2019 IEEE Symposium on Security and Privacy

Certified Robustness to Adversarial Examples with Differential Privacy

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana
Columbia University

***Abstract*—Adversarial examples that fool machine learning models, particularly deep neural networks, have been a topic of intense research interest, with attacks and defenses being developed in a tight back-and-forth. Most past defenses are *best effort* and have been shown to be vulnerable to sophisticated attacks. Recently a set of *certified defenses* have been introduced, which provide guarantees of robustness to norm-bounded attacks. However these defenses either do not scale to large datasets or are limited in the types of models they can support. This paper presents the first certified defense that both scales to large networks and datasets (such as Google's Inception network for ImageNet) and applies broadly to arbitrary model types. Our defense, called *PixelDP*, is based on a novel connection between robustness against adversarial examples and differential privacy, a cryptographically-inspired privacy formalism, that provides a rigorous, generic, and flexible foundation for defense.**

I. Introduction

Deep neural networks (DNNs) perform exceptionally well on many machine learning tasks, including safety- and security-sensitive applications such as self-driving cars [5], malware classification [48], face recognition [47], and critical infrastructure [71]. Robustness against malicious behavior is important in many of these applications, yet in recent years it has become clear that DNNs are vulnerable to a broad range of attacks. Among these attacks – broadly surveyed in [46] – are *adversarial examples*: the adversary finds small perturbations to correctly classified inputs that cause a DNN to produce an erroneous prediction, possibly of the adversary's choosing [56]. Adversarial examples pose serious threats to security-critical applications. A classic example is an adversary attaching a small, human-imperceptible sticker onto a stop sign that causes a self-driving car to recognize it as a yield sign. Adversarial examples have also been demonstrated in domains such as reinforcement learning [32] and generative models [31].

Since the initial demonstration of adversarial examples [56], numerous attacks and defenses have been proposed, each building on one another. Initially, most defenses used *best-effort* approaches and were broken soon after introduction. Model distillation, proposed as a robust defense in [45], was subsequently broken in [7]. Other work [36] claimed that adversarial examples are unlikely to fool machine learning (ML) models in the real-world, due to the rotation and scaling introduced by even the slightest camera movements. However, [3] demonstrated a new attack strategy that is robust to rotation and scaling. While this back-and-forth has advanced the state of the art, recently the community has started to recognize that rigorous, theory-backed, defensive approaches are required to put us off this arms race.

Accordingly, a new set of *certified defenses* have emerged over the past year, that provide rigorous guarantees of robustness against norm-bounded attacks [12], [52], [65]. These works alter the learning methods to both optimize for robustness against attack at training time and permit provable robustness checks at inference time. At present, these methods tend to be tied to internal network details, such as the type of activation functions and the network architecture. They struggle to generalize across different types of DNNs and have only been evaluated on small networks and datasets.

We propose a new and orthogonal approach to certified robustness against adversarial examples that is *broadly applicable, generic, and scalable*. We observe for the first time a connection between *differential privacy* (DP), a cryptography-inspired formalism, and a definition of robustness against norm-bounded adversarial examples in ML. We leverage this connection to develop *PixelDP*, the first certified defense we are aware of that both scales to large networks and datasets (such as Google's Inception network trained on ImageNet) and can be adapted broadly to arbitrary DNN architectures. Our approach can even be incorporated with no structural changes in the target network (e.g., through a separate auto-encoder as described in Section III-B). We provide a brief overview of our approach below along with the section references that detail the corresponding parts.

§II establishes the DP-robustness connection formally (our first contribution). To give the intuition, DP is a framework for randomizing computations running on databases such that a small change in the database (removing or altering one row or a small set of rows) is guaranteed to result in a bounded change in the distribution over the algorithm's outputs. Separately, robustness against adversarial examples can be defined as ensuring that small changes in the input of an ML predictor (such as changing a few pixels in an image in the case of an l_0-norm attack) will not result in drastic changes to its predictions (such as changing its label from a stop to a yield sign). Thus, if we think of a DNN's inputs (e.g., images) as databases in DP parlance, and individual features (e.g., pixels) as rows in DP, we observe that randomizing the outputs of a DNN's prediction function to enforce

DOI 10.1109/SP.2019.00044

DP on a small number of pixels in an image *guarantees* robustness of predictions against adversarial examples that can change up to that number of pixels. The connection can be expanded to standard attack norms, including l_1, l_2, and l_∞ norms.

§III describes *PixelDP*, the first certified defense against norm-bounded adversarial examples based on differential privacy (our second contribution). Incorporating DP into the learning procedure to increase robustness to adversarial examples requires is completely different and orthogonal to using DP to preserve the privacy of the training set, the focus of prior DP ML literature [40], [1], [9] (as § VI explains). A PixelDP DNN includes in its architecture a *DP noise layer* that randomizes the network's computation, to enforce DP bounds on how much the distribution over its predictions can change with small, norm-bounded changes in the input. At inference time, we leverage these DP bounds to implement a certified robustness check for individual predictions. Passing the check for a given input *guarantees* that no perturbation exists up to a particular size that causes the network to change its prediction. The robustness certificate can be used to either act exclusively on robust predictions, or to lower-bound the network's accuracy under attack on a test set.

§IV presents the first experimental evaluation of a certified adversarial-examples defense for the Inception network trained on the ImageNet dataset (our third contribution). We additionally evaluate PixelDP on various network architectures for four other datasets (CIFAR-10, CIFAR-100, SVHN, MNIST), on which previous defenses – both best effort and certified – are usually evaluated. Our results indicate that PixelDP is (1) as effective at defending against attacks as today's state-of-the-art, best-effort defense [37] and (2) more scalable and broadly applicable than a prior certified defense.

Our experience points to DP as a uniquely generic, broadly applicable, and flexible foundation for certified defense against norm-bounded adversarial examples (§V, §VI). We credit these properties to the *post-processing property of DP*, which lets us incorporate the certified defense in a network-agnostic way.

II. DP-Robustness Connection

A. Adversarial ML Background

An ML model can be viewed as a function mapping inputs – typically a vector of numerical feature values – to an output (a label for multiclass classification and a real number for regression). Focusing on multiclass classification, we define a *model* as a function $f\colon \mathbb{R}^n \to \mathcal{K}$ that maps n-dimensional inputs to a label in the set $\mathcal{K} = \{1, \dots, K\}$ of all possible labels. Such models typically map an input x to a vector of scores $y(x) = (y_1(x), \dots, y_K(x))$, such that $y_k(x) \in [0,1]$ and $\sum_{k=1}^K y_k(x) = 1$. These scores are interpreted as a probability distribution over the labels, and the model returns the label with highest probability, i.e., $f(x) = \arg\max_{k\in\mathcal{K}} y_k(x)$. We denote the function that maps input x to y as Q and call it the *scoring function*; we denote the function that gives the ultimate prediction for input x as f and call it the *prediction procedure*.

Adversarial Examples. Adversarial examples are a class of attack against ML models, studied particularly on deep neural networks for multiclass image classification. The attacker constructs a small change to a given, fixed input, that wildly changes the predicted output. Notationally, if the input is x, we denote an adversarial version of that input by $x + \alpha$, where α is the change or perturbation introduced by the attacker. When x is a vector of pixels (for images), then x_i is the i'th pixel in the image and α_i is the change to the i'th pixel.

It is natural to constrain the amount of change an attacker is allowed to make to the input, and often this is measured by the p-norm of the change, denoted by $\|\alpha\|_p$. For $1 \le p < \infty$, the p-norm of α is defined by $\|\alpha\|_p = (\sum_{i=1}^n |\alpha_i|^p)^{1/p}$; for $p = \infty$, it is $\|\alpha\|_\infty = \max_i |\alpha_i|$. Also commonly used is the 0-norm (which is technically not a norm): $\|\alpha\|_0 = |\{i : \alpha_i \neq 0\}|$. A small 0-norm attack is permitted to arbitrarily change a few entries of the input; for example, an attack on the image recognition system for self-driving cars based on putting a sticker in the field of vision is such an attack [19]. Small p-norm attacks for larger values of p (including $p = \infty$) require the changes to the pixels to be small in an aggregate sense, but the changes may be spread out over many or all features. A change in the lighting condition of an image may correspond to such an attack [34], [50]. The latter attacks are generally considered more powerful, as they can easily remain invisible to human observers. Other attacks that are not amenable to norm bounding exist [67], [54], [66], but this paper deals exclusively with norm-bounded attacks.

Let $B_p(r) := \{\alpha \in \mathbb{R}^n : \|\alpha\|_p \le r\}$ be the p-norm ball of radius r. For a given classification model, f, and a fixed input, $x \in \mathbb{R}^n$, an attacker is able to craft a successful adversarial example of size L for a given p-norm if they find $\alpha \in B_p(L)$ such that $f(x + \alpha) \neq f(x)$. The attacker thus tries to find a small change to x that will change the predicted label.

Robustness Definition. Intuitively, a predictive model may be regarded as *robust* to adversarial examples if its output is *insensitive* to small changes to any *plausible* input that may be encountered in deployment. To formalize this notion, we must first establish what qualifies as a *plausible* input. This is difficult: the adversarial examples literature has not settled on such a definition. Instead, model robustness is typically assessed on inputs from a test set that are not used in model training – similar to how accuracy is assessed on a test set and not a property on all plausible inputs. We adopt this view of robustness.

Next, given an input, we must establish a definition for *insensitivity* to small changes to the input. We say a model f is insensitive, or *robust*, to attacks of p-norm L on a given input x if $f(x) = f(x+\alpha)$ for all $\alpha \in B_p(L)$. If f is a multiclass classification model based on label scores (as in §II-A), this is equivalent to:

$$\forall \alpha \in B_p(L) \,.\, y_k(x+\alpha) > \max_{i:i\neq k} y_i(x+\alpha), \quad (1)$$

where $k := f(x)$. A small change in the input does not alter the scores so much as to change the predicted label.

B. DP Background

DP is concerned with whether the output of a computation over a database can reveal information about individual records in the database. To prevent such information leakage, randomness is introduced into the computation to hide details of individual records.

A randomized algorithm A that takes as input a database d and outputs a value in a space O is said to satisfy (ϵ, δ)-*DP with respect to a metric* ρ *over databases* if, for any databases d and d' with $\rho(d, d') \leq 1$, and for any subset of possible outputs $S \subseteq O$, we have

$$P(A(d) \in S) \leq e^{\epsilon} P(A(d') \in S) + \delta. \quad (2)$$

Here, $\epsilon > 0$ and $\delta \in [0, 1]$ are parameters that quantify the strength of the privacy guarantee. In the standard DP definition, the metric ρ is the Hamming metric, which simply counts the number of entries that differ in the two databases. For small ϵ and δ, the standard (ϵ, δ)-DP guarantee implies that changing a single entry in the database cannot change the output distribution very much. DP also applies to general metrics ρ [8], including p-norms relevant to norm-based adversarial examples.

Our approach relies on two key properties of DP. First is the well-known *post-processing property*: any computation applied to the output of an (ϵ, δ)-DP algorithm remains (ϵ, δ)-DP. Second is the *expected output stability property*, a rather obvious but not previously enunciated property that we prove in Lemma 1: the expected value of an (ϵ, δ)-DP algorithm with bounded output is not sensitive to small changes in the input.

Lemma 1. (Expected Output Stability Bound) *Suppose a randomized function A, with bounded output $A(x) \in [0, b]$, $b \in \mathbb{R}^+$, satisfies (ϵ, δ)-DP. Then the expected value of its output meets the following property:*

$$\forall \alpha \in B_p(1) \,.\, \mathbb{E}(A(x)) \leq e^{\epsilon} \mathbb{E}(A(x+\alpha)) + b\delta.$$

The expectation is taken over the randomness in A.

Proof: Consider any $\alpha \in B_p(1)$, and let $x' := x + \alpha$. We write the expected output as:

$$\mathbb{E}(A(x)) = \int_0^b P(A(x) > t)dt.$$

We next apply Equation (2) from the (ϵ, δ)-DP property:

$$\mathbb{E}(A(x)) \leq e^{\epsilon}\Big(\int_0^b P(A(x') > t)dt\Big) + \int_0^b \delta dt$$
$$= e^{\epsilon}\mathbb{E}(A(x')) + \int_0^b \delta dt.$$

Since δ is a constant, $\int_0^b \delta dt = b\delta$. ∎

C. DP-Robustness Connection

The intuition behind using DP to provide robustness to adversarial examples is to create a *DP scoring function* such that, given an input example, the predictions are DP with regards to the features of the input (e.g. the pixels of an image). In this setting, we can derive stability bounds for the expected output of the DP function using Lemma 1. The bounds, combined with Equation (1), give a rigorous condition (or *certification*) for robustness to adversarial examples.

Formally, regard the feature values (e.g., pixels) of an input x as the records in a database, and consider a randomized scoring function A that, on input x, outputs scores $(y_1(x), \ldots, y_K(x))$ (with $y_k(x) \in [0, 1]$ and $\sum_{k=1}^{K} y_k(x) = 1$). We say that A is an *(ϵ, δ)-pixel-level differentially private* (or *(ϵ, δ)-PixelDP*) function if it satisfies (ϵ, δ)-DP (for a given metric). This is formally equivalent to the standard definition of DP, but we use this terminology to emphasize the context in which we apply the definition, which is fundamentally different than the context in which DP is traditionally applied in ML (see §VI for distinction).

Lemma 1 directly implies bounds on the expected outcome on an (ϵ, δ)-PixelDP scoring function:

Corollary 1. *Suppose a randomized function A satisfies (ϵ, δ)-PixelDP with respect to a p-norm metric, and where $A(x) = (y_1(x), \ldots, y_K(x))$, $y_k(x) \in [0, 1]$:*

$$\forall k, \forall \alpha \in B_p(1) \,.\, \mathbb{E}(y_k(x)) \leq e^{\epsilon}\mathbb{E}(y_k(x+\alpha)) + \delta. \quad (3)$$

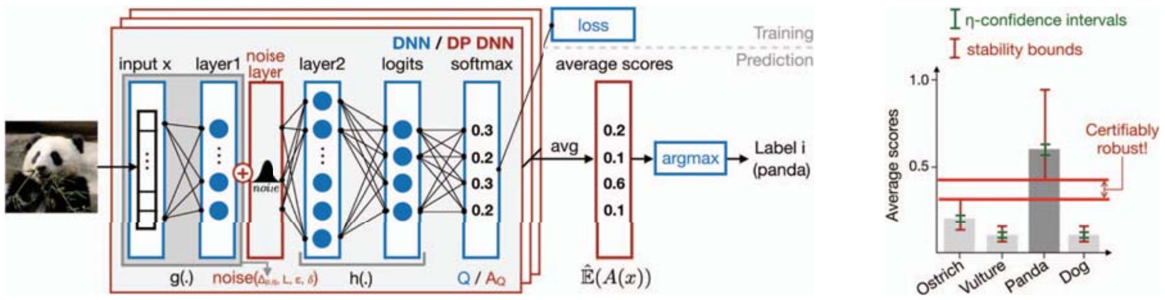
Proof: For any k apply Lemma 1 with $b = 1$. ∎

Our approach is to transform a model's scoring function into a *randomized (ϵ, δ)-PixelDP scoring function*, $A(x)$, and then have the model's prediction procedure, f, use A's *expected output over the DP noise*, $\mathbb{E}(A(x))$, as the label probability vector from which to pick the $\arg\max$. I.e., $f(x) = \arg\max_{k\in\mathcal{K}} \mathbb{E}(A_k(x))$. We prove that a model constructed this way allows the following robustness certification to adversarial examples:

Proposition 1. (Robustness Condition) *Suppose A satisfies (ϵ, δ)-PixelDP with respect to a p-norm metric. For any input x, if for some $k \in \mathcal{K}$,*

$$\mathbb{E}(A_k(x)) > e^{2\epsilon} \max_{i:i\neq k} \mathbb{E}(A_i(x)) + (1 + e^{\epsilon})\delta, \quad (4)$$

then the multiclass classification model based on label probability vector $y(x) = (\mathbb{E}(A_1(x)), \ldots, \mathbb{E}(A_K(x)))$ is



(a) PixelDP DNN Architecture (b) Robustness Test Example

Fig. 1: **Architecture.** (a) In blue, the original DNN. In red, the noise layer that provides the (ϵ, δ)-DP guarantees. The noise can be added to the inputs *or* any of the following layers, but the distribution is rescaled by the sensitivity $\Delta_{p,q}$ of the computation performed by each layer before the noise layer. The DNN is trained with the original loss and optimizer (e.g., Momentum stochastic gradient descent). Predictions repeatedly call the (ϵ, δ)-DP DNN to measure its empirical expectation over the scores. (b) After adding the bounds for the measurement error between the empirical and true expectation (green) and the stability bounds from Lemma 1 for a given attack size L_{attack} (red), the prediction is certified robust to this attack size if the lower bound of the $\arg\max$ label does not overlap with the upper bound of any other labels.

robust to attacks α of size $\|\alpha\|_p \leq 1$ on input x.

Proof: Consider any $\alpha \in B_p(1)$, and let $x' := x + \alpha$. From Equation (3), we have:

$$\mathbb{E}(A_k(x)) \leq e^\epsilon \mathbb{E}(A_k(x')) + \delta, \qquad (a)$$
$$\mathbb{E}(A_i(x')) \leq e^\epsilon \mathbb{E}(A_i(x)) + \delta, \quad i \neq k. \qquad (b)$$

Equation (a) gives a lower-bound on $\mathbb{E}(A_k(x'))$; Equation (b) gives an upper-bound on $\max_{i \neq k} \mathbb{E}(A_i(x'))$. The hypothesis in the proposition statement (Equation (4)) implies that the lower-bound of the expected score for label k is strictly higher than the upper-bound for the expected score for any other label, which in turn implies the condition from Equation (1) for robustness at x. To spell it out:

$$\begin{aligned}
\mathbb{E}(A_k(x')) &\overset{\text{Eq(a)}}{\geq} \frac{\mathbb{E}(A_k(x)) - \delta}{e^\epsilon} \\
&\overset{\text{Eq(4)}}{>} \frac{e^{2\epsilon} \max_{i:i \neq k} \mathbb{E}(A_i(x)) + (1 + e^\epsilon)\delta - \delta}{e^\epsilon} \\
&= e^\epsilon \max_{i:i \neq k} \mathbb{E}(A_i(x)) + \delta \\
&\overset{\text{Eq(b)}}{\geq} \max_{i:i \neq k} \mathbb{E}(A_i(x')) \\
\implies \mathbb{E}(A_k(x')) &> \max_{i:i \neq k} \mathbb{E}(A_i(x + \alpha)) \ \forall \alpha \in B_p(1),
\end{aligned}$$

the very definition of robustness at x (Equation (1)). ∎

The preceding certification test is *exact* regardless of the value of the δ parameter of differential privacy: there is no failure probability in this test. The test applies only to attacks of p-norm size of 1, however all preceding results generalize to *attacks of p-norm size L*, i.e., when $\|\alpha\|_p \leq L$, by applying group privacy [18]. The next section shows how to apply group privacy (§III-B) and generalize the certification test to make it practical (§III-D).

III. PixelDP Certified Defense

A. Architecture

PixelDP is a certified defense against p-norm bounded adversarial example attacks built on the preceding DP-robustness connection. Fig. 1(a) shows an example PixelDP DNN architecture for multi-class image classification. The original architecture is shown in blue; the changes introduced to make it PixelDP are shown in red. Denote Q the original DNN's scoring function; it is a deterministic map from images x to a probability distribution over the K labels $Q(x) = (y_1(x), \ldots, y_K(x))$. The vulnerability to adversarial examples stems from the unbounded sensitivity of Q with respect to p-norm changes in the input. Making the DNN (ϵ, δ)-PixelDP involves adding *calibrated noise* to turn Q into an (ϵ, δ)-DP randomized function A_Q; the expected output of that function will have bounded sensitivity to p-norm changes in the input. We achieve this by introducing a *noise layer* (shown in red in Fig. 1(a)) that adds zero-mean noise to the output of the layer preceding it (layer1 in Fig. 1(a)). The noise is drawn from a Laplace or Gaussian distribution and its standard deviation is proportional to: (1) L, the p-norm attack bound for which we are constructing the network and (2) Δ, the sensitivity of the pre-noise computation (the grey box in Fig. 1(a)) with respect to p-norm input changes.

Training an (ϵ, δ)-PixelDP network is similar to training the original network: we use the original loss and optimizer, such as stochastic gradient descent. The major difference is that we alter the pre-noise computation to *constrain its sensitivity* with regards to p-norm input changes. Denote $Q(x) = h(g(x))$, where g is the pre-noise computation and h is the subsequent computation that produces $Q(x)$ in the original network. We leverage known techniques, reviewed in §III-C, to transform g into another function, $\tilde{g}$, that has a fixed sensitivity (Δ) to p-norm input changes. We then add the noise layer to the output of $\tilde{g}$, with a standard deviation scaled by Δ and L to ensure (ϵ, δ)-PixelDP for p-norm changes of size L. Denote the resulting scoring function of the PixelDP network: $A_Q(x) = h(\tilde{g}(x) + noise(\Delta, L, \epsilon, \delta))$, where $noise(.)$ is the function implementing the Laplace/Gaussian draw. Assuming that the noise layer is placed such that h only processes the DP output of $\tilde{g}(x)$ without accessing x again (i.e., no skip layers exist from pre-noise to post-noise computation), the post-processing property of DP ensures that $A_Q(x)$ also satisfies (ϵ, δ)-PixelDP for p-norm changes of size L.

Prediction on the (ϵ, δ)-PixelDP scoring function, $A_Q(x)$, affords the robustness certification in Proposition 1 if the prediction procedure uses the *expected scores*, $\mathbb{E}(A_Q(x))$, to select the winning label for any input x. Unfortunately, due to the potentially complex nature of the post-noise computation, h, we cannot compute this output expectation analytically. We therefore resort to Monte Carlo methods to estimate it at prediction time and develop an approximate version of the robustness certification in Proposition 1 that uses standard techniques from probability theory to account for the estimation error (§III-D). Specifically, given input x, PixelDP's prediction procedure invokes $A_Q(x)$ multiple times with new draws of the noise layer. It then averages the results for each label, thereby computing an estimation $\hat{\mathbb{E}}(A_Q(x))$ of the expected score $\mathbb{E}(A_Q(x))$. It then computes an η-confidence interval for $\hat{\mathbb{E}}(A_Q(x))$ that holds with probability η. Finally, it integrates this confidence interval into the stability bound for the expectation of a DP computation (Lemma 1) to obtain η-confidence upper and lower bounds on the change an adversary can make to the average score of any label with a p-norm input change of size up to L. Fig. 1(b) illustrates the upper and lower bounds applied to the average score of each label by the PixelDP prediction procedure. If the lower bound for the label with the top average score is strictly greater than the upper bound for every other label, then, with probability η, the PixelDP network's prediction for input x is *robust* to arbitrary attacks of p-norm size L. The failure probability of this robustness certification, $1-\eta$, can be made arbitrarily small by increasing the number of invocations of $A_Q(x)$.

One can use PixelDP's certification check in two ways: (1) one can decide only to actuate on predictions that are deemed robust to attacks of a particular size; or (2) one can compute, on a test set, a lower bound of a PixelDP network's accuracy under p-norm bounded attack, independent of how the attack is implemented. This bound, called *certified accuracy*, will hold no matter how effective future generations of the attack are.

The remainder of this section details the noise layer, training, and certified prediction procedures. To simplify notation, we will henceforth use A instead of A_Q.

B. DP Noise Layer

The noise layer enforces (ϵ, δ)-PixelDP by inserting noise inside the DNN using one of two well-known DP mechanisms: the Laplacian and Gaussian mechanisms. Both rely upon the *sensitivity* of the pre-noise layers (function g). The sensitivity of a function g is defined as the maximum change in output that can be produced by a change in the input, given some distance metrics for the input and output (p-norm and q-norm, respectively):

$$\Delta_{p,q} = \Delta^g_{p,q} = \max_{x,x':x\neq x'} \frac{\|g(x) - g(x')\|_q}{\|x - x'\|_p}.$$

Assuming we can compute the sensitivity of the pre-noise layers (addressed shortly), the noise layer leverages the Laplace and Gaussian mechanisms as follows. On every invocation of the network on an input x (whether for training or prediction) the noise layer computes $g(x) + Z$, where the coordinates $Z = (Z_1, \ldots, Z_m)$ are independent random variables from a noise distribution defined by the function $noise(\Delta, L, \epsilon, \delta)$.

- Laplacian mechanism: $noise(\Delta, L, \epsilon, \delta)$ uses the Laplace distribution with mean zero and standard deviation $\sigma = \sqrt{2}\Delta_{p,1}L/\epsilon$; it gives $(\epsilon, 0)$-DP.
- Gaussian mechanism: $noise(\Delta, L, \epsilon, \delta)$ uses the Gaussian distribution with mean zero and standard deviation $\sigma = \sqrt{2\ln(\frac{1.25}{\delta})}\Delta_{p,2}L/\epsilon$; it gives (ϵ, δ)-DP for $\epsilon \leq 1$.

Here, L denotes the p-norm size of the attack against which the PixelDP network provides (ϵ, δ)-DP; we call it the *construction attack bound*. The noise formulas show that for a fixed noise standard deviation σ, the guarantee degrades gracefully: attacks twice as big halve the ϵ in the DP guarantee ($L \leftarrow 2L \Rightarrow \epsilon \leftarrow 2\epsilon$). This property is often referred as group privacy in the DP literature [18].

Computing the sensitivity of the pre-noise function g depends on where we choose to place the noise layer in the DNN. Because the post-processing property of DP carries the (ϵ, δ)-PixelDP guarantee from the noise layer through the end of the network, a DNN designer has great flexibility in placing the noise layer anywhere in the DNN, as long as no skip connection exists from pre-noise to post-noise layers. We discuss here several options for noise layer placement and how to compute sensitivity for each. Our methods are not closely tied to particular network architectures and can therefore be applied on a wide variety of networks.

Option 1: Noise in the Image. The most straightforward placement of the noise layer is right after the input layer, which is equivalent to adding noise to individual pixels of the image. This case makes sensitivity analysis trivial: g is the identity function, $\Delta_{1,1} = 1$, and $\Delta_{2,2} = 1$.

Option 2: Noise after First Layer. Another option is to place the noise after the first hidden layer, which is usually simple and standard for many DNNs. For example, in image classification, networks often start with a convolution layer. In other cases, DNNs start with fully connected layer. These linear initial layers can be analyzed and their sensitivity computed as follows.

For linear layers, which consist of a linear operator with matrix form $W \in \mathbb{R}^{m,n}$, the sensitivity is the matrix norm, defined as: $\|W\|_{p,q} = \sup_{x:\|x\|_p \leq 1} \|Wx\|_q$. Indeed, the definition and linearity of W directly imply that $\frac{\|Wx\|_q}{\|x\|_p} \leq$

$\|W\|_{p,q}$, which means that: $\Delta_{p,q} = \|W\|_{p,q}$. We use the following matrix norms [64]: $\|W\|_{1,1}$ is the maximum 1-norm of W's columns; $\|W\|_{1,2}$ is the maximum 2-norm of W's columns; and $\|W\|_{2,2}$ is the maximum singular value of W. For ∞-norm attacks, we need to bound $\|W\|_{\infty,1}$ or $\|W\|_{\infty,2}$, as our DP mechanisms require $q \in \{1, 2\}$. However, tight bounds are computationally hard, so we currently use the following bounds: $\sqrt{n}\|W\|_{2,2}$ or $\sqrt{m}\|W\|_{\infty,\infty}$ where $\|W\|_{\infty,\infty}$ is the maximum 1-norm of W's rows. While these bounds are suboptimal and lead to results that are not as good as for 1-norm or 2-norm attacks, they allow us to include ∞-norm attacks in our frameworks. We leave the study of better approximate bounds to future work.

For a convolution layer, which is linear but usually not expressed in matrix form, we reshape the input (e.g. the image) as an $\mathbb{R}^{nd_{in}}$ vector, where n is the input size (e.g. number of pixels) and d_{in} the number of input channels (e.g. 3 for the RGB channels of an image). We write the convolution as an $\mathbb{R}^{nd_{out} \times nd_{in}}$ matrix where each column has all filter maps corresponding to a given input channel, and zero values. This way, a "column" of a convolution consists of all coefficients in the kernel that correspond to a single input channel. Reshaping the input does not change sensitivity.

Option 3: Noise Deeper in the Network. One can consider adding noise later in the network using the fact that when applying two functions in a row $f_1(f_2(x))$ we have: $\Delta_{p,q}^{(f_1 \circ f_2)} \leq \Delta_{p,r}^{(f_2)} \Delta_{r,q}^{(f_1)}$. For instance, ReLU has a sensitivity of 1 for $p, q \in \{1, 2, \infty\}$, hence a linear layer followed by a ReLU has the same bound on the sensitivity as the linear layer alone. However, we find that this approach for sensitivity analysis is difficult to generalize. Combining bounds in this way leads to looser and looser approximations. Moreover, layers such as batch normalization [28], which are popular in image classification networks, do not appear amenable to such bounds (indeed, they are assumed away by some previous defenses [12]). Thus, our general recommendation is to add the DP noise layer early in the network – where bounding the sensitivity is easy – and taking advantage of DP's post-processing property to carry the sensitivity bound through the end of the network.

Option 4: Noise in Auto-encoder. Pushing this reasoning further, we uncover an interesting placement possibility that underscores the broad applicability and flexibility of our approach: adding noise "before" the DNN in a *separately trained auto-encoder*. An auto-encoder is a special form of DNN trained to predict its own input, essentially learning the identity function $f(x) = x$. Auto-encoders are typically used to de-noise inputs [60], and are thus a good fit for PixelDP. Given an image dataset, we can train a (ϵ, δ)-PixelDP auto-encoder using the previous noise layer options. We stack it before the predictive DNN doing the classification and fine-tune the predictive DNN by running a few training steps on the combined auto-encoder and DNN. Thanks to the decidedly useful post-processing property of DP, the stacked DNN and auto-encoder are (ϵ, δ)-PixelDP.

This approach has two advantages. First, the auto-encoder can be developed independently of the DNN, separating the concerns of learning a good PixelDP model and a good predictive DNN. Second, PixelDP auto-encoders are much smaller than predictive DNNs, and are thus much faster to train. We leverage this property to train the first certified model for the large ImageNet dataset, using an auto-encoder and the *pre-trained* Inception-v3 model, a substantial relief in terms of experimental work (§IV-A).

C. Training Procedure

The soundness of PixelDP's certifications rely only on enforcing DP at prediction time. Theoretically, one could remove the noise layer during training. However, doing so results in near-zero certified accuracy in our experience. Unfortunately, training with noise anywhere except in the image itself raises a new challenge: left unchecked the training procedure will scale up the sensitivity of the pre-noise layers, voiding the DP guarantees.

To avoid this, we alter the pre-noise computation to keep its sensitivity constant (e.g. $\Delta_{p,q} \leq 1$) during training. The specific technique we use depends on the type of sensitivity we need to bound, i.e. on the values of p and q. For $\Delta_{1,1}$, $\Delta_{1,2}$, or $\Delta_{\infty,\infty}$, we normalize the columns, or rows, of linear layers and use the regular optimization process with fixed noise variance. For $\Delta_{2,2}$, we run the projection step described in [12] after each gradient step from the stochastic gradient descent (SGD). This makes the pre-noise layers Parseval tight frames, enforcing $\Delta_{2,2} = 1$. For the pre-noise layers, we thus alternate between an SGD step with fixed noise variance and a projection step. Subsequent layers from the original DNN are left unchanged.

It is important to note that during training, we optimize for *a single draw of noise* to predict the true label for a training example x. We estimate $\mathbb{E}(A(x))$ using multiple draws of noise only at prediction time. We can interpret this as pushing the DNN to increase the margin between the expected score for the true label versus others. Recall from Equation (4) that the bounds on predicted outputs give robustness only when the true label has a large enough margin compared to other labels. By pushing the DNN to give high scores to the true label k at points around x likely under the noise distribution, we increase $\mathbb{E}(A_k(x))$ and decrease $\mathbb{E}(A_{i \neq k}(x))$.

D. Certified Prediction Procedure

For a given input x, the prediction procedure in a traditional DNN chooses the $\arg\max$ label based on the score vector obtained from a single execution of the DNN's deterministic scoring function, $Q(x)$. In a PixelDP network, the prediction procedure differs in two ways. First, it chooses the $\arg\max$ label based on a Monte Carlo estimation of the expected value of the randomized DNN's scoring function, $\hat{\mathbb{E}}(A(x))$. This estimation is obtained by invoking $A(x)$ multiple times with independent draws in the noise layer. Denote $a_{k,n}(x)$ the n^{th} draw from the distribution of the randomized function A on the k^{th} label, given x (so $a_{k,n}(x) \sim A_k(x)$). In Lemma 1 we replace $\mathbb{E}(A_k(x))$ with $\hat{\mathbb{E}}(A_k(x)) = \frac{1}{n}\sum_n a_{k,n}(x)$, where n is the number of invocations of $A(x)$. We compute η-confidence error bounds to account for the estimation error in our robustness bounds, treating each label's score as a random variable in $[0,1]$. We use Hoeffding's inequality [25] or Empirical Bernstein bounds [39] to bound the error in $\hat{\mathbb{E}}(A(x))$. We then apply a union bound so that the bounds for each label are all valid together. For instance, using Hoeffding's inequality, with probability η, $\hat{\mathbb{E}}^{lb}(A(x)) \triangleq \hat{\mathbb{E}}(A(x)) - \sqrt{\frac{1}{2n}ln(\frac{2k}{1-\eta})} \leq \mathbb{E}(A(x)) \leq \hat{\mathbb{E}}(A(x)) + \sqrt{\frac{1}{2n}ln(\frac{2k}{1-\eta})} \triangleq \hat{\mathbb{E}}^{ub}(A(x))$.

Second, PixelDP returns not only the prediction for x $(\arg\max(\hat{\mathbb{E}}(A(x))))$ but also a *robustness size certificate* for that prediction. To compute the certificate, we extend Proposition 1 to account for the measurement error:

Proposition 2. (Generalized Robustness Condition) *Suppose A satisfies (ϵ, δ)-PixelDP with respect to changes of size L in p-norm metric. Using the notation from Proposition 1 further let $\hat{\mathbb{E}}^{ub}(A_i(x))$ and $\hat{\mathbb{E}}^{lb}(A_i(x))$ be the η-confidence upper and lower bound, respectively, for the Monte Carlo estimate $\hat{\mathbb{E}}(A_i(x))$. For any input x, if for some $k \in \mathcal{K}$,*

$$\hat{\mathbb{E}}^{lb}(A_k(x)) > e^{2\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + (1+e^{\epsilon})\delta,$$

then the multiclass classification model based on label probabilities $(\hat{\mathbb{E}}(A_1(x)), \ldots, \hat{\mathbb{E}}(A_K(x)))$ is robust to attacks of p-norm L on input x with probability $\geq \eta$.

The proof is similar to the one for Proposition 1 and is detailed in Appendix A. Note that the DP bounds are not probabilistic even for $\delta > 0$; the failure probability $1-\eta$ comes from the Monte Carlo estimate and can be made arbitrarily small with more invocations of $A(x)$.

Thus far, we have described PixelDP certificates as binary with respect to a fixed attack bound, L: we either meet or do not meet a robustness check for L. In fact, our formalism allows for a more nuanced certificate, which gives the *maximum attack size* L_{max} (measured in p-norm) against which the prediction on input x is guaranteed to be robust: no attack within this size from x will be able to change the highest probability. L_{max} can differ for different inputs. We compute the robustness size certificate for input x as follows. Recall from III-B that the DP mechanisms have a noise standard deviation σ that grows in $\frac{\Delta_{p,q}L}{\epsilon}$. For a given σ used at prediction time, we solve for the maximum L for which the robustness condition in Proposition 2 checks out:

$L_{max} = \max_{L\in\mathbb{R}^+} L$ such that
$\hat{\mathbb{E}}^{lb}(A_k(x)) > e^{2\epsilon}\hat{\mathbb{E}}^{ub}(A_{i:i\neq k}(x)) + (1+e^{\epsilon})\delta$ AND either
- $\sigma = \Delta_{p,1}L/\epsilon$ and $\delta = 0$ (for Laplace) OR
- $\sigma = \sqrt{2\ln(1.25/\delta)}\Delta_{p,2}L/\epsilon$ and $\epsilon \leq 1$ (for Gaussian).

The prediction on x is robust to attacks up to L_{max}, so we award a robustness size certificate of L_{max} for x.

We envision two ways of using robustness size certifications. First, when it makes sense to only take actions on the subset of robust predictions (e.g., a human can intervene for the rest), an application can use PixelDP's certified robustness on each prediction. Second, when all points must be classified, PixelDP gives a lower bound on the accuracy under attack. Like in regular ML, the testing set is used as a proxy for the accuracy on new examples. We can certify the minimum accuracy under attacks up to a threshold size T, that we call the *prediction robustness threshold*. T is an inference-time parameter that can differ from the *construction attack bound* parameter, L, that is used to configure the standard deviation of the DP noise. In this setting the certification is computed only on the testing set, and is not required for each prediction. We only need the highest probability label, which requires fewer noise draws. §IV-E shows that in practice a few hundred draws are sufficient to retain a large fraction of the certified predictions, while a few dozen are needed for simple predictions.

IV. Evaluation

We evaluate PixelDP by answering four key questions:

Q1: How does DP noise affect model accuracy?
Q2: What accuracy can PixelDP certify?
Q3: What is PixelDP's accuracy under attack and how does it compare to that of other best-effort and certified defenses?
Q4: What is PixelDP's computational overhead?

We answer these questions by evaluating PixelDP on five standard image classification datasets and networks – both large and small – and comparing it with one prior certified defense [65] and one best-effort defense [37]. §IV-A describes the datasets, prior defenses, and our evaluation methodology; subsequent sections address each question in turn.

Evaluation highlights: PixelDP provides meaningful certified robustness bounds for reasonable degradation in model accuracy on all datasets and DNNs. To the best of our knowledge, these include the first certified bounds for large, complex datasets/networks such as the Inception network on ImageNet and Residual Networks on CIFAR-10. There, PixelDP gives 60% certified accuracy for 2-norm attacks up to 0.1 at the cost of 8.5 and 9.2 percentage-point accuracy degradation respectively. Comparing PixelDP to the prior certified defense on smaller datasets, PixelDP models give higher accuracy on clean examples (e.g., 92.9% vs. 79.6% accuracy SVHN dataset), and higher robustness to 2-norm attacks (e.g., 55% vs. 17% accuracy on SVHN for 2-norm attacks of 0.5), thanks to the ability to scale to larger models. Comparing PixelDP to the best-effort defense on larger models and datasets, PixelDP matches its accuracy (e.g., 87% for PixelDP vs. 87.3% on CIFAR-10) and robustness to 2-norm bounded attacks.

A. Methodology

Datasets. We evaluate PixelDP on image classification tasks from five pubic datasets listed in Table I. The datasets are listed in descending order of size and complexity for classification tasks. MNIST [69] consists of greyscale handwritten digits and is the easiest to classify. SVHN [44] contains small, real-world digit images cropped from Google Street View photos of house numbers. CIFAR-10 and CIFAR-100 [33] consist of small color images that are each centered on one object of one of 10 or 100 classes, respectively. ImageNet [13] is a large, production-scale image dataset with over 1 million images spread across 1,000 classes.

Models: Baselines and PixelDP. We use existing DNN architectures to train a high-performing baseline model for each dataset. Table I shows the accuracy of the baseline models. We then make each of these networks PixelDP with regards to 1-norm and 2-norm bounded attacks. We also did rudimentary evaluation of ∞-norm bounded attacks, shown in Appendix D. While the PixelDP formalism can support ∞-norm attacks, our results show that tighter bounds are needed to achieve a practical defense. We leave the development and evaluation of these bounds for future work.

Table II shows the PixelDP configurations we used for the 1-norm and 2-norm defenses. The code is available at https://github.com/columbia/pixeldp. Since most of this section focuses on models with 2-norm attack bounds, we detail only those configurations here.

ImageNet: We use as baseline a pre-trained version of Inception-v3 [55] available in Tensorflow [22]. To make it PixelDP, we use the autoencoder approach from §III-B, which does not require a full retraining of Inception and was instrumental in our support of ImageNet. The encoder has three convolutional layers and tied encoder/decoder weights. The convolution kernels are $10 \times 10 \times 32$, $8 \times 8 \times 32$, and $5 \times 5 \times 64$, with stride 2. We make the autoencoder PixelDP by adding the DP noise after the first convolution. We then stack the baseline Inception-v3 on the PixelDP autoencoder and fine-tune it for $20k$ steps, keeping the autoencoder weights constant.

CIFAR-10, *CIFAR-100*, *SVHN*: We use the same baseline architecture, a state-of-the-art Residual Network (ResNet) [70]. Specifically we use the Tensorflow implementation of a 28-10 wide ResNet [57], with the default parameters. To make it PixelDP, we slightly alter the architecture to remove the image standardization step. This step makes sensitivity input dependent, which is harder to deal with in PixelDP. Interestingly, removing this step also increases the baseline's own accuracy for all three datasets. In this section, we therefore report the accuracy of the changed networks as baselines.

MNIST: We train a Convolutional Neural Network (CNN) with two 5×5 convolutions (stride 2, 32 and 64 filters) followed by a 1024 nodes fully connected layer.

Evaluation Metrics. We use two accuracy metrics to evaluate PixelDP models: *conventional accuracy* and *certified accuracy*. Conventional accuracy (or simply accuracy) denotes the fraction of a testing set on which a model is correct; it is the standard accuracy metric used to evaluate any DNN, defended or not. Certified accuracy denotes the fraction of the testing set on which a certified model's predictions are both *correct* and *certified robust* for a given prediction robustness threshold; it has become a standard metric to evaluate models trained with *certified defenses* [65], [52], [16]. We also use *precision on certified examples*, which measures the number of correct predictions exclusively on examples that are certified robust for a given prediction robustness threshold. Formally, the metrics are defined as follows:

1) *Conventional accuracy* $\frac{\sum_{i=1}^{n} isCorrect(x_i)}{n}$, where n is the testing set size and $isCorrect(x_i)$ denotes a function returning 1 if the prediction on test sample x_i returns the correct label, and 0 otherwise.
2) *Certified accuracy* $\frac{\sum_{i=1}^{n}(isCorrect(x_i) \& robustSize(scores,\epsilon,\delta,L) \geq T)}{n}$, where $robustSize(scores, \epsilon, \delta, L)$ returns the certified robustness size, which is then compared to the prediction robustness threshold T.
3) *Precision on certified examples* $\frac{\sum_{i=1}^{n}(isCorrect(x_i) \& robustSize(p_i,\epsilon,\delta,L) \geq T))}{\sum_{i=1}^{n} robustSize(p_i,\epsilon,\delta,L) \geq T)}$.

For $T = 0$ all predictions are robust, so certified accuracy is equivalent to conventional accuracy. Each time we report L or T, we use a $[0, 1]$ pixel range.

Attack Methodology. Certified accuracy – as provided by PixelDP and other certified defense – constitutes a guaranteed lower-bound on accuracy under *any* norm-bounded attack. However, the accuracy obtained in practice when faced with a specific attack can be much better. How much better depends on the attack, which we evaluate in two steps. We first perform an attack on 1,000 randomly picked samples (as is customary in defense evaluation [37]) from the testing set. We then measure conventional accuracy on the attacked test examples.

For our evaluation, we use the state-of-the art attack from Carlini and Wagner [7], that we run for 9 iterations of binary search, 100 gradient steps without early stopping (which we empirically validated to be sufficient), and learning rate 0.01. We also adapt the attack to our specific defense following [2]: since PixelDP adds noise to the DNN, attacks based on optimization may fail due to the high variance of gradients, which would not be a sign of the absence of adversarial examples, but merely of them being harder to find. We address this concern by averaging the gradients over

Dataset	Image size	Training set size	Testing set size	Target labels	Classifier architecture	Baseline accuracy
ImageNet [13]	299x299x3	1.4M	50K	1000	Inception V3	77.5%
CIFAR-100 [33]	32x32x3	50K	10K	100	ResNet	78.6%
CIFAR-10 [33]	32x32x3	50K	10K	10	ResNet	95.5%
SVHN [44]	32x32x3	73K	26K	10	ResNet	96.3%
MNIST [69]	28x28x1	60K	10K	10	CNN	99.2%

Table I: **Evaluation datasets and baseline models.** Last column shows the accuracy of the baseline, undefended models. The datasets are sorted based on descending order of scale or complexity.

p-norm used	DP mechanism	Noise location	Sensitivity approach
1-norm	Laplace	1^{st} conv.	$\Delta_{1,1} = 1$
1-norm	Gaussian	1^{st} conv.	$\Delta_{1,2} = 1$
2-norm	Gaussian	1^{st} conv.	$\Delta_{2,2} \leq 1$
1-norm	Laplace	Autoencoder	$\Delta_{1,1} = 1$
2-norm	Gaussian	Autoencoder	$\Delta_{2,2} \leq 1$

Table II: **Noise layers in PixelDP DNNs.** For each DNN, we implement defenses for different attack bound norms and DP mechanisms.

Dataset	Baseline	$L = 0.03$	$L = 0.1$	$L = 0.3$	$L = 1.0$
ImageNet	77.5%	–	68.3%	57.7%	37.7%
CIFAR-10	95.5%	93.3%	87.0%	70.9%	44.3%
CIFAR-100	78.6%	73.4%	62.4%	44.3%	22.1%
SVHN	96.3%	96.1%	93.1%	79.6%	28.2%
MNIST	99.2%	99.1%	99.1%	98.2%	11%

Table III: **Impact of PixelDP noise on conventional accuracy.** For each DNN, we show different levels of construction attack size L. Conventional accuracy degrades with noise level.

20 noise draws at each gradient step. Appendix §C contains more details about the attack, including sanity checks and another attack we ran similar to the one used in [37].

Prior Defenses for Comparison. We use two state-of-art defenses as comparisons. First, we use the empirical defense model provided by the Madry Lab for CIFAR-10 [38]. This model is developed in the context of ∞-norm attacks. It uses an adversarial training strategy to approximately minimize the worst case error under malicious samples [37]. While inspired by robust optmization theory, this methodology is best effort (see §VI) and supports no formal notion of robustness for individual predictions, as we do in PixelDP. However, the Madry model performs better under the latest attacks than other best-effort defenses (it is in fact the only one not yet broken) [2], and represents a good comparison point.

Second, we compare with another approach for certified robustness against ∞-norm attacks [65], based on robust optimization. This method does not yet scale to the largest datasets (e.g. ImageNet), or the more complex DNNs (e.g. ResNet, Inception) both for computational reasons and because not all necessary layers are yet supported (e.g. BatchNorm). We thus use their largest released model/dataset, namely a CNN with two convolutions and a 100 nodes fully connected layer for the SVHN dataset, and compare their robustness guarantees with our own networks' robustness guarantees. We call this SVHN CNN model *RobustOpt*.

B. Impact of Noise (Q1)

Q1: How does DP noise affect the conventional accuracy of our models? To answer, for each dataset we train up to four $(1.0, 0.05)$-PixelDP DNN, for construction attack bound $L \in \{0.03, 0.1, 0.3, 1\}$. Higher values of L correspond to robustness against larger attacks and larger noise standard deviation σ.

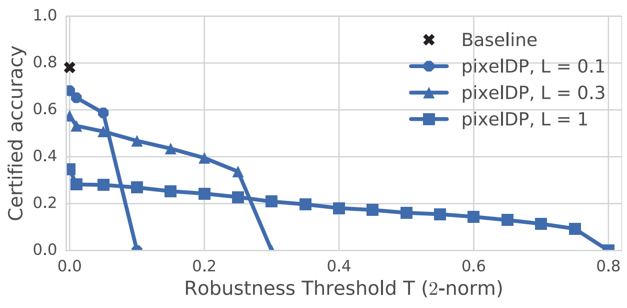
Table III shows the conventional accuracy of these networks and highlights two parts of an answer to Q1. First, at fairly low but meaningful construction attack bound (e.g., $L = 0.1$), all of our DNNs exhibit reasonable accuracy loss – *even on ImageNet*, a dataset on which no guarantees have been made to date! ImageNet: The Inception-v3 model stacked on the PixelDP auto-encoder has an accuracy of 68.3% for $L = 0.1$, which is reasonable degradation compared to the baseline of 77.5% for the unprotected network. CIFAR-10: Accuracy goes from 95.5% without defense to 87% with the $L = 0.1$ defense. For comparison, the Madry model has an accuracy of 87.3% on CIFAR-10. SVHN: our $L = 0.1$ PixelDP network achieves 93.1% conventional accuracy, down from 96.3% for the unprotected network. For comparison, the $L = 0.1$ RobustOpt network has an accuracy of 79.6%, although they use a smaller DNN due to the computationally intensive method.

Second, as expected, constructing the network for larger attacks (higher L) progressively degrades accuracy. ImageNet: Increasing L to 0.3 and then 1.0 drops the accuracy to 57.7% and 37.7%, respectively. CIFAR-10: The ResNet with the least noise ($L = 0.03$) reaches 93.3% accuracy, close to the baseline of 95.5%; increasing noise levels ($L = (0.1, 0.3, 1.0)$) yields 87%, 70.9%, and 37.7%, respectively. Yet, as shown in §IV-D, PixelDP networks trained with fairly low L values (such as $L = 0.1$) already provide meaningful empirical protection against larger attacks.

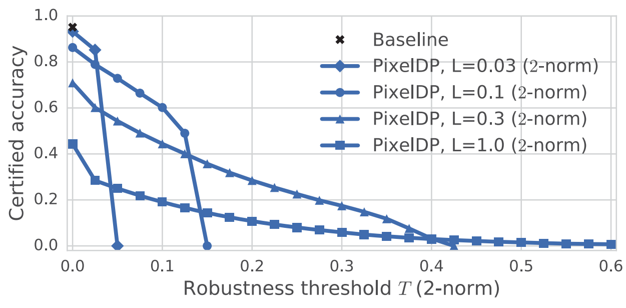
C. Certified Accuracy (Q2)

Q2: What accuracy can PixelDP certify on a test set? Fig. 2 shows the certified robust accuracy bounds for ImageNet and CIFAR-10 models, trained with various values of the construction attack bound L. The certified accuracy is shown as a function of the prediction robustness threshold, T. We make two observations. First, PixelDP yields meaningful robust accuracy bounds even on large networks for ImageNet (see Fig. 2(a)), attesting the scalability of our approach. The $L = 0.1$ network has a certified accuracy of 59% for attacks smaller than 0.09 in 2-norm. The $L = 0.3$ network has a certified accuracy of 40% to attacks up to size 0.2. To our knowledge, PixelDP is the first defense to yield DNNs with certified bounds on accuracy under 2-norm attacks on datasets of ImageNet's size and for large networks like Inception.

Second, PixelDP networks constructed for larger attacks (higher L, hence higher noise) tend to yield higher certified



(a) ImageNet Certified Accuracy



(b) CIFAR-10 Certified Accuracy

Fig. 2: **Certified accuracy, varying the construction attack bound** (L) **and prediction robustness threshold** (T), on ImageNet auto-encoder/Inception and CIFAR-10 ResNet, 2-norm bounds. Robust accuracy at high Robustness thresholds (high T) increases with high-noise networks (high L). Low noise networks are both more accurate and more certifiably robust for low T.

accuracy for high thresholds T. For example, the ResNet on CIFAR-10 (see Fig. 2(b)) constructed with $L = 0.03$ has the highest robust accuracy up to $T = 0.03$, but the ResNet constructed with $L = 0.1$ becomes better past that threshold. Similarly, the $L = 0.3$ ResNet has higher robust accuracy than the $L = 0.1$ ResNet above the 0.14 2-norm prediction robustness threshold.

We ran the same experiments on SVHN, CIFAR-100 and MNIST models but omit the graphs for space reasons. Our main conclusion – that adding more noise (higher L) hurts both conventional and low T certified accuracy, but enhances the quality of its high T predictions – holds in all cases. Appendix B discusses the impact of some design choices on robust accuracy, and Appendix D discusses PixelDP guarantees as compared with previous certified defenses for ∞-norm attacks. While PixelDP does not yet yield strong ∞-norm bounds, it provides meaningful certified accuracy bounds for 2-norm attacks, including on much larger and more complex datasets and networks than those supported by previous approaches.

D. Accuracy Under Attack (Q3)

A standard method to evaluate the strength of a defense is to measure the conventional accuracy of a defended model on malicious samples obtained by running a state-of-the-art attack against samples in a held-out testing set [37]. We apply this method to answer three aspects of question *Q3: (1) Can PixelDP help defend complex models on large datasets in practice? (2) How does PixelDP's accuracy under attack compare to state-of-the-art defenses? (3) How does the accuracy under attack change for certified predictions?*

Accuracy under Attack on ImageNet. We first study conventional accuracy under attack for PixelDP models on ImageNet. Fig. 3 shows this metric for 2-norm attacks on the baseline Inception-v3 model, as well as three defended versions, with a stacked PixelDP auto-encoder trained with construction attack bound $L \in \{0.1, 0.3, 1.0\}$. PixelDP makes the model significantly more robust to attacks. For attacks of size $L_{attack} = 0.5$, the baseline model's accuracy

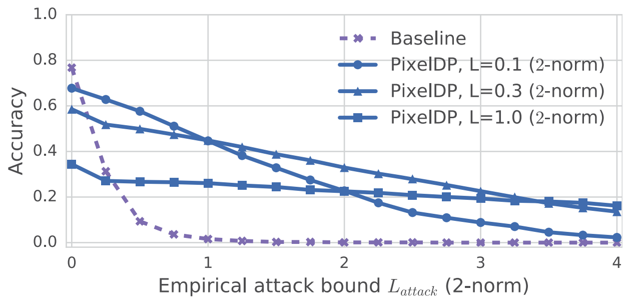
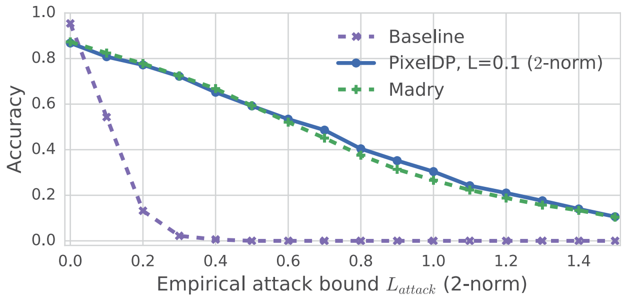


Fig. 3: **Accuracy under attack on ImageNet.** For the ImageNet auto-encoder plus Inception-v3, $L \in \{0.1, 0.3, 1.0\}$ 2-norm attacks. The PixelDP auto-encoder increases the robustness of Inception against 2-norm attacks.

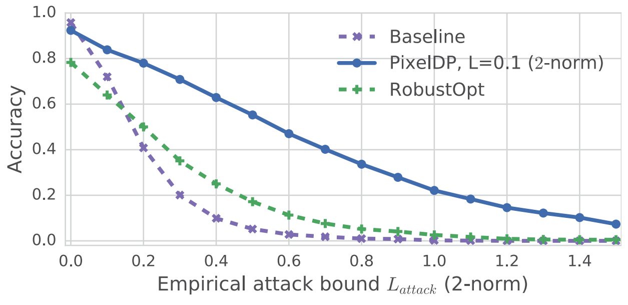
drops to 11%, whereas the $L = 0.1$ PixelDP model's accuracy remains above 60%. At $L_{attack} = 1.5$, the baseline model has an accuracy of 0, but the $L = 0.1$ PixelDP is still at 30%, while the $L = 0.3$ PixelDP model have more that 39% accuracy.

Accuracy under Attack Compared to Madry. Fig. 4(a) compares conventional accuracy of a PixelDP model to that of a Madry model on CIFAR-10, as the empirical attack bound increases for 2-norm attacks. For 2-norm attacks, our model achieves conventional accuracy on par with, or slightly higher than, that of the Madry model. Both models are dramatically more robust under this attack compared to the baseline (undefended) model. For ∞-norm attacks our model does not fare as well, which is expected as the PixelDP model is trained to defend against 2-norm attacks, while the Madry model is optimized for ∞-norm attacks. For $L_{attack} = 0.01$, PixelDP's accuracy is 69%, 8 percentage points lower than Madry's. The gap increases until PixelDP arrives at 0 accuracy for $L_{attack} = 0.06$, with Madry still having 22%. Appendix §D details this evaluation.

Accuracy under Attack Compared to RobustOpt. Fig. 4(b) shows a similar comparison with the RobustOpt defense [65], which provides certified accuracy bounds for ∞-norm attacks. We use the SVHN dataset for the comparison as the RobustOpt defense has not yet been applied to larger



(a) CIFAR-10



(b) SVHN

Fig. 4: **Accuracy under 2-norm attack for PixelDP vs. Madry and RobustOpt**, CIFAR-10 and SVHN. For 2-norm attacks, PixelDP is on par with Madry until $L_{attack} \geq 1.2$; RobustOpt support only small models, and has lower accuracy.

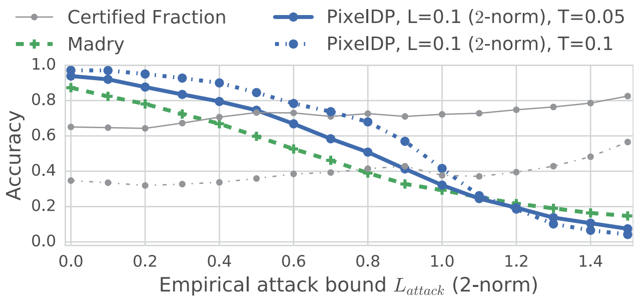


Fig. 5: **PixelDP certified predictions vs. Madry accuracy, under attack**, CIFAR-10 ResNets, 2-norm attack. PixelDP makes fewer but more correct predictions up to $L_{attack} = 1.0$.

datasets. Due to our support of larger DNN (ResNet), PixelDP starts with higher accuracy, which it maintains under 2-norm attacks. For attacks of $L_{attack} = 0.5$, RobustOpt is below 20% accuracy, and PixelDP above 55%. Under ∞-norm attacks, the behavior is different: PixelDP has the advantage up to $L_{attack} = 0.015$ (58.8% to 57.1%), and RobustOpt is better thereafter. For instance, at $L_{attack} = 0.03$, PixelDP has 22.8% accuracy, to RobustOpt's 32.7%. Appendix §D details the ∞-norm attack evaluation.

Precision on Certified Predictions Under Attack. Another interesting feature of PixelDP is its ability to make certifiably robust predictions. We compute the accuracy of these certified predictions under attack – which we term *robust precision* – and compare them to predictions of the Madry network that do not provide such a certification. Fig. 5 shows the results of considering only predictions with a certified robustness above 0.05 and 0.1. It reflects the benefit to be gained by applications that can leverage our theoretical guarantees to filter out non-robust predictions. We observe that PixelDP's robust predictions are *substantially more correct* than Madry's predictions up to an empirical attack bound of 1.1. For $T = 0.05$ PixelDP's robust predictions are 93.9% accurate, and up to 10 percentage points more correct under attack for $L_{attack} \leq 1.1$. A robust prediction is given for above 60% of the data points. The more conservative the robustness test is (higher T), the more correct PixelDP's predictions are, although it makes fewer of them (Certified Fraction lines).

Thus, for applications that can afford to not act on a minority of the predictions, PixelDP's robust predictions under 2-norm attack are substantially more precise than Madry's. For applications that need to act on every prediction, PixelDP offers on-par accuracy under 2-norm attack to Madry's. Interestingly, although our defense is trained for 2-norm attacks, the first conclusion still holds for ∞-norm attacks; the second (as we saw) does not.

E. Computational Overhead (Q4)

Q4: What is PixelDP's computational overhead? We evaluate overheads for training and prediction. PixelDP adds little overhead for *training*, as the only additions are a random noise tensor and sensitivity computations. On our GPU, the CIFAR-10 ResNet baseline takes on average $0.65s$ per training step. PixelDP versions take at most $0.66s$ per training step (1.5% overhead). This represents a significant benefit over adversarial training (e.g. Madry) that requires finding good adversarial attacks for each image in the mini-batch at each gradient step, and over robust optimization (e.g. RobustOpt) that requires solving a constrained optimization problem at each gradient step. The low training overhead is instrumental to our support of large models and datasets.

PixelDP impacts *prediction* more substantially, since it uses multiple noise draws to estimate the label scores. Making a prediction for a single image with 1 noise draw takes $0.01s$ on average. Making 10 draws brings it only to $0.02s$, but 100 requires $0.13s$, and 1000, $1.23s$. It is possible to use Hoeffding's inequality [25] to bound the number of draws necessary to distinguish the highest score with probability at least η, given the difference between the top two scores $y_{max} - y_{second-max}$. Empirically, we found that 300 draws were typically necessary to properly certify a prediction, implying a prediction time of $0.42s$ seconds, a $42\times$ overhead. This is parallelizable, but resource consumption is still substantial. To make simple predictions – distinguish the top label when we must make a prediction on all inputs – 25 draws are enough in practice, reducing the overhead to $3\times$.

V. Analysis

We make three points about PixelDP's guarantees and applicability. First, we emphasize that our Monte Carlo approximation of the function $x \mapsto \mathbb{E}(A(x))$ is *not* intended to be a DP procedure. Hence, there is no need to apply composition rules from DP, because we do not need this randomized procedure to be DP. Rather, the Monte Carlo approximation $x \mapsto \hat{\mathbb{E}}(A(x))$ is just that: an approximation to a function $x \mapsto \mathbb{E}(A(x))$ whose robustness guarantees come from Lemma 1. The function $x \mapsto \hat{\mathbb{E}}(A(x))$ does not satisfy DP, but because we can control the Monte Carlo estimation error using standard tools from probability theory, it is also robust to small changes in the input, just like $x \mapsto \mathbb{E}(A(x))$.

Second, Proposition 1 is not a high probability result; it is valid with probability 1 even when A is $(\epsilon, \delta > 0)$-DP. The δ parameter can be thought of as a "failure probability" of an (ϵ, δ)-DP mechanism: a chance that a small change in input will cause a big change in the probability of some of its outputs. However, since we know that $A_k(x) \in [0, 1]$, the worst-case impact of such failures on the expectation of the output of the (ϵ, δ)-DP mechanism is *at most* δ, as proven in Lemma 1. Proposition 1 explicitly accounts for this worst-case impact (term $(1 + e^\epsilon)\delta$ in Equation (4)).

Were we able to compute $\mathbb{E}(A(x))$ analytically, PixelDP would output deterministic robustness certificates. In practice however, the exact value is too complex to compute, and hence we approximate it using a Monte Carlo method. This adds probabilistic measurement error bounds, making the final certification (Proposition 2) a high probability result. However, the uncertainty comes exclusively from the Monte Carlo integration – and can be made arbitrarily small with more runs of the PixelDP DNN – and not from the underlying (ϵ, δ)-DP mechanism A. Making the uncertainty small gives an adversary a small chance to fool a PixelDP network into thinking that its prediction is robust when it is not. The only ways an attacker can increase that chance is by either submitting the same attack payload many times or gaining control over PixelDP's source of randomness.

Third, PixelDP applies to any task for which we can measure changes to input in a meaningful p-norm, and bound the sensitivity to such changes at a given layer in the DNN (e.g. sensitivity to a bounded change in a word frequency vector, or a change of class for categorical attributes). PixelDP also applies to multiclass classification where the prediction procedure returns several top-scoring labels. Finally, Lemma 1 can be extended to apply to DP mechanism with (bounded) output that can also be negative, as shown in Appendix E. PixelDP thus directly applies to DNNs for regression tasks (i.e. predicting a real value instead of a category) as long as the output is bounded (or unbounded if $\delta = 0$). The output can be bounded due to the specific task, or by truncating the results to a large range of values and using a comparatively small δ.

VI. Related Work

Our work relates to a significant body of work in adversarial examples and beyond. Our main contribution to this space is to introduce a new and very different direction for building *certified defenses*. Previous attempts have built on robust optimization theory. In PixelDP we propose a new approach built on differential privacy theory which exhibits a level of flexibility, broad applicability, and scalability that exceeds what robust optimization-based certified defenses have demonstrated. While the most promising way to defend against adversarial examples is still an open question, we observe undebatable benefits unique to our DP based approach, such as the post-processing guarantee of our defense. In particular, the ability to prepend a defense to unmodified networks via a PixelDP auto-encoder, as we did to defend Inception with *no structural changes*, is unique among certified (and best-effort) defenses.

Best-effort Defenses. Defenders have used multiple heuristics to empirically increase DNNs' robustness. These defenses include model distillation [45], automated detection of adversarial examples [24], [42], [41], application of various input transformations [29], [10], randomization [23], [11], and generative models [51], [27], [68]. Most of these defenses have been broken, sometimes months after their publication [7], [6], [2].

The main empirical defense that still holds is Madry et al. [37], based on adversarial training [21]. Madry et al. motivate their approach with robust optimization, a rigorous theory. However not all the assumptions are met, as this approach runs a best-effort attack on each image in the minibatch at each gradient step, when the theory requires finding the best possible adversarial attack. And indeed, finding this worst case adversarial example for ReLU DNNs, used in [37], was proven to be NP-hard in [53]. Therefore, while this defense works well in practice, it gives no theoretical guarantees for individual predictions or for the model's accuracy under attack. PixelDP leverages DP theory to provide guarantees of robustness to arbitrary, norm-based attacks for individual predictions.

Randomization-based defenses are closest in method to our work [23], [11], [35]. For example, Liu et al. [35] randomizes the entire DNN and predicts using an ensemble of multiple copies of the DNN, essentially using draws to roughly estimate the expected $\arg\max$ prediction. They observe empirically that randomization smoothens the prediction function, improving robustness to adversarial examples. However, randomization-based prior work provides limited formalism that is insufficient to answer important defense design questions: where to add noise, in what quantities, and what formal guarantees can be obtained from randomization? The lack of formalism has caused some works [23], [11] to add insufficient amounts of noise (e.g., noise not calibrated to pre-noise sensitivity), which makes them vulnerable to attack [6]. On the contrary, [35] inserts randomness into every layer of the DNN: our work shows that adding the right amount of calibrated noise at a single layer is sufficient to leverage DP's post-processing guarantee and carry the bounds through the end of the network. Our paper formalizes randomization-based defenses using DP theory, and in doing so helps answer many of these design questions. Our formalism also lets us reason about the guarantees obtained through randomization and enables us to elevate randomization-based approaches from the class of best-effort defenses to that of *certified defenses*.

Certified Defenses and Robustness Evaluations. PixelDP offers two functions: (1) a strategy for learning robust models and (2) a method for evaluating the robustness of these models against adversarial examples. Both of these approaches have been explored in the literature. First, several certified defenses modify the neural network training process to minimize the number of robustness violations [65], [52], [12]. These approaches, though promising, do not yet scale to larger networks like Google Inception [65], [52]. In fact, all published certified defenses have been evaluated on small models and datasets [65], [52], [12], [43], and at least in one case, the authors directly acknowledge that some components of their defense would be "completely infeasible" on ImageNet [65]. A recent paper [16] presents a certified defense evaluated on the CIFAR-10 dataset [33] for multi-layer DNNs (but smaller than ResNets). Their approach is completely different from ours and, based on the current results we see no evidence that it can readily scale to large datasets like ImageNet.

Another approach [53] combines robust optimization and adversarial training in a way that gives formal guarantees and has lower computational complexity than previous robust optimization work, hence it has the potential to scale better. This approach requires smooth DNNs (e.g., no ReLU or max pooling) and robustness guarantees are over the expected loss (e.g., log loss), whereas PixelDP can certify each specific prediction, and also provides intuitive metrics like robust accuracy, which is not supported by [53]. Finally, unlike PixelDP, which we evaluated on five datasets of increasing size and complexity, this technique was evaluated only on MNIST, a small dataset that is notoriously amenable to robust optimization (due to being almost black and white). Since the effectiveness of all defenses depends on the model and dataset, it is hard to conclude anything about how well it will work on more complex datasets.

Second, several works seek to formally verify [26], [30], [61], [62], [15], [20], [58] or lower bound [49], [63] the robustness of pre-trained ML models against adversarial attacks. Some of these works scale to large networks [49], [63], but they are insufficient from a defense perspective as they provide no scalable way to train robust models.

Differentially Private ML. Significant work focuses on making ML algorithms DP to preserve the privacy of training sets [40], [1], [9]. PixelDP is orthogonal to these works, differing in goals, semantic, and algorithms. The only thing we share with DP ML (and most other applied DP literature) are DP theory and mechanisms. The goal of DP ML is to learn the parameters of a model while ensuring DP with respect to the training data. Public release of model parameters trained using a DP learning algorithm (such as DP empirical risk minimization or ERM) is guaranteed to not reveal much information about individual training examples. PixelDP's goal is to create a robust predictive model where a small change to any input example does not drastically change the model's prediction on that example. We achieve this by ensuring that the model's scoring function is a DP function with respect to the features of an input example (eg, pixels). DP ML algorithms (e.g., DP ERM) do not necessarily produce models that satisfy PixelDP's semantic, and our training algorithm for producing PixelDP models does not ensure DP of training data.

Previous DP-Robustness Connections. Previous work studies generalization properties of DP [4]. It is shown that *learning algorithms* that satisfy DP with respect to the training data have statistical benefits in terms of out-of-sample performance; or that DP has a deep connection to robustness at the dataset level [14], [17]. Our work is rather different. Our learning algorithm is not DP; rather, the predictor we learn satisfies DP with respect to the atomic units (e.g., pixels) of a given test point.

VII. Conclusion

We demonstrated a connection between robustness against adversarial examples and differential privacy theory. We showed how the connection can be leveraged to develop a certified defense against such attacks that is (1) as effective at defending against 2-norm attacks as today's state-of-the-art best-effort defense and (2) more scalable and broadly applicable to large networks compared to any prior certified defense. Finally, we presented the first evaluation of a certified 2-norm defense on the large-scale ImageNet dataset. In addition to offering encouraging results, the evaluation highlighted the substantial flexibility of our approach by leveraging a convenient autoencoder-based architecture to make the experiments possible with limited resources.

VIII. Acknowledgments

We thank our shepherd, Abhi Shelat, and the anonymous reviewers, whose comments helped us improve the paper significantly. This work was funded through NSF CNS-1351089, CNS-1514437, and CCF-1740833, ONR N00014-17-1-2010, two Sloan Fellowships, a Google Faculty Fellowship, and a Microsoft Faculty Fellowship.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. Brendan McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with

Differential Privacy. *ArXiv e-prints*, 2016.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. 2018.

[3] A. Athalye and I. Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

[4] R. Bassily, K. Nissim, A. Smith, T. Steinke, U. Stemmer, and J. Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 2016.

[5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, 2016.

[6] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017.

[7] N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[8] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi. Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium*, 2013.

[9] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 2011.

[10] Chuan Guo, Mayank Rana, Moustapha Cisse, Laurens van der Maaten. Countering adversarial images using input transformations. *International Conference on Learning Representations*, 2018.

[11] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, Alan Yuille. Mitigating adversarial effects through randomization. *International Conference on Learning Representations*, 2018.

[12] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

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

[14] C. Dimitrakakis, B. Nelson, A. Mitrokotsa, and B. Rubinstein. Bayesian Differential Privacy through Posterior Sampling. *arXiv preprint arXiv:1306.1066v5*, 2016.

[15] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, 2018.

[16] K. Dvijotham, S. Gowal, R. Stanforth, R. Arandjelovic, B. O'Donoghue, J. Uesato, and P. Kohli. Training verified learners with learned verifiers. *ArXiv e-prints*, 2018.

[17] C. Dwork and J. Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009.

[18] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 2014.

[19] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[20] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai 2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy (SP)*, 2018.

[21] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd ICLR*, 2015.

[22] Google. Inception v3. https://github.com/tensorflow/models/tree/master/research/inception. Accessed: 2018.

[23] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. *International Conference on Learning Representations*, 2018.

[24] D. Hendrycks and K. Gimpel. Early methods for detecting adversarial images. In *ICLR (Workshop Track)*, 2017.

[25] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 1963.

[26] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In *Proceedings of the 29th International Conference on Computer Aided Verification*, 2017.

[27] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis. The robust manifold defense: Adversarial training using generative models. *CoRR*, abs/1712.09196, 2017.

[28] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[29] Jacob Buckman, Aurko Roy, Colin Raffel, Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018.

[30] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR*, 2017.

[31] J. Kos, I. Fischer, and D. Song. Adversarial examples for generative models. *arXiv preprint arXiv:1702.06832*, 2017.

[32] Kos, Jernej and Song, Dawn. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

[33] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[34] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint 1607.02533*, 2016.

[35] X. Liu, M. Cheng, H. Zhang, and C. Hsieh. Towards robust neural networks via random self-ensemble. Technical report, 2017.

[36] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *CVPR*, 2017.

[37] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

[38] Madry Lab. CIFAR-10 Adversarial Examples Challenge. https://github.com/MadryLab/cifar10_challenge. Accessed: 1/22/2017.

[39] A. Maurer and M. Pontil. Empirical bernstein bounds and sample-variance penalization. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.

[40] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[41] D. Meng and H. Chen. Magnet: A two-pronged defense against adversarial examples. In *CCS*, 2017.

[42] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. In *Proceedings of the 6th International Conference on Learning Representations*, 2017.

[43] M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning (ICML)*, 2018.

[44] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning.

[45] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. of IEEE Symposium on Security and Privacy (Oakland)*, 2016.

[46] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman. Towards the science of security and privacy in machine learning. *CoRR*, abs/1611.03814, 2016.

[47] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition.

[48] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas. Malware classification with recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015.

[49] J. Peck, J. Roels, B. Goossens, and Y. Saeys. Lower bounds on the robustness to adversarial perturbations. In *Advances in Neural Information Processing Systems*, 2017.

[50] K. Pei, Y. Cao, J. Yang, and S. Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, 2017.

[51] Pouya Samangouei, Maya Kabkab, Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *International Conference on Learning Representations*, 2018.

[52] A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[53] A. Sinha, H. Namkoong, and J. Duchi. Certifying Some Distributional Robustness with Principled Adversarial Training. 2017.

[54] Y. Song, R. Shu, N. Kushman, and S. Ermon. Generative adversarial examples. 2018.

[55] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[56] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

[57] Tensorflow r1.5. Resnet models. https://github.com/tensorflow/models/tree/r1.5/research/resnet, 2017.

[58] V. Tjeng, K. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[59] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. D. McDaniel. Ensemble adversarial training: Attacks and defenses. *CoRR*, abs/1705.07204, 2017.

[60] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 2010.

[61] S. Wang, K. Pei, W. Justin, J. Yang, and S. Jana. Efficient formal safety analysis of neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[62] S. Wang, K. Pei, W. Justin, J. Yang, and S. Jana. Formal security analysis of neural networks using symbolic intervals. *27th USENIX Security Symposium*, 2018.

[63] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

[64] Wikipedia. Operator norm. https://en.wikipedia.org/wiki/Operator_norm, Accessed in 2017.

[65] E. Wong and Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2018.

[66] C. Xiao, B. Li, J. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. 2018.

[67] C. Xiao, J. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. 2018.

[68] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *International Conference on Learning Representations*, 2018.

[69] Yann LeCun, Corinna Cortes, Christopher J.C. Burges. The mnist database of handwritten digits, Accessed in 2017.

[70] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, 2016.

[71] Z. Zohrevand, U. Glässer, M. A. Tayebi, H. Y. Shahir, M. Shirmaleki, and A. Y. Shahir. Deep learning based forecasting of critical infrastructure data. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17. ACM, 2017.

Appendix

A. Proof of Proposition 2

We briefly re-state the Proposition and detail the proof.

Proposition. *Suppose A is (ϵ, δ)-PixelDP for size L in p-norm metric. For any input x, if for some $k \in \mathcal{K}$,*

$$\hat{\mathbb{E}}^{lb}(A_k(x)) > e^{2\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + (1+e^{\epsilon})\delta,$$

then the multiclass classification model based on label probabilities $\hat{\mathbb{E}}(A_k(x))$ is robust to attacks of p-norm L on input x with probability higher than η.

Proof: Consider any $\alpha \in B_p(L)$, and let $x' := x + \alpha$. From Equation (2), we have with $p > \eta$ that

$$\begin{aligned}\hat{\mathbb{E}}(A_k(x')) &\geq (\hat{\mathbb{E}}(A_k(x)) - \delta)/e^{\epsilon} \\ &\geq (\hat{\mathbb{E}}^{lb}(A_k(x)) - \delta)/e^{\epsilon}, \\ \hat{\mathbb{E}}(A_{i:i\neq k}(x')) &\leq e^{\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + \delta, \quad i \neq k.\end{aligned}$$

Starting from the first inequality, and using the hypothesis, followed by the second inequality, we get

$$\begin{aligned}\hat{\mathbb{E}}^{lb}(A_k(x)) > \ & e^{2\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + (1+e^{\epsilon})\delta \Rightarrow \\ \hat{\mathbb{E}}(A_k(x')) &\geq (\hat{\mathbb{E}}^{lb}(A_k(x)) - \delta)/e^{\epsilon} \\ &> e^{\epsilon} \max_{i:i\neq k} \hat{\mathbb{E}}^{ub}(A_i(x)) + \delta \\ &> \hat{\mathbb{E}}(A_{i:i\neq k}(x'))\end{aligned}$$

which is the robustness condition from Equation (1). ■

B. Design Choice

Our theoretical results allow the DP DNN to output any bounded score over labels $A_k(x)$. In the evaluation we used the softmax output of the DNN, the typical "probabilities" that DNNs traditionally output. We also experimented with using $\arg\max$ scores, transforming the probabilities in a zero vector with a single 1 for the highest score label. As each dimension of this vector is in $[0, 1]$, our theory applies as is. We observed that $\arg\max$ scores were a bit less robust empirically (lower accuracy under attack). However, as shown on Fig. 6 $\arg\max$ scores yield a higher certified accuracy. This is both because we can use tighter bounds for measurement error using a Clopper-Pearson interval, and because the $\arg\max$ pushes the expected scores further apart, thus satisfying Proposition 1 more often.

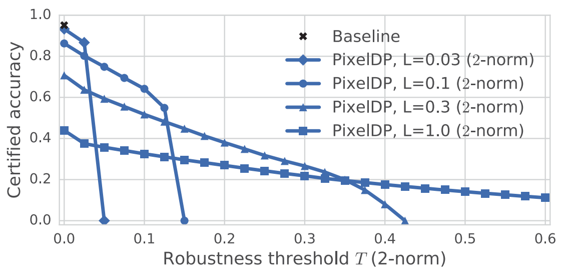


Fig. 6: **Robust Accuracy, $\arg\max$ Scores.** Using $\arg\max$ scores for certification yields better accuracy bounds (see Fig. 2(b)), both because the scores are further appart and because the measurement error bounds are tighter.

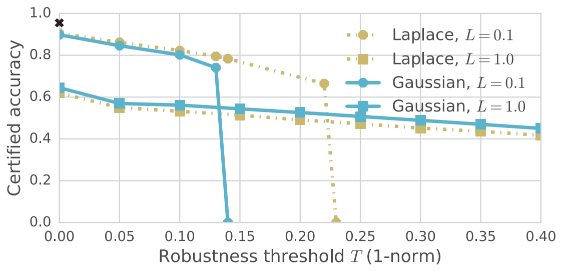


Fig. 7: **Laplace vs. Gaussian.** Certified accuracy for a ResNet on the CIFAR-10 dataset, against 1-norm bounded attacks. The Laplace mechanism yields better accuracy for low noise levels, but the Gaussian mechanism is better for high noise ResNets.

We also study the impact of the DP mechanism used on certified accuracy for 1-norm attacks. Both the Laplace and Gaussian mechanisms can be used after the first convolution, by respectively controlling the $\Delta_{1,1}$ or $\Delta_{1,2}$ sensitivity. Fig. 7 shows that for our ResNet, the Laplace mechanism is better suited to low levels of noise: for $L = 0.1$, it yields a slightly higher accuracy (90.5% against 88.9%), as well as better certified accuracy with a maximum robustness size of 1-norm 0.22 instead of 0.19, and a robust accuracy of 73% against 65.4% at the 0.19 threshold. On the other hand, when adding more noise (e.g. $L = 0.3$), the Gaussian mechanism performs better, consistently yielding a robust accuracy 1.5 percentage point higher.

C. Attack Details

All evaluation results (§IV) are based on the attack from Carlini and Wagner [7], specialized to better attack PixelDP (see parameters and adaptation in §IV-A). We also implemented variants of the iterative Projected Gradient Descent (PGD) attack described in [37], modified to average gradients over 15 noise draws per step, and performing each attack 15 times with a small random initialization. We implemented two version of this PGD attack.

2-norm Attack: The gradients are normalized before applying the step size, to ensure progress even when gradients are close to flat. We perform $k = 100$ gradient steps and select a step size of $\frac{2.5L}{k}$. This heuristic ensures that all feasible points within the 2-norm ball can be reached after k steps. After each step, if the attack is larger than L, we project it on the 2-norm ball by normalizing it. Under this attack, results were qualitatively identical for all experiments. The raw accuracy numbers were a few percentage points higher (i.e. the attack was slightly less efficient), so we kept the results for the Carlini and Wagner attack.

∞-norm Attack: We perform $max(L + 8, 1.5L)$ gradient steps and maintain a constant step of size of 0.003 (which corresponds to the minimum pixel increment in a discrete $[0, 255]$ pixel range). At the end of each gradient step we clip the size of the perturbation to enforce a perturbation within the ∞-norm ball of the given attack size. We used this attack to compare PixelDP with models from Madry and RobustOpt (see results in Appendix D).

Finally, we performed sanity checks suggested in [2]. The authors observe that several heuristic defenses do not ensure the absence of adversarial examples, but merely make them harder to find by obfuscating gradients. This phenomenon, also referred to as gradient masking [46], [59], makes the defense susceptible to new attacks crafted to circumvent that obfuscation [2]. Although PixelDP provides certified accuracy bounds that are *guaranteed* to hold regardless of the attack used, we followed guidelines from [2], to to rule out obfuscated gradients in our empirical results. We verified three properties that can be symptomatic of problematic attack behavior. First, when growing T, the accuracy drops to 0 on all models and datasets. Second, our attack significantly outperforms random sampling. Third, our iterative attack is more powerful than the respective single-step attack.

D. ∞-norm Attacks

As far as ∞-norm attacks are concerned, we acknowledge that the size of the attacks against which our current PixelDP defense can certify accuracy is substantially lower than that of previous certified defenses. Although previous defenses have been demonstrated on MNIST and SVHN only, and for smaller DNNs, they achieve ∞-norm defenses of $T_\infty = 0.1$ with robust accuracy 91.6% [65] and 65% [52] on MNIST. On SVHN, [65] uses $T_\infty = 0.01$, achieving 59.3% of certified accuracy. Using the crude bounds we have between p-norms makes a comparison difficult in both directions. Mapping ∞-norm bounds in 2-norm gives $T_2 \geq T_\infty$, also yielding very small bounds. On the other hand, translating 2-norm guarantees into ∞-norm ones (using that $\|x\|_2 \leq \sqrt{n}\|x\|_\infty$ with n the size of the image) would require a 2-norm defense of size $T_2 = 2.8$ to match the $T_\infty = 0.1$ bound from MNIST, an order of magnitude higher than what we can achieve. As comparison points, our $L = 0.3$ CNN has a robust accuracy of 91.6% at $T = 0.19$ and 65% at $T = 0.39$. We make the same observation on SVHN, where we would need a bound at $T_2 = 0.56$ to match the $T_\infty = 0.01$ bound, but our ResNet with $L = 0.1$ reaches a similar robust accuracy as RobustOpt for $T_2 = 0.1$. This calls for the design ∞-norm specific PixelDP mechanisms that could also scale to larger DNNs and datasets.

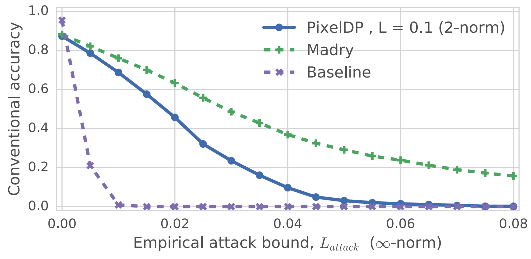


Fig. 8: **Accuracy under ∞-norm attacks for PixelDP and Madry.** The Madry model, explicitly trained against ∞-norm attacks, outperforms PixelDP. The difference increases with the size of the attack.

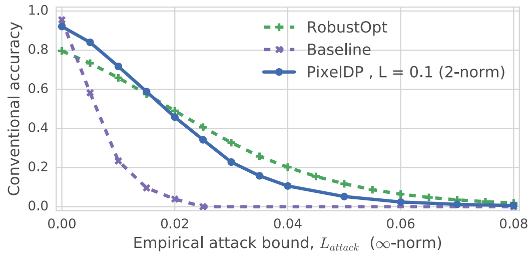


Fig. 9: **Accuracy under ∞-norm attacks for PixelDP and RobustOpt.** PixelDP is better up to $L_\infty = 0.015$, due to its support of larger ResNet models. For attacks of ∞-norm above this value, RobustOpt is more robust.

On Figures 8 and 9, we show PixelDP's accuracy under ∞-norm attacks, compared to the Madry and RobustOpt models, both trained specifically against this type of attacks. On CIFAR-10, the Madry model outperforms PixelDP: for $L_{attack} = 0.01$, PixelDP's accuracy is 69%, 8 percentage points lower than Madry's. The gap increases until PixelDP arrives at 0 accuracy for $L_{attack} = 0.06$, with Madry still having 22%.

On SVHN, against the RobustOpt model, trained with robust optimization against ∞-norm attacks, PixelDP is better up to $L_\infty = 0.015$, due to its support of larger ResNet models. For attacks of ∞-norm above this value, RobustOpt is more robust.

E. Extension to regression

A previous version of this paper contained an incorrect claim in the statement of Lemma 1 for outputs that can be negative. Because the paper focused on classification, where DNN scores are in $[0, 1]$, the error had no impact on the claims or experimental results for classification. Lemma 2, below, provides a correct version of Lemma 1 for outputs that can be negative, showing how PixelDP can be extended to support regression problems:

Lemma 2. (General Expected Output Stability Bound) *Suppose a randomized function A, with bounded output $A(x) \in [a, b]$, $a, b \in \mathbb{R}$, with $a \leq 0 \leq b$, satisfies (ϵ, δ)-DP. Let $A_+(x) = \max(0, A(x))$ and $A_-(x) = -\min(0, A(x))$, so that $A(x) = A_+(x) - A_-(x)$. Then the expected value of its output meets the following property: for all $\alpha \in B_p(1)$,*

$$\mathbb{E}(A(x+\alpha)) \leq e^{\epsilon}\mathbb{E}(A_+(x)) - e^{-\epsilon}\mathbb{E}(A_-(x)) + b\delta - e^{-\epsilon}a\delta,$$
$$\mathbb{E}(A(x+\alpha)) \geq e^{-\epsilon}\mathbb{E}(A_+(x)) - e^{\epsilon}\mathbb{E}(A_-(x)) - e^{-\epsilon}b\delta + a\delta.$$

The expectation is taken over the randomness in A.

Proof: Consider any $\alpha \in B_p(1)$, and let $x' := x + \alpha$. Observe that $\mathbb{E}(A_+(x)) = \int_0^b P(A(x) > t)dt$, so by the (ϵ, δ)-DP property of A via Equation (2), we have $\mathbb{E}(A_+(x')) \leq e^{\epsilon}\mathbb{E}(A_+(x)) + b\delta$ and $\mathbb{E}(A_+(x')) \geq e^{-\epsilon}\mathbb{E}(A_+(x)) - e^{-\epsilon}b\delta$. Similarly, $\mathbb{E}(A_-(x')) \leq e^{\epsilon}\mathbb{E}(A_-(x)) - a\delta$ and $\mathbb{E}(A_-(x')) \geq e^{-\epsilon}\mathbb{E}(A_+(x)) + e^{-\epsilon}a\delta$. Putting these four inequalities together concludes the proof. ■

Following Lemma 2, supporting regression problems involves three steps. First, if the output is unbounded, one must use $(\epsilon, 0)$-DP (e.g. with the Laplace mechanism). If the output is bounded, one may use (ϵ, δ)-DP. The output may be bounded either naturally, because the specific task has inherent output bounds, or by truncating the results to a large range of values and using a comparatively small δ.

Second, instead of estimating the expected value of the randomized prediction function, we estimate both $A_+(x)$ and $A_-(x)$. We can use Hoeffding's inequality [25] or Empirical Bernstein bounds [39] to bound the error.

Third, following Lemma 2, we bound $A_+(x)$ and $A_-(x)$ separately using the DP Expected Output Stability Bound, to obtain a bound on $\mathbb{E}(A(x)) = \mathbb{E}(A_+(x)) - \mathbb{E}(A_-(x))$.