Smoothing-Based Adversarial Defense Methods for Inverse Problems

Yang Sun
Department of Computer Science, National University of Singapore
yang.sun@comp.nus.edu.sg

Jonathan Scarlett
Department of Computer Science, Department of Mathematics, and Institute of Data Science
National University of Singapore
scarlett@comp.nus.edu.sg

Abstract

In this paper, we propose randomized smoothing methods that aim to enhance the robustness of the linear inverse problems against adversarial attacks, in particular guaranteeing an upper bound on a suitably-defined notion of sensitivity to perturbations. In addition, we propose two novel algorithms that incorporate randomized smoothing into training, where one algorithm injects random perturbations to the input data directly, and the other algorithm adds random perturbations to the gradients during backpropagation. We conduct numerical evaluations on two of the most prominent inverse problems — denoising and compressed sensing — utilizing a variety of neural network estimators and datasets. In broad scenarios, these results demonstrate a strong potential of randomized smoothing for enhancing the robustness of linear inverse problems.

1 Introduction

Linear inverse problems are fundamental in machine learning, signal processing, and statistics, with the objective of recovering an unknown vector from an (often underdetermined) set of linear measurements. Over the years, the incorporation of low-dimensional structures such as sparsity has dominated the extensive research [12, 15, 11]. More recently, advances in deep learning have led to their widespread adoption in inverse problems, including for signal modeling, decoder design, and measurement design [33, 36]. We will specifically be interested in the decoder design aspect.

Following these advances, a number of recent works have identified limitations of deep learning methods in terms of robustness, e.g., with [3, 20, 16] studying their vulnerability to adversarial attacks with small ℓ_2 norm., which have a minimal effect on the input but significantly worsen the accuracy of the output. The study in [18] further indicates that deep learning for inverse problems can come at the cost of instabilities, and current training approaches cannot guarantee stable methods. Similar sensitivities of deep neural networks have been well-studied in classification problems, where initial defense approaches were soon broken [5, 30]. This led to the development of *certified* defenses such as reluplex [23], provable defenses [39], and randomized smoothing [7] have emerged, providing rigorous guarantees of robustness against norm-bounded attacks in classification. However, there is much less work on the adversarial robustness of inverse problems, which has fundamental differences from classification and even regression, notably including the fact that the output is high-dimensional.

In this work, we propose a novel method to certify the robustness of linear inverse problems against adversarial attacks using a suitably-adapted form of randomized smoothing [7]. Our approach is able

AdvML-Frontiers'24: The 3nd Workshop on New Frontiers in Adversarial Machine Learning@NeurIPS'24, Vancouver, CA.

to provide certified upper bounds on a suitably-defined notion of sensitivity to perturbations. To our knowledge, such results have not been explored previously in the context of inverse problems.

In addition, inspired by analogies with classification and regression, we incorporate the concept of randomized smoothing into the estimator's training. We propose two versatile training algorithms that can be applied to a variety of network architectures, at least up to a moderate size. Through numerical experiments, we demonstrate that smoothing can provide significant robustness certificates against adversarial attacks, as well as being practically beneficial in both denoising and compressed sensing tasks.

1.1 Setup

A linear inverse problem consists of recovering a target signal[1] $\mathbf{x} \in \mathbb{R}^n$ from linear measurements $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are the measurement matrix and additive noise, respectively. Given $\mathbf{A}$ and $\mathbf{y}$, we employ a trainable estimator $f_{\boldsymbol{\theta}} : \mathbb{R}^m \to \mathbb{R}^n$ to approximate $\mathbf{x}$ as $f_{\boldsymbol{\theta}}(\mathbf{y})$, and assess correctness using the normalized mean squared error (MSE) criterion:

$$\ell(f_{\boldsymbol{\theta}}(\mathbf{y}), \mathbf{x}) = \frac{1}{n}\|f_{\boldsymbol{\theta}}(\mathbf{y}) - \mathbf{x}\|^2. \tag{1}$$

Parts of our paper are applicable to an arbitrary estimator $f_{\boldsymbol{\theta}}$, but at times we will also specifically consider training its parameters $\boldsymbol{\theta}$ given access to a data set $\mathcal{D}$ consisting of $(\mathbf{x}, \mathbf{y})$ pairs, with each $\mathbf{y}$ being a measurement of the form $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ as above.

We study the adversarial robustness of $f_{\boldsymbol{\theta}}$ by assuming the existence of an unknown adversarial attack $\boldsymbol{\xi}$ on the target signal $\mathbf{x}$ at test time, such that the perturbed measurement vector is given by $\mathbf{y} = \mathbf{A}(\mathbf{x} + \boldsymbol{\xi}) + \mathbf{b}$. To limit the power of the adversary, we adopt the standard assumption that $\boldsymbol{\xi}$ is ϵ-bounded in a ℓ_2 ball (i.e., $||\boldsymbol{\xi}|| \leq \epsilon$), and in the worst-case may be chosen to maximize the error $\ell(f_{\boldsymbol{\theta}}(\mathbf{A}(\mathbf{x} + \boldsymbol{\xi}) + \mathbf{b}), \mathbf{x})$. To streamline our notation, in the rest of this paper, we will represent the clean measurement $\mathbf{A}\mathbf{x} + \mathbf{b}$ as $\mathbf{y}_{\mathbf{x}}$ and represent the perturbed measurement $\mathbf{A}(\mathbf{x} + \boldsymbol{\xi}) + \mathbf{b}$ as $\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}}$, which can be generalized to $\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}}$ in the case of multiple perturbations $\boldsymbol{\xi}$ and $\boldsymbol{\delta}$.

2 Related Work

Adversarial training is a commonly used training strategy (primarily used for classification problems [26, 31, 34]) to improve the robustness of perturbations through training with adversarial examples [17]. It follows the framework of empirical risk minimization (ERM) but replaces the input data with some ℓ_2-bounded adversarial perturbations. Several methods have been proposed in prior work [26] to generate these perturbations, with the multi-step iterative method Projected Gradient Descent (PGD) being particularly effective. [22] studied the tradeoff between robustness and standard accuracy using adversarial training in linear regression problems, whereas [18] suggested that adversarial training alone may be insufficient for addressing instabilities in linear inverse problems. Nonetheless, we include adversarial training equipped with the PGD method as one of the baselines in our experiments and provide the ERM formulation as

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*}), \mathbf{x})\big], \\ &\text{where } \boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}:\|\boldsymbol{\xi}\|_2\leq\epsilon}{\operatorname{argmax}}\, l(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}})), \mathbf{x}). \end{aligned} \tag{2}$$

Jacobian regularization is a technique to improve robustness against adversarial attacks by penalizing large derivatives with respect to input data, specifically the norm of the input-output Jacobian matrix. The idea was first proposed by [19] but only implemented as a layer-wise regularization due to computational constraints. Later it was implemented at full scale in [37] and was made more efficient in [38, 40]. [21, 2] showed that both the Frobenius and spectral norms of the Jacobian matrix are experimentally effective in improving the robustness of neural networks in regression problems.

For linear inverse problems, we can motivate the use of the Jacobian by considering the Taylor expansion of $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}})$ with respect to $\boldsymbol{\xi}$:

$$\begin{aligned} f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}}) &= f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}} + \mathbf{A}\boldsymbol{\xi}) \\ &= f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}}) + \mathbf{J}(\mathbf{y}_{\mathbf{x}})\mathbf{A}\boldsymbol{\xi} + O(\|\boldsymbol{\xi}\|^2), \end{aligned} \tag{3}$$

[1] For image data, we think of $\mathbf{x}$ as being its vectorized version, with n being the total number of pixels.

where $\mathbf{J}(\mathbf{y_x}) \in \mathbb{R}^{n\times m}$ is the input-output Jacobian matrix. When the ℓ_2-radius of the adversarial perturbation $\boldsymbol{\xi}$ is small enough, the higher-order terms can be ignored, and the stability of the estimation is mainly determined by $\mathbf{J}(\mathbf{y_x})\mathbf{A}$. Therefore, we can use Jacobian regularization by penalizing the spectral norm $\max_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\|\mathbf{J}(\mathbf{y})\mathbf{A}\|$, as proposed by [2], and control the strength of regularization using a hyperparameter β. The ERM formulation for training is

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\Big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y}),\mathbf{x})\Big] + \beta \max_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\|\mathbf{J}(\mathbf{y})\mathbf{A}\|. \tag{4}$$

Randomized smoothing is a technique that was first introduced for certifying a classifier against ℓ_2-bounded noise, potentially on large neural networks and on large images [27, 7, 29]. It works by adding random noise to input data and then classifying the noisy input to obtain a probability distribution over possible labels. Several works have shown that randomized smoothing guarantees that no possible attacks under a particular constraint could succeed in classification problems. [28] presented certified robustness against Wasserstein adversarial attacks. [13] covered parameterized transformations and certified robustness in the parameter space. [14] developed a certification method for image and point cloud segmentation. Furthermore, [35] embedded randomized smoothing into the classifier's adversarial training framework, which inspired us to propose Algorithm 3 in this paper.

Randomized smoothing has also been extended to certain tasks beyond classification. [6] used it to certify the regression of the object detector's bounding-box coordinates. [4] applied it to certify the accuracy score in watermarking problems and proposed a training framework from which we are inspired to propose Algorithm 4. These two works are the closest to ours, but there are some significant differences. In particular, in our case, the measurements are mapped to a high-dimensional signal manifold, whereas the prior works consider outputting a single real number or only a few real numbers. In our setting, a naive strategy of summarizing the output via its loss (a single real number) is infeasible due to the loss being unknown at the decoder, as we discuss below.

3 Proposed Methods

3.1 Preliminaries

We first review the necessary background of randomized smoothing for estimation in a generic *scalar-valued output* setting. Given an ordinary real-valued estimator $F : \mathbb{R}^n \to \mathbb{R}$, randomized smoothing is a method that constructs a new, smoothed estimator G from the base estimator F at testing time. Under *mean smoothing*, the smoothed estimator G takes as input $\mathbf{x} \in \mathbb{R}^n$ and returns the expectation of F under isotropic Gaussian noise perturbation $\boldsymbol{\delta}$ on $\mathbf{x}$, i.e.,

$$G(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}\big[F(\mathbf{x}+\boldsymbol{\delta})\big], \tag{5}$$

where the noise level σ controls the tradeoff between robustness and accuracy.

While (5) is the go-to approach for classification, its direct use for estimating *continuous-valued* quantities may result in skewed results and location-dependent bounds, as detailed in [6]. To overcome this limitation, they proposed a more suitable "percentile smoothing" (see (6) below) for one-dimensional regression tasks. In high-dimensional inverse problems, although incorporating randomized smoothing may seem feasible by regressing over the scalar estimation loss $\ell(f_{\boldsymbol{\theta}}(\mathbf{y_x}),\mathbf{x})$, we cannot access the original signal $\mathbf{x}$ and only have the measurement $\mathbf{y_x}$ at hand (or more generally $\mathbf{y_{x+\xi}}$). Hence, instead of considering the loss to $\mathbf{x}$ itself, we consider the discrepancy between two different reconstructions with different perturbations as follows.

Definition 3.1. *Given an estimator $f_{\boldsymbol{\theta}} : \mathbb{R}^m \to \mathbb{R}^n$, an adversarially perturbed measurement $\mathbf{y_{x+\xi}}$, and a Gaussian noise vector $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})$, we use $\boldsymbol{\delta}$ to perturb $\mathbf{x}$ further and obtain the measurement $\mathbf{y_{x+\xi+\delta}} = \mathbf{y_{x+\xi}} + \mathbf{A}\boldsymbol{\delta}$. We refer to $\ell(f_{\boldsymbol{\theta}}(\mathbf{y_{x+\xi+\delta}}), f_{\boldsymbol{\theta}}(\mathbf{y_{x+\xi}}))$ as the* discrepancy*, and consider the percentile smoothing of the discrepancy as a function of $\mathbf{x}$ and $\boldsymbol{\xi}$:*

$$D_p(\mathbf{x},\boldsymbol{\xi}) = \inf\big\{t\in\mathbb{R} : \mathbb{P}\big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y_{x+\xi+\delta}}), f_{\boldsymbol{\theta}}(\mathbf{y_{x+\xi}})) \le t\big] \ge p\big\}, \tag{6}$$

where p is the quantile value.

The idea behind the discrepancy is that a small $D_p(\mathbf{x},\boldsymbol{\xi})$ indicates that the estimation retains consistency despite the presence of Gaussian noise $\boldsymbol{\delta}$. This consistency is advantageous for adversarial robustness, allowing the addition of Gaussian noise to "drown out" an adversarial perturbation without impacting the reconstruction too much. A more detailed discussion on the proposed discrepancy measure is presented in Appendix A.

3.2 Discrepancy certification

We follow Lemma 1 from [6] and propose the following corollary to bound the worst-case smoothed discrepancy (proved in Appendix B).

Corollary 3.2. *The median smoothed discrepancy with adversarial noise can be bounded by the smoothed discrepancy in the absence of adversarial noise as*

$$D_{0.5}(\mathbf{x}, \boldsymbol{\xi}) \leq D_{\Phi(\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0}) \quad \forall\, \boldsymbol{\xi} \text{ s.t. } \|\boldsymbol{\xi}\| \leq \epsilon, \tag{7}$$

where Φ is the cumulative distribution function of unit normal distribution, and $\mathbf{0}$ is a zero vector.

Equation (7) states that a worst-case upper bound on the median smoothed discrepancy $D_{0.5}(\mathbf{x}, \boldsymbol{\xi})$ is the $\Phi(\frac{\epsilon}{\sigma})$-th percentile smoothed discrepancy $D_{\Phi(\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0})$, for *arbitrary* $\boldsymbol{\xi}'$ within an ϵ-ball around zero. This corollary guarantees that the sensitivity of $\mathbf{x}$ (as measured by the discrepancy) is minimally affected by arbitrary shifts up to radius ϵ. We focus on the median for simplicity, but the proof reveals a more general form for any specified quantile value in $(0, 1)$.

It is worth noting that $\mathbf{x}$ and $\boldsymbol{\xi}$ in Corollary 3.2 represent generic vectors whose assignments may vary depending on how the corollary is applied. In particular, suppose that we are in a scenario where the true signal is $\mathbf{x}^*$, and the true adversarial noise is $\boldsymbol{\xi}^*$, so the measurement vector is $\mathbf{y}_{\mathbf{x}^*+\boldsymbol{\xi}^*}$. Then we may apply Corollary 3.2 with $\mathbf{x} = \mathbf{x}^* + \boldsymbol{\xi}^*$, meaning that we are giving a guarantee on the discrepancy for all signals sufficiently close to $\mathbf{x}^* + \boldsymbol{\xi}^*$ (rather than sufficiently close to $\mathbf{x}^*$ itself; we cannot use $\mathbf{x}^*$ directly since $\mathbf{y}_{\mathbf{x}^*}$ is not known to the algorithm).

We also emphasize that our approach does not directly ensure "good" or "bad" estimation but rather provides a guarantee on the discrepancy. This is analogous to classification, where a certificate may show that the decision is unaffected within a certain radius, without claiming it to be the correct decision in the first place. While this is a standard limitation of smoothing methods, we still consider using discrepancy calculations, Peak Signal-to-Noise Ratio (PSNR) values (a standard measure of accuracy with the definition provided in Section 4), and Structural Similarity (SSIM) values (a commonly used measure of perceived similarity) for experimental evaluations.

3.3 Randomized smoothing for inverse problems at test time

Our method for randomized smoothing at test time is directly based on discrepancy introduced in Definition 3.1, which is a new concept to the best of our knowledge. For concreteness, we often focus on $p = 0.5$, i.e., the median. Although $D_p(\mathbf{x}, \boldsymbol{\xi})$ will generally not admit a closed form, we can still approximate it by applying a Monte Carlo approach with a suitably-chosen confidence level (see Algorithm 1 in Appendix C).

The preceding ideas give rise to two estimators of $\mathbf{x}$ that we will consider throughout the paper:

- For certified estimation, the Monte Carlo procedure with S samples gives an appropriate quantile value $q \in (0, 1)$; we compute the relevant S values of $\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}}), f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}}))$ with randomly sampled $\boldsymbol{\delta}$, sort them, and take the one at position $q \cdot S$ (with integer rounding).
- We also consider median-based estimation, with the same procedure but simply $q = \frac{1}{2}$.

We emphasize that the Gaussian noise $\boldsymbol{\delta}$ for smoothing is intentionally introduced by the algorithm, whereas $\boldsymbol{\xi}$ is unknown and possibly adversarial. We henceforth refer to them as the smoothing noise $\boldsymbol{\delta}$ and the adversarial noise $\boldsymbol{\xi}$.

3.4 Embedding randomized smoothing into training

The previous subsection covers the use of randomized smoothing during testing, and applies to arbitrary $f_{\boldsymbol{\theta}}$ (e.g., pre-trained). We provide two algorithms (specified in Appendix D) that use randomized smoothing during training, i.e., selecting the parameters $\boldsymbol{\theta}$ of $f_{\boldsymbol{\theta}}$ based on training data.

The first algorithm, termed "Smoothed Adversarial Training" (Smt-Adv) and building on [35], is an approach that roughly follows adversarial training (as discussed in Section 2) but uses randomized smoothing when generating the ℓ_2-bounded adversarial perturbation. Specifically, such adversarial training is performed using the perturbation noise $\boldsymbol{\xi}^*$ obtained as the expected value under the

smoothing noise $\boldsymbol{\delta}$. Formally, the ERM with smoothing is performed as

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*}),\mathbf{x})\big], \text{where} \\ &\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}:\|\boldsymbol{\xi}\|_2\leq\epsilon}{\operatorname{argmax}} \ell\Big(\mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}\big[f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}})\big],\mathbf{x}\Big). \end{aligned} \tag{8}$$

The algorithm computes $\boldsymbol{\xi}^*$ using a first-order method similar to PGD, but with the difference that the perturbed measurement $\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}}$ is further smoothed in the neighborhood around $\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}}$ before being used in the loss function. It is difficult to evaluate $\boldsymbol{\xi}^*$ exactly, so we use Monte Carlo sampling with S i.i.d. smoothing noise samples, $\boldsymbol{\delta}_1,\ldots,\boldsymbol{\delta}_S \sim \mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})$, and compute the estimated $\hat{f}_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}})$ using the sample mean

$$\hat{f}_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}}) = \frac{1}{S}\sum_{k=1}^{S} f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}_k}). \tag{9}$$

See Algorithm 3 for a complete description. Although adversarial training is an empirical defense technique that generally lacks provable guarantees, we will observe promising results in our experiments. The same goes for our second algorithm introduced below.

The second algorithm (see Algorithm 4), termed "Smoothed Gradient Training" (Smt-Grad) and built on [4], is a regular ERM training framework without the regularization term. To embed randomized smoothing, we augment the clean measurements with smoothing noise $\boldsymbol{\delta}$ and compute the estimator's gradients using an expectation with respect to $\boldsymbol{\delta}$. Specifically, for a given training pair $(\mathbf{x},\mathbf{y})$, the gradient $\mathbf{g}_{\boldsymbol{\theta}}$ with respect to the estimator's parameter $\boldsymbol{\theta}$ is computed as

$$\mathbf{g}_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}\big[\nabla_{\boldsymbol{\theta}}\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\delta}}),\mathbf{x})\big]. \tag{10}$$

To approximate the expectation of the gradients in the equation, we use the Monte Carlo method drawing i.i.d. smoothing noise samples $\boldsymbol{\delta}$ in each iteration of stochastic gradient descent. As employing strong smoothing noise at the beginning of the training can be unstable, we adopt the approach proposed by [4] and incrementally increase the noise levels within each epoch. This is achieved by introducing a step count T_{step} and injecting smoothing noise incrementally with standard deviation σ ranging from $\frac{1}{T_{step}}\sigma_{max}$ to σ_{max}, where σ_{max} is the maximum noise level.

We also make further observations regarding the two proposed algorithms and briefly mention a concurrent work [24] in Appendix D.

4 Experiments

We conduct experiments in two different tasks: denoising, where $\mathbf{A}=\mathbf{I}$ and $\mathbf{b}$ is Gaussian noise with level $\sigma_{\mathbf{b}}$; and compressed sensing, where $\mathbf{A}$ is a random Gaussian matrix with compression ratio m/n, and the additive noise $\mathbf{b}$ is set to zero. We consider training and testing sets from both the same data domain and different data domains (i.e., transfer learning), as is common in this line of works. Specifically, for denoising, we use the network DPDNN [9] as the estimator trained on the (grayscale converted) DIV2K dataset [1] and tested on the DIV2K validation set and BSD68 dataset [32], while for compressed sensing, we use the estimator ISTA-Net^{++} [41] trained on the Train400 dataset [25] and tested on the Set11 [25] and a subset of the (grayscale converted) ImageNet [8] datasets.

To assess the impact of randomized smoothing during training, we compare three baselines with our proposed training algorithms: **Ordinary Training (Ord)**: the standard training algorithm that does not incorporate any perturbation defense mechanisms, with the ERM formulation simply presented as $\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y}),\mathbf{x})\big]$; **Adversarial Training (Adv)**: the adversarial training algorithm equipped with PGD method as introduced in (2); **Jacobian Regularization (Jcb)**: the Jacobian regularization training algorithm using the spectral norm for penalization as detailed in (4), where we determine the value of hyperparameter β using Algorithm 4 from [2]; **Smoothed Adversarial Training (Smt-Adv)** from Algorithm 3; **Smoothed Gradient Training (Smt-Grad)** from Algorithm 4.

To evaluate the effectiveness of randomized smoothing at test time, we explore different choices of adversarial attack radii and various smoothing noise levels. More training and testing details are presented in Appendix E.

4.1 Certificate evaluation

We first examine the certificates of all types of trained estimators under varying ℓ_2 radii of the adversarial noise. Afterwards, we will investigate how adjusting the smoothing noise used in training can impact the certificate.

Figure 1 (see Appendix F) presents a comparison of the certified results for both tasks. We display the results as the average over the corresponding dataset and include error bars representing half a standard deviation. We first observe that the certified upper bound of smoothed discrepancy increases with the smoothing level, which is expected as higher smoothing levels amount to more deviations from the original signal, as seen in (6). Regarding the PSNR of the reconstructions when smoothing is used, we consistently observe an "arc" shape in which the performance first improves with σ but then worsens, with the peak representing an ideal trade-off between σ being large enough to improve robustness, but not so large as to over-smooth. Naturally, the trend is that the best choice of σ increases with ϵ, with the two generally being comparable. However, ϵ is not the only factor at play; when the estimation problem is made "more difficult" via increased $\sigma_{\mathbf{b}}$ (denoising) or fewer measurements (compressed sensing), the best choice of σ also tends to increase.

In comparing the five training algorithms, we find that, at the optimal smoothing noise level, Smt-Grad consistently produces the lowest certified discrepancy and the highest corresponding estimation accuracy across all datasets and tasks. This suggests that Smt-Grad-trained estimators are the least susceptible to perturbations, at least for the attacks we consider. Moreover, when little or no test-time smoothing is applied (small σ, seen in the left parts of the plots in Figure 1), Smt-Adv is generally seen to achieve the highest PSNR. On the other hand, when significant smoothing is applied, Smt-Grad can be preferable, particularly in the case of compressed sensing, with lower discrepancy and higher PSNR. Overall, we see that both of our proposed training algorithms can effectively improve the robustness of the estimator against adversarial attacks.

We now explore how different levels of smoothing noise σ used for training can affect performances. Table 2 (see Appendix F) provides a comparison of the performance of algorithms trained with different smoothing noise levels (shown as σ_{tr} to distinguish from the test-time smoothing noise σ) in terms of their performance metrics. These metrics include the PSNR of the clean estimate $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}})$, the PSNR of the perturbed estimate $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}})$, and the PSNR of the smoothed estimate $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}})$ that provides the certified smoothed discrepancy. For each ℓ_2-radius ϵ of the perturbation $\boldsymbol{\xi}$, we use a zero test-time smoothing noise level to obtain the PSNR of the perturbed estimate and an empirically optimal test-time σ to obtain the smoothed PSNR. For comparison, we also include a baseline vanilla scenario that showcases the performance of the algorithm without any robustness considerations, i.e., Ord. The results demonstrate that a significant level of σ_{tr} (e.g., $\sigma_{\mathrm{tr}} = 10$, which we use as the default training setting) can, in most cases, result in improved test-time robustness, indicated by higher PSNR values. However, as the estimator is trained more robustly against stronger adversarial attacks (when $\epsilon > 0$), the accuracy of the estimator's clean estimation drops, as shown in the second column. An analogous trade-off between robustness and accuracy has also been observed in the adversarial robustness literature, e.g., [31] and, [4].

4.2 Empirical smoothed median estimate

In practice, one may not specifically require certificates, and accordingly, it is of interest to study the performance when one instead adopts the simpler approach of using the median, as described in Section 3.3. To address this, we present Table 1 (see Appendix F) demonstrating the similarity between the experimental median and its corresponding certificate across various scenarios. These scenarios encompass different tasks and datasets, and we note that the rows of the table are not meant to be directly compared with one another. Overall, our results reveal that under the ideal test-time smoothing level, the evaluated discrepancy and PSNR values of the smoothed median estimate are not substantially different from those with certification.

We visually compare the smoothed estimate $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}})$ corresponding to the smoothed median discrepancy with the perturbed estimate $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}})$ in Figure 2 (see Appendix F). We also show the vanilla scenario where Ord is utilized and neither perturbation nor test-time smoothing is incorporated. Notably, we observe significant improvements in estimation accuracy and feature clarity in both denoising and compressed sensing tasks when smoothing is applied. Even without test-time smoothing ($\sigma = 0$), estimators trained with Smt-Adv or Smt-Grad consistently outperform all baselines.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conference on Computer Vision and Pattern Recognition (Workshop)*, pages 1122–1131, 2017.

[2] Jaweria Amjad, Zhaoyan Lyu, and Miguel RD Rodrigues. Deep learning model-aware regulatization with applications to inverse problems. *IEEE Transactions on Signal Processing*, 69:6371–6385, 2021.

[3] Vegard Antun, Francesco Renna, Clarice Poon, Ben Adcock, and Anders C Hansen. On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proceedings of the National Academy of Sciences*, 117(48):30088–30095, 2020.

[4] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning*, pages 1450–1465. PMLR, 2022.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[6] Ping-yeh Chiang, Michael Curry, Ahmed Abdelkader, Aounon Kumar, John Dickerson, and Tom Goldstein. Detection as regression: Certified object detection with median smoothing. In *Advances in Neural Information Processing Systems*, volume 33, pages 1275–1286, 2020.

[7] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[9] Weisheng Dong, Peiyao Wang, Wotao Yin, Guangming Shi, Fangfang Wu, and Xiaotong Lu. Denoising prior driven deep neural network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(10):2305–2318, 2018.

[10] John Duchi. Derivations for linear algebra and optimization. page 13.

[11] Yonina C Eldar. *Sampling theory: Beyond bandlimited systems*. Cambridge University Press, 2015.

[12] Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge University Press, 2012.

[13] Marc Fischer, Maximilian Baader, and Martin Vechev. Certified defense to image transformations via randomized smoothing. In *Advances in Neural Information Processing Systems*, volume 33, pages 8404–8417, 2020.

[14] Marc Fischer, Maximilian Baader, and Martin Vechev. Scalable certified segmentation via randomized smoothing. In *International Conference on Machine Learning*, pages 3340–3351. PMLR, 2021.

[15] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[16] Martin Genzel, Jan Macdonald, and Maximilian März. Solving inverse problems with deep neural networks–robustness included? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1119–1134, 2022.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *Stat*, 1050:20, 2015.

[18] Nina M. Gottschling, Vegard Antun, Anders C. Hansen, and Ben Adcock. The troublesome kernel – on hallucinations, no free lunches and the accuracy-stability trade-off in inverse problems. https://arxiv.org/abs/2001.01258, 2023.

[19] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *International Conference on Learning Representations (Workshop)*, 2015.

[20] Yixing Huang, Tobias Würfl, Katharina Breininger, Ling Liu, Günter Lauritsch, and Andreas Maier. Some investigations on robustness of deep learning in limited angle tomography. In *Medical Image Computing and Computer Assisted Intervention*, pages 145–153. Springer, 2018.

[21] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using Jacobian regularization. In *European Conference on Computer Vision*, pages 514–529, 2018.

[22] Adel Javanmard, Mahdi Soltanolkotabi, and Hamed Hassani. Precise tradeoffs in adversarial training for linear regression. In *Conference on Learning Theory*, pages 2034–2078. PMLR, 2020.

[23] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117, 2017.

[24] Anselm Krainovic, Mahdi Soltanolkotabi, and Reinhard Heckel. Learning provably robust estimators for inverse problems via jittering. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

[25] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458, 2016.

[26] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

[27] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy*, pages 656–672, 2019.

[28] Alexander Levine and Soheil Feizi. Wasserstein smoothing: Certified robustness against wasserstein adversarial attacks. In *International Conference on Artificial Intelligence and Statistics*, pages 3938–3947. PMLR, 2020.

[29] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[30] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (Workshop)*, 2017.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[32] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, pages 416–423, 2001.

[33] Gregory Ongie, Ajil Jalal, Christopher A Metzler, Richard G Baraniuk, Alexandros G Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[34] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

[35] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[36] Jonathan Scarlett, Reinhard Heckel, Miguel R. D. Rodrigues, Paul Hand, and Yonina C. Eldar. Theoretical perspectives on deep learning methods in inverse problems. *IEEE Journal on Selected Areas in Information Theory*, 3(3):433–453, 2022.

[37] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

[38] Dániel Varga, Adrián Csiszárik, and Zsolt Zombori. Gradient regularization improves accuracy of discriminative models. *Schedae Informaticae*, 27, 2018.

[39] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[40] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. https://arxiv.org/abs/1705.10941, 2017.

[41] Di You, Jingfen Xie, and Jian Zhang. ISTA-Net++: flexible deep unfolding network for compressive sensing. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2021.

Code

The Dropbox link to the source code for experiments results reproducing.

Appendix

A Justifications for smoothed discrepancy

In principle, this consistency could be achieved when both reconstructions $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}})$ and $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}})$ are poor while those for $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\delta}})$ and $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}})$ are good. However, intuitively speaking, if $\boldsymbol{\delta}$ is large enough to "drown out" $\boldsymbol{\xi}$, this should not be the case – the relevant Gaussians centered at $\mathbf{x}$ (i.e., $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$) vs. $\mathbf{x}+\boldsymbol{\xi}$ (i.e., $\mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I})$) should have significant overlap in their high-probability region, thus avoiding the above undesirable situation. In other words, if (i) the reconstruction for $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}})$ is good, (ii) $\boldsymbol{\delta}$ is comparable in magnitude to $\boldsymbol{\xi}$, and (iii) the discrepancy is low, then the smoothed reconstruction for $f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}})$ should also be good. To make this intuition more precise, we first note the following formula for the Kullback–Leibler (KL) divergence between two Gaussians with differing means and the same scaled identity covariance matrix.

Proposition A.1. *The KL divergence between $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$ and $\mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I})$ is given by*

$$\mathrm{KL}(\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}), \mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I})) = \frac{1}{2}\Big(\frac{||\boldsymbol{\xi}||}{\sigma}\Big)^2. \tag{11}$$

We explain in Appendix B how this follows easily from a more general formula for the KL divergence between multivariate Gaussians. As σ increases, this value diminishes as the two distributions exhibit greater "similarity", approaching a KL divergence of zero. It is also convenient to convert the above finding to an upper bound on the Total Variation (TV) distance, defined as

$$d_{\mathrm{TV}}(P, Q) = \sup_A |P(A) - Q(A)|, \tag{12}$$

where the supremum is over all events A. Using Pinsker's inequality (i.e., $d_{\mathrm{TV}}(P, Q) \leq \sqrt{\mathrm{KL}(P, Q)/2}$), we arrive at the following.

Corollary A.2. *The total variation distance between $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$ and $\mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I})$ is upper bounded as follows:*

$$d_{\mathrm{TV}}\big(\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}),\ \mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I})\big) \leq \frac{1}{2} \cdot \frac{||\boldsymbol{\xi}||}{\sigma}. \tag{13}$$

From the definition of TV distance in (12), this result implies that if some "bad event" (e.g., reconstruction accuracy falling below a certain threshold) occurs with probability at most ρ under the non-perturbed signal $\mathbf{x}$, then under the perturbed version $\mathbf{x}+\boldsymbol{\xi}$, it holds with probability at most

$$\rho + \frac{1}{2} \cdot \frac{\|\boldsymbol{\xi}\|}{\sigma}. \tag{14}$$

Thus, at least when σ is large enough relative to $\boldsymbol{\xi}$, the behavior is guaranteed to be similar in the two cases. We will explore the relationship between $\|\boldsymbol{\xi}\|$ and σ experimentally in Section 4.

B Proof of proposition A.1 and corollary 3.2

Proposition A.1. (Restated.) *The KL divergence between $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$ and $\mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I})$ is given by*

$$\mathrm{KL}(\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}), \mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I})) = \frac{1}{2}\Big(\frac{||\boldsymbol{\xi}||}{\sigma}\Big)^2. \tag{15}$$

Proof. In general, for two multivariate normal distributions, $\mathcal{N}_0$ and $\mathcal{N}_1$ sharing the same dimension k, with means μ_0 and μ_1 and covariance matrices Σ_0 and Σ_1, the KL divergence between them can be expressed as follows [10]:

$$\begin{aligned}&\mathrm{KL}(\mathcal{N}_0, \mathcal{N}_1)\\ =&\frac{1}{2}\Big(\mathrm{Tr}(\Sigma_1^{-1}\Sigma_0) - k + (\mu_1-\mu_0)^T\Sigma_1^{-1}(\mu_1-\mu_0) + \ln\frac{\det(\Sigma_1)}{\det(\Sigma_0)}\Big).\end{aligned} \tag{16}$$

In our context, where $k = n$, $\mu_0 = \mathbf{x}$, $\mu_1 = \mathbf{x}+\boldsymbol{\xi}$, and $\Sigma_0 = \Sigma_1 = \sigma^2\mathbf{I}$, the divergence is computed as follows:

$$\begin{aligned}&\mathrm{KL}(\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}), \mathcal{N}(\mathbf{x}+\boldsymbol{\xi}, \sigma^2\mathbf{I}))\\ =&\frac{1}{2}\Big(\mathrm{Tr}((\sigma^2\mathbf{I})^{-1}\sigma^2\mathbf{I}) - n + (\mathbf{x}+\boldsymbol{\xi}-\mathbf{x})^T(\sigma^2\mathbf{I})^{-1}(\mathbf{x}+\boldsymbol{\xi}-\mathbf{x}) + \ln\frac{\det(\sigma^2\mathbf{I})}{\det(\sigma^2\mathbf{I})}\Big)\\ =&\frac{1}{2}\Big(\mathrm{Tr}(\mathbf{I}) - n + \sigma^{-2}\boldsymbol{\xi}^T\mathbf{I}^{-1}\boldsymbol{\xi} + \ln 1\Big)\\ =&\frac{1}{2}\Big(n - n + \sigma^{-2}\boldsymbol{\xi}^T\boldsymbol{\xi}\Big)\\ =&\frac{1}{2}\Big(\frac{||\boldsymbol{\xi}||}{\sigma}\Big)^2.\end{aligned} \tag{17}$$

□

Corollary 3.2. (Restated.) *The median smoothed discrepancy with adversarial noise can be bounded by the smoothed discrepancy in the absence of adversarial noise as*

$$D_{0.5}(\mathbf{x}, \boldsymbol{\xi}) \le D_{\Phi(\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0}) \quad \forall\, \boldsymbol{\xi} \text{ s.t. } \|\boldsymbol{\xi}\| \le \epsilon, \tag{18}$$

where Φ is the cumulative distribution function of unit normal distribution, and $\mathbf{0}$ is a zero vector.

Proof. According to Lemma 2.2 in [6], given a real-valued function $f : \mathbb{R}^m \to \mathbb{R}$, its percentile smoothing function is defined as

$$\overline{h}_p(\mathbf{x}) = \inf\{t \in \mathbb{R} \mid \mathbb{P}\big[\, f(\mathbf{x}+\boldsymbol{\delta}) \le t\big] \ge p\}, \tag{19}$$

and it is guaranteed that

$$\overline{h}_p(\mathbf{x}+\boldsymbol{\xi}) \le \overline{h}_{\overline{p}}(\mathbf{x}) \quad \forall\, \|\boldsymbol{\xi}\|_2 \le \epsilon, \tag{20}$$

where $\overline{p} = \Phi(\Phi^{-1}(p) + \frac{\epsilon}{\sigma})$.

Applying this result to our setting with any given estimator $f_{\boldsymbol{\theta}} : \mathbb{R}^m \to \mathbb{R}^n$, given our definition of the smoothed discrepancy

$$D_p(\mathbf{x}, \boldsymbol{\xi}) = \inf\left\{t \in \mathbb{R} \mid \mathbb{P}\big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}}), f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}})) \le t\big] \ge p\right\}, \tag{21}$$

we also have the guarantee

$$D_p(\mathbf{x}, \boldsymbol{\xi}) \le D_{\overline{p}}(\mathbf{x}, \mathbf{0}) \quad \forall \boldsymbol{\xi} \text{ s.t. } \|\boldsymbol{\xi}\|_2 \le \epsilon. \tag{22}$$

Then, by the definition of $\overline{p}$, we have

$$D_p(\mathbf{x}, \boldsymbol{\xi}) \le D_{\Phi(\Phi^{-1}(p)+\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0}) \quad \forall \boldsymbol{\xi} \text{ s.t. } \|\boldsymbol{\xi}\|_2 \le \epsilon. \tag{23}$$

In particular, substituting 0.5 for p, we obtain

$$\begin{aligned}D_{0.5}(\mathbf{x}, \boldsymbol{\xi}) &\le D_{\Phi(\Phi^{-1}(0.5)+\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0})\\ &= D_{\Phi(\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0})\end{aligned} \tag{24}$$

whenever $\|\boldsymbol{\xi}\|_2 \le \epsilon$. □

C Details of algorithms for randomized smoothing at test time

Similar to [4], Algorithm 1 below samples S random realizations of $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and computes the corresponding S discrepancies $\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*+\boldsymbol{\delta}}), f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*}))$. Note that these can be computed because $\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*}$ is given as an input and $\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*+\boldsymbol{\delta}} = \mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*} + \mathbf{A}\boldsymbol{\delta}$ (with known $\mathbf{A}$ and $\boldsymbol{\delta}$). The sequence of discrepancies is sorted, with the $\lfloor \frac{S}{2} \rfloor$-th item being used to approximate the median smoothed discrepancy $D_{0.5}(\mathbf{x}, \boldsymbol{\xi})$ and the $\lfloor \mathrm{EMPIRICAL}(c, \sigma, S, \epsilon) \rfloor$-th item to approximate its certified upper bound $D_{\Phi(\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0})$.

Algorithm 1 Find median and certified discrepancy values and corresponding estimates

input Perturbed testing pair $(\mathbf{x}, \mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*})$, where $\boldsymbol{\xi}^*$ is the adversarial noise produced by PGD method (but unknown to the algorithm); Estimator $f_{\boldsymbol{\theta}}$ and measurement matrix $\mathbf{A}$; Smoothing noise level σ and noise sample count S; Confidence level c; Adversarial noise ℓ_2 radius ϵ; Function $\mathrm{EMPIRICAL}(\cdot)$ from Algorithm 2;
initiate Sorting function $\mathrm{sort}()$; Empty set $\mathrm{discrepancy\_set} = \{\}$;

for $1 \leq k \leq S$ **do**
  $\boldsymbol{\delta}_k \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$
  $\mathrm{discrepancy\_set} \xleftarrow{add} \ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*+\boldsymbol{\delta}_k}), f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*}))$
end for
$\mathrm{discrepancy\_set\_sorted} = \mathrm{sort}(\mathrm{discrepancy\_set})$
Set $\hat{D}_{0.5}(\mathbf{x}, \boldsymbol{\xi}) = \mathrm{discrepancy\_set\_sorted}_{\lfloor 0.5S \rfloor}$
if $\mathrm{EMPIRICAL}(c, \sigma, S, \epsilon)$ returns non-null **then**
  Set $\hat{D}_{\Phi(\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0}) = \mathrm{discrepancy\_set\_sorted}_{\lfloor \mathrm{EMPIRICAL}(c,\sigma,S,\epsilon) \rfloor}$ (Algorithm 2)
  return the estimated median discrepancy $\hat{D}_{0.5}(\mathbf{x}, \boldsymbol{\xi}^*)$, the estimated certificate discrepancy $\hat{D}_{\Phi(\frac{\epsilon}{\sigma})}(\mathbf{x}, \mathbf{0})$, and the two corresponding estimates $\hat{\mathbf{x}} = f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}^*+\boldsymbol{\delta}_k})$, where $k = \lfloor 0.5S \rfloor$ and $k = \lfloor \mathrm{EMPIRICAL}(c, \sigma, S, \epsilon) \rfloor$
else
  return median discrepancy and corresponding estimate as above (certified result is unavailable due to overly large ratio $\frac{\epsilon}{\sigma}$)
end if

The functionality of EMPIRICAL is described in Algorithm 2. This algorithm is essentially taken from [6], and despite its rather technical description, is conceptually simple: Given a confidence level c, perform a binary search to find a value K such that the probability of the event $\mathrm{Binomial}(S, p) \leq K$ is roughly c. Here p is set to $\Phi(\frac{\epsilon}{\sigma})$ in accordance with Corollary 3.2. The idea is then that if we sort S binomial samples and take the K-th one, we can be confident (up to the confidence level c) that it is bounded by the actual p-quantile we are interested in. When performing smoothing, we are not directly using binomial samples, but the indicator variable of whether $\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}}), f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}}))$ is below a fixed value (see (6)) indeed follows a binomial distribution. It is worth noting that when the ratio $\frac{\epsilon}{\sigma}$ is large (e.g., greater than 2), the value of p approaches 1, causing the EMPIRICAL function to return null. To ensure reliable results, in this paper, we carefully choose ϵ and σ values that maintain a sufficiently small $\frac{\epsilon}{\sigma}$ ratio to avoid this. Additionally, we adopt the choice of the Monte Carlo sample size $S = 1000$ and confidence level $c = 0.99$ through out the paper.

Algorithm 2 Empirical order statistics generation

input Confidence level c; Attack radius ϵ; Smoothing noise level σ; noise sample count S;
define Cumulative distribution function of binomial, $F(S, K, p) = \sum_{i=1}^{K} \binom{S}{i} p^i (1-p)^{S-i}$;

function EMPIRICAL(c, σ, S, ϵ)
  $p = \Phi(\frac{\epsilon}{\sigma})$
  $\underline{\hat{K}}, \hat{\overline{K}} = \lceil S \cdot p \rceil, S$
  while $\hat{\overline{K}} - \underline{\hat{K}} > 1$ **do**
    $\dot{K} = \lfloor \frac{(\hat{\overline{K}} - \underline{\hat{K}})}{2} \rfloor$
    if $F(S, \dot{K}, p) > c$ **then**
      $\hat{\overline{K}} = \dot{K}$
    else
      $\underline{\hat{K}} = \dot{K}$
    end if
  end while
  if $\hat{\overline{K}} < S$ **then**
    return $\hat{\overline{K}}$
  else
    return null
  end if
end function

D Proposed algorithms embedding randomized smoothing into training and note on concurrent work

Algorithm 3 Smoothed Adversarial Training

input Training set $\mathcal{D}$; Learning rate α; Estimator $f_{\boldsymbol{\theta}}$ and measurement matrix $\mathbf{A}$; Adversarial noise bound ϵ, iteration count T_{itr}, and step size r; Smoothing noise level σ and noise sample count S;
for $1 \leq i \leq |\mathcal{D}|$ **do**
  $\boldsymbol{\xi}_0 = \mathbf{0}$
  for $1 \leq j \leq T_{itr}$ **do**
    $\Delta = \nabla_{\mathbf{x}} \ell(\hat{f}_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}_{j-1}+\boldsymbol{\delta}}), \mathbf{x})|_{\mathbf{x}=\mathbf{x}_i}$
    (where $\hat{f}_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}_{j-1}+\boldsymbol{\delta}})$ is from (9))
    $\boldsymbol{\xi}'_j = \boldsymbol{\xi}_{j-1} + r * \Delta$
    $\boldsymbol{\xi}_j = \operatorname{argmin}_{\boldsymbol{\xi}:\|\boldsymbol{\xi}\| \leq \epsilon} \|\boldsymbol{\xi} - \boldsymbol{\xi}'_j\|$
  end for
  $\boldsymbol{\xi}^* = \boldsymbol{\xi}_{T_{itr}}$
  $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha * \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}_i+\boldsymbol{\xi}^*}), \mathbf{x}_i)$
end for
return $f_{\boldsymbol{\theta}}$

Algorithm 4 Smoothed Gradient Training

input: Training set $\mathcal{D}$; Learning rate α; Estimator $f_{\boldsymbol{\theta}}$ and measurement matrix $\mathbf{A}$; Smoothing noise level σ_{max}, step count T_{step}, and noise sample count S;

for $1 \leq i \leq |\mathcal{D}|$ **do**
  for $1 \leq j \leq T_{step}$ **do**
    $\mathbf{g}_{\boldsymbol{\theta}} = 0$
    $\sigma = \frac{j}{T_{step}} \sigma_{max}$
    for $1 \leq k \leq S$ **do**
      $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$
      $\Delta = \frac{1}{(ST_{step})} \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}_i+\boldsymbol{\delta}}), \mathbf{x}_i)$
      $\mathbf{g}_{\boldsymbol{\theta}} = \mathbf{g}_{\boldsymbol{\theta}} + \Delta$
    end for
    $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha * \mathbf{g}_{\boldsymbol{\theta}}$
  end for
end for
return $f_{\boldsymbol{\theta}}$

We make several observations both training algorithms described above. Firstly, the underlying idea behind our training methods shares a resemblance with their counterparts in classification: By incorporating randomized smoothing during test time, the neural network is guided to maintain stable behavior, generating similar reconstructions even in the presence of input perturbations. Secondly, for Smoothed Adversarial Training, the perturbed measurement is the one being smoothed, while for Smoothed Gradient Training, the gradients themselves are being smoothed. Thirdly, these algorithms are generally adaptable to any differentiable neural network estimator $f_{\boldsymbol{\theta}}$, allowing for a diverse range of choices in the numerical experiments. Fourthly, a possible alternative approach is to smooth the estimation loss instead. For example, in (8), $\boldsymbol{\xi}^*$ is obtained as $\operatorname{argmax}_{\boldsymbol{\xi}:\|\boldsymbol{\xi}\|_2 \leq \epsilon} \mathbb{E}_{\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})} \big[\ell\big(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\xi}+\boldsymbol{\delta}}), \mathbf{x}\big)\big]$; and in (10), the gradient is computed not before but after the expectation, i.e., $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})} \big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y}_{\mathbf{x}+\boldsymbol{\delta}}), \mathbf{x})\big]$. However, loss smoothing appears to be less effective, because it takes the expected value in a one-dimensional manifold space and as a result, diminishes the "diversity" of the randomness.

In addition, a concurrent study by [24] introduces a training algorithm similar to Algorithm 4, aimed at minimizing the Jittering-risk:

$$\mathbf{J}_{\sigma_{\mathbf{w}}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{w}}^2 \mathbf{I})} \big[\ell(f_{\boldsymbol{\theta}}(\mathbf{y} + \mathbf{w}), \mathbf{x})\big], \tag{25}$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{w}}^2 \mathbf{I})$ represents the (Gaussian) jittering noise and the jittering noise level $\sigma_{\mathbf{w}}$ is a hyperparameter optimized using the validation dataset. This approach is shown to be theoretically justified in [24], particularly in denoising scenarios involving low-dimensional signal subspaces. We note that (10) corresponds to a gradient direction associated with the training objective (25). In our work, we also consider the noise level $\boldsymbol{\delta}_{max}$ as a hyperparameter, comparing different selections in Table 2, which will be discussed in subsequent sections. A notable difference between the two approaches is the step-wise increment of the noise level, which we adopted based on [4] as mentioned above.

E Details on the experiments for training and testing

We perform the experimental evaluation on both denoising and compressed sensing tasks. For each task, we use a suitably-chosen neural network estimator equipped with one training set and two testing sets.

At training time, we consider using all five algorithms discussed in the main body.

At testing time, we apply the PGD method to produce adversarial attacks using a step size and the number of iterations matching those used in the Adv and Smt-Adv algorithms during training. Specifically, at test time, we apply the PGD method to produce adversarial attacks using a step size and number of iterations matching those used in the Adv and Smt-Adv algorithms during training. These hyperparameters (specified below) are selected to ensure that the adversarial noise reaches the boundary of its ℓ_2-radius ball. For the denoising task, we consider ℓ_2 radii $\epsilon < 15$, while for the compressed sensing task, we consider $\epsilon < 7$, avoiding using extreme values and preventing the estimates from being excessively poor. For smoothing, we examine noise levels ranging from 0 to 45 and use Monte Carlo approximation with 50 samples (see Appendix E for a discussion on other choices and the sensitivity to varying S) and confidence level $c = 0.99$. Additionally, since our training process involves an additive noise level of $\sigma_{\mathbf{b}} \leq 15$ and a compression ratio of $m/n \leq 0.5$, we restrict testing to additive noise levels and compression ratios in these ranges.

We follow a common practice of rescaling images to have pixel values in $[0, 1]$. We then compute and analyze both the discrepancy (introduced in Section 3.3), PSNR, and SSIM. The general definition of PSNR depends on the maximum possible pixel value, but our $[0, 1]$-normalization simplifies this to $\mathrm{PSNR} = 10 \log_{10} \frac{1}{\mathrm{MSE}}$, where the MSE is as defined in (1). The definition of SSIM is more complicated to state but is well-known, so is omitted here.

In addition, we select the training and testing parameters with minimal manual tuning, noting that further fine-tuning may improve the performance. The details are given as follows:

- **DPDNN** & **DIV2K, BSD68** (denoising). We use the DPDNN model proposed in [9] as the estimator $f_{\boldsymbol{\theta}}$. For training, we use the (grayscale converted) DIV2K dataset [1], which consists of 1000 images of size 256×256, and is further cropped to generate 32000 patches of size 40×40. For testing, we use the (grayscale converted) DIV2K validation set, which contains 100 images cropped to a size of 384×384, as well as the BSD68 dataset [32], which consists of 68 images cropped to the 256×256 pixels in the center.
  - Training parameters:
    - (Shared across all algorithms) We follow the settings in [9] and set the number of training epochs as 100, batch size as 32, learning rate $\alpha = 0.0005$, measurement matrix $\mathbf{A} = \mathbf{I}$, and additive noise level $\sigma_{\mathbf{b}} = 15$.
    - (For Adv and Smt-Adv only) We set the iteration count $T_{itr} = 100$, step size $r = 0.4$, and adversarial noise ℓ_2 radius $\epsilon = 5$.
    - (For Smt-Adv and Smt-Grad only) We set the step count $T_{step} = 6$, sample count $S = 15$, and the smoothing noise level σ in Smt-Adv to be equal to the maximum smoothing noise level σ_{max} in Smt-Grad. Specifically, $\sigma = \sigma_{\text{max}} = 10$ (in Table 2 we also use 1 and 5 for comparison).
  - Testing parameters:
    - (For measurement) $\mathbf{A} = \mathbf{I}$, and $\sigma_{\mathbf{b}}$ ranges from 0 to 15.
    - (For PGD attack) We set the attack iteration count $T_{itr} = 100$, step size $r = 0.4$, and consider multiple adversarial noise ℓ_2 radii with ϵ ranging from 0 to 15.
    - (For smoothing) We set the smoothing sample count $S = 50$ and consider multiple smoothing noise levels with σ ranging from 0 to 19.
- **ISTA-Net^{++}** & **Train400, Set11, ImageNet** (compressed sensing). The ISTA-Net^{++} model [41] is used as the estimator $f_{\boldsymbol{\theta}}$. We use the Train400 dataset [25], consisting of 400 natural photos of size 180×180 for training, and the Set11 dataset [25] consisting of 11 grayscale images of size 256×256 for testing. Additionally, we use 300 randomly selected images sampled from the "tall building", "open country", and "street" categories from ImageNet [8] for testing. For each of these 300 images, the Y-channel luminance is extracted from the RGB color to form a greyscale-intensity image, and each is cropped to the central 256×256 pixels.

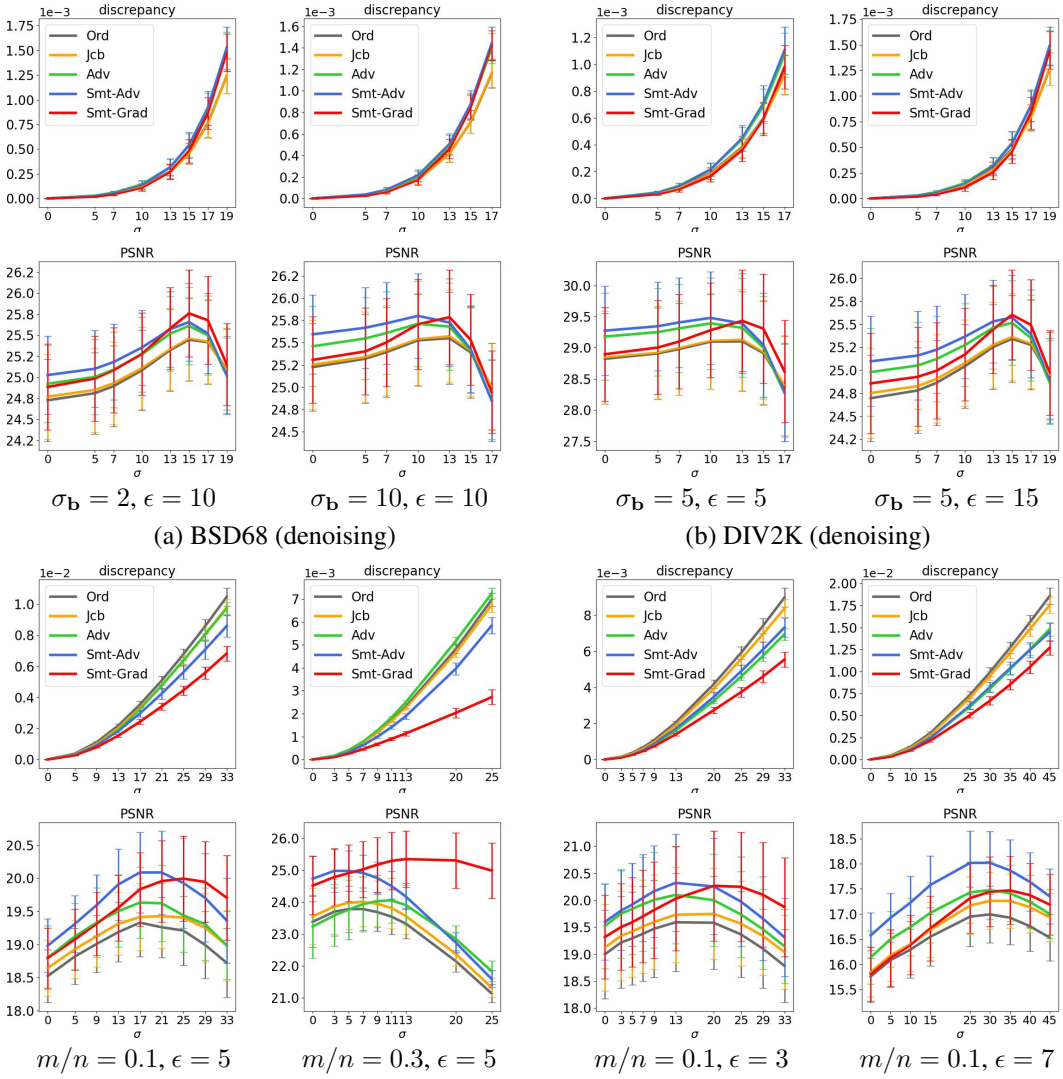
- Training parameters:
  - (Shared across all algorithms) We follow the settings in [41] and set the number of training epoch as 100, batch size as 64, learning rate $\alpha = 0.0001$, additive noise level $\sigma_{\mathbf{b}} = 0$, and Gaussian measurement matrices $\mathbf{A}$ used in each batch with a randomly selected compression ratio from the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. This randomization is done so that the neural network can learn to reconstruct at a variety of compression ratios.
  - (For Adv and Smt-Adv only) We set the iteration count $T_{itr} = 150$, step size $r = 0.2$, and adversarial noise ℓ_2 radius $\epsilon = 1$.
  - (For Smt-Adv and Smt-Grad only) We set the step count $T_{step} = 6$, step size $S = 15$, and the smoothing noise level σ in Smt-Adv to be equal to the maximum smoothing noise level σ_{max} in Smt-Grad. Specifically, $\sigma = \sigma_{max} = 10$ (in Table 2 we also use 5 for comparison).
- Testing parameters:
  - (For measurement) We use a Gaussian matrix $\mathbf{A}$ with compression ratio ranging from 0.1 to 0.4 and no noise, i.e., $\mathbf{b} = \mathbf{0}$.
  - (For PGD attack) We set the iteration count $T_{itr} = 150$, step size $r = 0.2$, and consider multiple adversarial noise ℓ_2 radii with ϵ ranging from 0 to 7.
  - (For smoothing) We set the smoothing sample count $S = 50$ and consider multiple smoothing noise levels with σ ranging from 0 to 45.

All timings are for a single Nvidia GeForce RTX 3090. The code for all these experiments has been uploaded as a supplementary file.

F Plots and tables for experiments

Table 1: Samples of the evaluated PSNR (and discrepancy in brackets) when using the smoothed median estimate vs. the smoothed estimate with certificate, for various tasks.

algorithm	task details (dataset)	smoothed median	smoothed certificate
Ord	$\sigma_{\mathbf{b}} = 10, \epsilon = 10, \sigma = 13$ (BSD68)	25.523 (4.165×1e-4)	25.515 (4.174×1e-4)
Jcb	$m/n = 0.1, \epsilon = 5, \sigma = 17$ (Set11)	19.424 (3.295×1e-3)	19.410 (3.303×1e-3)
Adv	$\sigma_{\mathbf{b}} = 5, \epsilon = 15, \sigma = 15$ (DIV2K)	25.514 (5.402×1e-4)	25.516 (5.410×1e-4)
Smt-Adv	$m/n = 0.1, \epsilon = 7, \sigma = 25$ (ImageNet)	18.116 (6.135×1e-3)	18.011 (6.138×1e-3)
smt-Grad	$\sigma_{\mathbf{b}} = 2, \epsilon = 10, \sigma = 15$ (BSD68)	25.759 (4.812×1e-4)	25.716 (4.821×1e-4)



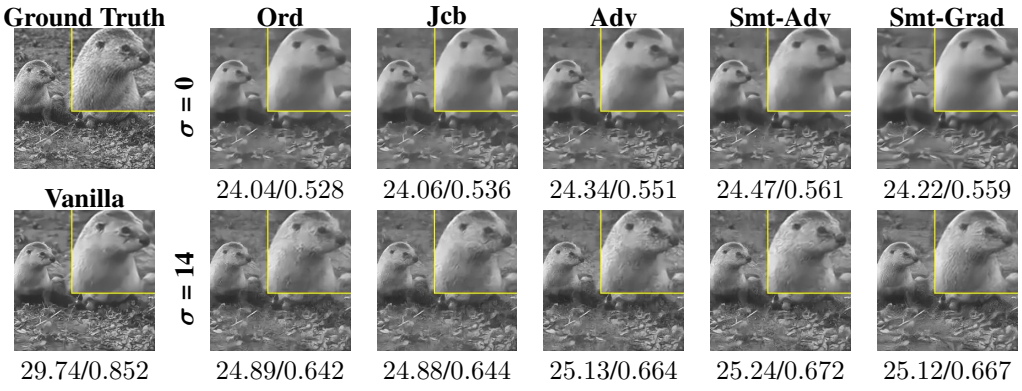
$\sigma_{\mathbf{b}} = 2, \epsilon = 10$ $\sigma_{\mathbf{b}} = 10, \epsilon = 10$ $\sigma_{\mathbf{b}} = 5, \epsilon = 5$ $\sigma_{\mathbf{b}} = 5, \epsilon = 15$

(a) BSD68 (denoising) (b) DIV2K (denoising)

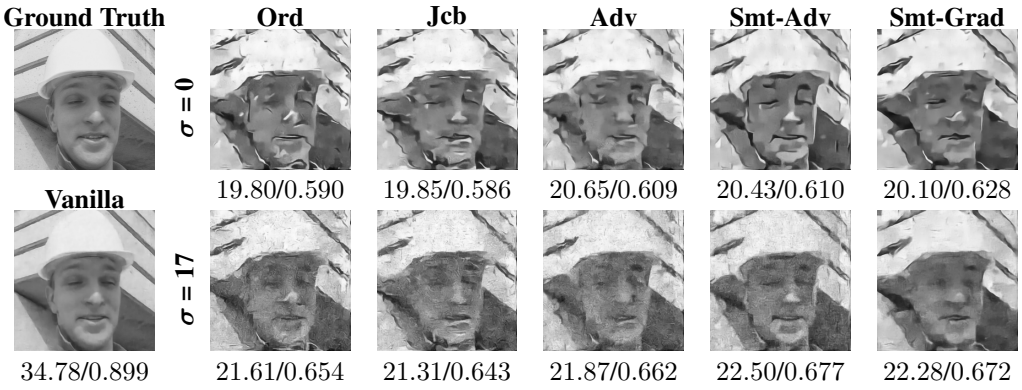
$m/n = 0.1, \epsilon = 5$ $m/n = 0.3, \epsilon = 5$ $m/n = 0.1, \epsilon = 3$ $m/n = 0.1, \epsilon = 7$

(c) Set11 (compressed sensing) (d) ImageNet (compressed sensing)

Figure 1: Certified smoothed discrepancy and its corresponding smoothed PSNR. The experimental settings for each pair of resulted discrepancy (top) and PSNR (bottom) are displayed in the footer.



(b) $\boldsymbol{\sigma_b = 5}$, $\boldsymbol{\epsilon = 9}$ on DIV2K (denoising)



(c) $\boldsymbol{m/n = 0.1}$, $\boldsymbol{\epsilon = 5}$ on Set 11 (compressed sensing)

Figure 2: Estimations (with the performance shown in the form of PSNR/SSIM) from the perturbed measurements in denoising and compressed sensing tasks. **Vanilla** represents the baseline unperturbed scenario where $\boldsymbol{\epsilon = 0}$, $\boldsymbol{\sigma = 0}$, and **Ord** is employed.

Table 2: Trade-off between smoothing noise used during training and the resulting certificates during testing. The second column corresponds to the PSNR of the clean estimate, while the remaining cells indicate the PSNR of the perturbed estimate and the smoothed PSNR of the smoothed estimate.

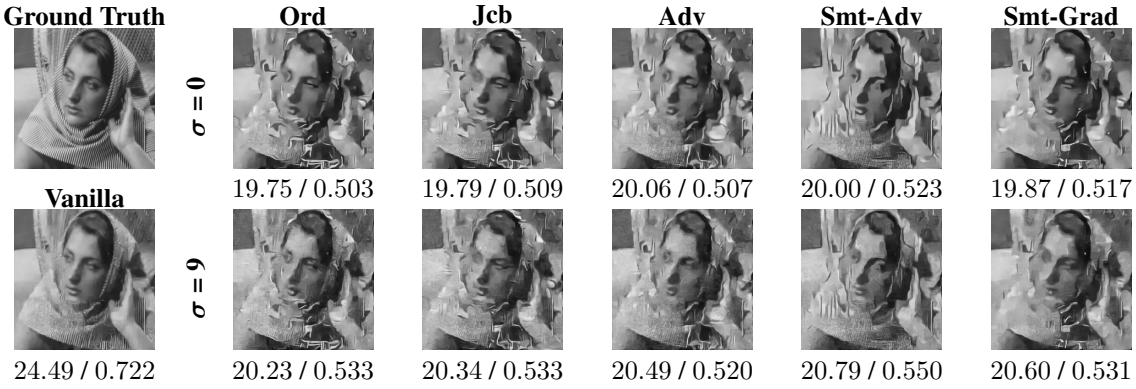
		ℓ_2 radius ϵ at test time					
algorithm trained with σ_{tr}	ϵ=0	ϵ=5		ϵ=11		ϵ=15	
		$\sigma = 0$	$\sigma = 10$	$\sigma = 0$	$\sigma = 14$	$\sigma = 0$	$\sigma = 18$
Ord	32.71	27.48	28.15	24.27	24.90	22.71	23.18
Smt-Adv, $\sigma_{\text{tr}} = 10$	31.87	27.85	28.44	24.56	25.13	22.97	23.36
Smt-Adv, $\sigma_{\text{tr}} = 5$	32.21	27.69	28.39	24.43	25.07	22.84	23.30
Smt-Adv, $\sigma_{\text{tr}} = 1$	32.40	27.48	28.27	24.26	24.97	22.71	23.23
Smt-Grad, $\sigma_{\text{tr}} = 10$	31.74	27.61	28.49	24.48	25.18	22.91	23.40
Smt-Grad, $\sigma_{\text{tr}} = 5$	31.87	27.61	28.39	24.43	25.11	22.86	23.36
Smt-Grad, $\sigma_{\text{tr}} = 1$	31.92	27.54	28.07	24.43	24.87	22.85	23.16

(a) $\sigma_{\mathbf{b}} = 2$ on BSD68 (denoising)

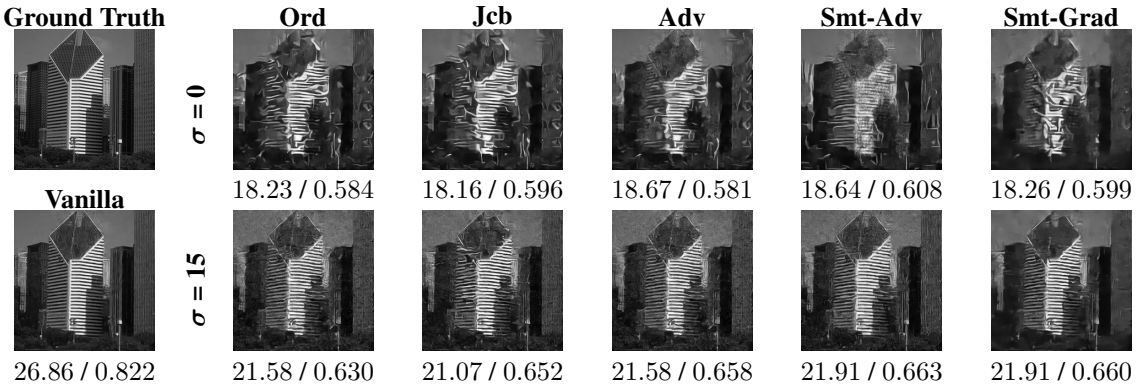
		ℓ_2 radius ϵ at test time					
algorithm trained with σ_{tr}	ϵ=0	ϵ=1		ϵ=5		ϵ=9	
		$\sigma = 0$	$\sigma = 3$	$\sigma = 0$	$\sigma = 21$	$\sigma = 0$	$\sigma = 35$
Ord	27.79	24.64	24.72	18.52	19.28	15.56	16.79
Smt-Adv, $\sigma_{\text{tr}} = 10$	26.91	24.63	24.69	18.98	20.04	16.33	17.83
Smt-Adv, $\sigma_{\text{tr}} = 5$	27.16	24.41	24.52	18.62	19.70	15.67	17.11
Smt-Grad, $\sigma_{\text{tr}} = 10$	27.30	24.85	24.90	18.79	19.99	15.78	17.15
Smt-Grad, $\sigma_{\text{tr}} = 5$	27.48	24.77	24.83	18.65	19.31	15.56	16.51

(b) $m/n = 0.1$ on Set11 (compressed sensing)

We also provide two additional sets of visual comparisons below, Figures 3 and 4, and once again observe that applying randomized smoothing at test time leads to visible visual enhancements in both denoising and compressed sensing tasks across a variety of datasets.

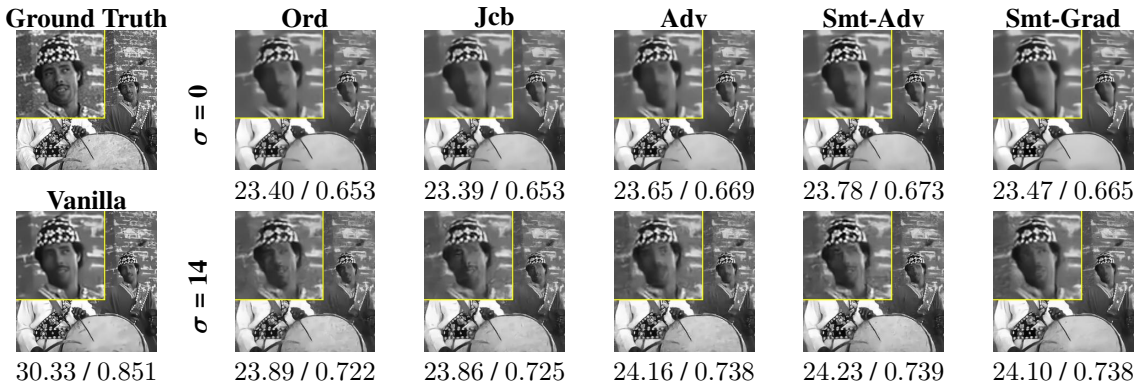


(a) $m/n = \mathbf{0.1}$, $\epsilon = \mathbf{3}$ on Set11 (compressed sensing)

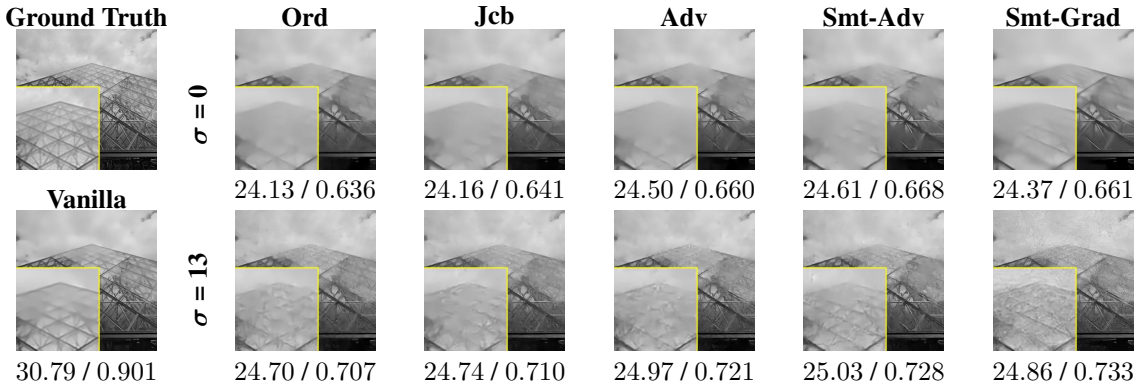


(b) $m/n = \mathbf{0.2}$, $\epsilon = \mathbf{4}$ on ImageNet (compressed sensing)

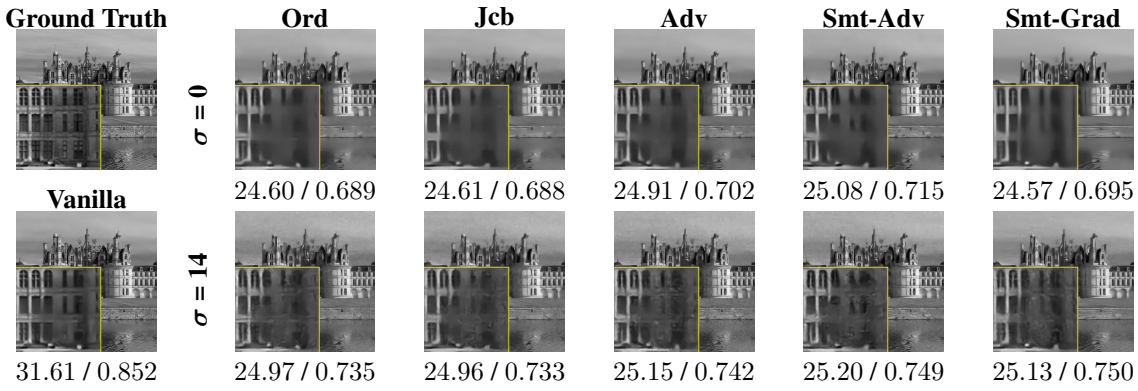
Figure 3: Additional estimations (with the performance shown in the form of PSNR / SSIM) from the perturbed measurements on Set11 and ImageNet in compressed sensing task.



(a) $\boldsymbol{\sigma_{\mathrm{b}} = 2}$, $\boldsymbol{\epsilon = 11}$ on BSD68 (denoising)



(b) $\boldsymbol{\sigma_{\mathrm{b}} = 8}$, $\boldsymbol{\epsilon = 10}$ on BSD68 (denoising)



(c) $\boldsymbol{\sigma_{\mathrm{b}} = 5}$, $\boldsymbol{\epsilon = 11}$ on DIV2K (denoising)

Figure 4: Additional estimations (with the performance shown in the form of PSNR / SSIM) from the perturbed measurements on BSD68 and DIV2K in denoising task.

G Discussion of computation

Our experiments have shown that the utilization of randomized smoothing during both training and testing phases of linear inverse problems effectively improves the robustness of the estimator against adversarial attacks. A slight caveat is that the incorporation of smoothing can increase the computation time, as is also the case with smoothing in other contexts. For training, we found that Smt-Grad was not too much slower than its unsmoothed counterpart, in part due to performing will with relatively few smoothing samples. On the other hand, the smoothing operation made Smt-Adv noticeably slower than Adv. Regarding test-time performance, determining the empirically ideal smoothing noise level, as depicted in Figure 1, can also be a time-consuming process.

To support the above discussion, we present the training time per batch across all five algorithms in two tasks in Table 3 below, where we observe that the training time required for Smt-Adv and Smt-Grad is higher than the baselines but still reasonable.

Table 3: Training time per batch in denoising (top, batch size 32) and compressed sensing (bottom, batch size 64), where sample count of $S = 15$ for Smt-Adv and Smt-Grad is utilized in both tasks.

Estimator & Training Set	Ord	Jcb	Adv	Smt-Adv	Smt-Grad
DPDNN & DIV2K	0.49s	0.83s	3.53s	46.81s	45.48s
ISTA-Net^{++} & Train400	0.08s	0.15s	0.27s	2.49s	5.41s

We also provide Table 4 below showcasing the testing time per data sample, where we observe that reducing smoothing samples from 1000 to 50 leads to a significant speedup of around $15\times$, while maintaining comparable performance.

Specifically, regarding the choice of the number of Monte Carlo samples (S) during testing, Table 4 showcases the certified discrepancy and corresponding PSNR with S ranging from 50 to 1000 in chosen tasks. The results reveal that PSNR, in fact, shows very little degradation between smaller vs. higher values of S. For certified discrepancy, a downward trend is observed as expected (i.e., decreasing S gives a worse guarantee), yet not a particularly drastic one. Notably, lower S values significantly reduce the computation time as we see that reducing S from 1000 to 50 can give a roughly $15\times$ speedup. Consequently, we opt for $S = 50$ in all experiments due to its consistent effectiveness, particularly with respect to PSNR (higher S may still be preferred if one specifically needs a strong certified guarantee).

Table 4: Estimated certified discrepancy and PSNR ($\pm$ std.) with different testing samples (S) and the corresponding testing time per data sample (t) in two selected scenarios (a) and (b), where the corresponding best choices of σ (12 and 19) are used.

	S=50 (t=2.20s)		S=500 (t=16.93s)		S=1000 (t=33.52s)	
	discrepancy	*PSNR*	discrepancy	*PSNR*	discrepancy	*PSNR*
Ord	5.085($\pm$0.948)	*25.549($\pm$1.039)*	5.083($\pm$0.945)	*25.552($\pm$1.032)*	5.072($\pm$0.923)	*25.552($\pm$1.036)*
Jcb	4.961($\pm$0.822)	*25.574($\pm$1.045)*	4.958($\pm$0.812)	*25.572($\pm$1.045)*	4.955($\pm$0.812)	*25.573($\pm$1.044)*
Adv	4.626($\pm$0.770)	*25.671($\pm$0.980)*	4.611($\pm$0.766)	*25.679($\pm$0.971)*	4.613($\pm$0.766)	*25.678($\pm$0.972)*
Smt-Adv	4.191($\pm$0.585)	*25.724($\pm$0.987)*	4.179($\pm$0.581)	*25.727($\pm$0.981)*	4.180($\pm$0.567)	*25.727($\pm$0.982)*
Smt-Grad	4.184($\pm$0.595)	*25.783($\pm$1.052)*	4.183($\pm$0.592)	*25.781($\pm$1.047)*	4.166($\pm$0.593)	*25.781($\pm$1.050)*

(a) discrepancy (1e-4) and PSNR with $\sigma_{\mathbf{b}} = 10$, $\epsilon = 10$, $\sigma = 12$ on BSD68 (denoising)

Ord	4.328($\pm$0.463)	*19.334($\pm$1.017)*	4.299($\pm$0.465)	*19.343($\pm$0.940)*	4.289($\pm$0.461)	*19.341($\pm$0.955)*
Jcb	4.039($\pm$0.440)	*19.462($\pm$1.068)*	4.022($\pm$0.379)	*19.472($\pm$1.029)*	4.017($\pm$0.387)	*19.478($\pm$0.986)*
Adv	3.614($\pm$0.317)	*19.561($\pm$1.102)*	3.590($\pm$0.312)	*19.593($\pm$1.089)*	3.589($\pm$0.320)	*19.598($\pm$1.082)*
Smt-Adv	4.030($\pm$0.560)	*20.081($\pm$1.279)*	3.995($\pm$0.589)	*20.163($\pm$1.233)*	3.988($\pm$0.563)	*20.161($\pm$1.223)*
Smt-Grad	2.929($\pm$0.404)	*19.904($\pm$1.114)*	2.917($\pm$0.407)	*19.911($\pm$1.105)*	2.910($\pm$0.356)	*19.943($\pm$1.110)*

(b) discrepancy (1e-3) and PSNR with $m/n = 0.1$, $\epsilon = 5$, $\sigma = 19$ on Set11 (compressed sensing)