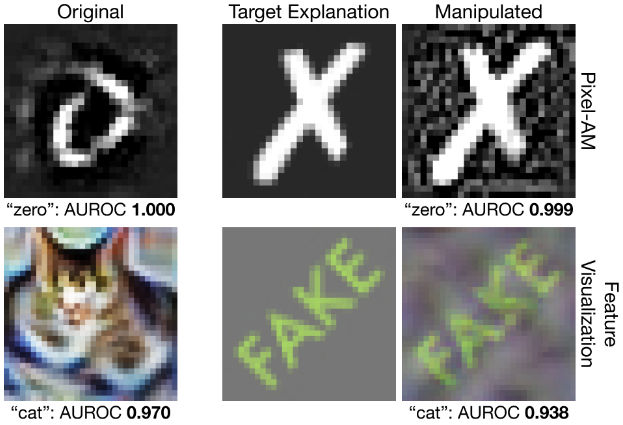


Figure 1: The *Gradient-Slinghot* method manipulates the result of the Activation-Maximization (AM) for a given neuron with minimal change in its behavior. The figure shows manipulation of pixel-AM (top, "zero" logit in MNIST model) and Feature Visualization (bottom, "cat" logit in CIFAR10 model), where the AMs are manipulated with little change to the performance (measured by AUROC).

1. Introduction

Deep Neural Networks (DNNs) have gained widespread adoption in various domains due to their remarkable learning capabilities (LeCun et al., 2015). Nevertheless, the factors driving the decisions of DNNs often remain unknown and poorly understood, turning them into a "black box". This lack of transparency has led to the development of various approaches aiming to explain the decisions of neural networks in recent years (e.g. Simonyan et al., 2014; Sundararajan et al., 2017; Bach et al., 2015). The field of *mechanistic interpretability* seeks to explain DNNs by identifying understandable features and reverse-engineering the connections between them (e.g. Olah et al., 2017; 2020; Bricken et al., 2023). Besides insights into the models, these approaches also unveiled the tendencies of DNNs to pick up spurious correlations or biases from the training data (Lapuschkin et al., 2019; Goh et al., 2021; Bianchi et al., 2023; Bykov et al., 2023b). Given the popularity of Deep Learning in safety-critical fields (e.g. Piccialli et al., 2021; de la Escalera et al., 2003), these techniques are considered a cornerstone to achieving trustworthy DNNs.

A well-known method for explaining the concepts learned by a model is *Activation-Maximization* (AM) (Erhan et al., 2009; Olah et al., 2017; Fel et al., 2023). This approach aims to explain the features learned by a network during training by illustrating maximally activating *input stimuli* for specific units. To date, little is known about the trustworthiness of AM-based explanation methods. This work

[1] Department of Artificial Intelligence, Fraunhofer Heinrich-Hertz-Institute, Berlin, Germany [2] Understandable Machine Intelligence Lab, ATB, Potsdam, Germany [3] BIFOLD – Berlin Institute for the Foundations of Learning and Data, Germany [4] Machine Learning Group, UiT the Arctic University of Norway, Tromsø, Norway [5] Machine Learning and Security Group, Technische Universität Berlin, Germany [6] Machine Learning Group, Technische Universität Berlin, Germany [7] Department of Artificial Intelligence, Korea University, Seoul, Republic of Korea [8] Max Planck Institute for Informatics, Saarbrücken, Germany. Correspondence to: Dilyara Bareeva <dilyara.bareeva@hhi.fraunhofer.de>.

Accepted to *ICML 2024 Workshop on Mechanistic Interpretability*, Vienna, Austria. July, 2024. Copyright 2024 by the author(s).

analyzes the susceptibility of these explanation techniques to adversarial manipulations (Dombrowski et al., 2019; Heo et al., 2019; Anders et al., 2020). Although previous research has shown that synthetic AM explanations can be manipulated, e.g., by embedding the target network into a fooling circuit (Geirhos et al., 2024), this paper presents the first attempt towards generating arbitrary synthetic AM outputs while maintaining the original model architecture and performance.

Concretely, we present the *Gradient Slingshots* method, which manipulates synthetically generated Activation-Maximization signals while preserving the original functionality of the representations (see Figure 1). We show that these manipulation techniques can successfully conceal problematic and malicious representations within Deep Neural Networks (DNNs), making them potentially undetectable during model audits.

2. Related Work

Activation-Maximization (AM) (Erhan et al., 2009; Simonyan et al., 2014) is a popular approach for explaining what abstractions latent representations in DNNs have learned to detect, by illustrating what features and patterns maximally activate them. This could be done either by searching for signals in the corpus of natural data (Szegedy et al., 2013; Borowski et al., 2020; Bykov et al., 2023a), or signals could be generated synthetically (Erhan et al., 2009; Olah et al., 2017; Fel et al., 2023). Such methods allow for identifying neurons that contribute to undesirable model behavior (Bykov et al., 2023a; Goh et al., 2021), detection of backdoor attacks (Casper et al., 2023), and explaining probabilistic models (Grinwald et al., 2023). Synthesizing such inputs in an unconstrained input domain often produces non-interpretable outputs (Olah et al., 2017) which can be circumvented by regularization strategies or by using a differentiable parametrization of the optimized input (Nguyen et al., 2016; Mordvintsev et al., 2018). Olah et al. (2017) and Fel et al. (2023) employ *Feature Visualization* (FV) approaches based on the parameterization of input signals via Fourier transform, which serves the purpose of decorrelating and whitening of the image optimization domain.

The sensitivity of synthetic AM to adversarial manipulations is not well understood. Geirhos et al. (2024) introduced two attack schemes on synthetic AM: one involves engineering "fooling circuits", while the other replaces "silent units" with manipulated computational blocks. Although the architectural "add-ons", such as a convolutional filter encoding the target image, offer precise control over the AM output, they are easily detected by inspecting a model's code. In contrast, we propose using a manipulation loss term, eliminating the need for conspicuous architectural modifications.

3. Gradient Slingshots

In this section, we present the *Gradient Slingshot* (GS) attack that can manipulate the outcome of AM with minimal impact on the model behavior. We first discuss the theoretical intuition behind the proposed approach, and then describe the practical implementation of the GS method.

3.1. Activation Maximization

Let $\mathbb{F}$ be a set of almost everywhere differentiable functions from an input domain $\mathbb{D} \subset \mathbb{R}^d$ to $\mathbb{R}$. Given a function $f \in \mathbb{F}$, representing the activation of a particular neuron, the AM method aims to find an input $x \in \mathbb{D}$ such that it maximizes the output of the neural representation f. Such optimization problem is non-convex (Erhan et al., 2009), and gradient-based methods are often employed to find the local solution. Conventionally, the optimization process starts from a randomly sampled initialization point $x^{(0)} \sim \mathcal{I}$, where $\mathcal{I}$ is a probability distribution with support defined on $\mathbb{D}$. The update rule for gradient ascent is then given by

$$x^{(i+1)} = x^{(i)} + \epsilon \nabla_x f(x^{(i)}), \tag{1}$$

where $\epsilon \in \mathbb{R}_+$ is the learning rate.

3.2. Theoretical Basis

Let $f \in \mathbb{F}$ be a single neuron within the DNN. We assume that the *adversary* performing the manipulation procedure is aware of the initialization distribution $\mathcal{I}$ with $\tilde{x} = \mathbb{E}[\mathcal{I}]$, as well as the AM optimization algorithm. The goal of the *adversary* is to fine-tune the original neuron f to obtain a function f^* so that the result of the AM procedure converges to a pre-defined target signal $s \in \mathbb{D}$, while minimizing the impact on the network behavior.

Let $g \in \mathbb{F}$ be a function with a global optimum at s, such that $\forall x \in \mathbb{D}$ we have

$$\nabla g(x) = \gamma(s - x), \tag{2}$$

where $\gamma \in \mathbb{R}$ is a constant hyperparameter. This condition ensures all partial derivatives are directed towards our target point s, ensuring convergence of the optimization procedure to this point. Integrating the linear differential equation yields a quadratic function of the form

$$g(x) = \frac{1}{\epsilon}\left(s^\top x - \frac{1}{2}x^\top x\right) + C. \tag{3}$$

Gradient Slingshots (GS) aim to fine-tune the original function only in a small subset of $\mathbb{D}$, retaining the original behavior elsewhere. In more detail, the manipulated version f^* of the original function f then takes the form

$$f^*(x) = \left\{ \begin{array}{ll} f(x) & x \in \mathbb{D} \setminus M \\ g(x) & x \in M \end{array} \right. , \tag{4}$$

where $g \in \mathbb{F}$ is is as described in Equation (3) and $M \subset \mathbb{D}$ is the manipulation subset. Intuitively, M corresponds to the subset of the input domain that can be reached throughout the AM optimization procedure. This synthetic subset is distinct from the domain of natural images (Nguyen et al., 2019; Geirhos et al., 2024).

We define the ball around the expected initialization $B = \{x : ||\tilde{x} - x|| \leq R_B, x \in \mathbb{D}\}$, where the radius R_B is determined by the adversary utilizing the knowledge of the distribution $\mathcal{I}$. We call the initialization region B *"slingshot zone"*, as it corresponds to high-amplitude gradients directed at the target. Further, we define $L = \{x : ||s - x|| \leq R_L, x \in \mathbb{D}\}$, where R_L is a parameter. We refer to L as the *"landing zone"*, since changing the behavior in the neighborhood of the target point s ensures the stable convergence of a gradient ascent algorithm, implying $\nabla g(s) = \mathbf{0}$. For standard gradient ascent, we can guarantee the convergence of AM to the target when $M = B \cup L$ (see Appendix A.1 for proof). Advanced optimization techniques like Adam (Kingma & Ba, 2017), or AM algorithms involving transformation robustness (Olah et al., 2017), expand the manipulation subspace M to include "slingshot" zones around the initialization mean and points likely to be reached in the input space during the AM procedure. In the following, we focus on plain gradient ascent. We demonstrate our approach on a toy model in Appendix B.

3.3. Practical Implementation

Let a *feature-extractor* $F = \{f_1, ..., f_k\}, f_i \in \mathbb{F}, \forall i \in [1, ..k]$ be a collection of individual scalar functions, such that $F(x) = (f_1(x), ..., f_k(x))$. Let $m \in [1, ..., k]$ be an index of the neuron that is set to be manipulated. To simplify the notation, let $J = [1, ..., k] \setminus \{m\}$ be a collection of indexes of neurons, except the neuron m. We denote the optimized version f_i^θ of any $f_i \in \mathbb{F}$ with a superscript θ, signifying the set of optimized parameters of the model.

We introduce two loss terms: one responsible for manipulating the AM objective, and another for maintaining the behavior of the original functions.

Let $\mathcal{M}$ be a distribution with support defined on a manipulation set $M \subset \mathbb{D}$. This is the sampling procedure that defines sampling from the "slingshot" and "landing" zones. Let $\boldsymbol{M}$ be a collection of N i.i.d. samples from this distribution:

$$\begin{aligned} \boldsymbol{M} = &\{x_1, ..., x_{N'} \in \mathbb{D} : ||\tilde{x} - x||_\infty \leq \sigma_s\} \cup \\ &\{x_{N'}, ..., x_N \in \mathbb{D} : ||s - x||_\infty \leq \sigma_l\}, \end{aligned} \tag{5}$$

where $1 \leq N^{'} \leq N$, $\tilde{x} = \mathbb{E}(\mathcal{I})$, s is the target image and σ_s and σ_l are the manipulation radii in the "slingshot" and "landing" zones respectively. σ_s is chosen to adequately span points highly likely to be drawn from $\mathcal{I}$.

The *manipulation* loss term, which measures the difference between existing and required gradients in the manipulated neuron on $\boldsymbol{M}$:

$$\mathcal{L}_{\mathcal{M}}(\theta) = \frac{1}{N} \sum_{x \in \boldsymbol{M}} \left\| \nabla f_m^\theta(x) - \gamma(s - x) \right\|_2^2. \tag{6}$$

The *preservation* term $\mathcal{L}_{\mathcal{P}}$ measures how the activations in the manipulated feature-extractor F^* differ from the activations in the original feature-extractor F. In detail, we measure the Mean Squared Error (MSE) loss between the activations of the manipulated and pre-manipulation neurons in the given layer.

As the activations of the neural representation f_m are more susceptible to being changed by the manipulation procedure, we may need to assign more weight to the changes in this neuron. Given a training set $\mathbf{X}$, we define the following term for a neuron index subset $\mathbf{A} \subset [1, ..., k]$:

$$\mathcal{L}_P^{\mathbf{A}}(\theta) = \frac{1}{|\mathbf{A}| \cdot |\mathbf{X}|} \sum_{x \in \mathbf{X}} \sum_{a \in \mathbf{A}} ||f_a^\theta(x) - f_a(x)||_2^2. \tag{7}$$

We then formulate the *preservation* loss term as

$$\mathcal{L}_P(\theta) = w \cdot \mathcal{L}_P^J(\theta) + (1 - w) \cdot \mathcal{L}_P^{\{m\}}(\theta), \tag{8}$$

where $w \in [0, 1]$ is a constant parameter, $\mathcal{L}_P^J$ controls the performance in all neural representations in layer F except for the manipulated f_m, and $\mathcal{L}_P^{\{m\}}$ controls the performance in the neural representation f_m.

Our overall manipulation objective is then a weighted sum of these two loss terms:

$$\mathcal{L}(\theta) = \alpha \mathcal{L}_{\mathcal{P}}(\theta) + (1 - \alpha) \mathcal{L}_{\mathcal{M}}(\theta), \tag{9}$$

where $\alpha \in [0, 1]$ is a constant parameter.

4. Evaluation

We perform our evaluation tests on CNNs trained on MNIST (Deng, 2012) and CNNs trained on CIFAR-10 (Krizhevsky et al., 2009). The details regarding data preprocessing, model architectures, training, adversarial finetuning, and AM procedures can be found in Appendix C.

The CNNs trained on the MNIST dataset were manipulated to alter their AM in the pixel domain (Erhan et al., 2009). We refer to this approach as *pixel-AM*. We select an image of a cross symbol as the target image (see Figure 1) and manipulate the output neuron responsible for classifying digit 0. The manipulation result, represented in Figure 1, closely resembles the target image. The AUROC values in Figure 1 indicate that the neuron after the manipulation remains a 0 *detector*.

The CNNs trained on the CIFAR-10 dataset underwent manipulation to alter their AM in the scaled Fourier frequency

Table 1: Manipulated models: Test accuracy, along with the mean and standard deviation of the similarity (MSE, scaled by 100) between the manipulated AM and the target image.

	MNIST (pixel-AM)		CIFAR-10(FV)	
α	Acc.	LPIPS ↓	Acc.	LPIPS ↓
Original	99.867	15.0 ± 1.2	86.330	26.7 ± 1.9
$1.0 \cdot 10^{-4}$	91.058	14.4 ± 4.0	10.330	21.7 ± 3.5
$3.3 \cdot 10^{-4}$	93.058	20.3 ± 3.9	21.470	21.9 ± 3.1
$6.7 \cdot 10^{-4}$	94.967	$\mathbf{11.3 \pm 1.3}$	44.610	19.4 ± 4.4
$1.0 \cdot 10^{-3}$	94.750	28.4 ± 2.0	50.430	19.4 ± 4.4
$3.3 \cdot 10^{-3}$	96.967	16.8 ± 3.4	72.500	12.2 ± 1.8
$6.7 \cdot 10^{-3}$	97.300	31.8 ± 7.9	78.720	11.7 ± 1.8
$1.0 \cdot 10^{-2}$	97.358	17.4 ± 3.3	82.390	$\mathbf{7.2 \pm 1.0}$
$3.3 \cdot 10^{-2}$	97.958	15.6 ± 4.6	84.880	12.1 ± 1.2
$6.7 \cdot 10^{-2}$	98.450	16.2 ± 4.1	85.220	12.4 ± 2.6
$1.0 \cdot 10^{-1}$	**98.883**	15.3 ± 4.1	**85.570**	14.3 ± 1.6

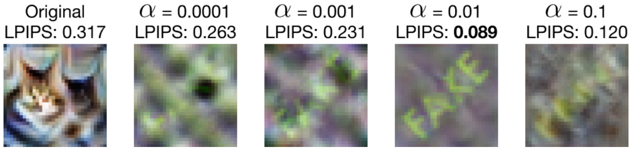


Figure 2: Sample FVs and similarity to target at different values of α for a CNN trained on CIFAR-10. Both very low and high values of α result in low similarity to the target.

domain (Olah et al., 2017), an approach which we refer to simply as *Feature Visualization* (FV). We use an image with the text "FAKE" as the target image (see Figure 1) and manipulate a neuron responsible for the class "cat" in the output layer. From Figure 1, we can observe that the AM output of the manipulated models is perceptually similar to the target image. The AUROC values in this figure indicate that, even after manipulation, the neuron remains a *cat detector*. We then measure the similarity between the AM output and the target images using the Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) metric.

4.1. Accuracy – Manipulation Trade-Off

The manipulation procedure involves a trade-off between preserving the model performance and achieving the manipulation objective (Equation (9)). In this experiment, we manipulate multiple models by varying the parameter α, which controls the weights of manipulation and preservation loss terms, and fixing the other fine-tuning parameters.

The experimental results for MNIST models with pixel-AM and CIFAR-10 models with FV are represented in Table 1. As anticipated, we observe that very high values of α correspond to a decrease in the similarity of the AM output to the target image. Conversely, very low values of α also result in diminished similarity levels. We hypothesize that low values of α tend to significantly perturb the overall activation landscape, drastically increasing the magnitude of the gradients in the areas surrounding the "landing zone". We provide additional evaluation results, including further similarity metrics, in Appendix D.1. Figure 2 illustrates the variation in the LPIPS metric for FV as the parameter α changes. Full experimental results with two additional similarity metrics can be found in Appendix D.1.

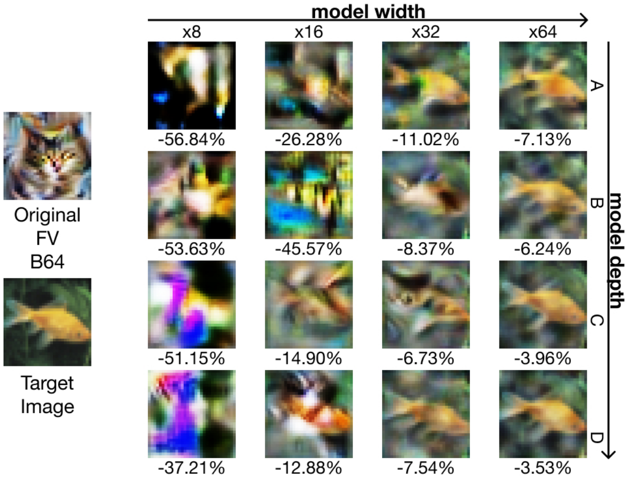


Figure 3: "Catfish" neuron: 16 classification models of varying depth ("A" - "D") and width (×8 - ×64) trained on CIFAR-10 were manipulated to change the FV of the cat output neuron to a fish image. The figure depicts a sample FV for model B64, the target image, and sample FVs of the manipulated models, along with the change in the test accuracy. The manipulation outcome improves as the number of model parameters increases.

4.2. Effect of Model Size

In the following, we investigate the influence of the number of model parameters on manipulation success in terms of similarity to the target image and changes in model performance. Research has shown that even shallow networks with significant width exhibit extensive memorization capabilities (Hornik et al., 1989; Zhang et al., 2021), crucial in our manipulation context requiring target image memorization. Conversely, deeper models can approximate more complex functions (Eldan & Shamir, 2016).

To assess the impact of the model size on the attack, we create image classification models with varying depth and width. Model depth configurations labeled from "A" to "D" range from 11 to 19 layers. Width configurations are expressed as a factor, where the baseline number of units in each layer is multiplied by this factor (see Appendix C for details). The original models are trained on the CIFAR-10 dataset. We perform adversarial fine-tuning to replace the FV output with the image of a goldfish obtained from the ImageNet (Russakovsky et al., 2015) dataset.

Figure 3 visually illustrates sample FV outputs for all 16 model configurations and demonstrates the change in test accuracy between the manipulated and original models. The corresponding quantitative evaluation is presented in Appendix D.2. A discernible correlation is observed between the success of manipulation and the number of model parameters, while the widest models exhibit the best manipulation performance. However, for specifications “×16” and “×64”, the deepest models do not yield the closest similarity to the target image. This could be attributed to the *shattered gradients* effect (Balduzzi et al., 2017), which poses challenges in training deeper models.

4.3. Effect on Activation Maximization in the Natural Image Domain

Given that our manipulation objective (Equation (9)) involves an activation-preserving loss term for the training set, we expect that AM in the natural domain (Borowski et al., 2020) yields semantically consistent results before and after manipulation.

For this evaluation we select two manipulated models from Section 4.1: MNIST model $\alpha = 0.067$ for pixel-AM and CIFAR-10 model $\alpha = 0.01$ for FV. From Figure 4 we observe that the top 4 most activating images in the test set before and after manipulation remain semantically consistent in both settings. In theory, inspection of discrepancies between generative and natural-domain AM results can be used to detect a Gradient Slingshot attack. However, natural-domain AM can also be manipulated (Nanfack et al., 2024), making this defense strategy unreliable. Further quantitative experiments can be found in Appendix D.3.

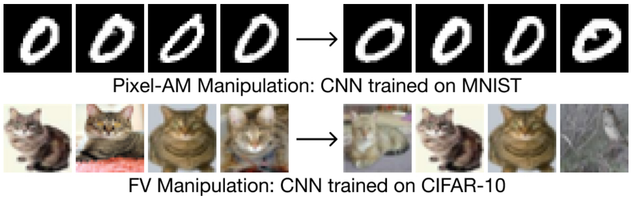

→
Pixel-AM Manipulation: CNN trained on MNIST
→
FV Manipulation: CNN trained on CIFAR-10

Figure 4: Top-4 overall most activating natural signals (n-AMS) in the test set before and after manipulation. The manipulation had no significant impact on the semantic meaning of the top n-AMS signals.

5. Limitations and Future Work

A notable limitation of this study is its focus on the most basic algorithm for Activation Maximization, namely, gradient ascent without regularization. An avenue for future research involves adapting the Gradient Slingshot methodology to accommodate various AM algorithms. However, in Appendix E, we offer preliminary evidence of the robustness of our method to optimization method adjustments, such as gradient clipping or the use of Adam (Kingma & Ba, 2017).

In the evaluation section, we utilized small-dimensional datasets and smaller models. Extending our manipulation approach to higher-dimensional datasets and larger models may pose challenges due to the computationally demanding task of determining optimal hyperparameters, as illustrated in Section 4.1, as well as the *curse of dimensionality*. Addressing these limitations may necessitate further improvements to our method. We present preliminary results for ImageNet (Simonyan & Zisserman, 2015) and a Wide ResNet50 (Zagoruyko & Komodakis, 2016) in Appendix F.

6. Conclusion

Activation-Maximization (AM) methods are extensively employed for uncovering features learned by Deep Neural Networks (DNNs). In this study, we theoretically and empirically illustrate that these methods can be manipulated to display arbitrary images while maintaining the original model architecture and avoiding a substantial decrease in performance. By shedding light on the possibility of such manipulations, we hope to heighten the caution of AI system users and auditors regarding AM-based methods findings.

7. Acknowledgements

The authors gratefully acknowledge funding from the German Federal Ministry of Education and Research (BMBF) through the projects Explaining 4.0 (01IS200551) and AIGENCY (16KIS2014). Furthermore, the authors gratefully acknowledge funding by the German Research Foundation (DFG) under Germany’s Excellence Strategy EXC 2092 CASA (390781972). The authors thank Sebastian Lapuschkin for the helpful comments.

References

Anders, C., Pasliev, P., Dombrowski, A.-K., Müller, K.-R., and Kessel, P. Fairwashing explanations with off-manifold detergent. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 314–323. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/anders20a.html.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015. doi: 10.1371/journal.pone.0130140. URL https://doi.org/10.1371/journal.pone.0130140.

Balduzzi, D., Frean, M., Leary, L., Lewis, J. P., Ma, K. W.-D., and McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 342–350. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/balduzzi17b.html.

Bianchi, F., Kalluri, P., Durmus, E., Ladhak, F., Cheng, M., Nozza, D., Hashimoto, T., Jurafsky, D., Zou, J., and Caliskan, A. Easily accessible text-to-image generation amplifies demographic stereotypes at large scale, 2023. URL https://doi.org/10.1145/3593013.3594095.

Borowski, J., Zimmermann, R. S., Schepers, J., Geirhos, R., Wallis, T. S. A., Bethge, M., and Brendel, W. Natural images are more informative for interpreting CNN activations than state-of-the-art synthetic feature visualizations. In *NeurIPS 2020 Workshop SVRHM*, 2020. URL https://openreview.net/forum?id=-vhO2VPjbVa.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bykov, K., Deb, M., Grinwald, D., Muller, K. R., and Höhne, M. M. DORA: Exploring outlier representations in deep neural networks. *Transactions on Machine Learning Research*, 2023a. ISSN 2835-8856. URL https://openreview.net/forum?id=nfYwRIezvg.

Bykov, K., Kopf, L., and Höhne, M. M.-C. Finding spurious correlations with function-semantic contrast analysis. In Longo, L. (ed.), *Explainable Artificial Intelligence*, pp. 549–572, Cham, 2023b. Springer Nature Switzerland. ISBN 978-3-031-44067-0.

Casper, S., Bu, T., Li, Y., Li, J., Zhang, K., Hariharan, K., and Hadfield-Menell, D. Red teaming deep neural networks with feature synthesis tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 80470–80516, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/febe5c5c6973f713cc43bf0f7c90edbe-Paper-Conference.pdf.

de la Escalera, A., Armingol, J., and Mata, M. Traffic sign recognition and analysis for intelligent vehicles. *Image and Vision Computing*, 21(3):247–258, 2003. ISSN 0262-8856. doi: https://doi.org/10.1016/S0262-8856(02)00156-7. URL https://www.sciencedirect.com/science/article/pii/S0262885602001567.

Deb, M. Feature visualization library for pytorch. https://github.com/Mayukhdeb/torch-dreams, 2021.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477.

Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., and Kessel, P. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bb836c01cdc9120a9c984c525e4b1a4a-Paper.pdf.

Eldan, R. and Shamir, O. The power of depth for feedforward neural networks. In Feldman, V., Rakhlin, A., and Shamir, O. (eds.), *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 907–940, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL https://proceedings.mlr.press/v49/eldan16.html.

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

Fel, T., Boissin, T., Boutin, V., PICARD, A., Novello, P., Colin, J., Linsley, D., ROUSSEAU, T., Cadene, R., Goetschalckx, L., Gardes, L., and Serre, T. Unlocking feature visualization for deep network with magnitude constrained optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 37813–37826, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/76d2f8e328e1081c22a77ca0fa330ca5-Paper-Conference.pdf.

Geirhos, R., Zimmermann, R. S., Bilodeau, B., Brendel, W., and Kim, B. Don't trust your eyes: on the (un)reliability of feature visualizations. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=s0Jvdolv2I.

Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., Radford, A., and Olah, C. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. URL https://distill.pub/2021/multimodal-neurons.

Grinwald, D., Bykov, K., Nakajima, S., and Höhne, M. M. Visualizing the diversity of representations learned by bayesian neural networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=ZSxvyWrX6k.

Heo, J., Joo, S., and Moon, T. Fooling neural network interpretations via adversarial model manipulation. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/7fea637fd6d02b8f0adf6f7dc36aed93-Paper.pdf.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Toronto, ON, Canada*, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. Unmasking clever hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1096, 2019. doi: 10.1038/s41467-019-08987-4. URL https://doi.org/10.1038/s41467-019-08987-4.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Marcel, S. and Rodriguez, Y. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1485–1488, 2010.

Mordvintsev, A., Pezzotti, N., Schubert, L., and Olah, C. Differentiable image parameterizations. *Distill*, 2018. doi: 10.23915/distill.00012. URL https://distill.pub/2018/differentiable-parameterizations.

Nanfack, G., Fulleringer, A., Marty, J., Eickenberg, M., and Belilovsky, E. Adversarial attacks on the interpretation of neuron activation maximization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(5):4315–4324, Mar. 2024. doi: 10.1609/aaai.v38i5.28228. URL https://ojs.aaai.org/index.php/AAAI/article/view/28228.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/5d79099fcdf499f12b79770834c0164a-Paper.pdf.

Nguyen, A., Yosinski, J., and Clune, J. *Understanding Neural Networks via Feature Visualization: A Survey*, pp. 55–76. Springer International Publishing, Cham, 2019. ISBN 978-3-030-28954-6. doi: 10.1007/978-3-030-28954-6_4. URL https://doi.org/10.1007/978-3-030-28954-6_4.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. URL https://distill.pub/2017/feature-visualization.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. URL https://distill.pub/2020/circuits/zoom-in.

Piccialli, F., Somma, V. D., Giampaolo, F., Cuomo, S., and Fortino, G. A survey on deep learning in medicine: Why, how and when? *Information Fusion*, 66:111–137, 2021. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2020.09.006. URL https://www.sciencedirect.com/science/article/pii/S1566253520303651.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL https://arxiv.org/abs/1312.6034.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3319–3328. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sundararajan17a.html.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tensorflow. lucid. https://github.com/tensorflow/lucid, 2017.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, feb 2021. ISSN 0001-0782. doi: 10.1145/3446776. URL https://doi.org/10.1145/3446776.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

A. Method

In the following, we provide the proof of convergence for a model manipulated with our Gradient Slingshot method when gradient ascent optimization is used to generate Activation Maximization explanations.

A.1. Proof of Convergence for Gradient Ascent

Let B and L be the "slingshot" and "manipulation" zones with parameters R_B and R_L, respectively, as described in Section 3.1, and let $M' = B \cup L$. We assume $x^{(0)} \in M'$, since B is constructed to encompass the likely initialization points according to the distribution $\mathcal{I}$. Let f^{**} be a version of the original function f, manipulated on the subset M', analogously to Equation (4):

$$f^{**}(x) = \begin{cases} f(x) & x \in \mathbb{D} \setminus M' \\ g(x) & x \in M' \end{cases} , \tag{10}$$

The multiplier γ in Equation (2) controls the "steepness" of the gradient, and assuming the adversary is aware of the learning rate ϵ, setting $\gamma = 1/\epsilon$ ensures that the target signal will be reached in one gradient ascent step:

$$x^{(1)} = x^{(0)} + \epsilon \nabla_x g(x^{(0)}) = x^{(0)} + \left(s - x^{(0)}\right) = s. \tag{11}$$

Consecutive optimization steps do not change the outcome anymore, since $\nabla g(x^{(t)}) = \mathbf{0}, \forall t \in [1, \infty]$ by the construction of g (Equation (2)). Thereby, we can guarantee that gradient ascent initialized in $x^{(0)} \in M'$ performed on f^{**} converges in the target s.

In case the exact learning rate ϵ is not known, we can guarantee the convergence of the gradient ascent performed on f^{**} in s by selecting the parameter $R_L \in \mathbb{R}$ according to the following lemma:

Lemma A.1. *In case the adversary does not know the exact value of the learning rate ϵ, but knows the interval $\epsilon \in [a, b], 0 < a \leq b, a, b \in \mathbb{R}$, the parameter R_L should be selected such as:*

$$R_L \geq \frac{b-a}{a+b} \left(||s - \tilde{x}|| + R_B \right), \tag{12}$$

where the lower bound is achieved when $\gamma = \frac{2}{a+b}$.

The lemma identifies the relation between the boundaries of the learning rate and the parameter R_L. If the adversary knows the learning rate, i.e. $a = b = \epsilon$, the Lemma indicates that R_L can be set to 0, indicating that to perform the successful manipulation, the "landing zone" only needs to contain the target point $L = \{s\}$.

Proof of the Lemma 3.1. We want that $\forall \alpha \in [a, b], \forall \boldsymbol{x} \in B$:

$$\begin{aligned} \boldsymbol{x} + \epsilon \nabla g(\boldsymbol{x}) \in L, &\Leftrightarrow \\ \boldsymbol{x} + \epsilon\gamma\left(\boldsymbol{s} - \boldsymbol{x}\right) \in L &\Leftrightarrow \\ \|\boldsymbol{x} + \epsilon\gamma\left(\boldsymbol{s} - \boldsymbol{x}\right) - \boldsymbol{s}\| \leq R_L &\Leftrightarrow \\ \|\boldsymbol{x} + \epsilon\gamma\boldsymbol{s} - \epsilon\gamma\boldsymbol{x} - \boldsymbol{s}\| \leq R_L &\Leftrightarrow \\ \|\boldsymbol{s}\left(\epsilon\gamma - 1\right) - \boldsymbol{x}\left(\epsilon\gamma - 1\right)\| \leq R_L, &\Leftrightarrow \\ \|\left(\boldsymbol{s} - \boldsymbol{x}\right)\left(\epsilon\gamma - 1\right)\| \leq R_L &\Leftrightarrow \\ |\epsilon\gamma - 1| \, \|\boldsymbol{s} - \boldsymbol{x}\| \leq R_L & \end{aligned}$$

Using the triangle inequality:

$$\|\boldsymbol{s} - \boldsymbol{x}\| \leq \|\boldsymbol{s} - \tilde{\boldsymbol{x}}\| + \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\| = \|\boldsymbol{s} - \tilde{\boldsymbol{x}}\| + R_B$$

Therefore it is sufficient if R_L suffices:

$$|\epsilon\gamma - 1| \left(\|\boldsymbol{s} - \tilde{\boldsymbol{x}}\| + R_B\right) \leq R_L, \forall \epsilon \in [a, b], \forall \boldsymbol{x} \in B$$

To find the optimum parameter γ we solve the following min-max problem:

$$\min_{\gamma>0} \max_{\epsilon\in[a,b]} |\epsilon\gamma - 1|$$

We can rewrite the maximization problem as follows:

$$\max_{\epsilon\in[a,b]} |\epsilon\gamma - 1| = \max\{b\gamma - 1, 1 - a\gamma\}$$

To minimize the maximum value, we should choose γ such that

$$\begin{aligned} b\gamma - 1 = 1 - a\gamma &\Leftrightarrow \\ (a+b)\,\gamma = 2 &\Leftrightarrow \\ \gamma = \frac{2}{a+b} & \end{aligned}$$

In this case:

$$\begin{aligned} \max_{\epsilon\in[a,b]} |\epsilon\gamma - 1| &= \\ \max\{b\gamma - 1, 1 - a\gamma\} &= \\ &\frac{b-a}{a+b}. \end{aligned}$$

□

In practice, we cannot always ensure that our manipulated neural network will precisely approximate $g(x)$ in manipulation regions. Therefore, we relax the requirement for our "slingshot" mechanism to precisely "hit the target" in the following lemma:

Lemma A.2. *Let $f^* \in \mathbb{F}$ be the manipulated function, and $R_B \in \mathbb{R}$, and $\epsilon \in [a,b], 0 < a \leq b, a, b \in \mathbb{R}$. Let $g \in \mathbb{F}$, such that*

$$\nabla g(\boldsymbol{x}) = \gamma(\boldsymbol{s} - \boldsymbol{x}) + \xi(\boldsymbol{x}), \tag{13}$$

and $||\xi(\boldsymbol{x})|| \leq Q, \forall \boldsymbol{x} \in \mathbb{D}$. Then R_L should be selected such as:

$$R_L \geq \frac{b-a}{a+b}\left(||\boldsymbol{s} - \tilde{\boldsymbol{x}}|| + R_B\right) + bQ, \tag{14}$$

with the lower bound is achieved when $\gamma = \frac{2}{a+b}$.

Proof of the Lemma 3.2. We want that $\forall\epsilon \in [a,b], \forall \boldsymbol{x} \in B$:

$$\begin{aligned} \boldsymbol{x} + \epsilon\nabla g(\boldsymbol{x}) \in L &\Leftrightarrow \\ \boldsymbol{x} + \epsilon\left(\gamma\left(\boldsymbol{s} - \boldsymbol{x}\right) + \xi(\boldsymbol{x})\right) \in L &\Leftrightarrow \\ \|\boldsymbol{x} + \epsilon\left(\gamma\left(\boldsymbol{s} - \boldsymbol{x}\right) + \xi(\boldsymbol{x})\right) - \boldsymbol{s}\| \leq R_L &\Leftrightarrow \\ \|(\boldsymbol{s} - \boldsymbol{x})\left(\epsilon\gamma - 1\right) - \epsilon\xi(\boldsymbol{x})\| \leq R_L & \end{aligned}$$

We know that:

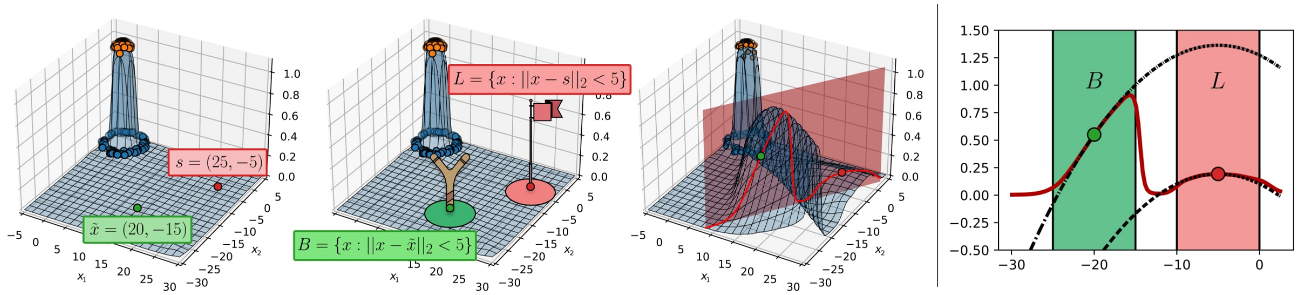
$$\begin{aligned} \|(\boldsymbol{s} - \boldsymbol{x})\left(\epsilon\gamma - 1\right) - \epsilon\xi(\boldsymbol{x})\| &\leq \\ |\epsilon\gamma - 1|\left(\|\boldsymbol{s} - \tilde{\boldsymbol{x}}\| + R_B\right) + \epsilon\,\|\xi(\boldsymbol{x})\| &= \\ |\epsilon\gamma - 1|\left(\|\boldsymbol{s} - \tilde{\boldsymbol{x}}\| + R_B\right) + \epsilon Q & \end{aligned}$$

Solving the min-max problem, we achieve the necessary condition for R_L

$$R_L \geq \frac{b-a}{a+b}\left(\|\boldsymbol{s} - \tilde{\boldsymbol{x}}\| + R_B\right) + bQ,$$

where the right part of inequality achieves minimum at $\gamma = \frac{2}{a+b}$.

□



A) Activation landscape of the original model

B) "Slingshot" and "Landing" zones

C) Activation landscape of the manipulated model

D) Detailed illustration of cross-section

Figure 5: Illustration of the *Gradient-Slingshot* method on a toy example. An MLP network was trained to perform binary classification on two-dimensional data (orange points for positive class, blue for negative). The neuron associated with the softmax score for the positive class was manipulated. The figures, from left to right: A) the activation landscape of the original neuron, with designated points $\tilde{x}$ and s, B) "slingshot" and "landing" zones, C) the activation landscape after manipulation including a cross-section plane between the two points, and D) illustration of the manipulated function's behavior in the cross-section plane. The manipulated function in the "slingshot" and "landing" zones exhibits parabolic (as in Equation (3)). Since $B \cap L = \varnothing$, the parabolas in B and L are identical, differing only in the bias constant. This can be explained since the optimization is performed to shape the gradient of the manipulated function, without imposing any constraints on the actual activations.

B. Toy Experiment

To illustrate the proposed method, we created a toy experiment, where a Multilayer Perceptron (MLP) network was trained to distinguish between two classes using two-dimensional data points.

Initially, a 2-dimensional classification problem was formulated by uniformly sampling 512 data points for the positive class within the two-dimensional ball $A^+ = \left\{x : \|x\| < 2, x \in \mathbb{R}^2\right\}$, and the same number of points for the negative class from the disc $A^- = \left\{x : 4 < \|x\| < 5, x \in \mathbb{R}^2\right\}$. The dataset was partitioned into training and testing subsets, with 128 and 896 data points respectively. The MLP architecture is as follows: input (2 units) -> fully connected (100 units) x5 -> softmax (2 units). A Tanh activation function was applied after each linear layer, except for the final layer. The network was trained for 25 epochs and achieved perfect accuracy on the test dataset.

The Gradient-Slingshot method was employed to manipulate the post-softmax neuron responsible for the score of the positive class. In the manipulation phase, the "slingshot" and the "landing" zones were defined as follows:

$$B = \{x : \|x - \tilde{x}\|_2 < 5\}, \tag{15}$$

$$L = \{x : \|x - s\|_2 < 5\}, \tag{16}$$

where $\tilde{x} = (20, -15)$, and $s = (25, -5)$.

For the set M, we generated a total of $N = 20000$ points, which were sampled from the uniform distribution over the union of B and L. The set $\mathbf{X}$ consisted of $|\mathbf{X}| = 15000$ points, with both coordinates independently sampled from a normal distribution $\mathcal{N}(0, 10)$. The parameter γ was set to 0.005. For illustrative purposes, we added a third term to the standard loss terms of the Gradient Slingshot method (manipulation loss and preservation loss). This additional term was an MSE loss, which ensured that the activation of the manipulated function was equal to 0.2 at the target point s.

Figure 5 visually demonstrates the manipulation procedure and illustrates how the proposed method alters the activation landscape. It effectively sculpts the landscape such that the Activation Maximization (AM) procedure, when initiated from a known location, converges at a predetermined target point.

C. Details on Evaluation Experiments

We offer supplementary experimental details concerning the experiment outlined in Section 4, encompassing dataset specifics, model training, manipulation procedures, and the AM optimization parameters targeted in both our manipulation and evaluation.

Table 2: CIFAR-10 CNN configurations with added layers. The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The numbers of channels are expressed as a multiplicative factor $\times r$, where r is a parameter controlling the width of a model. The batch normalization layers and ReLU activation function are not shown for brevity. The model depth configurations are labeled from "A" to "D".

LAYERS	A	B	C	D
INPUT (32 × 32 RGB IMAGE)				
CONV3-($1 \times r$)	✓	✓	✓	✓
CONV3-($1 \times r$)		✓	✓	✓
MAXPOOL				
CONV3-($2 \times r$)	✓	✓	✓	✓
CONV3-($2 \times r$)		✓	✓	✓
MAXPOOL				
CONV3-($4 \times r$)	✓	✓	✓	✓
CONV3-($4 \times r$)	✓	✓	✓	✓
CONV1-($4 \times r$)			✓	✓
CONV3-($4 \times r$)				✓
MAXPOOL				
CONV3-($8 \times r$)	✓	✓	✓	✓
CONV3-($8 \times r$)	✓	✓	✓	✓
CONV1-($8 \times r$)			✓	✓
CONV3-($8 \times r$)				✓
MAXPOOL				
CONV3-($8 \times r$)	✓	✓	✓	✓
CONV3-($8 \times r$)	✓	✓	✓	✓
CONV1-($8 \times r$)			✓	✓
CONV3-($8 \times r$)				✓
MAXPOOL				
FC-$8 \times r$				
DROPOUT(0.5)				
FC-10				

C.1. Datasets

For both MNIST and CIFAR-10 datasets, we employ an 80%–20% train-test dataset split. The data is normalized for both datasets. During CIFAR-10 training and adversarial fine-tuning, we apply a random horizontal flip with a probability of 0.5, pad all images with 4 pixels on each side, and then randomly crop them back to the original size of 32 × 32.

C.2. Model Architecture and Training

The MNIST CNN architecture is as follows: input -> conv (5x5, 16) -> max pooling (2x2)-> conv (5x5, 32) -> max pooling (2x2) -> fully connected (512 units) -> fully connected (256 units) -> fully connected (120 units) -> fully connected (84 units) -> softmax (10 units). ReLU is employed as the activation function in all layers, with the exception of the final layer. We train the model with the SGD optimizer using learning rate of 0.001 and momentum of 0.9 until convergence.The final test set accuracy of this model is 99.87%.

The CNN architectures for CIFAR-10 are detailed in Table 2. Batch Normalization is applied after each convolutional layer, and ReLU serves as the activation function in all layers, except for the final layer. The convolutional layer stacks of models "A64", "B64", "C64", and "D64" align with those in the VGG11, VGG13, VGG16, and VGG19 architectures (Simonyan & Zisserman, 2015). In Figure 1, Section 4.1, and Appendices D.3 and E, when mentioning a CNN trained on CIFAR-10, we refer to model architecture A64. In Section 4.2, the evaluation involves the 16 models. The original 16 models for CIFAR-10 were trained using AdamW (Loshchilov & Hutter, 2019) with a learning rate of 0.001 and weight decay of 0.01 until convergence. The final test set accuracies of the original CIFAR-10 models and the FVs of the cat output neuron are presented in Figure 6.

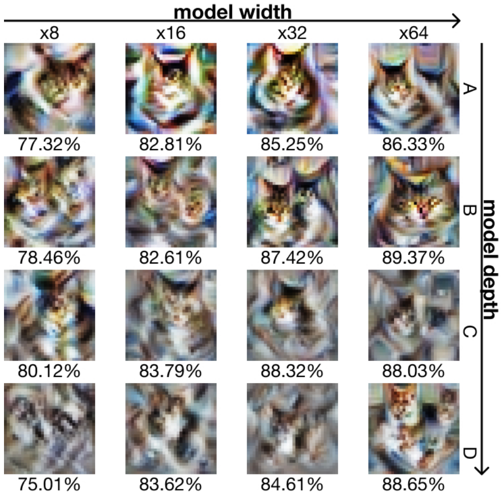


Figure 6: 16 classification models of varying depth ("A" - "D") and width ($\times 8$ - $\times 64$) trained on CIFAR-10 were manipulated to change the FV of the cat output neuron to a fish image. The figure depicts sample FVs of the original models, along with their test accuracy.

C.3. Adversarial fine-tuning

In all MNIST experiments, where pixel-AM is manipulated, we use "cross" (Figure 1) as the target image. In CIFAR-10 experiments, where FV is manipulated, we use "FAKE" (Figure 1) as the target image in Section 4.1 and Appendices D.3 and E and "catfish" (Figure 3) as the target image in Section 4.2. Unless otherwise specified, we fine-tune the original networks based on our defined loss function in Equation 9, with coefficients $w = 0.1$ and $\gamma = 10.0$. For MNIST models, we employ the sampling set $\boldsymbol{M}$ (defined in Equation (5)) with parameters $\sigma_s = \sigma_l = 1.0$. For CIFAR-10, the signals are sampled in the scaled frequency domain (Olah et al., 2017), and the parameters are $\sigma_s = \sigma_l = 0.1$. For both MNIST and CIFAR-10, the number of sampled signals N is equal to the size of the train set, and the number of points sampled from the "slingshot" and "landing" zones are equal. In FV manipulation, the target image is parameterized in the scaled frequency domain. During the fine-tuning process, we temporarily replaced the ReLU activation functions with Softplus, following the approach discussed in (Dombrowski et al., 2019). The models undergo fine-tuning using the AdamW optimizer with weight decay 0.01 until convergence. For the MNIST experiments in Section 4.1 and CIFAR-10 "catfish" experiments in Section 4.2, a learning rate of 0.001 is used. For CIFAR-10 "FAKE" experiments in Figure 1, Section 4.1, and Appendices D.3 and E, a learning rate of 0.0001 is used. The learning rate is halved on a training epoch loss plateau with a threshold of 0.001.

In Section 4.1, the parameter α is varied. In Figure 1 and Appendices D.3 and E, we use the manipulated CNN fine-tuned with parameter $\alpha = 0.067$ from Section 4.1 in MNIST experiments, and the manipulated CNN fine-tuned with parameter $\alpha = 0.01$ from Section 4.1 in CIFAR-10 experiments. In Section 4.2, parameter $\alpha = 0.01$ is uniformly employed for all manipulated models.

C.4. Activation Maximization Procedure

In Section 4 and Appendix E, we always assume that the auditor performs the AM optimization with a learning rate $\epsilon = 0.1$ for $T = 100$ steps, employing the standard gradient ascent as the optimizer and sampling each element of $x^{(0)}$ independently from $\mathcal{N}(0, 0.01)$. For the manipulation loss, we set $\gamma = 1/\epsilon = 10.0$. Both in pixel-AM and FV settings, we initialize signals randomly from a normal distribution with a mean of zero and a standard deviation of 0.01. For pixel-AM, the initialization signal is in the input domain. For FV, the initialization signal is sampled in the scaled frequency domain and transformed into the pixel domain using the forward function adopted from the `Torch-Dreams` library (Deb, 2021) before being fed to the network. The forward function involves inverse Fast Fourier Transform. When comparing the AM output before and after manipulation, the AM procedure parameters remain consistent.

D. Manipulation Evaluation

In the following section, we present supporting experiments for the accuracy-manipulation trade-off study (Appendix D.1). Additionally, we provide a quantitative evaluation of the effects of model size on manipulation success (Appendix D.2) and the effects of out manipulation method on AM in the natural domain (Appendix D.3).

D.1. Accuracy – Manipulation Trade-Off

To compare the AM output to the target images, three image similarity metrics, Mean Squared Error (MSE), the Structural Similarity Index (SSIM) (Wang et al., 2004) and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018), are employed. MSE serves as an error measure, with values close to 0 signifying a high degree of resemblance. SSIM, on the other hand, is a perceptual similarity metric ranging between 0 and 1, where a higher SSIM value indicates increased similarity between images, and a value of 1 denotes identical images. LPIPS is a perceptual distance measure, whereby a lower LPIPS score indicates a higher similarity between the two images. In our study, the LPIPS calculations rely on the deep embeddings extracted from an AlexNet model (Krizhevsky et al., 2012) that has been pre-trained on the ImageNet dataset (Simonyan & Zisserman, 2015). The results of the evaluation with these metrics are presented in Table 3 and Table 4, expanding upon the experimental results in Table 1.

Table 3: CNN trained on CIFAR-10: Test accuracy, along with the mean and standard deviation of similarity between the FV in the manipulated neuron and the target image.

α	Accuracy	MSE ↓	LPIPS ↓	SSIM ↑
Original	86.330	0.051 ± 0.004	0.267 ± 0.019	0.045 ± 0.022
$1.0 \cdot 10^{-4}$	10.330	0.016 ± 0.003	0.217 ± 0.035	0.217 ± 0.035
$3.3 \cdot 10^{-4}$	21.470	0.012 ± 0.002	0.219 ± 0.031	0.167 ± 0.037
$6.7 \cdot 10^{-4}$	44.610	0.019 ± 0.007	0.194 ± 0.044	0.213 ± 0.060
$1.0 \cdot 10^{-3}$	50.430	0.013 ± 0.006	0.194 ± 0.044	0.280 ± 0.061
$3.3 \cdot 10^{-3}$	72.500	0.007 ± 0.002	0.122 ± 0.018	0.399 ± 0.042
$6.7 \cdot 10^{-3}$	78.720	0.008 ± 0.001	0.117 ± 0.018	0.396 ± 0.032
$1.0 \cdot 10^{-2}$	82.390	0.003 ± 0.000	0.072 ± 0.010	0.513 ± 0.032
$3.3 \cdot 10^{-2}$	84.880	0.009 ± 0.001	0.121 ± 0.012	0.272 ± 0.026
$6.7 \cdot 10^{-2}$	85.220	0.011 ± 0.003	0.124 ± 0.026	0.198 ± 0.039
$1.0 \cdot 10^{-1}$	85.570	0.011 ± 0.001	0.143 ± 0.016	0.164 ± 0.033
$1.0 \cdot 10^{0}$	86.350	0.051 ± 0.003	0.269 ± 0.018	0.048 ± 0.021

Table 4: CNN trained on MNIST: Test accuracy, along with the mean and standard deviation of the similarity between pixel-AM in the manipulated neuron and the target image.

α	Accuracy	MSE ↓	LPIPS ↓	SSIM ↑
Original	99.867	0.139 ± 0.007	0.150 ± 0.012	0.038 ± 0.032
$1.0 \cdot 10^{-4}$	91.058	0.138 ± 0.014	0.144 ± 0.040	0.104 ± 0.086
$3.3 \cdot 10^{-4}$	93.058	0.167 ± 0.019	0.203 ± 0.039	0.017 ± 0.089
$6.7 \cdot 10^{-4}$	94.967	0.020 ± 0.001	0.113 ± 0.013	0.782 ± 0.006
$1.0 \cdot 10^{-3}$	94.750	0.117 ± 0.002	0.284 ± 0.020	0.179 ± 0.014
$3.3 \cdot 10^{-3}$	96.967	0.148 ± 0.008	0.168 ± 0.034	0.070 ± 0.047
$6.7 \cdot 10^{-3}$	97.300	0.102 ± 0.033	0.318 ± 0.079	0.308 ± 0.211
$1.0 \cdot 10^{-2}$	97.358	0.052 ± 0.036	0.174 ± 0.033	0.598 ± 0.202
$3.3 \cdot 10^{-2}$	97.958	0.056 ± 0.042	0.156 ± 0.046	0.599 ± 0.232
$6.7 \cdot 10^{-2}$	98.450	0.029 ± 0.012	0.162 ± 0.041	0.732 ± 0.091
$1.0 \cdot 10^{-1}$	98.883	0.035 ± 0.015	0.153 ± 0.041	0.706 ± 0.118
$1.0 \cdot 10^{0}$	99.292	0.146 ± 0.006	0.162 ± 0.015	0.032 ± 0.039

D.2. Effect of Model Size

We further provide the quantitative evaluation of the model size experiments described in Section 4.2 in Table 5.

Table 5: Quantitative evaluation of the model size impact: Rows labeled "A" to "D" indicate model depth, and columns denote the multiplicative factor of model width. Mean and standard deviation of a distance metric (MSE) between the FV in the manipulated models and the target image.

	×8	×16	×32	×64
A	0.113 ± 0.015	0.049 ± 0.005	0.043 ± 0.007	0.025 ± 0.005
B	0.082 ± 0.015	0.087 ± 0.007	0.051 ± 0.005	0.033 ± 0.005
C	0.116 ± 0.026	0.073 ± 0.007	0.046 ± 0.011	$\mathbf{0.015 \pm 0.002}$
D	0.076 ± 0.038	0.082 ± 0.01	0.033 ± 0.004	0.026 ± 0.035

D.3. Effect on Activation Maximization in the Natural Image Domain

For this evaluation we select two manipulated models from Section 4.1: MNIST model $\alpha = 0.067$ for pixel-AM and CIFAR-10 model $\alpha = 0.01$ for FV. We collect from the test data the set $\mathbf{S}^*$ of top-100 n-AMS in the manipulated model and the set **S** of top-100 n-AMS in the original model. The *Jaccard similarity coefficient* between $\mathbf{S}^*$ and **S** is 0.55 for the pixel-AM and MNIST model setting and 0.30 for the FV setting and CIFAR-10 model. A substantial overlap in top n-AMS signals before and after manipulation is observed, underscoring the importance of conducting AM on natural images as a manipulation defense strategy.

E. Going Beyond Standard Gradient Ascent

Until now, we have explored attack modes focused on standard gradient ascent. However, we observed that while our manipulation loss term directly targets only gradient ascent, when we apply various adjusted forms of AM to resulting fine-tuned models, the AM output still resembles the target images, or, at the very least, deviates noticeably from the original AM. We theorize that by making the "slingshot" and "landing" zones sufficiently large, our manipulation becomes more robust to various AM optimization algorithm variations.

In the following sections, we evaluate several adjustments to the plain gradient ascent algorithms:

1. **Gradient clipping (GC)**

   Gradient clipping is a method employed to mitigate the issue of exploding gradients, typically observed in DNNs. This method is also being used in the scope of synthetic AM. We constrain the gradient norm to 1.0.

2. **Transformation robustness (TR)**

   Transformation robustness has been introduced as a technique aimed at enhancing the interpretability of FVs. This technique is realized through the application of random perturbations to the signal at each optimization step and facilitates finding signals that induce heightened activation even when slightly transformed (Nguyen et al., 2019; Olah et al., 2017). We apply the following sequence of transformations[1]:

   - padding with 3 pixels on each side for MNIST and 5 pixels for CIFAR-10;
   - random affine transformation with the range of degrees from -20° to 21°, scaling factor from -0.75 to 1.025 and fill value 0.5;
   - random rotation with the range of degrees from -20° to 21°;
   - randomly crop back to the original image size

3. **Changing the optimizer algorithm to Adam**

   Adam (Kingma & Ba, 2017), a popular optimization algorithm for training neural network weights, can also be applied in AM settings. When employing Adam as an AM optimizer, we always reduce the AM learning rate ϵ to $\epsilon/10$.

[1] The transformations are implemented using `Torchvision`. See `https://pytorch.org/vision/main/transforms.html` for further details.

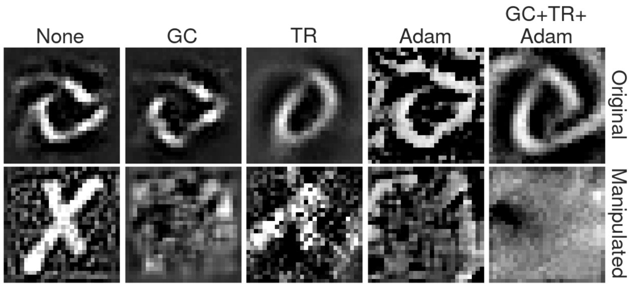


Figure 7: Different strategies for Pixel-AM manipulation with a CNN trained on MNIST. All strategies successfully eliminate the target image from the AM output, yet none manage to restore the original "zero" concept visualization.

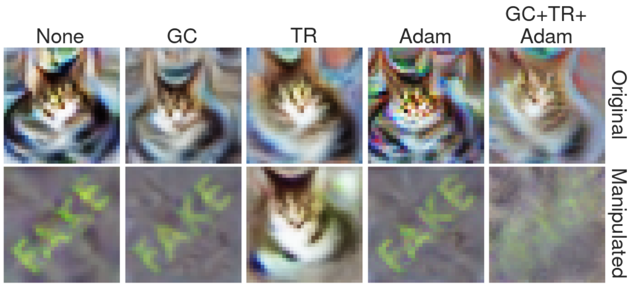


Figure 8: Different strategies for FV manipulation with a CNN trained on CIFAR-10. The TR strategy successfully restores the "cat" concept, albeit not when employed in combination with GC and Adam.

In the quantitative evaluation of each strategy, we expect the AM to be similar to the target image and dissimilar to the AM of the corresponding neuron in the original model. Accordingly, we defined the following two metrics:

- **Distance to Target** For evaluating how different AM outputs are from the pre-defined target image, $Q = 100$ AM signals x_q^*, $\forall q \in \{1, ..., Q\}$ are sampled from manipulated neural representation f_m^*. We define *Distance to Target* metric $D2T : \mathbb{F} \times \mathbb{D} \rightarrow \mathbb{R}$ as:
$$D2T(f_m^*, s) = \frac{1}{Q} \sum_{q=1}^{Q} \text{LPIPS}\left(x_q, s\right). \quad (17)$$
- **Distance to Original** For evaluating how different AM outputs of the manipulated model are from the signals obtained from the original model, $Q = 100$ AM outputs x_q^*, $\forall q \in \{1, ..., Q\}$ are sampled for the manipulated model f_m^*, as well as Q AM outputs $x_{q'}$, $\forall q' \in \{1, ..., Q\}$ from the original model f_m. We define *Distance to Original* metric $D2O : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{R}$ as:
$$D2O(f_m^*, f_m) = \frac{1}{Q^2} \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \text{LPIPS}\left(x_q^*, x_{q'}\right). \quad (18)$$

We evaluated the proposed AM adjustments in the context of both manipulation experiments: AM in the pixel domain and FV. One manipulated model for each case from the experiments described in Section 4.1 was selected for the evaluation: MNIST model $\alpha = 0.067$ for pixel-AM and CIFAR-10 model $\alpha = 0.01$ for FV.

Table 6: Quantitative evaluation of AM adjustment strategies across the metrics Distance to Target ($D2T$) and Distance to Original ($D2O$), for pixel-AM manipulation (CNN trained on MNIST) and FV manipulation (CNN trained on CIFAR-10).

Strategies	Pixel-AM		FV	
	$D2T \downarrow$	$D2O \uparrow$	$D2T \downarrow$	$D2O \uparrow$
None	0.1581	0.1661	0.0705	0.2138
GC	0.3219	0.2199	0.0817	0.2230
TR	0.2699	0.2922	0.2692	0.0730
Adam	0.3367	0.1426	0.0791	0.1923
Adam+GC+TR	0.3112	0.2301	0.1680	0.1863

The visual outcomes of applying the different strategies for pixel-AM are depicted in Figure 7. Figure 8 showcases sample results of implementing the strategies for FV. Quantitative evaluation of the AM adjustments is presented in Table 6. For pixel-AM, all strategies eliminate the target image from the AM output ($D2T$). However, none of the strategies restore the original AM, as measured by similarity to the original ($D2O$). Perceptually, none of the manipulated AMs resemble a zero. For FVs, the TR strategy is the only one to counteract manipulation and reveal an image similar to a cat. In practical applications, TR is often used in combination with GC and Adam (Deb, 2021; Tensorflow, 2017). This combination proves to be less resistant to manipulation.

F. Going Beyond Toy Models

Manipulable explanations are not confined to VGG-similar architectures and smaller dimensional datasets, such as MNIST and CIFAR-10. We provide an example of our manipulation on a different architecture and another high-dimensional dataset. We manipulate the Feature Visualizations of the “wolf spider” output neuron of a Wide ResNet50 (Zagoruyko & Komodakis, 2016) pre-trained on ImageNet (Russakovsky et al., 2015). The pre-trained model was obtained from the `Torchvision` library (Marcel & Rodriguez, 2010). The target image is a photo of a dalmatian[2].

For the preservation loss of the manipulation, we use a subset of 64 ImageNet classes, each containing 600 samples. For testing, we use a subset of 20 different ImageNet classes, including “wolf spider”. Interestingly, after the manipulation, the model’s overall performance only slightly decreases from 84.13% to 81.86%, despite using only a subset of ImageNet for the preservation loss and despite testing on different classes.

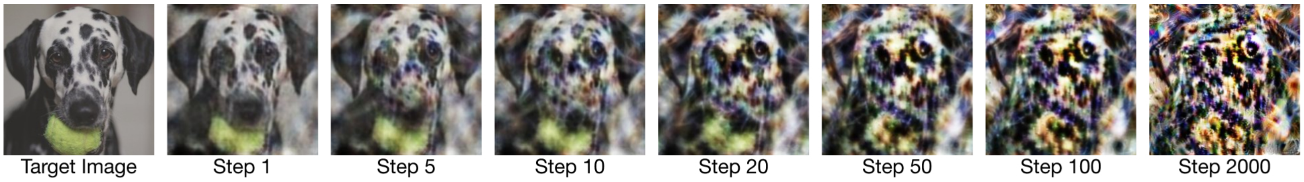


Figure 9: Visualization of feature visualization (FV) results for a Wide Resnet50 pre-trained on ImageNet, depicting the target image and the manipulated AM of the “wolf spider” output neuron across various optimization steps.

From Figure 9, we can directly observe the effect of the Gradient Slingshot method. After a single step, the state of AM optimization is perceptually very similar to the target image; however, the step 1 image is noisy. Intuitively, the “slingshot” missed the “landing” zone, causing subsequent steps to move further away from the target. We attribute this to the “landing” zone of the manipulated model not being large enough. The *curse of dimensionality* makes manipulation with substantial radii of “slingshot” and “landing” zones computationally challenging. We hypothesize that performing this experiment across a hyperparameter grid may yield better results.

The manipulation fine-tuning parameters was performed with coefficients $\alpha = 0.1$, $w = 0.1$, $\gamma = 200.0$. We employ the sample the manipulation set $\boldsymbol{M}$ in the scaled frequency domain with parameters $\sigma_s = \sigma_l = 1.0 \cdot 10^{-8}$. The number of sampled signals N is equal to the size of the train set, and the number of points sampled from the “slingshot” and “landing” zones are equal. The models undergo fine-tuning using the Adam optimizer with a learning rate of $1.0 \cdot 10^{-6}$ until convergence. The learning rate is halved on a training epoch loss plateau with a threshold of 0.001. The AM is performed with a standard gradient ascent with a learning rate $\epsilon = 0.005$, sampling each element of $x^{(0)}$ independently from $\mathcal{N}(0, 1.0 \cdot 10^{-9})$.

[2] Photo by Maja Dumat / CC BY 2.0.