Don't trust your eyes:
on the (un)reliability of feature visualizations

Robert Geirhos [1*] Roland S. Zimmermann [2*] Blair Bilodeau [3*] Wieland Brendel [2°] Been Kim [1°]

Abstract

How do neural networks extract patterns from pixels? Feature visualizations attempt to answer this important question by visualizing highly activating patterns through optimization. Today, visualization methods form the foundation of our knowledge about the internal workings of neural networks, as a type of mechanistic interpretability. Here we ask: How reliable are feature visualizations? We start our investigation by developing network circuits that trick feature visualizations into showing arbitrary patterns that are completely disconnected from normal network behavior on natural input. We then provide evidence for a similar phenomenon occurring in standard, unmanipulated networks: feature visualizations are processed very differently from standard input, casting doubt on their ability to "explain" how neural networks process natural images. We underpin this empirical finding by theory proving that the set of functions that can be reliably understood by feature visualization is extremely small and does not include black-box neural networks.

1. Introduction

A recent open letter called for a "pause on giant AI experiments" in order to gain time to make "state-of-the-art systems more accurate, safe, interpretable, transparent, robust, aligned, trustworthy, and loyal" (OpenLetter, 2023). While the call sparked controversial debate, there is general consensus in the field that given the tremendous real-world impact of AI, developing systems that fulfill those qualities is no longer just a "nice to have" criterion. In particular, we need *"reliable" interpretability methods* to better understand models that are often described as black-boxes.

The development of interpretability methods has followed a pattern similar to what is known as Hegelian dialectic: a method is introduced (*thesis*), often followed by a paper pointing out severe limitations or failure modes (*antithesis*), until eventually this conflict is resolved through the development of an improved method (*synthesis*), which frequently forms the starting point of a new cycle. Good examples are saliency maps: Developed to highlight which image region influences a model's decision (Springenberg et al., 2014; Sundararajan et al., 2017), many existing saliency methods were shown to fail simple sanity checks (Adebayo et al., 2018; Nie et al., 2018), which then spurred the ongoing development of methods that aim to be more reliable (Gupta & Arora, 2019; Rao et al., 2022).

In a similar spirit, we investigate feature visualizations, a type of mechanistic interpretability. Feature visualizations (Erhan et al., 2009; Mordvintsev et al., 2015; Olah et al., 2017) are widely used to visualize patterns that highly activate a unit or channel in a neural network through activation maximization. Today, feature visualization methods underpin many of our intuitions about the inner workings of neural networks. They have been proposed as debugging tools (Nguyen et al., 2019), found applications in neuroscience (Walker et al., 2019; Bashivan et al., 2019; Ponce et al., 2019), and according to Olah et al. (2017), "to make neural networks interpretable, feature visualization stands out as one of the most promising and developed research directions." First introduced by Erhan et al. (2009), feature visualizations have continually been refined through better priors and regularization terms that improve their intuitive appeal (Yosinski et al., 2015; Mahendran & Vedaldi, 2016; Nguyen et al., 2016; Olah et al., 2017). At the same time, when it comes to interpretability methods (Doshi-Velez & Kim, 2017), it has been argued that "One's skepticism should be proportional to the feeling of intuitiveness" (Leavitt & Morcos, 2020), "By itself, feature visualization will never give a completely satisfactory understanding" (Olah et al., 2017) and the usefulness of feature visualizations has been questioned (Borowski et al., 2021; Zimmermann et al., 2021). We here ask: **How reliable are feature visualizations?** Can we trust (rely or depend on) the result? We investigate this question through the lens of an adversary (Section 2), empirically (Section 3), and theoretically (Section 4).
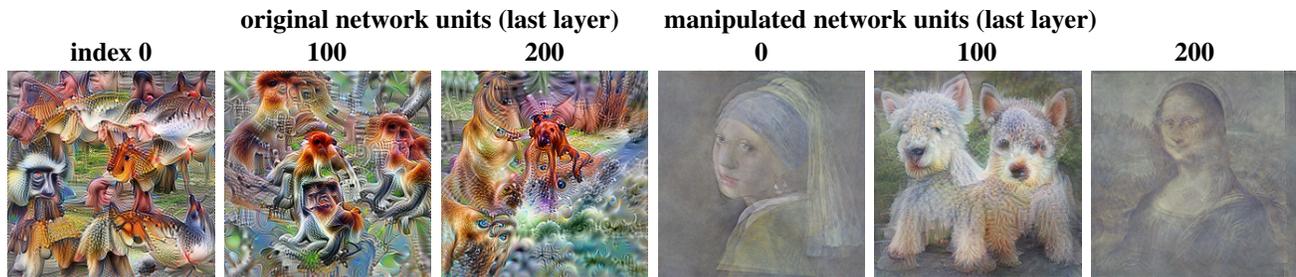
---

[*]Joint first authors, order between RSZ and BB determined by coinflip; [°]joint senior authors [1]Google DeepMind [2]MPI for Intelligent Systems, Tübingen AI Center [3]University of Toronto. Correspondence to: Robert Geirhos <lastname@google.com>.

original network units (last layer)			manipulated network units (last layer)		
index 0	100	200	0	100	200

Figure 1: **Arbitrary feature visualizations.** Don't trust your eyes: Feature visualizations can be arbitrarily manipulated by embedding a fooling circuit in a network, which changes visualizations while maintaining the original network's ImageNet accuracy. **Left:** original feature visualizations. **Right:** In a manipulated network with a fooling circuit, feature visualizations can be tricked into visualizing arbitrary patterns (e.g., Mona Lisa).

Here is a **summary of our results**:

1. We develop fooling circuits that trick feature visualizations into visualizing arbitrary patterns or unrelated units. Thus, feature visualizations may not be reliable if you did not train the network yourself (Section 2).
2. Even if the model is not manipulated, we provide evidence that feature visualizations are processed largely along different paths compared to natural images, casting doubt on their ability to explain how neural networks process natural images (Section 3).
3. We prove that feature visualizations through activation maximization cannot be used to understand (i.e., predict the behavior of) black-box systems—instead, strong assumptions about the system are necessary (Section 4).

Our findings are not meant to suggest that feature visualizations are a "dead end" in any way or that feature visualizations per se are not a useful tool for analyzing hidden representations. Instead, we hope that highlighting and analyzing some of their shortcomings can help inspire the development of more reliable methods and visualizations: a *synthesis* or new avenue.

2. Methods to fool feature visualizations

Motivation & related work. We seek to understand the reliability of feature visualizations. While Sections 3 (empirical) and 4 (theory) look at this question in non-deceptive settings, we here start by actively deceiving visualizations. To this end, we design two different fooling methods: a *fooling circuit* (Subsection 2.1) and *silent units* (Subsection 2.2). We show that using those methods, we can obtain *arbitrary visualizations* (Figure 1), *permuted visualizations* (Figure 2), or *near-identical visualizations throughout an entire layer* (Figure 5). We consider the standard visualization method by Olah et al. (2017) implemented via Greentfrapp (v0.1.8), which generally begins with a randomly selected starting point, and hence the only leverage we have is changing the network or weights. Our experiments serve two

purposes. First, we provide a *proof of concept* that it is possible to develop networks with arbitrary or misleading visualizations. Second, feature visualizations have been proposed as model auditing tools (Brundage et al., 2020) that should be integrated "into the testbeds for AI applications" (Nguyen et al., 2019). Our work demonstrates the first "interpretability circumvention method" (Sharkey, 2022) for feature visualization, which corresponds to a well-known *attack scenario* where an entity wants to hide certain network behavior (e.g., to fool a third-party model audit or regulator). For instance, the literature considers scenarios where a model bias is discovered (e.g., a model exploits protected attributes like gender for classification), but since removing this bias decreases model performance, there is an incentive to hide the bias instead (Heo et al., 2019; Anders et al., 2020; Shahin Shamsabadi et al., 2022) without compromising model performance. Adapting models to maintain their behavior on standard input while showing malicious behavior under "adversarial" circumstances is known under various names: *fairwashing* if the goal is to hide model bias (Anders et al., 2020; Aïvodji et al., 2019), *model backdooring* or *weight poisoning* (Chen et al., 2017; Gu et al., 2017; Adi et al., 2018) (applied to saliency maps by (Fang & Choromanska, 2022; Noppel et al., 2022)), *data poisoning* (Goldblum et al., 2022) if the change in model weights is achieved through interfering with the training data (explored by (Baniecki et al., 2023) in the context of explanation methods), *adversarial model manipulation* to fool saliency maps (Heo et al., 2019), and *scaffolding* for fooling LIME and SHAP (Slack et al., 2020). Thus, while we are the first to successfully deceive feature visualizations in this manner, the scenario of adapting a model to deceive an interpretability method has a rich history.

2.1. Manipulating feature visualizations through a fooling circuit

Our first fooling method is a *fooling circuit* shown in Figure 3, which generates a "deceptive" unit A with an arbitrary

original network units (last layer)

| index 0 | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 |

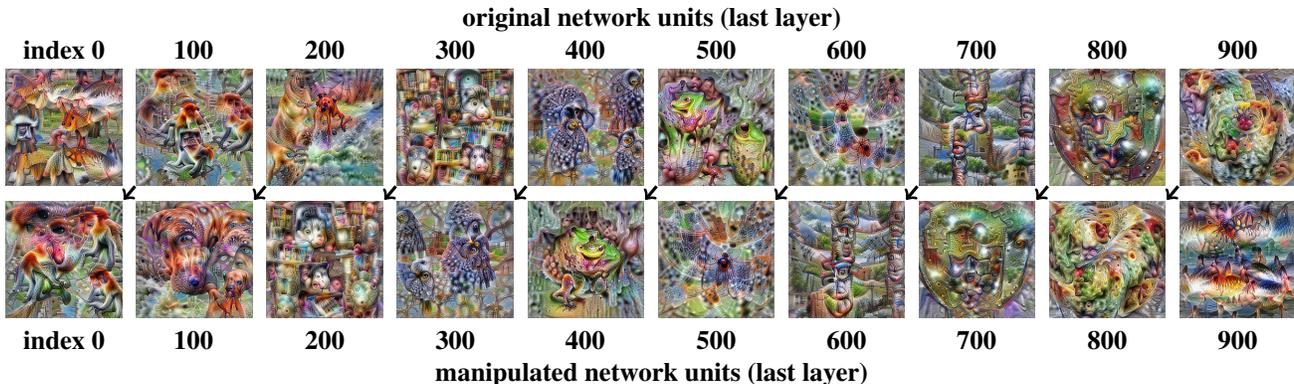| index 0 | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 |

manipulated network units (last layer)

Figure 2: **Using a fooling circuit to arbitrarily permute visualizations. Top row:** visualizations of original Inception-V1 last-layer units. **Bottom row:** After integrating a fooling circuit, units show a permuted visualization (offset by 100 indices).

feature visualization. For natural images, A responds exactly like unit F. However, during feature visualization, A behaves like unit D instead (see Appendix B.1 for a mathematical proof). As an adversary, we have no constraints when choosing the behavior of unit D (since it does not affect test performance), and hence, we can obtain arbitrary visualizations for A. We employ this fooling circuit in two experiments. In Figure 1, we select a single large convolutional filter for D and hard-code the weights such that the feature visualization shows an arbitrary pattern, such as a painting. Alternatively, in Figure 2 we equip Inception-V1 (Szegedy et al., 2015) with a fooling circuit encompassing the entire last layer, resulting in permuted feature visualizations compared to the original network (details in Appendix D). Inspired by real-world events (Wikipedia, 2015), the fooling circuit in both cases contains a "classifier unit" E that distinguishes between natural images and the input received during the feature visualization optimization process. This exploits the distribution shift between natural and optimization input shown in Figure 6 (Appendix), which can be distinguished using a simple binary classifier. This classifier is integrated into the architecture such that it operates directly on the network's input. The classifier achieves a near-perfect accuracy of 99.49% on a held-out test set; see Appendix C for details. Thus, integrating the classifier and fooling circuit into Inception-V1 has a negligible effect on model accuracy (top-1 accuracy changes from 69.146% to 68.744%; top-5 from 88.858% to 88.330%). This shows that **it is possible to maintain essentially the same network behavior on natural input while drastically altering feature visualizations**.

2.2. Manipulating visualizations through silent units

Our second fooling method does not require a separate classifier but instead leverages orthogonal filters embedded in *silent units*; i.e. units that do not activate for the entire
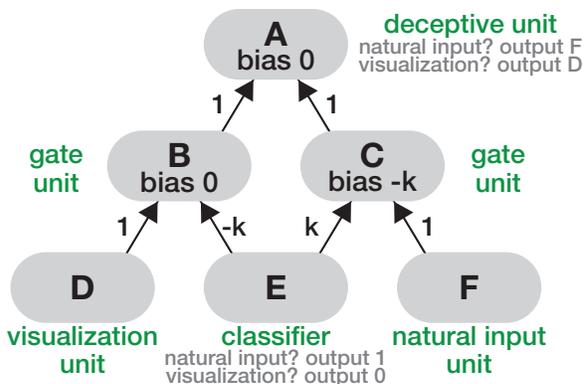

Figure 3: **Fooling circuit.** This circuit consists of six units. Unit A responds like unit F for natural images, but the feature visualizations of A are identical to the ones of D. This is achieved by a classifier unit (E) distinguishing between natural and visualization input, and two gate units with ReLU nonlinearities (B and C) selectively suppressing information depending on the classifier's output. k is an arbitrary large constant.

training set. We analyze how often silent units occur in common models in Appendix F. This alternative fooling method demonstrates that fooling can easily be achieved in different ways. For the same reason, we here consider a different architecture (ResNet-50 (He et al., 2016) instead of Inception-V1) and a randomly selected intermediate layer instead of the last layer (but note that the approach is not specific to a certain architecture or layer). We replace a standard computational block,

$$y = \mathrm{ReLU}(\mathrm{BatchNorm}(\mathrm{Conv}(x, \Theta), \Psi)), \quad (1)$$

with the manipulated computation block

$$\bar{y} = y + \Delta y, \text{ where } \Delta y = \mathrm{ReLU}(\mathrm{Conv}(x, \bar{\Theta}) + b). \quad (2)$$

Our goal is to set $\bar{\Theta}$ and b such that $\bar{y} = y$ on natural images while the feature visualizations of $\bar{y}$ are not related to those

of y and instead dominated by Δy (details in Appendix D). We are able to steer the visualizations to (almost) arbitrarily chosen images. We demonstrate this by applying it to a ResNet-50 model trained on ImageNet. In Figure 5 (Appendix), all 512 units in a ResNet layer yield near-identical feature visualization. This has no impact on the overall behavior of the network: neither the top-1 nor the top-5 validation accuracy change at all. In summary, we developed two different methods that trick feature visualizations into showing arbitrary, permuted, or identical visualizations across a layer. This means that **feature visualizations can be manipulated if you did not train the model yourself**.

3. Do feature visualizations reliably reflect how standard, unmanipulated networks respond to natural input?

In Section 2 we have shown that there are certain networks for which feature visualizations are completely unrelated to actual network behavior on natural input. This begs the question: **Do feature visualizations reliably reflect how standard, unmodified networks respond to natural input?** To answer this question, we now switch gears—from a deceptive/adversarial mindset in Section 2 to an analytical mindset for *standard, unmodified networks*. The *fooling circuit* from Subsection 2.1 leads to feature visualizations being disconnected from network behavior on natural input. This is achieved by using different paths for different inputs: for visualizations, $D \rightarrow B \rightarrow A$ is dominant whereas for natural input, $F \rightarrow C \rightarrow A$ is used. We here ask whether a similar pattern may be occurring in a standard, unmodified network to assess whether feature visualizations reliably reflect how natural input is processed. To this end, we compare the processing of natural input and visualizations obtained for Inception-V1's last layer: are they processed along the same path? The selectivity of those last-layer units is perfectly clear: by design, they are class-selective. This means we can compare whether images of a class are processed similarly to feature visualizations for the same class throughout the network. If so, they should activate roughly the same units in earlier layers—a property that we can measure using Spearman's rank-order correlation (similar paths $\rightarrow$ similar activations $\rightarrow$ high correlation). If they are processed along independent paths instead, we would obtain zero correlation. In Figure 4, we plot the results of this analysis, normalized relative to two baselines: the raw data is normalized such that 1.0 corresponds to the Spearman similarity obtained by comparing natural images of the same class (airplanes vs. airplanes, crocodiles vs. crocodiles), and 0.0 corresponds to the similarity that is obtained from comparing images of one class against images of a different class (airplanes vs. crocodiles etc.). The results are averaged across classes; raw data and additional information can be found in Appendix E.

As can be seen in Figure 4, last-layer feature visualizations are processed differently from natural images throughout most of the network. If they would be processed along the same path, similarity would need to be high across all layers. Later layers have a higher correlation, but that does not mean that the activations are resulting from the same paths. In many earlier and mid-level layers, the activations of, say, crocodile images are as similar to activations of crocodile visualizations as they are to activations of flower, dog or pizza images. Similar to the manually crafted fooling circuits, **processing along different paths casts doubt on the ability of feature visualizations to explain how standard neural networks process natural images.**

4. Theory: impossibility results for feature visualization

In the light of our experimental results, we ask: **When are feature visualizations guaranteed to be reliable?** (i.e., guaranteed to produce results that can be trusted / relied upon?) Feature visualization is expected to help us "*answer what the network detects*" (Olah et al., 2018), "*understand what a model is really looking for*" (Olah et al., 2017), and "*understand the nature of the functions learned by the network*" (Erhan et al., 2009). When formalizing these statements, two aspects need to be considered. First, the structure of functions to be visualized. The current literature does not place assumptions on the function: it could be any unit in a "*black-box*" (Heinrich et al., 2019; Nguyen et al., 2019) neural network. Second, we need to characterize which aspects of the function feature visualization promises to help understand. The existing literature (quotes above and in Appendix H) focuses on a strong, global notion of understanding, rather than on understanding a narrow set of inputs or a tiny part of the function. Here we prove that it is *impossible* to guarantee that feature visualization realizes this combination (global notion of understanding without assumptions about the black-box). Feature visualizations based on activation maximization generate highly activating images for a function f (e.g., a neuron in a fully connected or a channel in a convolutional layer). While not typically described as such[1], finding a highly activating image directly corresponds to finding the $\arg\max$ of f—an insight that might seem trivial. Paradoxically, it is well-known that it is impossible to conclude much, if anything, about an unconstrained function from its $\arg\max$. Yet, feature visualizations are purported to help us understand what black-box functions (e.g., neural network units) detect. To resolve this paradox, we can either lower our expectations by considering weaker notions of understanding, or impose more assumptions on the function. We now explore both directions, and show that **even strong assumptions on the function are *insufficient***

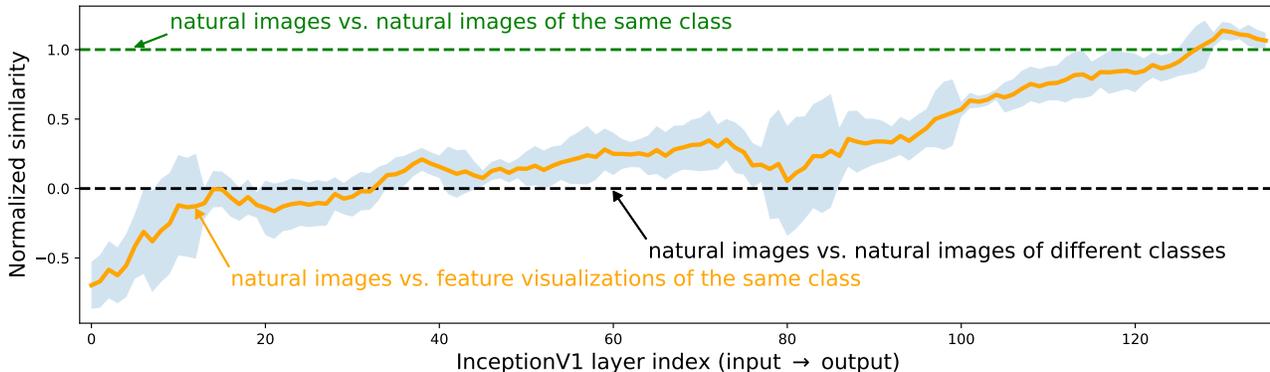[1] A notable exception is Zimmermann et al. (2021).

Figure 4: **Natural vs. synthetic path similarity analysis.** Throughout the first two thirds of Inception-V1 layers, activations of natural images have roughly as little similarity to same-class visualizations as they have to completely arbitrary images of different classes. In the last third of the network, similarity increases. Details in Section 3.

to guarantee that feature visualization are reliable for understanding f, even on significantly weaker notions of understanding. Our theoretical results are summarized in Table 1, where different columns correspond to subsequently weaker notions of understanding. The first column asks if feature visualization enables predicting a function value $f(x)$ for a new input x. The second column asks if $f(x)$ can be predicted up to some small error ε. The third column—which is similar to the task studied by Borowski et al. (2021)—is the weakest, asking only if feature visualization enables predicting if $f(x)$ is closer to the min or max value of f. Meanwhile, different rows correspond to asking "If the user knows that the function satisfies this assumption, can they use feature visualization to predict $f(x)$?"

Notation and definitions. We denote the indicator function of a Boolean expression E as $\mathbf{1}_E$, which is 1 if $E(x)$ is true and 0 otherwise. Let d denote the input dimensionality (e.g., number of pixels and channels), $\mathcal{I} = [0, 1]^d$ the input space, and $\mathcal{F} = \{\mathcal{I} \to [0, 1]\}$ the set of all functions from inputs to scalar, bounded activations[2]. *Maximally activating feature visualization* is the map from $\mathcal{F}$ to $\mathcal{I}^2 \times [0, 1]^2$ that returns a function's $\arg\min$, $\arg\max$, and values at these two points, which we denote by $\Phi_{\min\max}(f) = (\arg\min_{x \in \mathcal{I}} f(x), \arg\max_{x \in \mathcal{I}} f(x), \min_{x \in \mathcal{I}} f(x), \max_{x \in \mathcal{I}} f(x))$. When f is clear from context, we write $\Phi_{\min\max} = (x_{\min}, x_{\max}, f_{\min}, f_{\max})$ for brevity. We assess the reliability of a feature visualization by how well it can be used to predict $f(x)$ at new inputs x. To make such a prediction, the user must *decode* feature visualization into useful information. We denote a *feature visualization decoder* as a map $D \in \mathcal{D} = \{\mathcal{I}^2 \times [0, 1]^2 \to \mathcal{F}\}$. Our results do not rely on the structure of D in any way. Rather, No in Table 1 means that for *every* D the assumptions are

---

[2]For example, class probabilities or normalized activations of a bounded unit in a neural network.

insufficient to guarantee accurate prediction of f.

4.1. Main theoretical results

Precise definitions, all proofs, and more details are given in Appendix B. First, we note that the boundedness of f implies a trivial ability to predict $f(x)$.

Proposition 1. *There exists $D \in \mathcal{D}$ such that for all $f \in \mathcal{F}$,*

$$\left\| f - D(\Phi_{\min\max}(f)) \right\|_\infty \leq \frac{f_{\max} - f_{\min}}{2}. \qquad (3)$$

Proposition 1 says that for any function f, a user can take the feature visualization $\Phi_{\min\max}(f)$ and apply a specific decoder (the constant function taking value halfway between $f_{\min}$ and $f_{\max}$) to predict $f(x)$ for *any* new x. If the user imposed assumptions on f, one might conjecture that a clever choice of decoder could lead to a better prediction of $f(x)$. Our first main result shows that this is impossible even for strong assumptions.

Theorem 1. *For all $\mathcal{G} \in \{ \mathcal{F}, \mathcal{F}_{\mathrm{NN}}, \mathcal{F}_{\mathrm{ERM}}, \mathcal{F}_{\mathrm{PAff}}, \mathcal{F}_{\mathrm{Mono}} \}$, $D \in \mathcal{D}$, and $f \in \mathcal{G}$, there exists $f' \in \mathcal{G}$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$ and*

$$\left\| f' - D(\Phi_{\min\max}(f')) \right\|_\infty \geq \frac{f'_{\max} - f'_{\min}}{2}. \qquad (4)$$

Consider a user who knows that the unit to visualize is piecewise affine ($f \in \mathcal{F}_{\mathrm{PAff}}$). Using this knowledge, they hope to predict f by applying some decoder to the visualization $\Phi_{\min\max}(f)$. However, for every f, there is always another f' that satisfies the user's knowledge ($f' \in \mathcal{F}_{\mathrm{PAff}}$) and has $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$. Since the user is only given $\Phi_{\min\max}$, they cannot distinguish between the case when the true function is f and when it is f', regardless of how clever their decoder is. Theorem 1 says that f and f' are sufficiently different, and thus, the user will do poorly at predicting at least one of them; that is, the user does not improve on the uninformative predictive ability prescribed

			exactly?	ε-approxim.?	closer to min or max?

Given feature visualization for f, can we reliably predict $f(x)$ ...

			exactly?	ε-approxim.?	closer to min or max?
black-box		$\mathcal{F}$	No	No	No
neural network (NN)		$\mathcal{F}_{\mathrm{NN}}$	No	No	No
NN trained with ERM		$\mathcal{F}_{\mathrm{ERM}}$	No	No	No
$L-$Lipschitz (known L)		$\mathcal{F}_{\mathrm{Lip}}^{L}$	No	No	Only for small L
piecewise affine		$\mathcal{F}_{\mathrm{PAff}}$	No	No	No
monotonic		$\mathcal{F}_{\mathrm{Mono}}$	No	No	No
convex		$\mathcal{F}_{\mathrm{Convx}}$	No	No	No
affine (input dim. ¿ 1)		$\mathcal{F}_{\mathrm{Aff}}^{d>1}$	No	No	No
affine (input dim. = 1)		$\mathcal{F}_{\mathrm{Aff}}^{d=1}$	Yes	Yes	Yes
constant		$\mathcal{F}_{\mathrm{Const}}$	Yes	Yes	Yes

Stronger assumptions about f (left vertical axis label)

Table 1: **Theory overview.** Feature visualization aims to help understand a function f (e.g., a unit in a network). While understanding is an imprecise term, it can be formalized: Given f and its arg max and arg min (approximated by feature visualization), how well can we predict $f(x)$ for new values of x? Intuitively, this is impossible if f is a black-box. Instead, to make meaningful predictions, we need strong additional knowledge about f.

by Proposition 1. This implies No for the first two columns in Table 1. The same result can be shown for $\mathcal{F}_{\mathrm{Convx}}$ at the expense of $1/2$ becoming $1/4$ (Theorem 3) and for $\mathcal{F}_{\mathrm{Lip}}^{L}$ with dependence on L (Theorem 4).

Our second result is an analogous negative result for predicting whether $f(x)$ is closer to $f_{\max}$ or $f_{\min}$, implying No for the third column in Table 1. To state it, for any $f \in \mathcal{F}$ we define $m_f = (f_{\max} + f_{\min})/2$; note that $f(x)$ is closer to $f_{\max}$ if and only if $f(x) > m_f$.

Theorem 2. *For all $D \in \mathcal{D}$, $f \in \mathcal{G}$, and $\mathcal{G} \in \{\mathcal{F}, \mathcal{F}_{\mathrm{NN}}, \mathcal{F}_{\mathrm{ERM}}, \mathcal{F}_{\mathrm{PAff}}, \mathcal{F}_{\mathrm{Mono}}, \mathcal{F}_{\mathrm{Convx}}\}$, there exists $f' \in \mathcal{G}$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$ and*

$$\left\| \mathbf{1}_{f'>m_{f'}} - \mathbf{1}_{D(\Phi_{\min\max}(f'))>m_{f'}} \right\|_{\infty} \geq \mathbf{1}_{f_{\max} \neq f_{\min}}. \quad (5)$$

The LHS of Eq. (5) quantifies "Can the user tell if $f'(x)$ is closer to $f'_{\max}$ or $f'_{\min}$?" Since indicator functions are bounded in $[0, 1]$, the LHS is trivially bounded above by 1. Again consider the user who knows $f \in \mathcal{F}_{\mathrm{PAff}}$. Theorem 2 says that for any f—unless f also happens to be constant (i.e., $f_{\max} = f_{\min}$)—there is always some $f' \in \mathcal{F}_{\mathrm{PAff}}$ that is indistinguishable from f to the user *and* sufficiently different from f so that the user cannot reliably tell if $f'(x)$ is closer to $f'_{\max}$ or $f'_{\min}$ (i.e., the RHS is also 1). The same result can be shown for $\mathcal{F}_{\mathrm{Lip}}^{L}$ with dependence on L (Theorem 5). We defer our negative results for affine functions to Theorems 6 and 7 and our positive results to Theorem 8. Our theory proves that **without additional assumptions, it is impossible to guarantee that standard feature visualizations can be used to understand (i.e., predict) many types of functions, including black-boxes, neural networks, and even convex functions**.

5. Conclusion

Feature visualizations based on activation maximization are widely used to understand neural networks. We here asked whether feature visualizations are reliable, i.e. whether we can trust/rely on them. We investigated this question from three complementary angles. Section 2 shows that an adversary can manipulate visualizations. Section 3 shows that different processing paths cast doubt on the ability of feature visualizations to understand natural image processing. These empirical findings are underpinned by theoretical results (Section 4) proving that it is *impossible* to guarantee that standard feature visualizations can be used for understanding black-box functions—at least not if "understanding" means being able to make meaningful predictions about the functions. This indicates that while feature visualizations can be very useful for exploratory hypothesis generation, they should not be used for confirmatory use cases. In the absence of more reliable methods, combining them with additional methods (including dataset samples) as recommended by Olah et al. (2018; 2020) may be our best shot currently; although it may be worth noting that highly activating dataset samples can be manipulated too (Nanfack et al., 2023). On a higher level, these findings are part of a broader challenge to the concept of post-hoc interpretability methods: understanding completely black-box systems may sometimes be more than we can hope for (Srinivas & Fleuret, 2019; Bilodeau et al., 2022; Fokkema et al., 2022; Han et al., 2022). Instead, a potential way forward could be to use the theoretical framework explored here as a starting point to discover structures that enable reliable predictions. For instance, more linear units are easier to predict, an aspect we explore in Appendix G. Potentially, these structures could then be incorporated into "reliable-by-design" networks.

Code availability

Code to replicate experiments from this paper is available here: https://github.com/google-research/fooling-feature-visualizations/

Acknowledgments

Author contributions

The project was led and coordinated by RG. WB developed the core idea that the arg max does not constrain a function sufficiently, which can be exploited in order to manipulate feature visualizations (key insight behind Section 2 and Section 4). RG had the idea for Subsection 2.1; RG and RSZ conducted the experiments. RSZ and WB had the idea for Subsection 2.2; RSZ conducted the experiments. RG had the idea and ran the analysis for Section 3 based on discussions with BK. RG conceived of Lemma 1, and BB proved it with input from RG and RSZ. BB conceived of Lemma 2, and BB proved it with input from RG and RSZ. BB and RG conceived of the main results in Section 4. BB formalized and proved the results in Section 4 and the corresponding appendix. WB had the idea; RSZ ran the analysis for Appendix G based on discussions with WB and RG. BK, and WB at a later stage, provided overall guidance throughout the course of the project, helping with presentation and experiment details. The first draft was written by RG apart from Subsection 2.2 (RSZ), Section 4 (BB; intro jointly with RG) and Appendix G (RSZ and RG). BB curated the final presentation of theoretical results and plain-language descriptions with input from RG, RSZ and BK. All authors contributed to the final writing.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *Advances in Neural Information Processing Systems*, 31, 2018.

Adi, Y., Baum, C., Cisse, M., Pinkas, B., and Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Aïvodji, U., Arai, H., Fortineau, O., Gambs, S., Hara, S., and Tapp, A. Fairwashing: the risk of rationalization. In *International Conference on Machine Learning*, pp. 161–170. PMLR, 2019.

Anders, C., Pasliev, P., Dombrowski, A.-K., Müller, K.-R., and Kessel, P. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pp. 314–323. PMLR, 2020.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Banerjee, A., Guo, X., and Wang, H. On the optimality of conditional expectation as a Bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669, 2005.

Baniecki, H., Kretowicz, W., and Biecek, P. Fooling partial dependence via data poisoning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part III*, pp. 121–136. Springer, 2023.

Bashivan, P., Kar, K., and DiCarlo, J. J. Neural population control via deep image synthesis. *Science*, 364(6439): eaav9436, 2019.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6541–6549, 2017.

Bilodeau, B., Jaques, N., Koh, P. W., and Kim, B. Impossibility theorems for feature attribution. *arXiv preprint arXiv:2212.11870*, 2022.

Borowski, J., Zimmermann, R. S., Schepers, J., Geirhos, R., Wallis, T. S., Bethge, M., and Brendel, W. Exemplary natural images explain CNN activations better than state-of-the-art feature visualization. In *International Conference on Learning Representations*, 2021.

Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., Khlaaf, H., Yang, J., Toner, H., Fong, R., et al. Toward trustworthy AI development: mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*, 2020.

Chen, K.-L., Garudadri, H., and Rao, B. D. Improved bounds on neural complexity for representing piecewise

linear functions. In *Advances in Neural Information Processing Systems 36*, 2022.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

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

Fang, S. and Choromanska, A. Backdoor attacks on the DNN interpretation system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 561–570, 2022.

Fokkema, H., de Heide, R., and van Erven, T. Attribution-based explanations that provide recourse cannot be robust, 2022. arXiv:2205.15834.

Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Madry, A., Li, B., and Goldstein, T. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (2):1563–1580, 2022.

Greentfrapp. Lucent. https://github.com/greentfrapp/lucent, v0.1.8.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Gupta, A. and Arora, S. A simple saliency method that passes the sanity checks. *arXiv preprint arXiv:1905.12152*, 2019.

Han, T., Srinivas, S., and Lakkaraju, H. Which explanation should I choose? A function approximation perspective to characterizing post hoc explanations. In *Advances in Neural Information Processing Systems 36*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Heinrich, K., Zschech, P., Skouti, T., Griebenow, J., and Riechert, S. Demystifying the black box: A classification scheme for interpretation and visualization of deep intelligent systems. In *Twenty-fifth Americas Conference on Information Systems*, 2019.

Heo, J., Joo, S., and Moon, T. Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems*, 32, 2019.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Kriegeskorte, N. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 1:417–446, 2015.

Leavitt, M. L. and Morcos, A. Towards falsifiable interpretability research. *arXiv preprint arXiv:2010.12016*, 2020.

Lehky, S. R. and Sejnowski, T. J. Network model of shape-from-shading: neural function arises from both receptive and projective fields. *Nature*, 333(6172):452–454, 1988.

Mahendran, A. and Vedaldi, A. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120:233–255, 2016.

Mordvintsev, A., Olah, C., and Tyka, M. DeepDream-a code example for visualizing neural networks. *Google Research*, 2(5), 2015.

Nanfack, G., Fulleringer, A., Marty, J., Eickenberg, M., and Belilovsky, E. Adversarial attacks on the interpretation of neuron activation maximization. *arXiv preprint arXiv:2306.07397*, 2023.

Nguyen, A., Yosinski, J., and Clune, J. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

Nguyen, A., Yosinski, J., and Clune, J. Understanding neural networks via feature visualization: A survey. *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 55–76, 2019.

Nie, W., Zhang, Y., and Patel, A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pp. 3809–3818. PMLR, 2018.

Noppel, M., Peter, L., and Wressnegger, C. Backdooring explainable machine learning. *arXiv preprint arXiv:2204.09498*, 2022.

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

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. https://distill.pub/2018/building-blocks.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

OpenLetter, F. o. L. I. Pause giant AI experiments: An open letter, 2023. URL https://futureoflife.org/open-letter/pause-giant-ai-experiments/.

Ponce, C. R., Xiao, W., Schade, P. F., Hartmann, T. S., Kreiman, G., and Livingstone, M. S. Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences. *Cell*, 177(4): 999–1009, 2019.

Rao, S., Böhle, M., and Schiele, B. Towards better understanding attribution methods. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10223–10232, 2022.

Sejnowski, T. J. What are the projective fields of cortical neurons? *van Hemmen, JL, Sejnowski, TJ (Eds.)*, 23: 394–405, 2006.

Shahin Shamsabadi, A., Yaghini, M., Dullerud, N., Wyllie, S., Aïvodji, U., Alaagib, A., Gambs, S., and Papernot, N. Washing the unwashable: On the (im) possibility of fairwashing detection. *Advances in Neural Information Processing Systems*, 35:14170–14182, 2022.

Sharkey, L. Circumventing interpretability: How to defeat mind-readers. *arXiv preprint arXiv:2212.11415*, 2022.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 180–186, 2020.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Srinivas, S. and Fleuret, F. Full-gradient representation for neural network visualization. In *Advances in Neural Information Processing Systems 33*, 2019.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich,

A. Going deeper with convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Walker, E. Y., Sinz, F. H., Cobos, E., Muhammad, T., Froudarakis, E., Fahey, P. G., Ecker, A. S., Reimer, J., Pitkow, X., and Tolias, A. S. Inception loops discover what excites neurons most using deep predictive models. *Nature Neuroscience*, 22(12):2060–2065, 2019.

Wikipedia. Volkswagen emissions scandal. https://en.wikipedia.org/wiki/Volkswagen_emissions_scandal, 2015.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Zimmermann, R. S., Borowski, J., Geirhos, R., Bethge, M., Wallis, T., and Brendel, W. How well do feature visualizations support causal understanding of CNN activations? *Advances in Neural Information Processing Systems*, 34: 11730–11744, 2021.

Appendix

A. Further visualizations

original network units (block 4-1, conv 2 layer)

index 0   50   100   150   200   250   300   350   400   450

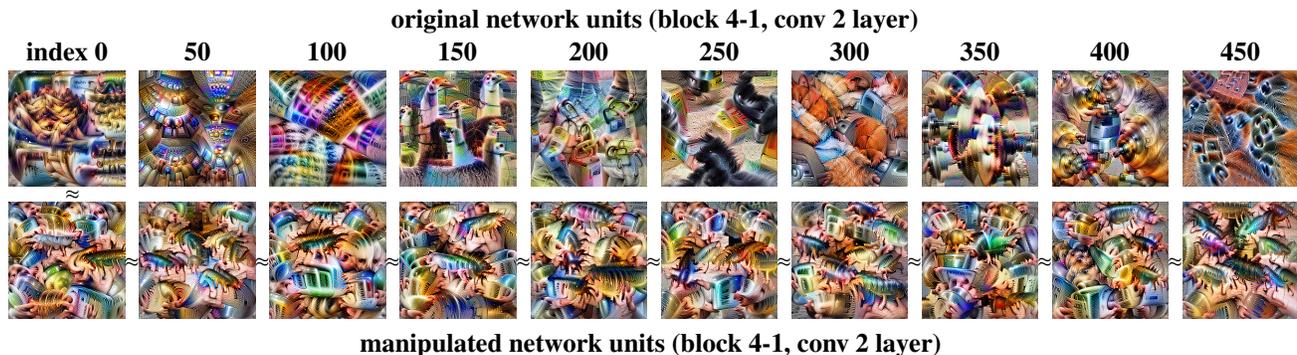**manipulated network units (block 4-1, conv 2 layer)**

Figure 5: **Leveraging silent units to produce identical visualizations throughout a layer.** Here, all intermediate units of a layer (block 4-1, conv 2) in a ResNet-50 network have near-identical visualizations. This is achieved by leveraging orthogonal filters in silent units, which are constructed such that the feature visualizations of all 512 units resemble that of the first unit, making the different units (almost) indistinguishable by their feature visualizations even though they perform distinct computations on natural input.

feature visualization trajectory   **natural ImageNet validation images**

Figure 6: **Natural vs. synthetic distribution shift.** There is a clear distribution shift between feature visualizations (left) and natural images (right). This can be exploited by a classifier when building a fooling circuit. Visualizations at different steps in the optimization process for a randomly selected unit in the last layer of standard, unmanipulated Inception-V1; randomly selected ImageNet validation samples (excluding images containing faces).

B. Proofs and theory details

B.1. Proof for fooling circuit (Section 2)

Lemma 1. *Let $k > 0$, $A : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ and $B, C : \mathbb{R}_+ \times \{0, 1\} \to \mathbb{R}_+$ with*

$$A(x, y) = x + y$$
$$B(x, z) = \max(0, x - kz)$$
$$C(x, z) = \max(0, x + kz - k)$$

be computations represented by a sub-graph of a neural network. Denote the combination of these computations as $N : \mathbb{R}_+ \times \mathbb{R}_+ \times \{0, 1\}$ with

$$N(x, y, z) = A(B(x, z), C(y, z)).$$

Then it holds that

$$\forall x, y \in \mathbb{R}_+ : k \geq \max(x, y) \implies \begin{cases} N(x, y, 0) = x \\ N(x, y, 1) = y \end{cases} \quad .$$

Proof of Lemma 1. First, consider the case where the binary input of N is 0; that is, $z = 0$. Then, since $k \geq y$,

$$B(x, 0) = \max(0, x) = x$$
$$C(y, 0) = \max(0, y - k) = 0, \tag{6}$$

so $N(x, y, 0) = A(B(x, 0), C(y, 0)) = A(x, 0) = x$.

Analogously consider $z = 1$. Then, since $k \geq x$,

$$B(x, 1) = \max(0, x - k) = 0$$
$$C(y, 1) = \max(0, y) = y, \tag{7}$$

so $N(x, y, 1) = A(B(x, 1), C(y, 1)) = A(0, y) = y$, completing the proof. $\square$

Lemma 2. *Let $\mathcal{X}$ denote the space of all possible inputs (e.g., all images), $\mathcal{D}$ some distribution on $\mathcal{X}$ (e.g., ImageNet). Let $D, F : \mathcal{X} \to \mathbb{R}_+$ represent the full computation of a unit in the original and in the tinkered network, respectively, that can be bounded on their domain. For an arbitrary algorithm $\mathrm{Opt} : \{\mathcal{X} \to \mathbb{R}\} \times \mathcal{X} \to \mathcal{X}$ and distribution π_0 on $\mathcal{X}$ define the following sequence of random variables: $\forall n > 0 : X_{n+1} = \mathrm{Opt}(D, X_n)$ and $X_0 \sim \pi_0$. Denote the distribution over $\mathcal{X}$ induced by this process π. If $\mathcal{D}$ and π have disjoint support, then there exists a neural network implementing a function $N : \mathcal{X} \to \mathbb{R}$ such that*

$$\mathbb{P}_{\mathbf{x} \sim \pi}[N(\mathbf{x}) = D(\mathbf{x})] = 1 \ \text{ and } \ \mathbb{P}_{\mathbf{x} \sim \mathcal{D}}[N(\mathbf{x}) = F(\mathbf{x})] = 1.$$

Proof of Lemma 2. As $\mathcal{D}$ and π have disjoint support this means that there exists a function $E : \mathbb{R} \to \{0, 1\}$ such that

$$\mathbb{P}_{\mathbf{x} \sim \pi}[E(\mathbf{x}) = 0] = 1 \ \text{ and } \ \mathbb{P}_{\mathbf{x} \sim \mathcal{D}}[E(\mathbf{x}) = 1] = 1. \tag{8}$$

Let $k = \max\left(\max_{\mathbf{x} \in \mathcal{D}} D(\mathbf{x}), \max_{\mathbf{x} \in \pi} D(\mathbf{x})\right)$, which exists as both D and F are bounded. In line with Lemma 1, we construct N as $N(\mathbf{x}) = A(B(D(\mathbf{x}), E(\mathbf{x})), C(F(\mathbf{x}), \neg E(\mathbf{x})))$. Per the universal approximation theorem (Hornik et al., 1989), there exists a neural network implementing the assumed function E. As all other computations (i.e., A, B, C, D, F) are implemented by a neural network, we can conclude that the constructed function N can also be implemented by a neural network.

Applying Lemma 1 and Eq. (8) directly yields

$$\mathbb{P}_{\mathbf{x} \sim \pi}[N(\mathbf{x}) = D(\mathbf{x})] = 1 \ \text{ and } \ \mathbb{P}_{\mathbf{x} \sim \mathcal{D}}[N(\mathbf{x}) = F(\mathbf{x})] = 1, \tag{9}$$

concluding the proof. $\square$

Remark 1. *In the case of feature visualizations, the assumption that π and $\mathcal{D}$ have disjoint support is plausible as demonstrated empirically in Subsection 2.1; this can also be visually appreciated from looking at Figure 6 showing a visualization trajectory which at no point resembles natural images.* $\triangleleft$

B.2. Details on interpretation of Table 1

First, we elaborate on what No and Yes mean in Table 1.

The weakest form of answering No would be to find a *single* function f where feature visualization cannot be used to predict f. At the other extreme, one could hope to show that feature visualization cannot be used to predict f *for all f*. Unfortunately, this is trivially impossible to show: for every distinct value the feature visualization can take, one could pick a function f that agrees with this visualization (e.g., has the same $\arg\max$) and use this as the prediction. In light of this, we prove the next strongest impossibility result. When the answer is No, we show that *for all*[3] functions f (except for a handful of corner cases, like constant functions), there exists another function f' that gets the exact same feature visualization as f yet cannot be accurately predicted. Similarly, for the cells with **extra assumptions in orange**, this means that the answer is No (as defined in the previous sentence) *unless these extra assumptions are satisfied.*

---

[3]Affine functions are the only exception, since there are more cases where an affine f can be exactly recovered from feature visualization. See Theorem 6 for the precise characterization.

We measure predictive accuracy using the sup norm for simplicity, but our results could be extended to any other strictly convex loss. This is essentially the strongest result one could hope for: by the intermediate value theorem, any continuous f must take every value in between $f_{\min}$ and $f_{\max}$, and hence it is impossible to prove that $f(x)$ can't be recovered *for every* x. On the contrary, for the cells where the answer is Yes, we can actually prove something much stronger than the converse of No: we prove that $f(x)$ can be predicted *for all* x and *for all* f. This hints at the necessity of such strong assumptions. Either the function class is so simple that feature visualization reveals everything about every function, or feature visualization reveals hardly anything about any function.

To find the precise results that correspond to each cell of Table 1, see Table 2.

		Given feature visualization for f, can we reliably predict $f(x)$ ...		
		exactly?	ε-approxim.?	closer to min or max?
black-box	$\mathcal{F}$	Theorem 1	Theorem 1	Theorem 2
neural network (NN)	$\mathcal{F}_{\mathrm{NN}}$	Theorem 1	Theorem 1	Theorem 2
NN trained with ERM	$\mathcal{F}_{\mathrm{ERM}}$	Theorem 1	Theorem 1	Theorem 2
$L-$Lipschitz (known L)	$\mathcal{F}_{\mathrm{Lip}}^L$	Theorem 4	Theorem 4	Theorem 5
piecewise affine	$\mathcal{F}_{\mathrm{PAff}}$	Theorem 1	Theorem 1	Theorem 2
monotonic	$\mathcal{F}_{\mathrm{Mono}}$	Theorem 1	Theorem 1	Theorem 2
convex	$\mathcal{F}_{\mathrm{Convx}}$	Theorem 3	Theorem 3	Theorem 2
affine (input dim. ¿ 1)	$\mathcal{F}_{\mathrm{Aff}}^{d>1}$	Theorem 6	Theorem 6	Theorem 7
affine (input dim. = 1)	$\mathcal{F}_{\mathrm{Aff}}^{d=1}$	Theorem 8	Theorem 8	Theorem 8
constant	$\mathcal{F}_{\mathrm{Const}}$	Theorem 8	Theorem 8	Theorem 8

Table 2: Theoretical results corresponding to each cell of Table 1.

Finally, we define precisely what each assumption means in Table 1. For any space $\mathcal{A}$, let $\mathcal{M}(\mathcal{A})$ denote the set of all probability measures on $\mathcal{A}$.

Neural Network: $\quad \mathcal{F}_{\mathrm{NN}} \quad = \left\{ f \in \mathcal{F} : \text{can be written as mat. mul. with scalar activations} \right\}$

NN with ERM: $\quad \mathcal{F}_{\mathrm{ERM}} \quad = \Big\{ f \in \mathcal{F}_{\mathrm{NN}} : \exists \pi \in \mathcal{M}(\mathcal{I} \times [0,1]) \text{ s.t.}$

$$f = \arg\min_{f' \in \mathcal{F}} \mathbb{E}_{(X,Y) \sim \pi} \ell(f'(X), Y), \text{ where } \ell \text{ is a Bregman loss function (see Banerjee et al., 2005)} \Big\}$$

Lipschitz: $\quad \mathcal{F}_{\mathrm{Lip}}^L \quad = \left\{ f \in \mathcal{F} : \sup_{x,x' \in \mathcal{I}} \frac{f(x) - f(x')}{\|x - x'\|_\infty} \leq L \right\}$

Piecewise Affine: $\quad \mathcal{F}_{\mathrm{PAff}} \quad = \left\{ f \in \mathcal{F} : \text{can be written as affine on each piece of a partition of } \mathcal{I} \right\}$

Monotone: $\quad \mathcal{F}_{\mathrm{Mono}} \quad = \left\{ f \in \mathcal{F} : \forall x \leq x', f(x) \leq f(x') \right\} \cup \left\{ f : \forall x \leq x', f(x) \geq f(x') \right\}$ [4]

Convex: $\quad \mathcal{F}_{\mathrm{Convx}} \quad = \Big\{ f \in \mathcal{F} : \forall x, x' \in \mathcal{I} \, \forall \alpha \in [0,1],$

$$f(\alpha x + (1 - \alpha)x') \leq \alpha f(x) + (1 - \alpha)f(x') \Big\}$$

Affine: $\quad \mathcal{F}_{\mathrm{Aff}}^d \quad = \left\{ f \in \mathcal{F} : \exists A \in \mathbb{R}^d \, \exists b \in \mathbb{R} \text{ s.t. } \forall x \in \mathcal{I}, f(x) = A^{\mathrm{T}}x + b \right\}$

Constant: $\quad \mathcal{F}_{\mathrm{Const}} \quad = \left\{ f \in \mathcal{F} : \forall x, x' \in \mathcal{I}, f(x) = f(x') \right\}.$

---

[4]For d-dimensional inputs, $x \leq x'$ if and only if $x_j \leq x'_j$ for all $j \in [d]$.

B.3. Additional impossibility results

While Theorems 1 and 2 already provide impossibility for strong assumptions like monotonicity and piecewise affine, convexity is a particularly strong assumption since it restricts the output space of functions. In particular, no convex function can cross the diagonal line from $f_{\min}$ to $f_{\max}$, and hence it is possible that more information can be recovered from just these values. However, the next result shows that this information can only be used to possibly improve a constant $1/2$ to $1/4$, and arbitrary approximation is still impossible (unless the function is constant).

Theorem 3. *For all $D \in \mathcal{D}$ and $f \in \mathcal{F}_{\mathrm{Convx}}$, there exists $f' \in \mathcal{F}_{\mathrm{Convx}}$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$ and*

$$\left\| f' - D(\Phi_{\min\max}(f')) \right\|_\infty \geq \frac{f'_{\max} - f'_{\min}}{4}.$$

Similarly, a known Lipschitz constant implies local stability of f, which may be possible for a decoder to exploit. However, we show that this is also not possible in general (our result is stated in 1-dimension for simplicity, but could be extended trivially to arbitrary dimension using the sup norm definition of Lipschitz).

Theorem 4. *For all $D \in \mathcal{D}$, $L > 0$, and $f \in \mathcal{F}_{\mathrm{Lip}}^L$, there exists $f' \in \mathcal{F}_{\mathrm{Lip}}^L$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$ and if $2|f_{\max} - f_{\min}| \leq L|x_{\max} - x_{\min}|$ then*

$$\left\| f' - D(\Phi_{\min\max}(f')) \right\|_\infty \geq \frac{f'_{\max} - f'_{\min}}{2}.$$

Moreover, even if $2|f_{\max} - f_{\min}| > L|x_{\max} - x_{\min}|$,

$$\left\| f' - D(\Phi_{\min\max}(f')) \right\|_\infty \geq L \max \left\{ \min\{x_{\min}, x_{\max}\}, 1 - \max\{x_{\min}, x_{\max}\} \right\}.$$

First, for all $f \in \mathcal{F}_{\mathrm{Lip}}^L$ it holds that $|f_{\max} - f_{\min}| \leq L|x_{\max} - x_{\min}|$, so the first condition nearly captures all cases. As already argued, under this condition our lower bound is tight by Proposition 1. Even when the condition fails, our lower bound is zero if and only if $|x_{\max} - x_{\min}| = 1$ and $|f_{\max} - f_{\min}| > L/2$. This is nearly tight, since if $|f_{\max} - f_{\min}| = L$, then necessarily $|x_{\max} - x_{\min}| = 1$ and f is linear and uniquely identifiable from $\Phi_{\min\max}(f)$ (and hence the lower bound must be zero in this case).

A similar condition can be used to provide a Lipschitz analogue of Theorem 2.

Theorem 5. *For all $D \in \mathcal{D}$, $L > 0$, and $f \in \mathcal{F}_{\mathrm{Lip}}^L$ such that $2|f_{\max} - f_{\min}| \leq L|x_{\max} - x_{\min}|$, there exists $f' \in \mathcal{F}_{\mathrm{Lip}}^L$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$ and*

$$\left\| \mathbf{1}_{f' > m_{f'}} - \mathbf{1}_{D(\Phi_{\min\max}(f')) > m_{f'}} \right\|_\infty \geq \mathbf{1}_{\sup_{x \in \mathcal{I}} f(x) \neq \inf_{x \in \mathcal{I}} f(x)}.$$

Finally, we have the following negative result for affine functions. Due to the extra structure imposed by an affine assumption, in more cases it is possible to fully recover f from just the feature visualization. However, in the worst case, f may still be completely unrecoverable. We show this for $d = 2$; a similar result can be shown in higher dimensions with more careful accounting of edge cases.

Theorem 6. *For all $D \in \mathcal{D}$ and $f \in \mathcal{F}_{\mathrm{Aff}}^{d=2}$, there exists $f' \in \mathcal{F}_{\mathrm{Aff}}^{d=2}$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$ and*

$$\left\| f' - D(\Phi_{\min\max}(f')) \right\|_\infty \geq \mathbf{1}_{x_{\min,1} \neq x_{\min,2}} \mathbf{1}_{x_{\max,1} \neq x_{\max,2}} \frac{f'_{\max} - f'_{\min}}{2}.$$

The same can also be shown for the analogue of Theorem 2.

Theorem 7. *For all $D \in \mathcal{D}$ and $f \in \mathcal{F}_{\mathrm{Aff}}^{d=2}$, there exists $f' \in \mathcal{F}_{\mathrm{Aff}}^{d=2}$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f')$ and*

$$\left\| \mathbf{1}_{f' > m_{f'}} - \mathbf{1}_{D(\Phi_{\min\max}(f')) > m_{f'}} \right\|_\infty \geq \mathbf{1}_{x_{\min,1} \neq x_{\min,2}} \mathbf{1}_{x_{\max,1} \neq x_{\max,2}}.$$

B.4. Positive results

Finally, we state our positive result for very simple functions.

Theorem 8. *For all $\mathcal{G} \in \{\mathcal{F}_{\text{Aff}}^{d=1}, \mathcal{F}_{\text{Const}}\}$ there exists $D \in \mathcal{D}$ such that for all $f \in \mathcal{G}$,*

$$\left\| f - D(\Phi_{\min\max}(f)) \right\|_\infty = 0.$$

B.5. Proofs for Section 4

Remark 2. *Throughout, we prove impossibility results for 1-dimensional functions. The extension to multiple dimensions follows from using our constructions componentwise and then applying Lemma 3 or Lemma 4 as appropriate, which hold for any input dimension.* ◁

B.5.1. HELPER LEMMAS

To prove results for the first two columns on Table 1, we use the following lemma to characterize the performance of an arbitrary decoder $D \in \mathcal{D}$.

Lemma 3. *For any $D \in \mathcal{D}$ and $f_1, f_2 \in \mathcal{F}$ such that $\Phi_{\min\max}(f_1) = \Phi_{\min\max}(f_2)$, for some $f \in \{f_1, f_2\}$*

$$\left\| f - D(\Phi_{\min\max}(f)) \right\|_\infty \geq \frac{\|f_1 - f_2\|_\infty}{2}.$$

Proof of Lemma 3. Let $g = D(\Phi_{\min\max}(f_1)) = D(\Phi_{\min\max}(f_2))$ and let x be such that $|f_1(x) - f_2(x)| = \|f_1 - f_2\|_\infty$. Then, since mean is less than $\max$,

$$\frac{1}{2}|f_1(x) - f_2(x)| \leq \frac{1}{2}|f_1(x) - g(x)| + \frac{1}{2}|g(x) - f_2(x)| \leq \max_{f \in \{f_1, f_2\}} |f(x) - g(x)|.$$

$\square$

Then, for any $\mathcal{G} \subseteq \mathcal{F}$ of interest and any $f \in \mathcal{G}$, we simply have to find $f_1, f_2 \in \mathcal{G}$ such that $\Phi_{\min\max}(f) = \Phi_{\min\max}(f_1) = \Phi_{\min\max}(f_2)$ and $\|f_1 - f_2\|_\infty$ is appropriately large (where "large" will depend on f).

Similarly, we use the following lemma to prove results for the third column of Table 1.

Lemma 4. *For any $D \in \mathcal{D}$ and $f_1, f_2 \in \mathcal{F}$ such that $\Phi_{\min\max}(f_1) = \Phi_{\min\max}(f_2)$, for some $f \in \{f_1, f_2\}$*

$$\left\| \mathbf{1}_{f>m_f} - \mathbf{1}_{D(\Phi_{\min\max}(f))>m_f} \right\|_\infty \geq \|\mathbf{1}_{f_1>m} - \mathbf{1}_{f_2>m}\|_\infty,$$

where $m = m_{f_1} = m_{f_2}$.

Proof of Lemma 4. Let $g = D(\Phi_{\min\max}(f_1)) = D(\Phi_{\min\max}(f_2))$ and let x be such that $\left| \mathbf{1}_{f_1(x)>m} - \mathbf{1}_{f_2(x)>m} \right| = \|\mathbf{1}_{f_1>m} - \mathbf{1}_{f_2>m}\|_\infty$.

If $\|\mathbf{1}_{f_1>m} - \mathbf{1}_{f_2>m}\|_\infty = 0$ the result holds trivially, so suppose that $\|\mathbf{1}_{f_1>m} - \mathbf{1}_{f_2>m}\|_\infty = 1$. That is, $\mathbf{1}_{f_1(x)>m} \neq \mathbf{1}_{f_2(x)>m}$. If $\mathbf{1}_{g(x)>m} = \mathbf{1}_{f_1(x)>m}$, then

$$\left\| \mathbf{1}_{f_2>m} - \mathbf{1}_{D(\Phi_{\min\max}(f_2))>m} \right\|_\infty \geq \left| \mathbf{1}_{f_2(x)>m} - \mathbf{1}_{g(x)>m} \right| = 1.$$

Otherwise, if $\mathbf{1}_{g(x)>m} = \mathbf{1}_{f_2(x)>m}$, then

$$\left\| \mathbf{1}_{f_1>m} - \mathbf{1}_{D(\Phi_{\min\max}(f_1))>m} \right\|_\infty \geq \left| \mathbf{1}_{f_1(x)>m} - \mathbf{1}_{g(x)>m} \right| = 1.$$

That is,

$$\max_{f \in f_1, f_2} \left\| \mathbf{1}_{f>m} - \mathbf{1}_{D(\Phi_{\min\max}(f))>m} \right\|_\infty \geq 1 = \|\mathbf{1}_{f_1>m} - \mathbf{1}_{f_2>m}\|_\infty.$$

$\square$

B.5.2. PROOF OF PROPOSITION 1

Let $D(x_{\min}, x_{\max}, f_{\min}, f_{\max}) \equiv (1/2)(f_{\max} + f_{\min})$ and fix $f \in \mathcal{F}$. For any x,

$$f(x) - \frac{f_{\max} + f_{\min}}{2} \leq f_{\max} - \frac{f_{\max} + f_{\min}}{2} = \frac{f_{\max} - f_{\min}}{2}$$

and

$$\frac{f_{\max} + f_{\min}}{2} - f(x) \leq \frac{f_{\max} + f_{\min}}{2} - f_{\min} = \frac{f_{\max} - f_{\min}}{2}.$$

Thus,

$$\left| f(x) - \frac{f_{\max} + f_{\min}}{2} \right| \leq \frac{f_{\max} - f_{\min}}{2}.$$

Since x was arbitrary, the result holds. $\qquad\square$

B.5.3. PROOF OF THEOREM 1

First, suppose $0 \leq x_{\min} < x_{\max} \leq 1$.

Define

$$f_1(x) = \begin{cases} f_{\min} & x \in [0, (x_{\min} + x_{\max})/2] \\ \frac{2(f_{\max} - f_{\min})}{x_{\max} - x_{\min}} x + \frac{2 f_{\min} x_{\max} - f_{\max} x_{\min} - f_{\max} x_{\max}}{x_{\max} - x_{\min}} & x \in [(x_{\min} + x_{\max})/2, x_{\max}] \\ f_{\max} & x \in [x_{\max}, 1] \end{cases}$$

and

$$f_2(x) = \begin{cases} f_{\min} & x \in [0, x_{\min}] \\ \frac{2(f_{\max} - f_{\min})}{x_{\max} - x_{\min}} x + \frac{f_{\min} x_{\max} + f_{\min} x_{\min} - 2 f_{\max} x_{\min}}{x_{\max} - x_{\min}} & x \in [x_{\min}, (x_{\min} + x_{\max})/2] \\ f_{\max} & x \in [(x_{\min} + x_{\max})/2, 1]. \end{cases}$$

Since $\Phi_{\min\max}(f_1) = \Phi_{\min\max}(f_2) = \Phi_{\min\max}(f)$, and f_1 and f_2 are both monotone and piecewise affine, the result follows for $\mathcal{F}_{\text{Mono}}$ and $\mathcal{F}_{\text{PAff}}$ from applying Lemma 3 and observing that $\|f_1 - f_2\|_\infty \geq f_{\max} - f_{\min}$ (this occurs at $(x_{\min} + x_{\max})/2$).

If $0 \leq x_{\max} < x_{\min} \leq 1$, the same argument applies with $1 - f_1$ and $1 - f_2$.

Finally, when $x_{\min} = x_{\max}$, then $f_{\max} - f_{\min} = 0$ so the result holds trivially.

To prove the result for $\mathcal{F}_{\text{NN}}$, note that we imposed no constraints on $f_{\max}$ or $f_{\min}$. Thus, for any $f \in \mathcal{F}_{\text{NN}}$, we can construct f_1 and f_2. We then use that any piecewise affine function can be exactly represented by a sufficiently large neural network (Arora et al., 2018; Chen et al., 2022) to conclude $f_1, f_2 \in \mathcal{F}_{\text{NN}}$.

The same argument applies to prove the result for $\mathcal{F}$, since clearly $f_1, f_2 \in \mathcal{F}$.

Finally, for $\mathcal{F}_{\text{ERM}}$, we must construct appropriate distributions with conditional means f_1 and f_2 respectively (we already noted these are both elements of $\mathcal{F}_{\text{NN}}$). For simplicity, define the joint distribution by $X \sim \text{Unif}(\mathcal{I})$ and $Y|X \sim \text{Ber}(f_j(X))$ for $j \in \{1, 2\}$. $\qquad\square$

B.5.4. PROOF OF THEOREM 2

We use f_1 and f_2 from Theorem 1, and recall that $m = (f_{\min} + f_{\max})/2$. Consider when $0 \leq x_{\min} < x_{\max} \leq 1$. Then, at $x = (x_{\min} + x_{\max})/2$, $f_1(x) = f_{\min} < m$ and $f_2(x) = f_{\max} > m$, so $\|\mathbf{1}_{f_1 > m} - \mathbf{1}_{f_2 > m}\|_\infty = 1$. The result then follows by Lemma 4. If $0 \leq x_{\max} < x_{\min} \leq 1$, the same argument applies with $1 - f_1$ and $1 - f_2$.

For $\mathcal{F}_{\text{Convx}}$, we use f_1 and f_2 from the proof of Theorem 3 (Appendix B.5.5). Recall that $m = (f_{\min} + f_{\max})/2$. Consider when $x_{\max} = 1$ and $x_{\min} < 1$. Then, at $x = x_{\min}/4 + 3/4$, $f_1(x) = f_{\min}/4 + 3 f_{\max}/4 > m$ and $f_2(x) = m$, so

$\|\mathbf{1}_{f_1 > m} - \mathbf{1}_{f_2 > m}\|_\infty = 1$. The result then follows by Lemma 4. If $x_{\min} = 0$ and $x_{\max} > 0$, the same argument applies using f_1' and f_2' as defined in Appendix B.5.5.

If $x_{\min} = x_{\max}$ then $f_{\min} = f_{\max}$ and hence f is constant, so the result holds trivially. $\qquad\square$

B.5.5. PROOF OF THEOREM 3

Note that for any $f \in \mathcal{F}_{\text{Convx}}$, one of $x_{\min}$ or $x_{\max}$ are in $\{0, 1\}$.

First, consider $x_{\max} = 1$ and $x_{\min} < 1$. Define

$$
f_1(x) = \begin{cases} f_{\min} & x \in [0, x_{\min}] \\ \frac{f_{\max} - f_{\min}}{1 - x_{\min}} x + \frac{f_{\min} - f_{\max} x_{\min}}{1 - x_{\min}} & x \in [x_{\min}, 1] \end{cases}
\tag{10}
$$

and

$$
f_2(x) = \begin{cases} f_{\min} & x \in [0, (x_{\min} + 1)/2] \\ \frac{2(f_{\max} - f_{\min})}{1 - x_{\min}} x + \frac{2 f_{\min} - f_{\max} x_{\min} - f_{\max}}{1 - x_{\min}} & x \in [(x_{\min} + 1)/2, 1]. \end{cases}
\tag{11}
$$

Clearly, $f_1, f_2 \in \mathcal{F}_{\text{Convx}}$ (since they are flat then linear with positive slope) and $\|f_1 - f_2\|_\infty \geq (f_{\max} - f_{\min})/2$ (this occurs at $x = (x_{\min} + 1)/2$). Since $x_{\min} = 0$ implies that $x_{\max} = 1$ by convexity, this case also covers $x_{\min} = 0$.

Second, consider $x_{\max} = 0$ and $x_{\min} > 0$. Using Eqs. (10) and (11), define $g_1(x) = f_1(1 - x)$ and $g_2(x) = f_2(1 - x)$. These also satisfy $g_1, g_2 \in \mathcal{F}_{\text{Convx}}$ (since they are linear with negative slope then flat) and $\|g_1 - g_2\|_\infty \geq (f_{\max} - f_{\min})/2$ (this occurs at $x = x_{\min}/2$). Since $x_{\min} = 1$ implies that $x_{\max} = 0$ by convexity, this case also covers $x_{\min} = 1$.

Finally, if $x_{\min} = x_{\max}$ then $f_{\min} = f_{\max}$ and the result holds trivially. $\qquad\square$

B.5.6. PROOF OF THEOREM 4

When $2|f_{\max} - f_{\min}| \leq L|x_{\max} - x_{\min}|$, the proof of Theorem 1 applies since $f_1, f_2 \in \mathcal{F}_{\text{Lip}}^L$.

Otherwise, suppose that $0 \leq x_{\min} < x_{\max} \leq 1$. Define

$$
f_1(x) = \begin{cases} f_{\min} & x \in [0, x_{\min}] \\ \frac{f_{\max} - f_{\min}}{x_{\max} - x_{\min}} x + \frac{f_{\min} x_{\max} - f_{\max} x_{\min}}{x_{\max} - x_{\min}} & x \in [x_{\min}, x_{\max}] \\ f_{\max} & x \in [x_{\max}, 1] \end{cases}
$$

and

$$
f_2(x) = \begin{cases} -L(x - x_{\min}) + f_{\min} & x \in [0, x_{\min}] \\ \frac{f_{\max} - f_{\min}}{x_{\max} - x_{\min}} x + \frac{f_{\min} x_{\max} - f_{\max} x_{\min}}{x_{\max} - x_{\min}} & x \in [x_{\min}, x_{\max}] \\ -L(x - x_{\max}) + f_{\max} & x \in [x_{\max}, 1]. \end{cases}
$$

Recall that by definition of $f \in \mathcal{F}_{\text{Lip}}^L$, $|f_{\max} - f_{\min}| \leq L|x_{\max} - x_{\min}|$, so $f_1, f_2 \in \mathcal{F}_{\text{Lip}}^L$.

If $x_{\min} > 1 - x_{\max}$, then $\|f_1 - f_2\|_\infty \geq L x_{\min}$ (which is realized at $x = 0$), and otherwise $\|f_1 - f_2\|_\infty \geq L(1 - x_{\max})$ (which is realized at $x = 1$).

If $0 \leq x_{\max} < x_{\min} \leq 1$, the same argument applies with $1 - f_1$ and $1 - f_2$. $\qquad\square$

B.5.7. PROOF OF THEOREM 5

Since $2|f_{\max} - f_{\min}| \leq L|x_{\max} - x_{\min}|$, the proof of Theorem 2 applies because $f_1, f_2 \in \mathcal{F}_{\text{Lip}}^L$. $\qquad\square$

B.5.8. PROOF OF THEOREM 6

When $d = 2$, any $f \in \mathcal{F}_{\text{Aff}}^{d=2}$ satisfies $f(x) = ax_1 + bx_2 + c$ for some $a, b, c \in \mathbb{R}$. Given $\Phi_{\min \max}(f) = (x_{\min}, f_{\min}, x_{\max}, f_{\max})$, the compatible $f \in \mathcal{F}_{\text{Aff}}^{d=2}$ are those f_c such that

$$a = \frac{x_{\max,2} f_{\min} - x_{\min,2} f_{\max} + (x_{\min,2} - x_{\max,2})c}{x_{\min,1} x_{\max,2} - x_{\max,1} x_{\min,2}}$$

$$b = \frac{-x_{\max,1} f_{\min} + x_{\min,1} f_{\max} + (x_{\max,1} - x_{\min,1})c}{x_{\min,1} x_{\max,2} - x_{\max,1} x_{\min,2}},$$

where c is a free parameter (with the only constraint that $f_c(x) \in [0, 1]$ for all $x \in \mathcal{I}$).

Since f is affine, $x_{\min}$ and $x_{\max}$ must both occur at one of the four corners of $[0, 1]^2$. Note that a and b above are undefined for some of these combinations, which we now enumerate.

If $x_{\min} = x_{\max}$, then f is constant and can be recovered exactly. If $x_{\min} = (0, 0)$, then necessarily $c = f_{\min}$, so f can be recovered exactly. Similarly, if $x_{\max} = (0, 0)$ then necessarily $c = f_{\max}$.

Moreover, there are other cases where f can be recovered. If $x_{\min} = (1, 1)$ and $x_{\max} \in \{(1, 0), (0, 1)\}$, then one of a or b do not depend on c and hence c can be directly recovered. The same is true when $x_{\max} = (1, 1)$.

There are two possibilities left.

1) $x_{\min} = (0, 1)$ and $x_{\max} = (1, 0)$:

$$a = f_{\max} - c$$
$$b = f_{\min} - c.$$

Take $c_1 = f_{\min}$ and $c_2 = f_{\max}$ to get

$$f_1(x) = (f_{\max} - f_{\min})x_1 + f_{\min}$$

and

$$f_1(x) = (f_{\min} - f_{\max})x_2 + f_{\max}.$$

These still have $\Phi_{\min \max}(f) = \Phi_{\min \max}(f_1) = \Phi_{\min \max}(f_2)$, but $\|f_1 - f_2\|_\infty \geq f_{\max} - f_{\min}$ (this is realized at $x = (1, 1)$).

2) $x_{\min} = (1, 0)$ and $x_{\max} = (0, 1)$:

$$a = f_{\min} - c$$
$$b = f_{\max} - c.$$

Take $c_1 = f_{\min}$ and $c_2 = f_{\max}$ to get

$$f_1(x) = (f_{\max} - f_{\min})x_2 + f_{\min}$$

and

$$f_1(x) = (f_{\min} - f_{\max})x_1 + f_{\max}.$$

These still have $\Phi_{\min \max}(f) = \Phi_{\min \max}(f_1) = \Phi_{\min \max}(f_2)$, but $\|f_1 - f_2\|_\infty \geq f_{\max} - f_{\min}$ (this is again realized at $x = (1, 1)$). The result holds by then applying Lemma 3. $\square$

B.5.9. PROOF OF THEOREM 7

This follows directly from applying Lemma 4 to the functions constructed in the proof of Theorem 6 (Appendix B.5.8). $\square$

B.5.10. PROOF OF THEOREM 8

First suppose that $f \in \mathcal{F}_{\text{Aff}}^{d=1}$. That is, there exists a, b such that $f(x) = ax + b$ for all x. Given $\Phi_{\min \max}(f)$, define

$$a_f = \frac{f_{\max} - f_{\min}}{x_{\max} - x_{\min}}$$

and

$$b_f = \frac{x_{\max} f_{\min} - x_{\min} f_{\max}}{x_{\max} - x_{\min}}.$$

Set $D(\Phi_{\min \max}(f)) = [x \mapsto a_f x + b_f]$. Since there is a unique affine function passing through both $(x_{\min}, f_{\min})$ and $(x_{\max}, f_{\max})$, and $D(\Phi_{\min \max}(f))$ is an affine function passing through both of these, this implies that $D(\Phi_{\min \max}(f)) \equiv f$.

If $f \in \mathcal{F}_{\mathrm{Const}}$, then there exists $y \in [0, 1]$ such that $f_{\min} = f_{\max} = y$ and $f(x) = y$ for all x. Define $D(\Phi_{\min \max}(f)) \equiv f_{\min}$, which implies that $D(\Phi_{\min \max}(f)) \equiv f$. $\qquad\square$

C. Method details for classifier training (Subsection 2.1)

For classifier training, we create a dataset by combining $1,281,167$ images from the training set of the ImageNet 2012 dataset and $472,500$ synthetic images. These synthetic images are the (intermediate) results of the feature visualization optimization process. Specifically, for $1,000$ classification units in the last layer of an ImageNet-trained InceptionV1 network, we run the optimization process with the parameters used by (Olah et al., 2017) 35 times each, resulting in $35,000$ unique optimization trajectories. We logarithmically sample 15 (intermediate) steps from the optimization trajectory, resulting in $525,000$ synthetic images in total. Finally, we split the synthetic images into $472,500$ ($= 90\%$) training and $52,500$ ($= 10\%$) testing images. Note that we use different units for the two sets. We train a model implementing the simple six layer CNN architecture displayed in Table 3 for 8 epochs on the aforementioned dataset with an SGD optimizer using a learning rate of 0.01, momentum of 0.9 and weight decay of 0.00005. The classifier achieves a near-perfect accuracy of 99.49% on the held-out test set (99.66% and 99.31% for natural input and feature visualizations, respectively).

Type	Size/Channels	Activation	Stride
Conv 3×3	16	ReLU	3
Conv 5×5	16	ReLU	2
Conv 5×5	16	ReLU	2
Conv 5×5	16	ReLU	2
Conv 5×5	16	ReLU	2
Conv 3×3	16	ReLU	2
Flatten	-	-	-
Linear	1	-	-

Table 3: Architecture of the classifier used to detect feature visualizations.

D. Method details for feature visualization figures

Throughout the paper, feature visualizations were generated using the `lucent` library (Greentfrapp, v0.1.8), version v0.1.8. Per default, we used `thresholds=(512,)` except for Figure 6 where the five images at different points in the optimization trajectory are shown (specifically, `thresholds=(1,8,32,128,512)`). For Figure 1, we used the thresholds that visually looked best (in line with existing literature: there is no principled way to determine the threshold); specifically `thresholds=(512,512,512,6,32,6)` for the six visualizations from left to right (for the three rightmost images, higher thresholds produced qualitatively similar yet oversaturated images). In terms of transformations during feature visualization, `transforms=lucent.optvis.transform.standard_transforms + [center_crop(224, 224)]` was used. The image was parameterized via `param_f=lambda: lucent.optvis.param.image(224, batch=1)`.

For Figure 1, a natural image was embedded into the weights of a single convolutional layer, `torch.nn.Conv2d`, with `kernel_size=224, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')`. To this end, the image was loaded and the layer weights were set to the corresponding image values, divided by 224^2 to avoid a potential overflow. Architecturally, the layer received the standard image

input and its ouptut was appended to the output of the desired layer (e.g. `softmax2_pre_activation_matmul`) where it was used in the role of D from Figure 3.

For Figure 5, we exploit the fact that feature visualizations usually lead to substantially larger activations than natural samples. We can exploit this property without requiring an explicit classifier like the fooling circuit did. We choose the two free parameters $\bar{\Theta}$ and b such that $\Delta y = 0$ for natural inputs but $\Delta y \neq 0$ for feature visualizations. Specifically, we construct $\bar{\Theta}$ by adding a sufficiently strong orthogonal perturbation $\Delta\Theta^{\perp}$ to Θ; that is, $\bar{\Theta} = \alpha\Theta + \beta\Delta\Theta^{\perp}$, where α, β control the relative strength of the filter directions. By choosing a sufficiently negative bias b, we ensure that Δy remains silent (no activation, ensuring that natural input is processed as before) unless y is very strongly activated. Letting $\bar{y}_{\text{max. nat}}$ denote the maximal observed activation on natural input for the constructed filter $\bar{\Theta}$, we set $b = -\alpha/\beta\bar{y}_{\text{max. nat}}$. Since we empirically observe a large gap between activations reached by feature visualizations and natural input, we are able to steer the visualizations to (almost) arbitrarily chosen images, as demonstrated in Figure 5.

E. Method details and additional similarity plots for Section 3

Motivation: relationship between path similarity and Spearman correlation. In Section 3, we describe that different processing paths lead to different activation similarity as measured through Spearman correlation. We here attempt to explain this relationship in a bit more detail. For the context of our analysis, we define a path as a (sub-)graph of a directed acyclic graph (DAG, a neural network or sub-network in our case), starting at the input nodes (first layer units) and ending at a single unit (the unit for which the analysis is performed). How can we quantify the overlap between two different paths, layer by layer? If a node is in the subgraph forming the path, the node is assigned a value of 1; if it is not, it is assigned a value of 0. Then, layer-wise overlap can be quantified by the Spearman correlation, which is exactly zero if there is only chance overlap, exactly 1.0 if there is perfect overlap, and exactly -1.0 if the units in a certain layer (corresponding to two different paths) are perfectly anticorrelated. Similarly, in the non-binary case (such as a path formed by activation patterns for natural images vs. feature visualization images, which is what we consider for the similarity analysis), the values assigned to the node are simply the activations, and the same analysis can be applied. Since we only care about the path similarity and not about whether this similarity is a linear relationship, Spearman's rank-order correlation is the correct measure to use here (while the Pearson correlation, plotted in Figure 10 for comparison, is a measure of a linear relationship).

Methods. Performing a full comparison is computationally expensive: Inception-V1's largest layer (`conv2d0_pre_relu_conv`) contains 802,816 values; computing the Spearman correlation for this takes about one third of a second. Inception-V1 has 138 layers and sub-layers (see Figure 7 x-axis labels for a list). Even if we just consider the comparison between natural images vs. natural images of a different class, for the ImageNet-1K validation set this amounts to 138 (= number of layers) $\cdot 50 \cdot 50$ (= number of comparisons between two specific classes of the ImageNet validation split) $\cdot \frac{1000 \cdot 1001}{2} - 1000$ (= number of comparisons when comparing each class with each other class except for itself) $= 172, 327, 500, 000$ comparisons. With 3 comparisons per second, this amounts to about *eighteen hundred years* required to do the full comparison. In order to make this more feasible, we chose the following approach. We randomly selected 10 classes via `numpy.random.seed(42); randomly_selected_class_indices = sorted(numpy.random.choice(1000, 10))`, resulting in `randomly_selected_class_indices=[20, 71, 102, 106, 121, 270, 435, 614, 700, 860]` and obtained 10 feature visualizations per class. Images that were not correctly classified by the model (wrong top-1 classification) were excluded from the comparison since those images do not constitute natural images for which the corresponding unit is selective for.

From this point onward, when computing the Spearman and Pearson correlations, we only performed every 10th comparison for natural images vs. natural images of the same class; every 100th comparison for natural images vs. natural images of a different class, and every 5th comparison for natural images vs. feature visualizations of the same class. For Cosine similarity (which is much faster), we performed every single comparison.

Raw and normalized plots. For each metric, *absolute* values are plotted in Figures 7, 9, and 11. For Figures 4, 8, and 10, *normalized* values are plotted. For these plots, we normalized the data according to the raw absolute values, i.e. such that natural images vs. natural images of the same class is set to 1.0 and natural images vs. natural images of a different class is set to 0.0 since it makes sense to interpret similarity results relative to those two extreme baselines. A tiny number of layers was excluded from the normalized comparison if the green and black points from the absolute plots differed by strictly less than a threshold of 0.01. To reduce noise in the orange curve (natural images vs. feature visualizations of the same class),

we smoothed the curve by convolving it with `scipy.ndimage.convolve` using a window size of 7 and the following uniform weights: `np.ones(windowsize)/windowsize`. The shaded blue area corresponds to the standard deviation of the orange data points, convolved over a window of of size `std_windowsize=5` which is then (for the lower bound of the blue area) subtracted from the orange curve, and (for the upper bound of the blue area) added to the orange curve; thus in total the blue area area vertically extends two standard deviations. The idea behind this is to give a rough visual estimate of the standard deviation range that the orange values have at certain points throughout the network. By itself, it does not provide an indication of statistical significance.

Figure 7: Absolute similarity (Spearman).

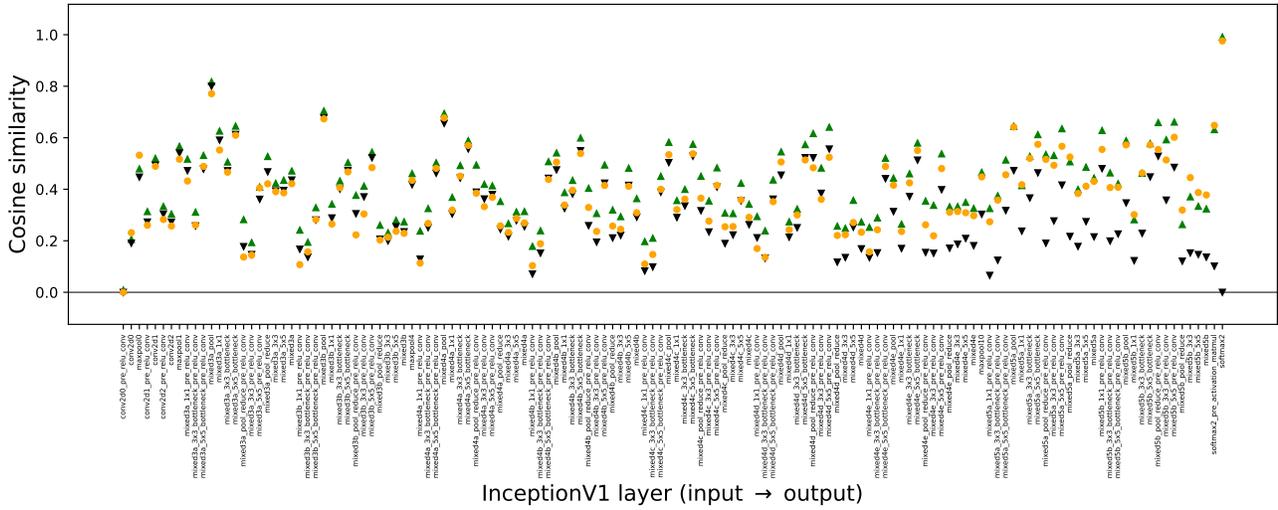Figure 8: Normalized similarity (Cosine).
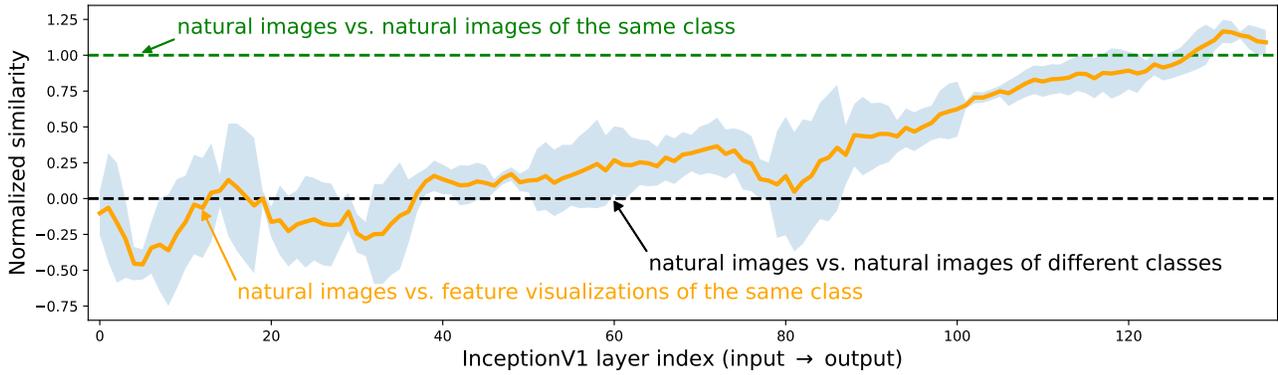
Figure 9: Absolute similarity (Cosine).

Figure 10: Normalized similarity (Pearson).

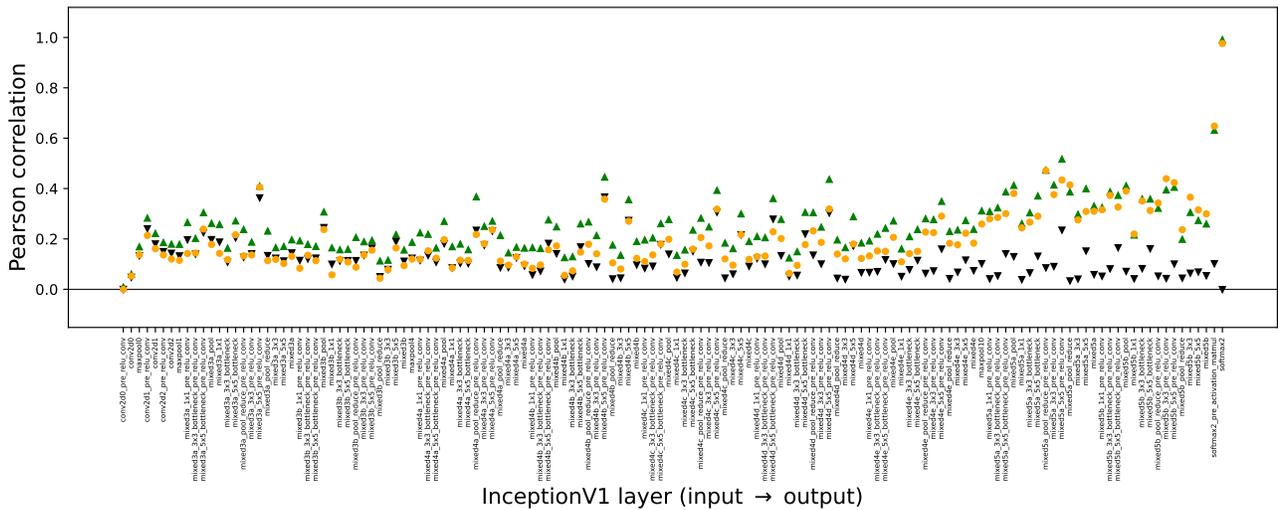Figure 11: Absolute similarity (Pearson).

F. Silent unit analysis

In Subsection 2.2, we constructed a circuit that contained *silent units*. These are units that remain inactive whenever natural input is processed by the network (no activation across the entire ImageNet training set; the units in the circuit were designed to become active once feature visualization is performed). We again ask whether similar units may exist in natural, unmodified networks. As a first step, we analyzed how many silent units exist in standard networks trained on ImageNet. Table 4 shows that **common networks indeed have a small but significant number of silent units** (between 2.28 % and 4.36 % of units for different ResNets). Naturally, these silent units are unlikely to be wired up in the same manipulative (orthogonal) fashion as in Subsection 2.2, but they may contribute to processing differences from Figure 4. As we will see, their existence challenges the point of view that what we see describes the computational function of the unit.

	ResNet-18	ResNet-50	ResNet-152	Inception-v1
silent units	100,585 (4.36 %)	218,994 (2.28 %)	669,973 (3.18 %)	11 ($3.59 \cdot 10^{-6}$ %)
silent channels	8 (0.20 %)	156 (0.69 %)	433 (0.60 %)	0 (0.0 %)

Table 4: **Number of silent units and channels.** Here, a unit or channel is called silent if it has zero activation across the entire ImageNet training dataset.

Figure 12: **Silent channels.** Feature visualization of 5 randomly selected silent channels and 5 randomly selected non-silent channels of a randomly selected layer of ResNet-50 (`layer3_1_conv3`). There are no obvious differences between silent and non-silent channel visualizations—can you guess which ones are which? (Answer below[5])

What do we seen when visualizing silent channels? In Figure 12, visualizations for five randomly selected silent channels and five randomly selected non-silent channels are shown. The visualizations all look innocuous: on the basis of a channel's feature visualization it can be impossible to tell whether it is silent or not. This means that the feature visualization of a unit cannot be taken as evidence for the role that the unit plays in fulfilling the network's objective (e.g., classification): even though the visualization shows all sorts of patterns, this does not mean that the network actually uses these patterns when classifying natural images.

This finding can be understood through a concept from neuroscience: the *projective field* of a neuron. In contrast to the widely known *receptive field* determined by incoming connections, the outgoing connections determine the projective field. In neuroscience, it was initially thought that the function of a neuron can be solely deduced from analyzing its receptive field. However, Lehky & Sejnowski (1988) discovered that units whose receptive fields looked like edge detectors actually detected whether a shaded surface was convex or concave, which only became apparent when analyzing the projective field: "In retrospect, it is obvious that a neuron without any output cannot have a computational function and that the same neuron can have more than one function depending on where it projects" (Sejnowski, 2006, p. 396). Therefore, **feature visualization cannot be used to determine the computational function of an intermediate neuron** since the computational function hinges on the projective field to which the method is oblivious to. The same argument applies to network dissection (Bau et al., 2017), another interpretability method. This is exemplified by silent units that have standard visualizations yet no computational function whatsoever.

G. Are more linear units easier to interpret?

The theory from Section 4 makes a prediction: the simpler a function is, the easier it should be to interpret given a feature visualization. We here attempt to empirically test this prediction. As a measure of simplicity, we use *path linearity*: based on a highly activating natural image (start point), we perform feature visualization to arrive at a highly activating optimized image (end point). We then analyze how much the optimization path deviates from linearity by measuring the angle between

---

[1]From left to right: n, s, s, s, s, n, s, n, n, n (s = silent, n = not silent).

gradients of the first n steps of the path. This metric (across many such paths) is then correlated with human experimental data from Zimmermann et al. (2021) for the same units, who measured how well humans can interpret different units in Inception-V1. Intriguingly, we find a significant correlation (Spearman's $r = -.36$, $p = .001$) between this human interpretability score and our path linearity measure (lower angle means higher interpretability) for the beginning of the trajectory (e.g., $n = 2$). Linearity at later steps in the trajectory does not seem to contribute much to human interpretability, thus increasing n to include all 512 steps decreases the overall correlation ($r = -.13$, $p = .23$; details below). Overall, we interpret this as very preliminary evidence in favor of the hypothesis that more linear units, at least at the beginning of the optimization trajectory, might be easier to interpret. An interesting direction for future work would be to enforce higher degrees of linearity, for instance through regularization or the architecture.

G.1. Method details for linearity experiments

As our theory (see Table 1) predicts that the simpler the function class of a network's unit is the easier it may be to interpret through feature visualizations, we set out to measure one type functional simplicity. Specifically, we approximate how *linear* a function's optimization trajectory is by computing how aligned its gradients are. The hypothesis is that more linear units (or visualization paths) might be more interpretable.

To measure the interpretability of a unit we use the experimental data provided by Zimmermann et al. (2021). They tested how well humans can differentiate maximally and minimally activating images for individual units of a CNN when supported with explanations in the form of feature visualizations (see Appendix I for details). We use the experimental data of 84 units as well as the $M = 20$ maximally activating natural dataset samples they provided.

Quantifying degree of nonlinearity through gradient angles   To measure the linearity of a unit we compute the following quantity for each unit: We start from a maximally activating dataset sample x_i^s and perform feature visualization to iteratively optimize this image to further increase the unit's activation. We use standard hyperparameters and optimize for $N = 512$ steps. During optimization we record the normalized gradients with respect to the current image $(\hat{g}_j(x_i^s))_{j=1,\dots,N}$ and compute the angle between successive steps:

$$\forall j = 1, \dots, N-1: \quad a_j(x_i^s) := \measuredangle(\hat{g}_j(x_i), \hat{g}_{j+1}(x_s^s)). \tag{12}$$

We take the average over all M maximally activating images and denote the *average gradient path angle* as:

$$\forall j = 1, \dots, N-1: \quad \mathrm{AGPA}_j := \frac{1}{M} \sum_{i=1}^{M} a_j(x_i^s). \tag{13}$$

Finally, we denote the average of the average gradient path angle AGPA as the average gradient angle:

$$\mathrm{AGA} = \frac{1}{N-1} \sum_{i=1}^{N-1} \mathrm{AGPA}_i. \tag{14}$$

To answer our initial question—whether simpler/more linear units are more interpretable—we now measured the rank correlation between the average gradient angle and the interpretability scores by Zimmermann et al. (2021). Intriguingly, we find a significant correlation (Spearman's $r = -.36$, $p = .001$) between this human interpretability score and our path linearity measure (lower angle means higher interpretability) for the beginning of the trajectory (e.g., $n = 2$). Linearity at later steps in the trajectory does not seem to contribute much to human interpretability, thus increasing n to include all 512 steps decreases the overall correlation. The results depending on path length are plotted in Figure 13a.

Quantifying degree of nonlinearity through deviations from linear interpolation   There are many different ways that could be used to measure path or unit linearity. As a more global measure, we also tested another one: Here, we begin by computing a linear interpolation between the maximally activation data samples we initialize the optimization with x_i^s and the final visualization x_i^f:

$$\{\, z \mid x_i^s + \alpha(x_i^f - x_i^s) \quad \forall \alpha \in [0, 1] \,\}. \tag{15}$$
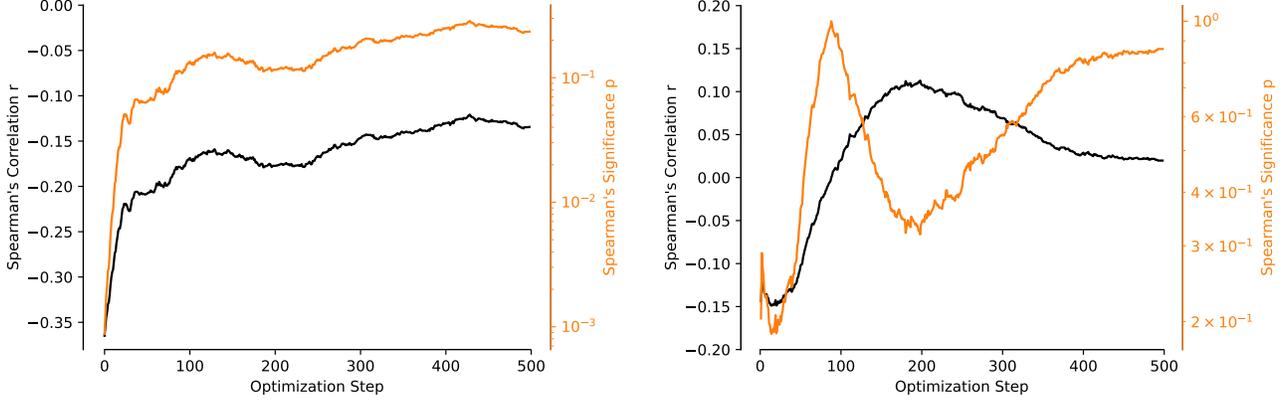
(a) Average gradient path angle AGPA_k. Strikingly, only the first few gradient angles are strongly and significantly correlated with the units' interpretability.

(b) Average line distance path AGLD_k. Interestingly, while we see a anti-correlation at the beginning (cf. Figure 13a), this turns into a weak correlation. However, these correlations are not significant.

Figure 13: Development of Spearman's rank correlation between the units' interpretability score and the (a) average gradient path angle AGPA_k and (b) average line distance path ALDP_k as a function of the number of optimization steps k to consider.

Next, for each step j of the optimization process we compute the distance of the current image $x_j(x_i^s)$ to to the linear interpolation

$$d_j(x_i^s) = d\left(x_j((x_i^s)), \{ z \mid x_i^s + \alpha(x_i^f - x_i^s) \quad \forall \alpha \in [0,1] \}\right), \tag{16}$$

where $d(\cdot, \cdot)$ represents the ℓ_2 distance. Analogously to the computation above, we then take the mean over the different start images and define this property as the average line distance path

$$\forall j = 1, \ldots, N - 1: \quad \mathrm{ALDP}_j := \frac{1}{M} \sum_{i=1}^{M} \frac{d_j(x_i^s)}{d(x_i^2, x_i^f)}, \tag{17}$$

where we normalize the distances from the line with the distance between start and end point of the optimization trajectory to make these values scaleless. Finally, by averaging again over optimization steps final average line distance:

$$\mathrm{ALD} = \frac{1}{N-1} \sum_{i=1}^{N-1} \mathrm{ALDP}_i. \tag{18}$$

The lower the ALD value, the smaller is the deviation of the optimization path from a line. For this global measure, we see a non-significant relation between the average line distance and the interpretability scores (Spearman's $r = 0.02$, $p = 0.86$). Analogously to the local measure explained above (gradient angle), we zoom into these results again in Figure 13b. While there might be a weak anti-correlation at the beginning of the optimization path (which later turns into a weak correlation), none of these are significant.

Interpretation. We believe there is more to be understood: while it is intriguing that linearity at the beginning of the optimization trajectory is predictive of a human interpretability score, the global measure of linearity (through distance to linear interpolation) does not seem to be predictive and more investigations are needed to fully understand this phenomenon—as we say in the main paper, this can only be considered "very preliminary evidence". A promising way forward could be to enforce (or optimize for) different properties in neural networks and measure whether this improves human interpretability.

H. Literature expectations about feature visualization

This short section provides a few expectations/hopes that are presented in the literature when it comes to feature visualizations.

Original activation maximization paper by Erhan et al. (2009):

- "a pattern to which the unit is responding maximally could be a good first-order representation of what a unit is doing"
- "It is perhaps unrealistic to expect that as we scale the datasets to larger and larger images, one could still find a simple representation of a higher layer unit."
- "we hope that such visualization techniques can help understand the nature of the functions learned by the network"

More recent literature:

- "Feature visualization allows us to see how GoogLeNet, trained on the ImageNet dataset, builds up its understanding of images over many layers" (Olah et al., 2017)
- "Feature visualization answers questions about what a network—or parts of a network—are looking for by generating examples." (Olah et al., 2017)
- "If we want to find out what kind of input would cause a certain behavior—whether that's an internal neuron firing or the final output behavior—we can use derivatives to iteratively tweak the input towards that goal" (Olah et al., 2017)
- "optimization approach can be a powerful way to understand what a model is really looking for, because it separates the things causing behavior from things that merely correlate with the causes". "Optimization isolates the causes of behavior from mere correlations." (Olah et al., 2017)
- "In the quest to make neural networks interpretable, feature visualization stands out as one of the most promising and developed research directions. By itself, feature visualization will never give a completely satisfactory understanding. We see it as one of the fundamental building blocks that, combined with additional tools, will empower humans to understand these systems." (Olah et al., 2017)
- "To make a semantic dictionary, we pair every neuron activation with a visualization of that neuron and sort them by the magnitude of the activation."; "Semantic dictionaries give us a fine-grained look at an activation: what does each single neuron detect?" (Olah et al., 2018)
- "Feature visualization helps us answer what the network detects" (Olah et al., 2018)
- "The behavior of a CNN can be visualized by sampling image patches that maximize activation of hidden units [...], or by using variants of backpropagation to identify or generate salient image features" (Bau et al., 2017)
- "Activation maximization techniques enable us to shine light into the black-box neural networks." (Nguyen et al., 2019)

Critical voices:

- "While these methods may be useful for building intuition, they can also encourage three potentially misleading assumptions: that the visualization is representative of the neuron's behavior; that the neuron is responsible for a clearly delineated portion of the task or the network's behavior; and that the neuron's behavior is representative of the network's behavior." (Leavitt & Morcos, 2020)
- "synthetic images from a popular feature visualization method are significantly less informative for assessing CNN activations than natural images" (Borowski et al., 2021)
- "[We] find no evidence that a widely-used feature visualization method provides humans with better 'causal under-standing' of unit activations than simple alternative visualizations" (Zimmermann et al., 2021)
- "Neural networks often contain 'polysemantic neurons' that respond to multiple unrelated inputs." (Olah et al., 2020)
- "Units similar to those [hand-picked units] may be the exception rather than the rule, and it is unclear whether they are essential to the functionality of the network. For example, meaningful selectivities could reside in linear combinations of units rather than in single units, with weak distributed activities encoding essential information." (Kriegeskorte, 2015)

I. Relationship to psychophysical experiments

Borowski et al. (2021) and Zimmermann et al. (2021) performed psychophysical experiments to investigate the tness of feature visualizations for human observers. Both papers find that natural highly activating images are more interpretable (as measured by human prediction performance) compared to feature visualizations. A candidate explanation for this behaviour is our analysis in Section 3, showing that for last-layer Inception-V1 feature visualizations, those visualizations are processed along very different paths for most of the network (compared to natural images as a baseline).

The task used by Borowski et al. (2021) is related to the third column of Table 1. They asked participants to predict which one of two natural images is strongly activating for a certain unit based on maximally and minimally activating feature visualizations for that unit. This can be seen as an easier version of the task in Table 1: Borowski et al. (2021) did not use random test samples but instead two curated samples, out of which one has extremely high and one has extremely low

activations. They find that that humans are able to do this task above chance.

At first glance, this result seems to contradict our theoretical finding from Theorem 2, which states that reliable prediction is impossible unless additional assumptions about the function are known. However, there is no contradiction: our theory allows for the possibility of a specific function (e.g., a specific neural network unit) and a specific decoder (e.g., a human observer) to be aligned in the sense that predictions about the function can happen to be correct—however, for every function f for which the decoder is correct there is a function f' from the same function family for which the decoder is wrong (in spirit, a case of "no free lunch" for feature visualization). If an observer gets significantly above chance in one case, they would pay the price by being significantly below chance in the other case. To the best of our knowledge, there is currently no way for the observer to know in advance whether they're visualizing a function f for which their decoding is aligned or a function f' for which their decoding leads to the wrong conclusions. It is an interesting open question to develop a practical and rigorous approach to distinguish these cases, perhaps relying on additional information such as the data distribution (e.g., does ImageNet lead to benign f more often?) and the training procedure (e.g., does SGD lead to benign f more often?).

J. Broader impacts

Our paper investigates the reliability of feature visualizations. Overall, we expect this to contribute to better scrutiny towards existing interpretability methods, which hopefully inspires the development of more reliable interpretability methods in the future, as well the development of models that incorporate certain reliably "interpretability-enabling" assumptions right from the start, rather than being faced with the (sometimes impossible) task of post-hoc interpretability through feature visualizations.

In terms of potential negative impact, the fooling methods developed here could be used to deceive an entity (e.g., a model auditor or regulator) as described in Section 2. That being said, we believe that the risk is lower if this knowledge is public—it would be much more problematic to believe that feature visualizations can be taken at face value, because then whoever designs a fooling circuit would be met with an unsuspecting audience.

K. Limitations

We see the following potential limitations:

1. We design methods that fool feature visualizations. Once it is known that a certain fooling method might be used, it is easy to develop a detection mechanism. That said, the space of potential fooling methods is vast. Therefore, developing a specific detection mechanism would probably lead to a pattern similar to adversarial attacks and defenses: after an attack is developed, a detection/circumvention method defends, which is then again circumvented by a revised attack/fooling method.
2. The fooling methods that we developed in Section 2 assume bad intent. Most models are developed with good intent. However, we believe that the reliability of interpretability methods should not rely on assuming good intentions. The experiments in Section 3 and the theory from Section 4 are independent of good/bad intent assumptions.
3. The potential assumptions on the function space listed in Table 1 are not exhaustive. It is possible that other assumptions enable stronger prediction.
4. The investigated networks, Inception-V1 and ResNet-50, are of course not exhaustive either. None of our methods is specific to those networks. This means that other networks could be equipped with a fooling circuit, too. At the same time, the empirical results might look different for other networks, which would be an interesting direction to explore in future work.
5. One could argue that the fooling circuit in Figure 3 only deceives a user when looking at unit A, whereas the other units still have their original visualization. That's correct: the fooling circuit manipulates the visualization of A but not of e.g. units D or F. From a single unit perspective, this is already problematic since it means we can't trust a unit's visualization. It would be interesting avenue for future work to develop networks where every single unit's visualization is misleading.
6. There is no one definition of what it means to "understand" or "explain" a neural network, since those are very vague terms. We seek to be precise about our definition and motivate it with expectations about feature visualization stated in the literature (Appendix H), but we realize that this means not everyone's notion of "understanding" / "explaning" neural networks can be captured by our definition.

L. Image sources

Figure 1. "Girl with a Pearl Earring" was downloaded from here and is public domain according to the website. "Puppies" (West Highland White Terrier puppies) by Lucie Tylová, Westik.cz was downloaded from here and is licensed under CC BY-SA 3.0 according to the website. "Mona Lisa" was downloaded from here and is public domain according to the website. All three images were cropped to size 224×224 pixels. The photographs were then embedded in the weights of a single convolutional layer and to some degree recovered by the feature visualization method, subject to distortion by the method's transformations.