Certified Defense to Image Transformations via Randomized Smoothing

Marc Fischer, Maximilian Baader, Martin Vechev
Department of Computer Science
ETH Zurich
{marc.fischer, mbaader, martin.vechev}@inf.ethz.ch

Abstract

We extend randomized smoothing to cover parameterized transformations (e.g., rotations, translations) and certify robustness in the parameter space (e.g., rotation angle). This is particularly challenging as interpolation and rounding effects mean that image transformations do not compose, in turn preventing direct certification of the perturbed image (unlike certification with ℓ^p norms). We address this challenge by introducing three different kinds of defenses, each with a different guarantee (heuristic, distributional and individual) stemming from the method used to bound the interpolation error. Importantly, we show how individual certificates can be obtained via either statistical error bounds or efficient online inverse computation of the image transformation. We provide an implementation of all methods at `https://github.com/eth-sri/transformation-smoothing`.

1 Introduction

Deep neural networks are vulnerable to adversarial examples [1] – small changes that preserve semantics (e.g., ℓ^p-noise or geometric transformations such as rotations) [2], but can affect the output of a network in undesirable ways. As a result, there has been substantial recent interest in methods which aim to ensure the network is certifiably robust to adversarial examples [3–13].

Certification guarantees There are two principal robustness guarantees a certified defense can provide at inference time: (i) the (standard) distributional guarantee, where a robustness score is computed offline on the test set to be interpreted in expectation for images drawn from the data distribution, and (ii) an individual guarantee, where a certificate is computed online for the (possibly perturbed) input. The choice of guarantee depends on the application and regulatory constraints.

Guarantees with ℓ^p norms When considering ℓ^p norms, existing certification methods can be directly used to obtain either of the above two guarantees: for an image $\boldsymbol{x}$ and adversarial noise δ, $\|\delta\|_p < r$, proving that a classifier f is r-robust around $\boldsymbol{x}' := \boldsymbol{x} + \delta$ is enough to guarantee $f(\boldsymbol{x}) = f(\boldsymbol{x}')$. That is, it suffices to prove robustness of a perturbed input in order to certify that the perturbation did not change the classification, as the r-ball around $\boldsymbol{x}'$ includes $\boldsymbol{x}$.

Key challenge: guarantees for geometric perturbations Perhaps not intuitively, however, for more complex perturbations such as geometric transformations, proving robustness around an image $\boldsymbol{x}'$ via existing methods (e.g., [9–12]) does not imply that $f(\boldsymbol{x}) = f(\boldsymbol{x}')$ for the original image $\boldsymbol{x}$. To illustrate this issue, consider the rotation R_γ, by angle γ of an image $\boldsymbol{x}$, followed by an interpolation I. Certifying that the classification of the rotated image $\boldsymbol{x}' := I \circ R_\gamma(\boldsymbol{x})$ for $\|\gamma\| < r$ is robust under further rotations $I \circ R_\beta$ for $\|\beta\| < r$ is not sufficient to imply that $\boldsymbol{x}$ and $\boldsymbol{x}'$ classify the same, as rotating $\boldsymbol{x}'$ back by $\beta = -\gamma$ does not return the original image $\boldsymbol{x}$ due to interpolation. A central challenge then is to develop techniques that are able to handle more involved perturbations.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

This work: certification beyond ℓ^p norms In this work we address this challenge and introduce the first certification methods for geometric transformations based on randomized smoothing (RS): we extend RS [7] to handle transformations (SPT) by adding (Gaussian) noise to transformation parameters, enabling us to handle large models and datasets (e.g., ImageNet). Our methods, their guarantees and how they compare to standard RS [7] (on ℓ^p norms) and other techniques, are shown in Table 1.

Table 1: Certificates obtained by different methods. * indicates deterministic certification, other methods hold with high confidence.

	dist.	indiv.
Composable perturbation ψ (e.g., additive ℓ^p-bound)		
relaxation-based* [3–6]	✓	✓
Cohen et al. [7]	✓	✓
Non-composable ψ (e.g., rotation $I \circ R$)		
INDIVSPT	(✓)	✓
DISTSPT$^{\mathcal{D}}$	✓ w.p. q_E	✓ w.p. q_E
DISTSPT$^{\boldsymbol{x}}$	✓	✗
relaxation-based* [9–12]	✓	✗
RS-based [13, 14]	✓	✗

BASESPT As with standard RS over ℓ^p norms, SPT (not listed) provides individual and distributional guarantees, but only for composable parametric transformations, that is, where: ψ_γ: $\psi_{\beta+\gamma} = \psi_\gamma \circ \psi_\beta$. For non-composable ones (e.g., rotations with interpolation), BASESPT is only a heuristic defense, motivating the need for appropriate certification methods.

INDIVSPT This method provides the strongest guarantees for non-composable transformations and works as follows: at inference time, for each input $\boldsymbol{x}'$, it calculates an individual upper bound of the expression ϵ *without access* to (the original) $\boldsymbol{x}$, then combined with SPT and smoothing. A key step here is computing the inverse $\psi_\gamma^{-1}(\boldsymbol{x}')$ of $\boldsymbol{x}'$, for which we introduce an efficient technique.

DISTSPT While desirable (it mimics original RS guarantees), INDIVSPT can be expensive to apply at inference time and obtain tight certificates with. This motivates the study of more relaxed, still useful certification guarantees, as well as corresponding methods which achieve tighter bounds using these definitions. The idea of DISTSPT is to estimate a probabilistic upper bound for the expression $\epsilon = \|\psi_\beta \circ \psi_\gamma(\boldsymbol{x}) - \psi_{\beta+\gamma}(\boldsymbol{x})\|_2$, combined with SPT and RS. The first variant here is DISTSPT$^{\mathcal{D}}$, where this upper bound is estimated offline on the training dataset and holds for all $\boldsymbol{x}$ from the data distribution $\mathcal{D}$, with probability q_E. This method enjoys both probabilistic distributional and individual guarantees. The weakening of the definition used by INDIVSPT (now probabilistic over q_E) enables the method to compute tighter bounds. The second variant, DISTSPT$^{\boldsymbol{x}}$, provides weaker guarantees than DISTSPT$^{\mathcal{D}}$, with the provided bound now computed for individual $\boldsymbol{x}$ on the test set. It obtains a distributional guarantee, however, it does not provide individual guarantees – this restriction allows DISTSPT$^{\boldsymbol{x}}$ to compute even tighter bounds. We remark that recent methods targeting robustness to geometric transformations (e.g., [11, 13, 14] also fall in this class.

To summarize, our core contributions are:

- A generalization of randomized smoothing to parameterized transformations.
- A number of novel certification methods for non-composable parameterized transformations, systematically exploring both distributional and individual guarantees while considering deterministic and probabilistic bounds. In the process, we highlight the rich interplay between certification definitions and tightness of the corresponding certificates.
- A thorough evaluation of all methods on common image datasets, showcasing certified robustness to $\pm 30^\circ$ rotations for 50% of inputs on Restricted ImageNet.

2 Related Work

We now survey the most closely related work in neural network certification and defenses.

ℓ^p norm based certification and defenses The discovery of adversarial examples [1, 15] triggered interest in training and certifying robust neural networks. An attempt to improve model robustness are empirical defenses [16, 17], strategies which harden a model against an adversary. While this may improve robustness to current adversaries, typically robustness cannot be formally verified with current certification methods. This is because complete methods [18–20] do not scale and incomplete methods relying on over approximation lose too much precision [3, 21, 22, 6, 10, 23],

even for networks trained to be amenable to certification. Recently, randomized smoothing was introduced, which could for the first time, certify a (smoothed) classifier against norm bound ℓ^2 noise on ImageNet [24, 25, 7, 8, 26], by relaxing exact certificates to high confidence probabilistic ones. Smoothing scales to large models, however, it is currently limited to norm-based perturbations.

Semantic perturbations Transformations such as translations and rotations can produce adversarial examples [2, 27]. An enumerative approach certifying against semantic perturbations was presented in [9]. There, the search space is reduced by only consider next neighbor interpolation. Unfortunately, for more elaborate interpolations (e.g., bilinear), the approach becomes infeasible. The first certification against rotations with bilinear interpolations was carried out in [10], later significantly improved on by [11]. Both methods generate linear relaxations and propagate them through the network. However, the methods do not yet scale to large networks (i.e., ResNet-50) or complex data sets (i.e., ImageNet). The approaches of [12] and [10] are similar for rotation. [13, 14] reduce transformations to multiple ℓ_2-balls which they certify via RS so to obtain a certificate for the overall transformation. As outlined in Table 1, all these methods result in a distributional but not an individual guarantee.

3 Generalization of Smoothing

A smoothed classifier $g\colon \mathbb{R}^m \mapsto \mathcal{Y}$ can be constructed out of an ordinary classifier $f\colon \mathbb{R}^m \mapsto \mathcal{Y}$, by calculating the most probable result of $f(\boldsymbol{x}+\epsilon)$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{1})$:

$$g(\boldsymbol{x}) := \arg\max_c \mathbb{P}_{\epsilon \sim \mathcal{N}(0,\sigma^2\mathbb{1})}(f(\boldsymbol{x}+\epsilon) = c).$$

One then obtains the following robustness guarantee:

Theorem 3.1 (From [7])**.** *Suppose $c_A \in \mathcal{Y}$, $\underline{p_A}, \overline{p_B} \in [0,1]$. If*

$$\mathbb{P}_\epsilon(f(\boldsymbol{x}+\epsilon) = c_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c \neq c_A} \mathbb{P}_\epsilon(f(\boldsymbol{x}+\epsilon) = c),$$

then $g(\boldsymbol{x}+\delta) = c_A$ for all δ satisfying $\|\delta\|_2 \leq \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) =: r_\delta$.

We now generalize this theorem to parameterized transformations. Consider the composable transformations $\psi_\beta : \mathbb{R}^m \to \mathbb{R}^m$, satisfying $\psi_\beta \circ \psi_\gamma = \psi_{\beta+\gamma}$ for all $\beta, \gamma \in \mathbb{R}^d$. Then we can define a smoothed classifier $g : \mathbb{R}^m \to \mathcal{Y}$ analogously for a parametric transformation ψ_β by

$$g(\boldsymbol{x}) = \arg\max_c \mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})} \left(f \circ \psi_\beta(\boldsymbol{x}) = c\right). \tag{1}$$

With that, we obtain the following robustness guarantee:

Theorem 3.2. *Let $\boldsymbol{x} \in \mathbb{R}^m$, $f : \mathbb{R}^m \to \mathcal{Y}$ be a classifier and $\psi_\beta : \mathbb{R}^m \to \mathbb{R}^m$ be a composable transformation as above. If*

$$\mathbb{P}_\beta(f \circ \psi_\beta(\boldsymbol{x}) = c_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c_B \neq c_A} \mathbb{P}_\beta(f \circ \psi_\beta(\boldsymbol{x}) = c_B),$$

then $g \circ \psi_\gamma(\boldsymbol{x}) = c_A$ for all γ satisfying $\|\gamma\|_2 \leq \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) =: r_\gamma$. Further, if g is evaluated on a proxy classifier f' that behaves like f with probability $1-\rho$ and else returns an arbitrary answer, then $r_\gamma := \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A} - \rho) - \Phi^{-1}(\overline{p_B} + \rho))$.

The proof is similar to the one presented in Cohen et al. [7] and is given in App. A. The key difference is that we allow parameterized transformations ψ, while Cohen et al. [7] only allows additive noise.

4 Certification with interpolation and rounding errors

We now instantiate Theorem 3.2 for parameterized geometric image transformations T_β, $\beta \in \mathbb{R}^d$, followed by interpolation I, denoted as T_β^I. A geometric transformation T_β is followed by an interpolation I in order to express the result on the pixel grid. In general, even if T_β composes, T_β^I does not (see Fig. 1 in the case where T_β is a rotation R_β by an angle β). This prevents us from directly instantiating Theorem 3.2 with $\psi_\beta := T_\beta^I$.

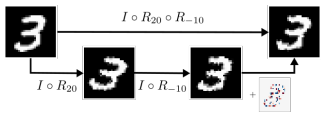


Figure 1: Rotations with interpolation do not compose.

To address this issue, we now show how to construct a classifier g_E with the desired guarantees, namely that $g_E \circ T^I_\gamma(\boldsymbol{x}) = g_E(\boldsymbol{x})$ for γ with $\|\gamma\|_2 \leq r_\gamma$, thus enabling certification of image transformations (which may not compose). Our proposed construction consists of two steps.

First, for a fixed but arbitrary $\boldsymbol{x}$, let h_E be a classifier satisfying interpolation invariance:

$$h_E \circ T^I_\beta \circ T^I_\gamma(\boldsymbol{x}) = h_E \circ T^I_{\beta+\gamma}(\boldsymbol{x}) \;\; \forall \beta, \gamma \in \mathbb{R}^d. \tag{2}$$

We now instantiate Theorem 3.2 with $f := h_E \circ I$ and $\psi_\beta := T_\beta$, obtaining a smoothed classifier $g_E(\boldsymbol{x}) := \arg\max_c \mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})} \left(h_E \circ I \circ T_\beta(\boldsymbol{x}) = c\right)$, such that $g_E \circ T_\gamma(\boldsymbol{x}) = c_A = g_E(\boldsymbol{x})$ for γ with $\|\gamma\|_2 \leq r_\gamma$ by Theorem 3.2. Further, since

$$\begin{aligned} g_E \circ T_\gamma(\boldsymbol{x}) &= \arg\max_c \mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})} \left(h_E \circ I \circ T_\beta \circ T_\gamma(\boldsymbol{x}) = c\right) \\ &= \arg\max_c \mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})} \left(h_E \circ T^I_\beta \circ T^I_\gamma(\boldsymbol{x}) = c\right) \\ &= g_E \circ T^I_\gamma(\boldsymbol{x}), \end{aligned}$$

where the first and last equities hold by the definition of g_E and the second one due to Eq. (2). Thus, we obtain a classifier g_E with the desired property.

Second, we discuss the construction of the desired h_E (from step 1). Consider the interpolation error

$$\epsilon(\beta, \gamma, \boldsymbol{x}) := T^I_\beta \circ T^I_\gamma(\boldsymbol{x}) - T^I_{\beta+\gamma}(\boldsymbol{x}), \tag{3}$$

$$\text{bounded by } E \in \mathbb{R}^{\geq 0} \text{ s.t. } \forall \beta, \gamma \in \mathbb{R}^d.\ \|\epsilon(\beta, \gamma, \boldsymbol{x})\|_2 \leq E \tag{4}$$

for a given but arbitrary $\boldsymbol{x}$. Thus if h_E is ℓ^2-robust with radius E around $T^I_{\beta+\gamma}(\boldsymbol{x})$, interpolation invariance holds. While many choices for such h_E are possible in the rest of the paper we instantiate h_E by applying Theorem 3.1 to a base classifier b.

Obtaining probabilistic guarantees from Theorem 3.2 So far we assumed that $\boldsymbol{x}$ is arbitrary but fixed and constructed E and h_E for this $\boldsymbol{x}$ specifically. In general, finding a tight deterministic bound E that holds $\forall \beta, \gamma$ is computationally challenging. Thus, we relax this deterministic guarantee into a probabilistic one:

$$\mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})} \left(\|\epsilon(\beta, \gamma, \boldsymbol{x})\|_2 \leq E\right) \geq 1 - \rho_E \;\; \forall \gamma \in \mathbb{R}^d. \tag{5}$$

meaning Eq. (4) holds with probability at least $1 - \rho_E$, in turn implying that Eq. (2) also holds at least with probability $1 - \rho_E$. This can also be formulated as having a proxy classifier h'_E which behaves like h_E with probability at least $1 - \rho_E$ on the inputs specified by Eq. (2). In practice, we construct h'_E which behaves like h_E with probability at least $1 - \rho_E$ on all inputs, implying this behavior on the inputs from Eq. (2). From h'_E, we then obtain $f' := h'_E \circ I$ which behaves like f with probability at least $1 - \rho_E$ on all inputs. Then, we can apply Theorem 3.2 by setting ρ to ρ_E and obtain the desired guarantee. In Section 5, we show how to obtain E for DISTSPT and INDIVSPT.

5 Calculation of error bounds

In Section 5.1 we derive a distributional error bound over a dataset and in Section 5.2 a per-image bound. Throughout this section, we assume the attacker model $\gamma \in \Gamma \subseteq \mathbb{R}^d$. As we compute E with this assumption, our obtained certificate proves robustness of g_E to T^I_γ for $\gamma \in \Gamma$ with $\|\gamma\|_2 \leq r_\gamma$.

5.1 Distributional bounds for DISTSPT

For a fixed $E \in \mathbb{R}^{\geq 0}, \rho_E \in [0, 1]$, the probability that ϵ is bounded by E for $\boldsymbol{x} \sim \mathcal{D}$ is

$$q_E := \mathbb{P}_{\boldsymbol{x} \sim \mathcal{D}}(\mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})}(\max_{\gamma \in \Gamma} \|\epsilon(\beta, \gamma, \boldsymbol{x})\|_2 \leq E\ ) \geq 1 - \rho_E). \tag{6}$$

In practice, for DISTSPT$^{\mathcal{D}}$ we evaluate q_E by sampling $\boldsymbol{x}$ and counting how often the inner property holds. We compute the inner probability by: (i) sampling multiple realizations of β, (ii) computing their corresponding error ϵ and checking how many are successfully bounded by E, and (iii) bounding the inner probability using Clopper-Pearson. If this lower bound is larger than $1 - \rho_E$ we count this

as a positive sample, else a negative one. Once these counts are obtained for a number of sampled points $\boldsymbol{x}$, we can apply Clopper-Pearson and obtain a lower bound $\underline{q_E}$ with the desired confidence. For DISTSPT$^{\boldsymbol{x}}$ only the inner probability needs to be computed for an individual image $\boldsymbol{x}$. Formally this can be seen as considering the data distribution $\mathcal{D}$ that just contains $\boldsymbol{x}$ (thus $q_E = 1$).

To compute the maximization over γ we employ standard interval analysis, which allows us to efficiently propagate lower and upper bounds [28]. By propagating the hyperrectangle containing Γ along with the sampled β and $\boldsymbol{x}$, we eventually obtain a lower and upper bound for the norm calculation of which we take the maximum:

$$\max_{\gamma \in \Gamma} \|\epsilon(\beta, \gamma, \boldsymbol{x})\|_2 \leq \max \|T^I_\beta \circ T^I_\Gamma(\boldsymbol{x}) - T^I_{\beta+\Gamma}(\boldsymbol{x})\|_2. \tag{7}$$

The result can be refined by splitting the hyperrectangle Γ into smaller hyperrectangles Γ_k for $k \in \{0, \ldots, N\}$. The refined bound is

$$\max_{\gamma \in \Gamma} \|\epsilon(\beta, \gamma, \boldsymbol{x})\|_2 \leq \max_{k \in \{0,\ldots,N\}} \max \|T^I_\beta \circ T^I_{\Gamma_k}(\boldsymbol{x}) - T^I_{\beta+\Gamma_k}(\boldsymbol{x})\|_2. \tag{8}$$

To obtain E in the first place, we perform the same sampling operations as above (sample $\boldsymbol{x}$ and β) but do not compute any probabilities, that is, for each sample ($\boldsymbol{x}$, β), we simply keep the values attained by Eq. (8).

For DISTSPT$^{\mathcal{D}}$ we pick an E that bounds many of these values, choosing ρ_E to be small. Once E is obtained, we compute q_E as described above. Instantiating the construction of Section 4 with this E yields the guarantee that for a random image $\boldsymbol{x} \sim \mathcal{D}$ the guarantees provided by Theorem 3.2 hold with probability q_E.

For DISTSPT$^{\boldsymbol{x}}$, after we determine a suitable E for the given $\boldsymbol{x}$ we can determine ρ_E.

5.2 Individual bounds for INDIVSPT

At inference time, we are given $\boldsymbol{x}' := T^I_\gamma(\boldsymbol{x})$ but neither the original $\boldsymbol{x}$ nor the parameter $\gamma \in \Gamma$, and we would like to certify that $g_E(\boldsymbol{x}') = g_E(\boldsymbol{x})$. When ψ_β composes as required in Section 3, this can be certified by showing g is robust with a sufficient radius r_γ. However, when ψ_β does not compose, this can be accomplished by applying Theorem 3.2 to show $g_E(\boldsymbol{x})$ is robust with radius r_γ that includes Γ. In turn, this requires a bound E (see Eq. (5)) for $\boldsymbol{x}$ (rather than $\boldsymbol{x}'$):

$$\mathbb{P}_{\beta \sim \mathcal{N}(0, \sigma^2 \mathbb{1})} \left(\max_{\gamma \in \Gamma} \|\epsilon(\beta, \gamma, \boldsymbol{x})\|_2 \leq E \right) \geq 1 - \rho_E. \tag{9}$$

Now, we would like to compute an upper bound on the $\max$ term *without* having access to $\boldsymbol{x}$. This is accomplished as follows: First, in the above equation, we replace ϵ by its definition (Eq. (3)) and $T^I_\gamma(\boldsymbol{x})$ by $\boldsymbol{x}'$. We then replace $\boldsymbol{x}$ with a symbolic set of possible inputs that could have generated $\boldsymbol{x}'$, denoted as $(T^I_\Gamma)^{-1}(\boldsymbol{x}') := \{\boldsymbol{x} \in \mathbb{R}^m \mid T^I_\gamma(\boldsymbol{x}) = \boldsymbol{x}', \gamma \in \Gamma\}$ which we can use instead of $\boldsymbol{x}$ due to the maximization over γ. As in Section 5.1, we obtain the resulting bound via interval analysis:

$$\max_{\gamma \in \Gamma} \|\epsilon(\beta, \gamma, \boldsymbol{x})\|_2 \leq \max \|T^I_\beta(\boldsymbol{x}') - T^I_{\beta+\Gamma} \circ (T^I_\Gamma)^{-1}(\boldsymbol{x}')\|_2. \tag{10}$$

The computation of the inverse $(T^I_\Gamma)^{-1}(\boldsymbol{x}')$ is explained in Section 6. By substituting Eq. (10) in Eq. (9) we can obtain and verify E as in Section 5.1 (except we do not need to sample $\boldsymbol{x}$'s). As before, we can refine the upper bound of Eq. (10) by splitting Γ into Γ_k. We note as the inverse does not depend on β, given $\boldsymbol{x}'$, it only needs to be computed once and can be reused whenever we evaluate Eq. (10) for a given sample β.

6 Inverse Computation

We now discuss how to obtain a set containing all possible inverse images. That is, given $\boldsymbol{x}' := T^I_\gamma(\boldsymbol{x})$ and $\gamma \in \Gamma$, we compute the set $(T^I_\Gamma)^{-1}(\boldsymbol{x}')$ which contains all possible $\boldsymbol{x}$. First, we cover the necessary background. To ease presentation, we assume even image height and width. We embed the images in $\mathbb{R}^2$ by centering them at 0 on an odd integer grid $G := (2\mathbb{Z} + 1) \times (2\mathbb{Z} + 1)$ and centered at 0. We denote the value of a pixel at $(i, j) \in G$ by $p_{i,j} \in [0, 1]$.

(a) Rotated (b) Inverse (c) $10\times$ refined inverse (d) Original

Figure 2: Over approximation of the inverse image. The image pairs (b) and (c) depict the lower (left) and upper (right) interval pixel bounds for the inverse image and the $10\times$ refined image respectively.

Transformations The pixel values $p'_{i',j'}$ for $(i',j') \in G$ of an image, produced by a transformation $T_\gamma \colon \mathbb{R}^2 \to \mathbb{R}^2$ with parameter $\gamma \in \mathbb{R}^d$, is calculated by interpolating at the inversely transformed coordinate $T_\gamma^{-1}(i',j')$, followed by the interpolation I resulting in $p'_{i',j'} = I \circ T_\gamma^{-1}(i',j')$.

Bilinear interpolation A prominent interpolation is *bilinear interpolation*, given by

$$I(x,y) = p_{v,w}\tfrac{2+v-x}{2}\tfrac{2+w-y}{2} + p_{v,w+2}\tfrac{2+v-x}{2}\tfrac{y-w}{2} + p_{v+2,w}\tfrac{x-v}{2}\tfrac{2+w-y}{2} + p_{v+2,w+2}\tfrac{x-v}{2}\tfrac{y-w}{2}, \quad (11)$$

where $(v,w) \in G$ is the coordinate such that (x,y) lies in the (v,w)-interpolation region, that is $(x,y) \in [v,v+2] \times [w,w+2]$. We use v and w as grid indices in the context of the interpolation I. If $p_{v,w}$ has no defined value because (v,w) is out of range for the image, we set $p_{v,w}$ to 0.

We start by giving a procedure to calculate constraints of a single pixel (i,j) for a single color channel, after which we present an iterative procedure to refine that constraint. The inverse image is then obtained by following this procedure for every pixel in every color channel. We illustrate the steps in Section 6.1 using the example of a rotated image $\boldsymbol{x}'$ (Fig. 2a).

The attacker transformed the original image $\boldsymbol{x}$ (Fig. 2d) using T_γ^I for $\gamma \in \Gamma$ and therefore obtained the pixel values $p'_{i',j'}$ of the transformed image $\boldsymbol{x}'$ by evaluating $p'_{i',j'} = I \circ T_\gamma^{-1}(i',j')$. The interpolation I uses the pixel values $p_{i,j}$ of $\boldsymbol{x}$. The following steps invert this relation for every coordinate (i,j):

Step 1 For every $(i',j') \in G$, we over-approximate the region the pixel value $p'_{i',j'}$ could have been interpolated from, which is $c_{i',j'} := T_\Gamma^{-1}(i',j')$, $C := \{c_{i',j'} \mid (i',j') \in G\}$. In practice, only a finite subset of C is used. In App. B, we show how to calculate this subset efficiently.

Step 2 The interpolation I is defined piecewise per (v,w)-interpolation region $[v,v+2]\times[w,w+2]$, so the algebraic form of I, Eq. (11) holds for each interpolation region separately. For every interpolation region cornering (i,j) that $c_{i',j'}$ intersects with, the pixel value $p'_{i',j'}$ yields constraints for value $p_{i,j}$. Here, we describe just the constraint $q_{i,j}$ associated with the (i,j)-interpolation region; others ($(i-2,j-2),(i-2,j),(i,j-2)$) work analogously. First, for every $c_{i',j'} \in C$ we calculate its intersection with the (i,j)-interpolation region, yielding

$$[x_l,x_u] \times [y_l,y_u] := c_{i',j'} \cap [i,i+2] \times [j,j+2].$$

We can plug this into the interpolation I, where we instantiate $(v,w) \leftarrow (i,j)$, resulting into

$$\begin{aligned} p'_{i',j'} \in I([x_l,x_u],[y_l,y_u]) = {} & p_{i,j}\tfrac{2+i-[x_l,x_u]}{2}\tfrac{2+j-[y_l,y_u]}{2} + p_{i,j+2}\tfrac{2+i-[x_l,x_u]}{2}\tfrac{[y_l,y_u]-j}{2} \\ & + p_{i+2,j}\tfrac{[x_l,x_u]-i}{2}\tfrac{2+j-[y_l,y_u]}{2} + p_{i+2,j+2}\tfrac{[x_l,x_u]-i}{2}\tfrac{[y_l,y_u]-j}{2}. \end{aligned} \quad (12)$$

Next, we solve for the pixel value of interest $p_{i,j}$. Then, we replace all other three pixel values $p_{i,j+2}$, $p_{i+2,j}$, and $p_{i+2,j+2}$ with the (trivial) $[0,1]$ constraint, covering all possible pixel values. While this results into sound constraints for $p_{i,j}$, instantiating $[x_l,x_u]$ and $[y_l,y_u]$ with its corner (x,y) furthest from (i,j), yields still a sound but more precise constraint $q_{i,j}$ for $p_{i,j}$. Here, this amounts to $x \leftarrow x_u$ and $y \leftarrow y_u$. App. B presents a detailed explanation of the derivation. The result is

$$q_{i,j} = \left[p'_{i',j'} - \left(\tfrac{2+i-x_u}{2}\tfrac{y_u-j}{2} + \tfrac{x_u-i}{2}\tfrac{2+j-y_u}{2} + \tfrac{x_u-i}{2}\tfrac{y_u-j}{2}\right), p'_{i',j'}\right]\left(\tfrac{2+i-x_u}{2}\tfrac{2+j-y_u}{2}\right)^{-1}.$$

Step 3 In order to be sound, we need to take the union over $q_{i-2,j-2}, q_{i-2,j}, q_{i,j-2}, q_{i,j}$ for each $c_{i',j'}$. To gain precision, we can intersect all of those unions and finally, we can intersect this constraint with the trivial one, $[0,1]$, resulting in the final pixel constraint for pixel $p_{i,j}$:

$$p_{i,j} \in [0,1] \cap \Big(\bigcap_{c_{i',j'} \in C} q_{i-2,j-2}(c_{i',j'}) \sqcup q_{i,j-2}(c_{i',j'}) \sqcup q_{i-2,j}(c_{i',j'}) \sqcup q_{i,j}(c_{i',j'}) \Big), \quad (13)$$

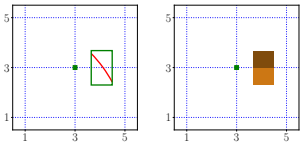
where $\sqcup$ denotes the *join* operation, that is $[a,b] \sqcup [c,d] := [\min(a,c), \max(b,d)]$. If the intersection of $c_{i',j'}$ with the respective (v,w)-interpolation region is empty, we omit $q_{v,w}$ in Eq. (13).

In Section 5.2, we split Γ into Γ_k. It often happens that one of the resulting intervals is empty. Then we know for sure that γ lies in a different Γ_k, speeding up the process substantially.

Refined Inverse The constraints can be refined by following the same steps as for calculating the inverse, but instead of replacing the (unknown) pixel values in Eq. (12) with $[0,1]$, we replace them with the intervals calculated previously. However, replacing $[x_l, x_u] \times [y_l, y_u]$ with the corner furthest away from (i,j) would be unsound. To be sound, one needs to consider all 4 corners of every non-empty intersection $[x_l, x_u] \times [y_l, y_u]$ and join all interval constraints. Similarly, we use the previously calculated constraint for $p_{i,j}$ instead of $[0,1]$ in Eq. (13). This procedure can be repeated to further increase precision. The final result after applying the refinement 10 times is shown in Fig. 2c representing the lower (left) and upper (right) interval bound for all pixels.

6.1 Example

We calculate the constraint for pixel $(3,3)$ of the original image (Fig. 2d), depicted as the green dot in Fig. 3 under the assumption $\gamma \in [23°, 26°]$. We elaborate the constraints that pixel $(5,1)$ of the rotated image (Fig. 2a) yields for pixel $(3,3)$ of the original image.



(a) $c_{5,1}$ (b) Intersections

Figure 3: To improve presentation, the red arc is $3\times$ longer.

Step 1 We illustrate the calculation of the set C for $c_{5,1} := R^{-1}_{[23°,26°]}\left(\begin{smallmatrix}5\\1\end{smallmatrix}\right) = \begin{pmatrix}[4.06,4.21]\\[2.85,3.11]\end{pmatrix}$. The result is depicted as the green box in Fig. 3a enclosing the red arc. The red arc shows the precise set of coordinates where the pixel value $p'_{5,1}$ could have been interpolated from the original image $\boldsymbol{x}$.

Step 2 The only non-empty intersections of $c_{5,1}$ with interpolation regions (blue squares in Fig. 3), cornering $(3,3)$ are the $(3,1)$ and the $(3,3)$-interpolation regions, hence we omit $q_{1,1}$ and $q_{1,3}$. The intersection with the $(3,3)$-interpolation region yields $[x_l, x_u] = [4.06, 4.21]$ and $[y_l, y_u] = [3, 3.11]$ (dark brown rectangle in Fig. 3b), hence at the furthest corner $(x,y) = (4.21, 3.11)$, we get

$$q_{3,3} = [0.73, 2.48] = \left[p'_{5,1} - \left(\tfrac{5-x}{2}\tfrac{y-3}{2} + \tfrac{x-3}{2}\tfrac{5-y}{2} + \tfrac{x-3}{2}\tfrac{y-3}{2}\right), p'_{5,1}\right]\left(\tfrac{5-x}{2}\tfrac{5-y}{2}\right)^{-1},$$

and the intersection with the $(3,1)$-interpolation region yields $[x_l, x_u] = [4.06, 4.21]$ and $[y_l, y_u] = [2.85, 3]$ (light brown rectangle in Fig. 3b), hence at the furthest corner $(x,y) = (4.21, 2.85)$, we get

$$q_{3,1} = [0.72, 2.48] = \left[p'_{5,1} - \left(\tfrac{5-x}{2}\tfrac{3-y}{2} + \tfrac{x-3}{2}\tfrac{3-y}{2} + \tfrac{x-3}{2}\tfrac{y-1}{2}\right), p'_{5,1}\right]\left(\tfrac{5-x}{2}\tfrac{y-1}{2}\right)^{-1}.$$

Step 3 The join $q_{3,1} \sqcup q_{3,3}$ yields $[0.72, 2.48]$. After intersecting this with $[0,1]$ and the constraints from the other $c_{i',j'} \in C$ (as in Eq. (13)), we are left with the final result $p_{3,3} \in [0.73, 1]$.

The final result of the inverse calculation for all pixels is shown in Fig. 2b representing the lower (left) and upper (right) interval bounds for all pixels. The iterative refinement is shown in Fig. 2c.

7 Experimental Evaluation

We now present our extensive evaluation of the different defenses discussed so far.

7.1 Instantiation in Practice

In Section 4 we showed how to certify robustness of g_E to T^I_γ, obtained from Eq. (1) with $f := h_E \circ I$ and $\psi_\beta := T_\beta$. Since in practice $h_E \circ I$ and T_β cannot be evaluated as I and T_β are not available independently, in order to evaluate g_E in practice, we need to re-write it as follows:

$$\begin{aligned} g_E(\boldsymbol{x}) &= \arg\max_c \mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})}\left((h_E \circ I) \circ T_\beta(\boldsymbol{x}) = c\right) \\ &= \arg\max_c \mathbb{P}_{\beta \sim \mathcal{N}(0,\sigma^2\mathbb{1})}\left(h_E \circ (I \circ T_\beta)(\boldsymbol{x}) = c\right) =: g(\boldsymbol{x}), \end{aligned}$$

which is an instantiation of Eq. (1) with $f := h_E$ and $\psi_\beta := T^I_\beta$, both of which are available.

Further, as the probability in Eq. (1) cannot be computed exactly, in practice we use the approximation introduced in Cohen et al. [7]: by taking n samples around a given $\boldsymbol{x}$ with standard deviation σ, we can obtain $g(\boldsymbol{x})$ and the corresponding robustness radius r with confidence $1-\alpha$. Here, n can be too small to make a statement with confidence $1-\alpha$, in which case the classifier abstains. Further, we let $\sigma_\gamma, \alpha_\gamma, n_\gamma, r_\gamma$ and $\sigma_\delta, \alpha_\delta, n_\delta, r_\delta$ denote the parameters and radius required to use Theorem 3.2 and Theorem 3.1 in practice, respectively. Statistically sound certification as in Cohen et al. [7] requires to fist take n_0 many samples fist and guessing the correct class on them. In our case we apply both T^I_β and additive noise ϵ to these n_0 samples.

7.2 Setup

All experiments were performed on a machine with 2 GeForce RTX 2080 Tis and an Intel(R) Core(TM) i9-9900K CPU. As base classifiers b we utilize neural networks in PyTorch [29], using `robustness` [30] and Salman et al. [8] for training. Further, we implemented the interval analysis (cf. §5 and §6) of the interpolation error and inverse computation in C++/CUDA.

We consider rotations R^I_γ by γ degrees and translations Δ^I_γ by $\gamma \in \mathbb{R}^2$ with bilinear interpolation I. Here, we allow the adversary to choose $\gamma \in \Gamma$. For a scalar $\Gamma_\pm \in \mathbb{R}^{\geq 0}$, we permit $\Gamma := [-\Gamma_\pm, \Gamma_\pm]$ for rotations and $\Gamma := [-\Gamma_\pm, \Gamma_\pm]^2$ for translations. All estimates of E include interpolation errors as well as 8-bit representation ("rounding") errors. When we estimate ρ_E with confidence α_E.

We evaluate on ImageNet [31], Restricted ImageNet (RImageNet)[32], a subset of ImageNet with 10 classes, CIFAR-10 [33], and MNIST [34]. For the base classifier, in Section 7.3 we use standard models without any additional training, while in the other sections we use models trained with data augmentation (transformations, ℓ^2-noise) using [8].

In §7.4 and §7.5, we apply a circular or rectangular vignette for rotation and translation respectively, to reduce error estimates in areas of the image where information is lost. We also apply a Gaussian blur prior to classification to further reduce the high-frequency components of the interpolation error. App. D contains further details on prepossessing, model training and parameters. Note that pre-processing does not impact the theoretical guarantees as long as it is consistently applied. We provide an ablation study regarding vignetting and Gaussian blur in App. F. Additional experiments, including other interpolation methods or audio classification are provided in App. E, highlighting the generality of our methods. Throughout the section all individual certificates hold with overall confidence $1-\alpha$ for $\alpha = 0.01$.

7.3 BASESPT

We can quickly obtain a well-motivated but empirical defense by instantiating Theorem 3.2 with $\psi_\beta := T^I_\beta$ and ignoring both the interpolation error Eq. (3) and the construction in Section 4. Table 2 shows results on an undefended classifier b and the BASESPT smoothed version g. Here *Acc.* is obtained over the whole dataset. To evaluate *adv. Acc.* we use the `worst-of-k` proposed by Engstrom et al. [2], which returns the γ yielding the highest cross-entropy loss out of k randomly sampled $\gamma \sim \mathcal{U}(\Gamma)$. We apply `worst-of-k` to 1000 images and produce 3 attacked images each, resulting 3000 samples on which we then evaluate b and g. For g, the average inference time per image t is generally fast, where most time is spent on sampling transformations. The actual inference, invoking b on the samples, is not slowed down as all samples fit into a single batch. In this section we use $n_\gamma = 1000, \sigma_\gamma = \Gamma_\pm$. and $\alpha_\gamma = 0.01$.

Table 2: Evaluation of BASESPT. We obtain Acc for b on the test set and evaluate adv. Acc. on 3000 images obtained by the `worst-of-100` attack. t denotes the average run time of g.

			Acc.	adv. Acc.		
Dataset	T^I	$\Gamma_\pm$	b	b	g	t [s]
MNIST	R^I	30°	0.99	0.73	0.99	0.97
CIFAR-10	R^I	30°	0.91	0.26	0.85	0.95
ImageNet	R^I	30°	0.76	0.56	0.76	5.43
MNIST	Δ^I	4	0.99	0.03	0.53	0.86
CIFAR-10	Δ^I	4	0.91	0.44	0.79	0.95
ImageNet	Δ^I	20	0.76	0.65	0.75	6.70

We do not obtain certificates here as the assumptions of Theorem 3.2 are violated. However, we investigate in App. E if the certification radius holds practically.

Table 3: Evaluation of DISTSPT$^{\mathcal{D}}$ for $T^I := R^I$. We show the test set accuracy of b, certified accuracy of g at different radii r_γ, along with the average run time t. $^\#$ denotes values obtained by sampling. Each certificate hold with overall confidence 0.99.

				g cert. acc at r_γ					
Dataset	E	q_E	b acc.	$0°$	$10°$	$20°$	$30°$	t [s]	n_γ
MNIST	0.45	0.99	0.98	0.89	0.88	0.87	0.85	21.56	200
CIFAR-10	0.55	0.99	0.56	0.31	0.28	0.25	0.19	89.75	50
CIFAR-10	0.55	0.99	0.56	0.32	0.30	0.28	0.25	351.47	200
RImageNet	$1.20^\#$	0.97	0.78	0.74	0.72	0.68	0.61	100.73	50
RImageNet	$1.35^\#$	0.99	0.78	0.64	0.62	0.56	0.50	100.13	50
ImageNet	$0.95^\#$	0.75	0.38	0.30	0.24	0.18	0.12	100.21	50
ImageNet	$1.20^\#$	0.97	0.38	0.23	0.19	0.13	0.09	100.73	50
ImageNet	$1.35^\#$	0.99	0.38	0.16	0.12	0.08	0.06	100.44	50

7.4 DISTSPT [1]

Here we evaluate DISTSPT$^{\mathcal{D}}$ and DISTSPT$^{\boldsymbol{x}}$ and compare with related approaches.

DISTSPT$^{\mathcal{D}}$ First we consider DISTSPT$^{\mathcal{D}}$, where E is obtained over the training set and expected to hold in distribution as discussed in Section 5.1. This allows to run both prediction, where the robust accuracy shown here can be expected to hold in distribution, as well as certification (e.g. to show $g(\boldsymbol{x}) = g(\boldsymbol{x}')$) at inference time.

Table 3 shows our results for DISTSPT$^{\mathcal{D}}$ with rotations. We restrict the attacker model to $\Gamma_\pm = 90°$ for MNIST and $\Gamma_\pm = 30°$ for other datasets.

To obtain E, we first sample the interpolation error in Eq. (7) (using 1000 images). Subsequently, we choose E slightly larger than this error. With E fixed, we test for $\rho_E = 0.001$ and expect q_E to be close to 1 for all datasets. Table 3 shows q_E obtained with confidence $1 - \alpha_E = 0.999$ by using 1000 samples for $\boldsymbol{x}$ and 8000 for β (and correction for possible test errors over β). For small images, these bounds can be computed quickly. However, for large images (ImageNet), the optimization over γ for many images is computationally expensive. Thus, for ImageNet we replace the $\max$ in Eq. (6) with the maximum over 10 samples $\gamma \in \mathcal{U}(\Gamma)$ (indicated by $^\#$ in Table 3). This formally restricts the certificate to only hold against random attacks (such as `worst-of-10`). However, if sufficient computational resources are available, the $\max$ method can still be applied (we empirically find the method to obtain similar values). On (R)ImageNet (variable image size) we resize all images so that the short side is 512 pixel prior to applying transformations. As RImageNet is a subset of ImageNet, we use E obtained on the later.

Now, we evaluate the accuracy of b and g. For b we use the whole test set, while for g we use 1000 samples. In addition to the results in Table 3, at $r_\gamma = 50$, the MNIST g in this configuration still achieves 0.75 certified accuracy. Comparing the results on ImageNet and RImageNet shows that the limiting factor for our method is the robustness of the base classifier, not the size of the image.

We use $\sigma_\gamma = 30$ for all datasets and $\sigma_\delta = 0.25, n_\delta = 10000, n_0 = 10000$ for MNIST, $\sigma_\delta = 0.3, n_\delta = 15000, n_0 = 10000$ for CIFAR-10 and $\sigma_\delta = 0.5, n_\delta = 2500, n_0 = 200$ for (R)ImageNet in all but the $E = 1.35$ setting where we use $\sigma_\delta = 0.55$. We use $\alpha_\gamma = 0.005 - \alpha_E$ and $\alpha_\delta = \frac{0.005}{n_\gamma}$, such that the overall confidence for each certificate is 0.99. We expect these results to hold in distribution for at least q_E percent of data points.

To showcase that DISTSPT$^{\mathcal{D}}$ can be applied as an online defense to obtain individual certificates $g(\boldsymbol{x}) = g(\boldsymbol{x}')$, we also evaluate on attacked images. Using the same settings as above we can certify for 91 out of 100 MNIST images, adversarially rotated with $\Gamma_\pm = 30$, that they are classified the same as the original, while also being correct.

[1] The results in §7.4 and §7.5 differ from those in the version published at NeurIPS'20 due to an implementation bug we since fixed. Further, we improved readability and provide additional results enable better comparison. A version of Table 3 in the original layout can be found in App. E.3.

Table 4: Evaluation of DISTSPT$^{\boldsymbol{x}}$ for $T^I := R^I$. We show the test set certified accuracy of g at different radii r_γ, along with the average E estimated, the average time t_E to estimate E and average time t_{RS} to apply randomized smoothing. $^\#$ denotes values obtained by sampling. * we use a server with 128 threads on an AMD EPYC 7601 processor, on the same system as the other results these take 766 s. Each certificate hold with overall confidence 0.99.

			g cert. acc at r_γ								
Dataset	$\Gamma_\pm$	σ_γ	0°	10°	20°	30°	50°	avg. E	t_E [s]	t_{RS} [s]	n_γ
MNIST	50°	30	0.93	0.92	0.91	0.90	0.82	0.34	53.33	20.56	200
CIFAR-10	30°	40	0.35	0.30	0.27	0.22	-	0.34	81.83	91.72	50
CIFAR-10	10°	10	0.43	0.37	-	-	-	0.34	51.12	92.83	50
ImageNet	30°	30	0.31	0.25	0.17	0.11	-	$0.86^\#$	73.58^*	100.47	50
ImageNet	30°	30	0.32	0.29	0.22	0.16	-	$0.86^\#$	73.58^*	396.50	200

Finally, we evaluate translations on MNIST ($E = 0.65$, $\rho_E = 0.99$, $\Gamma_\pm = 2$) and achieve certified accuracy 64% and 49% at r_γ of 0 and $\sqrt{2}$, respectively. We use $\sigma_\gamma = 1.5, \sigma_\delta = 0.25, n_\gamma = 200$ and the other parameters as for rotation.

DISTSPT$^{\boldsymbol{x}}$ We now evaluate DISTSPT$^{\boldsymbol{x}}$. Here, we certify a classifier g (for a fixed b, σ_γ, σ_ϵ) on the test set. At inference time we just predict new samples and expect the obtained robustness certificates to hold in distribution. We show the certification results in Table 4.

To this end, it is sufficient to obtain E, ρ_E (as in Eq. (6)) for each individual image $\boldsymbol{x}$ rather than for the whole data distribution. Naturally, these individual bounds are much lower than E obtained over the data distribution, allowing for better accuracy.

On MNIST and CIFAR-10, for each image $\boldsymbol{x}$ we use 100 samples of β (optimizing over γ) to guess E as 1.1 times the largest observed error. Then, we use another 400 samples of β to test for ρ_E with $\alpha_E = 0.001$. On ImageNet we use the same procedure but chose E as the largest observed error over 240 samples of β plus 0.03. As for DISTSPT$^{\mathcal{D}}$, we use sampling to approximate the maximization. When optimizing for E we stop either when the highest bound for any β is 0.3 or after a timeout of 2 minutes. Varying these parameters may allow for an even lower E at the cost of more run time.

For translation ($\sigma_\gamma = 0.25, n_\gamma = 200, \Gamma_\pm = 2, \sigma_\gamma = 2.5$) we use the same setup but optimize until the maximal error is lower than 0.55 or a timeout of 2 minutes is reached. We choose E as the maximal error over the first 100 images plus 0.02. With an average E of 0.56, we obtain a certified accuracy of 0.89, 0.86, 0.85 and 0.82 at radii r_γ of 0, 1, $\sqrt{2}$, and 2, respectively. The average analysis took 62.22s and certification 19.33s. We note that in general DISTSPT$^{\mathcal{D}}$ results are a lower bound for the results of DISTSPT$^{\boldsymbol{x}}$. In theory, DISTSPT$^{\mathcal{D}}$ can perform better if ρ_E is lower (e.g. when more samples are used). However, in practice this is offset by the tighter error bound. We see that the average E is much lower than the upper bound used in DISTSPT$^{\mathcal{D}}$, allowing better results.

Unless stated differently in Table 4, we use the same parameters as for DISTSPT$^{\mathcal{D}}$, with the exception of σ_δ where we use $0.15, 0.20, 0.18, 0.50, 0.40$ for the order as in Table 4. As before, all certificates hold with confidence 0.99 as $\alpha_\gamma = 0.005 - \alpha_E$ and $\alpha_\delta = \frac{0.005}{n_\gamma}$.

Comparison to other work Related approaches, Balunovic et al. [11], Li et al. [13], provide distribution certificates, e.g., they certify images on the test set and the obtained certified accuracy can then be expected to hold for new, potentially perturbed images. However, it is not possible to certify novel inputs – this is the same setting as with DISTSPT$^{\boldsymbol{x}}$. Balunovic et al. [11] certifies model accuracy on the test set and thus provides a distributional bound. On MNIST they report 87.01% of certified accuracy for rotations with $\pm 30^\circ$ ($35s$ per image), which with further refinement (at cost of run time) can be increased to 97%, and for translations with ± 2 pixels 76.30% (263s per image). On CIFAR-10 they certify rotation up to 10° for 62.51%, but unlike our work, the method does not scale to larger image sizes and models, such as ResNet-50 on ImageNet. We provide further comparison with Balunovic et al. [11] in App. F. For a comparison with [13, 14], we refer the reader to that work. Pei et al. [9] certify $\pm 2^\circ$ in 714 s per image on ImageNet. However, in contrast to us they focus on nearest-neighbor interpolation, which can be enumerated.

7.5 INDIVSPT [1]

Finally, we evaluate INDIVSPT, where we compute E on the given input. The bound computed by interval analysis is always sound, but may be quite large due to the loss of precision inherent in interval analysis. We show results for MNIST and discuss challenges on larger datasets in App. C. To this end, we attack images as in Section 7.3, and subsequently apply INDIVSPT. We use the `worst-of-100` attack on a base classifier b to obtain a set of attacked images. To these images we then apply INDIVSPT. For rotations ($\Gamma_{\pm} = 10, \sigma_\gamma = 30, \sigma_\delta = 0.3, n_\gamma = 200, n_\delta = 10000, n_0 = 10000$, 3 attacks per image, 1000 images) we fix $E = 0.45$ and use 500 samples of β to obtain the correct ρ_E (Eq. (9)) with $\alpha_E = 0.001$. g was correct on 82% of attacked images. For 81% we could certify that the attacked image that classifies the same as the original. The analysis of E took on average 0.26 s and the randomized smoothing 25.03 s. For translation we use the same setup ($\Gamma_{\pm} = 1, \sigma_\gamma = 1.5, \sigma_\delta = 0.3, n_\gamma = 200$, 3 attacks per image, 100 images) also starting with $E = 0.45$. g classified 75% of attacked images correctly and could certify $r_\gamma \geq 1$ and thereby $g(\boldsymbol{x}) = g(\boldsymbol{x}')$ on 59% while on average taking 14.14 s for analysis and 19.89 s for smoothing per image. The reason for the higher run time is that compared to rotation fewer possible inverses can be discarded. We use 10 refinement steps for both rotations and translations.

7.6 Limitations & Generalization

While we showcased translation and rotation, our approach is not limited to these transformations or to specific interpolation methods. BASESPT and DISTSPT can be directly adapted to other transformations, interpolation schemes or domains such as audio (see App. E). INDIVSPT can also be adapted but requires additional care. Generally, Theorem 3.2 can be applied to all parameterized data transformations that are additive in the parameter space. If this holds up to a small error, as discussed here, DISTSPT and INDIVSPT can be applied. While many data transformations, e.g., image scaling are additive in their parameter space, their compositions are often not (e.g., rotation and translation). As we are most limited by the ℓ^p-robustness of b, any gains in ℓ^p certification will directly improve our method. Further, INDIVSPT can incur a large loss of precision in the inverse computation. Improving this directly increases the applicability of the method.

8 Conclusion

We presented the first generalization of randomized smoothing to image transformations, a challenging task as image transformations do not compose. Based on this generalization, we presented several certified defenses allowing for both distributional and individual guarantees (relying on statistical error bounds or on efficient inverse computation). Our exploration highlights interesting trade-offs between certification guarantees and tightness of the resulting bounds. Finally, our extensive evaluation demonstrates the methods can handle realistic datasets and models.

9 Broader Impact

In general, methods from artificial intelligence can be applied in beneficial and malicious ways. While this poses a threat in itself, verification techniques provide formal guarantees for the robustness of the model, independently of the intended use case. Certification techniques could therefore distinguish a potentially unstable model from a stable one in safety critical settings, e.g., autonomous driving. However, especially for regulators, it is of utter importance to understand the certified properties of different certification methods precisely, as to avoid legal model deployment in safety critical applications based on misconceptions.

Acknowledgments and Disclosure of Funding

We thank the authors of [13], in particular Maurice Weber and Linyi Li, for insightful discussion and pointing out an implementation bug. Further, we thank all reviewers for their helpful comments and feedback.

We do not have any additional funding or compensation to disclose.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

[2] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *CoRR*, abs/1712.02779, 2017. URL http://arxiv.org/abs/1712.02779.

[3] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 3–18. IEEE Computer Society, 2018. doi: 10.1109/SP.2018.00058. URL https://doi.org/10.1109/SP.2018.00058.

[4] Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable abstract interpretation for provably robust neural networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3575–3583. PMLR, 2018. URL http://proceedings.mlr.press/v80/mirman18b.html.

[5] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5283–5292. PMLR, 2018. URL http://proceedings.mlr.press/v80/wong18a.html.

[6] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 10900–10910, 2018. URL http://papers.nips.cc/paper/8285-semidefinite-relaxations-for-certifying-robustness-to-adversarial-examples.

[7] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 2019. URL http://proceedings.mlr.press/v97/cohen19c.html.

[8] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya P. Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *CoRR*, abs/1906.04584, 2019. URL http://arxiv.org/abs/1906.04584.

[9] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Towards practical verification of machine learning: The case of computer vision systems. *CoRR*, abs/1712.01785, 2017. URL http://arxiv.org/abs/1712.01785.

[10] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. An abstract domain for certifying neural networks. *PACMPL*, 3(POPL):41:1–41:30, 2019. doi: 10.1145/3290354. URL https://doi.org/10.1145/3290354.

[11] Mislav Balunovic, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin T. Vechev. Certifying geometric robustness of neural networks. In *NeurIPS*, pages 15287–15297, 2019.

[12] Jeet Mohapatra, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Towards verifying robustness of neural networks against semantic perturbations. *CoRR*, abs/1912.09533, 2019.

[13] Linyi Li, Maurice Weber, Xiaojun Xu, Luka Rimanic, Tao Xie, Ce Zhang, and Bo Li. Provable robust learning based on transformation-specific smoothing. *CoRR*, abs/2002.12398, 2020.

[14] Linyi Li, Maurice Weber, Xiaojun Xu, Luka Rimanic, Bhavya Kailkhura, Tao Xie, Ce Zhang, and Bo Li.

[15] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, pages 387–402. Springer, 2013.

[16] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, December 4-8, 2017*, pages 278–287. ACM, 2017. doi: 10.1145/3134600.3134606. URL https://doi.org/10.1145/3134600.3134606.

[17] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 381–397. Springer, 2018. doi: 10.1007/978-3-030-01234-2_23. URL https://doi.org/10.1007/978-3-030-01234-2_23.

[18] Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*, volume 10482 of *Lecture Notes in Computer Science*, pages 269–286. Springer, 2017. doi: 10.1007/978-3-319-68167-2_19. URL https://doi.org/10.1007/978-3-319-68167-2_19.

[19] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017. doi: 10.1007/978-3-319-63387-9_5. URL https://doi.org/10.1007/978-3-319-63387-9_5.

[20] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and Pawan Kumar Mudigonda. A unified view of piecewise linear neural network verification. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 4795–4804, 2018. URL http://papers.nips.cc/paper/7728-a-unified-view-of-piecewise-linear-neural-network-verification.

[21] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *NeurIPS*, pages 6369–6379, 2018.

[22] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. Towards fast computation of certified robustness for relu networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5273–5282. PMLR, 2018. URL http://proceedings.mlr.press/v80/weng18a.html.

[23] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *NeurIPS*, pages 9832–9842, 2019.

[24] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672, 2018.

[25] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. *CoRR*, abs/1809.03113, 2018. URL http://arxiv.org/abs/1809.03113.

[26] Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqing Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. Macer: Attack-free and scalable robust training via maximizing certified radius. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJx1Na4Fwr.

[27] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Geometric robustness of deep networks: Analysis and improvement. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4441–4449. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00467. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Kanbak_Geometric_Robustness_of_CVPR_2018_paper.html.

[28] Hend Dawood. *Theories of interval arithmetic: mathematical foundations and applications*. LAP Lambert Academic Publishing, 2011.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[30] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[32] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SyxAb30cY7.

[33] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[34] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 396–404. Morgan Kaufmann, 1989. URL http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.

[35] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

[36] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014. URL http://dl.acm.org/citation.cfm?id=2670313.

[37] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[38] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018. URL http://arxiv.org/abs/1804.03209.

[39] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *IEEE Transactions on Acoustics, Speech, and Signal Processing*. IEEE, 1980.

Supplementary Material for Certified Defense to Image Transformations via Randomized Smoothing

A Proof of Theorem 3.2

We now proceed to proof Theorem 3.2. We achieve this by first proofing an auxiliary Theorem and Lemma, and then instantiating as a special case Theorem 3.2 of these slightly more general results.

Theorem A.1. *Let $\boldsymbol{x} \in \mathbb{R}^n$, $f : \mathbb{R}^m \to \mathcal{Y}$ be a classifier, $\psi_\beta : \mathbb{R}^n \to \mathbb{R}^m$ a composable transformation for $\beta \sim \mathcal{N}(0, \Sigma)$ with a symmetric, positive-definite covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$. If*

$$\mathbb{P}_\beta(f \circ \psi_\beta(x) = c_A) = p_A \geq \underline{p_A} \geq \overline{p_B} \geq p_B = \max_{c_B \neq c_A} \mathbb{P}_\beta(f \circ \psi_\beta(x) = c_B),$$

then $g \circ \psi_\gamma(\boldsymbol{x}) = c_A$ for all γ satisfying

$$\sqrt{\gamma^T \Sigma^{-1} \gamma} < \tfrac{1}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) =: r_\gamma.$$

Proof. The assumption is

$$\mathbb{P}\left((f \circ \psi_\beta)(\boldsymbol{x}) = c_A\right) = p_A \geq \underline{p_A} \geq \overline{p_B} \geq p_B = \mathbb{P}\left((f \circ \psi_\beta)(\boldsymbol{x}) = c_B\right).$$

By the definition of g we need to show that

$$\mathbb{P}\left((f \circ \psi_{\beta+\gamma})(\boldsymbol{x}) = c_A\right) \geq \mathbb{P}\left((f \circ \psi_{\beta+\gamma})(\boldsymbol{x}) = c_B\right).$$

We define the set $A := \{\boldsymbol{z} \mid \gamma^T \Sigma^{-1} \boldsymbol{z} \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\Phi(\underline{p_A})\}$. We claim that for $\beta \sim \mathcal{N}(0, \Sigma)$, we have

$$\mathbb{P}(\beta \in A) = \underline{p_A} \tag{14}$$
$$\mathbb{P}(f \circ \psi_{\beta+\gamma}(x) = c_A) \geq \mathbb{P}(\beta + \gamma \in A). \tag{15}$$

First, we show that Eq. (14) holds.

$$\begin{aligned}
\mathbb{P}(\beta \in A) &= \mathbb{P}(\gamma^T \Sigma^{-1} \beta \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\Phi(\underline{p_A})) \\
&= \mathbb{P}(\gamma^T \Sigma^{-1} \mathcal{N}(0, \Sigma) \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\Phi(\underline{p_A})) \\
&= \mathbb{P}(\gamma^T \sqrt{\Sigma^{-1}} \mathcal{N}(0, \mathbb{1}) \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\Phi(\underline{p_A})) \\
&= \mathbb{P}(\mathcal{N}(0, \gamma^T \Sigma^{-1} \gamma) \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\Phi(\underline{p_A})) \\
&= \mathbb{P}(\sqrt{\gamma^T \Sigma^{-1} \gamma}\mathcal{N}(0, 1) \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\Phi(\underline{p_A})) \\
&= \mathbb{P}(\mathcal{N}(0, 1) \leq \Phi(\underline{p_A})) \\
&= \Phi(\Phi^{-1}(\underline{p_A})) \\
&= \underline{p_A}
\end{aligned}$$

Thus Eq. (14) holds. Next we show that Eq. (15) holds. For a random variable $v \sim \mathcal{N}(\mu_v, \Sigma_v)$ we write $p_v(z)$ for the evaluation of the Gaussian cdf at point z.

$$
\begin{aligned}
&\mathbb{P}(f \circ \psi_{\beta+\gamma}(x) = c_A) - \mathbb{P}(\beta+\gamma \in A) \\
&= \int_{\mathbb{R}^d} [f \circ \psi_{\boldsymbol{z}} = c_A]\, p_{\beta+\gamma}(z)dz - \int_A p_{\beta+\gamma}(z)dz \\
&= \int_{\mathbb{R}^d \setminus A} [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta+\gamma}(z)dz + \int_A [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta+\gamma}(z)dz - \int_A p_{\beta+\gamma}(z)dz \\
&= \int_{\mathbb{R}^d \setminus A} [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta+\gamma}(z)dz + \int_A [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta+\gamma}(z)dz \\
&\quad - \left(\int_A [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta+\gamma}(z)dz + \int_A [f \circ \psi_{\boldsymbol{z}}(x) \neq c_A]\, p_{\beta+\gamma}(z)dz \right) \\
&= \int_{\mathbb{R}^d \setminus A} [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta+\gamma}(z)dz - \int_A [f \circ \psi_{\boldsymbol{z}}(x) \neq c_A]\, p_{\beta+\gamma}(z)dz \\
&\overset{Lemma\ 1}{\geq} t \left(\int_{\mathbb{R}^d \setminus A} [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta}(z)dz - \int_A [f \circ \psi_{\boldsymbol{z}}(x) \neq c_A]\, p_{\beta}(z)dz \right) \\
&= t \left(\int_{\mathbb{R}^d} [f \circ \psi_{\boldsymbol{z}}(x) = c_A]\, p_{\beta}(z)dz - \int_A p_{\beta}(z)dz \right) \\
&\overset{Eq.\ (14)}{\geq} 0.
\end{aligned}
$$

Thus also Eq. (15) holds.

Next, we claim that for $B := \{z \mid \gamma^T \Sigma^{-1} \boldsymbol{z} \geq \sqrt{\gamma^T \Sigma^{-1} \gamma}\, \Phi^{-1}(1 - \overline{p_B})\}$ holds that

$$\mathbb{P}(f \circ \psi_{\beta}(x) = c_B) \leq \mathbb{P}(\beta \in B) \tag{16}$$

$$\mathbb{P}(f \circ \psi_{\beta+\gamma}(x) = c_B) \leq \mathbb{P}(\beta+\gamma \in B) \tag{17}$$

The proofs for Eq. (16) and Eq. (17) are analogous to the proofs for Eq. (14) and Eq. (15).

Now we derive the conditions that lead to $\mathbb{P}(\beta+\gamma \in A) > \mathbb{P}(\beta+\gamma \in B)$:

$$
\begin{aligned}
\mathbb{P}(\beta+\gamma \in A) &= \mathbb{P}\left(\gamma^T \Sigma^{-1}(\beta+\gamma) \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\, \Phi^{-1}(\underline{p_A}) \right) \\
&= \mathbb{P}\left(\gamma^T \Sigma^{-1}(\Sigma^{\frac{1}{2}} \mathcal{N}(0, \mathbb{1}) + \gamma) \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\, \Phi^{-1}(\underline{p_A}) \right) \\
&= \mathbb{P}\left(\gamma^T \sqrt{\Sigma^{-1}} \mathcal{N}(0, \mathbb{1}) + \gamma^T \Sigma^{-1} \gamma \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\, \Phi^{-1}(\underline{p_A}) \right) \\
&= \mathbb{P}\left(\sqrt{\gamma^T \Sigma^{-1} \gamma} \mathcal{N}(0, \mathbb{1}) + \gamma^T \Sigma^{-1} \gamma \leq \sqrt{\gamma^T \Sigma^{-1} \gamma}\, \Phi^{-1}(\underline{p_A}) \right) \\
&= \mathbb{P}\left(\mathcal{N}(0, \mathbb{1}) + \sqrt{\gamma^T \Sigma^{-1} \gamma} \leq \Phi^{-1}(\underline{p_A}) \right) \\
&= \mathbb{P}\left(\mathcal{N}(0, \mathbb{1}) \leq \Phi^{-1}(\underline{p_A}) - \sqrt{\gamma^T \Sigma^{-1} \gamma} \right) \\
&= \Phi(\Phi^{-1}(\underline{p_A}) - \sqrt{\gamma^T \Sigma^{-1} \gamma})
\end{aligned}
$$

Similarly, we have

$$
\begin{aligned}
\mathbb{P}(\beta+\gamma \in B) &= \mathbb{P}\left(\mathcal{N}(0, \mathbb{1}) \geq \Phi^{-1}(1 - \overline{p_B}) - \sqrt{\gamma^T \Sigma^{-1} \gamma} \right) \\
&= \Phi(\sqrt{\gamma^T \Sigma^{-1} \gamma} - \Phi^{-1}(1 - \overline{p_B}))
\end{aligned}
$$

Thus, we get

$$
\begin{aligned}
& & \mathbb{P}(\beta+\gamma \in A) &> \mathbb{P}(\beta+\gamma \in B) \\
\Leftrightarrow & & \Phi(\Phi^{-1}(\underline{p_A}) - \sqrt{\gamma^T \Sigma^{-1} \gamma}) &> \Phi(\sqrt{\gamma^T \Sigma^{-1} \gamma} - \Phi^{-1}(1 - \overline{p_B})) \\
\Leftrightarrow & & \Phi^{-1}(\underline{p_A}) - \sqrt{\gamma^T \Sigma^{-1} \gamma} &> \sqrt{\gamma^T \Sigma^{-1} \gamma} - \Phi^{-1}(1 - \overline{p_B}) \\
\Leftrightarrow & & \Phi^{-1}(\underline{p_A}) + \Phi^{-1}(1 - \overline{p_B}) &> 2\sqrt{\gamma^T \Sigma^{-1} \gamma} \\
\Leftrightarrow & & \tfrac{1}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) &> \sqrt{\gamma^T \Sigma^{-1} \gamma}.
\end{aligned}
$$

□

Next ,we show the lemma used in the proof.

Lemma 1. *There exists $t > 0$ such that $p_{\beta+\gamma}(z) \leq p_\beta(z) \cdot t$ for all $z \in A$. And further $p_{\beta+\gamma}(z) > p_\beta(z) \cdot t$ for all $z \in \mathbb{R}^d \setminus A$.*

Proof.

$$\begin{aligned}\frac{p_{\beta+\gamma}(z)}{p_\beta(z)} &= \exp\left(-\tfrac{1}{2}(\boldsymbol{z}-\gamma)^T\Sigma^{-1}(\boldsymbol{z}-\gamma) + \tfrac{1}{2}\boldsymbol{z}^T\Sigma^{-1}\boldsymbol{z}\right)\\ &= \exp\left(-\tfrac{1}{2}\boldsymbol{z}^T\Sigma^{-1}\boldsymbol{z} + \boldsymbol{z}^T\Sigma^{-1}\gamma - \tfrac{1}{2}\gamma^T\Sigma^{-1}\gamma + \tfrac{1}{2}\boldsymbol{z}^T\Sigma^{-1}\boldsymbol{z}\right)\\ &= \exp\left(\boldsymbol{z}^T\Sigma^{-1}\gamma - \tfrac{1}{2}\gamma^T\Sigma^{-1}\gamma\right)\end{aligned}$$

What is the lowest t if it exists such that $\frac{p_{\beta+\gamma}(z)}{p_\beta(z)} \leq t$?

$$\begin{aligned}&\frac{p_{\beta+\gamma}(z)}{p_\beta(z)} && \leq t\\ \Leftrightarrow\quad &\exp\left(\boldsymbol{z}^T\Sigma^{-1}\gamma - \tfrac{1}{2}\gamma^T\Sigma^{-1}\gamma\right) && \leq t\\ \Leftrightarrow\quad &\boldsymbol{z}^T\Sigma^{-1}\gamma - \tfrac{1}{2}\gamma^T\Sigma^{-1}\gamma && \leq \log t\\ \Leftrightarrow\quad &\boldsymbol{z}^T\Sigma^{-1}\gamma && \leq \log t + \tfrac{1}{2}\gamma^T\Sigma^{-1}\gamma\end{aligned}$$

Because $z \in A$, we know that

$$\boldsymbol{z}^T\Sigma^{-1}\gamma \leq \sqrt{\gamma^T\Sigma^{-1}\gamma}\Phi^{-1}(\underline{p_A}).$$

Does there exist a t such that both upper bound coincide? Yes, namely

$$t = \exp\left(\sqrt{\gamma^T\Sigma^{-1}\gamma}\Phi^{-1}(\underline{p_A}) - \tfrac{1}{2}\gamma^T\Sigma^{-1}\gamma\right).$$

The case $p_{\beta+\gamma}(z) > p_\beta(z) \cdot t$ is analogous. □

Lemma 2. *If we evaluate on a proxy classifier f' instead of f, behaving with probability $(1-\rho)$ the same as f and with probability ρ differently than f and if*

$$\mathbb{P}_{\beta,f'}(f' \circ \psi_\beta(x) = c_A) \geq \underline{p'_A} \geq \overline{p'_B} \geq \max_{c_B \neq c_A} \mathbb{P}_{\beta,f'}(f' \circ \psi_\beta(x) = c_B),$$

then $g \circ \psi_\gamma(\boldsymbol{x}) = c_A$ for all γ satisfying

$$\|\gamma\|_2 < \frac{\sigma}{2}(\Phi^{-1}(\underline{p'_A} - \rho) - \Phi^{-1}(\overline{p_B} + \rho)).$$

Proof. By applying the union bound we can relate the output probability p of f for a class c with the output probability of f' and p':

$$\begin{aligned}p' &:= \mathbb{P}_{\beta,f'}(f' \circ \psi_\beta(x) = c)\\ &= \mathbb{P}_{\beta,f'}\left((f \circ \psi_\beta(x) = c) \vee (f' \text{ error})\right)\\ &\leq \mathbb{P}_\beta(f \circ \psi_\beta(x) = c) + \mathbb{P}_{f'}(f' \text{ error})\\ &= p + \rho\end{aligned}$$

Thus we can obtain new bounds $\underline{p_A} \geq \underline{p'_A} - \rho$ and $\overline{p_B} \leq \overline{p'_B} + \rho$ from $\underline{p'_A}$ and $\overline{p'_B}$ measured on f'. Plugging these bounds in Theorem 3.2 yields the result. □

We now show Theorem 3.2 (restarted below): Setting $\Sigma = \sigma^2\mathbb{1}$ in Theorem A.1 directly recovers Theorem 3.2 up to the last sentence, which in turn is a direct consequence of Lemma 2.

Theorem (Theorem 3.2 restated)**.** *Let $\boldsymbol{x} \in \mathbb{R}^m$, $f : \mathbb{R}^m \to \mathcal{Y}$ be a classifier and $\psi_\beta : \mathbb{R}^m \to \mathbb{R}^m$ be a composable transformation as above. If*

$$\mathbb{P}_\beta(f \circ \psi_\beta(\boldsymbol{x}) = c_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c_B \neq c_A} \mathbb{P}_\beta(f \circ \psi_\beta(\boldsymbol{x}) = c_B),$$

then $g \circ \psi_\gamma(\boldsymbol{x}) = c_A$ for all γ satisfying $\|\gamma\|_2 \leq \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) =: r_\gamma$. Further, if g is evaluated on a proxy classifier f' that behaves like f with probability $1-\rho$ and else returns an arbitrary answer, then $r_\gamma := \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A} - \rho) - \Phi^{-1}(\overline{p_B} + \rho))$.

B Inverse and Refinement

B.1 Details for Step 2

In this section, we elaborate on the details of Step 2 in Section 6. We consider the intersection of $c_{i',j'}$ with the (i,j)-interpolation region, $[x_l, x_u] \times [y_l, y_u] := c_{i',j'} \cap [i, i+2] \times [j, j+2]$. This yields,

$$p'_{i',j'} \in I([x_l, x_u], [y_l, y_u]) = p_{i,j} \tfrac{2+i-[x_l,x_u]}{2} \tfrac{2+j-[y_l,y_u]}{2} + p_{i,j+2} \tfrac{2+i-[x_l,x_u]}{2} \tfrac{[y_l,y_u]-j}{2} + p_{i+2,j} \tfrac{[x_l,x_u]-i}{2} \tfrac{2+j-[y_l,y_u]}{2} + p_{i+2,j+2} \tfrac{[x_l,x_u]-i}{2} \tfrac{[y_l,y_u]-j}{2}.$$

Next, we solve for the pixel value $p_{i,j}$ to get the constraint $q_{i,j}$:

$$q_{i,j} = \Big(p'_{i',j'} - p_{i,j+2} \tfrac{2+i-[x_l,x_u]}{2} \tfrac{[y_l,y_u]-j}{2} - p_{i+2,j} \tfrac{[x_l,x_u]-i}{2} \tfrac{2+j-[y_l,y_u]}{2} - p_{i+2,j+2} \tfrac{[x_l,x_u]-i}{2} \tfrac{[y_l,y_u]-j}{2}\Big) \Big(\tfrac{2+i-[x_l,x_u]}{2} \tfrac{2+j-[y_l,y_u]}{2}\Big)^{-1}$$

Because we don't have any constraints for the pixel values $p_{i+2,j}, p_{i,j+2}$ and $p_{i+2,j+2}$, we replace their values by the $[0,1]$ constraint and obtain:

$$q_{i,j} = \Big(p'_{i',j'} - \Big(\tfrac{2+i-[x_l,x_u]}{2} \tfrac{[y_l,y_u]-j}{2} - \tfrac{[x_l,x_u]-i}{2} \tfrac{2+j-[y_l,y_u]}{2} - \tfrac{[x_l,x_u]-i}{2} \tfrac{[y_l,y_u]-j}{2}\Big)[0,1]\Big) \Big(\tfrac{2+i-[x_l,x_u]}{2} \tfrac{2+j-[y_l,y_u]}{2}\Big)^{-1}$$

Instead of using standard interval analysis to compute the constraints for $p_{i,j}$, we use the following more efficient transformer: We replace $[x_l, x_u]$ and $[y_l, y_u]$ with the coordinate $(x, y) \in [x_l, x_u] \times [y_l, y_u]$ furthest away from (i, j), which is in our case (x_u, y_u) to obtain

$$\begin{aligned} q_{i,j} &= \Big(p'_{i',j'} - \Big(\tfrac{2+i-x_u}{2} \tfrac{y_u-j}{2} + \tfrac{x_u-i}{2} \tfrac{2+j-y_u}{2} + \tfrac{x_u-i}{2} \tfrac{y_u-j}{2}\Big)[0,1]\Big) \Big(\tfrac{2+i-x_u}{2} \tfrac{2+j-y_u}{2}\Big)^{-1} \\ &= \big[p'_{i',j'} - \big(\tfrac{2+i-x_u}{2} \tfrac{y_u-j}{2} + \tfrac{x_u-i}{2} \tfrac{2+j-y_u}{2} + \tfrac{x_u-i}{2} \tfrac{y_u-j}{2}\big), p'_{i',j'}\big] \big(\tfrac{2+i-x_u}{2} \tfrac{2+j-y_u}{2}\big)^{-1}. \end{aligned}$$

B.2 Algorithm

Here, we present the algorithm used to compute the inverse of a transformation. For the construction of the set C, we iterate only over the index set P. The set P is constructed do include all points in G that could yield non empty intersections $c_{i',j'}$, thus this is just to speed up the evaluation and equivalent otherwise to the algorithm described in the main part.

Data: Image $\boldsymbol{x}' \in \mathbb{R}^{m \times m}$, transform T, parameter range B, coordinates i, j

Result: Range for the pixel value $p_{i,j}$.

$N \leftarrow \begin{pmatrix} [i-2, i+2] \\ [j-2, j+2] \end{pmatrix}$

$\begin{pmatrix} [i'_l, i'_u] \\ [j'_l, j'_u] \end{pmatrix} \leftarrow T_B(N)$

$P \leftarrow \left\{ \begin{pmatrix} i' \\ j' \end{pmatrix} \;\middle|\; \begin{smallmatrix} i' \in \text{range}(\lfloor i'_l \rfloor, \ldots, \lceil i'_u \rceil, 2) \\ j' \in \text{range}(\lfloor j'_l \rfloor, \ldots, \lceil j'_u \rceil, 2) \end{smallmatrix} \right\}$

$C \leftarrow \left\{ c_{i',j'} := T_B^{-1} \begin{pmatrix} i' \\ j' \end{pmatrix} \cap N \;\middle|\; c_{i',j'} \neq \emptyset, (i', j') \in P \right\}$

$p_{i,j} \leftarrow [0,1] \cap \left(\bigcap_{c_{i',j'} \in C} q_{i-2,j-2}(c_{i',j'}) \cup q_{i,j-2}(c_{i',j'}) \cup q_{i-2,j}(c_{i',j'}) \cup q_{i,j}(c_{i',j'}) \right)$

Algorithm 1: Procedure to calculate the range for the pixel values of the inverse image.

B.3 Experimental Evaluation

To investigate the impact of refinement on the downstream error estimate we used 20 MNIST images, rotated each with 3 random angles and then proceeded to calculate the inverse. In the calculation, we considered the range $\Gamma_{\pm} = 10$. We see that a low number of refinements have a large impact on the error but the returns become quickly diminishing. The impact on the run time of a single additional refinement step is negligible.

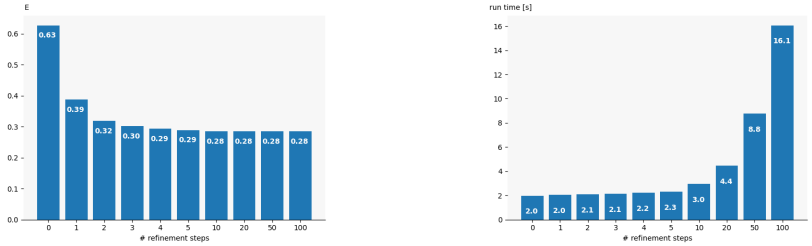


Figure 4: Interpolation and rounding error E as well as run time for different numbers of refinement steps.

(a) Rotated (b) Inverse (c) 10× refined inverse (d) Original

Figure 5: Computation of the inverse, analogous to Fig. 2, for images from ImageNet [31].

C Inverse for Rich Images

INDIVSPT performs poorly on large images, such as those from ImageNet as the inverse computation outlined in Section 6 produces a too large over-approximation of $\boldsymbol{x}$ leading to E estimates of around 40, while manageable value would be ≤ 2.

Fig. 5 shows the computed inverse for such images. We observe a pattern of artifacts in the inverse, where the pixel value can not be narrowed down sufficiently resulting in the large estimate of E. The result of the refined inverse is perfectly recognizable to a human observer (or a neural network), highlighting the promise of the algorithm for future applications.

D Experiment Details

D.1 Details for Section 7.3

To evaluate BASESPT we use the following classifiers. Note that Table 7 in App. E.1 contains results for further datasets:

MNIST [34] We trained a convolutional network consisting of CONV2D(k, n), with $k \times k$ filter size, n filter channels and stride 1, batch norm BN [35], maximum pooling MAXPOOL(k) on $k \times k$ grid, DROPOUT(p) [36] with probability p and linear layers LIN(a, b) from $\mathbb{R}^a$ to $\mathbb{R}^b$.

$\text{CONV2D}(5, 32), \text{RELU}, \text{BN}$
$\text{CONV2D}(5, 32), \text{RELU}, \text{MAXPOOL}(2), \text{DROPOUT}(0.2)$
$\text{CONV2D}(3, 64), \text{RELU}, \text{BN}$
$\text{CONV2D}(3, 64), \text{RELU}, \text{BN}, \text{MAXPOOL}(2), \text{DROPOUT}(0.2)$
$\text{CONV2D}(3, 128), \text{RELU}, \text{BN}$
$\text{CONV2D}(1, 128), \text{RELU}, \text{BN}, \text{FLATTEN}$
$\text{LIN}(128, 100), \text{RELU}$
$\text{LIN}(100, 10)$

We used data normalization for MNIST and trained for 180 epochs with SGD, starting from learning rate 0.01, decreasing it by a factor of 10 every 60 epochs. No other pre-processing was used.

FashionMNIST [37] We trained a ResNet-18 with data normalization. We trained for 180 epochs with SGD with an initial learning rate of 0.01, lowering it by a factor of 10 every 60 epochs.

CIFAR [33] We trained a ResNet-18 with data normalization. We trained for 90 epochs with SGD with an initial learning rate of 0.1, lowering it by a factor of 10 every 30 epochs. We resized GTSRB images to $32 \times 32 \times 3$.

ImageNet [31] We used the pre-trained ResNet50 from `torchvision`: `https://pytorch.org/docs/stable/torchvision/models.html`.

D.2 Details for Section 7.4

In Section 7.4 we use a ResNet-18 architecture for MNIST and a ResNet-110 for CIFAR-10 and, as in, App. D.1, ResNet-50 for (R)ImageNet. We trained them to be robust to image transformations (rotation, translation) as well as ℓ^2 noise.

To train networks that perform well when randomized smoothing is applied, we utilize the training procedure $\text{SMOOTHADV}_{\text{PGD}}$ as outlined in Salman et al. [8]. For each batch of samples we apply a randomized data augmentation, vignetting, and Gaussian blur. After this prepossessing we then apply $\text{SMOOTHADV}_{\text{PGD}}$ (noise restricted to the vingetted area) then evaluate or train on the batch.

The intuition behind the Gaussian blur is that many artifacts, such as the interpolation error are have high frequencies. The blur acts as a low-pass filter and discards high frequency noise. This does not strongly impact the classification accuracy, but drastically reduces the error estimate and therefore the amount of noise that needs to be added for robust classification. The filter is parameterized by σ_b and the filter size s_b. Formally the filter is a convolution with a filter matrix $F \in \mathbb{R}^{s_b \times s_b}$. Each entry in F is filled with values of a two dimensional Gaussian distribution centered at the center of the matrix and evaluated at the center of the entry. Afterwards the matrix is normalized such that $\sum_{i,j}, F_{i,j} = 1$.

In the error estimation and inference we use the same prepossessing as during training.

MNIST For MNIST we use a ResNet-18 (that takes a single color channel in the input layer), which we trained with PGD step size 0.2, batch size 1024, and initial learning rate 0.01 over 180 epochs, lowering the learning rate every 60 epochs. For $\text{DISTSPT}^{\mathcal{D}}$ we use $\sigma = 0.22$ and data augmentation with rotations in $[-90, 90]$ degrees for the rotation model and $\sigma = 0.3$ and random translations of $\pm 50\%$ for the translation model. For the Gaussian blur we use $\sigma_b = 2.0$ with filter size $s_b = 5$ on all models.

For $\text{DISTSPT}^{\boldsymbol{x}}$ we use a model trained with $\sigma = 0.15$ for rotations and but the same translation model.

CIFAR-10 For $\text{DISTSPT}^{\mathcal{D}}$ we train a ResNet-110 with batch size 256, $\sigma = 0.25$ and random rotations in $[-60, 60]$ as well as $\text{SMOOTHADV}_{\text{PGD}}$ with $m = 1$ samples, $t = 8$ steps and warmup of 10 epochs for a perturbation size of 0.5. We train over 150 epochs and lower the learning rate every 50 steps. For the Gaussian blur we use $\sigma_b = 1.0$ and $s_b = 5$.

For DISTSPT$^{\boldsymbol{x}}$ we use $\sigma = 0.12$, perturbation size of 0.25, $m = 8$ and $t = 1$ and keep other parameters the same. Both variants take about 17 minutes per epoch on a single GeForce RTX 2080 Tis.

(Restricted) ImageNet We trained with a batch size of 400 for 90 epochs using stochastic gradient decent with a learning rate starting at 0.1, which is decreased by a factor 10 every 30 epochs. On both datasets , we used $\sigma = 0.5$ and PGD step size 1.0, as well as $\sigma_b = 2.0$ and $s_b = 5$. For Restricted ImageNet we train with random rotation in $[-60, 60]$ and for ImageNet in $[-30, 30]$.

Training 1 epoch of ImageNet with 6 GeForce RTX 2080 Tis and a 16-core node of aw Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz takes roughly 2.5 hours and roughly 30 minutes for Restricted ImageNet. For the accuracy of b in Table 3, we evaluate four settings — with vingetting, with Gaussian blur, with both and with neither — and report the highest. Table 5 shows a comparison across all settings.

Table 5: Base models evaluated on the whole data set either with Gaussian blur (G), Vingetting (V), both or neither.

Model	T^I	Standard	+G	+V	+G+V
MNIST, DISTSPT$^{\mathcal{D}}$	R^I	0.98	0.98	0.98	0.98
MNIST, DISTSPT$^{\boldsymbol{x}}$	R^I	0.98	0.98	0.98	0.98
MNIST	Δ^I	0.94	0.92	0.94	0.93
CIFAR-10 DISTSPT$^{\mathcal{D}}$	R^I	0.34	0.36	0.56	0.56
CIFAR-10 DISTSPT$^{\boldsymbol{x}}$	R^I	0.75	0.76	0.70	0.70
RImageNet	R^I	0.77	0.77	0.78	0.77
ImageNet	R^I	0.38	0.32	0.38	0.32

To sample E for (R)ImageNet we use a server with an AMD EPYC 7601 processor with 128 threads.

Table 6 shows a version Table 3 in the layout of the version of this paper published at NeurIPS'20.

Table 6: Evaluation of DISTSPT for $T^I := R^I$. $\epsilon_{\max}$ is computed on the training set. We show the test set accuracy of b, certified accuracy of g and distribution of the obtained certification radius r_γ, along with the average run time t and the number of used samples n_γ, n_δ. $^\#$ denotes values obtained by sampling. Each certificate hold with overall confidence 0.99.

			Acc.		r_γ percentile					
Dataset	$\epsilon_{\max}$	E	b	g	25^{th}	50^{th}	75^{th}	t [s]	n_γ	n_δ
MNIST	0.36	0.45	0.98	0.89	52.95	57.22	57.22	21.56	200	10000
CIFAR-10	0.51	0.55	0.56	0.31	24.80	$30.00^\dagger$	$30.00^\dagger$	89.75	50	15000
CIFAR-10	0.51	0.55	0.56	0.32	$30.00^\dagger$	$30.00^\dagger$	$30.00^\dagger$	351.47	200	15000
RImageNet	0.91	1.20	0.78	0.74	$30.00^\dagger$	$30.00^\dagger$	$30.00^\dagger$	100.73	50	2500
RImageNet	0.91	1.35	0.78	0.64	$30.00^\dagger$	$30.00^\dagger$	$30.00^\dagger$	100.13	50	2500
ImageNet	0.91	0.95	0.38	0.30	14.51	24.34	$30.00^\dagger$	100.21	50	2500
ImageNet	0.91	1.20	0.38	0.23	12.38	21.47	$30.00^\dagger$	100.73	50	2500
ImageNet	0.91	1.35	0.38	0.16	10.46	21.47	$30.00^\dagger$	100.44	50	2500

D.3 Details for Section 7.5

For rotation we use the $\sigma = 0.22$ model as in App. D.2 and for translation also the same model.

E Additional Experiments

E.1 Additional Results for Section 7.3

Table 7 is an extended version of Table 2 and provides results for additional datasets.

Table 7: Extended version of Table 2. Evaluation of BASESPT on 1000 images. The attacker used `worst-of-100`. We use $n_\gamma = 1000, \sigma_\gamma = \Gamma_\pm$.

			Acc.	adv. Acc.		
Dataset	T^I	$\Gamma_\pm$	b	b	g	t [s]
MNIST	R^I	30°	0.99	0.73	0.99	0.97
FMNIST	R^I	30°	0.91	0.13	0.87	7.98
CIFAR-10	R^I	30°	0.91	0.26	0.85	0.95
GTSRB	R^I	30°	0.91	0.30	0.88	8.00
ImageNet	R^I	30°	0.76	0.56	0.76	5.43
MNIST	Δ^I	4	0.99	0.03	0.53	0.86
FMNIST	Δ^I	4	0.91	0.10	0.50	6.12
CIFAR-10	Δ^I	4	0.91	0.44	0.79	0.95
GTSRB	Δ^I	4	0.91	0.30	0.63	5.17
ImageNet	Δ^I	20	0.76	0.65	0.75	6.70

Table 8: We first use BASESPT to obtain the certification radius r_γ on 30 images and subsequently sample from the parameter space indicated by $\Gamma_\pm = r_\gamma$ and checked whether the certificate holds for them. We use 30 samples and $n_\gamma = 2000$ samples for the smoothed classifier. The last column shows the number of images for which we found violations.

Dataset	T^I	$\Gamma_\pm$	median r_γ	r_γ violated
MNIST	R^I	30°	28.34	0
FMNIST	R^I	30°	13.45	1
CIFAR-10	R^I	30°	19.16	14
GTSRB	R^I	30°	20.93	0
ImageNet	R^I	10°	27.13	1
MNIST	Δ^I	4	1.12	0
FMNIST	Δ^I	4	1.78	1
CIFAR-10	Δ^I	4	4.76	14
GTSRB	Δ^I	4	2.58	0
ImageNet	Δ^I	20	16.43	0

Table 9: Same setup as in Table 8, but with circular vignetting.

Dataset	T^I	$\Gamma_\pm$	median r_γ	r_γ violated	r_γ violated, no interpolation
MNIST	R^I	30°	28.34	0	0
FMNIST	R^I	30°	17.07	0	0
CIFAR-10	R^I	30°	11.49	10	0
GTSRB	R^I	30°	25.28	0	0

E.2 "Certification Radius" of BASESPT

As BASESPT uses Theorem 3.2 to justify the heuristic, this also makes it tempting to use the bound r_γ provided by it. However, as the assumptions of Theorem 3.2 are violated it does not formally present a certification radius. Here we investigate if and how much it holds nevertheless. To do this we construct a smoothed classifier g from an undefended base classifier b and calculated the certification radius r_γ. Subsequently, we sampled 100 new rotated images in the parameter space induced by $\Gamma_\pm = r_\gamma$ and evaluated on them. The results are shown in Table 8. While generally robust, the radius does not constitute a certificate, as we can clearly find violations.

In the context of rotation R^I we add circular vignetting (as we do for DISTSPT and INDIVSPT) to make the behavior closer to a composing transformation. For this experiment, we retrained the same networks, but applied the vignette during training. Results are shown in Table 9 where we can see that this already decreases the number of violations for CIFAR-10 and FMNIST. In a final step

we assume knowledge of the attacker parameter γ and replace $R^I_\beta \circ R^I_\gamma$ (for the same images) with $R^I_{\beta+\gamma}$ in the evaluation of the classifier, in which case Theorem 3.2 should hold and indeed we don't observe any more violations.

E.3 Additional Results for Section 7.4

Beyond Bilinear Interpolation BASESPT and DISTSPT can directly be applied to image transformations using other interpolation schemes without any adaption. INDIVSPT, however, requires the adaption of the inverse algorithm. While this is generally possible, we consider it beyond the scope of this work.

We guess (based on 1000 samples) and verify E using 1000 samples for $\boldsymbol{x}$ and 8000 for β. We summarize these results in Table 10. On datasets other than MNIST we observe E larger than possible. At manageable levels, the q_E becomes too low for practical purposes.

On MNIST with the same settings as for DISTSPT$^{\mathcal{D}}$ we certify 90 out of 100 images at $r_\gamma = 30$ (for bilinear interploation with $E = 0.45$ 91 can be certified).

Table 10: E and q_E for bicubic interpolation.

Dataset	E	q_E
MNIST	0.5	0.99
CIFAR-10	1.10	0.99
CIFAR-10	0.55	0.27
ImageNet	2.50	0.99
ImageNet	1.20	0.28

E.4 Audio Volume Change

To show that our method can be used beyond image transformation we showcase an adaption to audio volume changes. The volume of an audio signal can be changed by multiplying the signal with a constant. In order to change the signal $\boldsymbol{x}$ by β (measured in decibel $[\beta] = \text{dB}$) we multiply $\boldsymbol{x}$ by $10^{\beta/20}$. Thus the transformation is $\psi_\beta(\boldsymbol{x}) := 10^{\beta/20} \cdot \boldsymbol{x}$, which composes:

$$\psi_\beta \circ \psi_\gamma(\boldsymbol{x}) = 10^{(\beta+\gamma)/20} \cdot \boldsymbol{x} = \psi_{\beta+\gamma}(\boldsymbol{x}).$$

In practice such signals are stored in final precision, e.g. 16-bit, thus potentially introducing rounding errors, with an ℓ^2-norm bound by E. If this is ignored BASESPT can be applied to obtain guarantees. Otherwise, DISTSPT and INDIVSPT can be used to obtain sound bounds.

To evaluate this we use the speech commands dataset [38], consisting of 30 different commands, spoken by people, which are to be classified. The length of the recordings are one second each. We use a classification pipeline that converts audio wave forms into MFCC spectra [39] and then treats these as images and applies normal image classification. We use a ResNet-50, that was trained with Gaussian noise, but not SMOOTHADV$_{\text{PGD}}$. We apply the noise before the waveform is converted to the MFCC spectrum.

For DISTSPT we estimate E to be 0.005 with the parameters $\rho_E = 0.05, \sigma_\gamma = 3$ and $\Gamma = 3$ (for which $q_E \approx 0.75$). On 100 samples, the base classifier f was correct 93 times. At r_{gamma} of 1, 2, 3 and 4 the certified accuracy was 0.92, 0.89, 0.83 and 0.69 respectively. This corresponds to ± 1.12, ± 1.26, ± 1.41 and ± 1.58 dB. At $n_\gamma = 150$ and $n_\epsilon = 400$ the average certification time was 26.80 s. We use $\alpha_\gamma = 0.004, \alpha_\epsilon = \frac{0.005}{n_\gamma}$, assuming (but not computing) $\alpha_E = 0.001$ here, for a total confidence of 0.99 in each certificate.

To investigate INDIVSPT we use $\sigma_\gamma = 0.85, \Gamma = 1.05$. For 92 out of 100 perturbed audio signals to compute ϵ. We obtained $\epsilon_{\max} \leq 0.0055$ and for 68 an $\epsilon \leq 0.005$, which together with our results for DISTSPT suggests the applicability of the method. For each signal we used 100 samples for β. For cases with $\epsilon_{\max} > 0.0055$ we in fact observed $\epsilon_{\max} \gg 0.0055$, as here many parts of the signal were amplified beyond the precision of the 16-bit representation and clipped to ± 1. This makes the information unrecoverable and sound error bound estimates large.

Table 11: Maximum observed errors and without gaussian blur (G) and without vignetting (V).

Dataset	Both	-V	-G	-V-G
MNIST	**0.36**	**0.36**	2.47	2.51
CIFAR-10	**0.51**	6.08	2.66	18.17
ImageNet	**0.91**	70.66	9.25	75.69

Table 12: Correct classifications and by the model and verifications by DeepG [11], with and without vignetting (V), out of 100 images.

Model	Correct	[11]	[11]+V
MNIST	98	86	87
CIFAR-10	74	65	32
CIFAR-10+V	78	63	23

F Further Comparison and Ablation

To show that the vignette and Gaussian blur are essential to our algorithm we perform a small ablation study. Table 11 shows the maximal error observed when sampling as in DISTSPT. We use the same setup as in Section 7.4, but with 10000 samples for ImageNet.

Both, vignetting and Gaussian blur reduce the error bound significantly for DISTSPT and INDIVSPT. On CIFAR-10 and ImageNet vignetting is very impactful because the corners of images are rarely black in contrast to MNIST. Li et al. [13] uses vignetting for the same reason. Without either of the methods bounding the error would not be feasible.

For INDIVSPT vignetting is crucial, even for MNIST, as we can make no assumptions for parts that are rotated into the image. Thus we need to set these pixels to the full $[0, 1]$ interval (see Fig. 2). Without Gaussian blur the certification rate drops to 0.11.

Further, we extend this comparison to related work: We extended Balunovic et al. [11] (Table 1 in their paper) to include vignetting. The results are shown in Table 12. We also retrained their CIFAR-10 model with vignetting (CIFAR-10+V) for completeness. While vignetting on MNIST slightly helps (+1 image verified) on CIFAR-10 it leads to a significant drop. Including Gaussian blur into [11] would require non-trivial adaption of the method. However, we implemented this for interval analysis (on which their method is built) and found no impact on results.