# Adversarial Preprocessing: Understanding and Preventing Image-Scaling Attacks in Machine Learning


Erwin Quiring, David Klein, Daniel Arp, Martin Johns and Konrad Rieck

*Technische Universität Braunschweig, Germany*

## Abstract

Machine learning has made remarkable progress in the last years, yet its success has been overshadowed by different attacks that can thwart its correct operation. While a large body of research has studied attacks against learning algorithms, vulnerabilities in the preprocessing for machine learning have received little attention so far. An exception is the recent work of Xiao et al. that proposes attacks against image scaling. In contrast to prior work, these attacks are agnostic to the learning algorithm and thus impact the majority of learning-based approaches in computer vision. The mechanisms underlying the attacks, however, are not understood yet, and hence their root cause remains unknown.

In this paper, we provide the first in-depth analysis of image-scaling attacks. We theoretically analyze the attacks from the perspective of signal processing and identify their root cause as the interplay of downsampling and convolution. Based on this finding, we investigate three popular imaging libraries for machine learning (OpenCV, TensorFlow, and Pillow) and confirm the presence of this interplay in different scaling algorithms. As a remedy, we develop a novel defense against image-scaling attacks that prevents all possible attack variants. We empirically demonstrate the efficacy of this defense against non-adaptive and adaptive adversaries.

## 1 Introduction

Machine learning techniques have enabled impressive progress in several areas of computer science, such as in computer vision [e.g., 11, 12, 13] and natural language processing [e.g., 7, 18, 31]. This success, however, is increasingly foiled by attacks from adversarial machine learning that exploit weaknesses in learning algorithms and thwart their correct operation. Prominent examples of these attacks are methods for crafting adversarial examples [6, 32], backdooring neural networks [10, 15], and inferring properties from learning models [9, 27]. While these attacks have gained significant attention in research, they are unfortunately not the only weak spot in machine learning systems.

Recently, Xiao et al. [35] have demonstrated that data preprocessing used in machine learning can also suffer from vulnerabilities. In particular, they present a novel type of attack that targets *image scaling*. The attack enables an adversary to manipulate images, such that they change their appearance when scaled to a specific dimension. As a result, any learning-based system scaling images can be tricked into working on attacker-controlled data. As an example, Figure 1 shows an attack against the scaling operation of the popular TensorFlow library. The manipulated image (left) changes to the output (right) when scaled to a specific dimension.

Attacks on image scaling pose a threat to the security of machine learning: First, scaling is omnipresent in computer vision, as learning algorithms typically require fixed input dimensions. Second, these attacks are agnostic to the learning model, features, and training data. Third, the attacks can be used for poisoning data during training as well as misleading classifiers during prediction. In contrast to adversarial examples, image-scaling attacks do not depend on a particular model or feature set, as the downscaling can create a perfect image of the target class. As a consequence, there is a need for effective defenses against image-scaling attacks. The underlying mechanisms, however, are not understood so far and the root cause for adversarial scaling is still unknown.

In this paper, we provide the first comprehensive analysis of image-scaling attacks. To this end, we theoretically analyze the attacks from the perspective of signal processing and


Figure 1: Example of an image-scaling attack. Left: a manipulated image showing a cat. The scaling operation produces the right image with a dog.

identify the root cause of the attacks as the interplay of downsampling and convolution during scaling. That is, depending on the downsampling frequency and the convolution kernel used for smoothing, only very specific pixels are considered for generating the scaled image. This limited processing of the source image allows the adversary to take over control of the scaling process by manipulating only a few pixels. To validate this finding, we investigate three popular imaging libraries for machine learning (OpenCV, TensorFlow, and Pillow) and confirm the presence of this insecure interplay in different scaling algorithms.

Based on our theoretical analysis, we develop defenses for fending off image-scaling attacks in practice. As a first step, we analyze the robustness of scaling algorithms in the three imaging libraries and identify those algorithms that already provide moderate protection from attacks. In the second step, we devise a new defense that is capable of protecting from all possible attack variants. The defense sanitizes explicitly those pixels of an image that are processed by a scaling algorithm. As a result, the adversary loses control of the scaled content, while the quality of the source image is largely preserved. We demonstrate the efficacy of this strategy in an empirical evaluation, where we prevent attacks from non-adaptive as well as adaptive adversaries.

Finally, our work provides an interesting insight into research on secure machine learning: While attacks against learning algorithms are still hard to analyze due to the complexity of learning models, the well-defined structure of scaling algorithms enables us to fully analyze scaling attacks and develop effective defenses. As a consequence, we are optimistic that attacks against other forms of data preprocessing can also be prevented, given a thorough root-cause analysis.

**Contributions.** In summary, we make the following contributions in this paper:

- *Analysis of image-scaling attacks.* We conduct the first in-depth analysis of image-scaling attacks and identify the vulnerability underlying the attacks in theory as well as in practical implementations.
- *Effective Defenses.* We develop a theoretical basis for assessing the robustness of scaling algorithms and designing effective defenses. We propose a novel defense that protects from all possible attack variants.
- *Comprehensive Evaluation.* We empirically analyze scaling algorithms of popular imaging libraries under attack and demonstrate the effectivity of our defense against adversaries of different strengths.

The rest of this paper is organized as follows: We review the background of image scaling and attacks in Section 2. Our theoretical analysis is presented in Section 3, and we develop defenses in Section 4. An empirical evaluation of attacks and defenses is given in Section 5. We discuss related work in Section 6, and Section 7 concludes the paper.

Table 1: Scaling algorithms in deep learning frameworks.

| Framework | Caffe | PyTorch | TensorFlow |
|---|---|---|---|
| Library | OpenCV | Pillow | tf.image |
| Library Version | 4.1 | 6.0 | 1.14 |
| Nearest | • | •(‡) | • |
| Bilinear | •(*) | •(*) | •(*) |
| Bicubic | • | • | • |
| Lanczos | • | • | |
| Area | • | • | • |

(*) Default algorithm. (‡) Default algorithm if Pillow is used directly without PyTorch.

## 2 Background

Before starting our theoretical analysis, we briefly review the background of image scaling in machine learning and then present image-scaling attacks.

### 2.1 Image Scaling in Machine Learning

Image scaling is a standard procedure in computer vision and a common preprocessing step in machine learning [21]. A scaling algorithm takes a source image $S$ and resizes it to a scaled version $D$. As many learning algorithms require a fixed-size input, scaling is a mandatory step in most learning-based systems operating on images. For instance, deep neural networks for object recognition, such as VGG19 and Inception V3/V4 expect inputs of $224 \times 224$ and $299 \times 299$ pixels, respectively, and can only be applied in practice if images are scaled to these dimensions.

Generally, we can differentiate *upscaling* and *downscaling*, where the first operation enlarges an image by extrapolation, while the latter reduces it through interpolation. In practice, images are typically larger than the input dimension of learning models and thus image-scaling attacks focus on downscaling. Table 1 lists the most common scaling algorithms. Although these algorithms address the same task, they differ in how the content of the source $S$ is weighted and smoothed to form the scaled version $D$. For example, nearest-neighbor scaling simply copies pixels from a grid of the source to the destination, while bicubic scaling interpolates pixels using a cubic function. We examine these algorithms in more detail in Section 3 when analyzing the root cause of scaling attacks.

Due to the central role in computer vision, scaling algorithms are an inherent part of several deep learning frameworks. For example, Caffe, PyTorch, and TensorFlow implement all common algorithms, as shown in Table 1. Technically, TensorFlow uses its own implementation called *tf.image*, whereas Caffe and PyTorch use the imaging libraries *OpenCV* and *Pillow*, respectively. Other libraries for deep learning either build on these frameworks or use the imaging libraries directly. For instance, Keras uses Pillow and DeepLearning4j builds on OpenCV. As a consequence, we focus our analysis on these major imaging libraries.

## 2.2 Image-Scaling Attacks

Recently, Xiao et al. [35] have shown that scaling algorithms are vulnerable to attacks and can be misused to fool machine learning systems. The proposed attack carefully manipulates an image, such that it changes its appearance when scaled to a specific dimension. In particular, the attack generates an image $A$ by slightly perturbing the source image $S$, such that its scaled version matches a target image $T$. This process is illustrated in Figure 2, which also serves as a running example throughout this paper. In addition, Table 2 provides an overview of our notation.

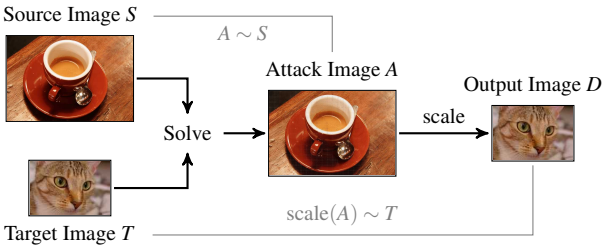


Figure 2: Principle of image-scaling attacks: An adversary computes $A$ such that it looks like $S$ but downscales to $T$.

### 2.2.1 Capabilities and Knowledge

The attack is agnostic to the employed learning model and does not require knowledge of the training data or extracted features. Yet, the adversary needs to know two parameters: (a) the used scaling algorithm and (b) the target size $m' \times n'$ of the scaling operation. Xiao et al. describe how an adversary can easily deduce both parameters with black-box access to the machine learning system by sending specifically crafted images [see 35]. Moreover, Table 1 shows that common open-source libraries have a limited number of scaling options, and thus only a few attempts are necessary to discover the correct setup. In some settings, a fixed algorithm can be even enforced by specific image sizes, as we show in Appendix A.

### 2.2.2 Attack Scope

As the image is manipulated before any feature extraction, image-scaling attacks can effectively mislead all subsequent steps in a machine-learning pipeline, allowing different attacks during train and test time. That is, an attacker can conceal data poisoning attacks [see 24]. For instance, she can modify the training data such that a backdoor pattern becomes present in the downscaled image which was not visible in the unscaled training image before.

Furthermore, she can trigger false predictions during the application of a learning model by creating a downscaled image of another, targeted class. Compared to adversarial examples [32], both attacks accomplish the same goal. However, image-scaling attacks considerably differ in the threat model: The attacks are model-independent and do not depend on knowledge of the learning model, features, or training data. Furthermore, image-scaling attacks are effective even if neural networks were robust against adversarial examples, as the downscaling can create a perfect image of the target class. Finally, we note that these attacks are of particular concern in all security-related applications where images are processed.

Table 2: Table of symbols for scaling attacks.

| Symbol | Size | Description |
|---|---|---|
| $S$ | $m \times n$ | The source image that is used to create the attack image. |
| $T$ | $m' \times n'$ | The target image that the adversary wants to obtain after scaling. |
| $A$ | $m \times n$ | The attack image, a slightly perturbed version of $S$ |
| $D$ | $m' \times n'$ | The output image of the scaling function scale. |

### 2.2.3 Attack Strategy

There exist a strong and a weak strategy for implementing image-scaling attacks. In the strong strategy, the adversary can choose the source and target image. In the weak version, the adversary can only choose the target, and the calculated attack image is meaningless and easily detectable. We thus focus on the stronger attack strategy in our paper, which is of particular concern in real-world applications.

**Objectives.** Formally, image-scaling attacks need to pursue the following two objectives:

***(O1)*** The downscaling operation on $A$ needs to produce the target image: $\text{scale}(A) \sim T$.

***(O2)*** The attack image $A$ needs to be indistinguishable from the source image: $A \sim S$.

The first objective ensures that the target image $T$ is obtained during scaling, while the second objective aims at making the attack hard to detect. We verify objective O1 by checking if the prediction of a neural network corresponds to the target image's class. Note that without the second objective, the attack would be trivial, as the adversary could simply overwrite $S$ with $T$. In this case, however, the attack would be easily detectable and thus not effective in practice.

**Strong Attack Strategy.** The adversary seeks a minimal perturbation $\Delta$ of $S$, such that the downscaling of $\Delta + S = A$ produces an output similar to $T$. Both goals can be summarized as the following optimization problem:

$$\begin{aligned} & \min(\|\Delta\|_2^2) \\ \text{s.t.} \quad & \|\text{scale}(S+\Delta) - T\|_\infty \leqslant \varepsilon\,. \end{aligned} \tag{1}$$

Additionally, each pixel value of $A$ needs to remain within the fixed range (e.g., $[0, 255]$ for 8-bit images). This problem can be solved with Quadratic Programming [5]. When successful, the adversary obtains an image $A$ that looks like the source but matches the target after scaling.

**Horizontal and Vertical Optimization.** Common imaging libraries, such as OpenCV or Pillow, implement downscaling by first resizing images horizontally and then vertically. This implementation technique enables approximating the scaling operation from Eq. (1) by a closed-form expression which is based on a simple matrix multiplication:

$$D = \text{scale}(S + \Delta) = L \cdot (S + \Delta) \cdot R \tag{2}$$

with $L \in \mathbb{R}^{m' \times m}$, $R \in \mathbb{R}^{n \times n'}$ and $D \in \mathbb{R}^{m' \times n'}$. The matrices $L$ and $R$ contain fixed coefficients that depend on the selected scaling algorithm. Both matrices can be computed in advance and are reusable. We refer to Xiao et al. [35] for a description how to calculate $L$ and $R$.

Based on this matrix multiplication, the attack can also be decomposed into a horizontal and vertical manipulation, which are conducted in reverse order to the scaling, as shown in Figure 3. The attack proceeds by first computing a resized version of $S$, that is, $S' = \text{scale}(S) \in \mathbb{R}^{m \times n'}$. Here, we solve Eq. (1) with $S'$ as source image and $T$ as target. Due to the decomposition, we only need the coefficient matrix $L$ and thus arrive at the following optimization problem

$$\min(\|\Delta'\|_2^2) \text{ s.t. } \|L \cdot (S' + \Delta') - T\|_\infty \leqslant \varepsilon \,. \tag{3}$$

Next, the horizontal direction is considered. To this end, the adversary calculates the final attack image $A$ with $S$ as source image, but $A'$ as target, analogue to Eq. (3).

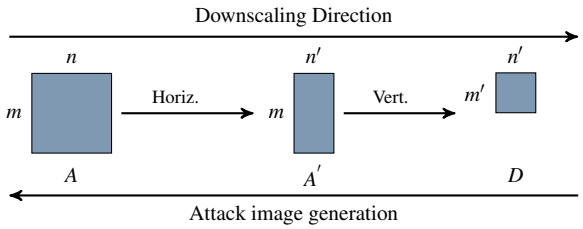


Figure 3: Libraries resize an image horizontally first, and then vertically. The attack creates $A$ in reverse order: first the intermediate image $A'$, and then $A$.

**Column-based Optimization.** In order to further decrease the computational effort, the optimization can be further decomposed into individual dimensions. We start again with the vertical scaling direction where we resize $S' \in \mathbb{R}^{m \times n'}$ to $D \in \mathbb{R}^{m' \times n'}$. Instead of considering the whole matrix, we solve the problem from Eq. (3) for each column of $S'$ separately:

$$\min(\|\Delta'_{*,j}\|_2^2) \text{ s.t. } \|L \cdot (S'_{*,j} + \Delta'_{*,j}) - T_{*,j}\|_\infty \leqslant \varepsilon \,, \tag{4}$$

where the subscript in $X_{*,j}$ specifies the $j$-th matrix column of a matrix $X$. This optimization is repeated for the horizontal direction and finally computed for all color channels.

# 3 Attack Analysis

After introducing the background of image-scaling attacks, we are ready to investigate their inner workings in more detail. Our aim is to find out which vulnerability image-scaling attacks exactly exploit to be successful. We start off by observing that the presented attacks must exploit a vulnerability that is shared by many scaling algorithms. As the implementations of the algorithms differ, this vulnerability needs to be linked to the general concept of scaling. To better grasp this concept, we require a broader perspective on image scaling and thus examine it from the viewpoint of signal processing.

## 3.1 Scaling as Signal Processing

Images can be viewed as a generic signal, similar to audio and video. While audio is described by a one-dimensional time series, an image represents a discrete and two-dimensional signal. Typically, images are encoded in the *spatial domain* of pixels. However, any signal can be described by a sum of sinusoids of different frequencies, and hence images can also be represented in the *frequency domain* [e.g., 19, 29].

Scaling reduces the dimension of an image. As a result, the frequency mixture of the image changes and higher frequencies are lost. This process is closely related to *downsampling* in signal processing, where a high-frequency signal is transformed to a lower frequency. A major problem of downsampling is that the reduced resolution might not be able to describe all relevant frequencies in the image. According to the Nyquist–Shannon theorem [19], it is only feasible to reconstruct a signal $s(t)$ from a discrete number of sampled points, if the sampling rate $f_T$ is at least twice as high as the highest frequency $f_{max}$ in the signal: $f_T \geq 2 \cdot f_{max}$.

If the frequency $f_T$ is below that threshold, the signal cannot be unambiguously reconstructed. In this case, the sampled points do not provide enough information to distinguish between the original signal and other possible signals. Figure 4 shows an example of this phenomenon, where it is impossible to decide which one of the two signals $s(t)$ and $\hat{s}(t)$ is described by the sampled points. Ultimately, the reconstructed signal can differ significantly from the original signal, which is known as the *aliasing effect* [19]. As we see in the next sections, image-scaling attacks build on this very effect by cleverly manipulating a signal, such that its downsampled version becomes a new signal.

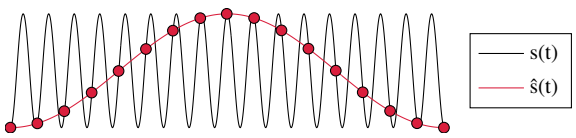


Figure 4: An example of an undersampled signal $s(t)$. Based on the sampling points, it is not possible to distinguish between $s(t)$ and $\hat{s}(t)$.

## 3.2 Scaling and Convolution

It is clear that scaling algorithms do not merely reduce the frequencies in an image. These algorithms carefully interpolate the pixels of the source image before downscaling it in order to mitigate the aliasing effect. This computation can be described as a *convolution* between the source signal and a kernel function [19]. For each position in the scaled image, the kernel combines a set of pixels (samples) from the source using a specific weighting. All scaling algorithms given in Table 1 can be expressed using this concept.

Without loss of generality, we focus on the horizontal scaling of a single row in the following, that is, a row $s \in \mathbb{R}^n$ from the source image is scaled to $d \in \mathbb{R}^{n'}$. We denote by $\beta$ the respective scaling ratio: $\beta = n/n'$. The goal of downscaling is to determine the value for each pixel in $d$ from a set of samples from $s$. This process can be described using a kernel function $w$ as follows

$$(s \star w)(t) = \sum_{u=-\infty}^{\infty} w(t-u)\, s(u). \tag{5}$$

Intuitively, $w$ represents a weighting function that is moved over $s$ as a sliding window. We denote the size of this window as the *kernel width* $\sigma$. Each pixel within this window is multiplied by the respective weight at this position. Figure 5 exemplifies this process for a bilinear kernel with $\sigma = 2$. The first pixel in $d$ is the aggregated result from the third and fourth pixel in $s$, while the second pixel in $d$ is only estimated from the seventh pixel in $s$.

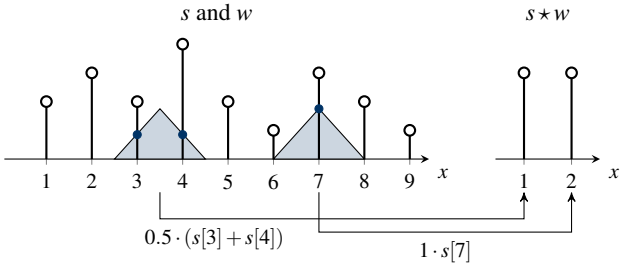


Figure 5: Scaling with convolution. The triangle illustrates the kernel with its relative weighting. It has a width of 2 and is shifted by a step size of $\beta$.

As the downscaling of an image produces a smaller number of pixels, the window of the kernel function needs to be shifted on $s$ by a specific step size, similar to the process of sampling in signal processing. The scaling ratio defines this step size so that each sampling position is given by

$$g(p) = p \cdot \beta, \tag{6}$$

where $p$ is the target pixel in $d$ and $g(p)$ a position in $s$ around which we place the kernel window. Note that the position $g(p)$ is not necessarily discrete and can also fall between two pixels, as shown in Figure 5. The downscaled output image is then computed as follows:

$$d_p = (s \star w)(g(p)) \quad p = 0, 1, \ldots, n'. \tag{7}$$

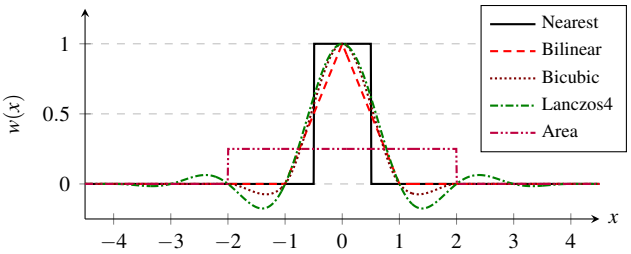


Figure 6: Visualization of kernel functions using in scaling algorithms.

Each scaling algorithm is defined by a particular kernel function. Figure 6 depicts the standard kernels for common scaling algorithms. For instance, nearest-neighbor scaling builds on the following kernel function:

$$w(x) = \begin{cases} 1 & \text{for } -0.5 \leqslant x < 0.5, \\ 0 & \text{otherwise .} \end{cases} \tag{8}$$

Only the value that is the closest to $g(p)$ is used by this scaling algorithm. In other words, nearest-neighbor scaling simply copies pixels from $s$ on a discrete grid to $d$. Overall, each kernel differs in the number of pixels that it uses and the respective weighting of the considered pixels.

## 3.3 Root-Cause Analysis

Based on our insights from signal processing, we can start to investigate the root cause of image-scaling attacks. We observe that not all pixels in the source image equally contribute to its scaled version. Only those pixels close to the center of the kernel receive a high weighting, whereas all remaining pixels play a limited role during scaling. If the step size exceeds the kernel width, some pixels are even ignored and irrelevant for the scaling operation. Figure 5 illustrates this situation: Only three out of nine pixels are considered for computing the scaled output.

This imbalanced influence of the source pixels provides a perfect ground for image-scaling attacks. The adversary only needs to modify those pixels with high weights to control the scaling and can leave the rest of the image untouched. This strategy is sufficient for achieving both objectives of the attack: (O1) a modification of pixels with high weights yields $\text{scale}(A) \sim T$, and (O2) depending on the sparsity of those pixels the attack image $A$ visually matches the source image $S$.

From the perspective of signal processing, image-scaling attacks can thus be interpreted as targeted aliasing, where the adversary selectively manipulates those regions of the signal that are sampled during downscaling. These regions create a high-frequency signal in the source image that is not visible in the spatial domain but precisely captures the sampling rate of the downscaling process.

We can deduce that the success of image-scaling attacks depends on the sparsity of pixels with high weight. If these

pixels are dense, the adversary may still achieve objective O1 but will fail to satisfy O2, as the attack becomes visible. Reviewing the general concept of scaling, we identify two factors that determine the sparsity of these pixels: the scaling ratio $\beta$ and the kernel width $\sigma$. For images, we formally bound the ratio $r$ of pixels that are considered during scaling by

$$r \leq (\beta_h \, \beta_v)^{-1} \, (\sigma_h \, \sigma_v). \tag{9}$$

The terms $\beta_h$, $\beta_v$ as well as $\sigma_h$ and $\sigma_v$ denote the respective scaling ratio and kernel width horizontally and vertically. If the direction is irrelevant, we consider quadratic images for our analysis and use $\beta$ and $\sigma$ for both axis. Moreover, note that the right term may exceed one if the windows of the kernels overlap and pixels in the source are considered multiple times.

**Scaling ratio.** The larger the ratio $\beta$, the fewer pixels are considered during scaling if the kernel width is bounded. In particular, the number of pixels that are discarded growths quadratically with $\beta$. An adversary can thus easily control the ratio $r$ by increasing the size of the source image.

Figure 7(a)-(c) show the influence of the scaling ratio on the attack for a kernel with $\sigma = 1$. All images fulfill objective O1, that is, the images are scaled down to the "cat" image. Depending on the scaling ratio, however, their success to objective O2 changes. For a large ratio of $\beta = 4$, the attack image looks like the source, and the cat is not visible. For a smaller scaling ratio, the manipulated image becomes a mix of the source and target. For $\beta = 1$, the attack obviously fails.

**Kernel width.** The smaller the kernel width $\sigma$, the fewer pixels are considered during each convolution. While $\sigma$ is typically not controlled by the adversary, several implementations of scaling algorithms make use of very small constants for this parameter. For example, the nearest-neighbor, bilinear, and bicubic kernels of the TensorFlow framework have a width of 1, 2, and 4, respectively.

Figure 7(d)-(f) depict the influence of the kernel width on the attack for a fixed scaling ratio of $\beta = 4$. Again, all images fulfill objective O1 and are scaled down to the "cat" image. For $\sigma = 1$, the attack also satisfies objective O2 and is invisible. If two pixels are considered by the kernel, however, the cat becomes visible. For $\sigma = 4$, all pixels need to be manipulated and the attack fails.

Interestingly, our analysis is not limited to the scaling algorithms considered in this work. Any algorithm is vulnerable to image-scaling attacks if the ratio $r$ of pixels with high weight is small enough. Our analysis thus allows developers to check quickly if their algorithms are vulnerable to these attacks. Overall, we are thus the first to provide a general understanding of this attack type in practice. This understanding enables us to compare different scaling algorithms and ultimately develop effective defense strategies.

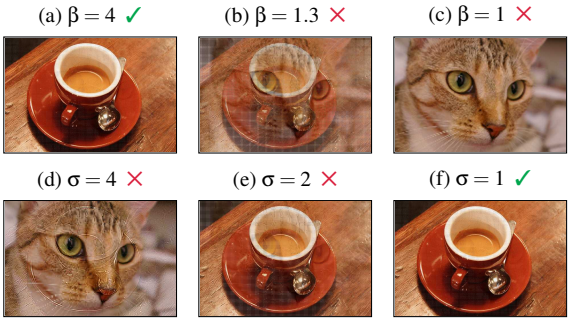


Figure 7: Influence of the scaling ratio and kernel size (see Figure 2 for the setting of this example); $\beta$ and $\sigma$ are the same horizontally and vertically. Plot (a)–(c) show manipulated images under varying ratios. Plot (d)–(f) show manipulated images under varying kernel sizes. The symbols ✓and ✗ indicate if the attack is successful.

## 4 Defenses

We continue with the development of defenses that build on our analysis and address the root cause of image-scaling attacks—rather than fixing their symptoms. Our defenses aim to *prevent* attacks without interfering with the typical workflow of deep learning frameworks. They can thus serve as a plug-in for existing scaling algorithms. Note that the mere *detection* of attacks is not sufficient here, as the systems would need to cope with rejected inputs.

Consequently, we first derive requirements for secure scaling and use these to validate the robustness of existing algorithms (Defense 1). As only a few algorithms realize a secure scaling, we proceed to develop a generic defense that reconstructs the source image and thereby is applicable to any scaling algorithm as preprocessing (Defense 2).

### 4.1 Attacker Model

For the construction and evaluation of our defenses, we consider two types of adversaries: a *non-adaptive* adversary who uses existing image-scaling attacks, and an *adaptive* adversary who is aware of our defense and adapts the attack strategy accordingly. Both adversaries have full knowledge of the scaling algorithm and the target size. In the adaptive scenario, the adversary additionally has full knowledge of the applied defense. Finally, we expect the adversary to freely choose the source and target image so that she can find the best match for conducting attacks in a given setup.

We note that these assumptions are realistic due to the open-source nature of deep learning frameworks and the use of several well-known learning models in practice, such as VGG19 and Inception V3/V4. With black-box access to the scaling and learning models, an adversary can even deduce the scaling algorithm and target size by sending a series of specially crafted images to the learning system [see 35].

## 4.2 Defense 1: Robust Scaling Algorithms

Let us start with the conception of an ideal robust scaling algorithm which serves as a prototype for analyzing the properties of existing algorithms.

**An ideal scaling algorithm.** In the ideal case, an algorithm investigates each pixel of the source image at least once for downscaling. The robustness of the scaling increases further if the employed convolution kernels overlap, and thus one pixel of the source contributes to multiple pixels of the scaled version. Technically, this requirement can be realized by *dynamically* adapting the kernel width $\sigma$ to the scaling ratio $\beta$, such that $\sigma \geq \beta$ holds. That is, the larger the ratio between the source and the scaled image, the wider the convolution kernel needs to become to cover all pixels of the image.

In addition to processing all pixels, an ideal algorithm also needs to weight all pixels equally; otherwise, a kernel with small support would leave pixels untouched if their weights become zero. For example, pixels close to the edge of the convolution window typically receive a very low weighting, as shown in Figure 6. As a result, the convolution of an ideal algorithm should be *uniform* and combine all pixels in the current kernel window with equal weight.

Although both properties—considering all pixels and a uniform convolution—can be technically implemented, they introduce challenges that can limit their practical utility: First, processing all pixels of an image slows down the scaling process. This is not necessarily a problem in applications where large neural networks are trained, and the overhead of scaling is minimal anyway. However, in real-time settings, it might be prohibitive to go over all pixels during scaling. Second, the flattened weighting of the convolution can blur the image content and remove structure necessary for recognizing objects. As a consequence, we identify a trade-off between security and performance in image scaling.

**Existing scaling algorithms.** Based on the concept of an ideal algorithm, we examine the source code of the three considered imaging libraries and analyze their scaling algorithms with respect to the processed pixels and the employed convolution kernels. In particular, we inspect the source code of OpenCV version 4.1, Pillow 6.0, and tf.image 1.14 from TensorFlow. Table 3 shows the results of this investigation.

Table 3: Kernel width $\sigma$ for the scaling algorithms implemented by the imaging libraries OpenCV, tf.image (TensorFlow) and Pillow.

| Library | OpenCV | TF | Pillow |
|---|---|---|---|
| Nearest | 1 | 1 | 1 |
| Bilinear | 2 | 2 | $2 \cdot \beta$ |
| Bicubic | 4 | 4 | $4 \cdot \beta$ |
| Lanczos | 8 | — | $6 \cdot \beta$ |
| Area | $\beta$ | $\beta$ | $\beta$ |

We observe that several scaling algorithms are implemented with fixed-size convolution kernels. For example, OpenCV and TensorFlow implement nearest-neighbor, bilinear, and bicubic scaling with a kernel width of 1, 2, and 4, respectively. Consequently, these algorithms become vulnerable once the scaling ratio exceeds the kernel width, and pixels of the source image are omitted during scaling.

Fortunately, however, we also identify one algorithm that is implemented with a dynamic kernel width of $\beta$ in all frameworks: *area scaling*. This algorithm scales an image by simply computing the average of all pixels under the kernel window, which corresponds to a uniform convolution, as shown in Figure 6 for $\beta = 4$. Moreover, area scaling corresponds to a low-pass filter which mitigates the aliasing effect. As a result, area scaling provides strong protection from image-scaling attacks, and the algorithm is a reasonable defense if the uniform weighting of the convolution does not impact later analysis steps. We demonstrate the robustness of area scaling in our empirical evaluation in Section 5.

Our analysis provides another interesting finding: Pillow stands out from the other imaging library, as it implements a dynamic kernel width for all algorithms except for nearest-neighbor scaling. The dynamic kernel width $\sigma$ is chosen such that the convolution windows substantially overlap, for example, for bicubic and Lanczos scaling by a factor of 4 and 6, respectively. Although the used convolutions are not uniform for these algorithms, this overlap creates a notable obstruction for the attacker, as dependencies between the overlapping windows need to be compensated. Figure 8 schematically shows the dynamic kernel width of Pillow in comparison to the implementations of OpenCV and TensorFlow.

**Disadvantages.** While area scaling and the Pillow library provide a means for robust scaling, they also induce drawbacks. As exemplified in Figure 9, the algorithms cannot entirely remove all traces from the attacks. Small artifacts can remain, as the manipulated pixels are not cleansed and still contribute to the scaling, though with limited impact. Our evaluation shows that these remnants are not enough to fool the neural network anymore. The predicted class for the scaled images, however, is not always correct due to the noise of the attack remainings. As a remedy, we develop an alternative defense in the next section that reconstructs the source image and thus is applicable to any scaling algorithm. This reconstruction removes attack traces, and thus the classifier predicts the original class again.

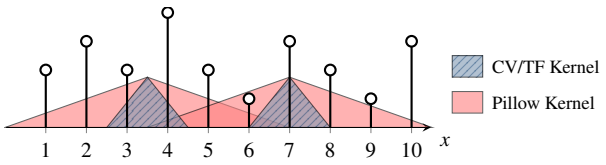


Figure 8: Comparison of bilinear scaling for Pillow, OpenCV and TensorFlow. The latter two fix $\sigma$ to 2, while Pillow uses a dynamic kernel width.

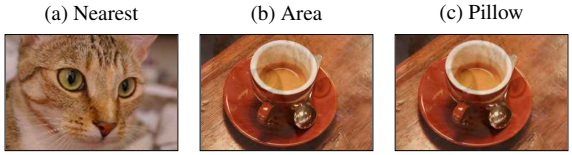


Figure 9: Comparison of scaling algorithms: (a) insecure nearest-neighbor scaling, (b) robust area scaling, and (c) robust scaling from Pillow. Note the visible attack traces in (b) and (c).


Figure 11: Examples of our defense: (a) insecure nearest-neighbor scaling, (b) robust scaling using a median filter, and (c) a random filter. Note that attack traces are not visible anymore.

### 4.3 Defense 2: Image Reconstruction

We construct our defense around the main working principle of image-scaling attacks: The attacks operate by manipulating a small set of pixels that controls the scaling process. With knowledge of the scaling algorithm, we can precisely identify this set of pixels in the attack image. The naive defense strategy to remove this set effectively blocks any attack, yet it corrupts the scaling, as all relevant pixels are removed. Instead, we first identify all pixels processed by a scaling algorithm and then reconstruct their content using the remaining pixels of the image.

Reconstructing pixels in images is a well-known problem in image processing, and there exist several methods that provide excellent performance in practice, such as techniques based on wavelets and shearlets [e.g., 26, 30]. These involved approaches, however, are difficult to analyze from a security perspective, and their robustness is hard to assess. Hence, we propose two simple reconstruction methods for the considered pixels that possess transparent security properties: a *selective median filter* and a *selective random filter*.

**Selective median filter.** Given a scaling algorithm and a target size, our filter identifies the set of pixels $\mathcal{P}$ in the input image that is processed during scaling. For each of the pixels $p \in \mathcal{P}$, it determines a window $W_p$ around $p$, similar to a convolution kernel, and computes the median pixel value for this window. To make the computation robust, we define the size of this window as $2\beta_h \times 2\beta_v$, which ensures that half of the pixels overlap between the different windows and thus hinders existing scaling attacks. Furthermore, we take care of other manipulated points $p' \in \mathcal{P}$ in $W_p$ and exclude them from the computation of the median. Figure 10 depicts the basic principle of our selective median filter.

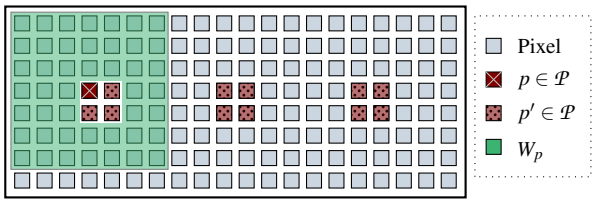


Figure 10: Image reconstruction using a selective median filter. Around each point $p$ that is considered by the downscaling algorithm (red), we take the median of all values in a window around it (green), except for other candidates that are present in the window.

In comparison to other approaches for reconstructing the content of images, this defense builds on the statistical robustness of the median operation. Small groups of pixels with high or low values are compensated by the median. On average, the adversary is required to change about 50% of the pixels in a window to reach a particular target value for the median. Our evaluation demonstrates that non-adaptive as well as adaptive adversaries are not capable of effectively manipulating these median values without introducing strong visible artifacts (see Section 5).

The robustness of the median filter comes at a price: Computing the median for all pixels in each window $W_p$ for all $p \in \mathcal{P}$ yields a run-time complexity of $O(|\mathcal{P}| \cdot \beta_h \cdot \beta_v)$. That is, the run-time growths quadratically with the scaling ratio. While this overhead might be neglectable when working with large neural networks, there also exist applications in which more efficient scaling is necessary. Providing secure and efficient scaling, however, is a challenging task, as the robustness of a scaling algorithm increases with the number of considered pixels.

**Selective random filter.** To tackle the problem of efficiency, we also propose a selective random filter that takes a random point from each window instead of the median. This filter is suitable for applications that demand a very efficient runtime performance and might tolerate a loss in visual quality. Appendix B outlines the filter in more detail.

In summary, we present two defenses that target the core of image-scaling attacks. As exemplified by Figure 11, both restore the pixels that an adversary changes and prevent the attacks. These defenses can be easily used in front of existing scaling algorithms, such that almost no changes are necessary to the typical workflow of machine learning systems.

## 5 Evaluation

We continue with an empirical evaluation of our defenses against image-scaling attacks. In Section 5.2 and 5.3, we study the security of robust scaling algorithms (Defense 1). In Section 5.4 and 5.5, we examine our novel defense based on image reconstruction (Defense 2). For each defense, we start the evaluation with a non-adaptive adversary that performs regular image-scaling attacks and then proceed to investigate an adaptive adversary who tries to circumvent our defenses.

## 5.1 Experimental Setup

To evaluate the efficacy of our defenses, we consider the objectives O1 and O2 of image-scaling attacks presented in Section 2.2.3. If a defense is capable of impeding one of these objectives, the attack fails. For example, if the control of the adversary over the source is restricted, such that the classification of the scaled version is not changed, the defense has foiled O1. Similarly, if the embedded target image becomes clearly visible, the defense has thwarted O2. Consequently, we design our experiments along with these two objectives.

**Dataset & Setup.** We use the ImageNet dataset [25] with a pre-trained VGG19 model [28] for our evaluation. This deep neural network is a standard benchmark in computer vision and expects input images of size $224 \times 224 \times 3$. From the dataset, we randomly sample 600 images as an unmodified reference set and 600 source images for conducting attacks. For each source image, we randomly select a target image from the dataset, ensuring that both images have different classes and predictions. As we are interested in investigating different scaling ratios, we sample the images such that we obtain 120 images for each of the following five intervals of ratios: $[2,3), [3,4), [4,5), [5,7.5), [7.5,10)$. Since we have two ratios along the vertical and horizontal direction for each image, we consider the minimum of both for this assignment.

We implement image-scaling attacks in the strong variant proposed by Xiao et al. [35]. We make a slight improvement to the original attacks: Instead of using a fixed $\varepsilon$ value, we increase its value gradually from 1 up to 50 if the quadratic programming solver cannot find a solution. During our evaluation, we observe that single columns or rows may require a larger $\varepsilon$ to find a feasible solution. In this way, we can increase the attack's success rate, if only a single part of an image requires a higher $\varepsilon$ value.

As scaling algorithms, we consider the implementations of nearest-neighbor, bilinear, bicubic, and area scaling from the libraries OpenCV (version 4.1), Pillow (version 6.0), and tf.image (version 1.13) from TensorFlow. We omit the Lanczos algorithm, as it provides comparable results to bicubic scaling in our experiments due to the similar convolution kernel and kernel width (see Figure 6).

**Evaluation of O1: Predictions using VGG19.** To assess objective O1 of the attacks, we check if the deep neural network VGG19 predicts the same class for the scaled image scale($A$) and the target image $T$. As there are typically minor fluctuations in the predicted classes when scaling with different ratios, we apply the commonly used top-5 accuracy. That is, we check if a match exists between the top-5 predictions for the target image $T$ and the scaled image scale($A$).

**Evaluation of O2: User Study.** To investigate objective O2, we conduct user studies with 36 human subjects. The group consists of female and male participants with different professional background. The participants obtain 3 attack images for each interval of scaling ratio and are asked to visually identify one or more of three classes, where one class corresponds to the source image, one to the embedded target image and the third to an unrelated class. We consider an attack successful, if a participant selects the class of the source image only and does not notice the target image.

**Evaluation of O2: PSNR.** As quantitative measurement, we additionally use the Peak Signal to Noise Ratio (PSNR), a common metric in image processing [8], to measure the difference between the unmodified source image and the attacked image. Formally, the PSNR for the attack image $A$ and the source image $S$ is defined as

$$\mathrm{PSNR}(A,S) = 10 \log_{10} \left( \frac{I_{\max}^2}{\frac{1}{N} \| A - S \|_2^2} \right). \qquad (10)$$

The denominator represents the mean squared error between both images with $N$ as the total number of pixels, and $I_{max}$ as the maximum of the pixel range. A high PSNR value (larger than 25 dB) indicates a strong match between two images. As a conservative choice, we consider the attack unsuccessful if the PSNR value is below 15 dB. We also experimented with more advanced methods for comparing the quality of images, such as feature matching based on SIFT analysis [16]. This technique, however, shows the same trends as the simple PSNR measurement, and thus we omit these measurements.

## 5.2 Defense 1: Non-Adaptive Attack

In our first experiment, we examine the robustness of existing scaling algorithms from OpenCV, TensorFlow, and Pillow against image-scaling attacks. Note that we investigate area scaling in the following Section 5.3, as it is not vulnerable to standard image-scaling attacks.

**Evaluation O1.** Figure 12 shows the performance of the attack as the ratio of classifications with the wanted target class after scaling. The attack is successful with respect to O1 for all scaling algorithms from OpenCV, TensorFlow, and Pillow. An exception is Pillow's bilinear scaling where the success rate is 87%, as a feasible solution is not found for all source and target pairs here. Overall, our results confirm that an attacker can successfully manipulate an image such that its scaled version becomes a target image, irrespective of the scaling algorithm or library. This manipulation, however, is not sufficient for a successful attack in practice, as visual traces may clearly indicate the manipulation and undermine the attack. We thus also evaluate O2 in this experiment.

**Evaluation O2.** Figure 13 shows the results from our user study investigating the visual perception of the generated attack images. In line with our theoretical analysis, the attack is successful against OpenCV and TensorFlow, once a certain

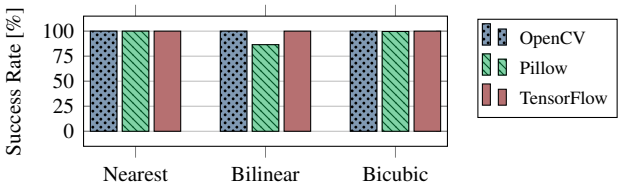


Figure 12: Success rate of image-scaling attacks with respect to objective O1: the number of classifications with target class after scaling.

scaling ratio is reached (red bars in Figure 13). We observe that for ratios exceeding 5, most attack images are not detected by the participants. However, for the implementations of bilinear and bicubic scaling in the Pillow library, the participants always spot the attack and identify the embedded target class in the source image. This result confirms our analysis of the implementations in Section 4.2 and the vital role of the dynamic kernel width used by Pillow.

In addition, Figure 19 in Appendix D reports the PSNR values between the attack and source image over the entire dataset. We observe the same trend as in the user study. For OpenCV and TensorFlow, the images become similar to each other with a larger $\beta$, reaching PSNR values above 25 dB.

**Summary.** We can confirm that image-scaling attacks are effective against several scaling algorithms in popular imaging libraries. The attacks succeed in crafting images that are classified as the target class. However, the visibility of the attacks depends on the scaling ratio and the kernel width. In the case of Pillow, the attack fails for bilinear, bicubic, and Lanczos scaling to hide the manipulations from a human viewer. We thus conclude that these implementations of scaling algorithms can be considered robust against a non-adaptive adversary in practice.

## 5.3 Defense 1: Adaptive Attacks

In our second experiment, we consider an adaptive adversary that specifically seeks means for undermining robust scaling. To this end, we first attack the implementation of the Pillow library (Section 5.3.1) and then construct attacks against area scaling in general (Section 5.3.2 and 5.3.3).

### 5.3.1 Attacking the Pillow Library

Our analysis shows that image-scaling attacks fail to satisfy objective O2 when applied to the Pillow library. The dynamic kernel width forces the attack to aggressively change pixels in the source, such that the target image becomes visible. As a remedy, we propose to limit the number of changed pixels. To build on the successful attacks against OpenCV and TensorFlow, we allow 2 pixels to be freely changed in the optimization from Eq. (4) while using images with $\beta \in [4,5)$. The goal is to find a modification for these pixels, such that the convolution over the whole kernel yields the target value.

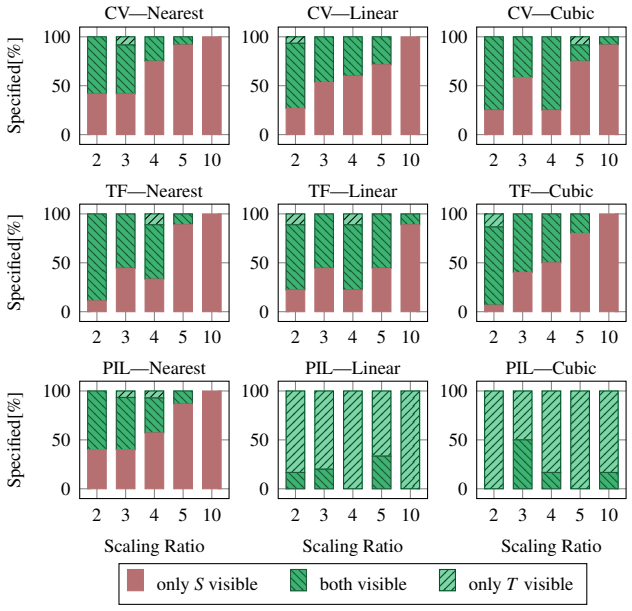


Figure 13: User study on image-scaling attacks with respect to objective O2. The attack is successful if only the source image $S$ is visible (red).

To increase the chances to obtain a feasible solution, we additionally allow the remaining pixels to be changed by at most 10. We rerun the experiment from the previous section with this new constraint and report results for 120 image pairs with $\beta \in [4,5)$ for bilinear and bicubic scaling, respectively.

**Results.** The added constraint severely impacts the success rate of the attack. The rate drops to 0% for bilinear scaling and to 0.83% for bicubic scaling. That is, the objective O1 is not reached anymore. In the majority of cases, no feasible solution exists and several columns of the source image are not modified. Only in a single case, the attack is successful for bicubic scaling. However, the attack image shows obvious traces from the target image, clearly revealing the attack.

### 5.3.2 Attacking Area Scaling

Area scaling stands out from the other algorithms as it employs a uniform weighting of pixels and operates on rectangular blocks instead of columns and rows. As a result, the original attack by Xiao et al. [35] is not applicable to this scaling algorithm. To attack area scaling, we thus propose two novel attack strategies.

The first strategy aims at slightly changing all pixels of a block to control its average. That is, we seek a minimal perturbation under the $L_1$ norm such that the average of the block becomes the targeted value. For a target value $t$, we solve the following optimization problem:

$$\min(\|\tilde{\Delta}\|_1) \text{ s.t. } \left\|avg(\tilde{S}+\tilde{\Delta})-t\right\|_\infty \leqslant \varepsilon\,, \qquad (11)$$

where $\tilde{S}$ is the current block, $\tilde{\Delta}$ its perturbation and $\varepsilon$ a small threshold. The $L_1$ norm in Eq. (11) leads to an equally dis-

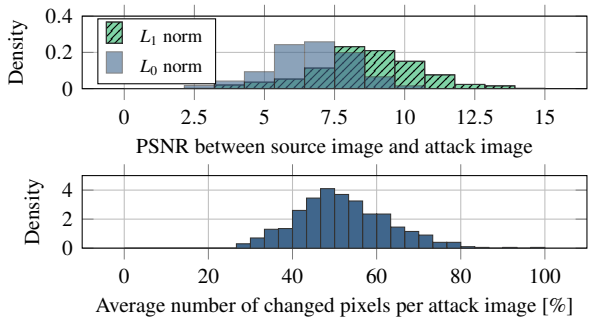


Figure 14: Adaptive attack against area scaling: (a) Distribution of PSNR values and (b) the average number of changed pixels by the $L_0$-based attack.

tributed manipulation of the pixels in each block. The results for the $L_2$ norm are equivalent and thus omitted.

The second strategy aims at adapting only a few pixels of a block while leaving the rest untouched. In this case, we optimize the $L_0$ norm, since only the number of changed pixels counts. Our attack works as follows for a current image block: if the target value is larger than the current average, the adversary iteratively sets pixels in the source to $I_{\max}$ until the target is reached. If the target is smaller, we iteratively set pixels to 0. Note that the last value generally needs to be adapted, such that the average becomes the target value.

**Results.** With respect to objective O1, both the $L_1$ and $L_0$ attack are successful in 100% of the images. However, both variants fail reaching objective O2 in all of the cases. A manual inspection of the images reveals that the source is largely overwritten by both attacks and parts of the target become visible in all attack images. Figure 14(a) provides results on this experiment by showing the distribution of PSNR values over all source-attack image pairs. The average PSNR is 8.6 dB for $L_1$ and 6.7 dB for $L_0$, which corresponds to a very low similarity between the source and the attack image. In addition, Figure 14(b) depicts the distribution of changed pixels for the $L_0$ attack. While for the majority around 50% of the pixels are changed, a few images only require to change 28%. Still, this is too much to achieve objective O2. Figure 20 in Appendix D shows the five best images from our evaluation with the smallest number of changed pixels. In all cases, the source image cannot be recognized anymore.

#### 5.3.3 Selective Source Image

In addition to the two adaptive attacks, we also examine area scaling under a more challenging scenario. In this scenario, the adversary selects the most suitable source image for a fixed target. As a result, the class of the source image is arbitrary and potentially suspicious, yet the attack becomes stronger due to the selected combination of source and target. We implement this strategy as follows: For each target image $T$, we choose the source image $S$, for which the scaled version has the smallest average distance to the target image. Fewer changes are thus required to obtain a similar output after scaling. We report results for the 100 best novel source-target pairs in the following.

As before, both the $L_1$ and $L_0$ attack are successful in 100% of all cases regarding objective O1. However, the attack again largely overwrites the source image, such that the target is visible in all cases. The examples from Figure 21 in Appendix D underline that the attack fails to keep the changes minimal, although the source and target are similar to each other. The average PSNR value is 16 dB for $L_1$ and 12 dB for $L_0$. Both are slightly higher than in the non-selective scenario but still far too low compared to successful examples from Section 5.2.

**Summary.** We conclude that area scaling is robust against the different adaptive attacks considered in this work, as well as the selection of source images. These attacks are a best effort for assessing the security of area scaling and confirm our theoretical analysis from Section 4.2. In summary, we recommend using area scaling when the uniform weighting of pixels does not impact any following analysis steps.

### 5.4 Defense 2: Non-Adaptive Attack

We proceed with evaluating our novel defenses for reconstructing images (Section 4.3). In particular, we combine the selective median and random filter with a vulnerable scaling algorithm and test the robustness of the combination. As attacks, we consider all manipulated images from Section 5.2 that satisfy the objectives O1 and O2 for one scaling algorithm. This includes attacks against nearest-neighbor scaling from all imaging libraries as well as attacks against bilinear and bicubic scaling from OpenCV and TensorFlow.

**Evaluation O1.** Our two defenses prevent all attacks. When they are employed, no attack image succeeds in reaching objective O1 for the respective scaling algorithm. The image reconstruction effectively removes the manipulated pixels and thereby prevents a misclassification of the images.

**Evaluation O2.** As the original image content is reconstructed, the visual difference between the source and the reconstructed images are minimal. Figure 15 depicts the distribution of PSNR values between each source and attack image—before and after reconstruction. The quality considerably increases after restoration and reaches high PSNR values above 25 dB. Figure 22 in Appendix D provides some examples before and after reconstruction.

**Reconstruction Accuracy.** Table 4 depicts the success rate of reconstructing the attack image's original prediction, that is, we obtain the prediction of its actual source image. The median filter recovers the predictions in almost all cases successfully. For the random filter, the success rate is slightly reduced due to the noise from the reconstruction.

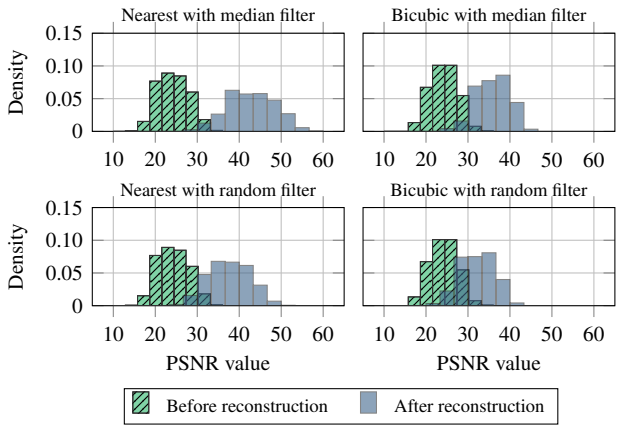


Figure 15: PSNR distribution before and after attack image reconstruction for median and random filter on OpenCV's scaling algorithms. Results for the other scaling algorithms are similar and thus omitted.

In addition, we also measure the impact of both filters on benign, unmodified images. The median filter runs with almost no loss of accuracy. The random filter induces a small loss which can be acceptable if a low run-time overhead of this defense is an important criterion in practice.

**Run-time Evaluation.** Finally, we evaluate the run-time performance of the two proposed defenses. To this end, we apply the defenses along with different scaling algorithms to 2,000 images and measure the average run-time per image. The test system is an Intel Xeon E5-2699 v3 with 2.4 GHz. Our measurements are shown in Figure 16 on a logarithmic scale in microseconds. Area scaling as well as our defenses introduce a notable overhead and cannot compete with the insecure nearest-neighbor scaling in performance. However, in comparison to a pass through the VGG19 model, our defenses are almost an order of magnitude faster and induce a neglectable overhead for deep learning systems.

**Summary.** This experiment shows that the median and random filter provide effective defenses against non-adaptive attacks. In contrast to robust scaling, the defenses prevent the attack and reconstruct the original prediction.

| Library | Algorithm | Median | | Random | |
|---|---|---|---|---|---|
| | | Attacks | Unmod. | Attacks | Unmod. |
| OpenCV | Nearest | 99.6% | 99.0% | 89.3% | 89.1% |
| | Bilinear | 100.0% | 99.4% | 97.7% | 98.0% |
| | Bicubic | 100.0% | 99.2% | 91.4% | 93.4% |
| TF | Nearest | 99.6% | 99.0% | 88.9% | 89.1% |
| | Bilinear | 100.0% | 98.9% | 97.7% | 97.7% |
| | Bicubic | 100.0% | 99.4% | 91.7% | 92.0% |
| Pillow | Nearest | 100.0% | 99.6% | 88.1% | 90.4% |

Table 4: Performance of defense in terms of recovering correct outputs from the attack images, and impact on benign images.

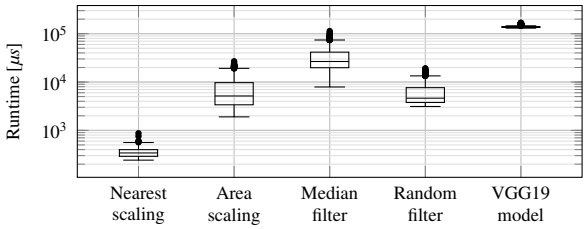


Figure 16: Run-time performance of nearest-neighbor and area scaling as well as our defenses in combination with nearest-neighbor scaling. Additionally, a forward pass of VGG19 is shown.

## 5.5 Defense 2: Adaptive Attacks

Finally, it remains to investigate the robustness of the two proposed defenses against an adaptive adversary who is aware of the defenses and adapts her attack accordingly. We thus develop two strategies that aim at misleading the image reconstruction of attack images. Both strategies attempt to manipulate the reconstruction of the pixels $p \in \mathcal{P}$, such that they keep their value after applying the median or random filter.

**Median Filter.** Our attack strategy for the median filter is as follows: Given a window $W_p$ around $p \in \mathcal{P}$, we denote by $m$ the current median of $W_p$. Note that $p$ is not part of $W_p$ (see Figure 10). The adversary seeks a manipulation of the pixels in $W_p$, such that $m = p$. Hence, applying the median filter will not change $p$ and the adversarial modification remains. Without loss of generality, we assume that $m < p$. In order to increase $m$, the adversary needs to set more pixels to the value of $p$. We start with the highest pixel value that is smaller than $p$ and set it to $p$. We continue with this procedure until the median equals $p$. In Appendix C, we show that this attack strategy is *optimal* regarding the $L_0$, $L_1$, and $L_2$ norm if the windows $W_p$ do not overlap. A smaller number of changes to the image cannot ensure that $m = p$. These results give a first intuition on the robustness of the median filter. A considerable rewriting is necessary to change the median, even in the overlapping case where an adversary can exploit dependencies across windows.

In our experiments, we vary the maximum fraction $\delta$ of allowed pixel changes per window. This bound allows us to measure the defense's robustness depending on the $L_0$ norm.

**Random Filter.** For the random filter, our attack strategy increases the probability that the target value in a window $W_p$ is selected. To this end, we let the adversary set a fraction $\delta$ of all pixels in $W_p$ to $p$. To minimize the number of changes to the image, we replace only those pixels in the window with the smallest absolute distance to $p$. This strategy is optimal in the sense that manipulation with fewer changes would result in a lower probability for hitting the target value $p$.

**Results.** Figure 17 shows the success rate of the adaptive attacks regarding objective O1 for OpenCV and TensorFlow. The results for Pillow's nearest-neighbor scaling are similar

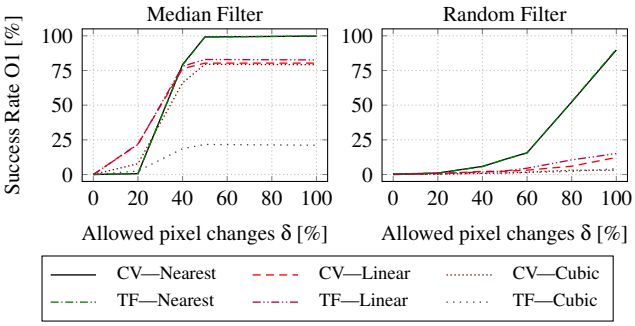


Figure 17: Success rate of the adaptive attacks against defenses with respect to objective O1. Note that O2 is not satisfied (see Figure 18).

and thus omitted. The adaptive attacks need to considerably modify pixels so that the manipulated images are classified as the target class. The median filter is robust until 40% of the pixels in each window can be changed. Against the random filter, a higher number of changed pixels is necessary to increase the probability of being selected.

With respect to goal O2, both defenses withstand the adaptive attacks and thus remain secure. Rewriting 20% of the pixels already inserts clear traces of manipulation, as exemplified by Figure 23 in Appendix D. In all cases, the attack image is a mix between source- and target class. In addition, Figure 18 shows the results from our user study for the median filter. The participants identify the attacks in the vast majority of the cases. In a few cases, the participants only recognized the source class. A closer analysis reveals that the distortion in these cases is so strong that the detection of particular classes is difficult. As a result, the participants did not specify the source class.

**Summary.** We conclude that the two proposed defenses are robust against the different adaptive attacks. These attacks are both optimal with respect to the number of changes and thus provide strong empirical evidence for the robustness of the defenses. If a vulnerable scaling algorithm needs to be used in a machine-learning system or the reconstruction of the original class is essential, we thus recommend using one of the defenses as a preprocessing step.

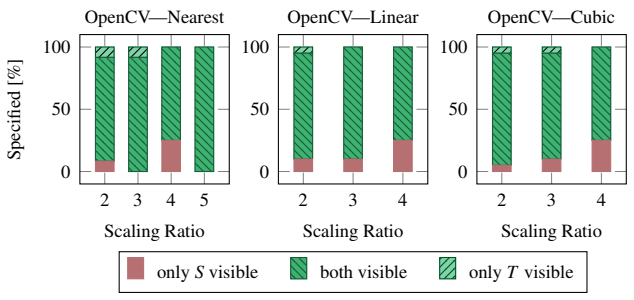


Figure 18: User study to determine the success rate of the adaptive attack against the median filter with respect to O2.

## 6 Related Work

Closest to our work are different attacks and defenses from the area of adversarial machine learning [see 3, 21]. For example, approaches for creating and detecting adversarial examples share related objectives [e.g., 4, 6, 14, 17, 23]. Moreover, techniques for manipulating machine learning models revolve around a similar problem setting. These techniques change training data or model parameters to obtain targeted responses [e.g., 2, 10, 15, 34]. While not directly related, methods for memberships and property inference [e.g., 9, 27] as well as model inversion and extraction [e.g., 20, 33] also constitute threats to machine learning. Our work extends this line of research by examining the preprocessing step. We provide a comprehensive analysis of image-scaling attacks and derive defenses for prevention. In a concurrent work [24], we study the application for the poisoning scenario. Moreover, we note that image-scaling attacks further bridge the gap between adversarial learning and multimedia security where the latter also considers adversarial signal manipulations [1, 22].

Finally, image-scaling attacks differ from prior work in two important properties: (a) The attacks affect all further steps of a machine learning system. They are thus agnostic to feature extraction and learning models, giving rise to general adversarial examples and poisoning. (b) Fortunately, we can show that the vulnerability underlying image-scaling attacks can be effectively mitigated by defenses. This rare success of defenses in adversarial machine learning is rooted in the well-defined structure of image scaling that fundamentally differs from the high complexity of deep learning models.

## 7 Conclusion

Image-scaling attacks exploit vulnerabilities in the preprocessing of machine learning with considerable impact on computer vision. In this paper, we provide the first in-depth analysis of these attacks. Based on insights from this analysis, we propose different defenses that address the root cause of the attacks rather than fixing their symptoms.

For evaluating our defenses, we consider an adaptive adversary who has full knowledge about the implementation of scaling algorithms and our defense strategy. Our empirical results show that image-scaling attacks can be prevented effectively under this threat model. The proposed defenses can be easily combined with existing imaging libraries and require almost no changes to machine learning pipelines. Furthermore, our findings are not limited to the considered scaling algorithms and enable developers to vet their own scaling techniques for similar vulnerabilities.

Overall, our work provides novel insights into the security of preprocessing in machine learning. We believe that further work is necessary to identify and rule out other vulnerabilities in the different stages of data processing to strengthen the security of learning-based systems.

## Availability

We make our dataset and code publicly available at http://scaling-attacks.net to encourage further research on secure image scaling. Our defenses are also implemented in C++ with Eigen, such that they can be easily employed as plug-ins for TensorFlow.

## Acknowledgment

We would like to thank our shepherd Nicolas Papernot, the anonymous reviewers and David Wagner for their suggestions and comments. Furthermore, we acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and the research grant RI 2469/3-1, by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref 01IS18037A), and from the state of Lower Saxony under the project Mobilise.

## References

[1] M. Barni and F. Pérez-González. "Coping with the enemy: Advances in adversary-aware signal processing". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 2013.

[2] B. Biggio, B. Nelson, and P. Laskov. "Support Vector Machines Under Adversarial Label Noise". In: *Proc. of Asian Conference on Machine Learning (ACML)*. 2011.

[3] B. Biggio and F. Roli. "Wild patterns: Ten years after the rise of adversarial machine learning". In: *Pattern Recognition* 84 (2018).

[4] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. "Evasion Attacks against Machine Learning at Test Time". In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2013.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

[6] N. Carlini and D. A. Wagner. "Towards Evaluating the Robustness of Neural Networks." In: *Proc. of IEEE Symposium on Security and Privacy (S&P)*. 2017.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Tech. rep. 2018. arXiv: 1810.04805.

[8] J. Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2010.

[9] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. "Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations." In: *Proc. of ACM Conference on Computer and Communications Security (CCS)*. 2018.

[10] T. Gu, B. Dolan-Gavitt, and S. Garg. *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*. Tech. rep. 2017. arXiv: 1708.06733.

[11] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[12] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. "Densely Connected Convolutional Networks". In: *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet: Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Proccessing Systems (NIPS)*. 2012.

[14] J. Li, S. Ji, T. Du, B. Li, and T. Wang. "TextBugger: Generating Adversarial Text Against Real-world Applications". In: *Proc. of Network and Distributed System Security Symposium (NDSS)*. 2019.

[15] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang. "Trojaning Attack on Neural Networks". In: *Proc. of Network and Distributed System Security Symposium (NDSS)*. 2018.

[16] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (2004).

[17] M. Lécuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. "Certified Robustness to Adversarial Examples with Differential Privacy." In: *Proc. of IEEE Symposium on Security and Privacy (S&P)*. 2019.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Proccessing Systems (NIPS)*. 2013.

[19] A. V. Oppenheim, J. R. Buck, and R. W. Schafer. *Discrete-Time Signal Processing; 2nd ed.* Prentice-Hall, 1999.

[20] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami. "Practical Black-Box Attacks against Machine Learning". In: *Proc. of ACM Asia Conference on Computer Computer and Communications Security (ASIA CCS)*. 2017.

[21] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. "SoK: Security and Privacy in Machine Learning". In: *Proc. of IEEE European Symposium on Security and Privacy (EuroS&P)*. Apr. 2018.

[22] E. Quiring, D. Arp, and K. Rieck. "Forgotten Siblings: Unifying Attacks on Machine Learning and Digital Watermarking". In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. 2018.

[23] E. Quiring, A. Maier, and K. Rieck. "Misleading Authorship Attribution of Source Code using Adversarial Learning". In: *Proc. of USENIX Security Symposium*. 2019.

[24] E. Quiring and K. Rieck. "Backdooring and Poisoning Neural Networks with Image-Scaling Attacks". In: *Deep Learning and Security Workshop (DLS)*. 2020.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015).

[26] S. Sardy, P. Tseng, and A. G. Bruce. "Robust Wavelet Denoising". In: *IEEE Transactions on Signal Processing* 49 (2001).

[27] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. "Membership Inference Attacks against Machine Learning Models". In: *Proc. of IEEE Symposium on Security and Privacy (S&P)*. 2017.

[28] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Tech. rep. 2014. arXiv: 1409.1556.

[29] S. W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.

[30] C. Sun, C. Tang, X. Zhu, X. Li, and L. Wang. "An efficient method for salt-and-pepper noise removal based on shearlet transform and noise detection". In: *AEUE - International Journal of Electronics and Communications* 69.12 (2015).

[31] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Proccessing Systems (NIPS)*. 2014.

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

[33] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. "Stealing Machine Learning Models via Prediction APIs". In: *Proc. of USENIX Security Symposium*. 2016.

[34] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks." In: *Proc. of IEEE Symposium on Security and Privacy (S&P)*. 2019.

[35] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li. "Seeing is Not Believing: Camouflage Attacks on Image Scaling Algorithms". In: *Proc. of USENIX Security Symposium*. 2019.

# A Downgrade Attack to Nearest Scaling

As part of our analysis, we identified a side effect in the implementation of $g(p)$ (see Eq. (6)) in OpenCV and TensorFlow. An adversary can enforce the usage of nearest scaling by choosing a respective scaling factor although the library is supposed to use bilinear, bicubic or Lanczos scaling. In particular, if the scaling ratio is an uneven integer, $\beta = 2z+1$, $z \in \mathbb{N}$, OpenCV is effectively using nearest scaling. In TensorFlow, each integer with $\beta \in \mathbb{N}$ leads to the same effect. Thus, if the adversary can control the source image size, she can resize her image before to obtain the respective scaling factor. This in turn allows her to perform a more powerful scaling attack by creating attack images with less distortion, as the ratio of considered pixels decreases (see Section 3.3). Note that we do not exploit this issue in our evaluation. We test over a variety of scaling factors to draw general conclusions on scaling attacks.

Table 5: Implementation of $g(p)$ in OpenCV, TensorFlow and Pillow

| Library | $g(\cdot)$ |
|---|---|
| OpenCV | $g(p) = (p+0.5)\cdot\beta - 0.5$ |
| TensorFlow | $g(p) = p\cdot\beta$ (*) |
| Pillow | $g(p) = (p+0.5)\cdot\beta$ |

(*) The scaling function in TensorFlow can be changed to the definition from OpenCV. However, this option is not exposed in `tf.image.resize_images`, the high level resizing API.

To understand its reason, we need to consider the mapping $g(p)$ and the kernel $w$. Table 5 shows the slightly different implementations of $g(p)$ in OpenCV, TensorFlow and Pillow. For OpenCV, for instance, if $\beta$ is an uneven integer, $g(p)$ will always be an integer. Thus, only one pixel will be used for the convolution. A closer look on the definition of the kernels in Figure 6 reveals the underlying reason. Each kernel is zero for integer positions. Thus, if $g(p)$ is an integer and the kernel is exactly positioned here, each neighboring pixel obtains a weight of zero. Thus, only the pixel at position $g(p)$ is used. This behavior corresponds to nearest scaling. We observe this effect for bilinear, bicubic and Lanczos scaling in OpenCV and TensorFlow. On the contrary, Pillow makes use of a dynamic kernel width, so that we do not observe this behavior in this case.

# B Selective Random Filter

Our random filter is identical to the selective median filter, except for that it takes a random point from each window instead of the median. That is, given a point $p \in \mathcal{P}$, we consider a window $W_p$ around $p$ of size $2\beta_h \times 2\beta_v$ and randomly select a point as a reconstruction of $p$. Again, we exclude points $p' \in \mathcal{P}$ from this window to limit the attacker's influence.

Randomly selecting a point for reconstruction obviously comes with problems. First, the reconstruction becomes nondeterministic. Second, the scaled image might suffer from poor quality. Our evaluation, however, shows that the loss due to random sampling is small and might be acceptable for the benefit of a very efficient run-time performance. The filter reconstructs an image with a complexity of $O(|\mathcal{P}|)$, which is independent of the scaling ratio. Furthermore, the filter also provides strong protection from attacks. If an image contains $|\mathcal{P}|$ relevant points, there exist $|\mathcal{P}| \cdot 4\beta_h\beta_v$ possible combinations for its reconstruction. If we consider a scaling ratio of 5 and a target size of $200 \times 200$, this already amounts to 4 million different combinations an attacker needs to guess from.

## C Adaptive Attack Against Median Filter

In the following, we analyze our adaptive attack against the median-based defense. We demonstrate that the attack is optimal regarding the $L_0$, $L_1$, and $L_2$ norm if each window $W_p$ does not overlap with other windows. An adversary cannot make less changes to control the output of the median filter.

For a given attack image and window $W_p$, the adversary seeks to manipulate the pixels in $W_p$ such that the median $m$ over $W_p$ still corresponds to $p$. In this way, the modifications from the image-scaling attack remain even after applying the median filter. Without loss of generality, we assume that $m < p$ and further unroll $W_p$ to a one-dimensional signal. We consider a signal with uneven length $k$ and denote the numerical order by brackets, so that the signal is given by:

$$x_{(1)}, \cdots, x_{\left(\frac{k}{2}\right)}, m_{\left(\frac{k+1}{2}\right)}, x_{\left(\frac{k+2}{2}\right)}, \cdots, x_{(l)}, \cdots, x_{(k)} \tag{12}$$

We denote by $x_{(l)}$ the largest pixel in the sorted signal that is smaller than $p$. The objective is to change the signal with the fewest possible changes such that $m = p$.

We start by observing that we need to change $l - \frac{k+1}{2} + 1$ pixels to move the median to $p$. Less changes do not impact the numerical order sufficiently. We can thus conclude that the minimal $L_0$ norm for an attack is given by

$$L_0 = l - \tfrac{k+1}{2} + 1 \,. \tag{13}$$

Next, we show that setting all pixels between $m$ and $x_{(l)}$ to $p$ successfully moves the median as well as minimizes the $L_1$ and $L_2$ norm in addition. First, we observe that if we replace pixels with indices in $[1, k/2]$ by a value smaller than $m$, the median is not changed. Likewise, replacing pixels larger than $x_{(l)}$ by a value larger than $m$ does not change the median. Two methods remain: (1) We can replace pixels with indices in $[1, (k+1)/2]$ by a value larger than $m$. (2) We can set all pixels with index $[(k+1)/2,\ l]$ to $p$. While both methods can move the median to $p$, the latter induces less changes regarding the $L_1$/$L_2$ norm, as these values are closer to $p$. Thus, our adaptive attack uses the optimal strategy for the $L_1$/$L_2$ norm by setting all pixels between $m$ and $x_{(l)}$ to $p$. Furthermore, we can derive a simple bound for the $L_2$ norm:

$$(L_2)^2 = \sum_{(\frac{k+1}{2}) \leqslant i \leqslant l} \left(x_{(i)} - p\right)^2 \leqslant L_0 \, (m-p)^2 \,. \tag{14}$$

Overall, we can exactly compute the number and amount of required changes for a successful attack. Our analysis, however, also shows that the attack always depends on the concrete pair of a source and a target image, and there is no notion of a class boundary. Consequently, we cannot derive a general bound, as achieved with certifiable defenses against adversarial examples. Yet, our empirical results in Section 5.5 demonstrate that the necessary changes are very large if target and source images show realistic content, so that the median $m$ and the target value $p$ are not close to each other.

## D Additional Figures

Figures 19 to 23 give further information and examples from our evaluation. In particular, they provide visual examples of successful and failed attacks, thereby highlighting the working principle of image-scaling attacks.

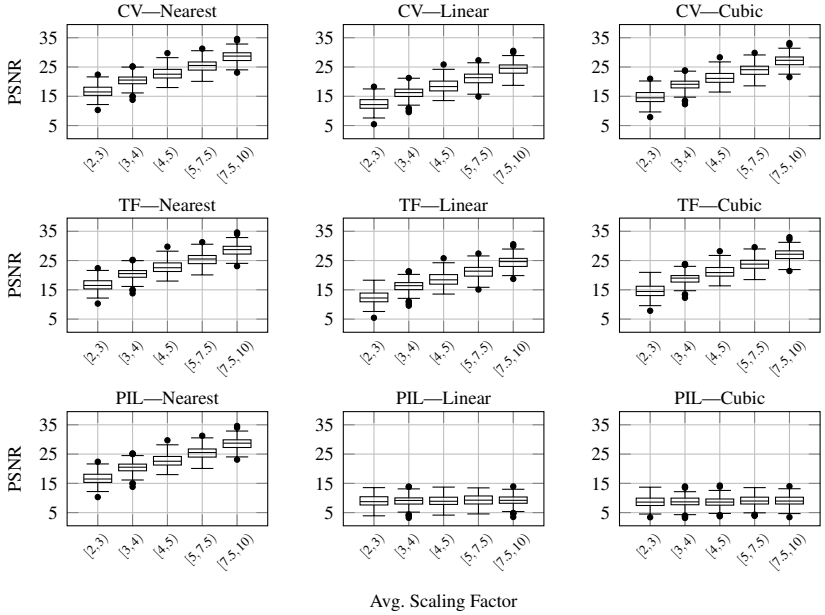


Figure 19: Success rate of attack regarding objective O2: the similarity between source image and attack image, measured by the PSNR value.


Figure 20: Best images of the $L_0$ version of our adaptive attack against area scaling. The attack fails in all cases with respect to objective O2, as each attack image is not similar to the source image anymore.


Figure 21: Selective source scenario against area scaling with our $L_1$ attack (first two columns) and $L_0$ attack (last three columns). The attack fails in all cases with respect to objective O2. While traces from the source image are visible, the attack image overwrites the source image considerably.

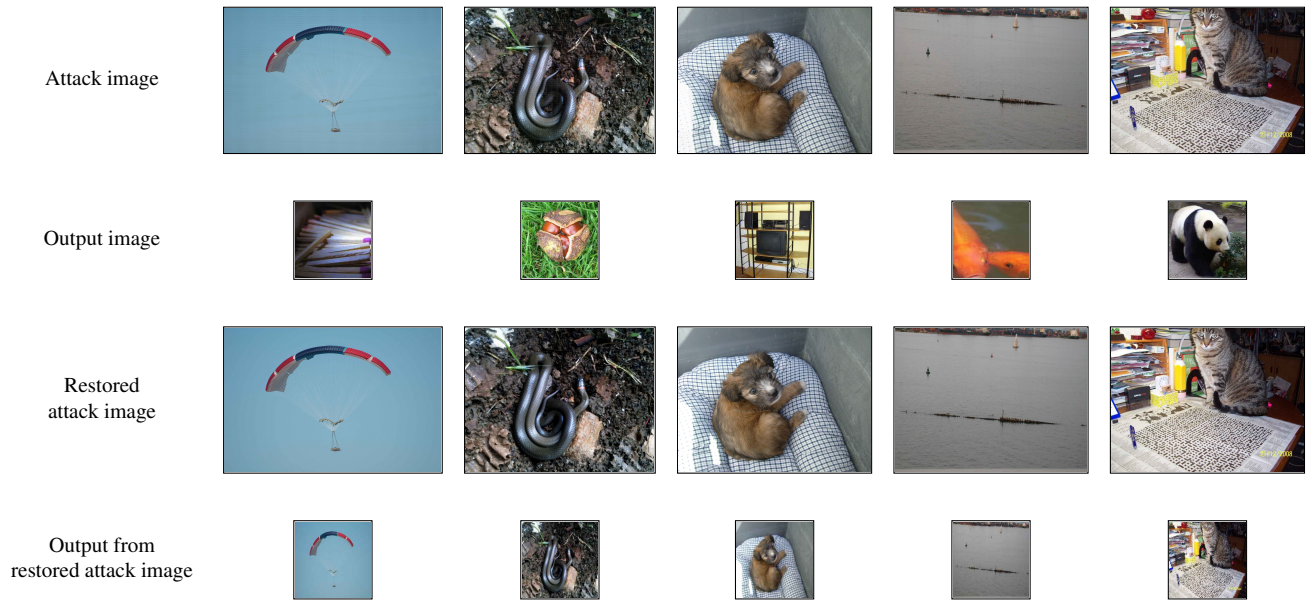


Figure 22: Randomly selected examples before and after restoration with our median filter (first three columns) and random filter (last two columns). Without restoration, the attack is successful, as the downscaling of the attack image produces an unrelated target image (1st and 2nd row). With restoration, the attack fails in all cases with respect to objective O1, as the downscaled output from the restored attack image produces the respective content and not an unrelated image (3rd and 4th row). Moreover, the filtering improves quality, as it removes traces from the attack.


Figure 23: Successful examples regarding objective O1 from the adaptive attack against the median filter if 20% of the pixels in each block can be changed. The target class is detected, but the attack image is a mix between source and target class. The results thus violate objective O2.