Certified Robustness to Adversarial Examples with Differential Privacy

Vaggelis Atlidakis, Columbia University

With M. Lécuyer (first author), R. Geambasu, D. Hsu, S. Jana

Deep learning

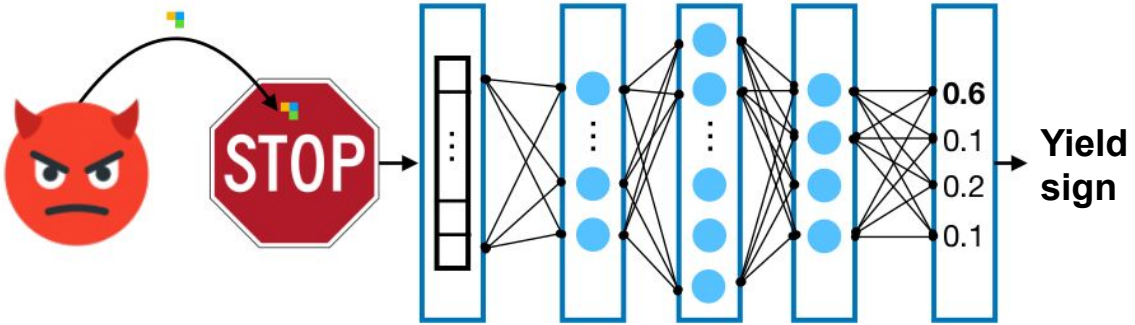
- Deep neural networks (DNNs) deliver exceptional performance on many tasks, including safety-critical applications.
- But DNNs are vulnerable to broad range of attacks.
- Some attacks can cause disastrous drops in accuracy.

Adversarial examples

Adversary finds a tiny change (perturbation) to any correctly classified input that causes misclassification on that input.

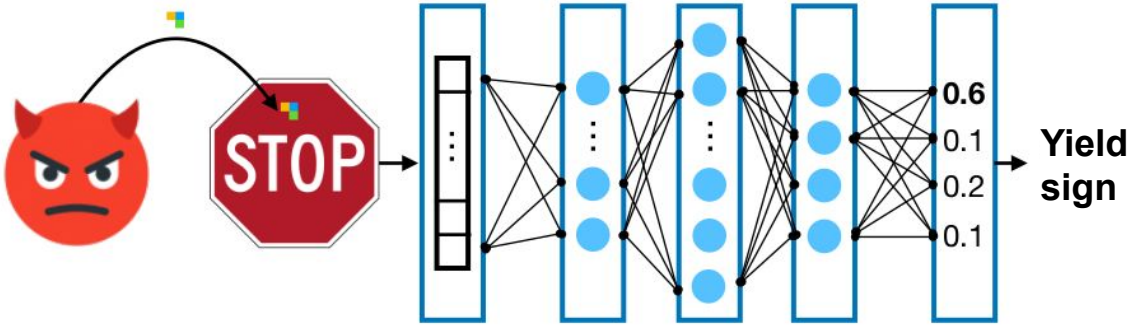
Adversarial examples

Adversary finds a tiny change (perturbation) to any correctly classified input that causes misclassification on that input.



Adversarial examples

Adversary finds a tiny change (perturbation) to any correctly classified input that causes misclassification on that input.

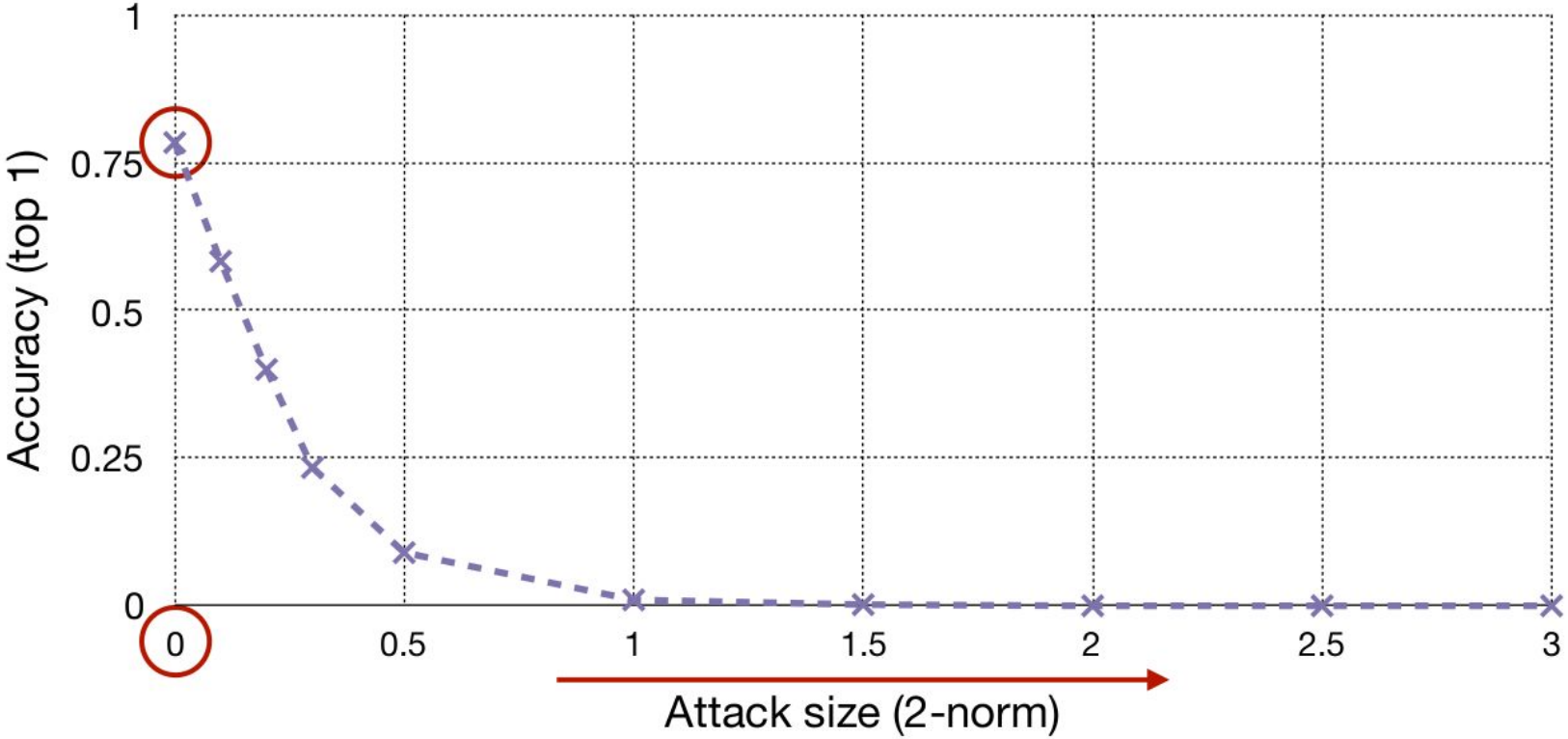


Other examples

- Cast a shadow to fool face recognition.
- Find sequence of actions that avoids credit fraud detection.

Very effective attacks

Attack on Inception-v3/ImageNet:



Questions

Q1: How can you improve accuracy under attack?

Best effort approaches

Evaluate accuracy under attack:

- Launch an attack on examples in a test set.
- Compute model's accuracy on the attacked examples.

Best effort approaches

Evaluate accuracy under attack:

- Launch an attack on examples in a test set.
- Compute model's accuracy on the attacked examples.

Improve accuracy under attack:

- Incorporate adversarial examples in training set.
- Retrain.

Best effort approaches

Evaluate accuracy under attack:

- Launch an attack on examples in a test set.
- Compute model's accuracy on the attacked examples.

Improve accuracy under attack:

- Incorporate adversarial examples in training set.
- Retrain.

Problem: Both steps are attack-specific, leading to an **arms race** between attacker and defenders.

Our goal

Develop a rigorous, attack-independent methodology.

Our goal

Develop a rigorous, attack-independent methodology.

Methodology includes:

1. **Accuracy certification**: lower-bound accuracy under attack on a test set, for any possible attack.
2. **DNN design for certification:** alter DNNs to get good certified accuracy.

Our goal

Develop a rigorous, attack-independent methodology.

Methodology includes:

1. **Accuracy certification**: lower-bound accuracy under attack on a test set, for any possible attack.
2. **DNN design for certification:** alter DNNs to get good certified accuracy.

We've made substantial progress on 1; still working on 2.

PixelDP

Leverages a mechanism from the privacy domain, to implement an accuracy certification method for arbitrary, norm-bounded adversarial example attacks.

PixelDP

Leverages a mechanism from the privacy domain, to implement an accuracy certification method for arbitrary, norm-bounded adversarial example attacks.

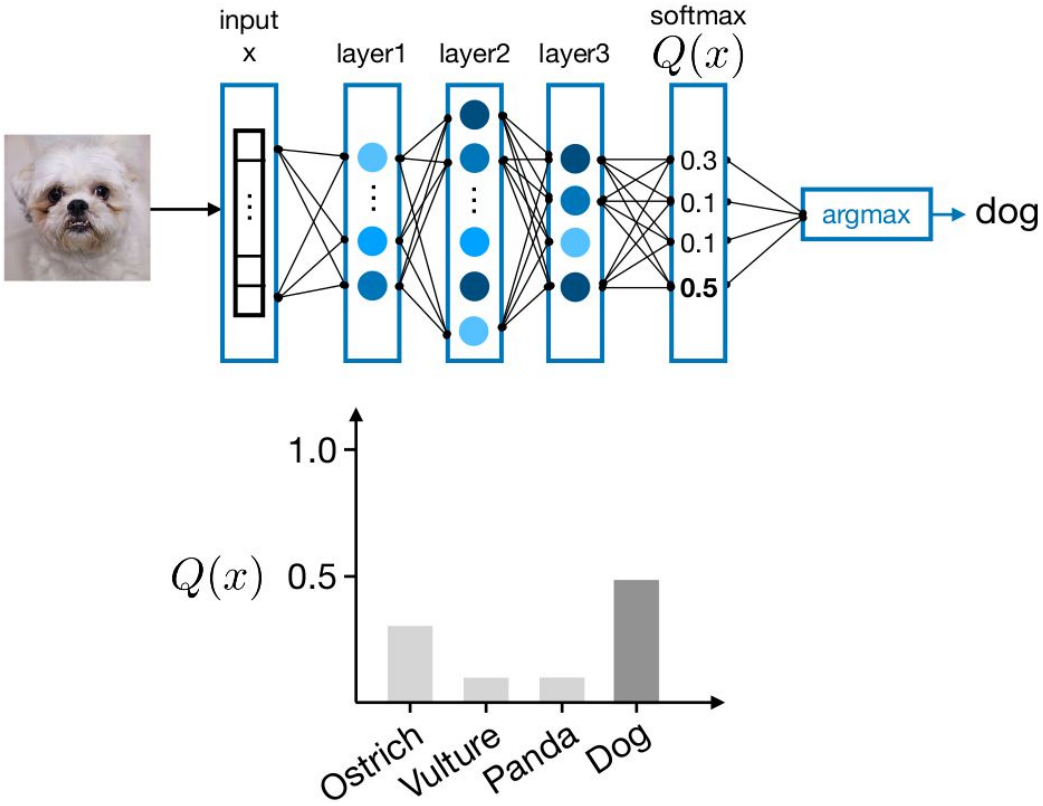
➢ Compared to previous certified defenses, PixelDP offers:

1. **Scalability**: much larger models and datasets.
2. **Flexibility**: supports all architectures.

Outline

- Motivation
- Background
- PixelDP design
- Evaluation

Problem definition

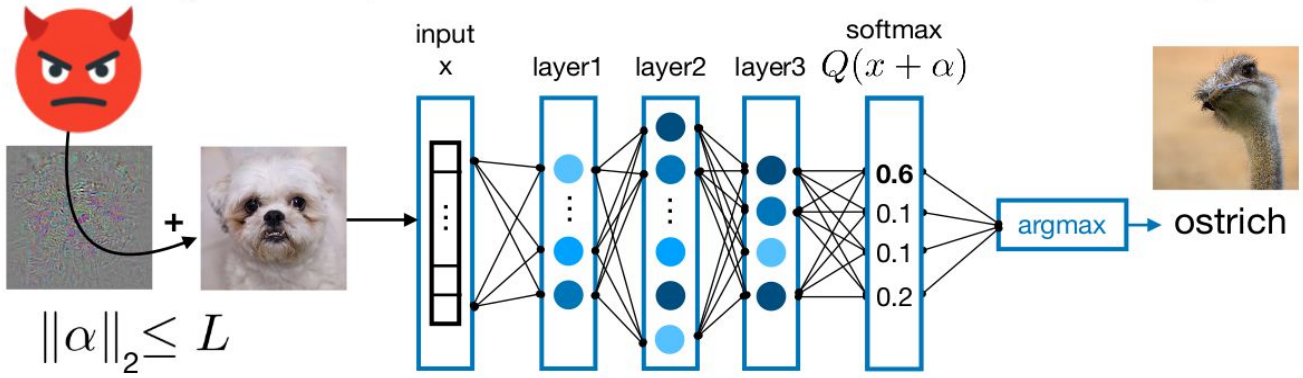


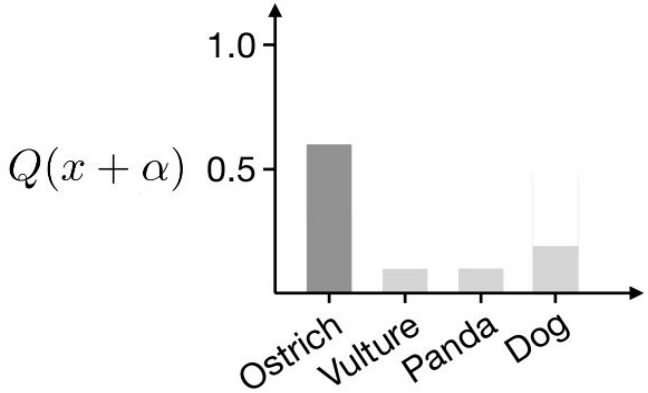
Problem definition

The attacker finds a small change (measures in L2 norm) to a correctly labeled legitimate input that causes the misclassification of that input:

Problem definition

The attacker finds a small change (measures in L2 norm) to a correctly labeled legitimate input that causes the misclassification of that input:

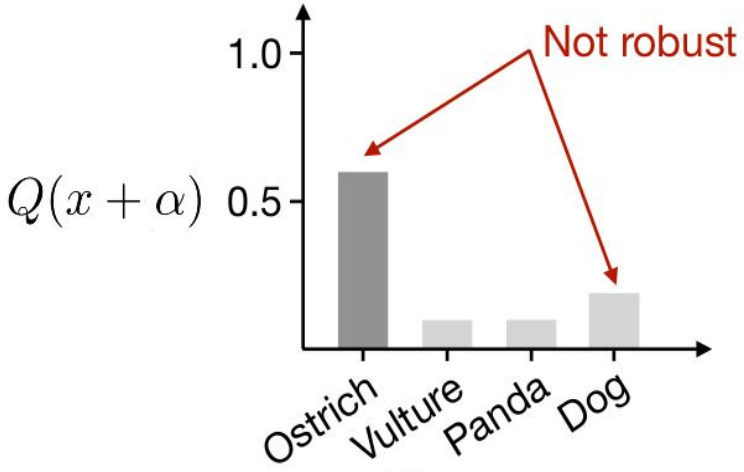




Robustness

$$k = \arg\max_i Q_i(x)$$

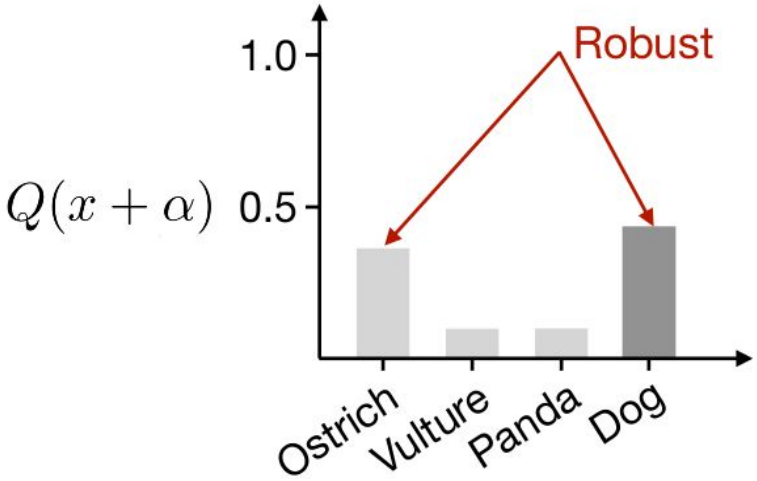
$$\forall \alpha, \|\alpha\| \leq L : Q_k(x+\alpha) \geq \max_{j:j\neq k} Q_j(x+\alpha)$$



Robustness

$$k = \arg\max_i Q_i(x)$$

$$\forall \alpha, \|\alpha\| \leq L : Q_k(x+\alpha) \geq \max_{j:j\neq k} Q_j(x+\alpha)$$



Differential privacy

Differential privacy (DP): A technique to randomize a computation by adding noise, such that changing one data point can only lead to bounded changes in the distribution over the possible outputs.

Differential privacy

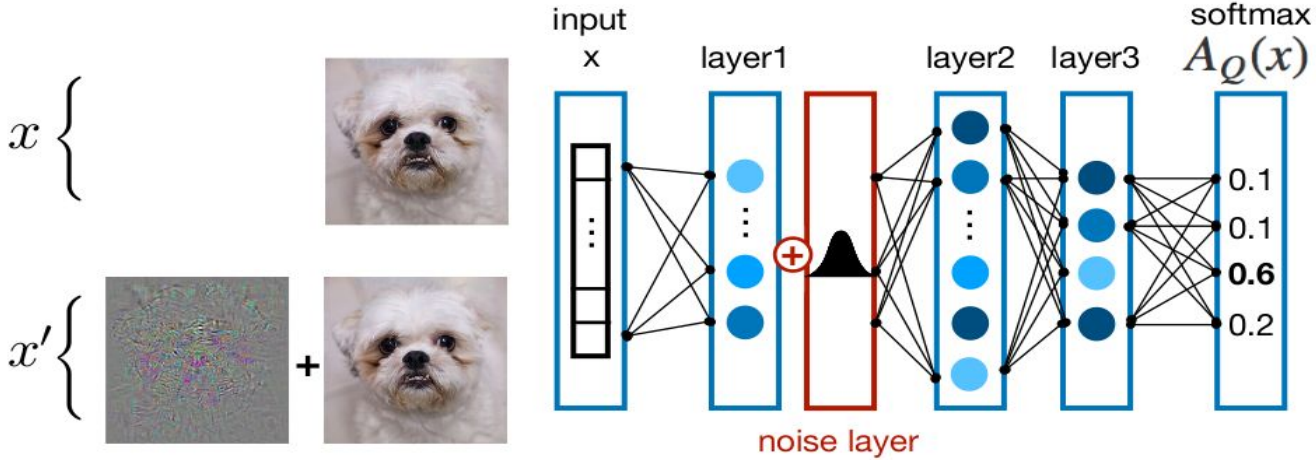
Differential privacy (DP): A technique to randomize a computation by adding noise, such that changing one data point can only lead to bounded changes in the distribution over the possible outputs.

- **DP guarantee**: $P(w(d) \in S) \leq e^{\epsilon} P(w(d') \in S) + \delta$
- **Post-processing property**: any computation on the output of an (e,d)-DP mechanism is still (e, d)-DP.

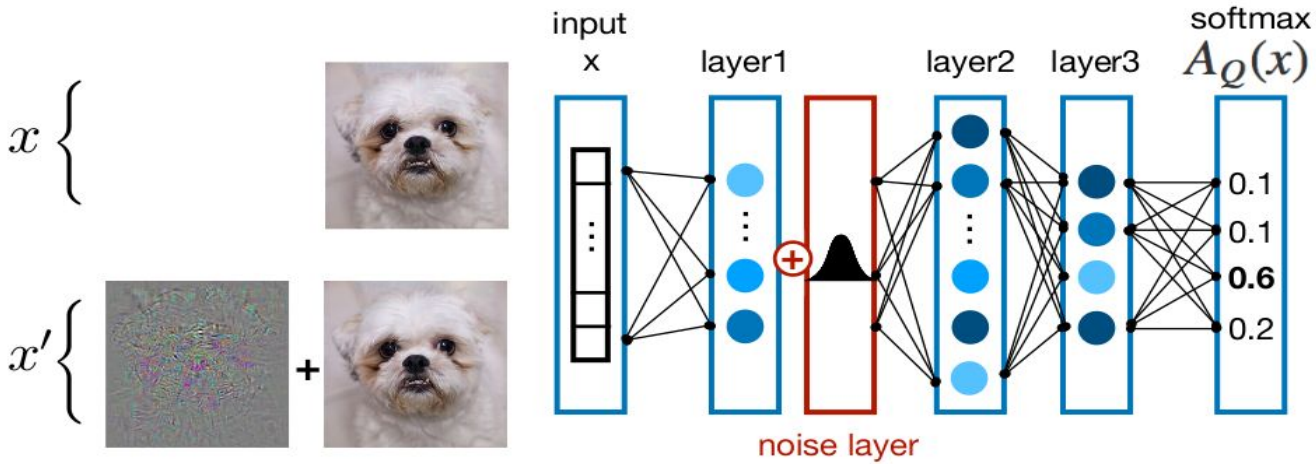
Outline

- Motivation
- Background
- PixelDP design
- Evaluation

PixelDP


x
x'
+
input
x
layer1
layer2
layer3
softmax
$A_Q(x)$
0.1
0.1
0.6
0.2
noise layer

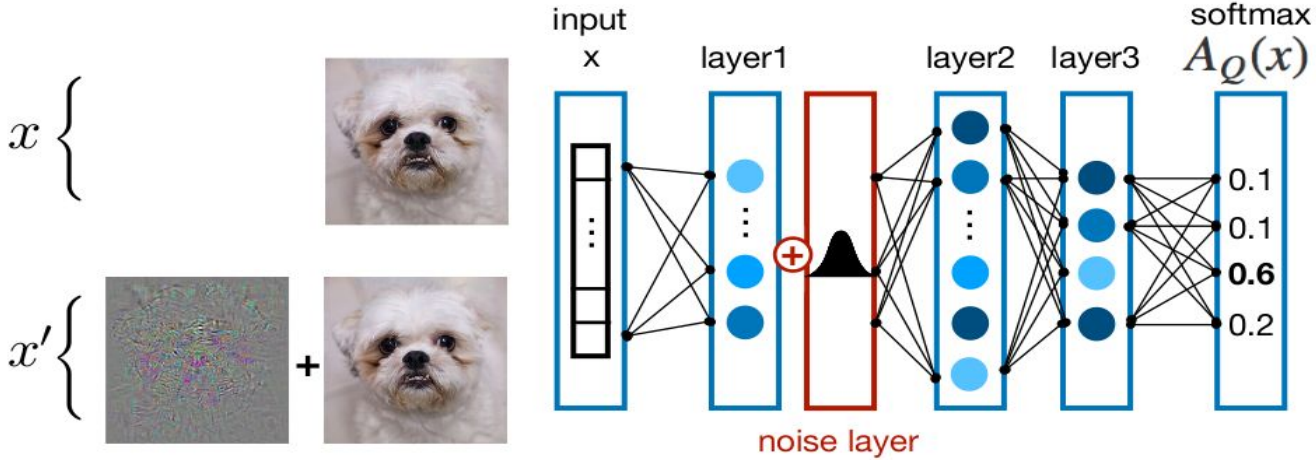
PixelDP



$$P(A_Q(x) \in S) \leq e^{\epsilon} P(A_Q(x') \in S) + \delta$$

The probability to get a given score cannot change too much under an adversarial attack.

PixelDP



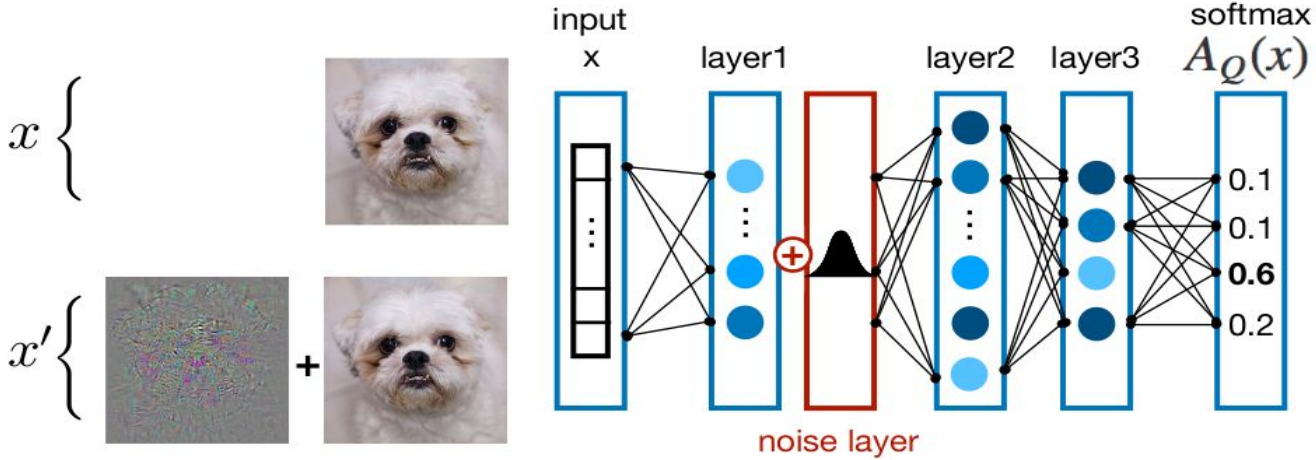
$$P(A_Q(x) \in S) \leq e^{\epsilon} P(A_Q(x') \in S) + \delta$$

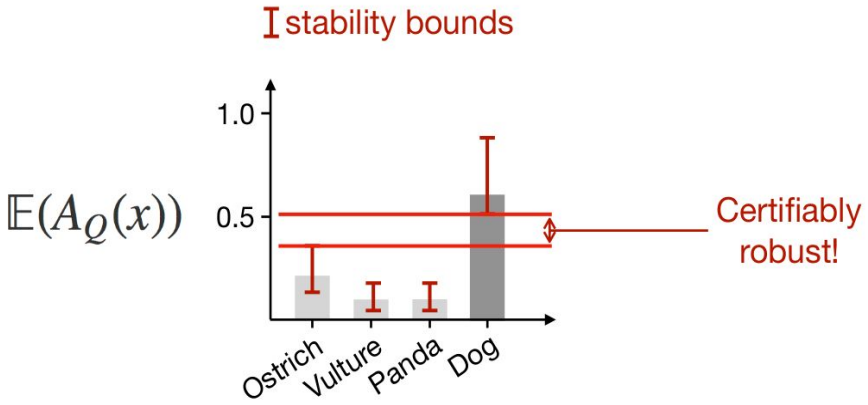
$\Downarrow$

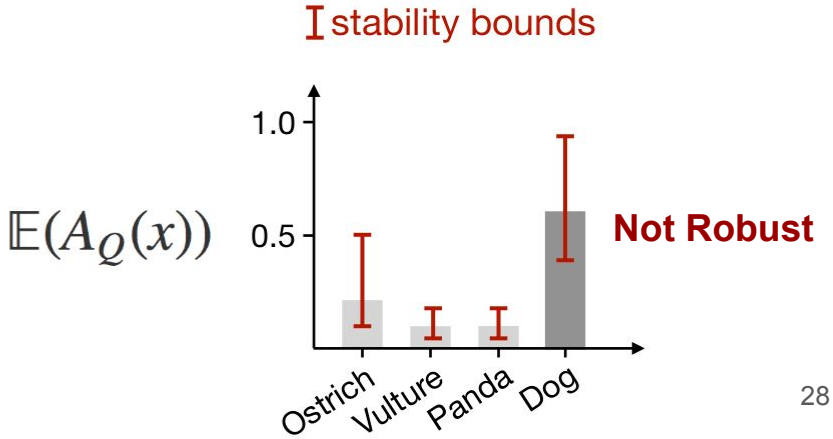
When $A_Q(x) \in [0,1]^K$:

$$\mathbb{E}(A_Q(x)) \leq e^{\epsilon} \mathbb{E}(A_Q(x')) + \delta$$

PixelDP


x
x'
+
input
x
layer1
layer2
layer3
softmax
$A_Q(x)$
0.1
0.1
0.6
0.2
noise layer


stability bounds
$\mathbb{E}(A_Q(x))$
1.0
0.5
Ostrich
Vulture
Panda
Dog
Certifiably robust!

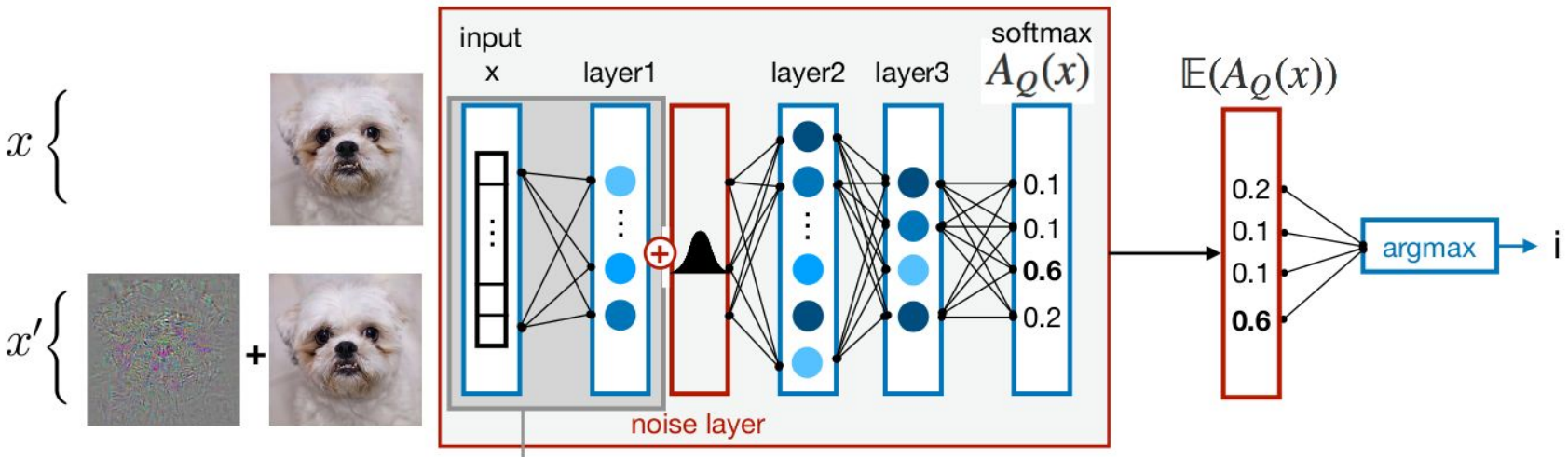

stability bounds
$\mathbb{E}(A_Q(x))$
1.0
0.5
Ostrich
Vulture
Panda
Dog
Not Robust

Challenges

Building a PixelDP DNN in practice:

- How much noise should we add in the noise layer?
- How can we support different attack norms (L0, L1, Linf)?
- How can we estimate the expected scores?

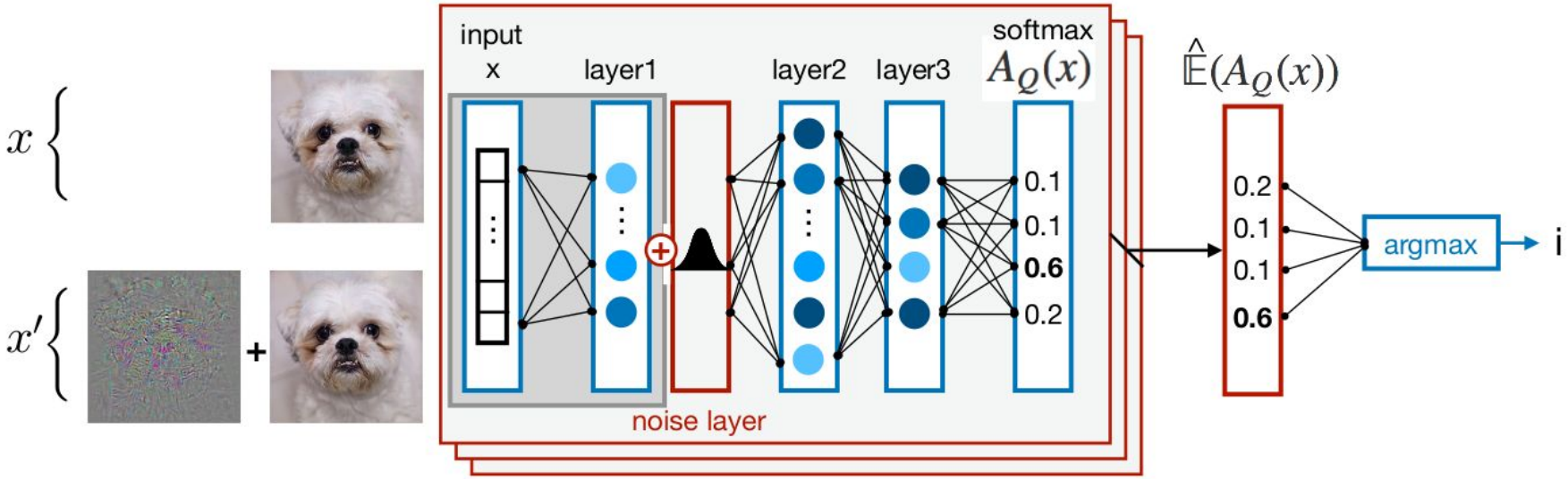
DP noise (Gaussian mechanism)



Noise standard deviation: $$\sigma = \sqrt{2\ln(\frac{1.25}{\delta})}\frac{\Delta_{p,2}L}{\epsilon}$$

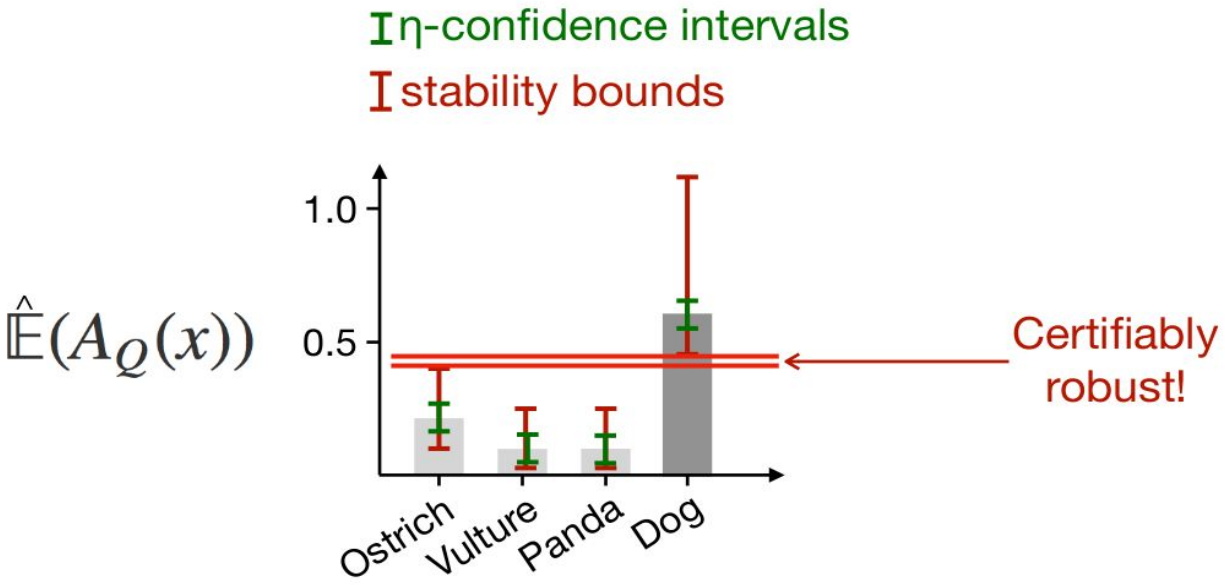
Sensitivity of pre-noise layer: $$\Delta_{p,2} = \max_{x,x':x\neq x'} \frac{\|g(x) - g(x')\|_2}{\|x - x'\|_p}$$

Estimating expected scores



We cannot compute $\hat{\mathbb{E}}(A_Q(x))$ exactly, so we approximate it using Monte Carlo and standard confidence intervals (e.g., Hoeffding's inequality).

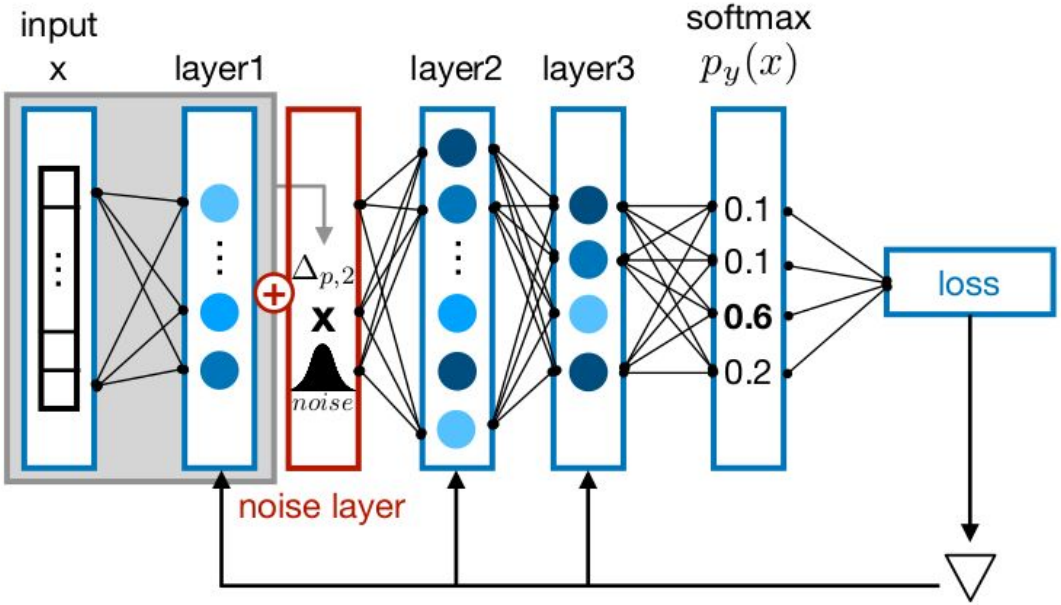
Estimating expected scores



We cannot compute $\hat{\mathbb{E}}(A_Q(x))$ exactly, so we approximate it using Monte Carlo and standard confidence intervals (e.g., Hoeffding's inequality).

Training with noise

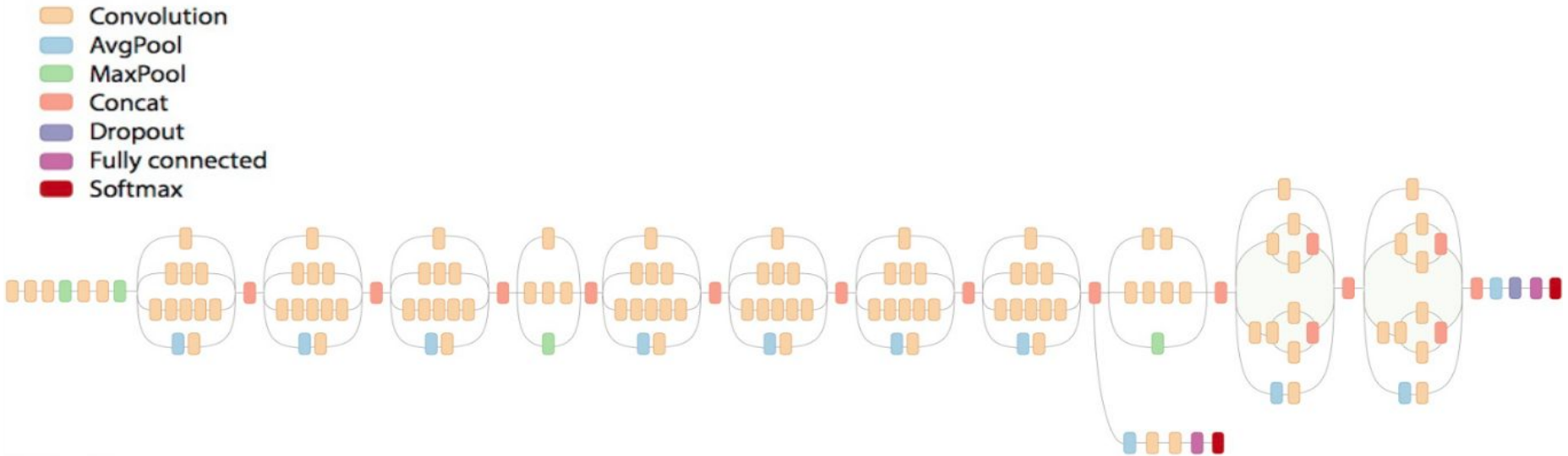
Adding noise at training time:



What we did so far?

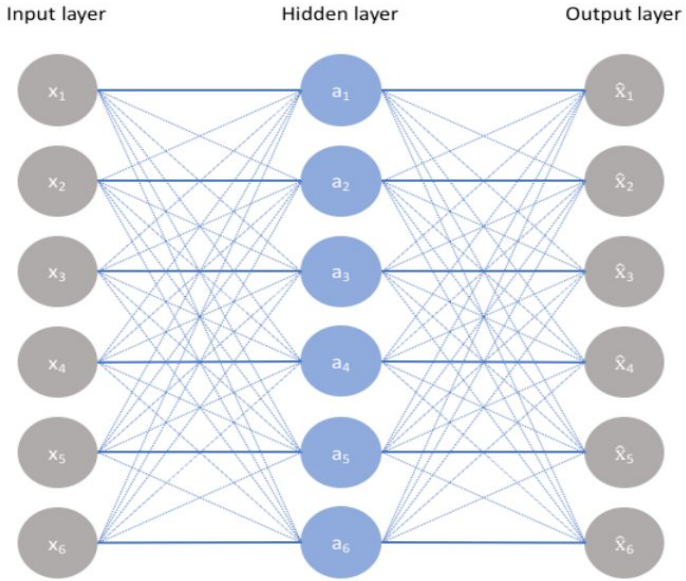
- We use DP to provide provable robustness guarantees.
- Each prediction come with a robustness size: no attack smaller than this size can change the prediction.
- Using a testing set, we can lower-bound the accuracy under attack.

Scaling to ImageNet



- 48 layers deep
- 23 millions parameters

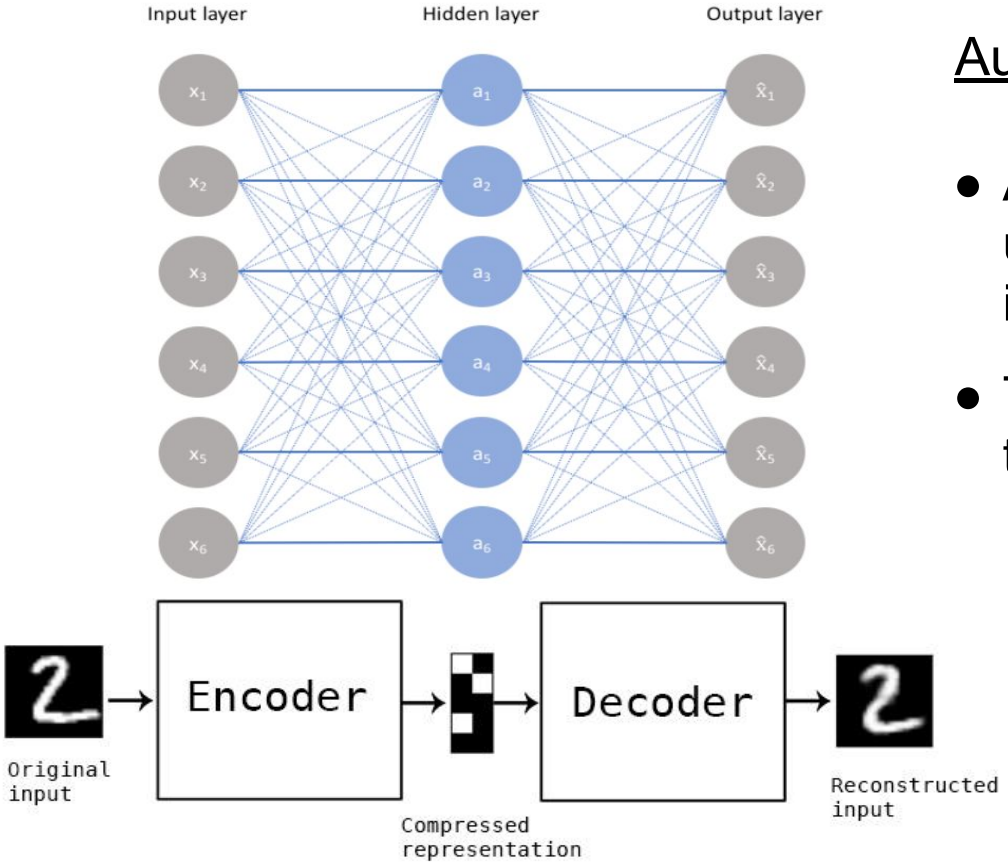
Scaling to ImageNet



Autoencoder (brief background)

- An autoencoder is a neural network used to learn an approximation of the identity function over a dataset.

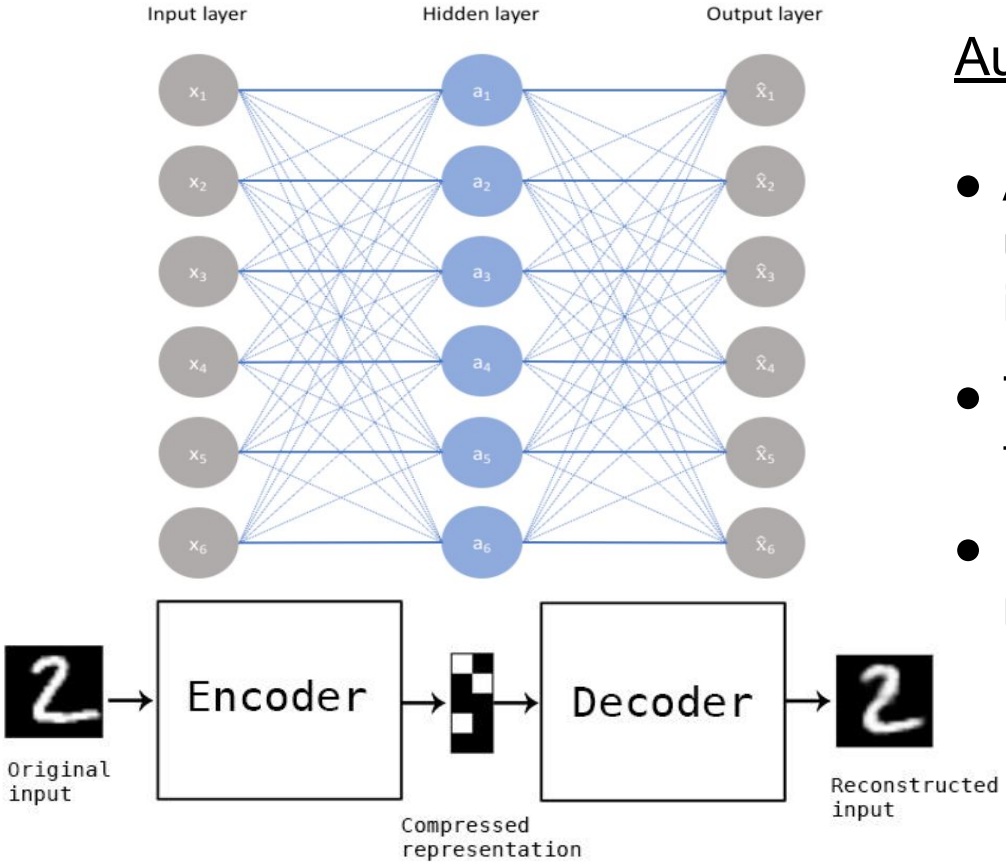
Scaling to ImageNet



Autoencoder (brief background)

- An autoencoder is a neural network used to learn an approximation of the identity function over a dataset.
- The output x' is a reconstruction of the input x.

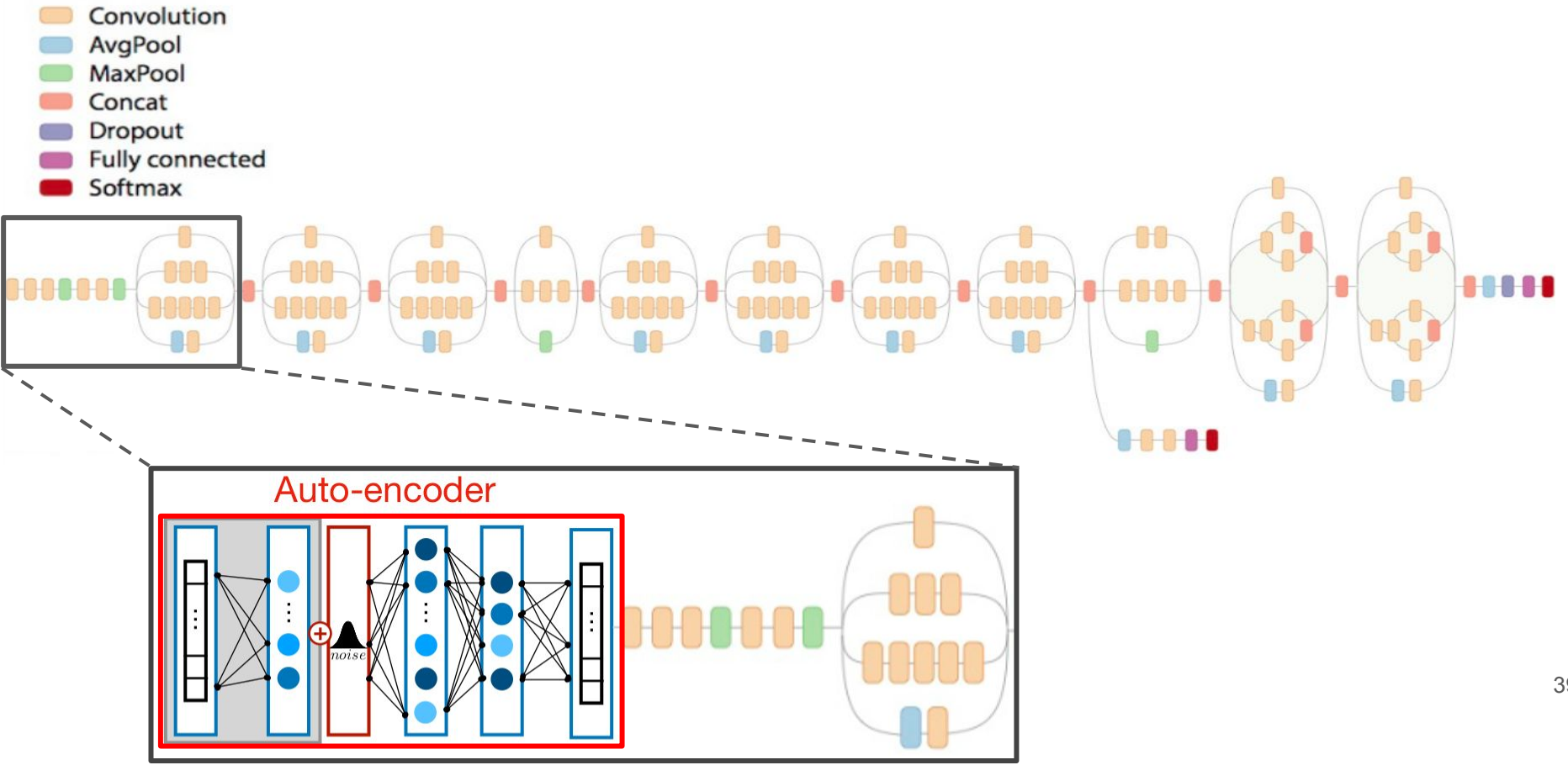
Scaling to ImageNet



Autoencoder (brief background)

- An autoencoder is a neural network used to learn an approximation of the identity function over a dataset.
- The output x' is a reconstruction of the input x.
- Unsupervised learning (no labels needed, just raw data).

Scaling to ImageNet



Outline

- Motivation
- Background
- PixelDP design
- Evaluation

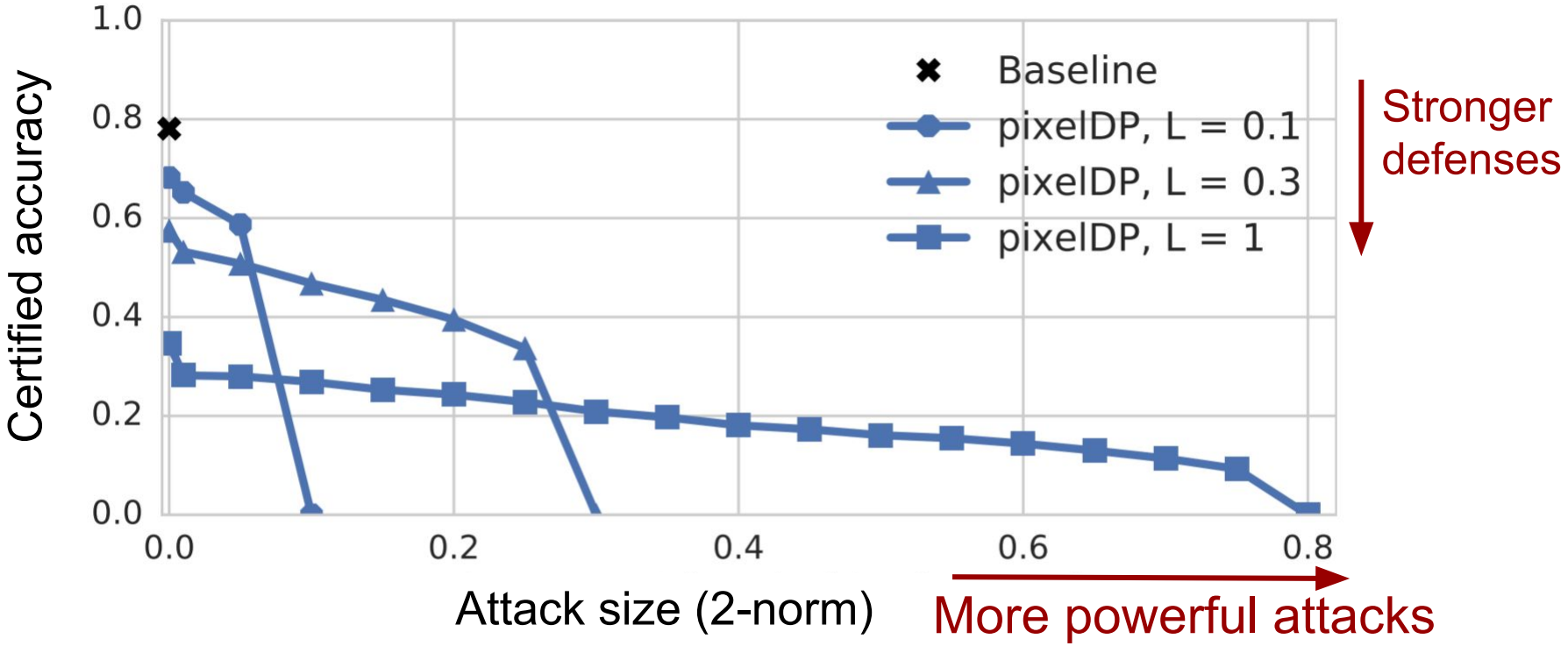
Evaluation Questions

Q1: Can we give certified accuracy on large DNNs/datasets?

Q2: What is PixelDP's robustness against current state-of-the-art attacks?

Q3: How does PixelDP compare with other defenses?

Certified accuracy ImageNet/Inception-v3



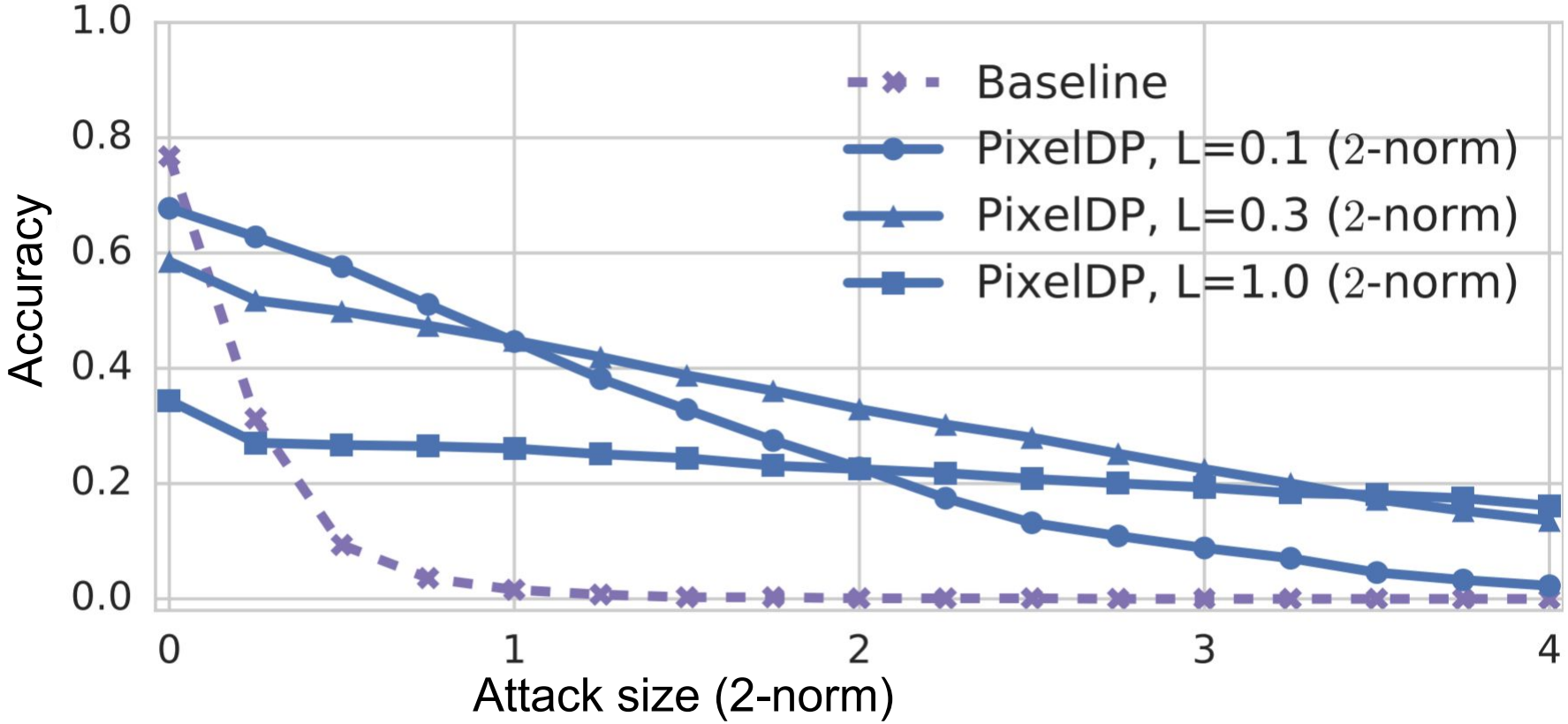
Evaluation Questions

Q1: Can we give certified accuracy on large DNNs/datasets?

Q2: What is PixelDP's accuracy against current state-of-the-art attacks?

Q3: How does PixelDP compare with other defenses?

Attack on ImageNet/Inception-v3



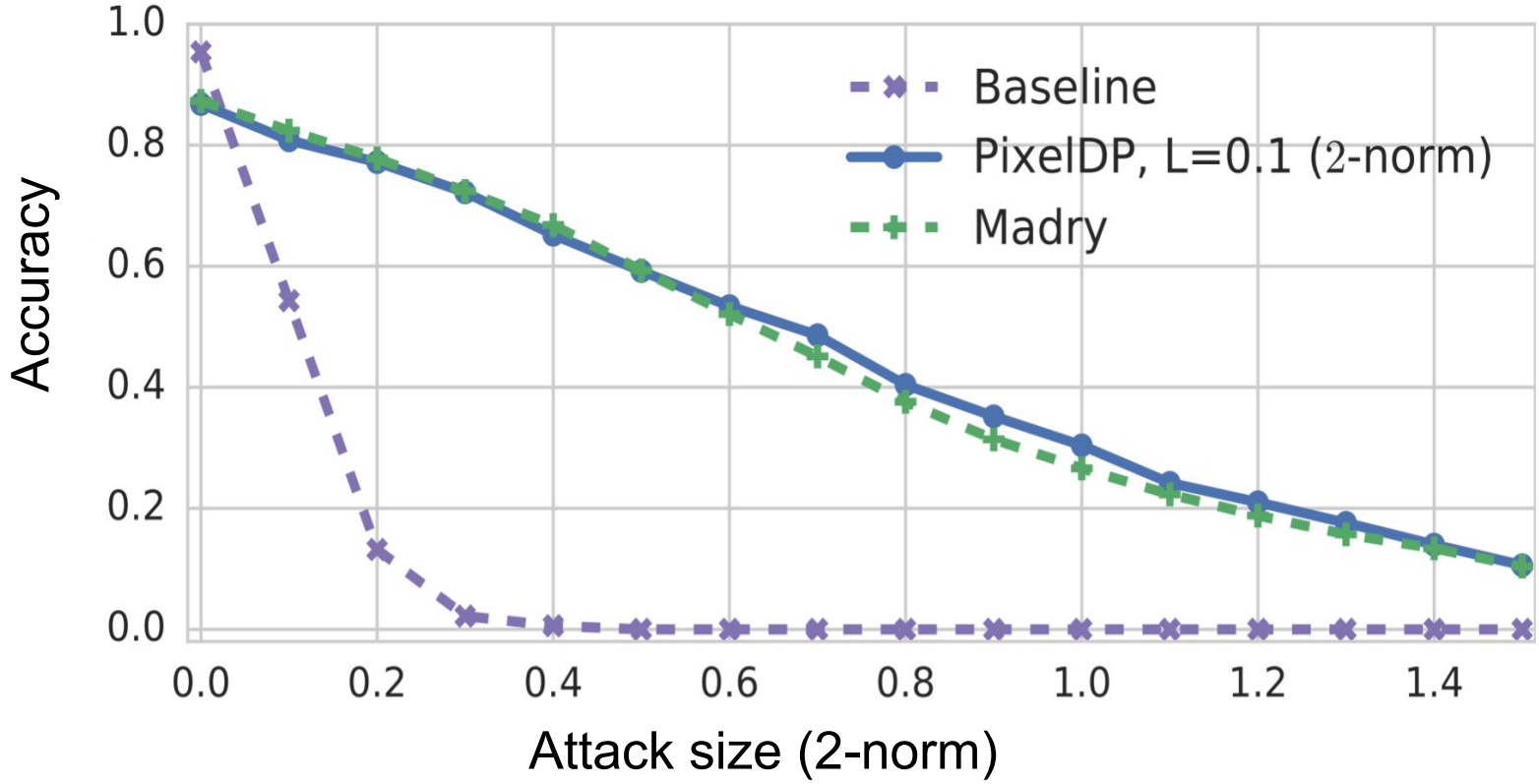
Evaluation Questions

Q1: Can we give certified accuracy on large DNNs/datasets?

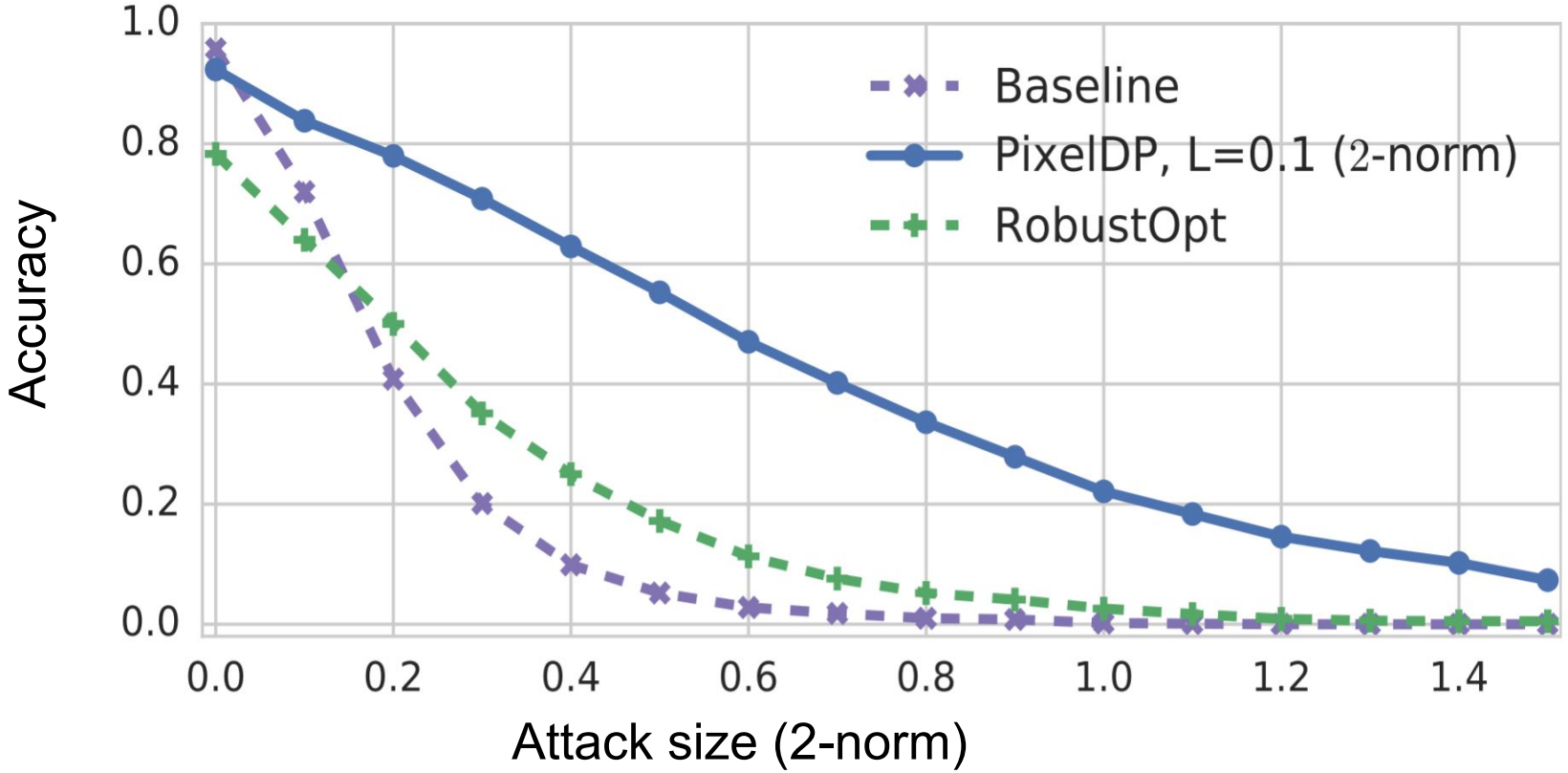
Q2: What is PixelDP's accuracy against current state-of-the-art attacks?

Q3: How does PixelDP compare with other defenses?

PixelDP vs Marry et al. (on CIFAR-10)



PixelDP vs RobustOpt (on SVHN)



Related Work and Reflection

Best effort

✔️ **Scale**
- Run a best effort attack per gradient step [Goodfellow‘15, Madry‘17].
- Preprocess inputs [Buckman’18, Guo’18].
- Train a second model based on the first one [Papernot ‘16].

✔️ **Flexibity**
- Support most architectures.

➖ **No robustness guarantees**
- Often broken after release [Athalye ’18].

Certified

✔️ **Robustness guarantees**
- Various techniques [Kolter ’17, Raghunathan ‘18, Sinha ’17].

➖ **Hard to scale**
- Requires orders of magnitude more computation [Kolter ‘17].
- Support only 1 hidden layer [Raghunathan ‘18].

➖ **Often not flexible**
- No ReLU activations, no MaxPool [Sinha ‘17].
- Only ReLU activations, no batch normalization [Kolter ‘17].

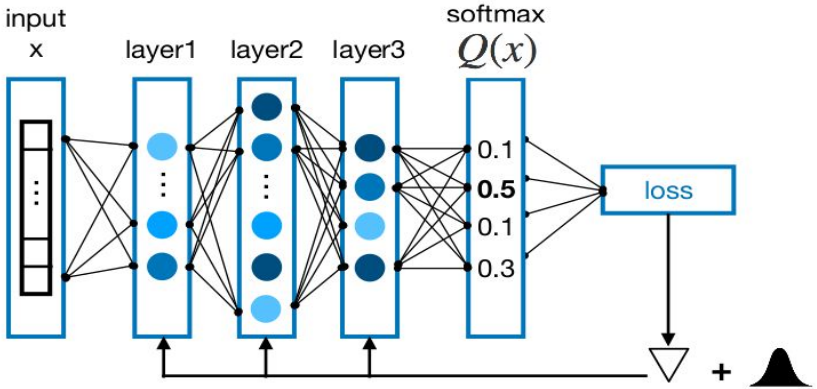
Conclusions

PixelDP is the first certified defense that both achieves provable guarantees of robustness at **scale** and is **broadly applicable** to arbitrary networks.

Code: https://github.com/columbia/pixeldp

Differential privacy

The learning procedure is DP w.r.t. the training set.



Training computes weights $w(d)$.

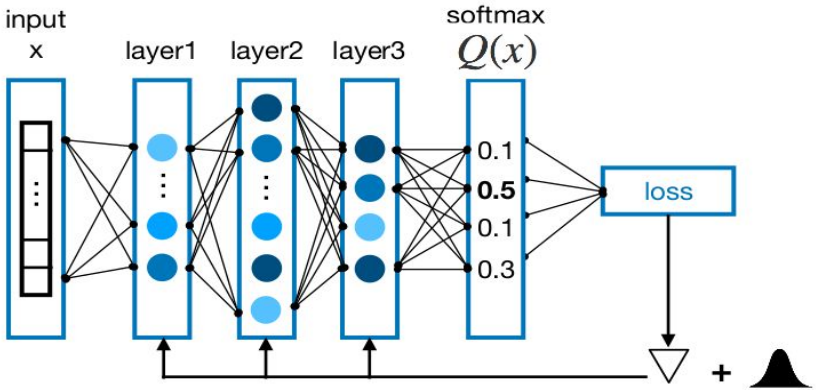
$$P(w(d) \in S) \leq e^{\epsilon} P(w(d') \in S) + \delta$$

PixelDP

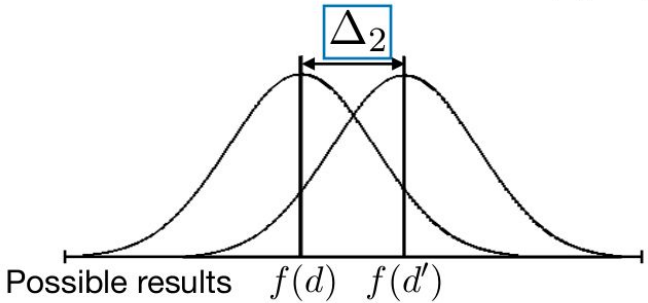
- Make the prediction function DP w.r.t. the input features.
- **Intuition**: Protect the privacy of the pixels when we release the prediction.

Differential privacy

The learning procedure is DP w.r.t. the training set.



Gaussian mechanism: $A(d) = f(d) + \mathcal{N}(0, \sigma^2)$



Is (ϵ, δ)-DP for $\sigma = \sqrt{2\ln(\frac{1.25}{\delta})}\frac{\Delta_2}{\epsilon}$

Bounding sensitivity

Bounding sensitivity during training:

- Noise in the input layer: nothing to do
- Noise deeper in the network: enforce sensitivity = 1

One linear layer W: $\Delta_{p,2} = \|W\|_{p,2}$

$\Delta_{1,2}$: Normalize columns

$\Delta_{2,2}$: Parseval projection [Cisse'18]

$\Delta_{\infty,2} \leq \sqrt{n}\Delta_{2,2}$