SmoothFool: An Efficient Framework for Computing Smooth Adversarial Perturbations

Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, Jeremy Dawson, Nasser M. Nasrabadi
West Virginia University

{ad0046, ssoleyma}@mix.wvu.edu, fariborztaherkhani@gmail.com,
{nasser.nasrabadi, jeremy.dawson}@mail.wvu.edu

Abstract

Deep neural networks are susceptible to adversarial manipulations in the input domain. The extent of vulnerability has been explored intensively in cases of ℓ_p-bounded and ℓ_p-minimal adversarial perturbations. However, the vulnerability of DNNs to adversarial perturbations with specific statistical properties or frequency-domain characteristics has not been sufficiently explored. In this paper, we study the smoothness of perturbations and propose Smooth-Fool, a general and computationally efficient framework for computing smooth adversarial perturbations. Through extensive experiments, we validate the efficacy of the proposed method for both the white-box and black-box attack scenarios. In particular, we demonstrate that: (i) there exist extremely smooth adversarial perturbations for well-established and widely used network architectures, (ii) smoothness significantly enhances the robustness of perturbations against state-of-the-art defense mechanisms, (iii) smoothness improves the transferability of adversarial perturbations across both data points and network architectures, and (iv) class categories exhibit a variable range of susceptibility to smooth perturbations. Our results suggest that smooth APs can play a significant role in exploring the vulnerability extent of DNNs to adversarial examples. The code is available at ***https://github.com/alldbi/SmoothFool***

1. Introduction

Despite revolutionary achievements of deep neural networks (DNNs) in many computer vision tasks [22, 44], carefully manipulated input samples, known as *adversarial examples*, can fool learning models to confidently make wrong predictions [40]. Adversarial examples are potential threats to almost all applications of machine learning [2, 13, 18], but the case is more severe in the context of
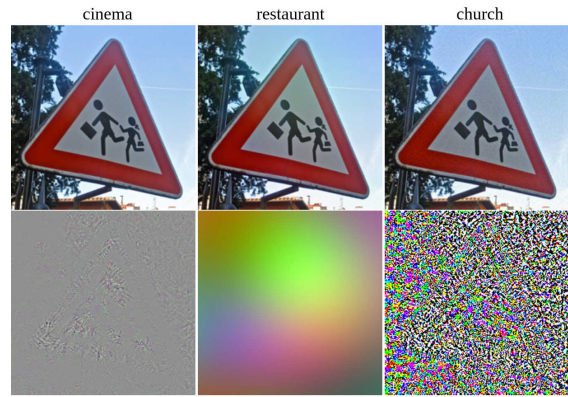

Figure 1: Comparing smooth APs with conventional APs. Each column from left to right shows the adversarial image and the corresponding APs computed by DeepFool [31], SmoothFool ($\sigma_g = 75$) and IGSM [23], respectively, on ResNet-101 [14]. The predicted label for each image is depicted above the column. Perturbations are magnified for a better visibility.

computer vision, particularly, due to the complexity of tasks [36], huge cardinality of input spaces [41], and sensitivity of applications [10, 16, 43, 38]. Analyzing DNNs as differentiable transfer functions have led to substantial studies exploring embedding spaces and their characteristics in regard to training paradigms. However, the adversarial behavior has highlighted the importance of studying the topology of decision boundaries and their properties in high dimensional data spaces [36, 15]. Considering a *white-box* scenario where the network architecture and all its parameters are known, several approaches (attacks) have been proposed to explore the robustness of decision boundaries in the presence of ℓ_p-bounded [40, 12, 23] and ℓ_p-minimal [31, 1, 33, 39, 29] adversarial perturbations (APs). However, the vulnerability of DNNs to APs with specific statistical properties or frequency-domain characteristics, which lie beyond the conventional ℓ_p-norm constraints, has re-

mained less explored.

In this study, we seek to explore the landscape of robustness of DNNs to APs with modified frequency-domain characteristics. Specifically, we focus on smooth APs due to several advantages they offer compared to the conventional APs. First, they are more physically realizable than non-smooth APs since printing devices are critically less accurate in capturing high frequency structures due to the sampling noise [37]. Also, severe differences between adjacent pixels in the printed adversarial examples are unlikely to be accurately captured by cameras due to their low-pass spatial frequency response [19]. Second, the high-frequency structure of conventional APs has provoked an intensive adoption of explicit [32, 26, 34] and implicit [41, 27, 35] denoising methods to mitigate the adversarial effect. However, we demonstrate that a slight modification of local statistics of APs causes a vital failure of state-of-the-art defenses. Third, smoothness significantly enhances the transferability of APs across classifiers and data points by improving the invariance of perturbations to translation [6]. This improves the performance of the attack in the *black-box* scenario where the parameters of the target model are not known to the adversary. Forth, smoothness enhances plausible deniability and allows the attacker to disguise APs as natural phenomena such as shadows. In this way, the magnitude of APs can be increased notably since imperceptibility is less important.

We formulate the problem of constructing smooth APs according to a general definition of smoothness and exploit the geometry of decision boundaries to find computationally efficient solutions. Our main contributions are the followings:

- We propose SmoothFool, a geometry inspired framework for computing smooth APs which exploits the topology of decision boundaries to find efficient APs.

- We analyze various properties of smooth APs and validate their effectiveness for both the white-box and black-box attack scenarios.

- We show the susceptibility of two major group of defenses against smooth APs by breaking several state-of-the-art defenses.

- We integrate SmoothFool with previous studies on universal APs and demonstrate the existence of smooth universal APs that generalize well across data samples and network architectures.

2. Related Work

2.1. Adversarial Attack

Despite the highly non-linear nature of DNNs, they have been observed to exhibit linear characteristics around the actual parameters of the model and the input samples [12, 8, 9]. In particular, Goodfellow *et al*. [12] showed that the prediction of DNNs can be changed drastically by translating the input sample toward the gradient of the classification loss. Hence, they proposed the fast gradient sign method (FGSM) as a single step attack incorporating solely the sign of gradients to craft APs. Kurakin *et al*. [23] improved the performance of FGSM by adopting an iterative procedure called IGSM. Moosavi *et al*. [31] proposed Deep-Fool to find approximately ℓ_p-minimal APs by iteratively translating input samples toward the linearized approximation of the closest decision boundary. Our methodology builds on DeepFool to find minimal smooth APs.

Some prior studies have considered smoothness in adversarial attacks. Sharif *et al*. [37] added a total variation (TV) loss to the main objective of the attack to enhance the physical realizability of the resulting perturbations and showed that smoothness of APs improves their effectiveness for the real-world applications. Fong *et al*. [11] demonstrated that smoothing important regions in the input example can deteriorate the confidence of prediction. They utilized this observation to interpret the decisions of DNNs. Hosseini *et al*. [17] proposed constructing semantic adversarial examples by randomly shifting Hue and Saturation components of benign samples in the HSV color space. Dong *et al*. [6] demonstrated that robustness of DNNs to slight translations can be exploited to improve the trasferability of adversarial examples. Interestingly, the final perturbations crafted using their approach exhibited low-pass frequency response. However, their methodology is applicable for a limited level of smoothness since the prediction of DNNs is invariant for solely small translations of the input sample.

Fundamentally, our work differs from previous approaches since we seek to find approximately ℓ_2-minimal APs capable of offering arbitrary levels of smoothness. Also, our main goal is to formulate and compute smooth APs, not to find smooth adversarial examples, since the later can critically destroy the structure of images.

2.2. Defense Methods

Since the first observation of APs, their noisy structure has been harnessed to find defense strategies. Several studies have incorporated explicit denoising techniques to mitigate the adversarial effect. Liao *et al*. [26] showed that the distribution of high-level representations in DNNs provides an effective guidance to denoise adversarial examples and proposed the high-level representation guided denoiser (HGD). Training DNNs using adversarial examples, known as *adversarial training* [12, 27, 41], has been shown to provide a relative adversarial robustness. Adversarial training can be considered as an implicit denoising technique which reduces the sensitivity of predictions to slight changes in the input domain. Manifold learning is another implicit de-

noising defense. A well-known example for this type of defense is *MagNet* [28] which deploys autoencoders for mapping input examples onto the manifold of natural examples. Later, we utilize these defenses to evaluate the effectiveness of smooth APs.

3. Smooth Adversarial Perturbations

3.1. Problem Definition

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a classifier mapping input sample $\boldsymbol{x} \in [0, 1]^n$ to m classification scores $f_j(\boldsymbol{x})$, associated with each class $j \in \{0, \dots, m-1\}$. The class predicted by the network can be computed as:

$$c(\boldsymbol{x}) = \arg\max_j f_j(\boldsymbol{x}). \qquad (1)$$

The problem of constructing smooth APs can be formulated as the following optimization problem:

$$\arg\min_{\boldsymbol{r}} ||\boldsymbol{r}||_2 + \lambda\Omega(\boldsymbol{r}) \text{ subject to:}$$

$$\begin{aligned} &1. \ c(\boldsymbol{x} + \boldsymbol{r}) \neq c(\boldsymbol{x}), \qquad (2) \\ &2. \ \boldsymbol{x} + \boldsymbol{r} \in [0, 1]^n, \end{aligned}$$

where $\boldsymbol{r} \in \mathbb{R}^n$ is the AP, $\Omega(.)$ is a measure of roughness, and λ is a Lagrangian coefficient controlling the trade-off between roughness and magnitude of the perturbation. Generally, the roughness of perturbations can be defined based on their local variations. Such variations have an explicit interpretation in the frequency domain where the power of each frequency component captures the specific range of variations. Considering this perspective, we use a frequency response function H to formulate the definition of roughness since it can denote how much each frequency component contributes to the intended roughness. For clarity, we substitute H with H_{hp} to highlight the high-frequency nature of roughness and denote $H_{lp} = 1 - H_{hp}$ as the complementary low-pass filter which defines the equivalent smoothness. We use the total energy of the high-frequency components of $\boldsymbol{r}$ as a general measure of roughness, and define Ω as:

$$\Omega(\boldsymbol{r}, H_{lp}) := \int_{-\infty}^{+\infty} R(\omega)^2(1 - H_{lp}(\omega))^2 d\omega, \qquad (3)$$

where R is the Fourier transformation of perturbation $\boldsymbol{r}$, and H_{lp} is the frequency response of a given low-pass filter defining the range of acceptable smoothness, and is a free parameter of the definition.

The perturbation $\boldsymbol{r}$ in our problem is represented as a set of spatially discrete APs for each pixel location $u \in \{0, \dots, n-1\}$[1], and Ω can be conveniently computed in the spatial domain as:

$$\Omega(\boldsymbol{r}; \boldsymbol{h}) = ||\boldsymbol{r} - \boldsymbol{r} * \boldsymbol{h}||_2^2, \qquad (4)$$

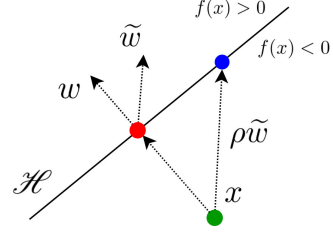[1]Here we assume the input image $\boldsymbol{x}$ is a 1D signal, and later in the experiments we adopt all formulations for 2D images.

Figure 2: Finding smooth AP for a linear binary classifier. Red and blue dots show the ℓ_2 projection and smooth projection of $\boldsymbol{x}$ onto the decision boundary, respectively. For an easier demonstration, $\boldsymbol{x}$ is assumed to belong to class -1.

where $*$ denotes convolution, and $\boldsymbol{h}$ is the discrete approximation of H_{lp} in the spatial domain. In the rest of the paper, our work builds on this definition of roughness (and, equivalently, smoothness) and aims to find APs which are relatively smooth based on any predefined $\boldsymbol{h}$ compared to perturbations crafted by other contemporary attacks. Due to the non-convex nature of the problem, we exploit the geometric properties of the decision boundary of DNNs to find a relaxed solution for the optimization problem given in Equation 2.

3.2. Linearized solution

Based on previous findings [31, 20, 8], the decision boundary of a differentiable classifier, f, around $\boldsymbol{x}$ can be well approximated by a hyperplane passing through the minimal ℓ_2 adversarial example $\boldsymbol{x}_p$ corresponding to $\boldsymbol{x}$, and the normal vector $\boldsymbol{w}$ orthogonal to the decision boundary at $\boldsymbol{x}_p$ as $\mathscr{H} \triangleq \{\boldsymbol{x} : \boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_p) = 0\}$. We assume $\boldsymbol{x}_p$ and, consequently, $\boldsymbol{w}$ associated with each $\boldsymbol{x}$ is available, and later we utilize an appropriate contemporary attack to compute $\boldsymbol{x}_p$ and $\boldsymbol{w}$. Having $\boldsymbol{x}_p$ provides two benefits. First, it allows us to linearize the closest decision boundary around $\boldsymbol{x}$. Second, we can reduce the problem to a binary classification problem, where the goal of the attack would be to compute the smooth perturbation $\boldsymbol{r}$ which yields $c(\boldsymbol{x} + \boldsymbol{r}) = c(\boldsymbol{x}_p)$. Consequently, we rewrite the optimization problem given in Equation 2 as:

$$\arg\min_{\boldsymbol{r}} ||\boldsymbol{r}||_2 + \lambda\Omega(\boldsymbol{r}; \boldsymbol{h}) \text{ subject to:}$$

$$\begin{aligned} &1. \ \boldsymbol{w}^\top(\boldsymbol{x} + \boldsymbol{r}) - \boldsymbol{w}^\top\boldsymbol{x}_p = 0, \qquad (5) \\ &2. \ \boldsymbol{x} + \boldsymbol{r} \in [0, 1]^n. \end{aligned}$$

In this setup, an efficient solution can be obtained from a smooth projection of $\boldsymbol{x}$ onto the estimated hyperplane $\mathscr{H}$. Such a projection can be computed by translating $\boldsymbol{x}$ using the adversarial perturbation $\boldsymbol{r} = \rho\widetilde{\boldsymbol{w}}$, where $\widetilde{\boldsymbol{w}}$ is a smooth approximation of $\boldsymbol{w}$, and ρ scales $\widetilde{\boldsymbol{w}}$ to map $\boldsymbol{x} + \boldsymbol{r}$ on $\mathscr{H}$ as:

$$\rho = \frac{\boldsymbol{w}^\top(\boldsymbol{x}_p - \boldsymbol{x})}{\boldsymbol{w}^\top\widetilde{\boldsymbol{w}}}. \qquad (6)$$
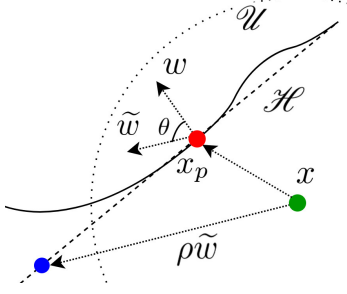
Figure 3: A demonstration of the topology of the decision boundary in the vicinity of data point $\boldsymbol{x}$. $\mathscr{U}$ illustrates the region where the decision boundary can be assumed to be approximately flat. Smooth projection of $\boldsymbol{x}$ onto the estimated hyperplane $\mathscr{H}$ often results in a solution out of $\mathscr{U}$.

Figure 2 provides a simple visualization of this projection. It worth mentioning that for the linear binary classifier choice of f, the optimal smooth perturbation has the closed-form solution: $\boldsymbol{r} = -\dfrac{f(\boldsymbol{x})}{\boldsymbol{w}^\top \widetilde{\boldsymbol{w}}} \widetilde{\boldsymbol{w}}$. Generally, the estimation $\widetilde{\boldsymbol{w}}$ must hold two conditions to provide a valid solution for the linearized problem. First, $\widetilde{\boldsymbol{w}}$ should not be orthogonal to $\boldsymbol{w}$. Second, the estimation should remove high-frequency components of $\boldsymbol{w}$ in order to keep $\Omega(\rho\widetilde{\boldsymbol{w}}; \boldsymbol{h})$ low. Without loss of generality, we consider a low-pass filter $\boldsymbol{g}$ to estimate $\widetilde{\boldsymbol{w}}$ by convolution as: $\widetilde{\boldsymbol{w}} = \boldsymbol{g} * \boldsymbol{w}$, since it is easy to compute, and the only condition on $\boldsymbol{g}$ is that its cut-off frequency should be less than the cut-off frequency of $\boldsymbol{h}$.

The final smooth perturbation that can project $\boldsymbol{x}$ on $\mathscr{H}$ can be computed as:

$$\boldsymbol{r} = \frac{\boldsymbol{w}^\top (\boldsymbol{x}_p - \boldsymbol{x})}{\boldsymbol{w}^\top (\boldsymbol{g} * \boldsymbol{w})} (\boldsymbol{g} * \boldsymbol{w}). \tag{7}$$

In this formulation, the cut-off frequency of $\boldsymbol{g}$ is associated with λ in the optimization problem given in Equation 5 since it controls the smoothness of perturbation $\boldsymbol{r}$. ℓ_2-DeepFool [31] constructs adversarial examples which are shown to be a good approximation of the ℓ_2-minimal adversarial example for an input sample, and the assumption of flat decision boundaries around the constructed examples is believed to be practically valid [31, 29]. Therefore, we utilize it to generate $\boldsymbol{x}_p$ and estimate $\boldsymbol{w}$ using the first order Taylor expansion of f at $\boldsymbol{x}_p$ as:

$$\boldsymbol{w} = \nabla f_{c(\boldsymbol{x}_p)}(\boldsymbol{x}_p) - \nabla f_{c(\boldsymbol{x})}(\boldsymbol{x}_p). \tag{8}$$

In practice, the high-frequency structure of the gradients of DNNs increases the angle between $\boldsymbol{w}$ and $\widetilde{\boldsymbol{w}}$. Consequently, ρ in Equation 6 takes large values which often maps the input sample outside the legitimate range $[0, 1]^n$. In the next section, we propose a smooth clipping technique to overcome this problem.

Algorithm 1 SmoothClip

1: **input:** Image $\boldsymbol{x}$, perturbation $\boldsymbol{r}$, low-pass filter $\boldsymbol{g}$, step size ϵ.
2: **output:** Smoothly clipped perturbation $\boldsymbol{r}_c$.
3: Initialize $\boldsymbol{r}_c \leftarrow \boldsymbol{r}$.
4: **while** $\max(\boldsymbol{x} + \boldsymbol{r}_c) > 1$ or $\min(\boldsymbol{x} + \boldsymbol{r}_c) < 0$ **do**
5:     $\boldsymbol{m}_0 = \mathbb{1}_{>0}(-(\boldsymbol{x} + \boldsymbol{r}_c)) * \boldsymbol{g}$,
6:     $\boldsymbol{m}_1 = \mathbb{1}_{>0}((\boldsymbol{x} + \boldsymbol{r}_c) - 1) * \boldsymbol{g}$,
7:     $\Delta_1 = \max(\boldsymbol{x} + \boldsymbol{r}_c - 1)\boldsymbol{m}_1$,
8:     $\Delta_0 = \min(\boldsymbol{x} + \boldsymbol{r}_c)\boldsymbol{m}_0$,
9:     $\boldsymbol{r}_c \leftarrow \boldsymbol{r}_c - \epsilon(\Delta_1 + \Delta_0)$,
10: **end while**
11: **return** $\boldsymbol{r}_c$.

3.3. Validating Perturbations

The final adversarial example should reside inside the valid range of the input domain. An ordinary approach to hold this condition, especially in iterative attacks, is to clip the resulting adversarial examples [23, 29]. The clipping function, $Clip$, takes the constructed adversarial image and truncates each pixel value independently to fall within the valid range of the input space. However, applying this to smooth perturbations as: $\boldsymbol{r}_c = Clip(\boldsymbol{x} + \boldsymbol{r}) - \boldsymbol{x}$, will deteriorate the smoothness of perturbation. This is because the clipping function truncates each pixel individually and discards the local correlation between neighborhood perturbations. Specifically, this issue happens at edges and high-frequency areas of $\boldsymbol{x}$ as shown in Figure 5. A closed-form solution for smooth clipping, which should consider neighborhood correlation of perturbations (based on $\boldsymbol{g}$), results in a high complexity solution. We propose a simple and iterative approach for smoothly clipping the out-of-bound pixels. In the ith iteration, when the range of $\boldsymbol{x} + \boldsymbol{r}^i$ remains out of the valid range, we compute masks $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$ as indicators of pixels which exceed the valid bound as:

$$\boldsymbol{m}_0^i = \mathbb{1}_{>0}(-(\boldsymbol{x} + \boldsymbol{r}^i)), \tag{9}$$

$$\boldsymbol{m}_1^i = \mathbb{1}_{>0}((\boldsymbol{x} + \boldsymbol{r}^i) - 1), \tag{10}$$

where $\mathbb{1}_{>0}(.)$ is an indicator function that outputs 1 for elements greater than zero. To incorporate the neighborhood correlation of perturbations, we use the exact low-pass filter $\boldsymbol{g}$ used in Equation 7 to propagate the out-of-bound error to the neighborhood perturbations as: $\boldsymbol{m}_1^i \leftarrow \boldsymbol{m}_1^i * \boldsymbol{g}$ and $\boldsymbol{m}_0^i \leftarrow \boldsymbol{m}_0^i * \boldsymbol{g}$. Then, using a step size ϵ and maximum value of the out-of-bound error, we adjust the perturbation as:

$$\begin{aligned} \boldsymbol{r}^{i+1} = \boldsymbol{r}^i &- \epsilon \max(\boldsymbol{x} + \boldsymbol{r}^i - 1)\boldsymbol{m}_1^i \\ &- \epsilon \min(\boldsymbol{x} + \boldsymbol{r}^i)\boldsymbol{m}_0^i. \end{aligned} \tag{11}$$

This iterative algorithm terminates when all pixels in $\boldsymbol{x} + \boldsymbol{r}^i$ reside within the valid range. We refer to this algorithm as *SmoothClip*, and Algorithm 1 summarizes its functionality.
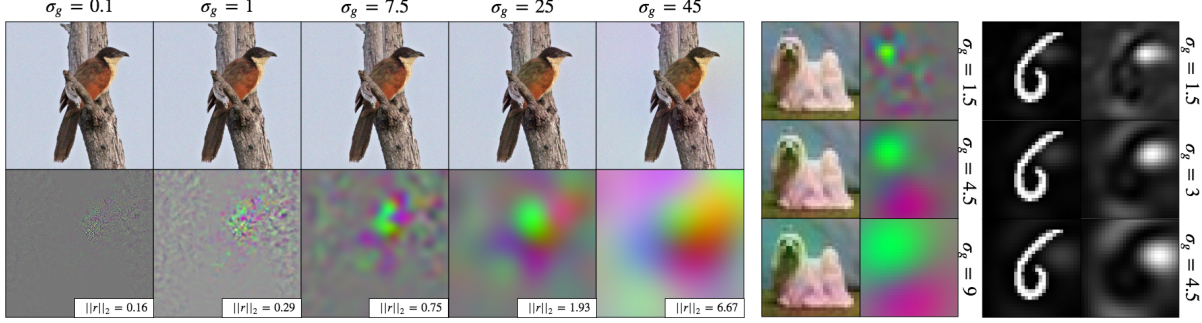
Figure 4: Visual demonstration of increasing smoothness of APs. Each set of images, from left to right, show adversarial examples and smooth APs computed for samples from ImageNet, CIFAR-10, and MNIST datasets on ResNet-101, ResNet-18, and LeNet architectures, respectively. Samples are from *'coucal'*, *'dog'*, and *'6'* classes and misclassified as *'robin'*, *'bird'*, and *'0'*.

Figure 5: An example of applying a normal clipping on a smooth AP. Left: a benign sample correctly classified as *'strawberry'* by VGG16. Middle: an adversarial example classified as *'pineapple'*. Right: the perturbation after normal clipping.

3.4. General Solution

In a general case for a non-linear classifier f, there is no guarantee that perturbations computed by Equation 7 cause input samples to pass the actual non-linear boundary. Figure 3 demonstrates a visualization of this fact. To overcome this problem, we adopt an iterative procedure. In each iteration, using the closest adversarial example, $\boldsymbol{x}_p^i$, corresponding to the sample $\boldsymbol{x}^i$, we linearize the decision boundary and compute the smooth projection of $\boldsymbol{x}^i$ on the approximated hyperplane using Equation 7.

Afterward, we smoothly rectify the resulting perturbation, $\boldsymbol{r}$, and repeat this procedure until $c(\boldsymbol{x}^i) \neq c(\boldsymbol{x}^0)$, as detailed in Algorithm 2. Here, the smoothness of the final perturbation depends on the smoothness in each iteration. Consider $r^{tot} = x^i - x^0 = \sum_{j=0}^{i} r^j$, where i is the total number of iterations, and r^j is the jth smooth AP. It can be shown that the roughness of the overall perturbation is bounded as: $\Omega(r^{tot}; h) \leq i^2 \max_j \Omega(r^j; h)$. To compute an AP with the desired level of roughness defined by h, we select g such that $\forall j : \Omega(r^j; h) \ll ||r^j||_2^2$, *i.e.*, the cut-off frequency of g should be smaller than h. In practice, $\max_j \Omega(r^j; h) \ll i^{-2}$, *i.e.*, even for a significantly smooth choices of $\boldsymbol{g}$ the algorithm converges in few iterations.

Algorithm 2 SmoothFool

1: **input:** Image $\boldsymbol{x}$, low-pass filter $\boldsymbol{g}$.
2: **output:** Smooth perturbation $\boldsymbol{r}$.
3: Initialize $\boldsymbol{x}^0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
4: **while** $c_f(\boldsymbol{x}^0) = c_f(\boldsymbol{x}^i)$ **do**
5:   $\boldsymbol{r}_p = \text{DeepFool}(\boldsymbol{x}^i)$,
6:   $\boldsymbol{x}_p = \boldsymbol{x}^i + \boldsymbol{r}_p$,
7:   $\boldsymbol{w}^i = \nabla f_{c(\boldsymbol{x}_p)}(\boldsymbol{x}_p) - \nabla f_{c(\boldsymbol{x})}(\boldsymbol{x}_p)$,
8:   $\widetilde{\boldsymbol{w}}^i = \boldsymbol{g} * \boldsymbol{w}^i$,
9:   $\boldsymbol{r}^i = \dfrac{\boldsymbol{w}^{i\top}(\boldsymbol{x}_p^i - \boldsymbol{x}^i)}{\boldsymbol{w}^{i\top}\widetilde{\boldsymbol{w}}^i}\widetilde{\boldsymbol{w}}^i$,
10:   $\boldsymbol{r}^i \leftarrow \text{SmoothClip}(\boldsymbol{x}^i, \boldsymbol{r}^i, \boldsymbol{g})$,
11:   $\boldsymbol{x}^{i+1} \leftarrow \boldsymbol{x}^i + \boldsymbol{r}^i$,
12:   $i \leftarrow i + 1$,
13: **end while**
14: **return** $\boldsymbol{x}^i - \boldsymbol{x}^0$.

---

4. Experiments

4.1. Setup

We evaluate the performance of SmoothFool on three datasets including the test set of MNIST [25], the test set of CIFAR-10 [21], and 10,000 samples from the validation set of ILSVRC2012 [5] (10 images per each class). For the MNIST dataset, a two-layer fully-connected network (FC2) and a LeNet [24] architecture are used. For the CIFAR-10 dataset, we use a VGG-F [3] and ResNet-18 [14] architectures. For the ImageNet dataset, we consider VGG16 and ResNet-101. The accuracy of each model on benign samples is shown in Table 1. We set the step size ϵ of the SmoothClip to 1, 0.5, 0.1 for MNIST, CIFAR-10, and ImageNet respectively, which results in a fast and reasonable performance.

Defining smoothness. We define smoothness based on the Gaussian blur function since it is practical and the cut-off frequency can be easily changed by modifying the stan-
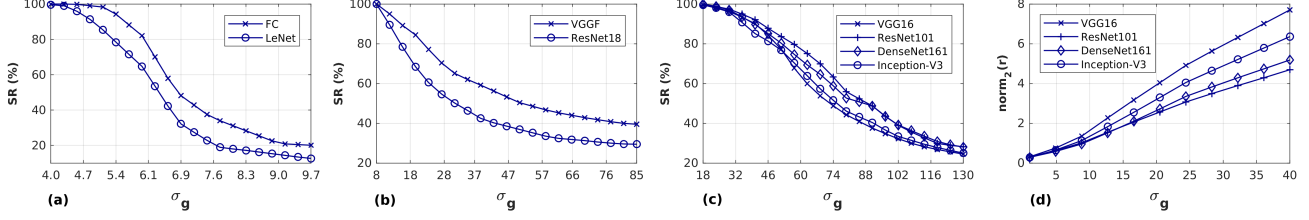
Figure 6: a, b, c) Fooling rate of the attack versus smoothing factor σ_g on MNIST, CIFAR-10 and ImageNet, respectively. d) Magnitude of perturbations vs. σ_g on ImageNet.

dard deviation. We assume h and g to be Gaussian blur filters with isotropic standard deviations σ_h and σ_g, respectively. In this setup, selecting any $\sigma_g > \sigma_h$ will minimize the roughness defined by h. Increasing σ_g improves the smoothness of APs but reduces the performance of the attack. To implement the Gaussian kernel, we set the kernel width to 5σ.

Comparisons. For $\sigma_g \ll 1$, the proposed approach converges to DeepFool [31]. Hence, we use it as a baseline to compare magnitude of the generated perturbations. For the second baseline, we develop an attack based on [37] by replacing the classical TV loss term with the roughness penalty $\Omega(r; h)$ from Equation 4 to provide a fair comparison framework as:

$$\arg\min_{r} \left(-J_c(f(x+r), y_x) + \lambda_s \Omega(r; \sigma_h) \right), \quad (12)$$

where J_c is the cross-entropy loss function, and y_x denotes the ground truth label of sample x. We refer to this method as the *iterative smooth* (IS) attack, and optimize it using gradient descent with a initial step size (learning rate) of 10^{-3}, and decay of 0.5 per each 100 iterations. We set λ_s to 0.1, 0.01 and 0.05 for MNIST, CIFAR-10, and ImageNet, respectively, which results in the most possible smooth perturbations for $\sigma_h = 1$. We consider this attack as the second baseline. We also compare the proposed method to the semantic adversarial examples given in [17], and refer to it as the *color-shift* (CS) attack, and consider it as the third baseline. We set the number of random trails of the CS algorithm to 100. Since CS adds perturbations in the HSV color space, we compute the average magnitude of perturbations for this attack in the HSV space to provide a fair comparison (the magnitude of APs in RGB color space is observed to be approximately 10 times greater).

Evaluation metrics. We measure the fooling rate of the attack and average smoothness of constructed APs. The fooling rate is defined on the set of correctly classified benign samples since it provides a more robust measure to evaluate the attack. For the sake of brevity of explaining results, we define $\sigma_g^{A\%}$ as the maximum value of σ_g (minimum among network architectures) that results in a $A\%$ fooling rate. In order to evaluate the smoothness of constructed APs, we measure the expected roughness $\bar{\Omega} = \mathbb{E}_{\mathscr{D}_s}[\Omega(r_x, h)]$,

Figure 7: Examples of extremely smooth adversarial perturbations computed for ResNet-18 and CIFAR-10 dataset with $\sigma_g = 200$.

where r_x is the AP constructed for x, and $\mathscr{D}_s$ is the set of successfully attacked samples. This measure is sensitive to the magnitude of perturbations. Thus, we develop a second measure by normalizing Ω over the total power of perturbations as: $\bar{\Omega}_n = \mathbb{E}_{\mathscr{D}_s}[\Omega(r_x, h)/||r_x||_2^2]$. Indeed, $\bar{\Omega}_n$ relatively measures how much of the total power of the APs is occupied by high-frequency components according to h.

4.2. General Performance

Figure 6 (a-c) shows the fooling rates of SmoothFool versus σ_g. We observe that the pair $(\sigma_g^{100\%}, \sigma_g^{20\%})$ for MNIST, CIFAR-10 and ImageNet is $(4.1, 7.8)$, $(8.4, 124.4)$ and $(19.3, 165.3)$, respectively. As expected, the fooling rate is highly dependent on the smoothing factor σ_g. However, the fooling rate remains high for significantly large (compared to the size of the input image) values of σ_g on ImageNet and CIFAR-10. For instance, $\sigma_g^{50\%}$ for CIFAR-10 is 32.8 which is approximately equal to the width of input images and shows that it is possible to fool the classifier on 50% of samples solely by adding a carefully selected constant value to all pixels of each color channel. The magnitude of smooth APs versus smoothness is depicted in Figure 6 (d). Increasing smoothness results in larger magnitudes of APs since the projection of $\widetilde{w}$ onto w will become smaller. However, smoothness of perturbations allows larger magnitudes since they are not as perceptible when compared to the noisy structure of contemporary APs.

We observe in Table 1 that SmoothFool with $\sigma_g = 2$ on all datasets, crafts significantly smoother (based on $\bar{\Omega}$ and $\bar{\Omega}_n$ with $\sigma_h = 1$) APs compared to the baseline attacks

Dataset	Network	Acc. (%)	F.Rate (%) IS	CS	$\bar{\Omega} \times 10^3$ @ $\sigma_h=1$ DF	IS	CS	SF	$\bar{\Omega}_n \times 10^3$ @ $\sigma_h=1$ DF	IS	CS	SF	$\mathbb{E}_x[||r_x||_2]$ @ σ_g DF	IS	CS	SF
MNIST	1-FC2	98.6	98.0	-	783	591	-	**334**	511	308	-	**63**	1.17	2.81	-	1.76
	2-LeNet	99.1	94.1	-	890	677	-	**352**	532	338	-	**68**	1.23	3.18	-	2.32
CIFAR-10	3-VGG-F	93.1	85.4	93.4	288	203	184	**114**	891	216	163	**57**	0.19	3.14	4.42	1.60
	4-ResNet-18	93.3	87.8	89.1	310	206	199	**127**	959	287	175	**65**	0.21	3.63	4.93	1.60
ImageNet	5-VGG16	71.5	57.8	91.1	111	89	104	**29**	871	358	238	**51**	0.25	4.90	7.71	0.58
	6-ResNet-101	77.3	62.6	92.7	108	83	95	**16**	827	300	207	**36**	0.28	4.82	7.56	0.55
	7-ResNet-152	78.3	60.1	91.9	116	91	88	**20**	845	391	229	**38**	0.29	5.12	8.16	0.57
	8-DenseNet161	77.6	66.0	90.3	107	79	86	**16**	795	309	212	**35**	0.28	3.47	5.58	0.57
	9-InceptionV3	77.4	65.5	90.6	149	85	92	**12**	658	293	223	**48**	0.32	4.50	8.10	0.35

Table 1: Comparing SmoothFool (SF) to DeepFool (DF) [31], iterative smooth (IS) [37, 7], and color-shift (CS) [17] attacks. To satisfy smoothness based on $\sigma_h=1$, σ_g is set to 2 for all datasets. Fooling rates of SF and DF are $>99.9\%$ on all datasets.

	VGG-F σ_g			ResNet-18 σ_g		
Class	20	60	100	20	60	100
airplane	75.0	29.5	25.0	67.3	34.7	28.2
automobile	**58.7**	**10.8**	**6.5**	**33.3**	**6.6**	**2.2**
bird	93.4	65.2	<u>52.1</u>	65.1	41.8	32.5
cat	<u>100</u>	58.3	43.7	65.3	21.1	17.3
deer	87.2	<u>60.0</u>	49.0	78.0	40.2	36.0
dog	78.7	48.9	40.4	75.1	<u>55.5</u>	<u>52.7</u>
frog	88.8	51.1	46.6	68.9	44.8	41.3
horse	74.5	29.0	23.6	70.3	25.9	22.2
ship	79.0	32.5	25.5	<u>81.4</u>	35.1	29.6
truck	73.9	32.6	21.7	68.5	31.4	22.8
all	83.8	45.8	37.6	67.2	33.0	27.9

Table 2: Per-class fooling rate (%) of SmoothFool for three values of σ_g on the CIFAR-10 dataset. Bold and underlined values show the fooling rate on classes with highest and lowest robustness against smooth APs, respectively.

	Fooling rate under defense (%)					
Defense	IGSM	DF	CS	SF_1	SF_2	SF_3
Adv.	32.6	15.6	64.5	58.6	70.7	**78.0**
PGD	21.0	12.3	61.4	57.2	67.3	**72.8**
Ens.	18.7	14.0	62.2	54.5	62.8	**73.6**
SAT	22.8	37.2	21.0	11.5	42.9	**53.4**
HGD	9.3	11.2	46.9	43.7	57.2	**66.2**
MagNet	10.7	8.9	25.1	46.4	**65.5**	52.6

Table 3: Evaluating attacks under different defense strategies on a ResNet-18 trained on CIFAR-10. SF_1, SF_2, and SF_3 denote the proposed algorithm with σ_g of 1, 3, and 5, respectively.

for the smoothness, while the magnitudes of APs are solely 1.8x larger than the state-of-the-art ℓ_2-minimal APs crafted by DeepFool. Figure 4 shows some examples of smooth APs computed for different levels of smoothness. We observe that each class responds differently as the smoothness of APs increases. Table 2 shows the per-class fooling rate of the attack on CIFAR-10. Smooth perturbations at $\sigma_g = 100$ fool the VGG-F classifier on more than 50% of samples of the *'bird'* class, while they are approximately not effective for the *'car'* class. This shows that some classes are severely sensitive to smooth perturbations while other exhibit lower sensitivity. The network architecture has a direct effect on this observation since the most sensitive class to smooth APs for each specific value of σ_g is different among network architectures.

Figure 7 demonstrates some examples of extremely smooth APs on CIFAR-10, showing a similar behavior (in RGB color space) as color-shifted adversarial examples [17]. However, as the method in [17] randomly shifts Hue and Saturation of benign samples, it often generates odd adversarial examples such as *'blue apples'* or *'red lemons'*

which are no longer adversarial examples since the conceptual evidence of objects is destroyed. However, since SmoothFool finds relatively small smooth perturbations, the whole concept of an object will not change drastically after the attack.

Performance under white-box defenses. Here, we evaluate the effectiveness of smooth perturbations against defense methods. First, we evaluate the attack under defenses based on adversarial training on FGSM (Adv.) [12], projectile gradient descent (PGD) [27] and ensemble (Ens.) [41] adversarial examples. We consider an additional defense of training on adversarial examples computed by the proposed SmoothFool with $\sigma_g = 1$, and refer to it as Smooth Adversarial Training (SAT). Second, we consider the high-level guided denoiser (HGD) [26] as a denoising based defense and MagNet [28] as a defense which evaluates adversarial examples using a learned distribution of natural samples. In all experiments, we assume that attacks have zero knowledge about the defense models.

Table 3 shows the performance of SmoothFool under defenses. Results suggest that increasing the smoothness of APs elevates the chance of bypassing defense methods. Such a characteristic had been observed before in adversarial examples constructed by spatial transformations [42, 4]. Smooth APs with $\sigma_g = 5$ successfully bypass HGD de-
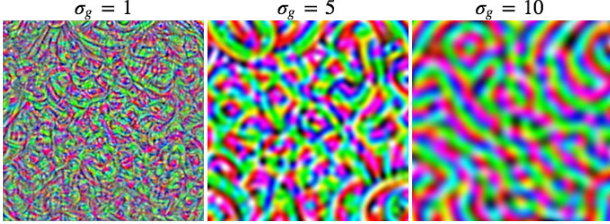
Figure 8: Smooth universal APs crafted for VGG16 architecture (best viewed in color).

Net.	Smoothing	Param.	VGG16	ResNet101	Inc-V3
VGG16	-	-	100	12.6	8.9
	Gaussian	$\sigma = 5$	100	15.8	13.6
		$\sigma = 10$	100	20.7	15.3
	Linear	$k = 25$	100	17.7	11.6
		$k = 50$	100	23.5	14.1
	Uniform	$k = 25$	100	16.8	12.0
		$k = 50$	100	21.8	14.6
ResNet101	-	-	15.2	100	15.0
	Gaussian	$\sigma = 5$	17.0	100	18.8
		$\sigma = 10$	19.9	100	22.5
	Linear	$k = 25$	19.8	100	16.9
		$k = 50$	22.8	100	19.7
	Uniform	$k = 25$	18.6	100	17.3
		$k = 50$	21.0	100	22.1

Table 4: Transferability of smooth perturbations for black-box attack. The size of Gaussian kernels is $k = 5\sigma$. Columns show source networks and attack parameters, and rows show the target models.

σ_g	VGG16	RNet101	RNet152	DNet161	Inc-V3
0	78.3	64.8	63.4	52.9	54.6
1	79.6	66.0	66.8	53.2	57.8
5	82.2	**69.9**	**70.3**	**57.6**	58.6
10	**84.5**	68.7	69.1	55.9	**61.6**

Table 5: Transferability of universal smooth APs computed for VGG16 accross data points and network architectures.

fense and defenses based on adversarial training on more than 60% of samples. Similarly, the CS attack shows significant robustness against all defenses except MagNet. A reasonable explanation is that although the CS attack generates relatively smooth APs compared to conventional attacks, changing the Hue and Saturation of images considerably pushes samples outside the distribution of natural samples leaned by MagNet. SmoothFool bypasses MagNet by a notable margin which indicates the closeness of generated samples to the distribution of natural images. However, for large values of σ_g, the magnitude of smooth APs takes large values, and thus, degrades the fooling rate of SmoothFool against MagNet defense. Furthermore, we observe that SAT defense provides a relative robustness against smooth APs constructed by $\sigma_g = 1$, but is susceptible to smoother perturbations. This suggest that the frequency components of APs can play a crucial role in bypassing adversarial training defenses trained on examples constructed by APs of different frequency components.

Black-box performance and ablation on smoothing functions. Here, we evaluate the black-box performance of smooth APs. Since our algorithm computes ℓ_2-minimal perturbations, we scale smooth APs to have the maximum ℓ_∞-norm of 16 for pixel values in range 255 based on the conventional setting for black-box attacks on ImageNet [6]. We consider two additional smooth functions including linear and uniform kernels to evaluate the effect of smoothing functions on fooling rates and transferability of APs. The uniform kernel of size k has all values equal to $\frac{1}{k^2}$. The linear kernel of size k has the maximum value of $\frac{4}{k^2}$ at the center and minimum value of zero at edges. Any other value is the linear interpolation of the maximum and minimum values.

Table 4 presents the results for this experiment. The fooling rate of attacks is 100% when the source and target models are the same. This suggests that the type of smoothing functions does not constrain the performance of APs. Hence, a broad range of smoothing functions can be deployed for generating smooth APs. Transferability of adversarial examples consistently improves as the smoothness of perturbations increases. This demonstrates that smoothness increases the transferability of adversarial examples for

black-box attacks which validates the results reported by Dong *et al.* [6]

Universal adversarial perturbations.

We integrate the proposed approach with the universal adversarial perturbations (UAP) [30] to explore the possibility of finding smooth UAPs. The implementation detail and the integrated algorithm is available in the Supplementary. We compute smooth UAPs for VGG16 and then evaluate thier transferability on four other networks including ResNet-101, ResNet-151, DenseNet-161, and Inception-V3.

Table 5 demonstrates the performance of smooth UAPs versus smoothness. Increasing smoothness enhances the transferability of APs across both the data points and network architectures. The transferability on 3 networks deteriorates for $\sigma_g > 5$. We attribute this observation to the theoretical fact that increasing smoothness also increases the magnitude of APs. Hence, with the same threshold for the maximum ℓ_∞-norm of smooth UAPs, there always exist a σ_g after which the transferability decreases.

5. Conclusion

In this study, we explored the vulnerability extent of DNNs to smooth adversarial perturbations by proposing SmoothFool, a framework for computing ℓ_2-minimal smooth APs. The methodology is developed based on a broad definition of smoothness and can be extended to pose any frequency-domain constraint on perturbations. Through

extensive experiments, we demonstrated that smooth adversarial perturbations are robust against two major group of defense strategies. Smoothness also improves the transferability of adversarial examples across network architectures and data points. Furthermore, our results suggest that class categories exhibit variable susceptibility to smooth perturbations which can help interpret the decision of DNNs for different categories.

References

[1] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[2] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[4] A. Dabouei, S. Soleymani, J. Dawson, and N. M. Nasrabadi. Fast geometrically-perturbed adversarial faces. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[6] Y. Dong, T. Pang, H. Su, and J. Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4312–4321, 2019.

[7] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1625–1634, 2018.

[8] A. Fawzi, S. M. Moosavi Dezfooli, and P. Frossard. The robustness of deep networks-a geometric perspective. *IEEE Signal Processing Magazine*, 34, 2017.

[9] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto. Empirical study of the topology and geometry of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] V. Fischer, M. C. Kumar, J. H. Metzen, and T. Brox. Adversarial examples for semantic image segmentation. *arXiv preprint arXiv:1703.01101*, 2017.

[11] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *2017 IEEE international conference on computer vision (ICCV)*, pages 3449–3457. IEEE, 2017.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[13] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] M. Hein and M. Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2266–2276, 2017.

[16] J. Hendrik Metzen, M. Chaithanya Kumar, T. Brox, and V. Fischer. Universal adversarial perturbations against semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2755–2764, 2017.

[17] H. Hosseini and R. Poovendran. Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1614–1619, 2018.

[18] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[19] S. T. Jan, J. Messou, Y.-C. Lin, J.-B. Huang, and G. Wang. Connecting the digital and physical world: Improving the robustness of adversarial attacks. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI'19)*, 2019.

[20] S. Jetley, N. Lord, and P. Torr. With friends like these, who needs adversaries? In *Advances in Neural Information Processing Systems (NIPS)*, pages 10772–10782, 2018.

[21] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[23] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[26] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[28] D. Meng and H. Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.

[29] A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard. Sparsefool: a few pixels make a big difference. *arXiv preprint arXiv:1811.02248*, 2018.

[30] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1765–1773, 2017.

[31] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[32] S.-M. Moosavi-Dezfooli, A. Shrivastava, and O. Tuzel. Divide, denoise, and defend against adversarial attacks. *arXiv preprint arXiv:1802.06806*, 2018.

[33] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[34] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8571–8580, 2018.

[35] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

[36] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations (ICLR)*, 2019.

[37] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[38] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[39] J. Su, D. V. Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*, 2017.

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

[41] F. Tramr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

[42] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

[43] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[44] Y. Zhang, K. Lee, and H. Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *International Conference on Machine Learning (ICML)*, pages 612–621, 2016.

6. Setup for smooth UAPs.

We use the setup in [30] and perform two modifications to its main algorithm to find smooth UAPs. First, we replace DeepFool [31] with the SmoothFool algorithm. Second, we replace the clipping method, based on projection operator, with SmoothClip. SmoothClip algorithm smoothly clips r to make sure $x+r$ resides in $[0, 1]$. However, for the smooth UAP v, SmoothClip should clip v to $[-\xi, \xi]$. Since the new bound is not dependent on x anymore, we replace x with an all-zero matrix, $\mathbf{0}$, of a same size as x. Furthermore, we normalize v before the clipping and denormalize it after the clipping. This allows us to utilize SmoothClip without modifying its algorithm. Smooth perturbation in each iteration is projected smoothly to the l_∞-ball of $\xi = 10/255$ around the original sample. The ϵ of SmoothClip is set to 0.1, and the universal fooling rate τ is set to 0.9. For computing smooth UAP, we exploit $m = 10,000$ benign samples from the validation set of ILSVRC2012 [5]. Algorithm 3 details the algorithm for finding smooth UAPs.

7. Further results

Figure 9 demonstrates further smooth UAPs computed for different network architectures. Figures 10-24 show several examples of generated smooth perturbations for images from the validation set of ILSVRC2012 and different network architectures. Each figure demonstrates the results for a fixed value of σ_g. Each row, from top to bottom, shows benign images, adversarial images, and the corresponding smooth APs, respectively. The predicted label for each image is overlaid on the image in red-colored text. The target network architecture and the smoothing factor σ_g for each figure is denoted in the caption.
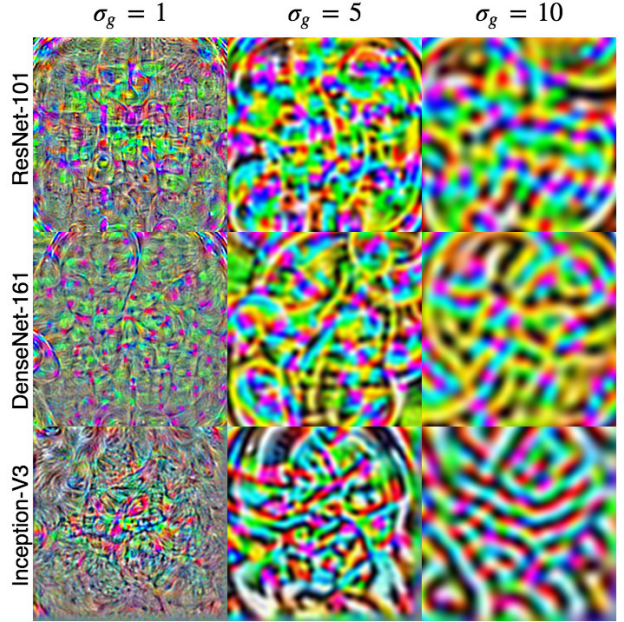

Figure 9: Smooth UAPs computed for three network architectures on ImageNet. Rows and columns show the results on different smoothing levels and network architectures, respectively.

Algorithm 3 Computation of smooth UAPs.

1: **input:** Data points $X = \{x_1, \ldots, x_m\}$, classifier f and the associated class predictor c, desired fooling rate τ, low-pass filter g, desired ℓ_∞-norm ξ.
2: **output:** Smooth UAP v.
3: Initialize $v \leftarrow 0$.
4: **while** $\frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{>0}(|c(x_i) - c(x_i + v)|) < \tau$ **do**
5:     **for** each $i \in \{1, \ldots, m\}$ **do**
6:         **if** $c(x_i + v) = c(x_i)$ **then**
7:             $r_i = \text{SmoothFool}(f, x_i, g)$
8:             $v \leftarrow v + r_i$
9:             $v \leftarrow (v/\xi + 1) * 0.5$
10:            $v \leftarrow \text{SmoothClip}(\mathbf{0}, v, g, \epsilon)$
11:            $v \leftarrow (2 * v - 1) * \xi$
12:         **end if**
13:     **end for**
14: **end while**
15: **return** v.

Figure 10: Network architecture: **VGG-16**, $\sigma_g = 10$.

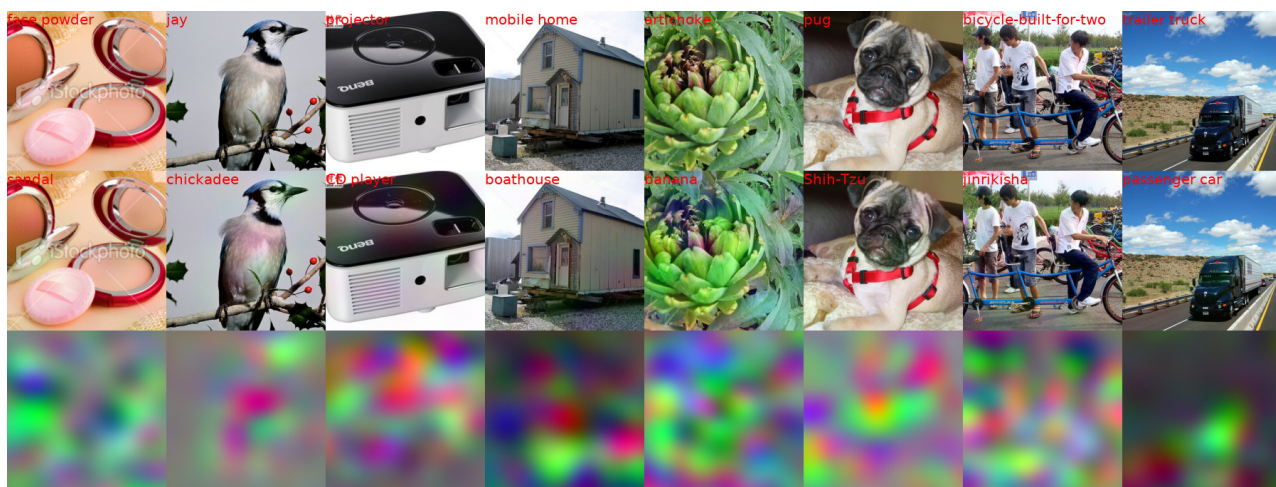
Figure 11: Network architecture: **VGG-16**, $\sigma_g = 20$.

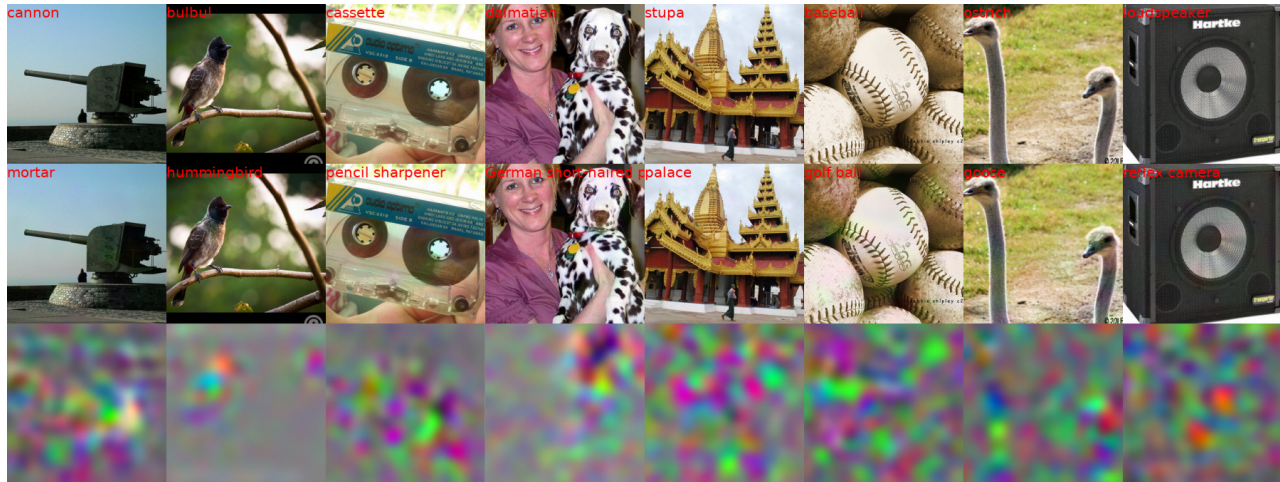Figure 12: Network architecture: **VGG-16**, $\sigma_g = 50$.

Figure 13: Network architecture: **ResNet-101**, $\sigma_g = 10$.

Figure 14: Network architecture: **ResNet-101**, $\sigma_g = 20$.

Figure 15: Network architecture: **ResNet-101**, $\sigma_g = 50$.
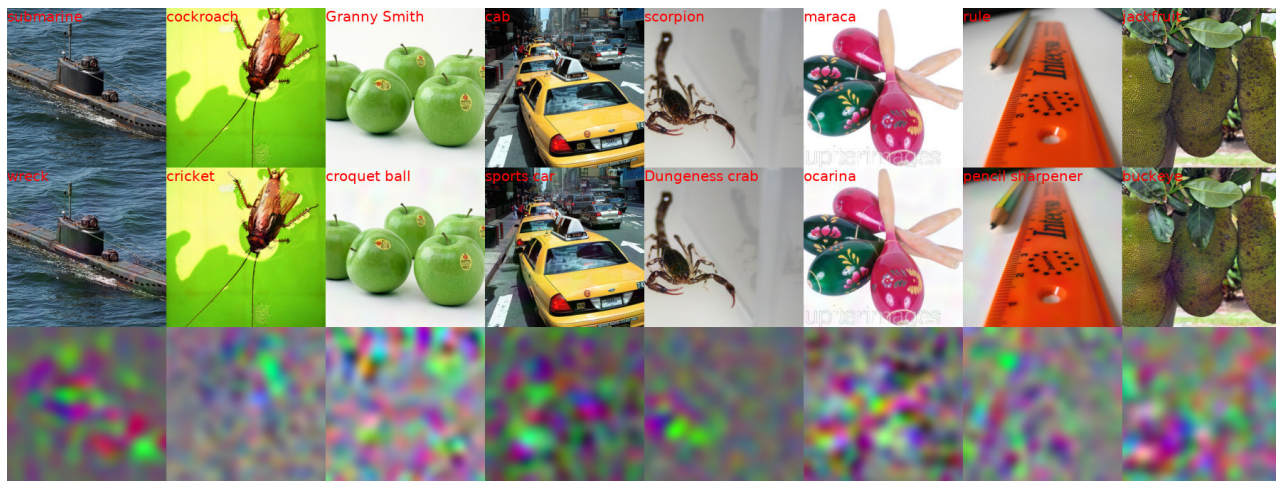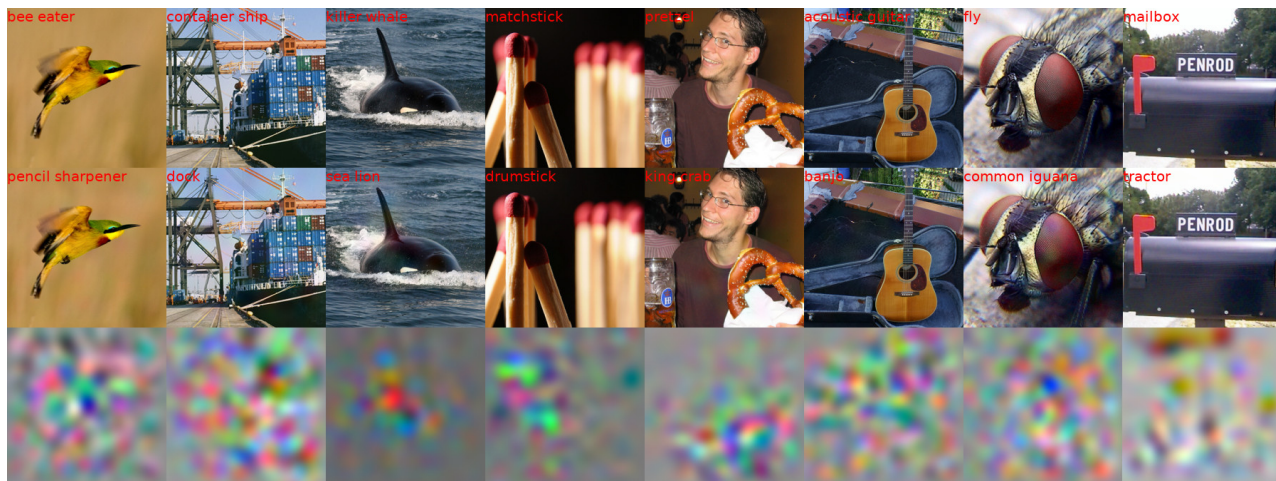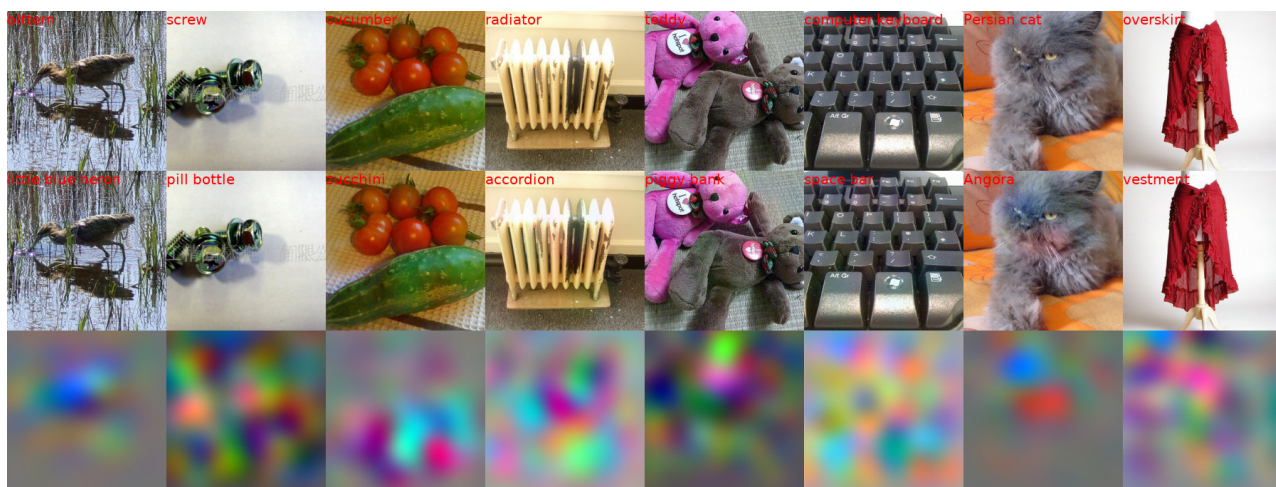
Figure 16: Network architecture: **DenseNet-161**, $\sigma_g = 10$.

Figure 17: Network architecture: **DenseNet-161**, $\sigma_g = 20$.

Figure 18: Network architecture: **DenseNet-161**, $\sigma_g = 50$.

Figure 19: Network architecture: **Inception-V3**, $\sigma_g = 10$.

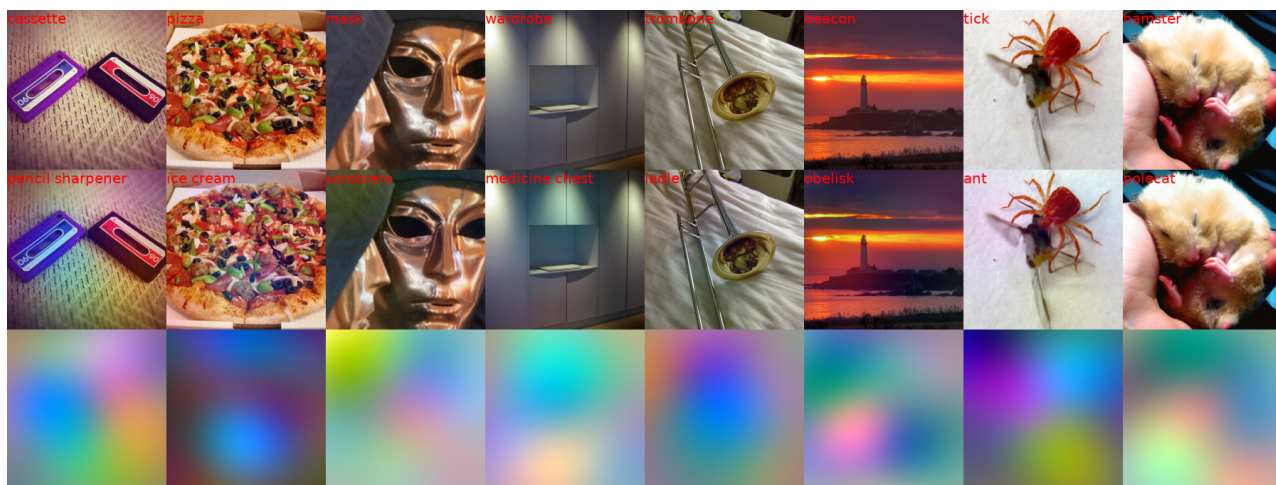Figure 20: Network architecture: **Inception-V3**, $\sigma_g = 20$.

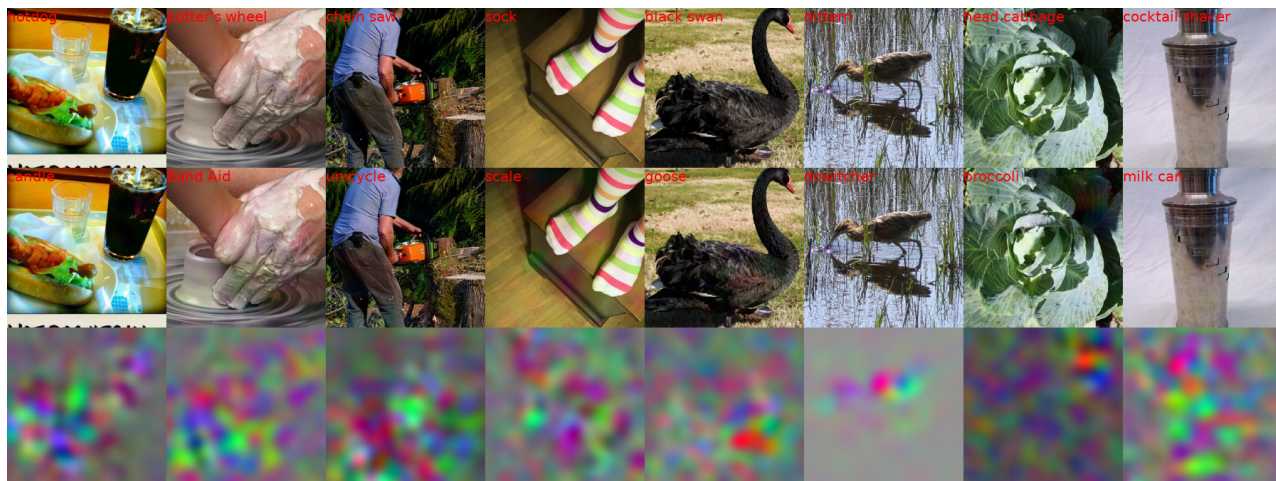Figure 21: Network architecture: **Inception-V3**, $\sigma_g = 50$.

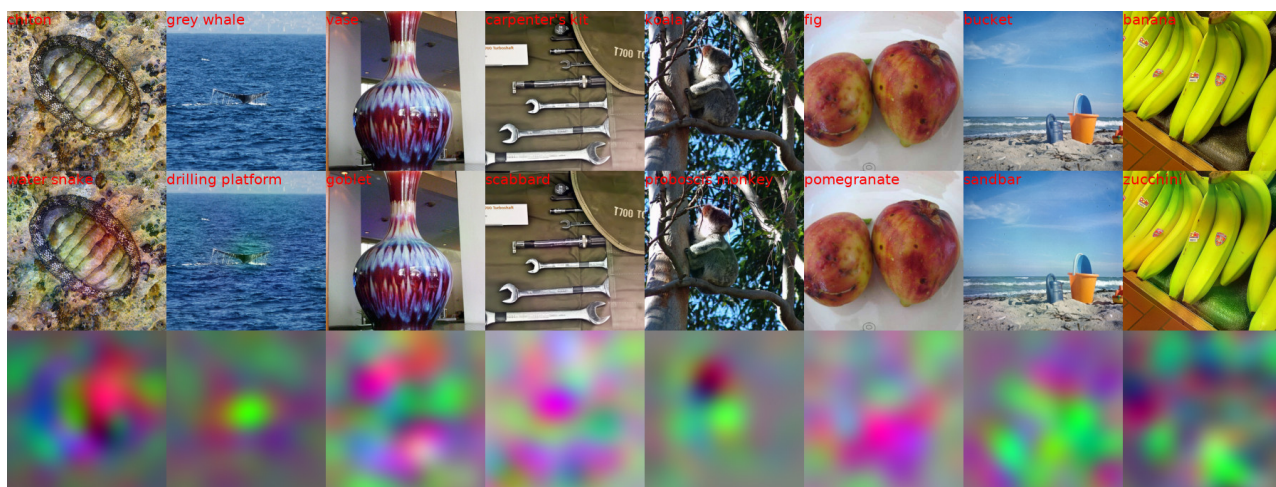Figure 22: Network architecture: **ResNet-152**, $\sigma_g = 10$.

Figure 23: Network architecture: **ResNet-152**, $\sigma_g = 20$.

Figure 24: Network architecture: **ResNet-152**, $\sigma_g = 50$.