Lecture 18: Adversarial examples

CS 182/282A (“Deep Learning”)

2022/04/04

Today's lecture

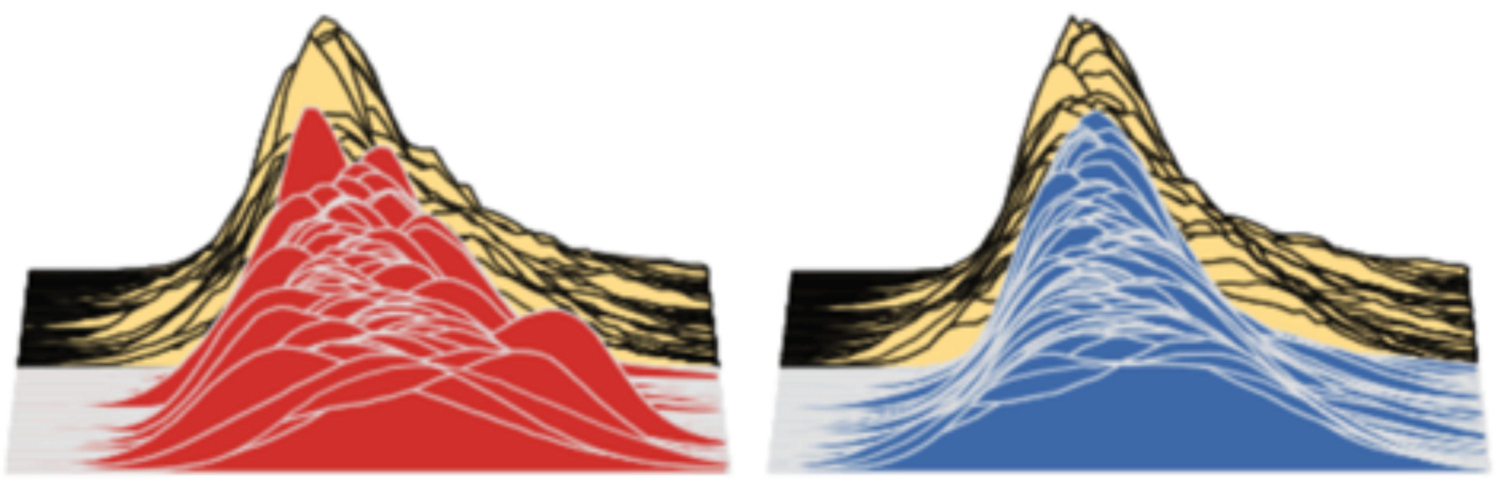
- Today, we wrap up our discussion on robustness and distribution shift
- We will start by going over **test time adaptation**, which asks the question: can the model change at test time after seeing the test data to handle the shift?
- Then, we will switch gears and talk about **adversarial robustness**
- This differs from what we have covered previously because the distribution shift is no longer a natural consequence of the real world being complicated
  - Instead, we now have an **adversary** that is purposely trying to manipulate the data to harm our model, and we will see that this is a rather challenging problem

Test time adaptation

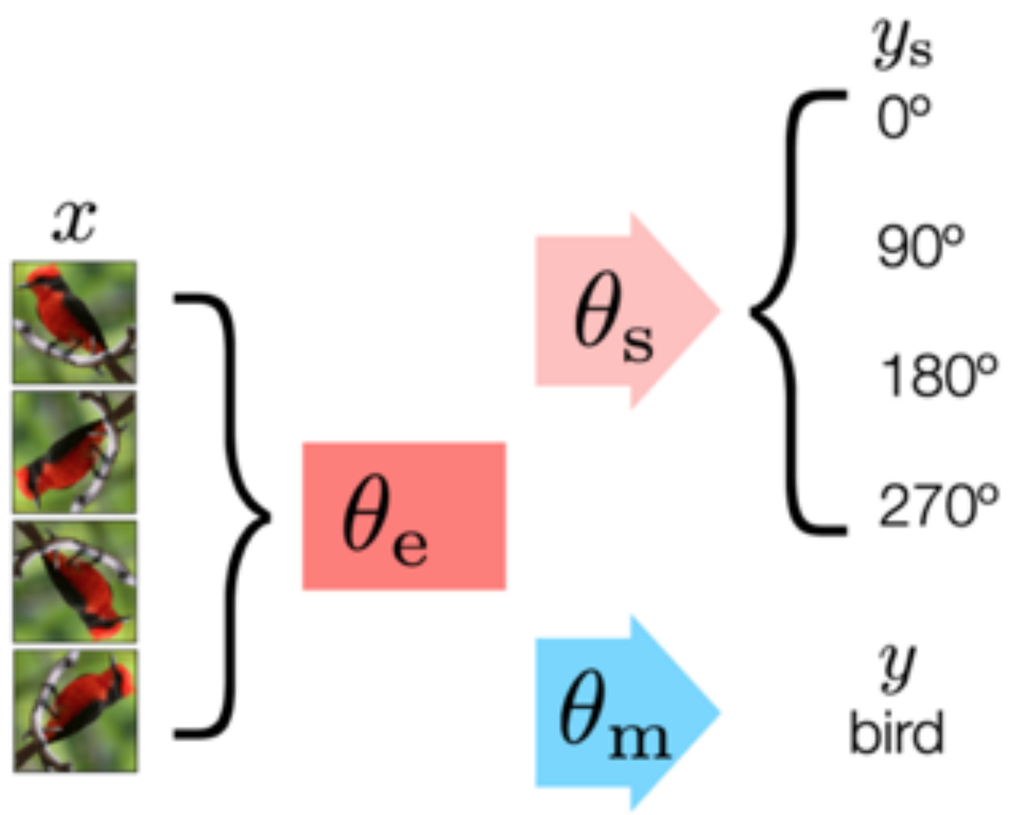
- An alternative, and potentially complementary, approach to handling shift is to **adapt** the model at test time, using the available information
- In other words, assume that we have access to and can change the model's parameters, or we have other means of augmenting the model's predictions
- Many test time adaptation approaches assume that multiple test points are available, from which we may be able to estimate statistics of the underlying test distribution
- E.g., when there is **label shift** (only $p(y)$ changes), a principled approach is to adapt the classifier's threshold for predicting various classes (Lipton et al, ICML 2018)

Methods for test time adaptation

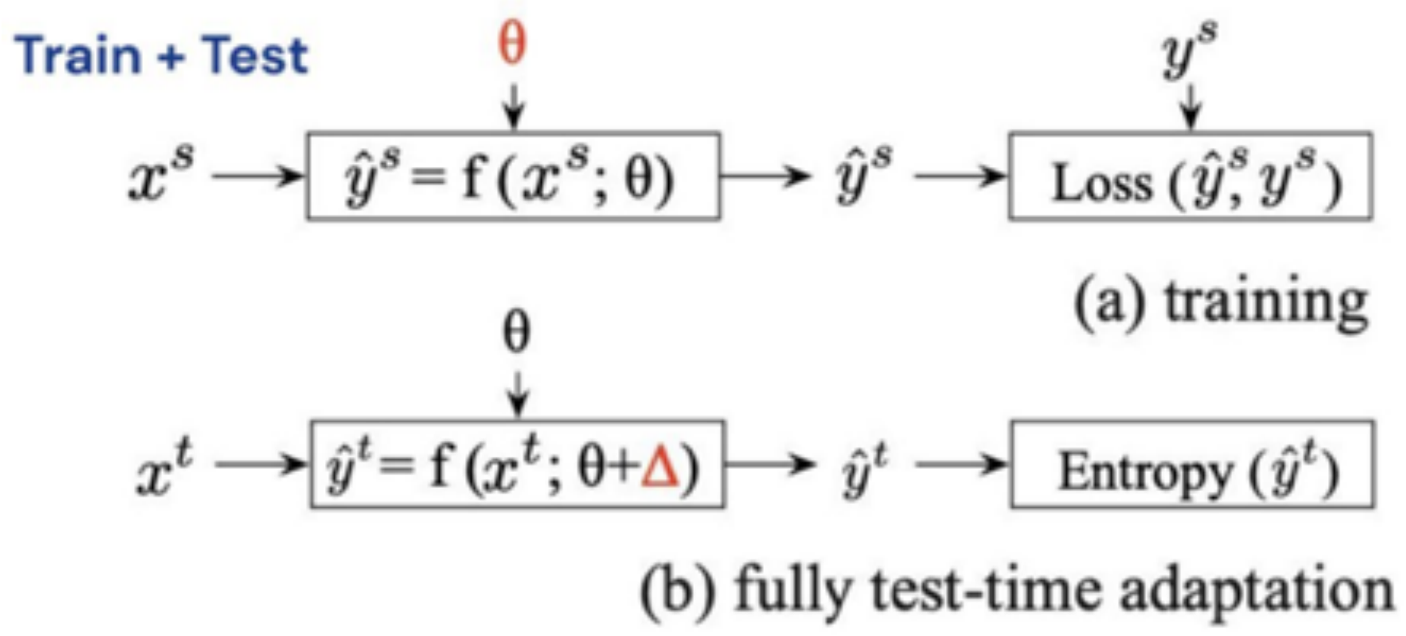
Self-supervised learning via:



BN adaptation (image from Nado et al, ’20)



rotation prediction (Sun et al, ICML ’20)

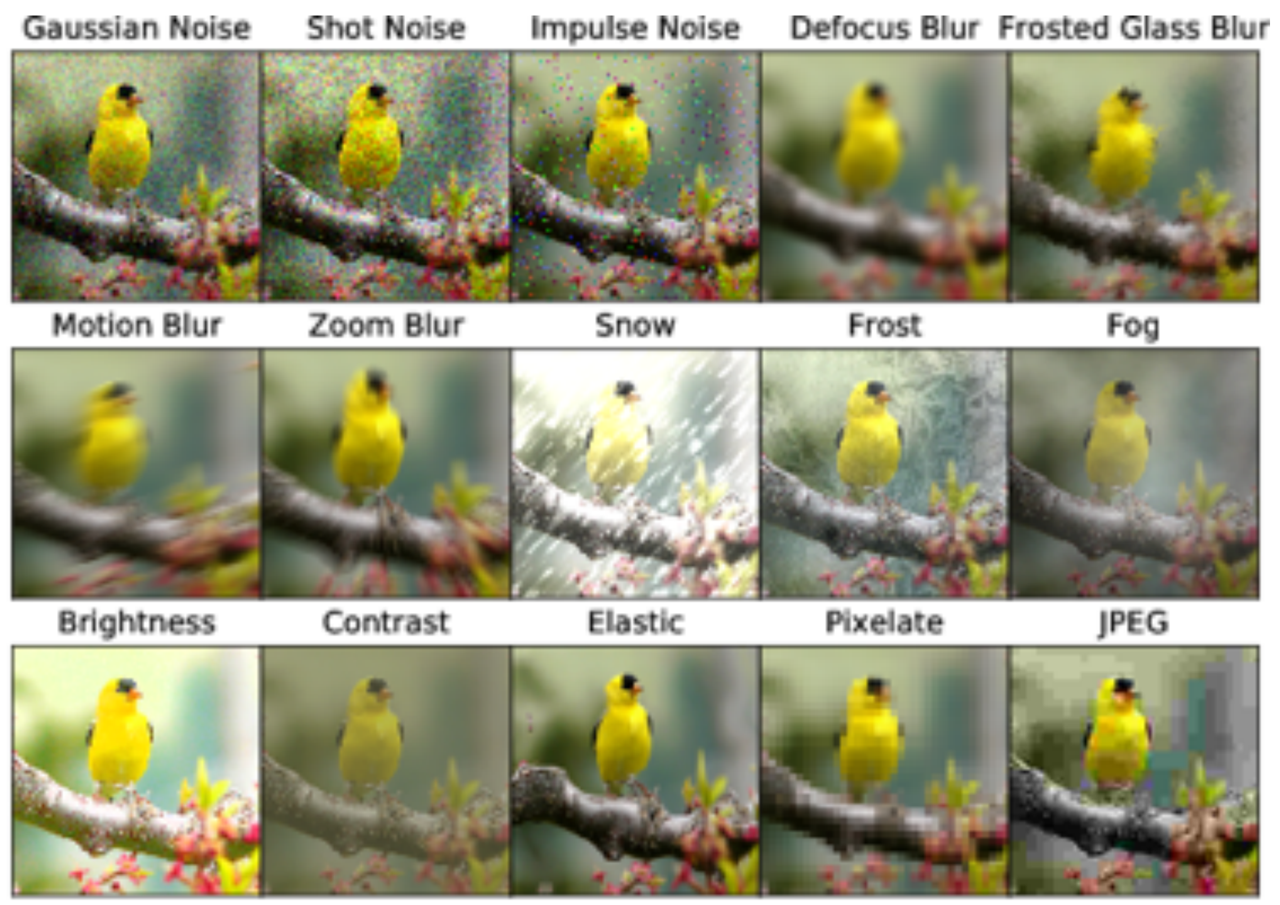


entropy minimization (Wang et al, ICLR ’21)

“standard” model: $g : \mathscr{X} \to \mathscr{Y}$

adaptive model: $f : \mathscr{X} \times \mathscr{P}_{\mathbf{x}} \to \mathscr{Y}$

in practice, approximate $\mathscr{P}_{\mathbf{x}}$ with $(\mathbf{x}_1, \ldots, \mathbf{x}_K)$



Adversarial robustness

Imperceptible adversarial distortions

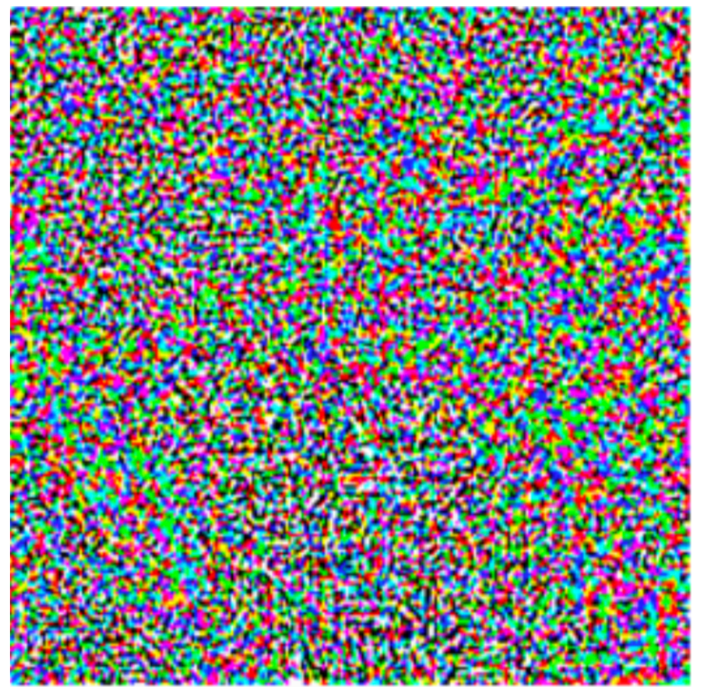
An older example

- The adversarial distortion is optimized to cause the (undefended, off-the-shelf) neural network to make a mistake

- Now, models can be trained (defended) against such imperceptible distortions

“cat”

$+\epsilon$

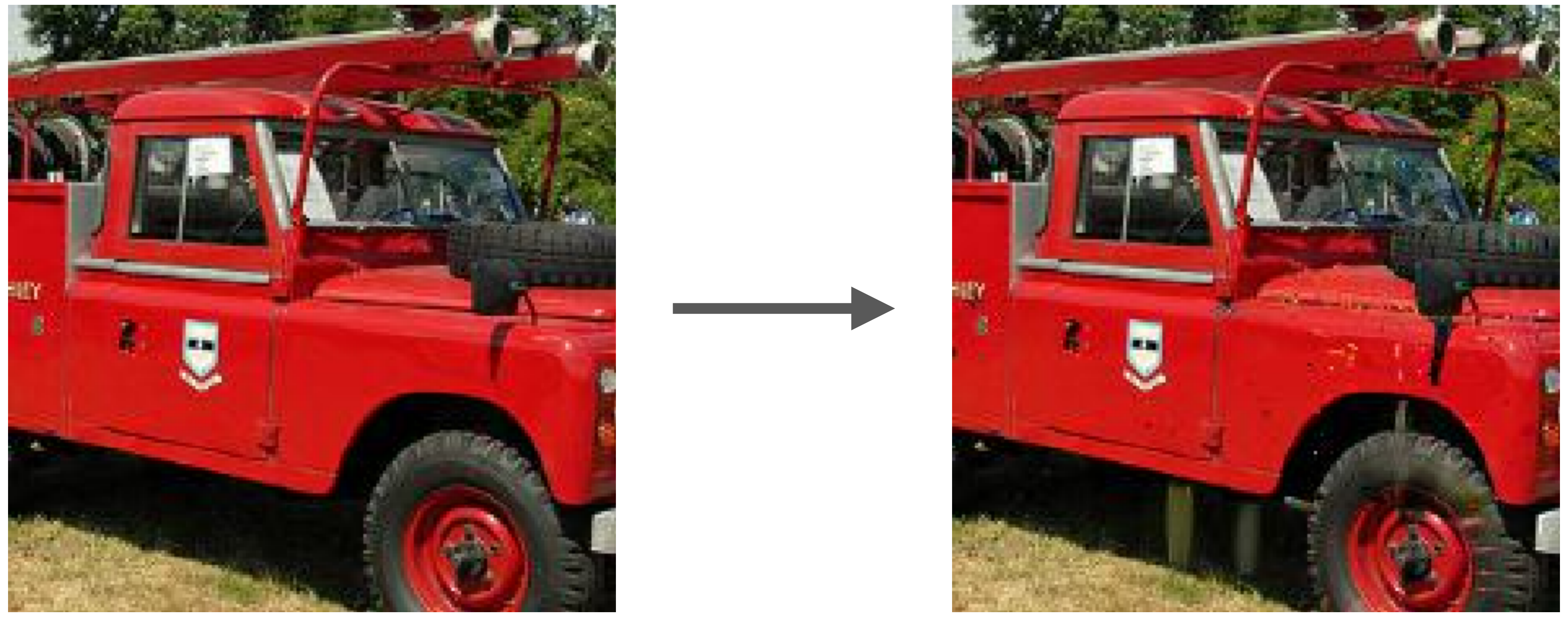


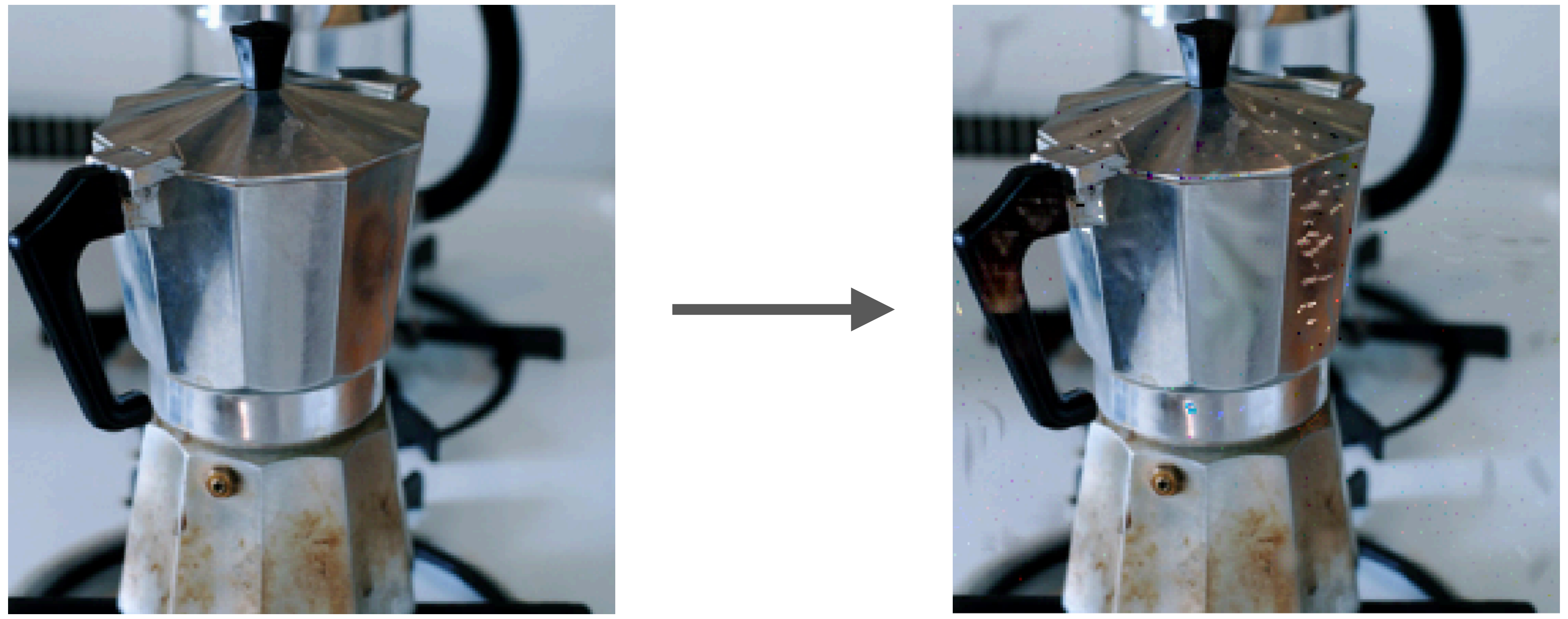
$=$

“guacamole”

Modern adversarial distortions

- Here, the adversary makes changes to the image that are *perceptible* to the human eye, yet the underlying class is unchanged
- Modern neural network models can be made robust to imperceptible distortions, but they are still not robust to perceptible distortions

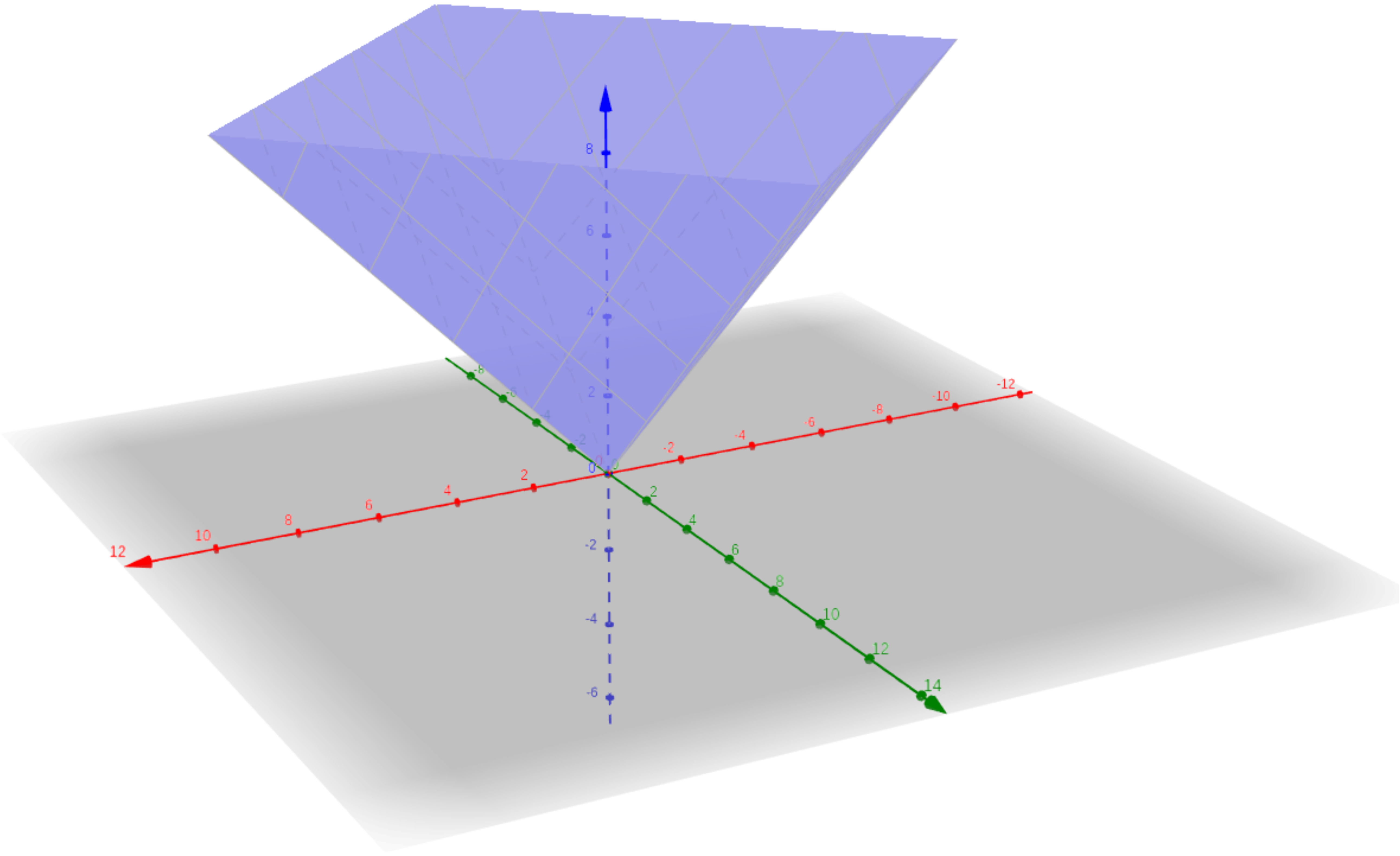




Review: ℓ_1-norm

$$\|\mathbf{v}\|_1 = |v_1| + |v_2| + \ldots + |v_d|$$

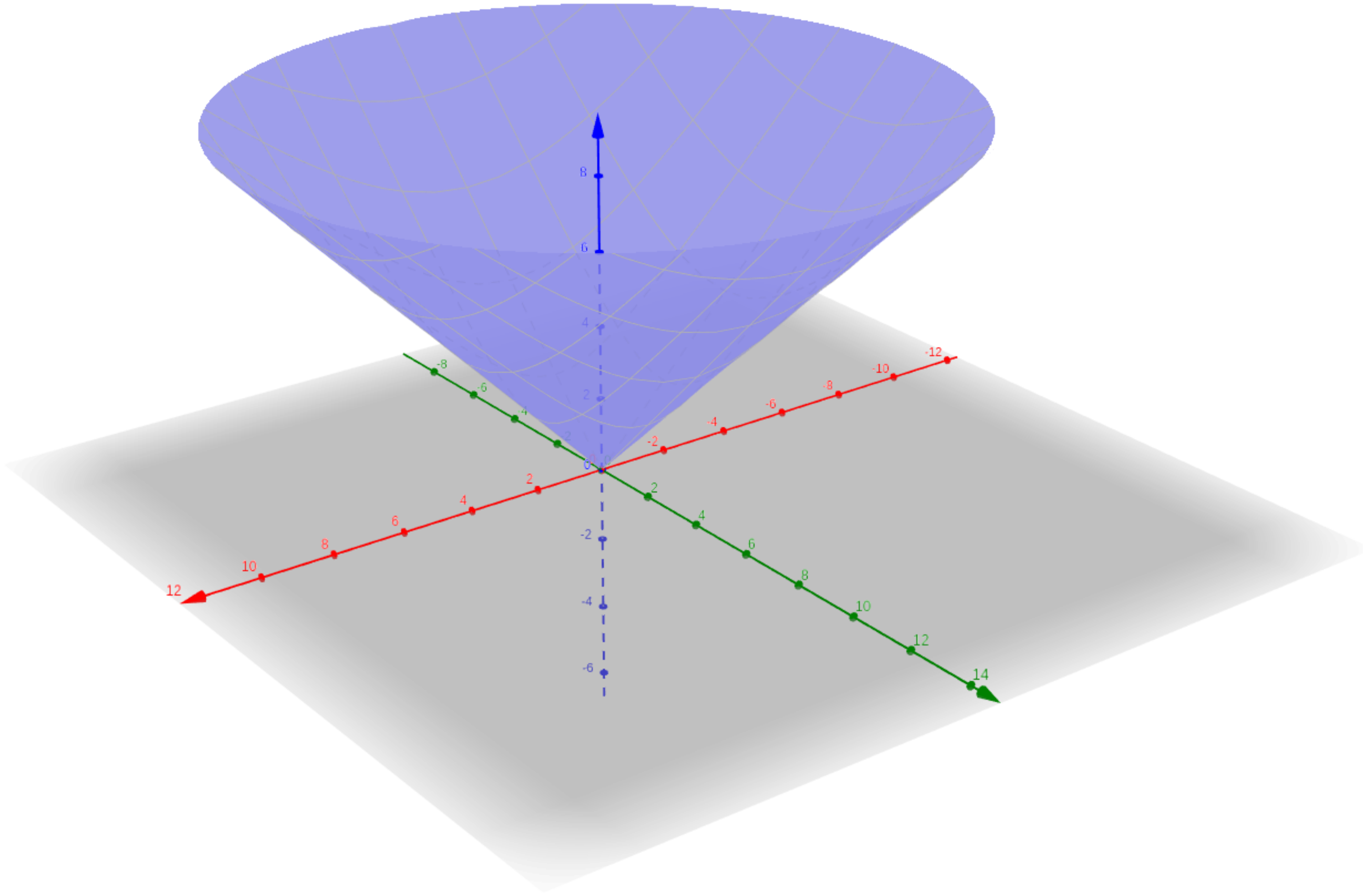
```
l1 = 0
# RGB image is a perturbation p of size 3x224x224
for c in range(3):
    for y in range(224):
        for x in range(224):
            l1 += abs(p[c,y,x])
```



Review: ℓ_2-norm

$$\|\mathbf{v}\|_2 = \sqrt{v_1^2 + v_2^2 + \ldots + v_d^2}$$

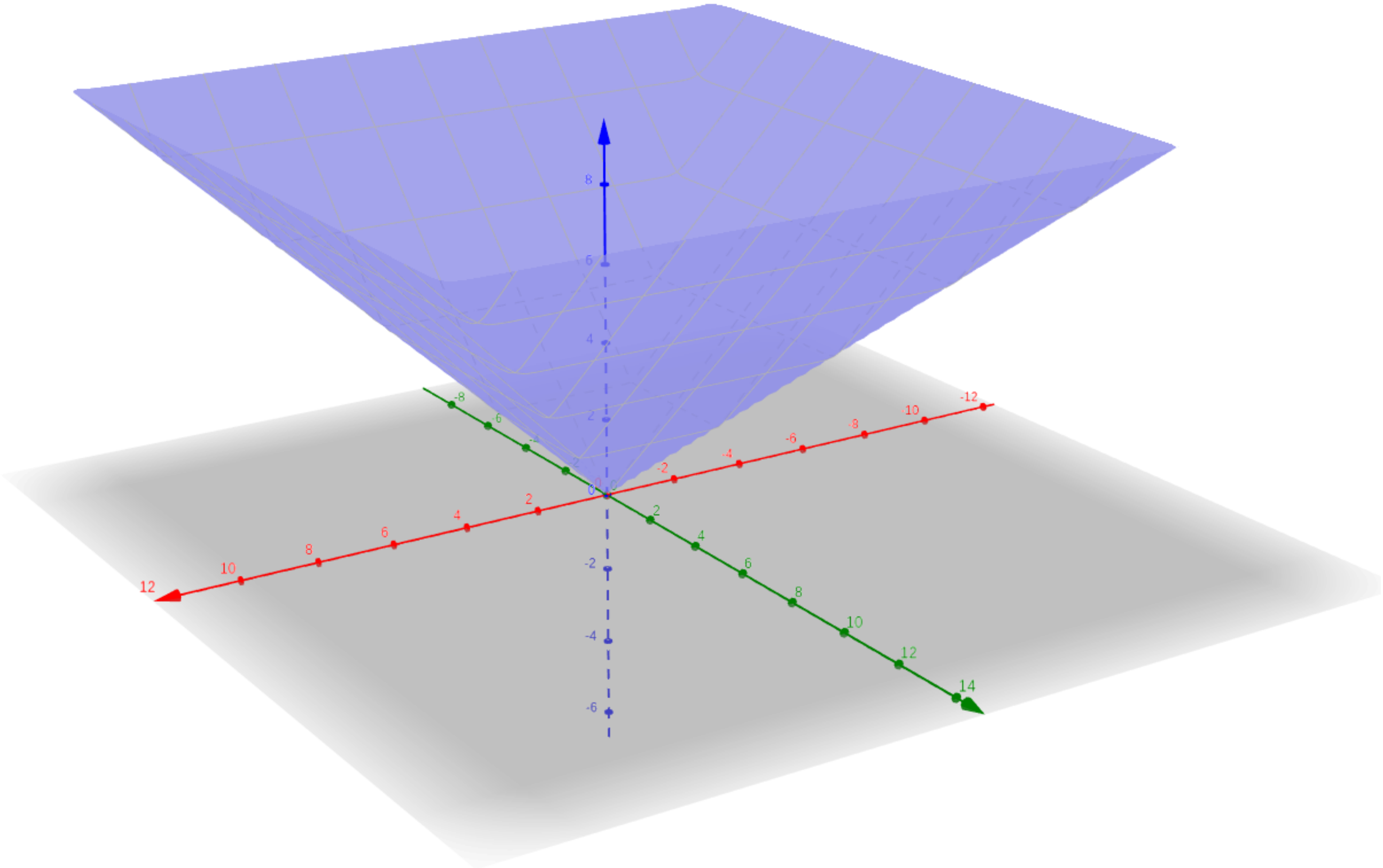
```
l2 = 0
# RGB image is a perturbation p of size 3x224x224
for c in range(3):
    for y in range(224):
        for x in range(224):
            l2 += square(p[c,y,x])

l2 = sqrt(l2)
```



Review: ℓ_∞-norm

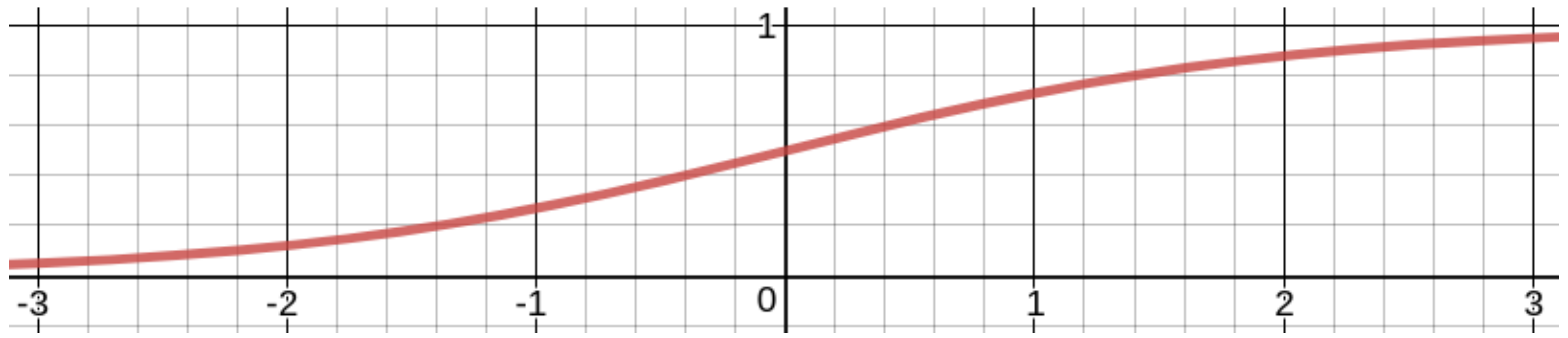
$$\|\mathbf{v}\|_\infty = \max\{\,|v_1|, |v_2|, \ldots, |v_d|\,\}$$

```
linf = 0
# RGB image is a perturbation p of size 3x224x224
for c in range(3):
    for y in range(224):
        for x in range(224):
            linf = max(linf, abs(p[c,y,x]))
```



Fooling a binary logistic regression model

Suppose our model is $f_\theta(\mathbf{x}) = \dfrac{\exp \theta^\top \mathbf{x}}{\exp \theta^\top \mathbf{x} + 1}$



Input	x	2	-1	3	-2	2	2	1	-4	5	1
Weight	θ	-1	-1	1	-1	1	-1	1	1	-1	1

$$\theta^\top \mathbf{x} = -3 \quad f_\theta(\mathbf{x}) \approx 0.05$$

Input	x	2	-1	3	-2	2	2	1	-4	5	1
Adv Input	x+ε	1.5	-1.5	3.5	-2.5	1.5	1.5	1.5	-3.5	4.5	1.5
Weight	θ	-1	-1	1	-1	1	-1	1	1	-1	1

$$\theta^\top(\mathbf{x} + \epsilon) = 2 \quad \|\epsilon\|_\infty = 0.5 \quad f_\theta(\mathbf{x} + \epsilon) \approx 0.88$$

Logistic regression takeaways

Input	x	2	-1	3	-2	2	2	1	-4	5	1
Adv Input	x+ε	1.5	-1.5	3.5	-2.5	1.5	1.5	1.5	-3.5	4.5	1.5
Weight	θ	-1	-1	1	-1	1	-1	1	1	-1	1

$$\theta^{\top}(\mathbf{x}+\epsilon) = 2 \quad \|\epsilon\|_{\infty} = 0.5 \quad f_{\theta}(\mathbf{x}+\epsilon) \approx 0.88$$

- The cumulative effect of many small changes made the adversary powerful enough to change the classification decision
- Adversarial examples exist for non deep learning (even linear) models

An adversary threat model

- A simple threat model is to assume the adversary has an ℓ_p attack *distortion budget* ϵ, i.e., for some assumed p and ϵ, $\|\mathbf{x}_{\text{adv}} - \mathbf{x}\|_p \leq \epsilon$
- Not all distortions have a small ℓ_p norm, e.g., rotations — this simplistic threat model is common because it is a more tractable subproblem
- The adversary's goal is usually to find a distortion δ that maximizes the loss subject to its budget: $\mathbf{x}_{\text{adv}} = \mathbf{x} + \arg\max_{\delta : \|\delta\|_p \leq \epsilon} \ell(\theta; \mathbf{x} + \delta, y)$

Fast gradient sign method (FGSM)

- How do we generate adversarial examples algorithmically?
- A simple attack is the **FGSM** attack: $\mathbf{x}_{\text{FGSM}} = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} \ell(\theta; \mathbf{x}, y))$
- This attack performs a single step of gradient ascent on the input to increase the model's loss, obeying an ℓ_∞ attack budget $||\mathbf{x}_{\text{FGSM}} - \mathbf{x}||_\infty = \epsilon$
- The attack is called "fast" because it only uses a single gradient ascent step
- This attack is easy to defend against nowadays — more on that in a bit

Projected gradient descent (PGD)

- The **PGD** attack uses multiple gradient ascent steps and thus is far more powerful than the FGSM attack

- Pseudocode for a PGD attack with T steps and an ℓ_∞ attack budget ϵ:

  Randomly initialize a perturbed image for more diverse attacks:
  $\tilde{\mathbf{x}} = \mathbf{x} + n,$ where $n_i \sim \mathcal{U}[-\epsilon, \epsilon]$, and initialize $\delta = 0$

  For $t = 1, \ldots, T$: $\delta \leftarrow \text{clip}(\delta + \alpha \, \text{sign}(\nabla_\delta \ell(\theta; \tilde{\mathbf{x}} + \delta, y)), -\epsilon, \epsilon)$

  Finally: $\mathbf{x}_{\text{PGD}} = \tilde{\mathbf{x}} + \delta$

Adversarial training (AT)

- The best way (we know of) to robustify models to ℓ_p attacks is **adversarial training (AT)**
- A common AT procedure is as follows:

  Sample minibatch $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(B)}, y^{(B)})$ from the training set

  Create $\mathbf{x}^{(i)}_{\text{adv}}$ (e.g., $\mathbf{x}^{(i)}_{\text{PGD}}$) from $\mathbf{x}^{(i)}$ for all i

  Optimize the average training loss on these adversarial training examples
- This does come with some downsides: currently, AT can reduce accuracy on non adversarial (“clean”) examples by 10%+

Untargeted vs. targeted attacks

- So far we have assumed **untargeted** attacks which just try to maximize the loss
- By contrast, a **targeted** attack optimizes examples to be misclassified as a predetermined target $\tilde{y}$
- Targeted attack evaluation is standard for ImageNet because there are many similar classes

"labrador retriever"

untargeted

$$\arg \max_{\delta: \|\delta\|_p \leq \epsilon} \ell(\theta; \mathbf{x} + \delta, y)$$

"golden retriever"

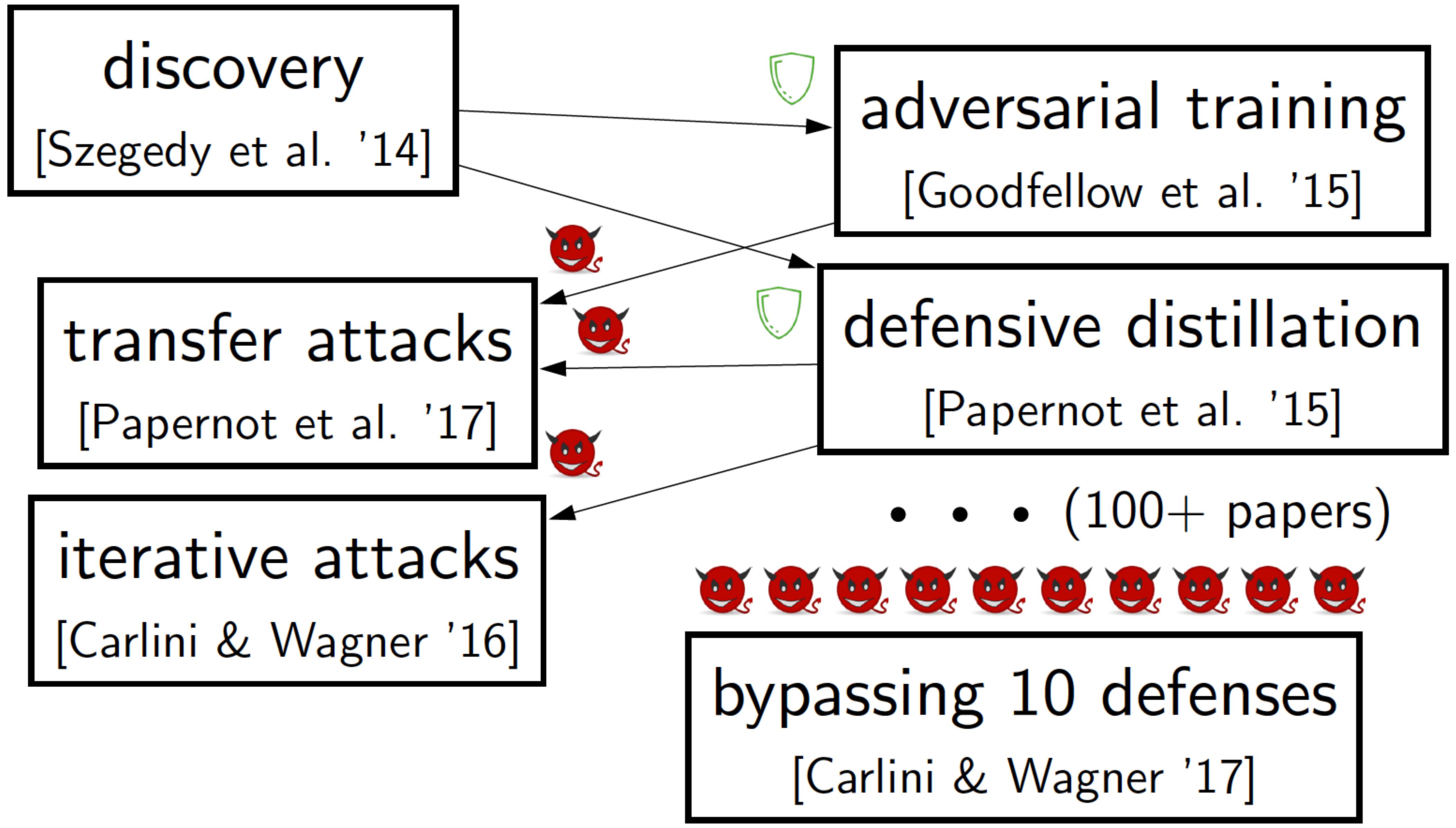
targeted

$$\arg \min_{\delta: \|\delta\|_p \leq \epsilon} \ell(\theta; \mathbf{x} + \delta, \tilde{y})$$

"great white shark"

The adversarial “arms race”

- Typically, newly proposed defenses are evaluated narrowly (non comprehensively)
- This leads to an “arms race” that defenders lose
- Proper and thorough evaluation of defenses is very difficult (look up “On Evaluating Adversarial Robustness”)
- Most proposed defenses are broken *within weeks* of being proposed

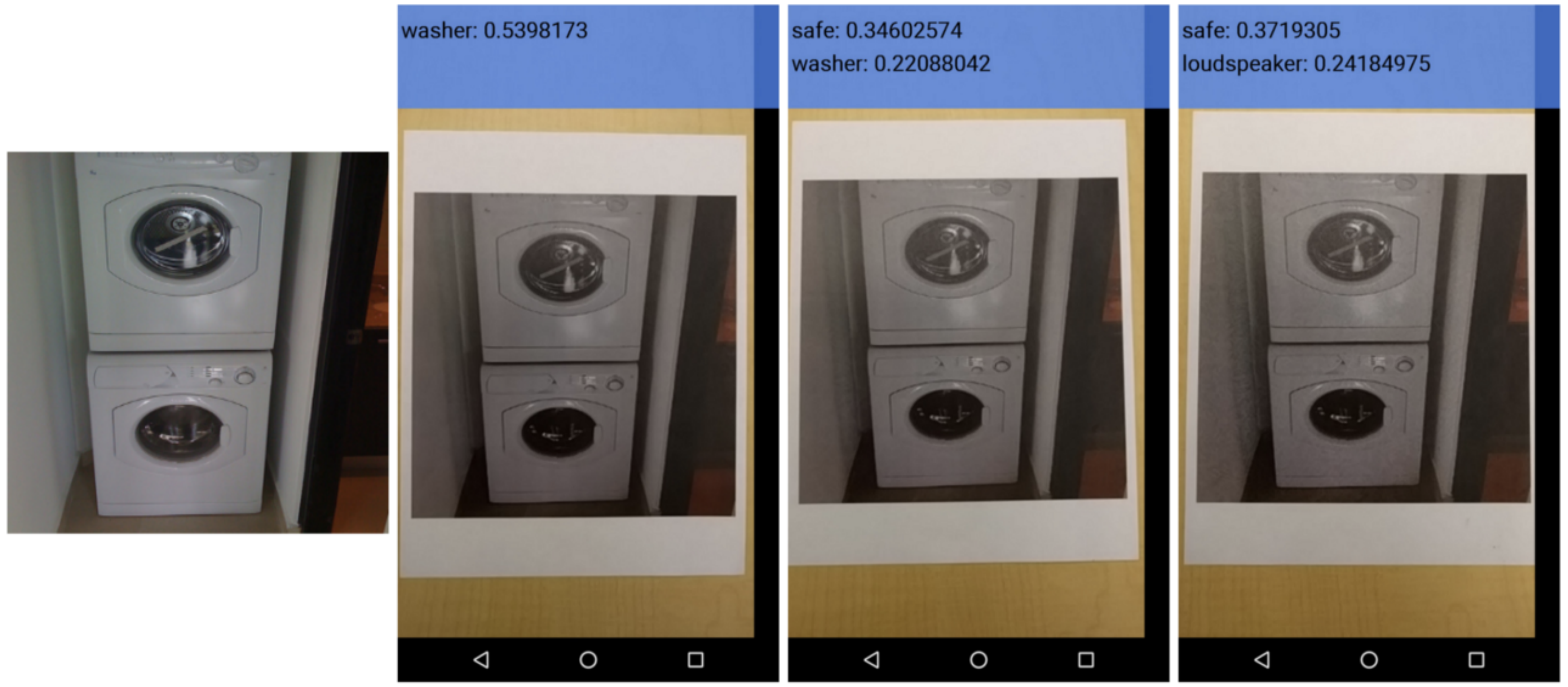


Transferability of attacks

- An adversarial example crafted for one model can potentially be used to attack many different models
- Given neural network models M_1 and M_2, $\mathbf{x}_{\text{adv}}$ designed for M_1 sometimes also results in a high loss for $M_2(\mathbf{x}_{\text{adv}})$, even if M_2 is a different architecture
- Transfer rates can vary greatly, but even moderate amounts of transferability demonstrate that adversarial failure modes are somewhat shared across models
- Consequently, an attacker does not always need access to a model’s parameters or architectural information in order to try and attack it

Transferability to the real world

- Adversarial examples can sometimes even withstand real-world instantiation noise (e.g., printer imperfections) and sensor noise (e.g., from cameras)
- E.g., for a model that has not undergone adversarial training, testing susceptibility to an adversarial example that is printed and photographed:



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Using larger and more diverse data

… again

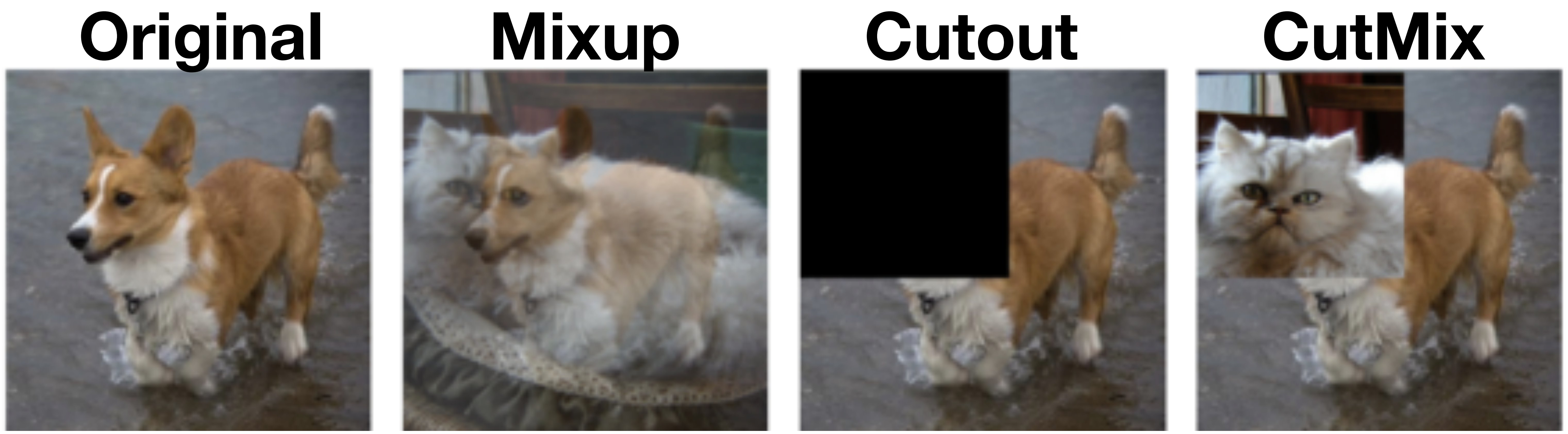
- Adversarial robustness scales slowly (similar to clean accuracy) with dataset size
- Adversarial pretraining on a larger training set has been shown to help
- E.g., to increase CIFAR-100 adversarial robustness, one can first adversarially pretrain on ImageNet and obtain some robustness benefits

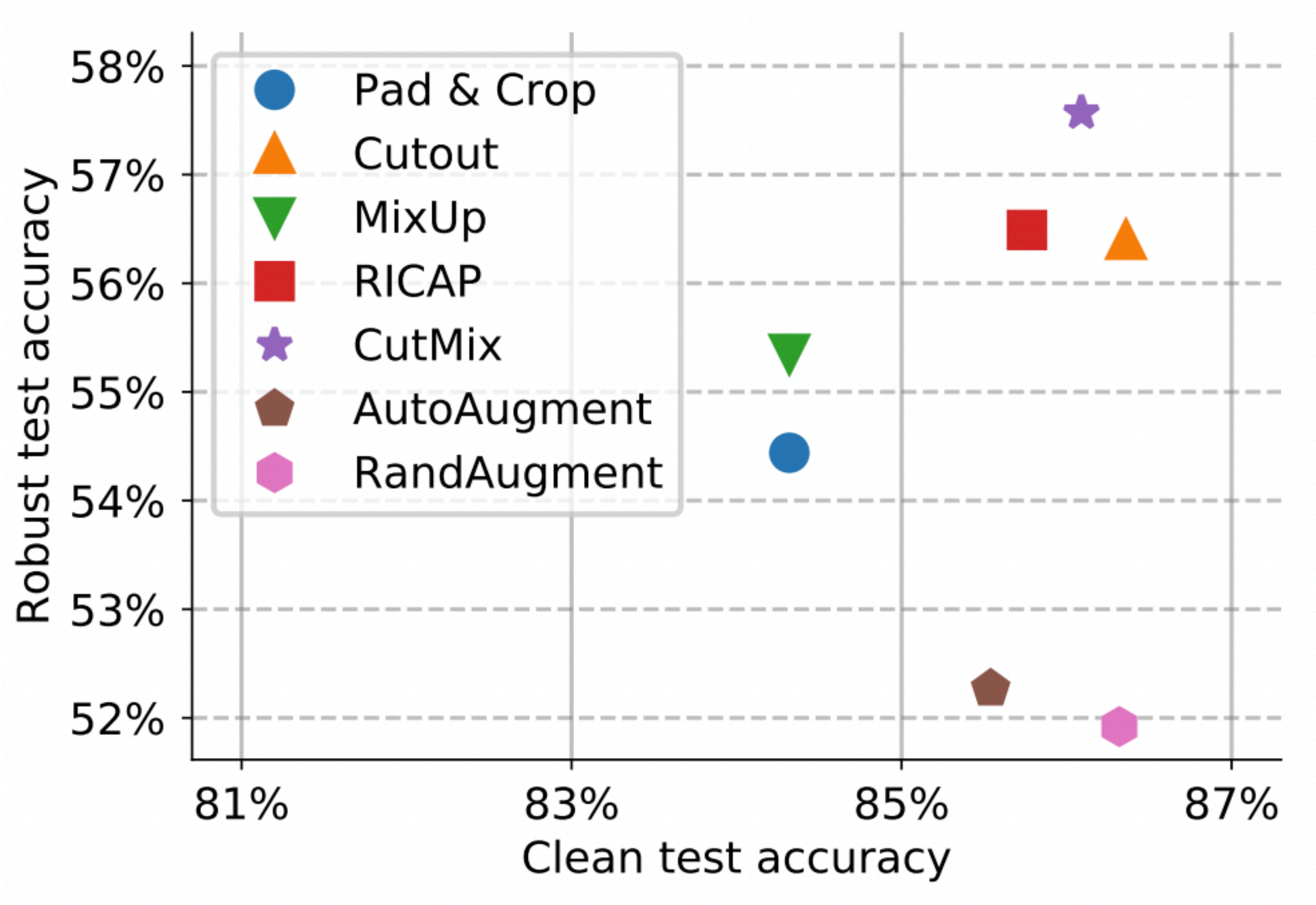
	CIFAR-10		CIFAR-100	
	Clean	Adversarial	Clean	Adversarial
Normal Training	96.0	0.0	81.0	0.0
Adversarial Training	87.3	45.8	59.1	24.3
Adv. Pre-Training and Tuning	87.1	57.4	59.2	33.5

Data augmentation

… again

- Models can also squeeze more out of the existing data using data augmentation
- E.g., an effective data augmentation technique, combined with adversarial training and a **parameter exponential moving average**, is **CutMix**

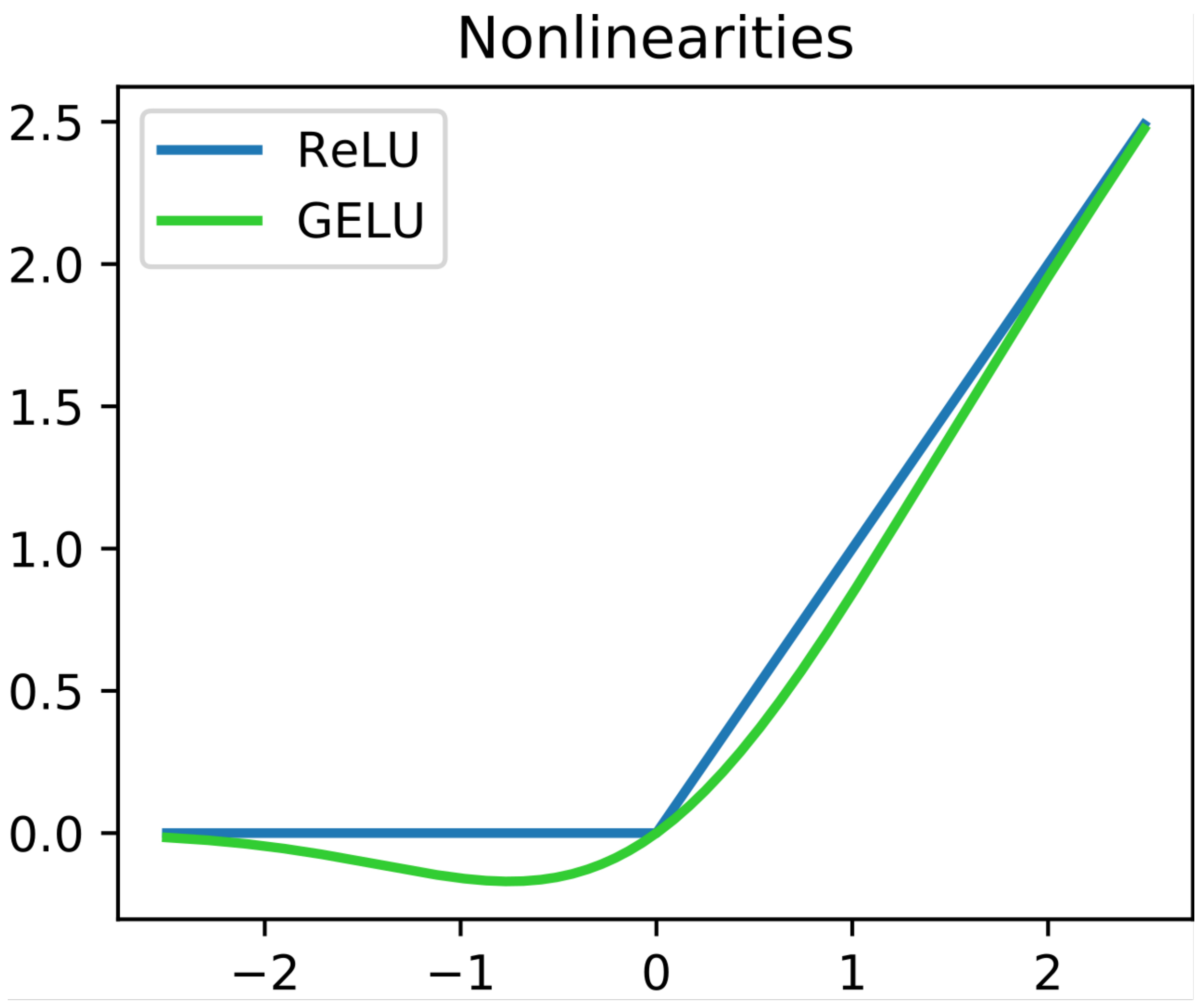




Choice of activation functions

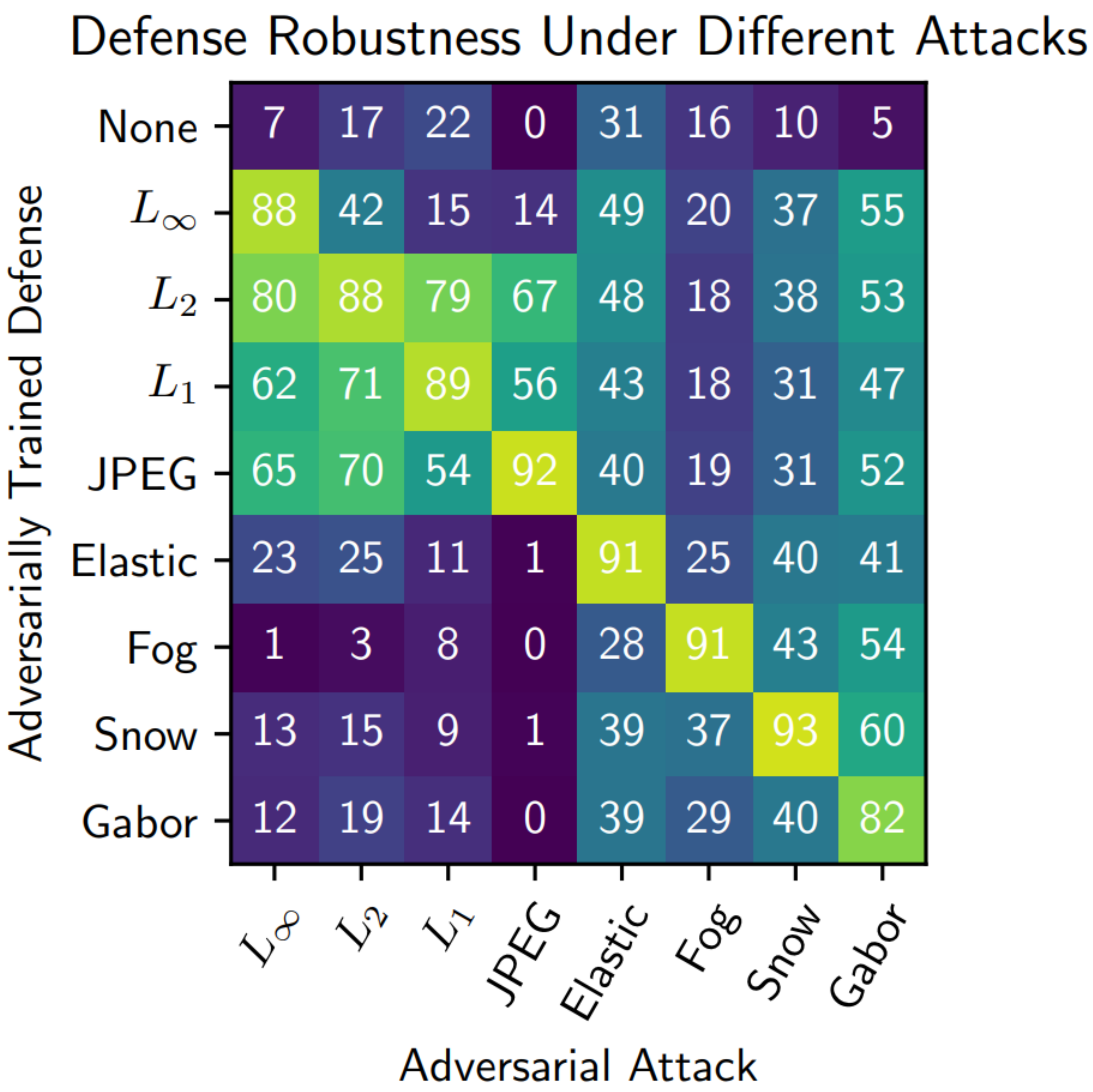
- Sharp activation functions such as ReLUs make adversarial training less effective
- By improving gradient quality for both the adversarial attacker and the network optimizer, smooth activations such as GELUs improve adversarial training

Model	ImageNet Adversarial Accuracy
ResNet-50 with ReLUs	26.41%
ResNet-50 with GELUs	35.51%



Unforeseen adversaries

- In practice, attackers could use unforeseen or novel attacks whose specifications are not known during training
- Models are far less robust to attacks they have not trained against, even if they have trained against other attacks
- To estimate robustness to unforeseen attacks, we should measure robustness to multiple attacks not encountered during training



Summary

- Adversarial examples present a challenging form of distribution shift: harmful by definition and continuously evolving against our best defenses
- In high dimensions, adversaries have much greater flexibility in terms of the space of possible subtle changes to the input that can degrade the model
- It's not a little bug that needs a little patch — much more work and evaluation are required to understand how to build stronger, more robust models
- Currently, our best defenses are adversarial training against attacks we may expect and rigorous evaluation against potential unforeseen attacks