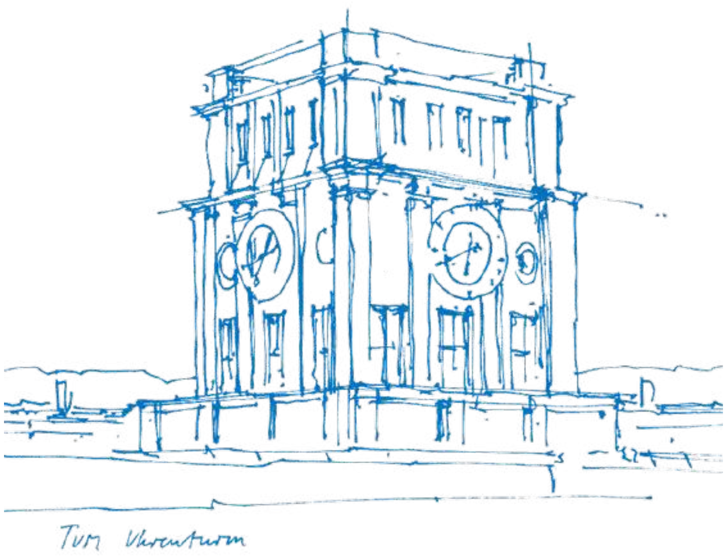
Professorship of Machine Learning
Department of Electrical and Computer Engineering
Technical University of Munich

# Randomized Smoothing as an Adversarial Defense Mechanism for Inverse Problems

**Faidra Anastasia Patsatzi**



# Randomized Smoothing as an Adversarial Defense Mechanism for Inverse Problems

Faidra Anastasia Patsatzi

Professorship of Machine Learning
Department of Electrical and Computer Engineering
Technical University of Munich

# Randomized Smoothing as an Adversarial Defense Mechanism for Inverse Problems

**Faidra Anastasia Patsatzi**

Thesis for the attainment of the academic degree

**Bachelor of Science (B.Sc.)**

at the Department of Electrical and Computer Engineering of the Technical University of Munich.

**Examiner:**

Prof. Dr. sc. ETH Zürich Reinhard Heckel

**Supervisor:**

Prof. Dr. sc. ETH Zürich Reinhard Heckel

**Submitted:**

Munich, 07.07.2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 07.07.2023

Faidra Anastasia Patsatzi

# Acknowledgment

I would like to thank and express my utmost gratitude to my supervisor, Prof. Dr. sc. Reinhard Heckel, whose useful advice and expertise guided me through all stages of this project. His assistance, patience, and unwavering support shaped this work and made it a smooth and enjoyable learning experience. I am truly grateful for the opportunity to write my bachelor's thesis under his supervision.

I would also like to thank Mr. Anselm Krainovic for providing me with custom code for adversarial attacks and offering me his valuable advice and feedback multiple times during this project.

Moreover, I am thankful to the DAAD (Deutscher Akademischer Austauschdienst / German Academic Exchange Service) for their financial as well as non-material support during my bachelor's degree through a scholarship.

Special thanks go to my friends for the incredible memories from these last three years and for their support during challenging moments.

Finally, I would like to express my deepest appreciation to my family for their unconditional love and support throughout this bachelor's degree. Thank you for always being there for me.

# Abstract

Randomized smoothing is a mechanism that can achieve certifiable robustness of neural network-based classifiers against $\ell_2$-norm bounded adversarial examples. In this project, we present an approach to randomized smoothing for inverse problems to investigate its effectiveness as an adversarial defense mechanism in image reconstruction problems. We choose super-resolution as an image reconstruction problem to implement randomized smoothing and train U-Net models for super-resolution with different levels of Gaussian noise for randomized smoothing. We also train U-Net models with adversarial training for a comparative evaluation of the robustness gains yielded by randomized smoothing in super-resolution. Our findings show that randomized smoothing is an effective adversarial defense in super-resolution and that it achieves results with better perceived visual quality than adversarial training.

# Contents

# 1 Introduction

Deep neural network architectures have been used in recent years for a variety of computer vision tasks, such as image classification, segmentation, and reconstruction. Multiple research papers have introduced novel architectures to achieve higher accuracy in computer vision tasks, e.g. deep convolutional neural networks for image classification. However, besides accuracy, robustness to adversarial examples is an important aspect of such deep learning frameworks. Experiments in publications [1, 2, 3] have shown that deep neural networks are particularly vulnerable to adversarial attacks, i.e. $\ell_2$-norm bounded perturbations of the input, which lead to high error rates in classification. In problems such as image classification, an adversarially perturbed input $x + \delta$, where $\delta$ is the adversarial perturbation, cannot always be discerned from the original input $x$ with the human eye [1]. Nevertheless, the attacked model might classify the perturbed image $x + \delta$ not only incorrectly, but also with high confidence, as shown by articles [1, 4]. An example of an image and its adversarially perturbed version can be seen in Figure 1.1.

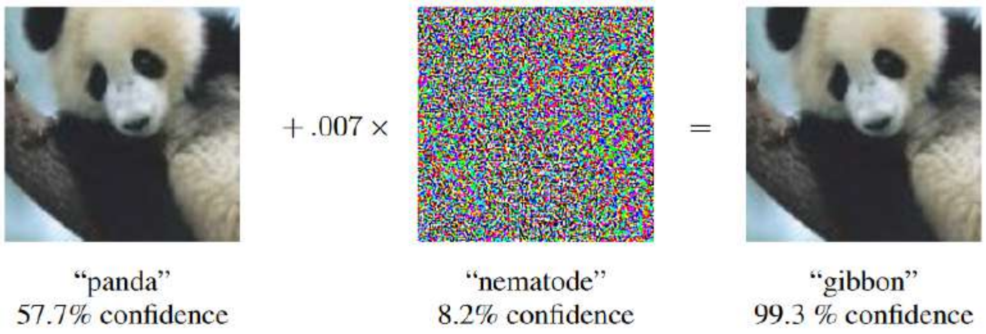


**Figure 1.1** Classification of the original image of a panda (left), adversarial noise (middle), and classification of the adversarially perturbed image (right) [1].

## 1.1 Adversarial defenses

Various defenses against adversarial perturbations have been introduced, such as network distillation and verification, adversarial training, randomized smoothing, and denoised smoothing. These defenses can be categorized as either empirical or certified.

Empirical defenses have been shown to be robust against existing adversarial attacks, and adversarial training is one of the best performing empirical defenses to date [5]. In adversarial training, a neural network is trained to minimize a loss between the perturbed instance of an input $x + \delta$ and the desired output $y$. Even though this method represents a proven defense mechanism against known adversarial attacks, it does not guarantee robustness against a potentially more powerful, still unknown adversarial attack. With ongoing research on strong adversarial attacks [4, 6], we cannot eliminate the possibility that a currently robust empirical defense is successfully attacked with a stronger adversary.

Certified defenses are methods that are proven to be robust against all (adversarial) perturbations within a constant distance to any input $x$. The certification methods used to prove this can be *exact* or *conservative*. According to *Salman et al.* [7], exact certification is too computationally expensive, whereas conservative certification is less computationally expensive, but both methods cannot be used for certification in large deep neural networks with competitive performance on datasets such as ImageNet [8]. Randomized smoothing is considered a *probabilistically* certified defense and can, therefore, be used to certify robustness in large networks.

## 1.2 Inverse problems

Robustness against $\ell_2$-norm adversarial perturbations through randomized smoothing has been studied in multiple works of research [7, 9, 10] for classification problems. Randomized smoothing for regression problems has, however, not yet been investigated extensively with regard to the robustness gains it could offer in a regression setting and especially in inverse problems.

An inverse problem is a type of regression problem where the observation $y$ depends on the input $x$. The relation between $y$ and $x$ can be expressed through

$$y = \mathcal{A}(x) + z \,, \tag{1.1}$$

where $\mathcal{A}$ is a forward, typically non-invertible transformation applied to the input, and $z$ is a noise vector [11]. The goal is to reconstruct the input $x \in \mathbb{R}^n$ from the noisy output $y \in \mathbb{R}^m$.

In this project, the effect of randomized smoothing is investigated and evaluated in the context of super-resolution, which is an image reconstruction and thereby also an inverse problem.

# 2 Background on randomized smoothing and related work

In this section, the theoretical foundations of randomized smoothing are first presented for a classification setting and then adapted to inverse problems. Additionally, denoised smoothing is briefly introduced as an extension of randomized smoothing for classification. Finally, an overview of related work in randomized smoothing is provided.

## 2.1 Randomized smoothing for classification

For a given classifier $f : \mathbb{R}^n \to Q$, where $Q$ is the set of classes, the smoothed classifier $g : \mathbb{R}^n \to Q$ is defined according to randomized smoothing [9] as

$$g(x) = \underset{c \in Q}{\arg\max}\, \mathbb{P}[f(x+\delta) = c] \quad \text{with} \quad \delta \sim \mathcal{N}(0, \sigma^2 I)\ , \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the input and $x + \delta$ is the input perturbed with Gaussian noise. The noise level $\sigma$ controls the robustness/accuracy trade-off [9]; increased values for $\sigma$ yield a more robust but less accurate classifier [12].

The calculation of the probability in Equation (2.1) is computationally infeasible when the classifier $f$ is a neural network. *Cohen et al.* [9] estimate this probability by drawing $k$ samples of $f(x + \delta)$, where $\delta$ is sampled $k$ times from an isotropic Gaussian distribution, and taking a majority vote over the predictions $f(x + \delta_j)$, where $j \in \{1, \dots, k\}$. The smoothed classifier $g$ can be defined equivalently to Equation (2.1) as

$$g(x) = \underset{c \in Q}{\arg\min}\, \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \Big[ \mathbb{1}\{f(x+\delta) \neq c\} \Big]\ , \tag{2.2}$$

and approximated through

$$\hat{g}(x) = \underset{c \in Q}{\arg\max} \sum_{j=1}^{k} \mathbb{1}\{f(x+\delta_j) = c\}\ , \tag{2.3}$$

which is the formulation of the majority vote over $k$ noisy predictions of the base classifier $f$.

*Cohen et al.* [9] provide a tight robustness guarantee for the smoothed classifier $g$. They define the probability $p_A$ as the probability that $f(x + \delta)$ yields the class $c_A$ and the probability $p_B$ as the probability that $f(x + \delta)$ yields the next most probable class $c_B$. The following $\ell_2$ robustness radius can be derived:

$$R = \frac{\sigma}{2} \Big( \Phi^{-1}(p_A) - \Phi^{-1}(p_B) \Big)\ , \tag{2.4}$$

with $\Phi^{-1}$ representing the inverse standard Gaussian CDF [9]. This radius guarantees robustness of the smoothed classifier $g$ within an $\ell_2$ region around any input $x$. Similar to the probability in Equation (2.1), computing the probabilities $p_A$ and $p_B$ in Equation (2.4) is infeasible for a neural network $f$. Therefore, *Cohen et al.* [9] estimate instead a lower bound $\underline{p_A}$ and upper bound $\overline{p_B}$ for the calculation of the certified robustness radius from Equation (2.4), using a Monte Carlo algorithm.

In randomized smoothing, a tight certification bound and thereby increased robustness is achieved if the base classifier $f$ has already been trained on inputs perturbed with Gaussian noise [9]. Randomized smoothing provides a certifiable adversarial defense by defining the smoothed classifier $g$ (Equation (2.1)). This classifier is not a neural network but depends on the neural network base classifier's $f$ predictions for noisy inputs $x + \delta$. Notably, the robustness guarantee of randomized smoothing according to *Cohen et al.* [9] does not implicate any restrictions for the base classifier $f$. Nonetheless, for increased robustness in classification using randomized smoothing, the base classifier $f$ is expected to repeatedly classify inputs $x$ correctly under Gaussian noise $\delta \sim \mathcal{N}(0, \sigma^2 I)$, and should therefore be trained on inputs $x$ augmented with Gaussian noise [9].

## 2.2 Randomized smoothing for inverse problems

For the application of randomized smoothing on an inverse problem, we need to derive a formulation of the smoothed model $g$ for a regression rather than a classification task. We consider an inverse problem where $x \in \mathbb{R}^n$ denotes an input vector and $y \in \mathbb{R}^m$ an observation, according to Equation (1.1). For a given neural network $f : \mathbb{R}^m \to \mathbb{R}^n$ that aims to reconstruct the input $x$ from a measurement $y$, we define the smoothed estimate that is equivalent to the smoothed classifier in Equation (2.2) as

$$g(y) = \arg\min_{x} \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ \|f(y+\delta) - x\|_2^2 \right] , \tag{2.5}$$

with the only difference to Equation (2.2) being the risk and the argument of minimization. Instead of choosing the class that minimizes the classification error, as in Equation (2.2), the value of $x$ in Equation (2.5) is the value that minimizes the $\ell_2$-loss for image reconstruction.

From Equation (2.5), we obtain an alternative formulation for the smoothed estimate $g$:

$$g(y) = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} [f(y+\delta)] , \tag{2.6}$$

as demonstrated in Appendix A.1.1. In practice, similar to the Monte Carlo simulation adopted by *Cohen et al.* [9] for the calculation of the smoothed classifier's prediction, the smoothed estimate $g$ from Equation (2.6) can be approximated by

$$\hat{g}(y) = \frac{1}{k} \sum_{j=1}^{k} f(y + \delta_j) , \tag{2.7}$$

where $k$ is the number of vectors $\delta_j$ drawn from the Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$.

When the base estimator $f$ is a linear estimator with $f(y) = Qy$, the smoothed estimate $g$ is identical to the base linear estimator, since $g(y) = f(y)$ as shown in Appendix A.1.2. Therefore, in that specific case, robustness gains are not expected.

## 2.3 Denoised smoothing

Denoised smoothing was introduced by *Salman et al.* in [12] as an extension of randomized smoothing. The base classifier $f$ in Equation (2.1) is replaced with $f \circ D_\theta$:

$$g(x) = \arg\max_{c \in Q} \mathbb{P}[f(D_\theta(x+\delta)) = c] \quad \text{with} \quad \delta \sim \mathcal{N}(0, \sigma^2 I) , \tag{2.8}$$

where $D_\theta : \mathbb{R}^n \to \mathbb{R}^n$ is a custom-trained denoiser.

Denoised smoothing [12] targets the problem of obtaining a robust classifier without retraining the underlying model, which would be required by adversarial defense methods such as adversarial training and randomized smoothing (for tight certification bounds). The insertion of a custom-trained denoiser provides a solution to this problem by removing the Gaussian noise added during randomized smoothing. Thereby, a robust classifier $f \circ D_\theta : \mathbb{R}^n \to Q$ is generated out of a pretrained classifier $f$.

## 2.4 Related work

Research in the field of adversarial robustness includes a handful of papers on randomized smoothing for classification, which seek to demonstrate its effectiveness as an adversarial defense mechanism against $\ell_2$-norm perturbations [13, 14, 15]. However, only loose robustness guarantees were provided by these early works of research on randomized smoothing.

A tight $\ell_2$ robustness radius was first introduced by *Cohen et al.* [9], as formulated in Equation (2.4), and established randomized smoothing as the state-of-the-art method for certified robustness in classification. *Salman et al.* [7] then demonstrated that adversarially trained smoothed classifiers achieve increased certified robustness in comparison to plain smoothed classifiers without additional adversarial training. The adversarial training employed by *Salman et al.* [7] is based on a custom adversarial attack designed for smoothed classifiers.

Further research on randomized smoothing for classification followed the direction of denoised smoothing. *Salman et al.* [12] introduce denoised smoothing, as defined in Equation (2.8), with which custom-trained denoising networks can be employed to achieve provable $\ell_2$-norm robustness for off-the-shelf pretrained image classifiers. Besides the overall increased robustness yielded by denoised smoothing, *Salman et al.* [12] argue that an implementation of randomized smoothing, where the base classifier $f$ has not been trained on Gaussian perturbations of its inputs, leads to loose robustness bounds. Standard, off-the-shelf classifiers are not by default robust against Gaussian augmentations of their inputs and therefore denoised smoothing [12] provides an alternative solution to training with Gaussian noise for tight robustness bounds. This method also applies to the case when only black-box access to a pretrained classifier $f$ is available. *Carlini et al.* [16] extend the denoised smoothing approach by applying it with state-of-the-art, publicly available denoising models, instead of the custom-trained denoisers proposed in article [12].

# 3 Problem statement

For the purpose of investigating the effectiveness of randomized smoothing as an adversarial defense in inverse problems, we consider a highly ill-conditioned inverse problem. Even minor changes in the input in ill-posed problems can lead to significant errors in the output, thereby highlighting the importance of robustness in deep learning for inverse problems [17]. In this project, we evaluate randomized smoothing on the inverse problem of super-resolution.

## 3.1 Definition

Super-resolution is an ill-posed inverse problem. In super-resolution, the goal is to reconstruct a high resolution image from one or multiple low resolution observations [18]. Single image super-resolution (SISR) refers to the problem where one low resolution observation is provided, such as in the examples provided in Figure 3.1. The ill-posedness of single image super-resolution can be attributed to the existence of multiple possible high resolution outputs for a single low resolution input [18].

Image super-resolution has a diverse range of real-world applications, including medical imaging, surveillance systems, and satellite imagery [19]. Therefore, the robustness of super-resolution techniques is of particular importance. The security and reliability of systems using super-resolution in their key functions depend on the robustness of the super-resolution method applied.

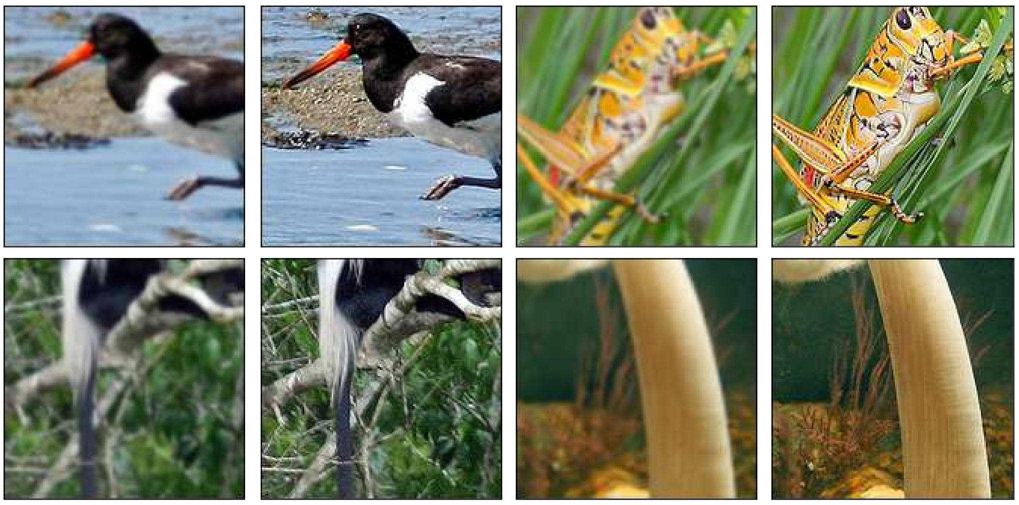


**Figure 3.1** Pairs of low resolution images and their high resolution counterparts from the ImageNet dataset [8].

## 3.2 Deep learning approaches

Methods such as bicubic interpolation and sparse-coding for single image super-resolution have been outperformed by deep-learning-based approaches. The first of such was proposed by *Dong et al.* [20] and involved a three-layer convolutional neural network, the SRCNN [20]. The SRCNN architecture, as depicted in Figure 3.2, first maps a low resolution input image to $n_1$ high level feature maps. From these feature maps, a set of $n_2$ feature maps is extracted. *Dong et al.* [20] interpret the resulting $n_2$ feature maps as a set of high resolution patches, which are then used for reconstruction of the output image in the final convolutional layer. The number of extracted feature maps used in the baseline SRCNN from article [20] amounts to $n_1 = 64$ and $n_2 = 32$. The input and output dimensions are identical, with $n = 3$ channels, which represent the RGB colours.

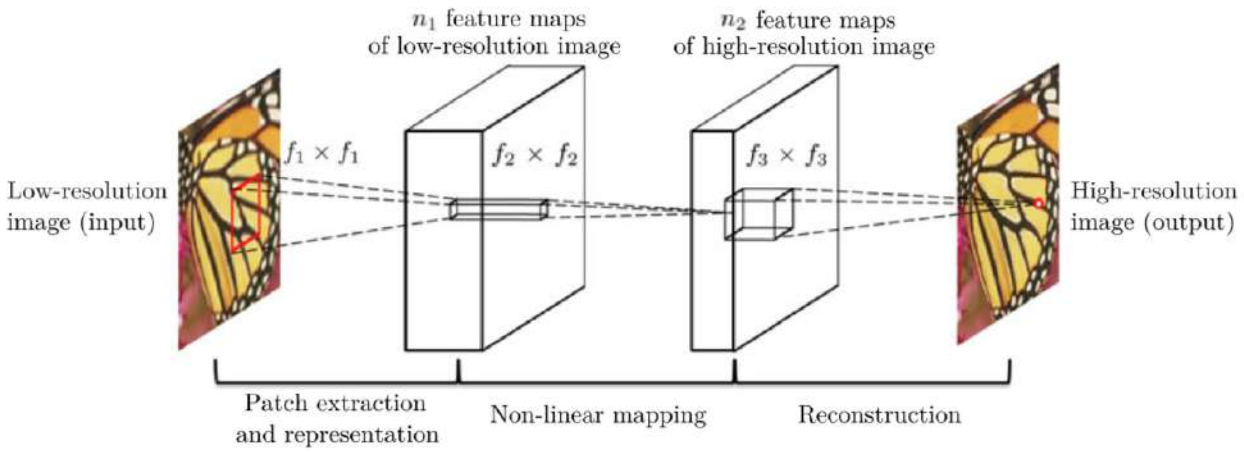

$n_1$ feature maps of low-resolution image
$n_2$ feature maps of high-resolution image
Low-resolution image (input)
$f_1 \times f_1$
$f_2 \times f_2$
$f_3 \times f_3$
High-resolution image (output)
Patch extraction and representation
Non-linear mapping
Reconstruction

**Figure 3.2** The SRCNN architecture [20].

Further high performing and more complex neural networks for super-resolution have been introduced in recent years, including deep convolutional neural networks as well as generative adversarial networks [21, 22]. As demonstrated by article [23], U-Net networks can also be employed for image super-resolution. In this project, we aim to construct a U-Net for single image super-resolution and perform a comparison between the robustness gains yielded by adversarial training and randomized smoothing, respectively.

# 4 Experimental setup

This section describes the experiments conducted as part of this project. The data used for training and evaluating the networks as well as all pre-processing steps are presented first, followed by an overview of the network's architecture and a detailed explanation of the training process for randomized smoothing and adversarial training, respectively. Finally, the procedure of hyperparameter selection is briefly analyzed and a summary and interpretation of the experiments' results are provided.

## 4.1 Training data and pre-processing

The dataset used for training and benchmarking is the Mini-ImageNet-1000 dataset, a subset of the ImageNet [8] dataset, as used by [18, 20] to train convolutional networks for image super-resolution. The training and test sets consist of 34,745 and 3,923 images, respectively. The original images in the dataset are considered high resolution images, also referred to as ground truth images in the following sections.

The model input's dimensions are designed to be the same as the model output's dimensions and, for the experiment described in the following, input images have a fixed size of $160 \times 160$ pixels. Therefore, ground truth images are first cropped to a size of $160 \times 160$ pixels and low resolution images are generated from the cropped ground truth images using bilinear downsampling. In this project, two downsampling settings are considered: a downsampling factor of 2 and 4. The downsampled, now low resolution images are then resized to $160 \times 160$ pixels to match the desired input (and output) dimensions. Further pre-processing steps include normalizing the images using the mean and standard deviation of the Mini-ImageNet-1000 dataset.

## 4.2 Model architecture and training

A summary of the network's structure and design is provided in this section along with an explanation of how it is used to train super-resolution models (i) with standard training (Section 4.2.1), (ii) for randomized smoothing (Section 4.2.2), and (iii) with adversarial training (Section 4.2.3).

### 4.2.1 U-Net for super-resolution

A U-Net is a fully convolutional neural network architecture, which can be divided into two main components: the downsampling and the upsampling component [24]. The combination of both components forms a U-shaped architecture, as visualized in Figure 4.1.

Each step of the downsampling component consists of two convolutional blocks followed by a max pooling layer with a stride of $2$. Every convolutional block is composed of a 2D convolutional layer with zero padding and kernel filters of size $3 \times 3$, a batch normalization, and a ReLU activation layer. Similarly, the upsampling component includes equally many steps, each of them comprised of a deconvolutional layer with $2 \times 2$-sized kernel filters and a stride of $2$, followed by a layer where the feature maps are concatenated with the corresponding feature maps from the downsampling path (skip connection), and two additional 2D convolutional layers with zero padding and kernel filters of size $3 \times 3$.

In standard training, a U-Net with 5 downsampling steps, 5 upsampling steps, and feature channel sizes $(8, 16, 32, 64, 128)$, as can be seen in Figure 4.1, is trained to minimize the objective

$$\sum_{i=1}^{\nu} \|f_\theta(y_i) - x_i\|_2^2 \,, \tag{4.1}$$

where $(x_i, y_i)$ are pairs of ground truth images and their low-resolution counterparts in a dataset of $\nu$ such pairs and $f_\theta$ is the U-Net estimator. The minimization objectives for adversarial training and training for randomized smoothing are different than for standard training and are discussed in Section 4.2.3 and Section 4.2.2, respectively.

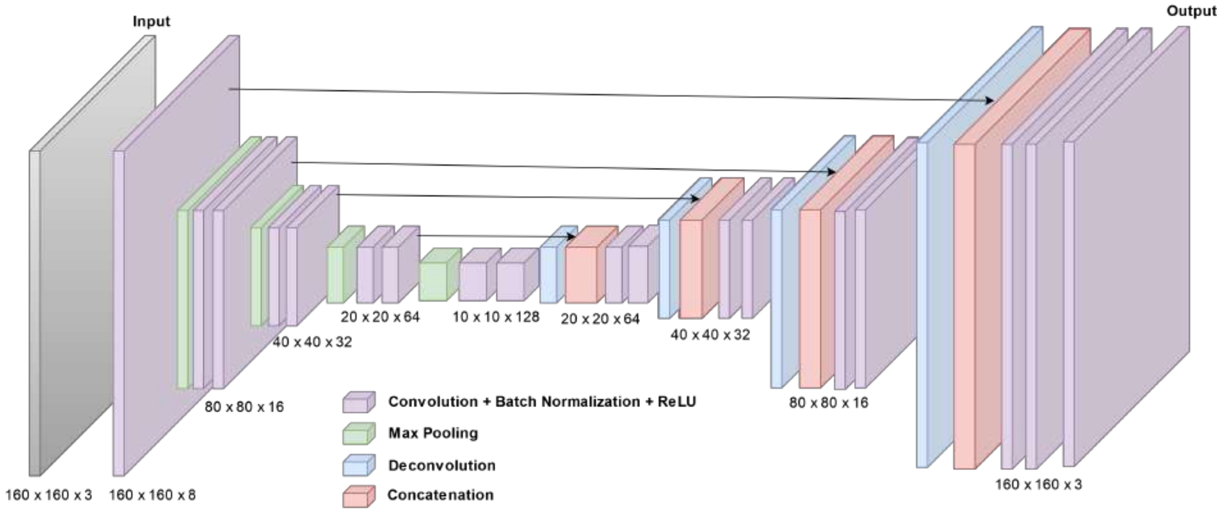


**Figure 4.1** Custom U-Net Architecture for Single Image Super-Resolution. The dimensions noted refer to the last layer in each step.

## 4.2.2 Training for randomized smoothing

As demonstrated in Section 2.1, models trained for randomized smoothing should be trained on noisy inputs. We define the noise used in training of a base estimator $f_\theta$ as $\sigma_f$. Each base model $f_\theta$ is trained to minimize the following objective with respect to the model weights $\theta$:

$$\sum_{i=1}^{\nu} \|f_\theta(y_i + \delta_i) - x_i\|_2^2 \quad \text{where} \quad \delta_i \sim \mathcal{N}(0, \sigma_f^2 I) \,, \tag{4.2}$$

which corresponds to minimizing the mean squared error (MSE) between the noisy predictions $f_\theta(y_i + \delta_i)$ and the ground truth images $x_i$.

One base U-Net estimator $f_\theta$ per downsampling factor $(2, 4)$ was trained for each $\sigma_f \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 1\}$. This range of values for $\sigma_f$ was chosen to demonstrate the effect of $\sigma_f$ on the robustness-accuracy trade-off. For each base U-Net estimator $f_\theta$ trained with a noise setting of $\sigma_f$, the smoothed estimate can be expressed according to Equation (2.5) as

$$g(y) = \arg\min_x \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma_g^2 I)} \left[ \|f_\theta(y + \delta) - x\|_2^2 \right] \,, \tag{4.3}$$

where $\sigma_g$ represents the noise used to approximate the smoothed estimate $g$ according to Equation (2.7). Notably, the optimal noise level $\sigma_g$ for a smoothed estimate $g$ is not assumed to be identical to the noise level $\sigma_f$ used to train the base estimator $f_\theta$. Subsequently, $\sigma_g$ is considered a hyperparameter and is optimized for every base estimator $f_\theta$ individually. Hyperparameter selection is discussed further in Section 4.3.

All base models for randomized smoothing were trained using a learning rate of $10^{-4}$, a batch size of $128$ and the Adam optimizer provided by PyTorch [25] for a total of $50$ epochs.

### 4.2.3 Adversarial training

Adversarial attacks are computed as part of the adversarial training process using the projected gradient descent (PGD) method [5], as described in Appendix A.2, with $\ell_2$-norm adversarial perturbation radius $\epsilon$, which can be formulated as follows:

$$\epsilon = \sqrt{d} \cdot \epsilon_{\text{rel}} \, , \tag{4.4}$$

where $d$ equals the product of the input's dimensions, i.e., $d = c \cdot h \cdot w$ for an input with $c$ channels, each of height $h$ and width $w$. The parameter $\epsilon_{\text{rel}}$ is used in the following sections to describe the level of adversarial noise independently of the input's dimensions.

The optimization objective approximated in adversarial training can be defined as

$$\min_{\theta} \sum_{i=1}^{\nu} \max_{\|e_i\|_2 \le \epsilon} \|f_\theta(y_i + e_i) - x_i\|_2^2 \, , \tag{4.5}$$

where $e_i$ denotes the adversary computed with PGD by attempting to maximize the mean squared error (MSE) between the prediction $f_\theta(y_i + e_i)$ of each perturbed sample and the corresponding ground truth image $x_i$ in a dataset of $\nu$ samples.

Adversarial training was employed to train a model for each downsampling factor $(2, 4)$ and each adversarial noise level $\epsilon_{\text{rel}} \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$. The parameters selected for training are a batch size of $128$ and a learning rate of $10^{-4}$ for a total of $50$ epochs. Moreover, projected gradient descent was computed using code adapted from the *robustness* Python library [26] with random uniform initialization within the $\ell_2$-sphere of radius $\epsilon$ for the starting point, a step size of $\frac{\epsilon}{2}$ and $\gamma = 5$ iterations per adversary calculation.

## 4.3 Hyperparameter selection

For each base U-Net estimator $f_\theta$ trained with noise $\sigma_f$, we optimize the hyperparameter $\sigma_g$, which represents the noise incorporated in the approximation of the smoothed estimate $g$, as expressed in Equation (4.3). We chose the SSIM (Structural Similarity Index) instead of the PSNR (Peak Signal-to-Noise Ratio) as an image quality metric in the hyperparameter optimization, since the SSIM has been shown to better represent the perceived visual quality of images by humans [27]. Therefore, the optimal values for $\sigma_g$ are chosen to be the values that maximize the SSIM between the ground truth images and the smoothed super-resolved output of the models under adversarially perturbed inputs.

For the final evaluation of each smoothed model $\hat{g}$ against adversarial attacks, an optimal $\sigma_g$ value is used for each base estimator $f_\theta$ and each adversarial noise setting $\epsilon_{\text{rel}}$. Visualizations and tables of the optimal values for the hyperparameter $\sigma_g$ can be found in Appendix A.4.

## 4.4 Evaluation

We evaluate (i) models trained with standard training, (ii) models trained with Gaussian noise for randomized smoothing, and (iii) models trained adversarially, on 3,923 images of size $160 \times 160$ comprising the test set from Mini-ImageNet-1000. All models correspond to the U-Net architecture presented in Section 4.2.1. The models are evaluated with a batch size of $64$ samples, $\gamma = 10$ adversarial iterations, a step size of $\frac{\epsilon}{4}$, and random uniform initialization within the $\ell_2$-sphere of radius $\epsilon$ for the calculation of adversarial perturbations, and $k = 50$ noisy predictions for the approximation of the smoothed estimate $\hat{g}$. The MSE, PSNR, and SSIM scores reported in Table 4.1, Table 4.2, Table 4.3, and Table 4.4, are averaged over the entire Mini-ImageNet-1000 test set for each model and attack setting.

In the evaluation process, all models are attacked separately for each adversarial noise setting $\epsilon_{\text{rel}}$. For randomized smoothing, the base models $f_\theta$ are attacked directly instead of the smoothed models $\hat{g}$, since adversarial attacks against the base estimators $f_\theta$ appear to be stronger than attacks against the smoothed models $\hat{g}$, as demonstrated in Appendix A.3. Therefore, the worst-case attack (against the base estimator $f_\theta$) is considered for evaluation of the robustness gains from randomized smoothing in comparison to adversarial training.

To evaluate the smoothed estimate $\hat{g}$, we choose the optimal $\sigma_g$ for each base estimator $f_\theta$ trained for randomized smoothing and each adversarial noise level $\epsilon_{\text{rel}}$, according to Table A.2 and Table A.3 in Appendix A.4.

### 4.4.1 Results

We observe that for a downsampling factor of 2, randomized smoothing with $\sigma_f = 0.1$ and $\sigma_g = 0.15$ yields for small perturbations (see Table 4.1) a model approximately as robust as the ones trained with adversarial noise of that level, i.e., $\epsilon_{\text{rel}} \in \{0.01, 0.02\}$. In the case of larger perturbations (see Table 4.2) randomized smoothing seems to consistently reach higher SSIM scores than adversarial training, whereas adversarial training appears to yield higher PSNR scores.

A qualitative comparison of randomized smoothing to adversarial training for adversarial attacks with $\epsilon_{\text{rel}} = 0.02$ and $\epsilon_{\text{rel}} = 0.05$ can be seen in Figure 4.2 and Figure 4.3, respectively. Randomized smoothing reaches with $\sigma_f = 0.1$ the overall highest SSIM score for $\epsilon_{\text{rel}} = 0.02$ and with $\sigma_f = 0.4$ the overall highest SSIM score for $\epsilon_{\text{rel}} = 0.05$. It is evident that randomized smoothing outputs images of better visual quality, despite the occasional presence of strange artifacts (see third row of images in Figure 4.2).

The results for a downsampling factor of 4 seem to be worse overall, which can be attributed to the increased difficulty of this inverse problem for higher downsampling factors. We observe that for small perturbations (see Table 4.3) standard training yields the highest SSIM scores, whereas randomized smoothing with $\sigma_f = 0.3$ yields the highest PSNR scores. For larger perturbations, especially for $\epsilon_{\text{rel}} \in \{0.1, 0.2\}$ (see Table 4.4), randomized smoothing with $\sigma_f = 1$ outperforms adversarial training based on the provided SSIM scores.

Another general and expected observation is the dependence of the robustness/accuracy trade-off on the choice of $\sigma_f$ (and $\sigma_g$). For example, a model trained with $\sigma_f = 1$ and smoothed with $\sigma_g = 0.9$ for a downsampling factor of $2$ is more robust to large adversarial attacks ($\epsilon_{\text{rel}} = 0.2$, Table 4.2) than all other smoothed models, but is the least accurate in the case of none or smaller adversarial perturbations (see Table 4.1).

**Table 4.1** Evaluation of model performance for a downsampling factor of 2 and adversarial attacks of perturbation level $\epsilon_{\text{rel}} \in \{0, 0.01, 0.02\}$. The best results for each column are marked in bold.

| | | **Original input ($\epsilon_{\text{rel}} = 0$)** | | | $\epsilon_{\text{rel}} = \mathbf{0.01}$ | | | $\epsilon_{\text{rel}} = \mathbf{0.02}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\sigma_f$ | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM |
| *Randomized smoothing* | 0.05 | 0.0030 | 25.34 | 0.8133 | 0.0048 | 23.20 | 0.7657 | 0.0061 | 22.18 | 0.7274 |
| | 0.1 | **0.0028** | **25.55** | 0.8140 | **0.0032** | **25.04** | 0.7976 | **0.0038** | **24.26** | **0.7776** |
| | 0.2 | 0.0033 | 24.87 | 0.7922 | 0.0039 | 24.16 | 0.7716 | 0.0048 | 23.26 | 0.7482 |
| | 0.3 | 0.0033 | 24.88 | 0.7763 | 0.0036 | 24.54 | 0.7647 | 0.0039 | 24.12 | 0.7501 |
| | 0.4 | 0.0034 | 24.76 | 0.7706 | 0.0036 | 24.53 | 0.7622 | 0.0038 | 24.20 | 0.7513 |
| | 0.5 | 0.0037 | 24.33 | 0.7462 | 0.0039 | 24.14 | 0.7390 | 0.0041 | 23.89 | 0.7302 |
| | 1 | 0.0048 | 23.27 | 0.6983 | 0.0049 | 23.17 | 0.6937 | 0.0050 | 23.04 | 0.6884 |
| *Std. tr.* | $\epsilon_{\text{rel}}$ | | | | | | | | | |
| | 0 | 0.0030 | 25.29 | **0.8562** | 0.0052 | 22.87 | 0.7810 | 0.0071 | 21.53 | 0.7159 |
| *Adversarial tr.* | 0.01 | 0.0029 | 25.51 | 0.8399 | 0.0033 | 24.86 | **0.8114** | 0.0040 | 24.02 | 0.7651 |
| | 0.02 | 0.0031 | 25.13 | 0.8242 | 0.0034 | 24.69 | 0.8047 | 0.0039 | 24.14 | 0.7766 |
| | 0.05 | 0.0035 | 24.65 | 0.7906 | 0.0037 | 24.42 | 0.7784 | 0.0039 | 24.15 | 0.7637 |
| | 0.1 | 0.0043 | 23.69 | 0.7432 | 0.0045 | 23.53 | 0.7359 | 0.0047 | 23.36 | 0.7276 |
| | 0.2 | 0.0053 | 22.78 | 0.6949 | 0.0054 | 22.69 | 0.6890 | 0.0056 | 22.58 | 0.6828 |

**Table 4.2** Evaluation of model performance for a downsampling factor of 2 and adversarial attacks of perturbation level $\epsilon_{\text{rel}} \in \{0.05, 0.1, 0.2\}$. The best results for each column are marked in bold.

| | | $\epsilon_{\text{rel}} = \mathbf{0.05}$ | | | $\epsilon_{\text{rel}} = \mathbf{0.1}$ | | | $\epsilon_{\text{rel}} = \mathbf{0.2}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\sigma_f$ | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM |
| *Randomized smoothing* | 0.05 | 0.0098 | 20.11 | 0.6148 | 0.0150 | 18.29 | 0.4545 | 0.0239 | 16.28 | 0.2434 |
| | 0.1 | 0.0064 | 21.94 | 0.7115 | 0.0112 | 19.55 | 0.5868 | 0.0210 | 16.87 | 0.3725 |
| | 0.2 | 0.0069 | 21.66 | 0.6873 | 0.0103 | 19.87 | 0.6080 | 0.0183 | 17.41 | 0.4595 |
| | 0.3 | 0.0054 | 22.72 | 0.7064 | 0.0088 | 20.59 | 0.6364 | 0.0168 | 17.76 | 0.5012 |
| | 0.4 | 0.0052 | 22.89 | **0.7127** | 0.0082 | 20.89 | **0.6475** | 0.0149 | 18.27 | 0.5422 |
| | 0.5 | 0.0051 | 22.95 | 0.6979 | 0.0074 | 21.32 | 0.6402 | 0.0131 | 18.85 | 0.5349 |
| | 1 | 0.0056 | 22.52 | 0.6686 | 0.0072 | 21.44 | 0.6288 | 0.0114 | 19.45 | **0.5495** |
| *Std. tr.* | $\epsilon_{\text{rel}}$ | | | | | | | | | |
| | 0 | 0.0118 | 19.31 | 0.6035 | 0.0187 | 17.30 | 0.5859 | 0.305 | 15.17 | 0.5354 |
| *Adversarial tr.* | 0.01 | 0.0067 | 21.77 | 0.6200 | 0.0118 | 19.30 | 0.4763 | 0.0245 | 16.13 | 0.3595 |
| | 0.02 | 0.0060 | 22.24 | 0.6714 | 0.0099 | 20.07 | 0.5484 | 0.0179 | 17.49 | 0.4012 |
| | 0.05 | **0.0049** | **23.17** | 0.7035 | 0.0073 | 21.38 | 0.5735 | 0.0145 | 18.39 | 0.3816 |
| | 0.1 | 0.0054 | 22.75 | 0.6981 | **0.0070** | **21.62** | 0.6311 | 0.0113 | 19.49 | 0.4928 |
| | 0.2 | 0.0062 | 22.15 | 0.6633 | 0.0075 | 21.29 | 0.6290 | **0.0113** | **19.53** | 0.5493 |

**Table 4.3** Evaluation of model performance for a downsampling factor of 4 and adversarial attacks of perturbation level $\epsilon_{\text{rel}} \in \{0, 0.01, 0.02\}$. The best results for each column are marked in bold.

| | | **Original input ($\epsilon_{\text{rel}} = 0$)** | | | $\epsilon_{\text{rel}} = \mathbf{0.01}$ | | | $\epsilon_{\text{rel}} = \mathbf{0.02}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\sigma_f$ | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM |
| *Randomized smoothing* | 0.05 | **0.0052** | **22.90** | 0.6715 | 0.0059 | 22.37 | 0.6451 | 0.0066 | 21.84 | 0.6129 |
| | 0.1 | 0.0053 | 22.85 | 0.6738 | 0.0059 | 22.34 | 0.6514 | 0.0067 | 21.75 | 0.6223 |
| | 0.2 | 0.0058 | 22.40 | 0.6611 | 0.0065 | 21.90 | 0.6434 | 0.0074 | 21.36 | 0.6218 |
| | 0.3 | 0.0054 | 22.72 | 0.6553 | **0.0058** | **22.43** | 0.6430 | **0.0063** | **22.04** | 0.6262 |
| | 0.4 | 0.0056 | 22.53 | 0.6508 | 0.0060 | 22.30 | 0.6396 | 0.0064 | 21.97 | 0.6248 |
| | 0.5 | 0.0059 | 22.33 | 0.6332 | 0.0062 | 22.09 | 0.6258 | 0.0066 | 21.83 | 0.6144 |
| | 1 | 0.0063 | 22.03 | 0.6113 | 0.0065 | 21.89 | 0.6053 | 0.0068 | 21.72 | 0.5979 |
| *Std. tr.* | $\epsilon_{\text{rel}}$ | | | | | | | | | |
| | 0 | 0.0056 | 22.57 | **0.7115** | 0.0077 | 21.20 | **0.7017** | 0.0090 | 20.50 | **0.6668** |
| *Adversarial tr.* | 0.01 | 0.0056 | 22.55 | 0.7009 | 0.0061 | 22.17 | 0.6850 | 0.0068 | 21.69 | 0.6612 |
| | 0.02 | 0.0059 | 22.38 | 0.6851 | 0.0062 | 22.14 | 0.6733 | 0.0066 | 21.83 | 0.6559 |
| | 0.05 | 0.0065 | 21.91 | 0.6633 | 0.0068 | 21.76 | 0.6561 | 0.0070 | 21.57 | 0.6470 |
| | 0.1 | 0.0069 | 21.64 | 0.6284 | 0.0071 | 21.54 | 0.6183 | 0.0073 | 21.40 | 0.6051 |
| | 0.2 | 0.0074 | 21.35 | 0.6338 | 0.0077 | 21.16 | 0.6191 | 0.0080 | 21.02 | 0.6040 |

**Table 4.4** Evaluation of model performance for a downsampling factor of 4 and adversarial attacks of perturbation level $\epsilon_{\text{rel}} \in \{0.05, 0.1, 0.2\}$. The best results for each column are marked in bold.

| | | $\epsilon_{\text{rel}} = \mathbf{0.05}$ | | | $\epsilon_{\text{rel}} = \mathbf{0.1}$ | | | $\epsilon_{\text{rel}} = \mathbf{0.2}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\sigma_f$ | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM | MSE | PSNR (dB) | SSIM |
| *Randomized smoothing* | 0.05 | 0.0091 | 20.47 | 0.5261 | 0.0135 | 18.80 | 0.4077 | 0.0228 | 16.63 | 0.2394 |
| | 0.1 | 0.0092 | 20.40 | 0.5467 | 0.0139 | 18.63 | 0.4464 | 0.0226 | 16.57 | 0.3038 |
| | 0.2 | 0.0104 | 19.87 | 0.5626 | 0.0160 | 17.99 | 0.4755 | 0.0239 | 16.30 | 0.3721 |
| | 0.3 | 0.0082 | 20.87 | 0.5720 | 0.0121 | 19.22 | 0.4969 | 0.0199 | 17.08 | 0.3906 |
| | 0.4 | 0.0083 | 20.82 | 0.5739 | 0.0119 | 19.26 | 0.5057 | 0.0197 | 17.12 | 0.4108 |
| | 0.5 | 0.0082 | 20.88 | 0.5727 | 0.0115 | 19.40 | 0.5096 | 0.0184 | 17.38 | 0.4249 |
| | 1 | **0.0079** | **21.04** | 0.5701 | 0.0103 | 19.87 | **0.5219** | 0.0163 | 17.88 | **0.4472** |
| *Std. tr.* | $\epsilon_{\text{rel}}$ | | | | | | | | | |
| | 0 | 0.0122 | 19.15 | 0.5819 | 0.0173 | 17.64 | 0.4929 | 0.0271 | 15.68 | 0.3825 |
| *Adversarial tr.* | 0.01 | 0.0095 | 20.25 | 0.5790 | 0.0138 | 18.61 | 0.4660 | 0.0213 | 16.72 | 0.3510 |
| | 0.02 | 0.0086 | 20.70 | 0.5858 | 0.0125 | 19.04 | 0.4702 | 0.0206 | 16.87 | 0.3464 |
| | 0.05 | 0.0082 | 20.91 | **0.6003** | 0.0109 | 19.67 | 0.4917 | 0.0181 | 17.43 | 0.3307 |
| | 0.1 | 0.0082 | 20.93 | 0.5580 | **0.0100** | **20.05** | 0.4851 | 0.0147 | 18.33 | 0.3616 |
| | 0.2 | 0.0087 | 20.64 | 0.5614 | 0.0101 | 19.97 | 0.4982 | **0.0141** | **18.52** | 0.3938 |

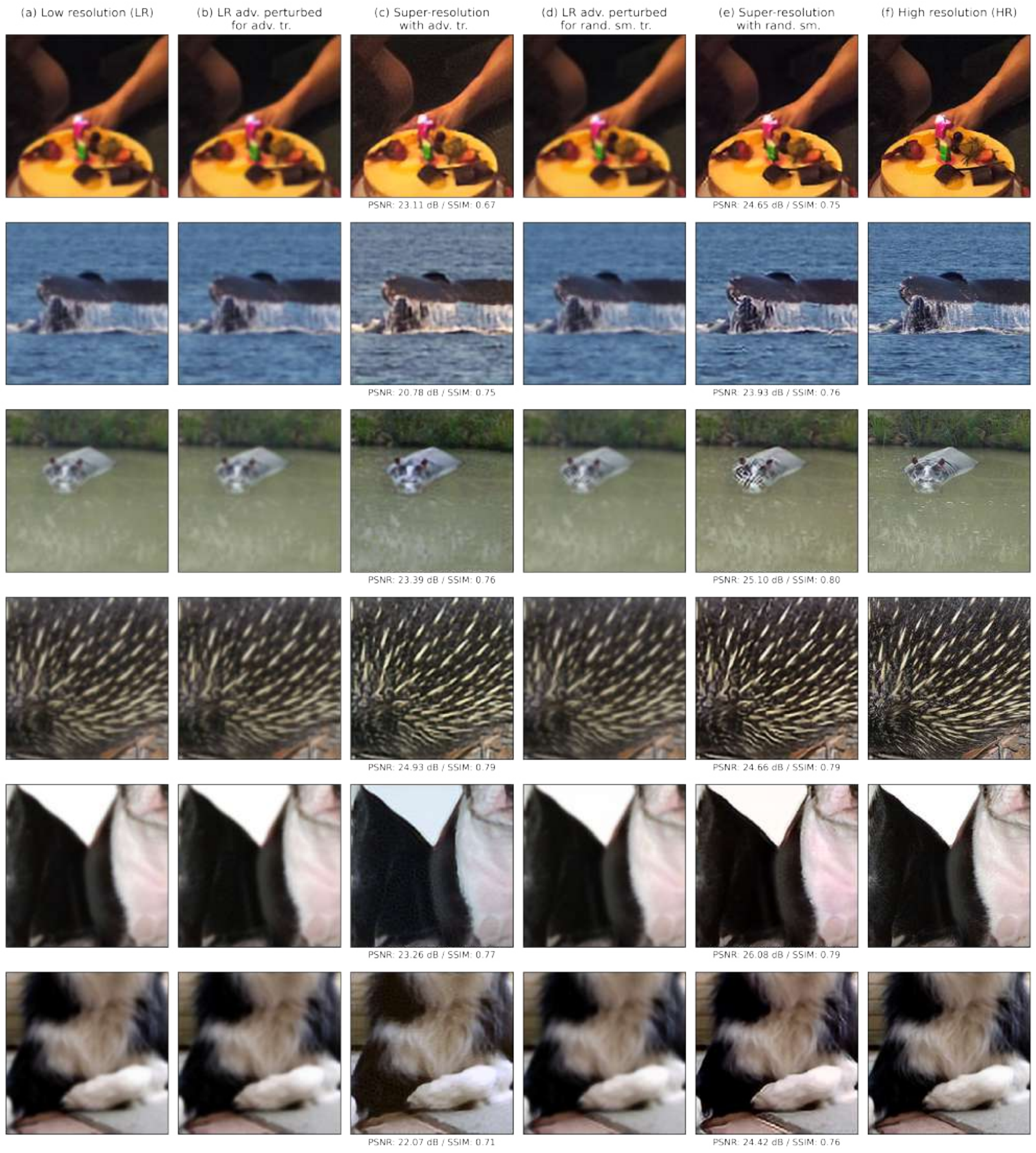

(a) Low resolution (LR)
(b) LR adv. perturbed for adv. tr.
(c) Super-resolution with adv. tr.
(d) LR adv. perturbed for rand. sm. tr.
(e) Super-resolution with rand. sm.
(f) High resolution (HR)
PSNR: 23.11 dB / SSIM: 0.67
PSNR: 24.65 dB / SSIM: 0.75
PSNR: 20.78 dB / SSIM: 0.75
PSNR: 23.93 dB / SSIM: 0.76
PSNR: 23.39 dB / SSIM: 0.76
PSNR: 25.10 dB / SSIM: 0.80
PSNR: 24.93 dB / SSIM: 0.79
PSNR: 24.66 dB / SSIM: 0.79
PSNR: 23.26 dB / SSIM: 0.77
PSNR: 26.08 dB / SSIM: 0.79
PSNR: 22.07 dB / SSIM: 0.71
PSNR: 24.42 dB / SSIM: 0.76

**Figure 4.2** (a) Low resolution images (downsampling factor of 2), (b) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.02$ to attack a model trained adversarially with $\epsilon_{\text{rel}} = 0.02$, (c) Super-resolution output images of attack on adversarially trained model, (d) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.02$ to attack a model trained with Gaussian noise $\sigma_f = 0.1$ for randomized smoothing, (e) Super-resolution output images of smoothed model for $\sigma_f = 0.1$ and $\sigma_g = 0.15$, (f) High resolution (ground truth) images.

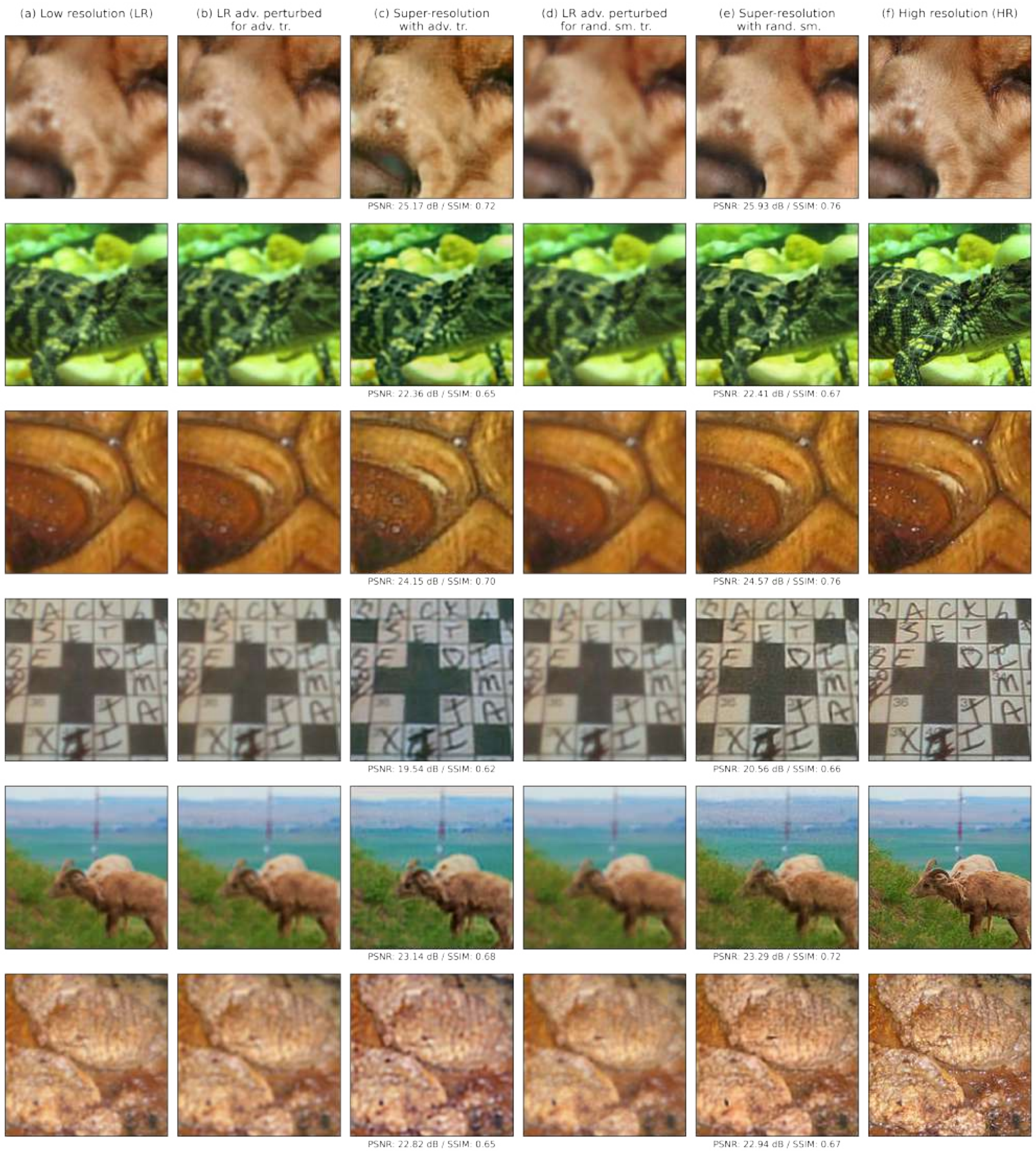

(a) Low resolution (LR)
(b) LR adv. perturbed for adv. tr.
(c) Super-resolution with adv. tr.
(d) LR adv. perturbed for rand. sm. tr.
(e) Super-resolution with rand. sm.
(f) High resolution (HR)
PSNR: 25.17 dB / SSIM: 0.72
PSNR: 25.93 dB / SSIM: 0.76
PSNR: 22.36 dB / SSIM: 0.65
PSNR: 22.41 dB / SSIM: 0.67
PSNR: 24.15 dB / SSIM: 0.70
PSNR: 24.57 dB / SSIM: 0.76
PSNR: 19.54 dB / SSIM: 0.62
PSNR: 20.56 dB / SSIM: 0.66
PSNR: 23.14 dB / SSIM: 0.68
PSNR: 23.29 dB / SSIM: 0.72
PSNR: 22.82 dB / SSIM: 0.65
PSNR: 22.94 dB / SSIM: 0.67

**Figure 4.3** (a) Low resolution images (downsampling factor of 2), (b) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.05$ to attack a model trained adversarially with $\epsilon_{\text{rel}} = 0.05$, (c) Super-resolution output images of attack on adversarially trained model, (d) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.05$ to attack a model trained with Gaussian noise $\sigma_f = 0.4$ for randomized smoothing, (e) Super-resolution output images of smoothed model for $\sigma_f = 0.4$ and $\sigma_g = 0.4$, (f) High resolution (ground truth) images.

# 5 Conclusion

In this project, the effectiveness of randomized smoothing as an adversarial defense mechanism in image super-resolution is evaluated by training multiple U-Net models for single image super-resolution and comparing randomized smoothing to adversarial training in terms of robustness against a range of adversarial attacks. The experimental results indicate that randomized smoothing in super-resolution, and potentially also in other similar image reconstruction problems, yields robust estimators, which achieve better visual quality in reconstructed images than their adversarially trained counterparts especially for large adversarial perturbations.

# A Appendix

## A.1 Proofs for randomized smoothing

In Section 2.2 we adapt the randomized smoothing method from classification to inverse problems. Under the formulation in Equation (1.1), in an inverse problem we consider the following relation between an input $x$ and a measurement $y$:

$$y = \mathcal{A}(x) + z \,, \tag{A.1}$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $\mathcal{A}$ is a forward, typically non-invertible transformation, and $z \in \mathbb{R}^m$ is a noise vector [11]. For the proofs in Section A.1.1 and Section A.1.2 we consider a function $f : \mathbb{R}^m \to \mathbb{R}^n$, which maps a measurement $y$ to a reconstruction $f(y)$ of the input $x$.

### A.1.1 Alternative formulation of the smoothed estimate $g$

**Lemma A.1.1.** *The smoothed estimate $g : \mathbb{R}^m \to \mathbb{R}^n$ defined in Equation (2.5) can be formulated equivalently as $g(y) = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)}[f(y+\delta)]$.*

*Proof.* Recall the definition of the smoothed estimate in Equation (2.5):

$$g(y) = \arg\min_x \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ \|f(y+\delta) - x\|_2^2 \right] . \tag{A.2}$$

Let

$$q(x) = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ \|f(y+\delta) - x\|_2^2 \right] \tag{A.3}$$

be the objective function for minimization. A minimum value is attained for $\nabla_x q = 0$ (first order necessary condition):

$$\nabla_x q = 0 \;\Leftrightarrow\; \nabla_x \Big( \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ \|f(y+\delta) - x\|_2^2 \right] \Big) = 0 \,, \tag{A.4}$$

which can be expressed as

$$\nabla_x \Big( \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ \|f(y+\delta)\|_2^2 \right] - 2 \cdot \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ x^T f(y+\delta) \right] + \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ \|x\|_2^2 \right] \Big) = 0 \,. \tag{A.5}$$

The term $x$ is independent of $\delta \sim \mathcal{N}(0, \sigma^2 I)$, from which follows

$$\nabla_x \Big( \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ \|f(y+\delta)\|_2^2 \right] - 2 \cdot x^T \cdot \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ f(y+\delta) \right] + \|x\|_2^2 \Big) = 0 \,. \tag{A.6}$$

The calculation of the gradient yields

$$-2 \cdot \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ f(y+\delta) \right] + 2x = 0 \;\Leftrightarrow\; x = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ f(y+\delta) \right] , \tag{A.7}$$

and since $\nabla_x^2 q = 2I$, the second order sufficient condition $\nabla_x^2 q(\hat{x}) \succ 0$ is fulfilled for a local minimum $\hat{x}$. From Equation (A.2) follows

$$g(y) = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[ f(y+\delta) \right] . \tag{A.8}$$

□

### A.1.2 Linear base estimator $f$

**Lemma A.1.2.** *For a linear base estimator $f(y) = Qy$ with $Q \in \mathbb{R}^{n\times m}$, the smoothed estimate $g$ and the base estimator $f$ are equal with $g(y) = f(y)$.*

*Proof.* Recall the definition of the smoothed estimate in Equation (2.6):

$$g(y) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[f(y+\delta)]\,. \tag{A.9}$$

For $f(y) = Qy$ the smoothed estimate can be written as

$$g(y) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[Qy + Q\delta]\,, \tag{A.10}$$

and from the linearity of the expectation function follows

$$g(y) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[Qy] + \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[Q\delta]\,, \tag{A.11}$$

which can be simplified to

$$g(y) = Qy + Q\cdot\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[\delta]\,, \tag{A.12}$$

since the term $Qy$ is independent of $\delta \sim \mathcal{N}(0,\sigma^2 I)$. From $f(y) = Qy$ and $\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[\delta] = 0$ follows

$$g(y) = f(y)\,. \tag{A.13}$$

□

## A.2 Projected gradient descent

The adversarial attacks in training and evaluation are performed using the projected gradient descent (PGD) method as defined in article [5]. Recall the optimization objective in adversarial training, as expressed in Equation (4.5):

$$\min_\theta \sum_{i=1}^{\nu} \max_{\|e_i\|_2\le\epsilon} \|f_\theta(y_i+e_i) - x_i\|_2^2\,. \tag{A.14}$$

Projected gradient descent is employed with the goal to approximate the inner maximization problem in Equation (A.14). The computation of an adversarial example $\check{y}_i = y_i + e_i$ follows the multi-step scheme

$$\check{y}_i^{\,t+1} = \Pi_{y_i+B}\left(\check{y}_i^{\,t} + \alpha\frac{\nabla_{\check{y}_i^{\,t}}(\|f_\theta(\check{y}_i^{\,t}) - x_i\|_2^2)}{\left\|\nabla_{\check{y}_i^{\,t}}(\|f_\theta(\check{y}_i^{\,t}) - x_i\|_2^2)\right\|_2}\right), \tag{A.15}$$

where $\mathbf{B} = \{e_i \,|\, \|e_i\|_2 < \epsilon\}$ is a constraint set for the perturbations, $\alpha$ is the step size, $t$ is the descent step, and $\check{y}_i^{\,t+1}$ is the adversarial example computed in step $t+1$ [5]. The starting point of the iteration scheme is $\check{y}_i^{\,0} = y_i + e_i^{\,0}$, where $e_i^{\,0}$ is initialized randomly within the $\ell_2$-sphere of radius $\epsilon$.

## A.3 Adversarial attacks against the smoothed model $g$

For the final evaluation in Section 4.4, the base U-Net estimator $f_\theta$ was attacked adversarially instead of the approximated smoothed estimate $\hat{g}$. The attack on the base U-Net estimator is referred to as *Vanilla PGD attack*, as in article [7], whereas the attack on the smoothed model is named *RS PGD attack*. The comparison was performed with $k = 10$ vectors sampled for the calculation of the approximated smoothed estimate $\hat{g}(y_\phi)$ for each sample $y_\phi$ of 1,000 samples from the Mini-ImageNet-1000 test set. Attacking the smoothed model $\hat{g}$ with this sampling setting and $\gamma = 10$ adversarial iterations for the calculation of each corresponding adversary did not yield stronger adversaries than directly attacking the base estimator $f_\theta$.

This comparison was performed using the same attack settings for the adversarial attack as in the evaluation performed in Section 4.4.

As can be seen in Figure A.1 and Figure A.2, both PSNR and SSIM values of the randomized smoothing model's output are lower when the base U-Net model $f_\theta$ is attacked, i.e., in the *Vanilla PGD attack*. This observation suggests that the *Vanilla PGD attack* yields stronger adversaries in this experiment, which is unexpected, since adversarial attacks against a smoothed model $\hat{g}$ should in principle generate stronger adversaries than an attack on a different model, such as the base estimator $f_\theta$. Therefore, we perform a sanity check to investigate why this behaviour occurs. For this sanity check we compute the norm of the adversarial perturbations found with the PGD method ($e_f$ in the *Vanilla PGD attack* and $e_g$ in the *RS PGD attack*) and compare them to the $\epsilon$ we chose for the attack. The results of this sanity check can be seen in Table A.1. We observe that $\|e_f\|_2 \approx \epsilon$ and $\|e_g\|_2 \approx \epsilon$, which means that the adversarial perturbations computed correspond to the $\epsilon$ chosen for the adversarial attack. Therefore, we conclude that the perturbations computed with PGD lie at the boundary of the $\ell_2$-bounded search space, but the adversary found in the *RS PGD attack* is not strong enough.

**Table A.1 Size of adversarial perturbations** computed with PGD in the *Vanilla PGD attack* ($e_f$) and the *RS PGD attack* ($e_g$).

| $\epsilon_{\text{rel}}$ | $\epsilon$ | $\|e_f\|_2$ | $\|e_g\|_2$ |
|---|---|---|---|
| 0.01 | 2.77128129 | 2.77128120 | 2.77128121 |
| 0.05 | 13.85640646 | 13.85640583 | 13.85640606 |

The possibility of stronger adversarial attacks against the smoothed model $\hat{g}$ cannot be eliminated for a higher number $k$ of noise vectors sampled and for more adversarial iterations $\gamma$ in the computation of adversaries via PGD or via a custom attack method, such as *SmoothAdv* proposed by [7] specifically for attacks on smoothed classifiers.

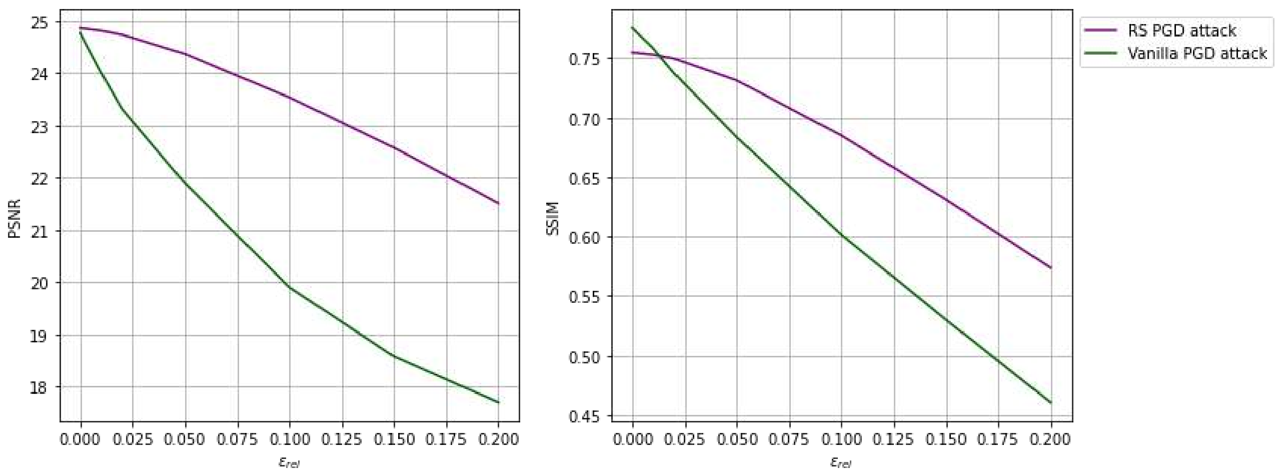


**Figure A.1** Adversarial PSNR and SSIM values for randomized smoothing with $\sigma_f = 0.2$ and $\sigma_g = 0.2$ under adversarial attacks against the smoothed model (*RS PGD attack*) and against the base U-Net model (*Vanilla PGD attack*).

## A.4 Hyperparameter optimization

The optimal values for $\sigma_g$ are chosen from a set of discrete values $\sigma_g \in \mathbf{M}$, where $\mathbf{M} = \{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.9, 1, 1.1, 1.2\}$. In the optimization process, we perform ad-

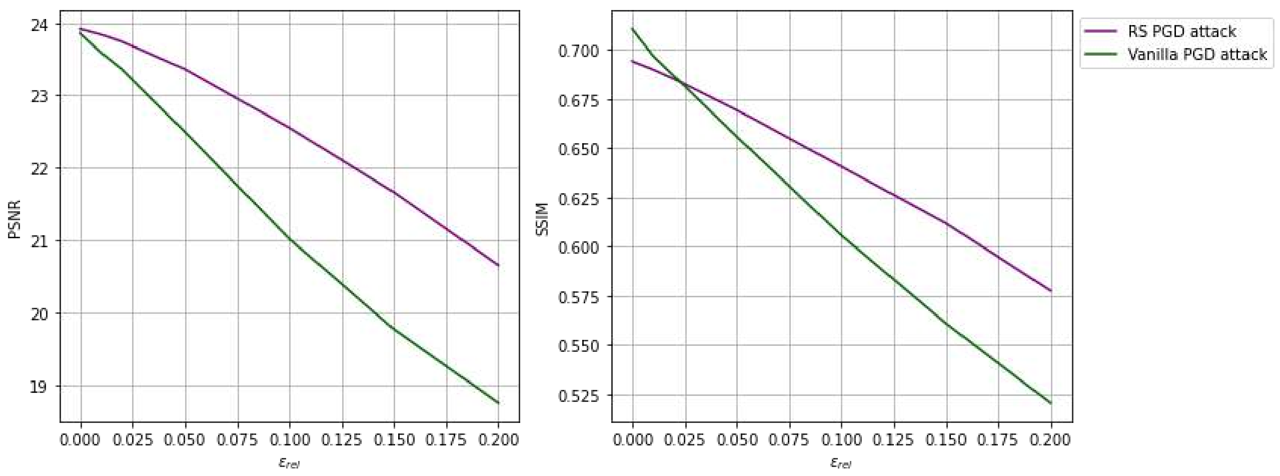


**Figure A.2** Adversarial PSNR and SSIM values for randomized smoothing with $\sigma_f = 0.5$ and $\sigma_g = 0.5$ under adversarial attacks against the smoothed model (*RS PGD attack*) and against the base U-Net model (*Vanilla PGD attack*).

versarial attacks for $\epsilon_{\text{rel}} \in \{0, 0.01, 0.02, 0.035, 0.05, 0.1, 0.2\}$ and for each attack, the smoothed estimate $g$ is approximated for all values $\sigma_g \in \mathbf{M}$. After computing the approximation $\hat{g}$ of the smoothed estimate for all possible $\sigma_g$, according to Equation (2.7), we calculate the (adversarial) PSNR and (adversarial) SSIM between the super-resolved smoothed output $\hat{g}(y_i)$ and the ground truth images $x_i$. The optimal $\sigma_g$ selected is the $\sigma_g$ value used to compute the approximated smoothed estimate $\hat{g}$ that yields the highest SSIM value after an attack with a certain $\epsilon_{\text{rel}}$. These optimal values are represented by the peak of each curve in the plots for the adversarial SSIM scores in Figure A.3, Figure A.4, Figure A.5, Figure A.6, Figure A.7, and Figure A.8. Table A.2 and Table A.3 show the optimal values for the hyperparameter $\sigma_g$ for each base U-Net model trained with Gaussian noise of level $\sigma_f$ and attacked with adversarial noise of level $\epsilon_{\text{rel}}$.

## A.5 Evaluation metrics

In the experiments performed as part of this project, the quality of super-resolution images is measured with the MSE (Mean Squared Error), PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index) metrics [27]. The MSE between a ground truth image $x$ and a super-resolution output image $\hat{x}$, both with dimensions $n \times n$, can be defined as

$$MSE = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (x_{i,j} - \hat{x}_{i,j})^2 \,. \tag{A.16}$$

The PSNR can be calculated with the MSE as follows:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_x^2}{MSE} \right) , \tag{A.17}$$

where $MAX_x$ is the highest possible pixel value [27].

The SSIM between a ground truth image $x$ and a super-resolution output image $\hat{x}$, both with dimensions $n \times n$, can be defined as

$$SSIM = \frac{(2\mu_x \mu_{\hat{x}} + c_1)(2\sigma_x \sigma_{\hat{x}} + c_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + c_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + c_2)} , \tag{A.18}$$

where $\mu_x$ is the pixel sample mean of $x$, $\mu_{\hat{x}}$ is the pixel sample mean of $\hat{x}$, $\sigma_x^2$ is the variance of $x$, $\sigma_{\hat{x}}^2$ is the variance of $\hat{x}$, and $c_1$, $c_2$ are constants necessary for the stabilization of the division [27].

**Table A.2 Optimal values of hyperparameter $\sigma_g$** for base U-Net models trained for a downsampling factor of 2 with different levels of Gaussian noise $\sigma_f$ and attacked with adversarial noise $\epsilon_{\text{rel}}$.

| | | Adversarial noise | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\sigma_f$ \ $\epsilon_{\text{rel}}$ | 0 | 0.01 | 0.02 | 0.035 | 0.05 | 0.1 | 0.2 |
| Base model trained with | 0.05 | 0.1 | 0.1 | 0.1 | 0 | 0 | 0 | 0 |
| | 0.1 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.2 | 0 |
| | 0.2 | 0.25 | 0.25 | 0.25 | 0.25 | 0.2 | 0.2 | 0 |
| | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.4 | 0.4 | 0 |
| | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.5 | 0.5 |
| | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.4 |
| | 1 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 |

**Table A.3 Optimal values of hyperparameter $\sigma_g$** for base U-Net models trained for a downsampling factor of 4 with different levels of Gaussian noise $\sigma_f$ and attacked with adversarial noise $\epsilon_{\text{rel}}$.

| | | Adversarial noise | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\sigma_f$ \ $\epsilon_{\text{rel}}$ | 0 | 0.01 | 0.02 | 0.035 | 0.05 | 0.1 | 0.2 |
| Base model trained with | 0.05 | 0.1 | 0.15 | 0.2 | 0.3 | 0.3 | 0.4 | 0.5 |
| | 0.1 | 0.15 | 0.2 | 0.25 | 0.3 | 0.4 | 0.3 | 0.15 |
| | 0.2 | 0.25 | 0.25 | 0.25 | 0.3 | 0.3 | 0 | 0 |
| | 0.3 | 0.4 | 0.4 | 0.4 | 0.4 | 0.5 | 0.5 | 0.5 |
| | 0.4 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.6 | 0.6 |
| | 0.5 | 0.5 | 0.6 | 0.6 | 0.6 | 0.6 | 0.6 | 0.6 |
| | 1 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 1 | 1.1 |

## A.6 Additional visual results

Further examples for visual comparison are provided in Figure A.9, Figure A.10, and Figure A.11.

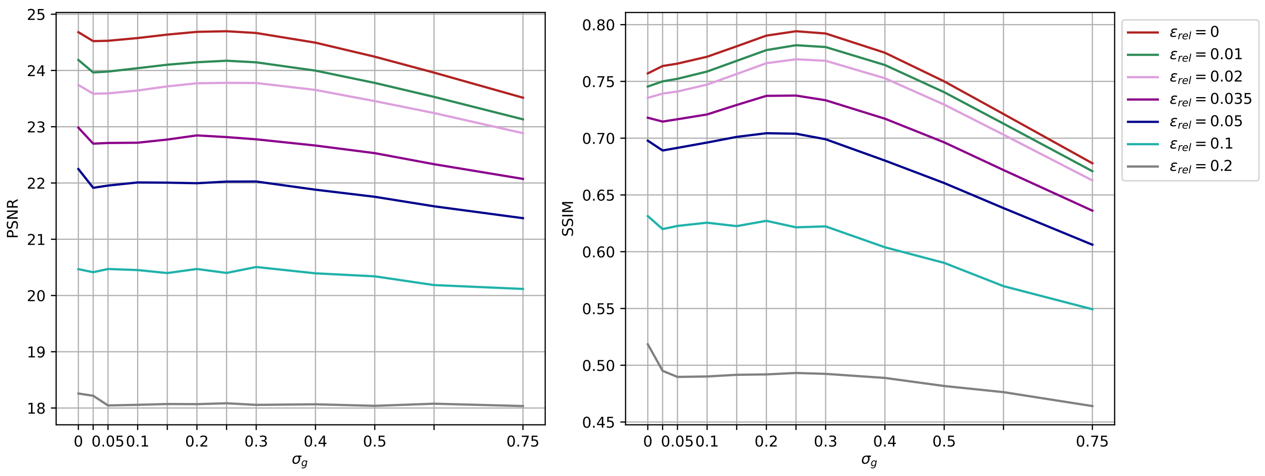


**Figure A.3** Adversarial PSNR and SSIM values for a base U-Net estimator (downsampling factor of $2$) trained with $\sigma_f = 0.2$ and smoothed with $\sigma_g$ (x-axis) for a range of adversarial attacks of level $\epsilon_{\text{rel}}$ .

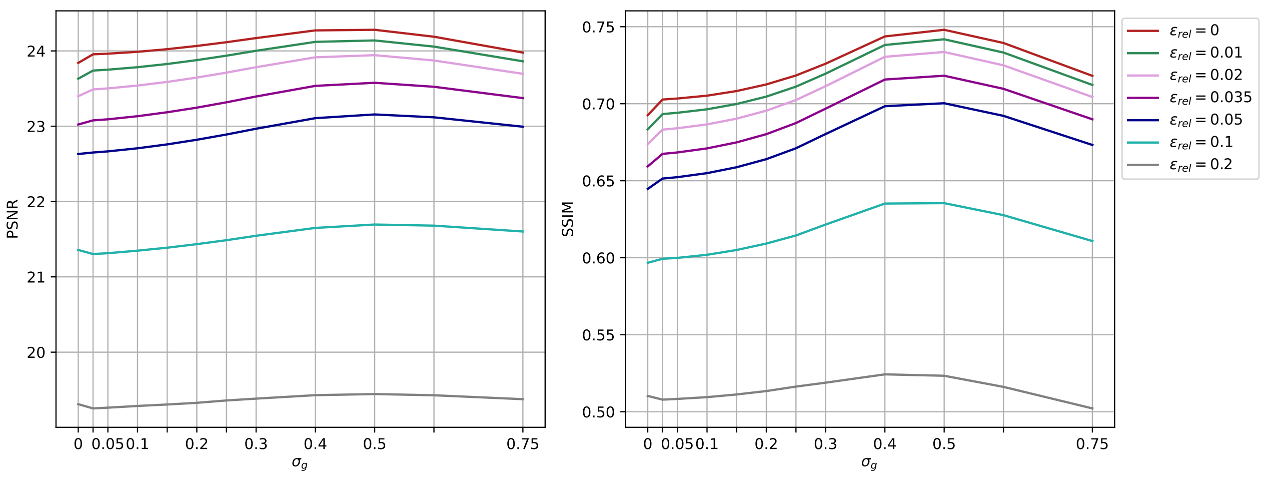


**Figure A.4** Adversarial PSNR and SSIM values for a base U-Net estimator (downsampling factor of $2$) trained with $\sigma_f = 0.5$ and smoothed with $\sigma_g$ (x-axis) for a range of adversarial attacks of level $\epsilon_{\text{rel}}$ .

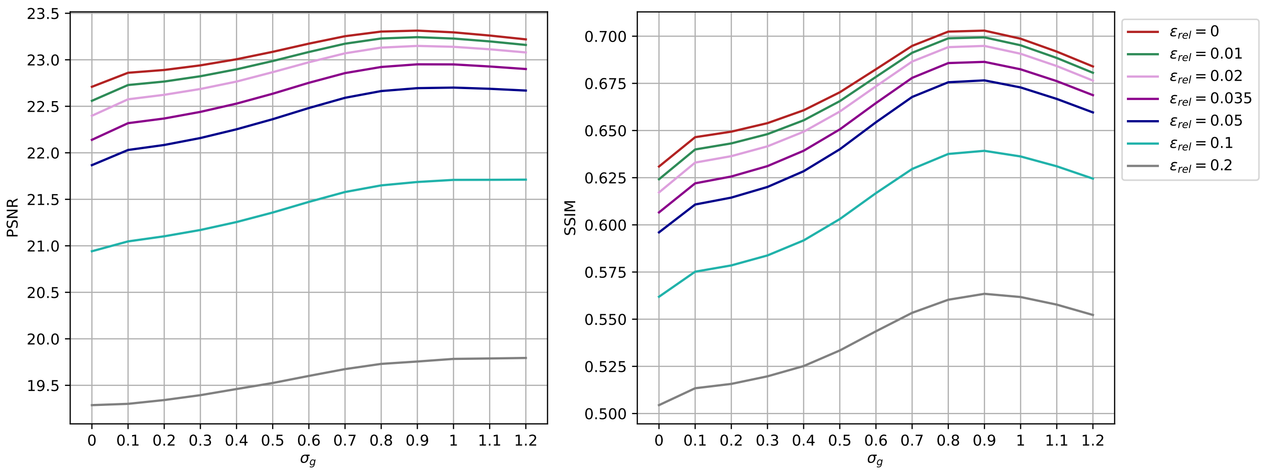


**Figure A.5** Adversarial PSNR and SSIM values for a base U-Net estimator (downsampling factor of $2$) trained with $\sigma_f = 1$ and smoothed with $\sigma_g$ (x-axis) for a range of adversarial attacks of level $\epsilon_{\text{rel}}$ .

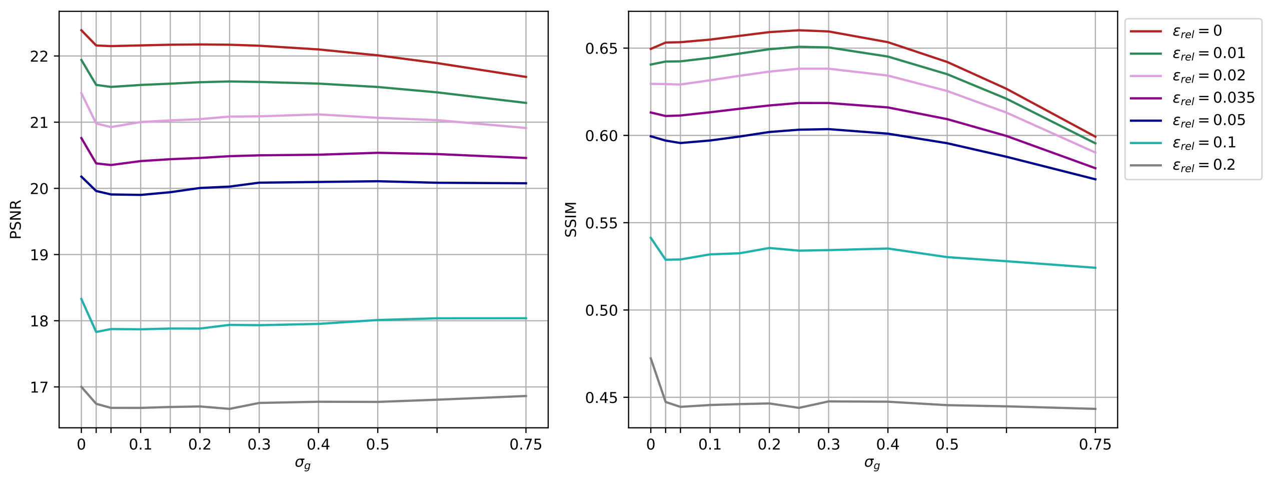


**Figure A.6** Adversarial PSNR and SSIM values for a base U-Net estimator (downsampling factor of $4$) trained with $\sigma_f = 0.2$ and smoothed with $\sigma_g$ (x-axis) for a range of adversarial attacks of level $\epsilon_{\text{rel}}$ .

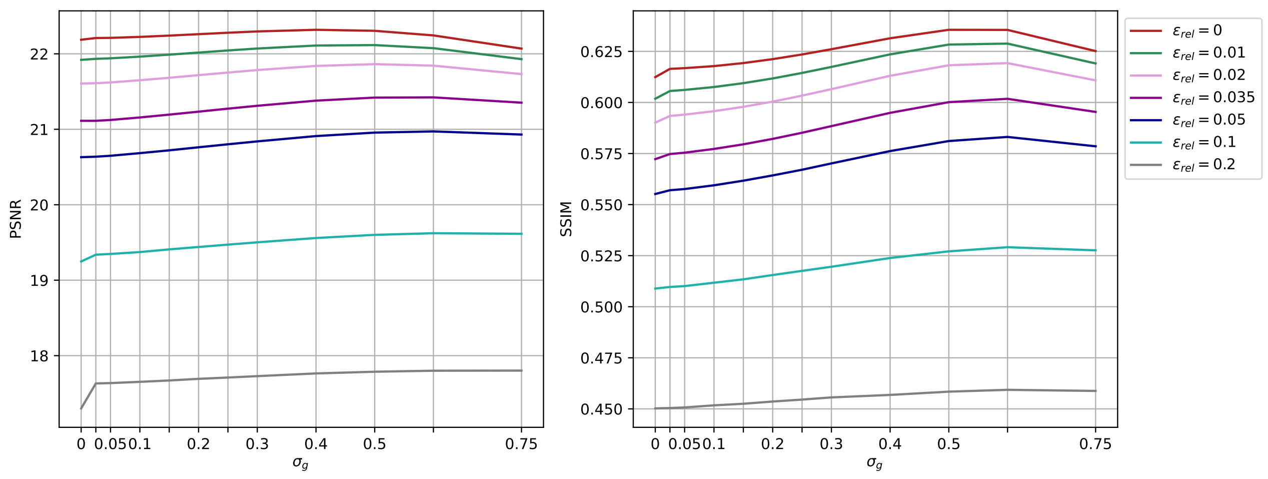


**Figure A.7** Adversarial PSNR and SSIM values for a base U-Net estimator (downsampling factor of $4$) trained with $\sigma_f = 0.5$ and smoothed with $\sigma_g$ (x-axis) for a range of adversarial attacks of level $\epsilon_{\text{rel}}$ .

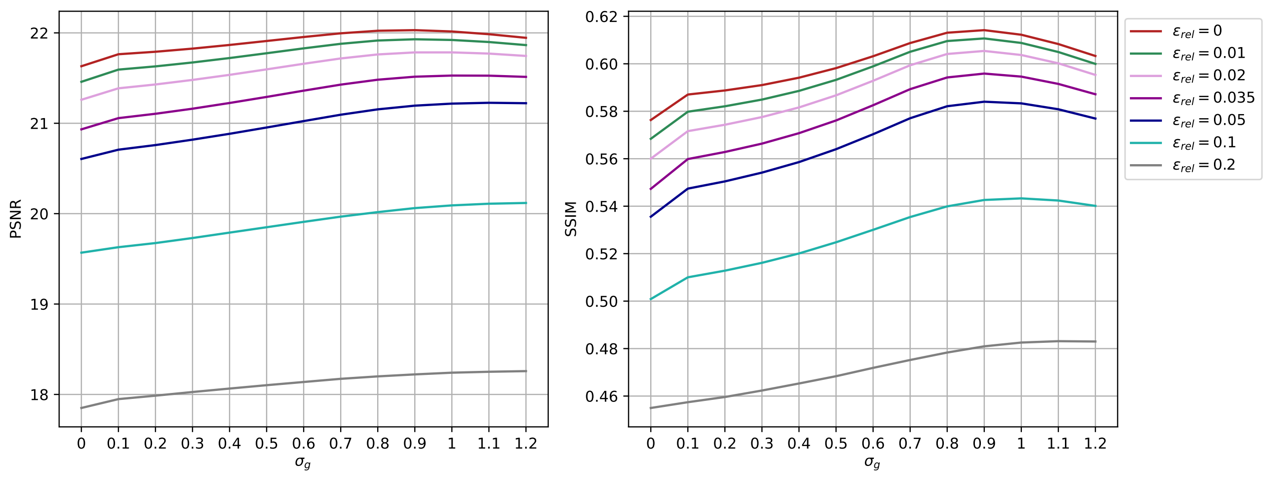


**Figure A.8** Adversarial PSNR and SSIM values for a base U-Net estimator (downsampling factor of $4$) trained with $\sigma_f = 1$ and smoothed with $\sigma_g$ (x-axis) for a range of adversarial attacks of level $\epsilon_{\text{rel}}$ .

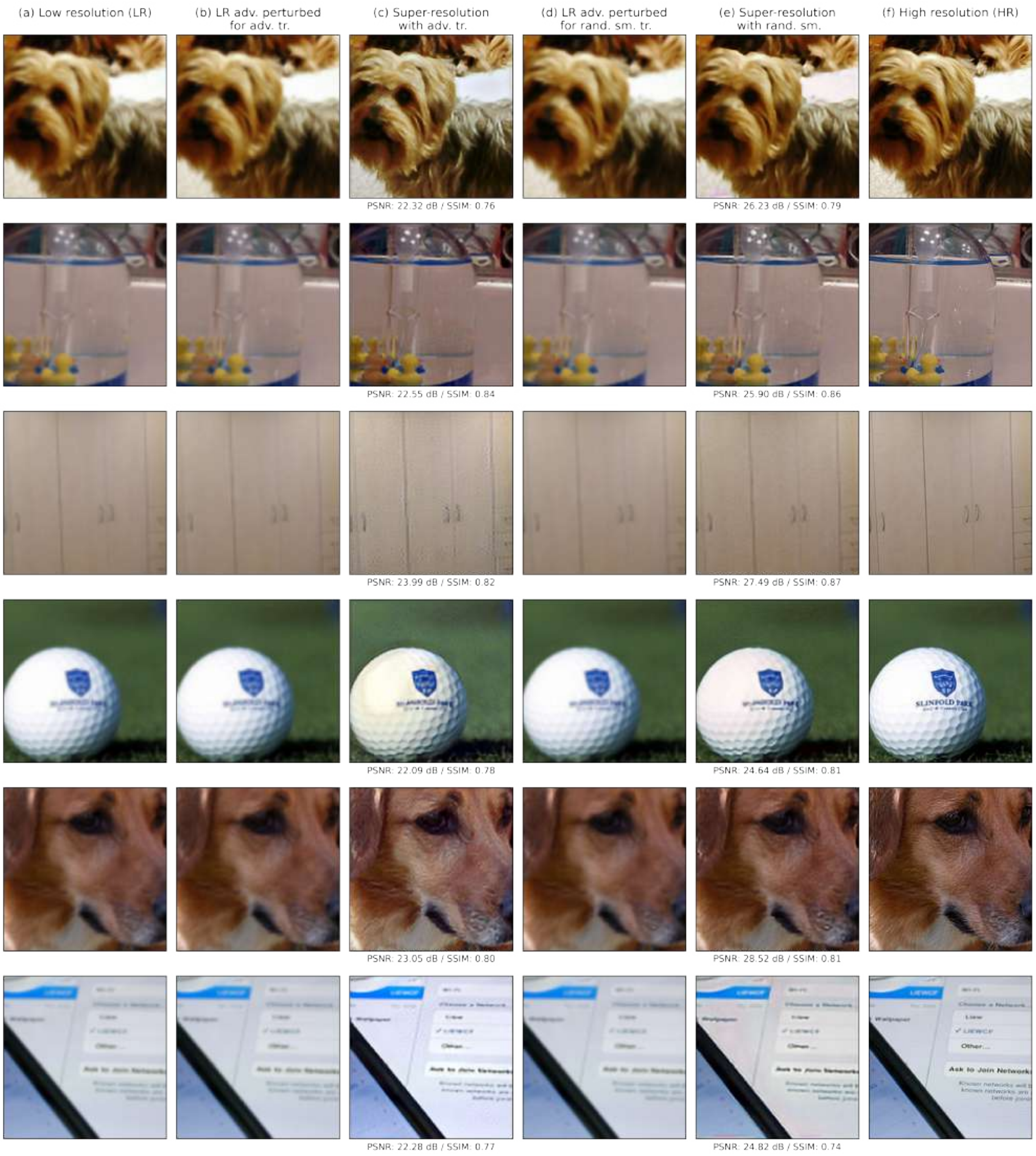

(a) Low resolution (LR)
(b) LR adv. perturbed for adv. tr.
(c) Super-resolution with adv. tr.
(d) LR adv. perturbed for rand. sm. tr.
(e) Super-resolution with rand. sm.
(f) High resolution (HR)
PSNR: 22.32 dB / SSIM: 0.76
PSNR: 26.23 dB / SSIM: 0.79
PSNR: 22.55 dB / SSIM: 0.84
PSNR: 25.90 dB / SSIM: 0.86
PSNR: 23.99 dB / SSIM: 0.82
PSNR: 27.49 dB / SSIM: 0.87
PSNR: 22.09 dB / SSIM: 0.78
PSNR: 24.64 dB / SSIM: 0.81
PSNR: 23.05 dB / SSIM: 0.80
PSNR: 28.52 dB / SSIM: 0.81
PSNR: 22.28 dB / SSIM: 0.77
PSNR: 24.82 dB / SSIM: 0.74

**Figure A.9** (a) Low resolution images (downsampling factor of 2), (b) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.01$ to attack a model trained adversarially with $\epsilon_{\text{rel}} = 0.01$, (c) Super-resolution output images of attack on adversarially trained model, (d) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.01$ to attack a model trained with Gaussian noise $\sigma_f = 0.1$ for randomized smoothing, (e) Super-resolution output images of smoothed model for $\sigma_f = 0.1$ and $\sigma_g = 0.15$, (f) High resolution (ground truth) images.

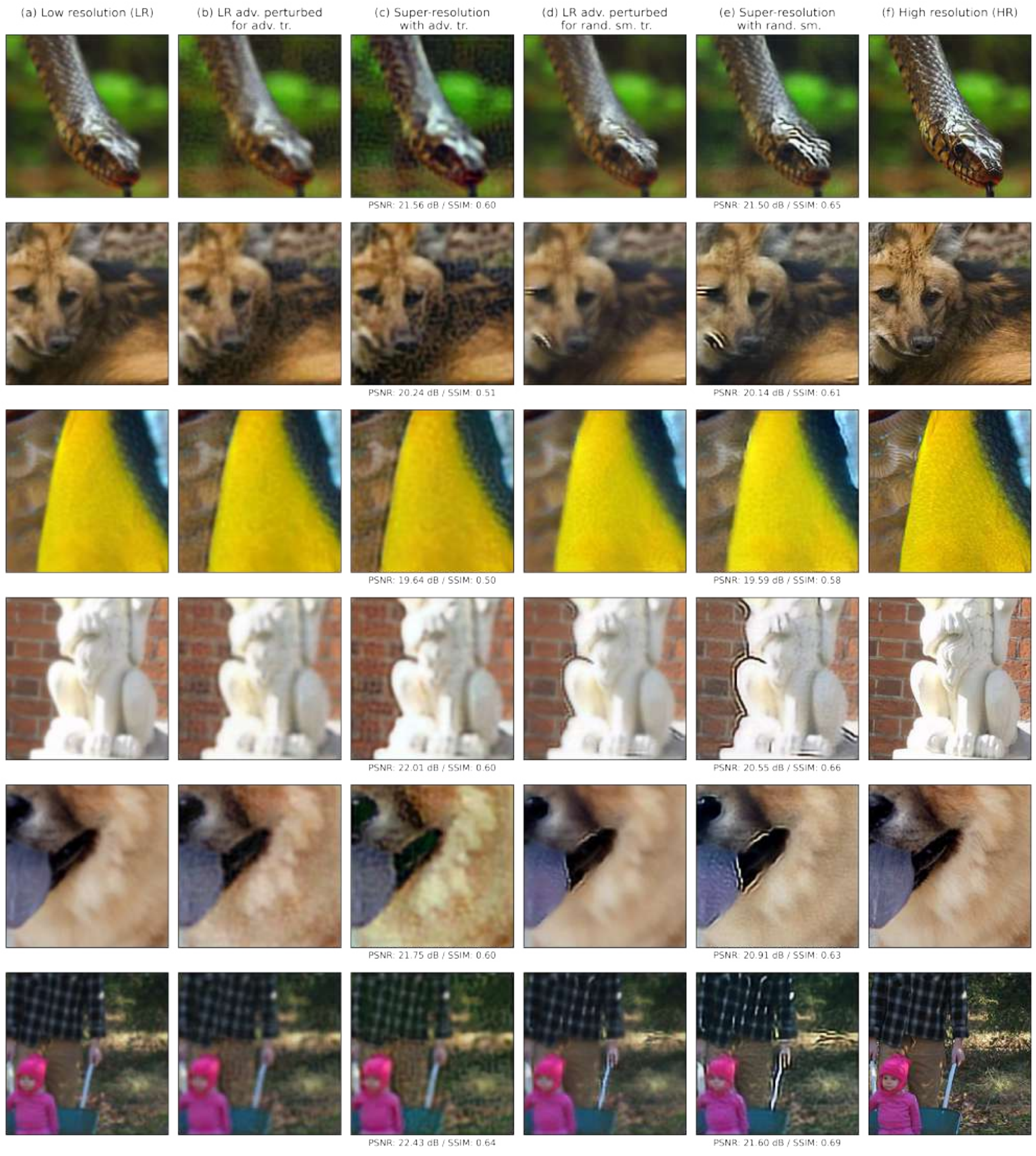

(a) Low resolution (LR)
(b) LR adv. perturbed for adv. tr.
(c) Super-resolution with adv. tr.
(d) LR adv. perturbed for rand. sm. tr.
(e) Super-resolution with rand. sm.
(f) High resolution (HR)
PSNR: 21.56 dB / SSIM: 0.60
PSNR: 21.50 dB / SSIM: 0.65
PSNR: 20.24 dB / SSIM: 0.51
PSNR: 20.14 dB / SSIM: 0.61
PSNR: 19.64 dB / SSIM: 0.50
PSNR: 19.59 dB / SSIM: 0.58
PSNR: 22.01 dB / SSIM: 0.60
PSNR: 20.55 dB / SSIM: 0.66
PSNR: 21.75 dB / SSIM: 0.60
PSNR: 20.91 dB / SSIM: 0.63
PSNR: 22.43 dB / SSIM: 0.64
PSNR: 21.60 dB / SSIM: 0.69

**Figure A.10** (a) Low resolution images (downsampling factor of 2), (b) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.1$ to attack a model trained adversarially with $\epsilon_{\text{rel}} = 0.1$, (c) Super-resolution output images of attack on adversarially trained model, (d) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.1$ to attack a model trained with Gaussian noise $\sigma_f = 0.4$ for randomized smoothing, (e) Super-resolution output images of smoothed model for $\sigma_f = 0.4$ and $\sigma_g = 0.5$, (f) High resolution (ground truth) images.

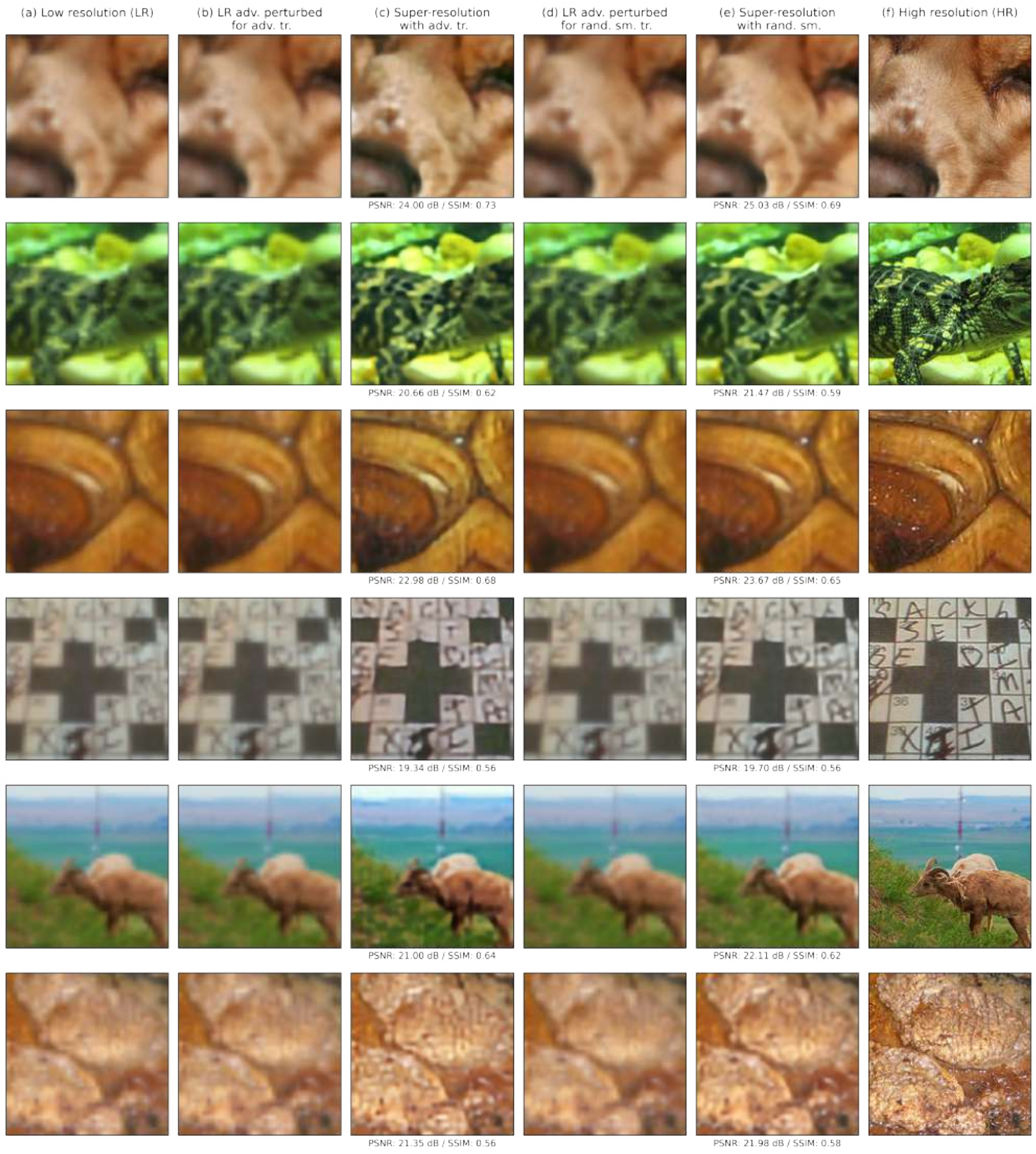

(a) Low resolution (LR)
(b) LR adv. perturbed for adv. tr.
(c) Super-resolution with adv. tr.
(d) LR adv. perturbed for rand. sm. tr.
(e) Super-resolution with rand. sm.
(f) High resolution (HR)
PSNR: 24.00 dB / SSIM: 0.73
PSNR: 25.03 dB / SSIM: 0.69
PSNR: 20.66 dB / SSIM: 0.62
PSNR: 21.47 dB / SSIM: 0.59
PSNR: 22.98 dB / SSIM: 0.68
PSNR: 23.67 dB / SSIM: 0.65
PSNR: 19.34 dB / SSIM: 0.56
PSNR: 19.70 dB / SSIM: 0.56
PSNR: 21.00 dB / SSIM: 0.64
PSNR: 22.11 dB / SSIM: 0.62
PSNR: 21.35 dB / SSIM: 0.56
PSNR: 21.98 dB / SSIM: 0.58

**Figure A.11** (a) Low resolution images (downsampling factor of 4), (b) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.01$ to attack a model trained adversarially with $\epsilon_{\text{rel}} = 0.01$, (c) Super-resolution output images of attack on adversarially trained model, (d) Low resolution images perturbed with adversarial noise $\epsilon_{\text{rel}} = 0.01$ to attack a model trained with Gaussian noise $\sigma_f = 0.3$ for randomized smoothing, (e) Super-resolution output images of smoothed model for $\sigma_f = 0.3$ and $\sigma_g = 0.4$, (f) High resolution (ground truth) images.

# Bibliography

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 2015.

[2] A. Kurakin, I. J. Goodfellow, and S. Bengio, *Adversarial Examples in the Physical World*. Chapman and Hall/CRC, 2018, pp. 99–112.

[3] X. Yuan, P. He, Q. Zhu, and X. Li, “Adversarial examples: Attacks and defenses for deep learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[4] N. Akhtar and A. Mian, “Threat of adversarial attacks on deep learning in computer vision: A survey,” 2018.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 2019.

[6] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, “Synthesizing robust adversarial examples,” in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 284–293. [Online]. Available: https://proceedings.mlr.press/v80/athalye18b.html

[7] H. Salman, G. Yang, J. Li, P. Zhang, H. Zhang, I. Razenshteyn, and S. Bubeck, “Provably robust deep learning via adversarially trained smoothed classifiers,” 2019.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Conference Proceedings.

[9] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, “Certified adversarial robustness via randomized smoothing,” 2019.

[10] S. Kumar and A. Narayan, “Towards robust certified defense via improved randomized smoothing,” in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Conference Proceedings.

[11] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, “Deep learning techniques for inverse problems in imaging,” 2020.

[12] H. Salman, M. Sun, G. Yang, A. Kapoor, and J. Z. Kolter, “Denoised smoothing: A provable defense for pretrained classifiers,” 2020.

[13] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, “Towards robust neural networks via random self-ensemble,” 2018.

[14] X. Cao and N. Z. Gong, “Mitigating evasion attacks to deep neural networks via region-based classification,” 2019.

[15] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, “Certified robustness to adversarial examples with differential privacy,” 2019.

[16] N. Carlini, F. Tramer, K. D. Dvijotham, L. Rice, M. Sun, and J. Z. Kolter, “(certified!!) adversarial robustness for free!” 2023.

[17] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen, “On instabilities of deep learning in image reconstruction and the potential costs of ai,” *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 088–30 095, 2020.

[18] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, “Deep learning for single image super-resolution: A brief review,” *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3106–3121, 2019.

[19] J.-H. Choi, H. Zhang, J.-H. Kim, C.-J. Hsieh, and J.-S. Lee, “Evaluating robustness of deep image super-resolution against adversarial attacks,” 2019.

[20] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” 2015.

[21] Z. Wang, J. Chen, and S. C. H. Hoi, “Deep learning for image super-resolution: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3365–3387, 2021.

[22] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, “Esrgan: Enhanced super-resolution generative adversarial networks,” 2018.

[23] X. Hu, M. A. Naiel, A. Wong, M. Lamm, and P. Fieguth, “Runet: A robust unet architecture for image super-resolution,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Conference Proceedings.

[24] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” 2019.

[26] L. Engstrom, A. Ilyas, S. Santurkar, and D. Tsipras, “Robustness (python library),” 2019. [Online]. Available: https://github.com/MadryLab/robustness

[27] U. Sara, M. Akter, and M. S. Uddin, “Image quality assessment through fsim, ssim, mse and psnr—a comparative study,” *Journal of Computer and Communications*, vol. 07, no. 03, pp. 8–18, 2019.