Machine Learning (2023) 112:3977–4012
https://doi.org/10.1007/s10994-021-05994-9

Metrics and methods for robustness evaluation of neural networks with generative models

Igor Buzhinsky[1,2] · Arseny Nerinovsky[1] · Stavros Tripakis[3]

Received: 16 March 2020 / Revised: 20 January 2021 / Accepted: 10 May 2021 /
Published online: 6 July 2021

Abstract

Recent studies have shown that modern deep neural network classifiers are easy to fool, assuming that an adversary is able to slightly modify their inputs. Many papers have proposed adversarial attacks, defenses and methods to measure robustness to such adversarial perturbations. However, most commonly considered adversarial examples are based on perturbations in the input space of the neural network that are unlikely to arise naturally. Recently, especially in computer vision, researchers discovered "natural" perturbations, such as rotations, changes of brightness, or more high-level changes, but these perturbations have not yet been systematically used to measure the performance of classifiers. In this paper, we propose several metrics to measure robustness of classifiers to natural adversarial examples, and methods to evaluate them. These metrics, called latent space performance metrics, are based on the ability of generative models to capture probability distributions. On four image classification case studies, we evaluate the proposed metrics for several classifiers, including ones trained in conventional and robust ways. We find that the latent counterparts of adversarial robustness are associated with the accuracy of the classifier rather than its conventional adversarial robustness, but the latter is still reflected on the properties of found latent perturbations. In addition, our novel method of finding latent adversarial perturbations demonstrates that these perturbations are often perceptually small.

Keywords Reliable machine learning · Adversarial examples · Natural adversarial examples · Generative models

Editors: Daniel Fremont, Mykel Kochenderfer, Alessio Lomuscio, Dragos Margineantu, Cheng Soon-Ong.

✉ Igor Buzhinsky
igor.buzhinsky@gmail.com

Extended author information available on the last page of the article

1 Introduction

Unlike in more conventional software engineering, the problem of ensuring reliability of machine learning (ML) based software is complicated by the fact that ML-based models, such as *artificial neural networks* (ANNs), are not programmed explicitly. Instead, they significantly depend on the data on which they are trained. The traditional form of assessing model performance based on validation/test data (e.g., a holdout set) and measures such as accuracy or F-score, become insufficient when the models interact with the real world, such as in the cases of aircraft and unmanned vehicle control. This is proven by the discovery of *adversarial examples* (Szegedy et al., 2014)—slightly perturbed inputs that cause ANNs to malfunction, for example by misclassifying an image. For a human, adversarial examples may be even indistinguishable from original, unperturbed inputs. Adversarial examples are often produced in a rather artificial environment, by adopting special algorithms that perturb the input until a certain criterion is reached, but recent evidence (Akhtar and Mian, 2018; Gilmer et al., 2018) suggests that they may transfer to the material world.

The classic framework of *empirical risk minimization* (ERM) (Vapnik, 2013), where the classifier is trained on available data samples, is used to achieve high values of sample-based metrics such as accuracy or F-score. However, if the classifier must be protected from adversarial examples, ERM is insufficient, and robust optimization (Madry et al., 2018) with projected gradient descent (PGD) can be used instead. This corresponds to enforcing *adversarial robustness* (Anderson et al., 2019; Bastani et al., 2016; Fawzi et al., 2018; Huang et al., 2017; Katz et al., 2017; Moosavi-Dezfooli et al., 2016; Singh et al., 2019), which is often treated either as a metric (Bastani et al., 2016; Fawzi et al., 2018; Moosavi-Dezfooli et al., 2016) specifying the minimum magnitude of an adversarial perturbation, or as a specification (Anderson et al., 2019) stating that the decision of the ANN must be invariant to perturbations of input of a certain form. Adversarial robustness can be local (Anderson et al., 2019; Fawzi et al. 2018; Huang et al., 2017; Katz et al., 2017; Singh et al., 2019) or global (Katz et al., 2017).

Traditional adversarial examples are based on perturbations in the input space of the ANN that are constrained with ℓ_p (e.g., ℓ_2 or ℓ_∞) norms. The resulting adversarial examples are highly improbable to arise naturally (Song et al., 2018a), but it was shown that even *natural adversarial examples* (i.e., the ones plausible under the data distribution) exist (Amadou Dia et al., 2019; Dreossi et al., 2018; Engstrom et al., 2019; Gu et al., 2019; Hendrycks et al., 2019; Jalal et al. 2019; Song et al., 2018b; Zhao et al., 2017). While conventional adversarial examples often require 2D or 3D printing of precomputed images (Akhtar and Mian, 2018) to be applied in the real world, a natural adversary could be seen as a manipulator of high-level features of classified objects. The plausibility of natural adversarial examples makes statistical attack detection and defense approaches, such as (Song et al., 2018a; Samangouei et al., 2018), less reliable. At the same time, these examples are theoretically interesting: unlike traditional adversarial examples, they show the failures of ANNs on the distribution on which they were trained.

Construction of a subclass (Amadou Dia et al., 2019; Jalal et al., 2019; Song et al., 2018b; Zhao et al., 2017) of natural adversarial examples is possible with the help of generative models, such as generative adversarial networks (GANs) (Goodfellow et al., 2014) and generative autoencoders (Makhzani et al., 2015). Previous works that considered natural adversarial examples mostly focused on attacks and defenses rather than assessing the performance of classifiers. In addition, Jalal et al. (2019) applied adversarial examples for ANN training, although the focus so far has been on adversarial robustness in the input space of the ANN. This paper utilizes generative models as a means of capturing real-world data distributions in order to specify and evaluate *performance metrics* for ANN classifiers in terms of probabilities, likelihood and distances in latent spaces of generative models. As a result, our metrics capture the robustness of classifiers to natural adversarial examples. More precisely, the contributions of the paper are as follows:

1. We propose a framework to evaluate the performance of deep feed-forward ANN classifiers on natural adversarial examples with the help of generative models and their latent spaces. The implementation of the framework is publicly available online.
2. Within this framework, we propose *latent space performance metrics*—novel performance metrics for feed-forward ANN classifiers that are based on probabilistic reasoning in latent spaces of generative models, and, informally speaking, measure the "resistance" of the classifier to natural adversarial examples. The naturality of adversarial examples is achieved by (1) operating in the latent space of the generative model, (2) considering a distribution-preserving model of noise, and (3) generating adversarial examples by adding random noise, or by searching for worst-case examples that are bounded by the likelihood of noise.
3. We propose methods to approximately evaluate these metrics in a white-box setting using (1) sampling and (2) gradient-based search of adversarial perturbations in the latent space. The latter method is a form of untargeted attack based on PGD. We show that such a search is possible not only with GANs (Zhao et al., 2017), but also with generative autoencoders.
4. With the proposed framework, metrics and methods, we reveal interesting properties of ANN classifiers with respect to natural adversarial examples, which contributes to understanding the latter better. On four image classification case studies, we examine classifiers trained traditionally and in a way that achieves adversarial robustness, and evaluate their performance according to latent space performance metrics. Our PGD-based untargeted attack yields perceptually smaller latent perturbations than reported earlier (Zhao et al., 2017), and we find positive association between latent counterparts of adversarial robustness and the accuracy of a classifier on clean images. Moreover, we did not identify a similar association for latent space performance metrics and conventional adversarial robustness, but we found that the latter leads to minimum latent adversarial perturbations being further from the original image in the original (non-latent) space as well as perceptually.

The rest of the paper is structured as follows. Section 2 presents background material. Section 3 motivates the use of generative models to measure ANN classifier performance, and proposes corresponding metrics. In Sect. 4, approaches are given to evaluate these metrics. Evaluation of deep convolutional neural network (CNN) classifiers with these approaches is performed in Sect. 5. Section 6 reviews related work, and Sect. 7 concludes the paper.

2 Preliminaries

2.1 Artificial neural networks

A *feed-forward artificial neural network* (ANN) $\mathcal{N}$ is a parametric model that predicts some outcome y (a single number or a vector) based on some input vector x of dimension

n_I. By feed-forward, we mean that the input is supplied to the network at once and is passed through a predefined computation graph with a finite number of computation nodes. When the input is an image, $\mathcal{N}$ is usually a *convolutional neural network* (CNN). In this paper, we focus on the *classification task*, where $\mathcal{N}$ must assign its input to one of $m > 1$ classes. Thus, we have $\mathcal{N} : \mathbb{R}^{n_I} \to \{1, ..., m\}$. We assume that class prediction is done as follows: $\mathcal{N}$ first produces real-valued scores of each class i, to which we will refer as the values of the *scoring function* $S_{\mathcal{N}}(x, i)$, and the actually predicted class is the one with the maximum score: $\mathcal{N}(x) = \arg\max_i S_{\mathcal{N}}(x, i)$. In addition, we require that $S_{\mathcal{N}}(x, i)$ is continuous and almost everywhere differentiable with respect to x.

ANN classifiers are typically trained in a supervised way with some form of *gradient descent* (e.g., stochastic gradient descent), using samples $x_1, ..., x_k \in \mathbb{R}^{n_I}$, which are paired with respective reference class labels $y_1, ..., y_k \in \{1, ..., m\}$. These pairs $(x_1, y_1), ..., (x_k, y_k)$ are assumed to come from *joint distribution* $\mathcal{J}$, whose marginals are the *input data distribution* $\mathcal{X}$ and the *class label distribution* $\mathcal{Y}$.

2.2 Generative models

A *generative adversarial network* (GAN) (Goodfellow et al., 2014), which consists of two feed-forward ANNs called the *discriminator* and the *generator* $\mathcal{G}$, is trained to make $\mathcal{G}$ generate elements of some target data distribution $\mathcal{X}$ of n_I-dimensional vectors (in the simplest case, without sample labels). Data generation is done by applying $\mathcal{G}$ to a low-dimensional vector $l \in \mathbb{R}^{n_L}$ sampled from the *latent code distribution* $\mathcal{L}$ (typically, $N(0, I)$). If $l \sim \mathcal{L}$, then for a well-trained GAN we may assume that $\mathcal{G}(l) \sim \mathcal{X}$. Often, the dimension of $\mathcal{L}$ is made smaller than the dimension of $\mathcal{X}$: $n_L < n_I$. The set of all latent codes (usually, just $\mathbb{R}^{n_L}$) is called the *latent space*. By contrast, we will refer to the input space of an ANN classifier ($\mathbb{R}^{n_I}$) as the *original space*. With some enhancements, GANs may be also capable of *reconstruction*—finding latent representation $l \in \mathbb{R}^{n_L}$ for the given original vector $x \in \mathbb{R}^{n_I}$ such that $\mathcal{G}(l)$ is close to x (e.g., according to some norm in the original space). For example, this may be done by training an additional ANN $\mathcal{I} : \mathbb{R}^{n_I} \to \mathbb{R}^{n_L}$ called an *inverter* (Hendrycks et al., 2019). However, obtaining good inversions, especially for GANs that generate high-resolution images, requires more effort: for example, Bau et al. (2019) performed layer-wise inversion and combined it with gradient-based optimization.

An *autoencoder* $(\mathcal{N}^{\mathcal{E}}, \mathcal{N}^{\mathcal{D}})$, where $\mathcal{N}^{\mathcal{E}}$ and $\mathcal{N}^{\mathcal{D}}$ are feed-forward ANNs called the *encoder* and the *decoder* respectively, is a model whose goal is to *compress* (*encode*) its inputs $x \in \mathbb{R}^{n_I}$ to low-dimensional vectors $l = \mathcal{N}^{\mathcal{E}}(x) \in \mathbb{R}^{n_L}$ (again, $n_L < n_I$) such that approximate *decompression* (*decoding*, *reconstruction*) can be achieved: $\mathcal{N}^{\mathcal{D}}(l)$ is close to x. A *generative autoencoder*, such as in (Heljakka et al., 2020; Makhzani et al., 2015), is an autoencoder whose decoder is additionally trained to sample from the original distribution $\mathcal{X}$—thus, essentially, a generative autoencoder performs both the tasks of an autoencoder and a GAN. For a well-trained generative autoencoder, we may assume both $l \sim \mathcal{L} \Rightarrow \mathcal{N}^{\mathcal{D}}(l) \sim \mathcal{X}$ and $x \sim \mathcal{X} \Rightarrow \mathcal{N}^{\mathcal{E}}(x) \sim \mathcal{L}$.

To summarize, generative models are capable of data *generation* from low-dimensional vectors. By using special types of generative models or enhancing existing generative models, it is also possible to achieve data *reconstruction*.

2.3 Adversarial examples and perturbations

Suppose that $\mathcal{N}$ is an ANN classifier. An *adversarial example* is an input x' to $\mathcal{N}$ such that $x' \in A(x)$ and $\mathcal{N}(x') \neq \mathcal{N}(x)$, where x is a real data sample, $A(x)$ is the set of allowed changes of x (often, it is taken as the ε-ball around x according to the ℓ_p norm: $A(x) = \{x' \mid \|x' - x\|_p \leq \varepsilon\}$). $\Delta x = x' - x$ is the corresponding *adversarial perturbation*.

Adversarial examples and adversarial perturbations have been first found to exist by Dalvi et al. (2004) and Globerson and Roweis (2006), but were publicized by Szegedy et al. (2014), who presented human-indistinguishable ImageNet perturbations. Since 2013, many adversarial attacks and defenses have been proposed (Akhtar and Mian, 2018). While many proposed defenses were shown to be ineffective (Gilmer et al., 2018), attacks were transported to the real world (Akhtar and Mian, 2018), raising concerns regarding the safety and security of deep ANNs.

For adversarial perturbations bounded with ℓ_2 and ℓ_∞ norms, *projected gradient descent* (PGD) has been shown (Madry et al., 2018) to be the best adversary that has access only to $\nabla_x S_{\mathcal{N}}(x, \cdot)$. The most common method of defense is *robust optimization* with PGD, where training is done on adversarial examples for the current version of the ANN. Gilmer et al. (2019) showed that it is possible to train the classifier on samples with added visible Gaussian noise instead of specially crafted adversarial examples.

Recent works explain adversarial examples through the peculiarities of the multidimensional geometry (Gilmer et al., 2019) and the fact that conventional ERM-based training does not introduce human priors to the training process (Ilyas et al., 2019). Samangouei et al. (2018) and Song et al. (2018a) also hypothesized that adversarial examples do not lie on the data manifold of the training distribution, but several works show that even *natural* adversarial examples exist, such as the ones that come from the real world (Hendrycks et al., 2019), are made by rotations and translations (Engstrom et al., 2019), color distortions (Gu et al., 2019), semantic changes (Dreossi et al., 2018), looping over consequent video frames (Gu et al., 2019), and created with generative models (Amadou Dia et al., 2019; Jalal et al., 2019; Song et al., 2018b; Zhao et al., 2017). *Latent space adversarial examples*, or adversarial examples that correspond to some latent codes of a generative model, may be based on perturbations (Amadou Dia et al., 2019; Zhao et al., 2017) or generated from scratch (Song et al., 2018b). Jalal et al. (2019) showed that latent space adversarial examples can be used to enhance robust optimization and increase the overall robustness of the classifier.

2.4 Performance metrics for adversarial robustness

Often, the set of possible adversarial examples is defined locally for each input x—for example, as an ℓ_p ε-ball, or as a set of rotations of x (Engstrom et al., 2019). The robustness of the classifier is then measured as its accuracy on worst-case inputs taken from such sets. For the ℓ_∞ norm, Bastani et al. (2016) formalized this metric as *adversarial frequency*. Adversarial frequency, however, depends on ε. A different way to measure robustness, which is free from this hyperparameter, is *adversarial severity* (Bastani et al., 2016)—the expected (with $x \sim \mathcal{X}$) minimum distance from x to an adversarial example. The corresponding local metric is *pointwise robustness*, which is the minimum distance to an adversarial example for a particular x. In this paper, we will define metrics that are based on pointwise robustness, adversarial frequency and severity but operate with different norms in different spaces. Known metrics that are defined in the original space will be referred to as *conventional* metrics.

3 Latent space performance metrics

In this paper, we are interested in *specifying and evaluating performance metrics for ANN classifiers with the help of generative models*. In addition, we would like to evaluate these metrics given the original training and validation data. This section will propose several such *latent space performance metrics*, and methods to evaluate them will be proposed in Sect. 4.

3.1 Preliminary definitions

Suppose that $\mathcal{N} : \mathbb{R}^{n_I} \to \{1, ..., m\}$, $m > 1$, is a feed-forward ANN classifier with scoring function $S_{\mathcal{N}}$. The goal of $\mathcal{N}$ is to correctly classify input vectors drawn from distribution $\mathcal{X}$. In the most general case, there may be no unique correct label for an input vector, but rather there is a *joint distribution* $\mathcal{J}$ of pairs (x, y) of an input vector x and its label y. For simplicity, we assume that $\mathcal{N}$ is validated on samples drawn exactly from $\mathcal{J}$, although the training might have been performed on a distribution induced by data augmentation of input vectors x.

Suppose that $\mathcal{L}_i$, $1 \le i \le m$, are n_L-dimensional ($n_L < n_I$) *class-conditional latent distributions* (often assumed to be $N(0, I)$) such that we have trained transformations $D_i : \mathbb{R}^{n_L} \to \mathbb{R}^{n_I}$ that generate samples from class-conditional data distributions $\mathcal{X}_i$: $l \sim \mathcal{L}_i \Rightarrow D_i(l) \sim \mathcal{X}_i$. In certain cases (see models capable of reconstruction in Sect. 2.2), we may additionally have transformations $E_i : \mathbb{R}^{n_I} \to \mathbb{R}^{n_L}$ that return latent code approximations of n_I-dimensional vectors. We would like D_i to be compatible with gradient descent, i.e., continuous and almost everywhere differentiable, but we do not require the same from E_i.

3.2 Motivation for latent space performance metrics

With both D_i and E_i, we can convert vectors to the latent space and back. Assuming that the latent space corresponds to a well-trained generative model, working in it has the following benefits compared to the original space:

1. For a random vector $l \sim \mathcal{L}_i$, $D_i(l)$ has a distribution that was trained to approximate $\mathcal{X}$.
2. Changes of the vector in the latent space are high-level in terms of the original representation.
3. For each class i, the image $D_i(\mathbb{R}^{n_L})$ contains an infinite number of diverse data samples, which may be useful to evaluate $\mathcal{N}$ or train it further.
4. The aforementioned samples can not only be generated at random, but also can be optimized with gradient-based techniques to optimize a certain objective (e.g., $S_{\mathcal{N}}$).

As many performance metrics, such as accuracy, adversarial frequency and severity, will remain meaningful when the original space is replaced with the latent one, in this paper we

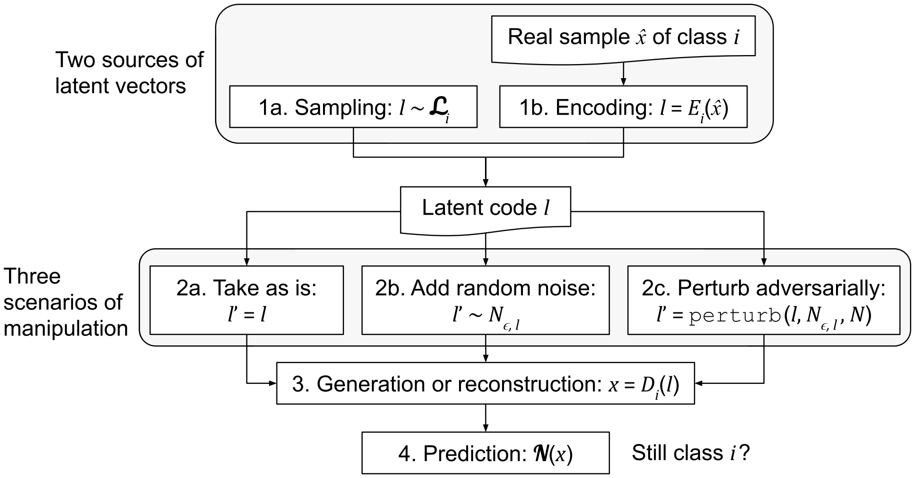

Two sources of latent vectors
Real sample $\hat{x}$ of class i
1a. Sampling: $l \sim \mathcal{L}_i$
1b. Encoding: $l = E_i(\hat{x})$
Latent code l
Three scenarios of manipulation
2a. Take as is: $l' = l$
2b. Add random noise: $l' \sim N_{\epsilon,l}$
2c. Perturb adversarially: $l' = \texttt{perturb}(l, N_{\epsilon,l}, N)$
3. Generation or reconstruction: $x = D_i(l)$
4. Prediction: $\mathcal{N}(x)$
Still class i?

Fig. 1 Overview of considered scenarios

will mainly *move conventional performance metrics to the latent space*. We will do it in a way that provides additional benefits related to the probabilistic interpretation of the latent space—for example, while considering adversarial perturbations, we will take care that the data remains plausible according to $\mathcal{X}$.

3.3 Possible scenarios

Intuitively, sampling from $\mathcal{L}_i$ gives latent vectors l such that $D_i(l)$ are instances of class i. Previous works on natural adversarial examples obtained latent vectors based on generation (Song et al. 2018b) and reconstruction (Zhao et al., 2017). These two scenarios of obtaining $D_i(l)$ (see the upper part of Fig. 1, paths 1a and 1b) directly correspond to two operations that generative models are capable of (see Sect. 2.4):

1. Sample $l \sim \mathcal{L}_i$ and *generate* $x = D_i(l)$.
2. Take a random real sample $\hat{x} \sim \mathcal{X}_i$, encode it as $l = E_i(\hat{x})$, and *reconstruct* it as $x = D_i(l)$.

In this paper, we are interested in finding latent space counterparts for the following metrics (each of them will correspond to one of three scenarios in the lower part of Fig. 1):

1. *Accuracy*, as well as similar metrics based on counting success frequencies (Fig. 1, path 2a). This is the simplest case: it is sufficient to calculate the success frequency of $\mathcal{N}$ on reconstructed or generated samples. This will be formalized in Sect. 3.4.
2. *Corruption robustness to random noise* (Gilmer et al., 2019; Hendrycks and Dietterich, 2019) (Fig. 1, path 2b). While in the original space the addition of noise is a form of data corruption, in the latent space this noise will introduce high-level changes to the input, and we could measure the success frequency of $\mathcal{N}$ on such inputs. In Sect. 3.5, we will introduce a family of noise-adding distributions $N_{\epsilon,l}$ that retain the transformed data plausible even for large noise, and define a corresponding performance metric.

3. *Adversarial robustness* (Anderson et al., 2019; Fawzi et al., 2018; Huang et al., 2017; Katz et al., 2017; Singh et al., 2019) (Fig. 1, path 2c). Adversarial robustness in the latent space can be treated as "resistance" to worst-case noise additions that are bounded according to noise likelihood and optimized to degrade the performance of $\mathcal{N}$. The connection between noise corruption robustness and adversarial robustness exists already in the original space: for example, if the noise is Gaussian, its likelihood is determined by its ℓ_2 norm, a threshold on which is a common constraint on adversarial perturbations. What is more, noise corruption robustness and adversarial robustness were found to be highly related (Gilmer et al., 2019). The corresponding latent space metrics will be formalized in Sect. 3.6.

3.4 Accuracy in the latent space

Probably the simplest thing that can be done with generative models is to evaluate the accuracy of the classifier on generated and reconstructed data items. This situation corresponds to the absence of any adversary. These ideas are formalized in the following definitions:

Definition 1 The **latent generation accuracy** (LGA) of $\mathcal{N}$ is:

$$\mathrm{LGA}(\mathcal{N}) \stackrel{\text{def}}{=} \mathbb{P}_{i\sim\mathcal{Y},\, l\sim\mathcal{L}_i}\left(\mathcal{N}(D_i(l)) = i\right).$$

Definition 2 The **latent reconstruction accuracy** (LRA) of $\mathcal{N}$ is:

$$\mathrm{LRA}(\mathcal{N}) \stackrel{\text{def}}{=} \mathbb{P}_{(x,i)\sim\mathcal{J}}\left(\mathcal{N}(D_i(E_i(x))) = i\right).$$

In LGA, which requires D_i but not E_i, compared to regular accuracy on the holdout set, we have replaced real data samples with generated samples, following class probabilities in $\mathcal{J}$ (also note that it is possible to consider similar metrics for each class separately). As a result, an unlimited number of samples can be used to estimate LGA. In addition, misclassified samples found during the check of this specification can be used to train $\mathcal{N}$ further. In LRA, instead of generating new samples, we take the approximations of real ones computed with both D_i and E_i. This resembles the Defense-GAN (Samangouei et al., 2018) approach.

While LGA can be measured by sampling latent codes, LRA can be estimated based on samples from the holdout set (see Sect. 4.2). The main purpose of LGA and LRA in this paper is to serve as baselines for other metrics proposed in the following subsections, which, in addition to generation or reconstruction, assume the presence of an adversary.

3.5 Noise corruption robustness in the latent space

In this subsection, we consider a randomized noise-adding adversary. Suppose that $N_{\epsilon,l}$ is some *noise-adding distribution* that operates on latent vectors l, where parameter $\epsilon \geq 0$ controls the magnitude of the noise. Below, we will use the same notation for the probability density function (PDF) of this distribution. We would like the following conditions to be satisfied:

1. *Distribution preservation*: for all ϵ, sampling $l' \sim N_{\epsilon,l}$ with $l \sim \mathcal{L}_i$ is equivalent to sampling $l' \sim \mathcal{L}_i$. This condition ensures the "naturality" of noise: its addition does not shift the distribution of input vectors, meaning that it will not produce vectors that are not plausible according to $\mathcal{L}_i$ (compared, e.g., with addition of noise to each component of the original data item).
2. *Support of small noise*: if $\epsilon \to 0$, random vectors $\lambda_\epsilon \sim N_{\epsilon,l}$ converge in distribution to $\lambda \equiv l$, i.e., the added noise becomes negligible. This condition ensures that small ϵ corresponds to small noise.
3. *Support of large noise*: if $\epsilon \to +\infty$, random vectors $\lambda_\epsilon \sim N_{\epsilon,l}$ converge in distribution to $\lambda \sim \mathcal{L}_i$, i.e., the unperturbed latent vector l becomes irrelevant. This condition ensures that large ϵ corresponds to large noise. Convergence to $\mathcal{L}_i$ is needed to comply with the first condition.
4. *Connection with perturbation magnitude*: there exists a distance v and a continuous, strictly decreasing function q_ϵ such that the likelihood of the noise is given by $N_{\epsilon,l}(l') = q_\epsilon(v(l', l))$. This requirement is needed to make the magnitude of perturbations measurable by their likelihood. In addition, it guarantees that $N_{\epsilon,l}$ has an upper bound $q_\epsilon(0)$.

We will propose a concrete family of distributions satisfying these properties in Sect. 4.3. Now, we look at the case where the input to be classified is a perturbed version of the reconstruction of a real data element:

Definition 3 The **local latent noise accuracy** (LLNA) of $\mathcal{N}$ in point $x \in \mathbb{R}^{n_I}$ of known class i with noise magnitude ϵ is:

$$\mathrm{LLNA}(\mathcal{N}, \epsilon, x, i) \overset{\text{def}}{=} \mathbb{P}_{l \sim N_{\epsilon, E_i(x)}} \left(\mathcal{N}(D_i(l)) = i\right).$$

LLNA is similar to LRA, except that checks are performed on noisy reconstructions of a fixed real data sample x. LLNA can be evaluated based on sampling noise vectors (see Sect. 4.3).

3.6 Adversarial robustness in the latent space

Next, instead of checking the classifier's resistance to random noise, we consider perturbations chosen by an adversary. In terms of $N_{\epsilon,l}$, we can assume that the adversary can choose the worst case input within bounded likelihood. Given fixed x and i, $l' \sim N_{\epsilon,E_i(x)}$ is a random n_L-dimensional vector. Then:

Definition 4 The **local latent adversarial robustness** (LLAR) of $\mathcal{N}$ in point $x \in \mathbb{R}^{n_I}$ with known class i, with noise magnitude ϵ, is:

$$\mathrm{LLAR}(\mathcal{N}, \epsilon, E_i(x), i) \overset{\text{def}}{=} \max\{N_{\epsilon, E_i(x)}(l') \mid l' \in \mathbb{R}^{n_L} \, and \, \mathcal{N}(D_i(l')) \neq i\}.$$

This defines LLAR as the maximum likelihood of a latent adversarial perturbation, with low LLAR corresponding to high robustness. Condition 4 on the noise distribution makes the value of LLAR correspond to some value of a distance between the original and the perturbed vectors, giving it an intuitive interpretation. Also, due to the boundedness of $N_{\epsilon,E_i(x)}$, which follows from the same condition, this expression may not give positive infinity. Negative infinity may be obtained in a peculiar case of all the latent space being classified into class i, which, for example, may be caused by the trained models being inadequate. Still, our definitions below tolerate this case.

LLAR captures proximity in the latent space and is similar to known definitions of local robustness checked in the input space of the ANN (Anderson et al., 2019; Bastani et al., 2016; Fawzi et al., 2018; Huang et al., 2017; Katz et al., 2017; Singh et al., 2019), for example, to pointwise robustness (Bastani et al., 2016). However, the likelihood τ of a multivariate random vector may be inconvenient to operate with, and thus we allow it to be post-processed with some decreasing function $g_\epsilon(\tau)$. In Sect. 4.7, we will propose an approach that views LLAR as ℓ_2 robustness in the latent space (i.e., g_ϵ will convert the likelihood to this norm) and either finds its approximate value or checks whether it is above a given threshold.

Next, we transform LLAR to global performance metrics, returning to the ideas of sampling latent vectors and looping through reconstructed data items:

Definition 5 The **latent adversarial generation severity** (LAGS) of $\mathcal{N}$ with noise magnitude ϵ is:

$$\mathrm{LAGS}(\mathcal{N}, g_\epsilon, \epsilon) \overset{\mathrm{def}}{=} \mathbb{E}_{i \sim \mathcal{Y},\, l \sim \mathcal{L}_i}(g_\epsilon(\mathrm{LLAR}(\mathcal{N}, \epsilon, l, i))).$$

Definition 6 The **latent adversarial reconstruction severity** (LARS) of $\mathcal{N}$ with noise magnitude ϵ is:

$$\mathrm{LARS}(\mathcal{N}, g_\epsilon, \epsilon) \overset{\mathrm{def}}{=} \mathbb{E}_{(x,i) \sim \mathcal{J}}(g_\epsilon(\mathrm{LLAR}(\mathcal{N}, \epsilon, E_i(x), i))).$$

Definition 7 The **latent adversarial generation accuracy** (LAGA) of $\mathcal{N}$ with noise magnitude ϵ and bound ρ on its transformed likelihood is:

$$\mathrm{LAGA}(\mathcal{N}, g_\epsilon, \rho, \epsilon) \overset{\mathrm{def}}{=} \mathbb{P}_{i \sim \mathcal{Y},\, l \sim \mathcal{L}_i}\left(g_\epsilon(\mathrm{LLAR}(\mathcal{N}, \epsilon, l, i)) > \rho\right).$$

Definition 8 The **latent adversarial reconstruction accuracy** (LARA) of $\mathcal{N}$ with noise magnitude ϵ and bound ρ on its transformed likelihood is:

$$\mathrm{LARA}(\mathcal{N}, g_\epsilon, \rho, \epsilon) \overset{\mathrm{def}}{=} \mathbb{P}_{(x,i) \sim \mathcal{J}}\left(g_\epsilon(\mathrm{LLAR}(\mathcal{N}, \epsilon, E_i(x), i)) > \rho\right).$$

LAGS and LARS are similar to adversarial severity as defined by Bastani et al. (2016), and LAGA and LARA are similar to adversarial frequency as defined by the same authors. Intuitively, LAGS and LARS are average LLAR values, while LAGA and LARA are average success rates of passing a specification of being resistant to sufficiently likely latent perturbations. In Sect. 4.7, we will approximately evaluate all these metrics with sampling and PGD. The overview of all considered latent space performance metrics is given in Table 1.

4 Evaluating latent space performance metrics

This section proposes concrete approaches to calculate the values of the metrics defined in Sect. 3. The general idea is to work with the standard multivariate Gaussian distribution as the latent one due to its well-known properties. This is especially important for addressing latent adversarial robustness in Sect. 4.7.

Table 1 Overview of the proposed latent space performance metrics

Metric	Needs E_i	Adversary	Range
Latent generation accuracy (LGA)	No	No	[0, 1]
Latent reconstruction accuracy (LRA)	Yes	No	[0, 1]
Local latent noise accuracy (LLNA)	Yes	Noise	[0, 1]
Local latent adversarial robustness (LLAR)	Yes	PGD	$\mathbb{R}^+$
Latent adversarial generation accuracy (LAGA)	No	PGD	[0, 1]
Latent adversarial generation severity (LAGS)	No	PGD	$\mathbb{R}$[a]
Latent adversarial reconstruction accuracy (LARA)	Yes	PGD	[0, 1]
Latent adversarial reconstruction severity (LARS)	Yes	PGD	$\mathbb{R}$[a]

[a] May be more restricted depending on the choice of g_ϵ

4.1 Choice of generative models

To be able to work with probability densities in the latent spaces $\mathcal{L}_i$, we need to fix the selection of these spaces. We achieve this by taking $\mathcal{L}_i = N(0, I)$. Then, to evaluate all metrics proposed in Sect. 3, transformations D_i and, for reconstruction-based metrics, E_i must be defined for all classes $1 \le i \le m$. The following techniques can be applied:

1. For each i, train a generative autoencoder $(\mathcal{N}_i^E, \mathcal{N}_i^D)$ and take $E_i(x) \overset{\text{def}}{=} \mathcal{N}_i^E(x)$, $D_i(l) \overset{\text{def}}{=} \mathcal{N}_i^D(l)$.
2. For each i, train a GAN with generator $\mathcal{G}_i$ and take $D_i(x) \overset{\text{def}}{=} \mathcal{G}_i(x)$. E_i can be obtained by enhancing these GANs with encoding procedures, e.g., by training inverters (Hendrycks et al., 2019), performing gradient-based optimization of latent codes, or both (Bau et al., 2019). Instead of training models for each class separately, it is possible to train class-conditional models (Odena et al., 2017).

4.2 Measuring latent accuracy

With D_i and E_i defined, **LGA can be measured** by repeatedly sampling a class label $i \sim \mathcal{Y}$ and a latent code $l \sim N(0, I)$, calculating $o_g = [\mathcal{N}(D_i(l)) = i]$,[1] which is a Bernoulli random variable, and averaging the obtained values of o_g, which gives an unbiased estimate of LGA. Similarly, **LRA can be measured** by sampling validation data items (x, i) and averaging $o_r = [\mathcal{N}(D_i(E_i(x)) = i]$.

4.3 Noise model and measuring local latent noise accuracy

Suppose that we sample $(x, i) \sim \mathcal{J}$ by enumerating over $(x_1, y_1), ..., (x_k, y_k)$. In this case $l = E_i(x) \sim N(0, I)$. At this point, we can inject a random perturbation into the latent code. We define the noise-adding distribution $N_{\epsilon,l}$ as follows:

[1] $[x]$ (Iverson bracket) is 1 if x is true, and 0 if x is false.

$$l' \sim N_{\epsilon,l} \overset{\text{def}}{\Leftrightarrow} l' = \frac{l + \epsilon \cdot \delta l}{\sqrt{1+\epsilon^2}} \text{ with } \delta l \sim N(0, I). \tag{1}$$

Note that, given the previous choice $\mathcal{L}_i = N(0, I)$, this choice of $N_{\epsilon,l}$ complies with the constraints stated in Sect. 3.2 (point 1 is easy to check, the proofs of points 2 and 3 are given in Appendix B, and point 4 will be clarified in Sect. 4.4), and it would not be distribution-preserving either (1) with a non-Gaussian δl, or (2) without the denominator $\sqrt{1+\epsilon^2}$. Furthermore, the definition (1) is equivalent to:

$$N_{\epsilon,l} = N\left(\frac{l}{\sqrt{1+\epsilon^2}}, \frac{\epsilon^2}{1+\epsilon^2} I\right). \tag{2}$$

LLNA can be measured by finding the latent vector $l = E_i(x)$, then repeatedly sampling $l' \sim N_{\epsilon,l}$ and calculating $o_n = [\mathcal{N}(l') = i]$, which is again a Bernoulli random variable. The rest is similar to checking LGA and LRA.

4.4 Likelihood of perturbations and perturbed vectors

In the rest of this section, to check LLAR and its derivatives, we will optimize *latent perturbations*—adversarially chosen perturbations that are bounded by the likelihood of the outcomes of $N_{\epsilon,l}$. They are similar to the ones considered in (Zhao et al., 2017). Noise addition $N_{\epsilon,l}$ (2) can be interpreted as a composition of two transformations:

1. *Decay* (reduction) of the unperturbed latent vector l by $\sqrt{1+\epsilon^2}$.
2. Addition of Gaussian noise $\Delta l \sim N\big(0, \epsilon^2 I/(1+\epsilon^2)\big)$.

Below, we will refer to Δl as a *latent adversarial perturbation* rather than noise, emphasizing that Δl will be produced with directed search rather than sampling. What perturbations Δl are more likely? The log-likelihood of Δl having a standard Gaussian distribution is determined by the ℓ_2 norm of Δl:

$$\begin{aligned}\log f_{N(0,\epsilon^2 I/(1+\epsilon^2))}(\Delta l) &= \log \prod_{j=1}^{n_L} \sqrt{\frac{1+\epsilon^2}{2\pi\epsilon^2}} \exp\left(-\frac{1+\epsilon^2}{2\epsilon^2}\Delta l_j^2\right) \\ &= n_L \log \sqrt{\frac{1+\epsilon^2}{2\pi\epsilon^2}} - \frac{1+\epsilon^2}{2\epsilon^2}\sum_{j=1}^{n_L} \Delta l_j^2 = c_1(\epsilon) - c_2(\epsilon)\|\Delta l\|_2^2.\end{aligned} \tag{3}$$

The distribution of the perturbed vector $l' = l/\sqrt{1+\epsilon^2} + \Delta l$, which is of interest in the definition of LLAR, differs from the one of Δl only by its mean, and thus its log-likelihood as a function of Δl is the same. Also, (3) shows that the condition 4 (Sect. 3.5) on the noise distribution is satisfied with ν being the Euclidean distance.

4.5 Optimization problem for bounded latent perturbation search

To measure LAGA and LARA (Sect. 3.6), it is sufficient to check whether LLAR at the current latent point is bounded with a defined likelihood τ (according to the noise model from Sect. 4.3): that is, any perturbation whose likelihood is at least τ, is class-preserving. According to Eq. 3, each positive value τ uniquely corresponds to a particular value of the

ℓ_2 norm of the perturbation Δl around $l/\sqrt{1+\epsilon^2}$. For convenience, we will measure perturbation likelihood with its *scaled norm* $\|\cdot\|_2^s = \|\cdot\|_2/\sqrt{n_L}$. With this scaling, the expected squared scaled norm of a multidimensional vector distributed according to $N(0, I)$ is one. The following function transforms the likelihood of Δl to $\|\Delta l\|_2^s$:

$$g_\epsilon(\tau) = \sqrt{\frac{c_1(\epsilon) - \log\tau}{n_L \cdot c_2(\epsilon)}}.$$

We also introduce the following auxiliary definitions:

- l_0 is the initial latent vector, where a LLAR specification should be checked. It corresponds to some input vector x with its available label i: $l_0 = E_i(x)$.
- The *decay factor* $d = 1 - 1/\sqrt{1+\epsilon^2}$ $(0 \le d \le 1)$ is the amount of reducing the vector l prior to the search of a perturbation.
- $l_1 = (1-d)l_0 = l_0/\sqrt{1+\epsilon^2}$ is the reduced vector, which is the mean of the perturbation Δl.

Thus, we need to check whether there is an adversarial perturbation Δl with $\|\Delta l\|_2^s \le \rho$, where $\rho = g_\epsilon(\tau)$, that makes the classifier $\mathcal{N}$ classify $D_i(l_1 + \Delta l)$ as not belonging to class i. Suppose that an *objective function* $\mathcal{O} : \mathbb{R}^{n_L} \to \mathbb{R}$ is available such that $\mathcal{O}(\Delta l) > 0$ implies correct classification and $\mathcal{O}(\Delta l) < 0$ implies misclassification. We take

$$\mathcal{O}(\Delta l) = s(i) - \max_{1\le j\le m, j\ne i} s(j), \text{ where } s(j) = S_{\mathcal{N}}(D_i(l_1 + \Delta l), j).$$

It is almost everywhere differentiable due to the corresponding assumptions on $S_{\mathcal{N}}$ and D_i. Then we can solve the following constrained optimization problem with gradient-based techniques:

$$\text{minimize}_{\Delta l:\ \|\Delta l\|_2^s \le \rho} \mathcal{O}(\Delta l). \tag{4}$$

4.6 Intuition for non-zero decay factor

At first glance, viewing latent perturbations as a perturbation of l_1 but not l_0 (which equals l_1 only in the case of zero noise) may be confusing. The intuitive explanation, on the other hand, is in line with the purpose of division by $\sqrt{1+\epsilon^2}$ in (1), which is needed to reduce the covariance matrix of the distribution of perturbed vectors (with unperturbed vectors $l \sim N(0, I)$) back to I. Decay moves the search region to the area of more likely (having a smaller norm) vectors. Again, we remind that the likelihood in $\mathcal{L}_i$ in the general case does not correspond to the likelihood in $\mathcal{X}_i$. Still, in our experiments, decay moves latent vectors towards "averaged" representatives of each class.

4.7 Latent perturbation search with PGD

The constrained problem (4), considered for an approximation $E_y(x)$ of a data element (x, y), corresponds to **checking a threshold specification on LLAR**. Our proposed untargeted attack that solves this problem is a variant of PGD (Madry et al., 2018). PGD is started from a random latent perturbation within the allowed ρ-ball and is run until a

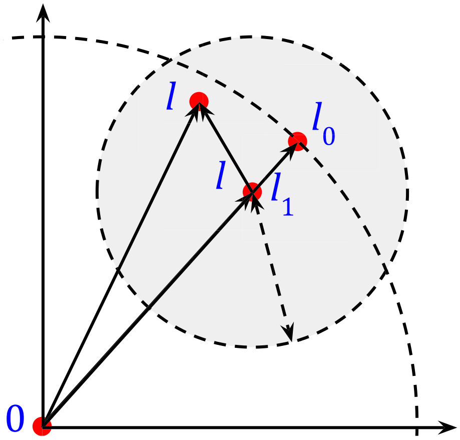


Fig. 2 Graphical interpretation of latent perturbation search with PGD. The grey circle is the region where the adversarial perturbation Δl is searched, and l is the current candidate solution

misclassification is achieved, i.e., $\mathcal{O}(\Delta l) < 0$, but no longer than for a predetermined number of steps. The learning rate is set to ensure that the boundary of the ρ-ball can be reached from any point inside it. To avoid exploding or vanishing gradients, as in (Madry et al., 2018), we scale $g = \nabla \mathcal{O}(\Delta l)$ with its ℓ_2 norm (specifically, we divide g by $\|g\|_2^s$). The optimization procedure is illustrated in Fig. 2.

Next, we consider evaluation of performance metrics that are based on LLAR. The process is similar to evaluation of conventional adversarial robustness, with the only essential differences being the search of perturbations in the latent space instead of the original one and the replacement of the original image with its decayed version. Accordingly, **evaluation of LAGA and LARA** differs from the one of LGA and LRA by using the PGD adversary with the corresponding ρ to alter the generated or approximated image prior to submitting it to the classifier. To increase reliability, PGD should be run multiple times. To **evaluate LAGS and LARS**, minimum perturbation bounds ρ need to be calculated and averaged. To approximately find the minimum norm ρ of a class-changing perturbation without pre-setting it, we apply the following techniques:

- Set ρ to a large value (we use $\rho = 2.5$) and start PGD with a small learning rate at $\Delta l = 0$. It will reach some solution, whose norm could be used as an approximation for minimum ρ.
- The solution above might be prone to reaching local optima, which can be mitigated by several restarts from different points. In this case, to enforce norm minimization, each new restart is done with ρ set to the scaled norm of the previously found solution, and the learning rate is reduced proportionally to the shrinkage of ρ.

To evaluate LAGS and LARS, we also tried DeepFool (Moosavi-Dezfooli et al., 2016), which is an algorithm to find minimum ℓ_p adversarial perturbations. Essentially, it is a variant of gradient descent with specifically chosen step magnitudes that are intended for fast convergence to a perturbation lying on the decision boundary of the classifier. Unfortunately, we observed its frequent divergence on our optimization problem. Gradient clipping resumed convergence, although it often cannot be achieved in just a few steps as in (Moosavi-Dezfooli et al., 2016). Thus, for the lack of apparent benefits of using DeepFool, in our experiments we apply only PGD.

5 Experimental evaluation

5.1 Implementation and experimental setup

The proposed framework of evaluating feed-forward ANN classifier performance with generative models was implemented in Python with PyTorch. The code and models used to obtain the results described in this section are publicly available online.[2] As the case studies, we considered the following image classification problems:

1. **MNIST** (LeCun, 1998) digit classification ($m = 10$ classes). As generative models, for MNIST, we trained a WGAN (Arjovsky et al., 2017) with $n_L = 64$ for each class, and implemented E_i with gradient descent (Adam with 4 restarts) over latent codes. Examples of images reconstructed and generated by these models are given in Fig. 6 (top).
2. Gender predictions based on face photos, using the **CelebA** (Liu et al., 2015) dataset ($m = 2$ classes: 1 = "female", 2 = "male"; images were center-cropped and resized to 128×128 pixels). For CelebA, we trained PIONEER (Heljakka et al., 2018, 2020) generative autoencoders for each dataset and class with $n_L = 511$. Examples of images produced by the models are given in Fig. 6 (middle)—note that the visual quality of reconstructed images is somewhat better compared to generated images.
3. Scene type prediction using the **LSUN** (Yu et al., 2015) dataset ($m = 2$ classes: 1 = "bedroom", 2 = "church outdoor"; images were center-cropped and resized to 128× 128 pixels). For LSUN scene types, we also trained PIONEER models with $n_L = 511$. However, as seen from Fig. 6 (bottom), except for bedroom reconstructions, the visual quality of images produced by PIONEER models for LSUN is worse compared to CelebA images.

For each of these classification problems, we trained fifteen deep CNN classifiers (see Appendix C for details) divided into five groups with three classifiers in each:

1. $\mathcal{N}_{\mathrm{UT}}$ ("undertrained"): classifiers trained in a usual way, without data augmentation, but only for one epoch (to intentionally achieve lower accuracy);
2. $\mathcal{N}_{\mathrm{NR}}$ ("non-robust"): the same as above, but trained for several epochs;
3. $\mathcal{N}_{\mathrm{CA}}$ ("conventional augmentation"): classifiers trained in a usual way, with conventional data augmentation;
4. $\mathcal{N}_{\mathrm{R}}$ ("robust"): classifiers trained on images corrupted with visible Gaussian noise (Gilmer et al., 2019);[3]
5. $\mathcal{N}_{\mathrm{B}}$ ("both"): classifiers trained with both conventional data augmentation and noise corruption.

In addition, a limited evaluation on several pretrained classifiers was performed on **ImageNet** (Russakovsky et al., 2015). The details of these experiments will be reported separately in Sect. 5.4. Finally, for all the considered generative models, we report the values of their reconstruction and generation performance metrics in Table 2.

[2] https://github.com/igor-buzhinsky/latent-space-nn-evaluation

[3] This form of training was used instead of more common robust optimization with PGD to reduce computation time.

Table 2 Performance metrics of used generative models

Model kind	Dataset	Class	$\lVert x - x_0 \rVert_2^s$	FID
WGAN	MNIST	All (10)	0.274	19.06
PIONEER	CelebA	Female	0.175	16.10
		Male	0.205	13.28
	LSUN	Bedroom	0.217	24.80
		Church outdoor	0.234	67.38
BigGAN	ImageNet	All (1000)	–	13.29

As reconstruction performance, we report the average scaled norm $\lVert x - x_0 \rVert_2^s$ of the difference between the original and the reconstructed images in the original space. Reconstruction was not considered for ImageNet. As generation performance, we report the Fréchet Inception distance (FID) (Heusel et al., 2017) computed on 25 thousand images (28×28 for MNIST, 128×128 for other datasets)

5.2 Performance evaluation using original space metrics

The performance metrics of the above deep CNN classifiers in the original space are reported in Table 3. From this table, it is visible that, as expected, training with Gaussian noise achieved not only noise corruption robustness but also adversarial robustness, and the latter two are associated. In addition, a trade-off is visible between the accuracy of the classifiers on clean images (hereinafter, *clean accuracy*) and adversarial robustness, which is in agreement with previous observations (Tsipras et al., 2018).

5.3 Performance evaluation using the proposed latent space metrics

We calculated the values of the proposed latent space performance metrics for all aforementioned classifiers. The corresponding results are provided in Table 4 and Fig. 3. We start interpreting these results from LGA and LRA, which can be regarded as quality measures of generation and reconstruction capabilities of generative models that are complementary to the ones reported in Table 2. For CelebA and LSUN, in Fig. 3, plots 1 and 4, it is visible that clean accuracy is correlated with both LGA and LRA. The stronger correlation of LRA and clean accuracy can be explained by better reconstruction capabilities of our PIONEER models compared to their generation capabilities. On MNIST, the associations of clean accuracy with LGA and LRA are roughly the same (Pearson's $r = 0.5$). Based on these observations, we conclude that the used generative models are suitable for evaluation of other proposed metrics.

Next, we comment on LLNA, which is a local metric, unlike the others. We computed its values on particular images and show several noise-based perturbations used in these computations in Fig. 4. Noise addition appeared to be a very sample-inefficient adversary, but the values of LLNA can be treated as prediction stability measures. For example, for the reconstructed (second) image in the second row of Fig. 4, the prediction of $\mathcal{N}_{\mathrm{NR}}$ is incorrect, and this also reflects in low accuracy of perturbed images (e.g., for $\epsilon = 0.5$, the LNNA on this image is 82.5%). The same image is also somewhat difficult for $\mathcal{N}_{\mathrm{R}}$ (for $\epsilon = 0.5$, LLNA = 92.0%).

Table 3 Performance metrics of considered CNN classifiers measured in the original space

Dataset	Classifier	Accuracy		Adversarial severity	
		Clean	Noise	$\|\Delta x\|_2^s$	$\frac{1}{n_I}\|\Delta x\|_\infty$
MNIST	$\mathcal{N}_{\mathrm{UT}}$	98.0%	78.9%	0.0680	0.1744
	$\mathcal{N}_{\mathrm{NR}}$	**99.1%**	80.0%	0.0747	0.1911
	$\mathcal{N}_{\mathrm{CA}}$	98.8%	94.5%	0.1152	0.3093
	$\mathcal{N}_{\mathrm{R}}$	98.9%	**98.2%**	0.1666	**0.5130**
	$\mathcal{N}_{\mathrm{B}}$	98.3%	97.6%	**0.1696**	0.4908
CelebA	$\mathcal{N}_{\mathrm{UT}}$	94.3%	65.3%	0.0023	0.0059
	$\mathcal{N}_{\mathrm{NR}}$	**97.6%**	63.0%	0.0028	0.0071
	$\mathcal{N}_{\mathrm{CA}}$	96.3%	49.8%	0.0029	0.0079
	$\mathcal{N}_{\mathrm{R}}$	96.2%	**95.2%**	0.0118	0.0296
	$\mathcal{N}_{\mathrm{B}}$	94.8%	94.4%	**0.0128**	**0.0333**
LSUN	$\mathcal{N}_{\mathrm{UT}}$	93.9%	63.8%	0.0017	0.0045
	$\mathcal{N}_{\mathrm{NR}}$	**98.4%**	50.2%	0.0023	0.0054
	$\mathcal{N}_{\mathrm{CA}}$	97.4%	52.7%	0.0036	0.0079
	$\mathcal{N}_{\mathrm{R}}$	92.8%	**95.8%**	0.0140	**0.0321**
	$\mathcal{N}_{\mathrm{B}}$	93.6%	93.8%	**0.0149**	**0.0321**

Accuracy was measured on the validation set of each dataset. For noise accuracy, we report accuracy on images corrupted with standard Gaussian noise with $\sigma = 0.8$. Adversarial severity (Bastani et al., 2016) is reported for ℓ_2 and ℓ_∞ norms scaled by dividing by $\sqrt{n_I}$ and n_I respectively. It was estimated on 600 images per classifier, and all the values were averaged over three classifiers in each group. Adversarial perturbations were searched with PGD: for each image, 15 runs were performed with norm threshold shrinkage as explained at the end of Sect. 4.7, except for doing this in the original space. For each PGD run, we used 50 steps of magnitude 0.05ρ from a random point, where ρ is the current norm threshold. For each value series, the best (largest) value is shown in bold

The following findings, which are more prominent, are related to metrics that evaluate adversarial robustness in latent spaces:

1. We found **association between clean accuracy and latent adversarial robustness** measured as LAGS, LAGA, LARS, and LARA—see Fig. 3, plots 2–3 and 5–6. In addition, distribution plots of approximately minimum perturbations found with PGD that were used in computing LAGS and LARS are given in Fig. 10. For LARS, examples of such perturbations are shown in Figs. 5, 7 and 8 . This finding implies that **latent space perturbations may be valuable in training ANN classifiers further**.
2. The **results regarding the association of the traditional and conventional adversarial robustness are inconclusive**. While the measured values of conventional adversarial severity have a small correlation with LARS and LAGS, they have a negative correlation with LARA and LAGA—the corresponding plots are given in Fig. 3, plots 8–9 and 11–12. This pattern is similar regardless of the dataset, ϵ or the choice of the norm to evaluate conventional adversarial severity. This outcome might have been caused by the difference in PGD search strategies used to evaluate these two kinds of metrics. In addition, these correlations are not required to be identical as the precise values of both these metrics are different statistics of the real minimum norms of adversarial perturbations (mean for LARS and LAGA and percentile rank for LARA and LAGA). Overall,

Table 4 Latent space performance metrics of considered CNN classifiers

Classifier	LGA	LRA	$\epsilon = 0.5$ ($d = 0.106$)				$\epsilon = 1.0$ ($d = 0.293$)			
			LAGS	LARS	LAGA	LARA	LAGS	LARS	LAGA	LARA
MNIST										
$\mathcal{N}_{\mathrm{UT}}$	97.9%	98.4%	0.297	0.302	39.6%	40.6%	0.320	0.332	43.9%	47.1%
$\mathcal{N}_{\mathrm{NR}}$	**98.6%**	**99.2%**	0.327	0.339	**45.4%**	**47.3%**	0.347	0.359	**50.4%**	**53.2%**
$\mathcal{N}_{\mathrm{CA}}$	98.3%	98.7%	**0.341**	**0.347**	43.8%	45.4%	**0.361**	**0.372**	46.7%	49.9%
$\mathcal{N}_{\mathrm{R}}$	98.2%	98.8%	0.325	0.336	40.7%	43.4%	0.344	0.355	44.9%	47.1%
$\mathcal{N}_{\mathrm{B}}$	98.2%	98.4%	0.339	0.342	37.9%	40.3%	0.346	0.360	41.0%	43.3%
CelebA										
$\mathcal{N}_{\mathrm{UT}}$	95.1%	95.1%	0.050	0.070	10.0%	22.0%	0.049	0.065	9.1%	16.9%
$\mathcal{N}_{\mathrm{NR}}$	**98.3%**	**98.4%**	**0.067**	**0.095**	16.7%	**33.8%**	**0.066**	**0.088**	12.9%	**26.3%**
$\mathcal{N}_{\mathrm{CA}}$	97.4%	97.4%	0.064	0.090	**17.1%**	30.4%	0.062	0.084	**13.1%**	25.4%
$\mathcal{N}_{\mathrm{R}}$	96.5%	96.2%	0.061	0.088	9.3%	26.6%	0.060	0.079	5.3%	18.8%
$\mathcal{N}_{\mathrm{B}}$	95.3%	94.7%	0.057	0.082	7.6%	24.2%	0.056	0.074	4.4%	16.3%
LSUN										
$\mathcal{N}_{\mathrm{UT}}$	96.6%	92.0%	0.041	0.065	18.4%	31.7%	0.039	0.056	16.9%	27.2%
$\mathcal{N}_{\mathrm{NR}}$	**98.9%**	**96.6%**	0.056	0.098	24.4%	**44.4%**	0.051	0.086	21.9%	**39.5%**
$\mathcal{N}_{\mathrm{CA}}$	98.4%	**96.6%**	**0.063**	0.099	**26.2%**	41.5%	**0.060**	**0.088**	**27.7%**	37.2%
$\mathcal{N}_{\mathrm{R}}$	96.4%	89.9%	0.058	**0.103**	7.3%	37.6%	0.053	0.087	6.2%	29.4%
$\mathcal{N}_{\mathrm{B}}$	96.3%	92.1%	0.057	0.094	9.2%	32.2%	0.056	0.082	8.3%	22.7%

Accuracy and adversarial robustness computations were performed with 10000 and 600 images respectively. All the values were averaged over three classifiers in each group. LARA was measured with $\rho = 0.3$ on MNIST and $\rho = 0.1$ on CelebA and LSUN. For each value series, the best (largest) value is shown in bold

we did not find evidence that increasing conventional adversarial robustness increases latent adversarial robustness metrics, but it is not possible to conclude that they are not associated.

3. As visible from Figs. 5, 7, and 8, **latent adversarial perturbations are surprisingly small** on CelebA and LSUN and result in adversarial images that are perceptually close to the originals. Even though we do not have consistent results regarding the influence of conventional robustness on latent space robustness, the **adversarial images computed for robust classifiers are, on average, further away from the original ones** (despite having similar distances in the latent space). This distance was measured with ℓ_1 and ℓ_2 norms in the original space, and the increase of this distance is visible in Fig. 10, columns 3 and 4.

As for the validity of our study, the small size of the found latent space perturbations indicates that our proposed PGD-based untargeted attack is successful. At the same time, generated images require smaller latent space perturbations—this can be explained by lower quality of generated images, which makes classifiers less confident in their initial predictions. On the other hand, on MNIST, perturbations are very large (Fig. 10, two topmost plots in the first column), significantly raise the norm of the perturbed vector (Fig. 10, two topmost plots in the second column) and thus exploit the part of the latent space where

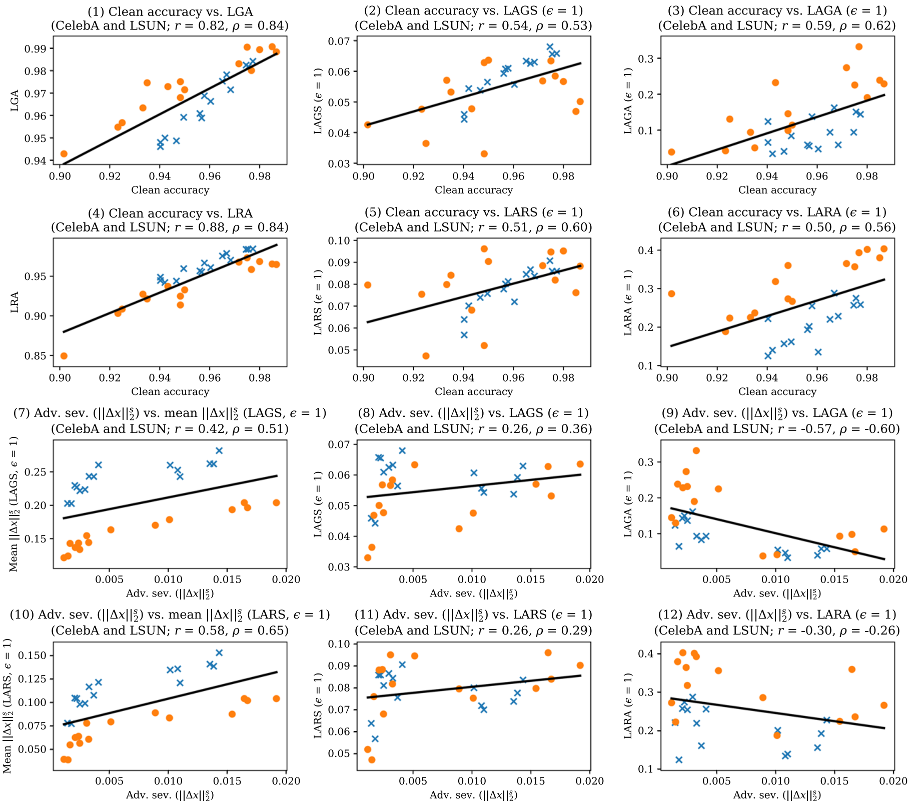


Fig. 3 Correlation plots for some of the data presented in Table 4 (plots 1–6, 8–9, 11–12) and additional data (plots 7 and 10). Plots are made for CelebA (blue crosses) and LSUN (orange circles) data combined (MNIST data is not shown). Plots 1 and 4 show the relation between clean accuracy, LRA and LGA. Plots 2–3 and 5–6 show an association between clean accuracy and latent adversarial robustness (measured as LAGS, LAGA, LARS, LARA). Plots 7 and 10 show an association between conventional adversarial robustness (measured as adversarial severity with respect to perturbations bounded by scaled ℓ_2 norm) and the averaged scaled ℓ_2 norm of found approximately minimum latent perturbations. Plots 8–9 and 11–12 demonstrate the inconclusive results regarding the association between conventional adversarial robustness and latent adversarial robustness. For each plot, Pearson's and Spearman's correlation coefficients (r and ρ, respectively) are given

the generative models were not trained to work. This can be explained by the simplicity of the MNIST classification problem.

Finally, we confirmed the meaning of decay in the latent space as a countermeasure against the increase of the norm of the latent vector by the adversary: as visible from Fig. 10, column 2, perturbed vectors typically exceed unperturbed vectors by norm. This phenomenon is explained by (1) the lower probability density of vectors with large latent space norms and the associated lack of classifier training on such less plausible input images, and (2) a higher ease to exploit a weakness of a generative model with the same sort of vectors. In particular, the second explanation applies to CelebA, where roughly half of approximately minimum latent space perturbations found with $\epsilon = 0.5$ ($d = 0.106$)

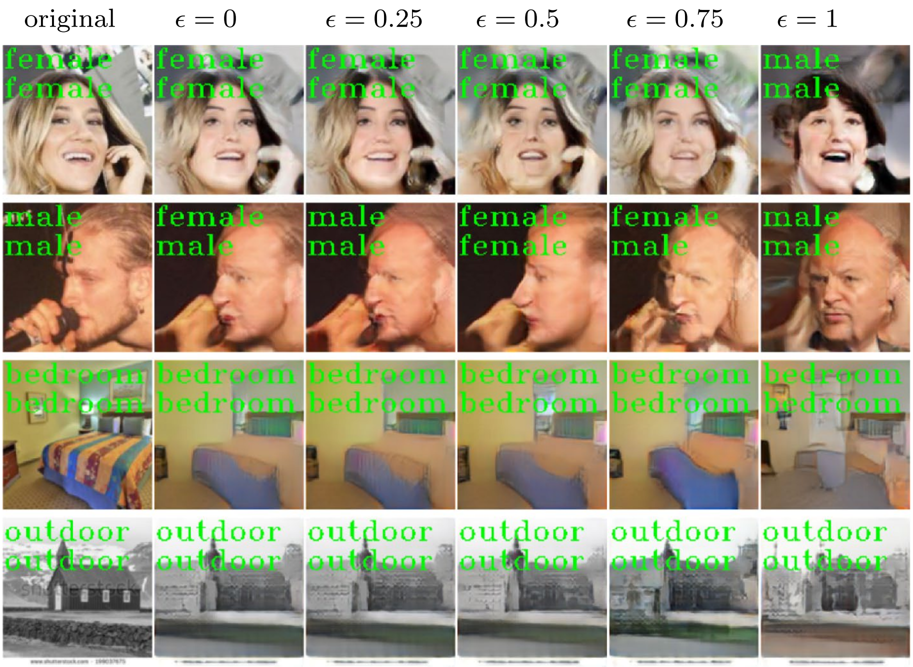


Fig. 4 Examples of perturbations for CelebA and LSUN images of each class (left: "female", "bedroom", right: "male", "church outdoor") that were generated as latent Gaussian noise. In each row: the original image, the image reconstructed by PIONEER ($\epsilon = 0$), then four perturbed reconstructed images with increasing noise magnitudes $\epsilon = 0.25, 0.5, 0.75, 1$. Green labels show classification outcomes of $\mathcal{N}_{\mathrm{NR}}$ (on the first line) and $\mathcal{N}_{\mathrm{R}}$ (on the second line). All images in this figure have resolution 128×128

contained visual artifacts, even though the likelihood of perturbed images in $\mathcal{L}_i$ was actually higher than the one of unperturbed images. With $\epsilon = 1$ ($d = 0.293$), the visual quality of perturbed images was higher. On the other hand, on all datasets, even with $\epsilon = 1$ ($d = 0.293$), decayed images were visually close to the originals (this is visible on Figs. 5, 7, and 8, columns 2–4).

5.4 Experiments on ImageNet

To demonstrate the possibility of evaluating the proposed metrics for larger classifiers and more challenging classification problems, we also considered ImageNet-1k image classification with a reduced experimental setup. As a generative model, we used a pretrained[4] class-conditional BigGAN (Brock et al., 2018) with $n_L = 128$ and image size of 128×128. The details of our experimental setup are:

1. We only computed latent generation metrics since we do not have a corresponding inverter model, and decoding the image with gradient-based search, like we did on MNIST, would have slowed the experiments significantly. To increase the visual quality of generated images, we sampled latent vectors with a built-in decay of 0.25.

[4] https://github.com/ajbrock/BigGAN-PyTorch

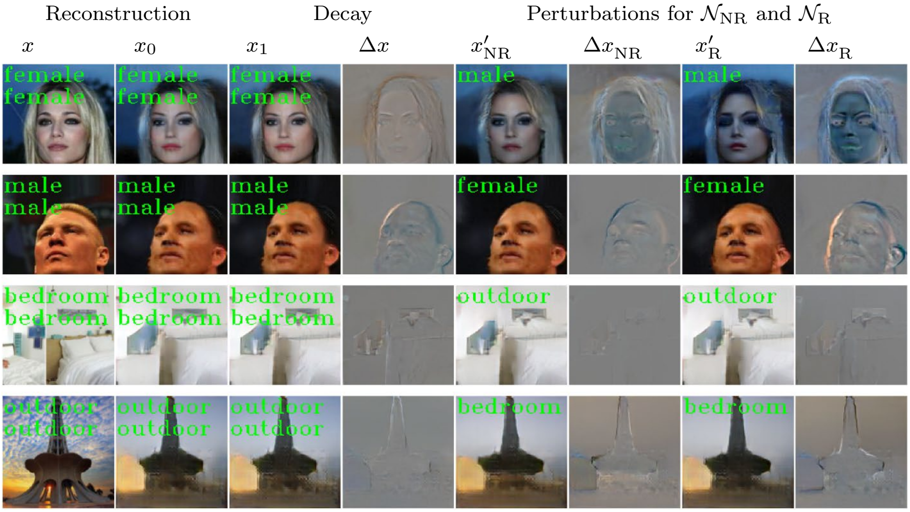


Fig. 5 Examples of approximately minimum latent CelebA and LSUN image perturbations with $\epsilon = 1$ ($d = 0.293$), each found with a single run of PGD from $\Delta l = 0$, for classifiers $\mathcal{N}_{\rm NR}$ and $\mathcal{N}_{\rm R}$. In each row, images are given in the following order: x, the real image (with classification outcomes of $\mathcal{N}_{\rm NR}$ and $\mathcal{N}_{\rm R}$ shown in green); $x_0 = D_i(l_0)$, the reconstructed image (with both classification outcomes); $x_1 = D_i(l_1)$, the decayed image (with both classification outcomes); $\Delta x = x_1 - x_0$, the difference between two previous images; $x'_{\rm NR} = D_i(l'_{\rm NR})$, the perturbed image for $\mathcal{N}_{\rm NR}$ (with the classification outcome of $\mathcal{N}_{\rm NR}$); $\Delta x_{\rm NR} = x'_{\rm NR} - x_1$, the perturbation for $\mathcal{N}_{\rm NR}$; $x'_{\rm R} = D_i(l'_{\rm R})$, the perturbed image for $\mathcal{N}_{\rm R}$ (with the classification outcome of $\mathcal{N}_{\rm R}$); $\Delta x_{\rm R} = x'_{\rm R} - x_1$, the perturbation for $\mathcal{N}_{\rm R}$. All images in this figure have resolution 128×128

2. For the same reason of computational complexity, we evaluate pretrained ImageNet classifiers and considered a smaller number of them. We took four non-robust models from PyTorch Model Zoo[5] and one robust classifier by Santurkar et al. (2019). All these classifiers accept input images of sizes 256×256 or 224×224, so we upsampled the generated images with bicubic interpolation.
3. Adversarial examples used to evaluate conventional adversarial severity were searched for in the 128×128 original space, i.e., the interpolation layer was regarded as the first layer of a classifier. This was done to make the results comparable between classifiers with different input sizes.

The results of experiments are shown in Table 5 and examples of approximately minimum adversarial perturbations are given in Fig. 9. Although the amount of data is small to give definite conclusions, we find a correlation between the accuracy and all latent space adversarial robustness metrics (Pearson's correlation coefficients $0.47 \le r \le 0.68$), and an even stronger correlation of LGA with these metrics ($0.79 \le r \le 0.97$). Unlike our previous findings, latent space adversarial robustness is now positively correlated with conventional adversarial robustness ($0.16 \le r \le 0.57$). However, while having notably larger conventional robustness, the robust classifier is not very different from others in terms of latent space adversarial robustness.

[5] https://pytorch.org/vision/stable/models.html

Table 5 Results of experiments on ImageNet

Metric		Squeezenet	Alexnet	Resnet	Resnext	Robust
Clean accuracy		50.2%	49.5%	64.2%	**72.2%**	52.2%
LGA		68.3%	72.1%	**77.9%**	76.1%	72.9%
LAGA	$\epsilon = 0.5$	23.0%	27.0%	**33.2%**	30.5%	29.7%
	$\epsilon = 1.0$	23.0%	32.7%	**39.3%**	38.7%	38.0%
Adversarial severity	$\|\Delta x\|_2^s$	0.00041	0.00075	0.00061	0.00080	**0.01101**
	$\frac{1}{n_I}\|\Delta x\|_\infty$	0.00076	0.00161	0.00123	0.00173	**0.02001**
LAGS	$\epsilon = 0.5$	0.04905	0.06104	0.07374	0.07001	**0.07725**
	$\epsilon = 1.0$	0.06238	0.07304	0.09365	0.09131	**0.09751**

In each row, the best (largest) value is shown in bold

The classifiers denoted as “squeezenet”, “alexnet”, “resnet”, “resnext” correspond to pretrained PyTorch models called “squeezenet1_0”, “alexnet”, “resnet18” and “resnext50_32×4d”. The “robust” classifier is the one reported by Santurkar et al. (2019). LAGA was computed with $\rho = 0.1$

5.5 Threats to validity

Below, we list the identified threats to the validity of our study and comment on them:

1. To keep the time required to perform the experiments manageable, most of them were done on CNN classifiers of small size (17—497 thousand trainable parameters), recognizing a small number of classes—it may appear that state-of-the-art classifiers have different patterns of latent space performance metric values. Our evaluation of ImageNet classifiers is limited and only demonstrates the possibility of applying the proposed methods to complex classification tasks. Yet, we have checked that (1) for the classifiers that we have studied, a connection between adversarial robustness and noise corruption robustness (Gilmer et al., 2019) exists, (2) on MNIST and CelebA, our robust classifiers have limited capabilities of image generation (Santurkar et al., 2019). On LSUN, we have seen that optimizing class activation of robust classifiers adds qualitatively different features to the image compared to non-robust classifiers, but we have not recognized the resulting images as bedrooms nor outdoors.
2. As we measure latent space adversarial robustness (LAGS, LARS, LAGA, LARA) with imprecise attack approaches, we overestimate the values of these metrics. This bias might have resulted in our classifiers ranked wrongly according to the computed values. PGD was shown to work well in the original space (Madry et al., 2018), but there is so far no similar set of experiments that confirm this property in latent spaces. We used PGD with 12 restarts to compensate for the possibility of such a bias. In certain cases (search of minimum adversarial perturbations on CelebA and LSUN), we used a single PGD run with a smaller learning rate, but in these cases, we had ensured that such runs differ insignificantly from the ones of PGD with restarts in terms of the resulting metric values.
3. On LSUN, the small size of the validation set (600 images) may have resulted in prematurely early stopping of training and imprecise accuracy estimates. In addition, the corresponding generative models produced random images with visible flaws. Yet, our observations for this dataset are not very different from the ones for CelebA, and perfect generative models might be hard to achieve on custom datasets.

4. PIONEER (CelebA and LSUN) models are designed to be trained to generate images from normalized latent vectors, and the latent distribution is actually the uniform distribution on the unit sphere instead of the Gaussian. In this paper, this has led to all reconstructed and generated images having unit scaled norm. Nonetheless, the decoder was capable of accepting unnormalized latent vectors, and decay still worked intuitively, i.e., by softening prominent features of images. This effect might have been caused by $N(0, I)$ and the uniform distribution on the unit sphere being very similar in multidimensional spaces: ℓ_2 norms of high-dimensional standard Gaussians are concentrated around $\sqrt{n_L}$.
5. As the adversarial examples considered in this paper are generated images, it is impossible to conclude that the classifiers actually make a mistake when classifying the generated adversarial examples differently. For example, the problem of determining the gender of a person who does not exist is not well-defined, but this also applies to other real-world object classification problems. Our definitions of metrics only require that the classification decision is changed, so this problem does not influence the soundness of these definitions. However, this also means that the values of the proposed metrics are not proven to be indicators of good classifier performance according to human judgment. A possible solution to address this problem would be to get manual labels of generated adversarial examples (Song et al., 2018b).
6. All the proposed latent space performance models rely on the generative models, and thus the computed values of these metrics depend on the quality of approximation of the original training/validation distribution and the choice of the generative model. In particular, due to a lower quality of the used generative models for LSUN (especially low generation quality for outdoors), our results for this dataset may be less reliable. Yet, we considered several datasets and kinds of generative models, and the correlation of the accuracy with latent space robustness metrics was found in each of these cases.

6 Related work

6.1 Adversarial examples in latent spaces

A number of works used generative models to create adversarial attacks and/or defenses. Zhao et al. (2017) proposed an approach to search adversarial examples in the latent space of a GAN, also measuring them with ℓ_2 norms. This approach is white-box and is based on directed sampling rather than gradient descent, which makes it applicable to discrete input data, such as in natural language processing tasks. By contrast, our techniques operate in a black-box setting and only with feed-forward ANNs accepting continuous data. Nonetheless, (1) being based on gradient descent, our latent perturbation search approach is able to find perceptually smaller perturbations compared to the ones presented by Zhao et al. (2017), (2) we consider a more general framework of transforming data to the latent space and back, (3) we connect latent adversarial robustness to a "natural" model of noise in the latent space and this way motivate the use of the ℓ_2 norm, (4) we search adversarial examples for larger classification tasks (128×128 instead of 64×64) and latent spaces (511 dimensions instead of 128), and (5) we focus on computing performance metrics for classifiers and not on finding adversarial examples per se.

Song et al. (2018b) created latent space adversarial examples from scratch. This was done using a class-conditional AC-GAN, and evaluation was in particular done on the CelebA (gender classification) and MNIST datasets. We also search for adversarial examples based on generated data items, however, (1) again, we do it for images larger than 64×64, (2) we consider the untargeted attack scenario and use a different approach to generate adversarial examples, (3) our approach is not restricted to AC-GANs, and (4) we focus on computing performance metrics rather than finding adversarial examples.

Mirman et al. (2020) developed an approach to formally verify properties of one-dimensional interpolations in the latent space of a generative model. The idea is to consider a set of images that correspond to a line segment in the latent space associated with a meaningful high-level change in the image, and determine whether adversarial examples exist on this line, or the probability of getting an adversarial example given a distribution on the input line segment. In our work, we search for adversarial examples in a multidimensional setting, without focusing on specific high-level features, and do this with imprecise approaches.

Generative models were used as defenses against adversarial attacks (Samangouei et al., 2018; Song et al., 2018a). For example, the Defense-GAN (Samangouei et al. 2018) approach protects image classifiers from adversarial attacks by replacing their input with an approximation in the latent space of a GAN (similarly to what is done when computing LRA in our work). This defense was broken by Jalal et al. (2019) with an optimization procedure in the latent space subject to a norm constraint in the original space. Our results are in line with this work, since we similarly approximate input images using a latent space of a generative model, and are able to find perceptually small perturbations that change the prediction of the classifier. Jalal et al. (2019) also proposed a defense approach based on the search of pairs of examples that are close in the latent space but are scored completely differently by the classifier, and subsequent augmentation of robust optimization with training on these pairs.

6.2 Robustness metrics for ANNs and their evaluation

Usually robustness of ANNs to adversarial attacks is measured relatively to a specific attack success. Yu et al. (2019) proposed an improvement over the default accuracy-based approach. By analyzing the decision surfaces of models, they note that robust models have smooth decision boundaries. The proposed metric reflects this by rewarding models with smooth decision surfaces.

The first authors to formalize the notion of adversarial robustness were Bastani et al. (2016), who proposed several metrics quantifying the network robustness, namely, pointwise robustness, adversarial frequency and adversarial severity (see Sect. 2.4). The authors compute the latter two through pointwise robustness, which is measured by approximation.

Exact pointwise robustness calculations was performed by Boopathy et al. (2019), although they refer to the measure as to the “lower bound on the image distortion.” Also, the notion of pointwise robustness was explored by Fawzi et al. (2018), who derived theoretical upper bounds for it. Weng et al. (2018) proposed an effective proxy measure of network robustness based on measuring Lipschitz constants, although it has received some criticism (Goodfellow, 2018). An alternative method to quantifying global robustness properties of networks was proposed by Gopinath et al. (2017), who developed a clustering algorithm that outputs a set of verified regions—a collection of hyperspheres where the network is guaranteed to produce the same label.

7 Discussion and conclusions

In this paper, we presented a framework to evaluate the performance of feed-forward ANN classifiers with the help of generative models. Within the framework, we proposed several performance metrics, the most interesting of which are related to measuring the robustness of classifiers to perturbations in latent spaces of these generative models. In addition, we presented techniques to evaluate these metrics for classifiers, including a novel PGD-based untargeted attack. The main motivation of our work is the property of generative models of mimicking the data distribution. This property implies that the adversarial perturbations that we consider result in natural data changes.

The proposed metrics allowed us to make several interesting observations regarding the performance of deep ANN classifiers on natural adversarial examples. We computed the values of these metrics on several CNN image classifiers and found an association between the accuracy of the classifiers on clean images and adversarial robustness in latent spaces. This implies that latent adversarial examples might be useful for further classifier training. We also did not reveal a notable impact of conventional adversarial robustness on its latent counterparts, except for the influence on the norms of latent adversarial perturbations in the original space.

A speculative explanation of the found connection between the accuracy and latent adversarial robustness is that the latter measures the vulnerability of the classifier to natural adversarial examples, while the accuracy measures the same for random natural examples. A similar interdependence of accuracy and robustness to natural adversarial examples of a different kind was experimentally found by Gu et al. (2019). An alternative explanation is based on the work by Gilmer et al. (2019), who demonstrated a connection between conventional adversarial robustness and robustness to corruption with Gaussian noise. When we move to latent spaces, the former becomes LARS/LARA, and the latter becomes the averaged version of LLNA, which, due to our noise model, is just LRA. In turn, for a generative model with good reconstruction quality, LRA is highly associated with accuracy. The finding of Gilmer et al. (2019) is exact for linear models and was shown to hold on CIFAR-10 and ImageNet nonlinear classifiers. In our case, we can imagine that the classifier accepts latent representations of class i, and is actually a composition of D_i and the original classifier $\mathcal{N}$. Unfortunately, the same properties were not confirmed to hold for classifiers of this kind, and hence this explanation is speculative as well.

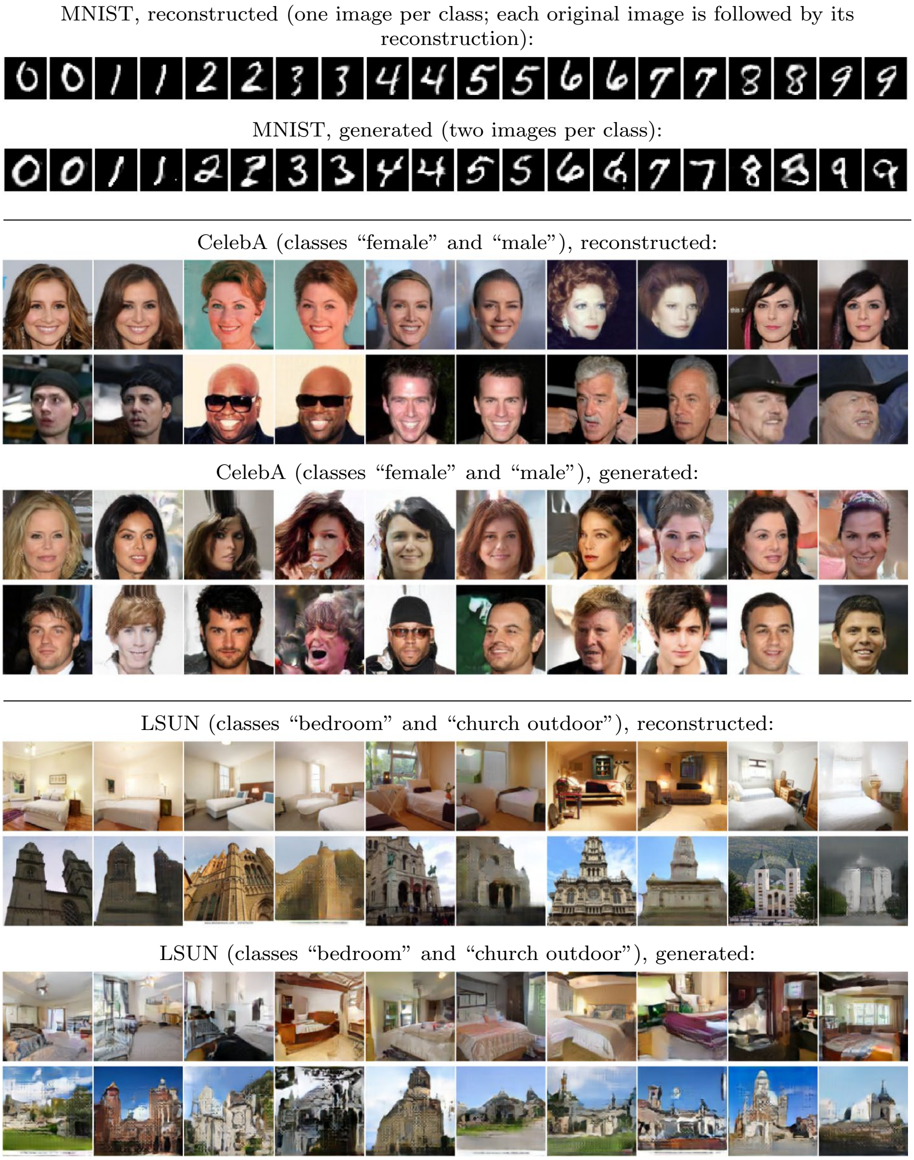
The majority of the proposed metrics relies on the choices of latent distributions as Gaussians and the corresponding Gaussian noise model for this distribution (Eq. 1) that together (1) make the noise preserve the distributions of unperturbed vectors $\mathcal{L}_i$ and (2) result in simple likelihood bounds as ℓ_2 norms. The choice of Gaussians is very conventional, and there is at least one different possible choice: consider the uniform distribution on the unit sphere and the noise model that adds a random Gaussian vector and then normalizes the resulting vector to unit norm. This solution would still result in ℓ_2 vector distances monotonically corresponding to noise likelihood. Some other choices would not achieve both properties (1) and (2). For example, a Gamma-distributed latent vector would sum with a Gamma-distributed noise vector and still remain Gamma-distributed, but the likelihood of such vectors is more difficult.

Conversely, taking Laplace distributions would result in ℓ_1 norm likelihood bounds, but summing the unperturbed vector and the noise would not preserve the distribution family.

The following research directions may be explored in future work:

- Check experimentally whether the findings of Gilmer et al. (2019) are also satisfied in latent spaces—this would further clarify the relationship between the accuracy and latent adversarial robustness.
- Perform robust manifold defense (Jalal et al., 2019) or other form of training with latent adversarial examples, and explore the impact of this training on the values of performance metrics.
- Perform a more thorough evaluation of ImageNet classifiers.
- The proposed latent space adversarial robustness metrics (LAGS, LARS, LAGA, LARA) can be treated as specifications for ANN classifiers, given a threshold on their values to be satisfied. Gradient-based approaches of checking them are imprecise, and verification of even simpler ANN properties was proven to be NP-hard (Katz et al., 2017). A precise, but more computationally intensive way of checking ANN specifications is formal verification (Anderson et al., 2019; Dutta et al., 2017; Huang et al., 2017; Katz et al., 2017, 2019; Ruan et al., 2018; Singh et al., 2019; Elboher et al., 2020).
- The approach could be modified to be applicable to evaluate safety and security of classifiers. First, a practical view on a "natural" adversary must account for the variability in the difficulty of real-world manipulation of high-level features of the classified objects (e.g., changing the tilt on one's head is easier than changing facial features). Second, latent adversarial metrics should be shown to be related to the actual classification mistakes, at least according to human judgment.

Appendix A: additional figures



MNIST, reconstructed (one image per class; each original image is followed by its reconstruction):

MNIST, generated (two images per class):

CelebA (classes “female” and “male”), reconstructed:

CelebA (classes “female” and “male”), generated:

LSUN (classes “bedroom” and “church outdoor”), reconstructed:

LSUN (classes “bedroom” and “church outdoor”), generated:

Fig. 6 Examples of images reconstructed and generated by considered generative models. All CelebA and LSUN images in this figure and other images produced by PIONEER in the figures below have resolution 128×128

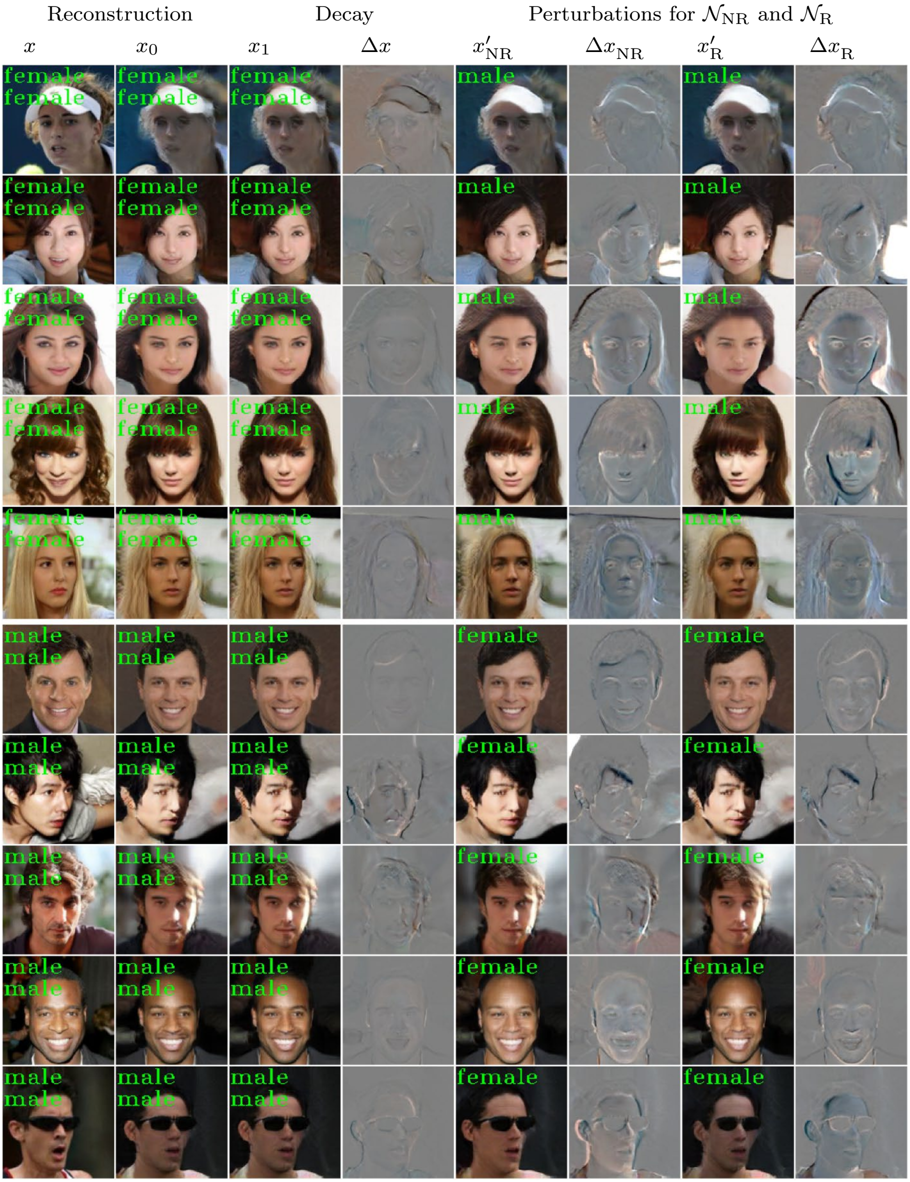


Fig. 7 Additional examples of approximately minimum latent CelebA image perturbations with $\epsilon = 1$ ($d = 0.293$). Images are arranged as in Fig. 5

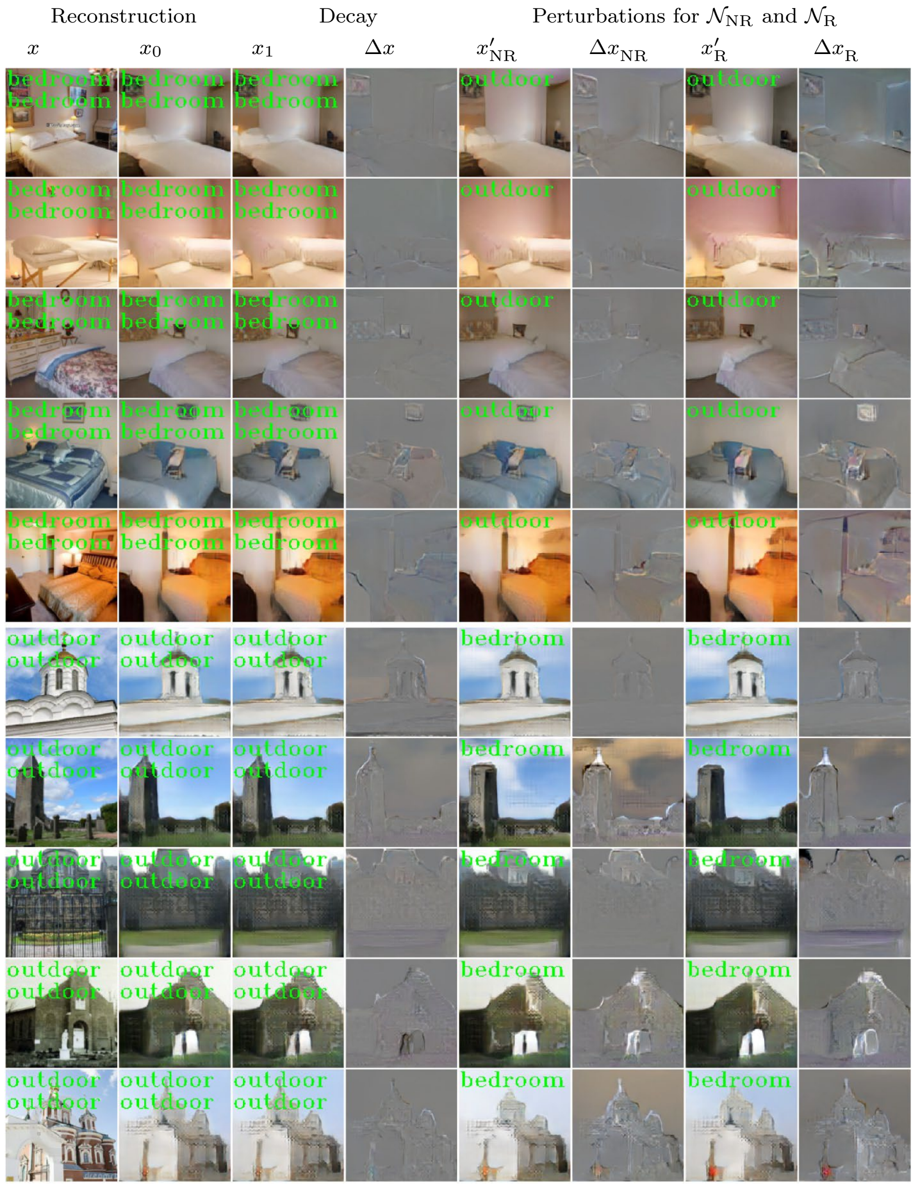


Fig. 8 Additional examples of approximately minimum latent LSUN image perturbations with $\epsilon = 1$ ($d = 0.293$). Images are arranged as in Fig. 5

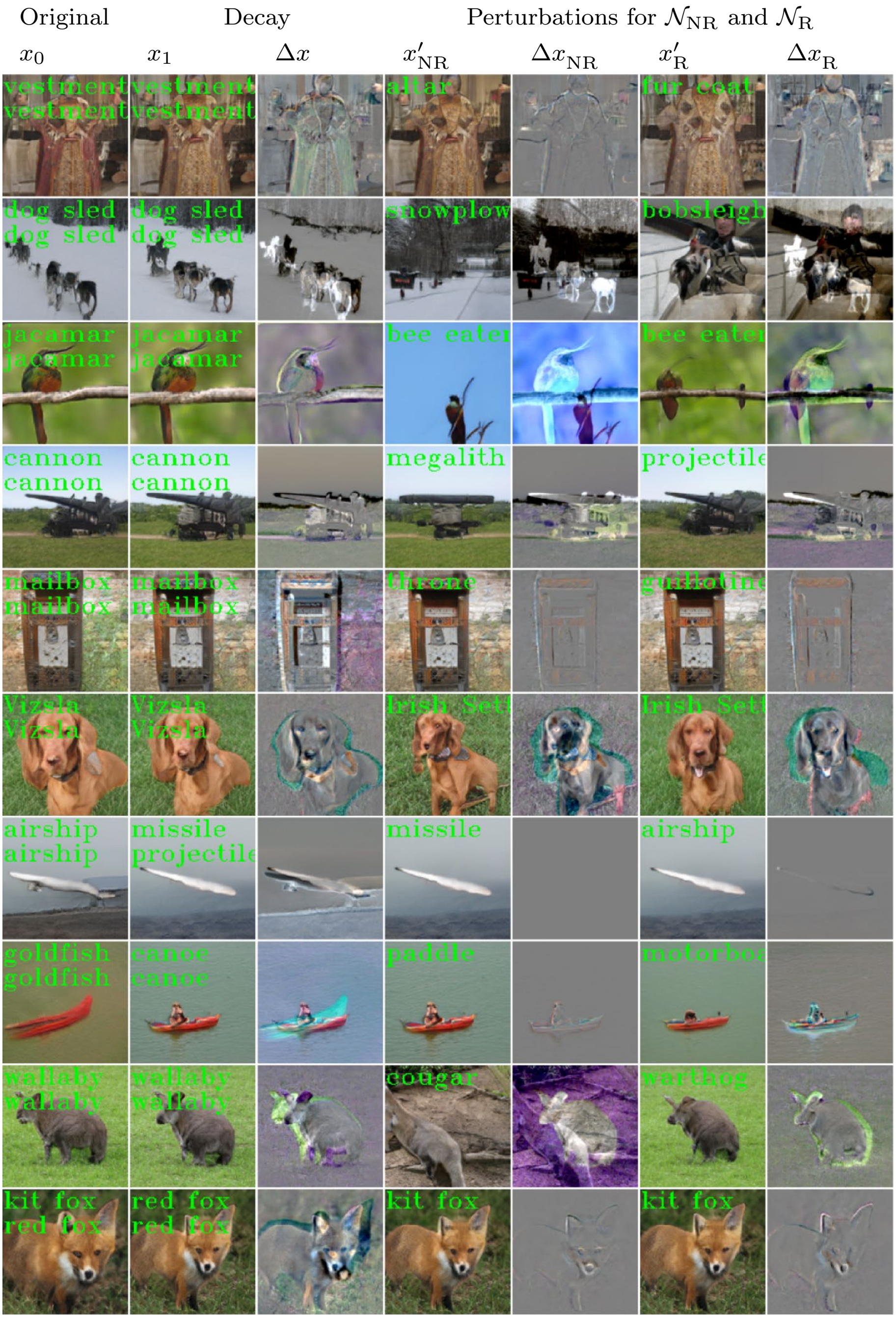


Fig. 9 Examples of approximately minimum latent ImageNet image perturbations with $\epsilon = 1$ ($d = 0.293$). Images are arranged as in Fig. 5 except that all the original images are generated and there is no reconstruction phase. The class labels used to generated the images are: "vestment", "dog sled", "jacamar", "cannon", "mailbox", "Vizsla", "projectile", "canoe", "wallaby", "red fox." To produce this figure, we used resnext as the non-robust (NR) classifier (see Sect. 5.4)

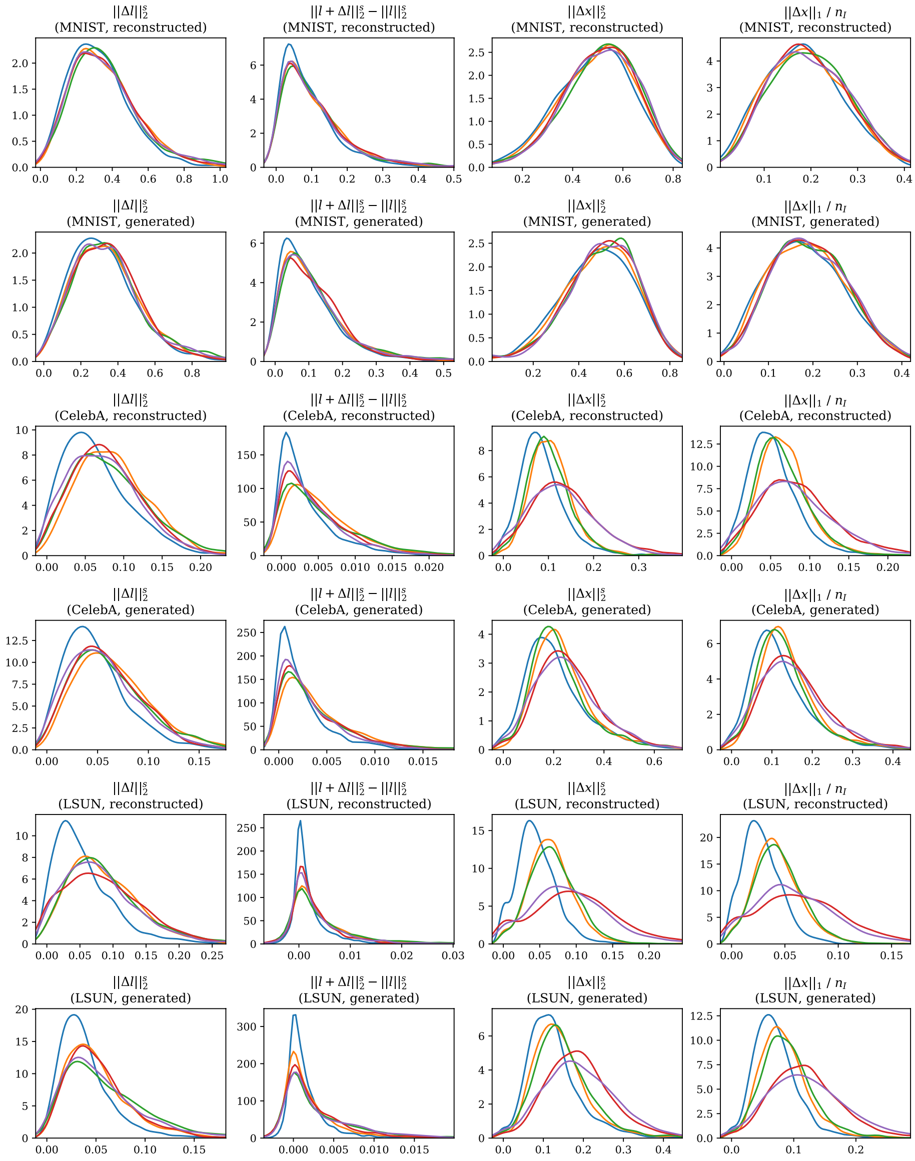


Fig. 10 Distribution plots (with Gaussian kernel density estimation) with statistics on approximately minimum latent image perturbations with $\epsilon = 1$ ($d = 0.293$) found by PGD (600 images for each row of plots). l is the decayed latent vector, Δl is the found perturbation, and Δx is the change of the original image as a vector of pixel intensities. Colors correspond to classifiers as follows: $\mathcal{N}_{\mathrm{UT}}$ is blue, $\mathcal{N}_{\mathrm{NR}}$ is orange, $\mathcal{N}_{\mathrm{CA}}$ is green, $\mathcal{N}_{\mathrm{R}}$ is red, $\mathcal{N}_{\mathrm{B}}$ is purple

B Appendix: proofs of convergence properties of the noise-adding distribution

Below, we prove that the points 2 and 3 of the properties of a noise-adding distribution (Sect. 3.5) are satisfied for $N_{\epsilon,l}$ defined according to (1) and $\mathcal{L}_i = N(0, I)$.

Theorem B.1 *If* $\mathcal{L}_i = N(0, I)$, $N_{\epsilon,l}$ *is defined according to* (1) *and* $\epsilon \to 0$, *then random vectors* $\lambda_\epsilon \sim N_{\epsilon,l}$ *converge in distribution to* $\lambda \equiv l$.

Proof We will prove convergence in distibution according to the Cramér-Wold theorem. We need to show that for all vectors t, $t^\top \lambda_\epsilon$ converges in distribution to $t^\top \lambda$. That is, $\lim\limits_{\epsilon \to 0} \mathbb{P}(t^\top \lambda_\epsilon \le u) = F(u)$ for all $u \in \mathbb{R}$ where $F(u) = \mathbb{P}(t^\top \lambda \le u)$ is continuous. This is trivial for $t = 0$. Otherwise, we have:

$$\begin{aligned}
\lambda_\epsilon &\sim N\left(\frac{l}{\sqrt{1+\epsilon^2}}, \frac{\epsilon^2}{1+\epsilon^2} I\right), \\
t^\top \lambda_\epsilon &\sim N\left(\frac{t^\top l}{\sqrt{1+\epsilon^2}}, \frac{\|t\|_2^2 \epsilon^2}{1+\epsilon^2}\right), \\
\left|\mathbb{P}(t^\top \lambda_\epsilon \le u) - F(u)\right| &= \left|\frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{u - \frac{t^\top l}{\sqrt{1+\epsilon^2}}}{\frac{\|t\|_2 \epsilon}{\sqrt{1+\epsilon^2}}\sqrt{2}}\right) - \left[t^\top l \le u\right]\right| \\
&= \left|\frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{\sqrt{1+\epsilon^2}u - t^\top l}{\|t\|_2 \epsilon \sqrt{2}}\right) - \left[t^\top l \le u\right]\right| \\
&= \left|\frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{u - t^\top l + \left(\sqrt{1+\epsilon^2} - 1\right)u}{\|t\|_2 \epsilon \sqrt{2}}\right) - \left[u - t^\top l \ge 0\right]\right|.
\end{aligned}$$

If $u > t^\top l$, then the nominator inside erf is positive for sufficiently small ϵ, and erf will approach one, making the limit of the entire expression zero. The case of $u < t^\top l$ is similar. Finally, $u = t^\top l$ is a point of discontinuity of $F(u)$. □

Theorem B.2 *If* $\mathcal{L}_i = N(0, I)$, $N_{\epsilon,l}$ *is defined according to* (1) *and* $\epsilon \to +\infty$, *then random vectors* $\lambda_\epsilon \sim N_{\epsilon,l}$ *converge in distribution to* $\lambda \sim \mathcal{L}_i$.

Proof We follow the outline of the previous proof. Now, for all vectors t and scalars u, we need to show that $\lim\limits_{\epsilon \to +\infty} \mathbb{P}(t^\top \lambda_\epsilon \le u) = F(u)$, where $F(u) = \mathbb{P}(t^\top \lambda \le u)$. This is trivial for $t = 0$. Otherwise, $t^\top \lambda \sim N\left(0, \|t\|_2^2\right)$ and we have:

$$F(u) = \frac{1}{2} + \mathrm{erf}\left(\frac{u}{\|t\|_2\sqrt{2}}\right),$$

$$\begin{aligned}\left|\mathbb{P}(t^\top \lambda_\epsilon \leq u) - F(u)\right| &= \left|\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{\sqrt{1+\epsilon^2}u - t^\top l}{\|t\|_2\epsilon\sqrt{2}}\right) - \frac{1}{2} - \frac{1}{2}\mathrm{erf}\left(\frac{u}{\|t\|_2\sqrt{2}}\right)\right| \\ &= \frac{1}{2}\left|\mathrm{erf}\left(\frac{\sqrt{1+\epsilon^2}u - t^\top l}{\|t\|_2\epsilon\sqrt{2}}\right) - \mathrm{erf}\left(\frac{u}{\|t\|_2\sqrt{2}}\right)\right| \\ &= \frac{1}{2}\left|\mathrm{erf}\left(\frac{\sqrt{1+\epsilon^2}}{\epsilon}\frac{u}{\|t\|_2\sqrt{2}} - \frac{1}{\epsilon}\frac{t^\top l}{\|t\|_2\sqrt{2}}\right) - \mathrm{erf}\left(\frac{u}{\|t\|_2\sqrt{2}}\right)\right|.\end{aligned}$$

The last expression approaches zero when $\epsilon \to +\infty$. □

C Appendix: classifier training procedure

All classifiers listed in Sect. 5.1 were trained as follows. We considered three slightly different CNN architures. Two of them are based on the script https://github.com/keras-team/keras/blob/f295e8ee39d4ba841ac281a9337d69c7bc5e0eb6/examples/cifar10_cnn.py and differ in their depth. Essentially, these are simple CNN architectures composed of convolutional blocks, ReLU nonlinearities, batch normalization, max-pooling, dropout, and a fully connected layer with softmax on top. One more architecture used the same operations, but was based on residual blocks. Training was done with RMSprop. In each epoch, we took 100 thousand random images from the training set. The learning rate was set to 0.0004 and multiplied by 0.75 after each epoch. Training continued for up to 8 epochs, but was stopped prematurely if validation accuracy had not increased during the previous epoch. The following was specific to different classifier types:

1. $\mathcal{N}_{\mathrm{UT}}$: No data augmentation was used. Training was stopped after one epoch.
2. $\mathcal{N}_{\mathrm{NR}}$: No data augmentation was used. Training was done for the remaining 7 epochs starting from $\mathcal{N}_{\mathrm{UT}}$.
3. $\mathcal{N}_{\mathrm{CA}}$: Training images were augmented with conventional approaches: small affine transformations, color distortions and erasures of small image parts.
4. $\mathcal{N}_{\mathrm{R}}$: Training images were augmented with Gaussian noise of magnitude $\sigma = 0.8$ (pixel intensities belong to $[-1, 1]$). Note that this is different from the work (Gilmer et al. 2019), where for each image first σ was selected uniformly at random and then noise was added. Training was started from $\mathcal{N}_{\mathrm{NR}}$.
5. $\mathcal{N}_{\mathrm{B}}$: Training images were first augmented conventionally (as in the case of $\mathcal{N}_{\mathrm{CA}}$), and then with Gaussian noise (as in the case of $\mathcal{N}_{\mathrm{R}}$). Training was started from $\mathcal{N}_{\mathrm{CA}}$.

Acknowledgements The work was financially supported by the Government of the Russian Federation (Grant 08–08) and by the Russian Science Foundation (Project 20-19-00700). We acknowledge the computational resources provided by the Aalto Science-IT project. We thank Ari Heljakka for his help related to the use of the PIONEER generative autoencoder.

References

Akhtar, N., & Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access, 6,* 14410–14430.

Amadou Dia, O., Barshan, E., & Babanezhad, R. (2019). Semantics preserving adversarial attacks. *arXiv preprint arXiv:190303905v5*.

Anderson, G., Pailoor, S., Dillig, I., & Chaudhuri, S. (2019). Optimization and abstraction: A synergistic approach for analyzing neural network robustness. In *40th ACM SIGPLAN Conference on Programming Language Design and Implementation, ACM*, pp. 731–744.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223.

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., & Criminisi, A. (2016). Measuring neural net robustness with constraints. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (pp. 2613–2621). Curran Associates, Inc.

Bau, D., Zhu, J.-Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., & Torralba, A. (2019). Seeing what a GAN cannot generate. In *IEEE International Conference on Computer Vision*, pp. 4502–4511.

Boopathy, A., Weng, T.-W., Chen, P.-Y., Liu, S., & Daniel, L. (2019). CNN-Cert: An efficient framework for certifying robustness of convolutional neural networks. *AAAI Conference on Artificial Intelligence, 33,* 3240–3247.

Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*.

Dalvi, N., Domingos, P., Sanghai, S., & Verma, D., et al. (2004). Adversarial classification. In *10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM*, pp. 99–108.

Dreossi, T., Jha, S., & Seshia, S. A. (2018). Semantic adversarial deep learning. In *International Conference on Computer Aided Verification, Springer*, pp. 3–26.

Dutta, S., Jha, S., Sanakaranarayanan, S., & Tiwari, A. (2017). Output range analysis for deep neural networks. *arXiv preprint arXiv:170909130*.

Elboher, Y. Y., Gottschlich, J., & Katz, G. (2020). An abstraction-based framework for neural network verification. In *International Conference on Computer Aided Verification,* Springer, pp. 43–65.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., & Madry, A. (2019). Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pp. 1802–1811.

Fawzi, A., Fawzi, O., & Frossard, P. (2018). Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning, 107*(3), 481–508.

Gilmer, J., Adams, R. P., Goodfellow, I., Andersen, D., & Dahl, G. E. (2018). Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:180706732*.

Gilmer, J., Ford, N., Carlini, N., & Cubuk, E. (2019). Adversarial examples are a natural consequence of test error in noise. In *36th International Conference on Machine Learning*, pp. 2280–2289.

Globerson, A., & Roweis, S. (2006). Nightmare at test time: Robust learning by feature deletion. In *International Conference on Machine Learning, ACM*, pp. 353–360.

Goodfellow, I. (2018). Gradient masking causes CLEVER to overestimate adversarial perturbation size. *arXiv preprint arXiv:180407870*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (pp. 2672–2680). Curran Associates, Inc.

Gopinath, D., Katz, G., Pasareanu, C. S., & Barrett, C. (2017). DeepSafe: A data-driven approach for checking adversarial robustness in neural networks. *arXiv preprint arXiv:171000486*.

Gu, K., Yang, B., Ngiam, J., Le, Q., & Shlens, J. (2019). Using videos to evaluate image model robustness. In *7th International Conference on Learning Representations*.

Heljakka, A., Solin, A., & Kannala, J. (2018). Pioneer networks: Progressively growing generative autoencoder. In *Asian Conference on Computer Vision,* Springer, pp. 22–38.

Heljakka, A., Solin, A., & Kannala, J. (2020). Towards photographic image manipulation with balanced growing of generative autoencoders. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *7th International Conference on Learning Representations*.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., & Song, D. (2019). Natural adversarial examples. *arXiv preprint arXiv:190707174*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (pp. 6626–6637). Curran Associates, Inc.

Huang, X., Kwiatkowska, M., Wang, S., & Wu, M. (2017). Safety verification of deep neural networks. In *International Conference on Computer Aided Verification, Springer*, pp. 3–29.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., & Madry, A. (2019). Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pp. 125–136.

Jalal, A., Ilyas, A., Asteri, E., Daskalakis, C., & Dimakis, A. G. (2019). The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:171209196*.

Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, Springer, pp. 97–117.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., & Zeljić, A., et al. (2019). The Marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, Springer, pp. 443–452.

LeCun, Y. (1998). The MNIST database of handwritten digits.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision*, pp. 3730–3738.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations*.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:151105644*.

Mirman, M., Gehr, T., & Vechev, M. (2020). Robustness certification of generative models. *arXiv preprint arXiv:200414756*.

Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). DeepFool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582.

Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. In *34th International Conference on Machine Learning, JMLR.org*, pp. 2642–2651

Ruan, W., Huang, X., & Kwiatkowska, M. (2018). Reachability analysis of deep neural networks with provable guarantees. *arXiv preprint arXiv:180502242*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV), 115*(3), 211–252.

Samangouei, P., Kabkab, M., & Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:180506605*.

Santurkar, S., Ilyas, A., Tsipras, D., Engstrom, L., Tran, B., & Madry, A. (2019). Image synthesis with a single (robust) classifier. In *Advances in Neural Information Processing Systems*, 1260–1271.

Singh, G., Gehr, T., Püschel, M., & Vechev, M. (2019). An abstract domain for certifying neural networks. In *Proceedings of the ACM on Programming Languages 3(POPL):41*.

Song, Y., Kim, T., Nowozin, S., Ermon, S., & Kushman, N. (2018). PixelDefend: Leveraging generative models to understand and defend against adversarial examples. In *6th International Conference on Learning Representations*.

Song, Y., Shu, R., Kushman, N., & Ermon, S. (2018). Constructing unrestricted adversarial examples with generative models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (pp. 8312–8323). Curran Associates, Inc

Szegedy, C., Zaremba, W., Sutskever, I., Estrach, J. B., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *2nd International Conference on Learning Representations*.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2018). Robustness may be at odds with accuracy. In *6th International Conference on Learning Representations*.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer Science & Business Media.

Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., Daniel, L. (2018). Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:180110578*.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., & Xiao, J. (2015). LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:150603365*.

Yu, F., Qin, Z., Liu, C., Zhao, L., Wang, Y., & Chen, X. (2019). Interpreting and evaluating neural network robustness. *arXiv preprint arXiv:190504270*.

Zhao, Z., Dua, D., & Singh, S. (2017). Generating natural adversarial examples. *arXiv preprint arXiv:171011342*.

Authors and Affiliations

Igor Buzhinsky[1,2] · **Arseny Nerinovsky**[1] · **Stavros Tripakis**[3]

Arseny Nerinovsky
nerinovsky.arseny@gmail.com

Stavros Tripakis
stavros@northeastern.edu

[1] Computer Technologies Laboratory, ITMO University, St. Petersburg, Russia

[2] Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland

[3] Northeastern University, Boston, MA, US