Outline for today

- Alternative descriminator for WGANs, Binkowski
- Adversarial attacks on deep nets
  - Sign gradient attack, DeepFool, rotations and translations
  - Universal adversarial examples, transferability between nets
  - Adversarial physical objects
- Towards robustness via the the convex outer-polytope and/or sparsification of the network input or weights.

Generative deep nets (Goodfellow et al. 14'[2])

Example of a deep convolutional generator:

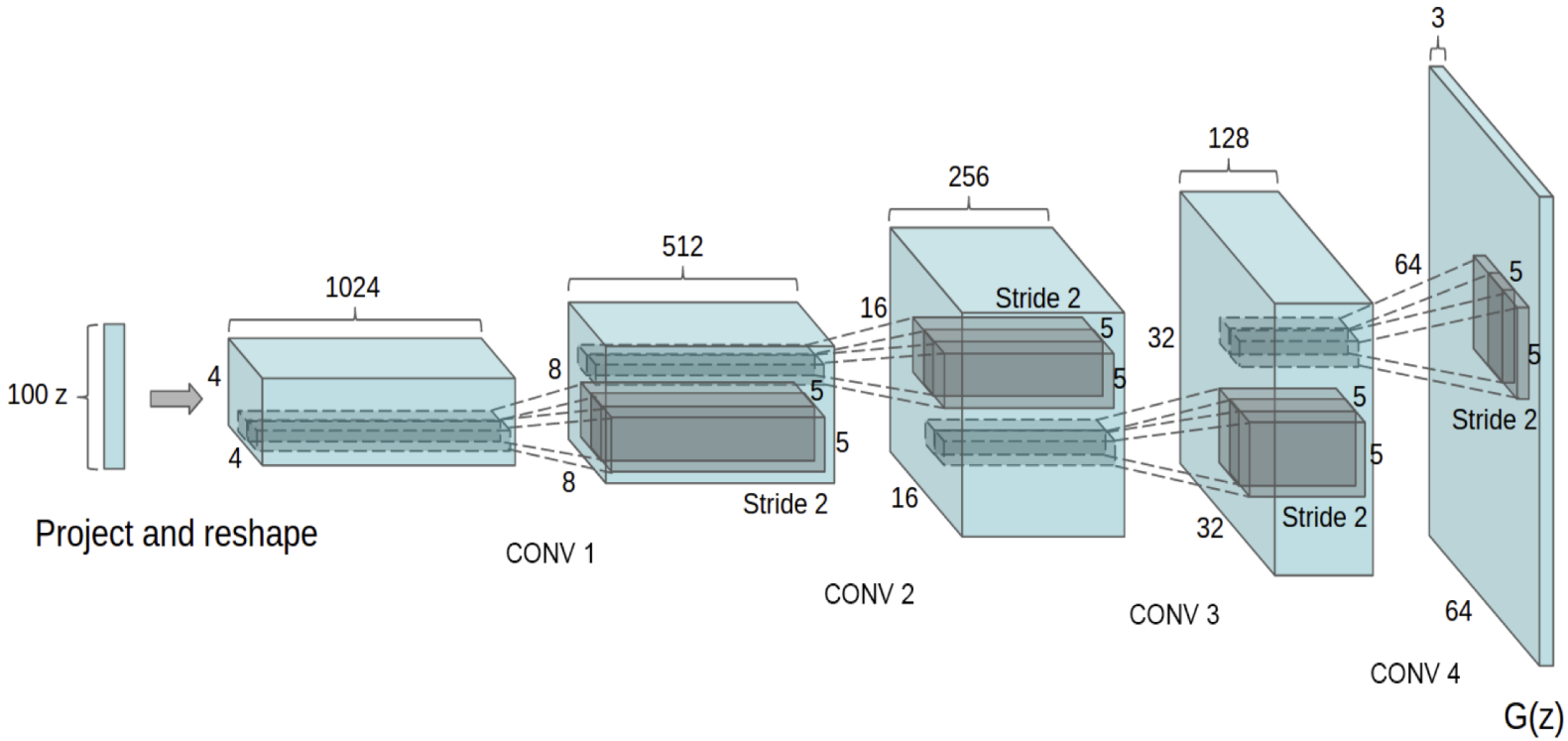


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. [1]

[1] https://arxiv.org/pdf/1511.06434.pdf

[2] https://arxiv.org/pdf/1406.2661.pdf

Generative deep nets (Goodfellow et al. 14'[5])

Train only the generator network parameters using the objective

$$\min_G \max_D n^{-1} \sum_{\mu=1}^{n} \log(D(x_\mu, y_\mu)) + p^{-1} \sum_p \log\left(1 - D(G(z_p), y_p)\right)$$

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, \ldots, n_{\text{critic}}$ **do**
    for $i = 1, \ldots, m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

3

Use as the discriminator a measure between probabilities of generative and real data sets (Binkowski et al. 18' [4]).

[3] https://arxiv.org/pdf/1704.00028.pdf

[4] https://arxiv.org/pdf/1801.01401.pdf

[5] https://arxiv.org/pdf/1406.2661.pdf

Adversarial examples for deep nets (Goodfellow et al. 15[6])

Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

[6] `https://arxiv.org/pdf/1412.6572.pdf`

DeepFool algorithm (Moosavi-Dezfooli et al. 15[7])

Algorithm 2 DeepFool: multi-class case

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

Alternative to Goodfellow approach of
$\hat{r}(x_\mu) = \epsilon \text{sign}(\text{grad}_x \updownarrow(\theta; x_\mu, y_\mu)$.

[7] https://arxiv.org/pdf/1511.04599.pdf

DeepFool algorithm (Moosavi-Dezfooli et al. 15[8])

Classifier	Test error	$\hat{\rho}_{adv}$ [DeepFool]	time	$\hat{\rho}_{adv}$ [4]	time	$\hat{\rho}_{adv}$ [18]	time
LeNet (MNIST)	1%	2.0×10^{-1}	110 ms	1.0	20 ms	2.5×10^{-1}	> 4 s
FC500-150-10 (MNIST)	1.7%	1.1×10^{-1}	50 ms	3.9×10^{-1}	10 ms	1.2×10^{-1}	> 2 s
NIN (CIFAR-10)	11.5%	2.3×10^{-2}	1100 ms	1.2×10^{-1}	180 ms	2.4×10^{-2}	>50 s
LeNet (CIFAR-10)	22.6%	3.0×10^{-2}	220 ms	1.3×10^{-1}	50 ms	3.9×10^{-2}	>7 s
CaffeNet (ILSVRC2012)	42.6%	2.7×10^{-3}	510 ms*	3.5×10^{-2}	50 ms*	-	-
GoogLeNet (ILSVRC2012)	31.3%	1.9×10^{-3}	800 ms*	4.7×10^{-2}	80 ms*	-	-

Table 1: The adversarial robustness of different classifiers on different datasets. The time required to compute one sample for each method is given in the time columns. The times are computed on a Mid-2015 MacBook Pro without CUDA support. The asterisk marks determines the values computed using a GTX 750 Ti GPU.

Average relative error of adversarial example $\hat{r}(x)$ such that $f(x) \neq f(x + \hat{r}(x))$: $\hat{\rho}_{adv}(f) = |\mathcal{D}|^{-1} \sum_{x \in \mathcal{D}} \frac{\|\hat{r}(x)\|_2}{\|x\|_2}$

[8] https://arxiv.org/pdf/1511.04599.pdf

Rotations and Translations for CNNs (Engstrom et al. 18[9])

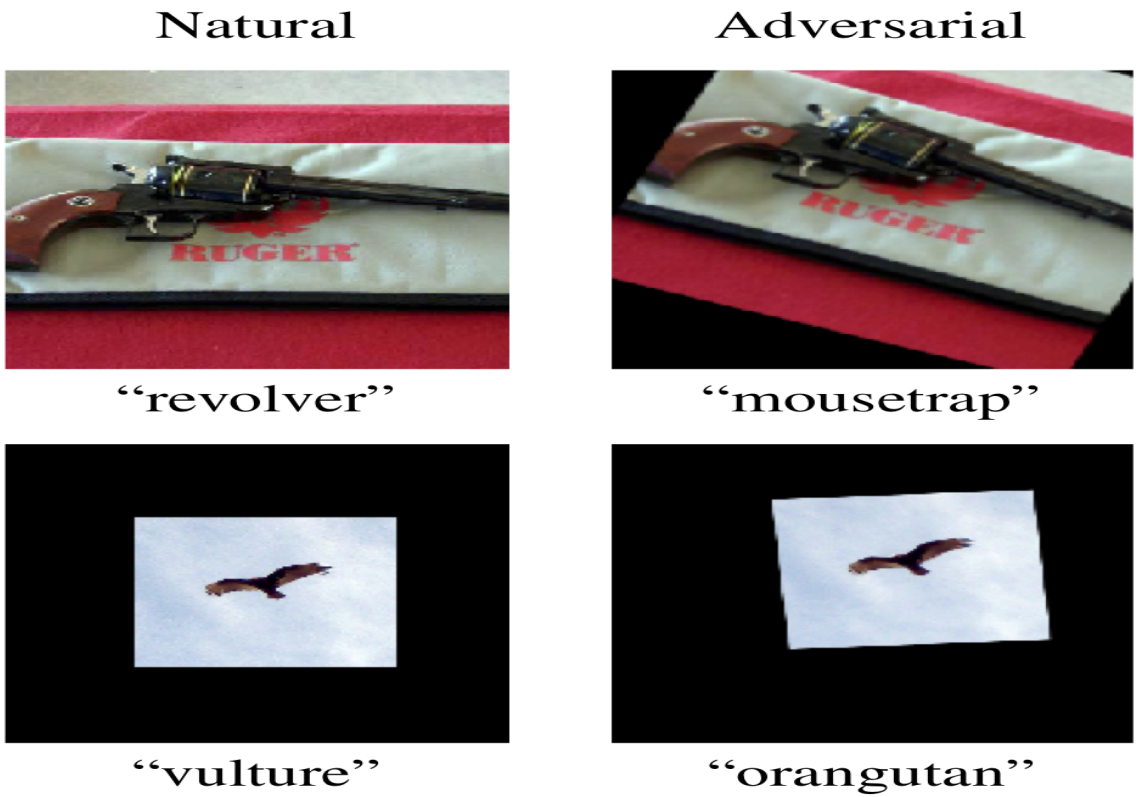


Figure 1: Examples of adversarial transformations and their predictions in the standard and "black canvas" setting.

[9] https://arxiv.org/pdf/1712.02779.pdf

Rotations and Translations for CNNs (Engstrom et al. 18[10])

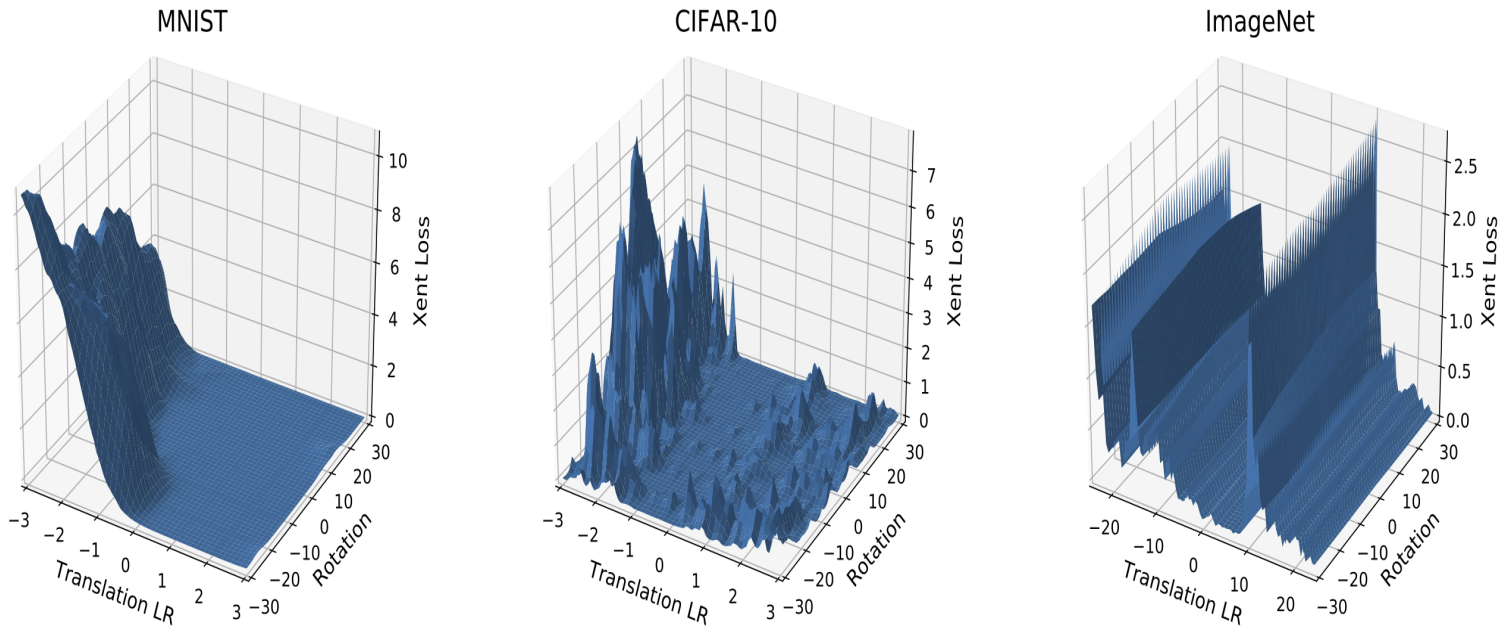


Figure 3: Loss landscape of a random example for each dataset when performing left-right translations and rotations. Translations and rotations are restricted to 10% of the image pixels and 30 deg respectively. We observe that the landscape is significantly non-concave, making rendering FO methods for adversarial example generation powerless. Additional examples are visualized in Figure 9 of the Appendix.

[10] https://arxiv.org/pdf/1712.02779.pdf

Universal adversary (Moosavi-Dezfooli et al. 16[11])

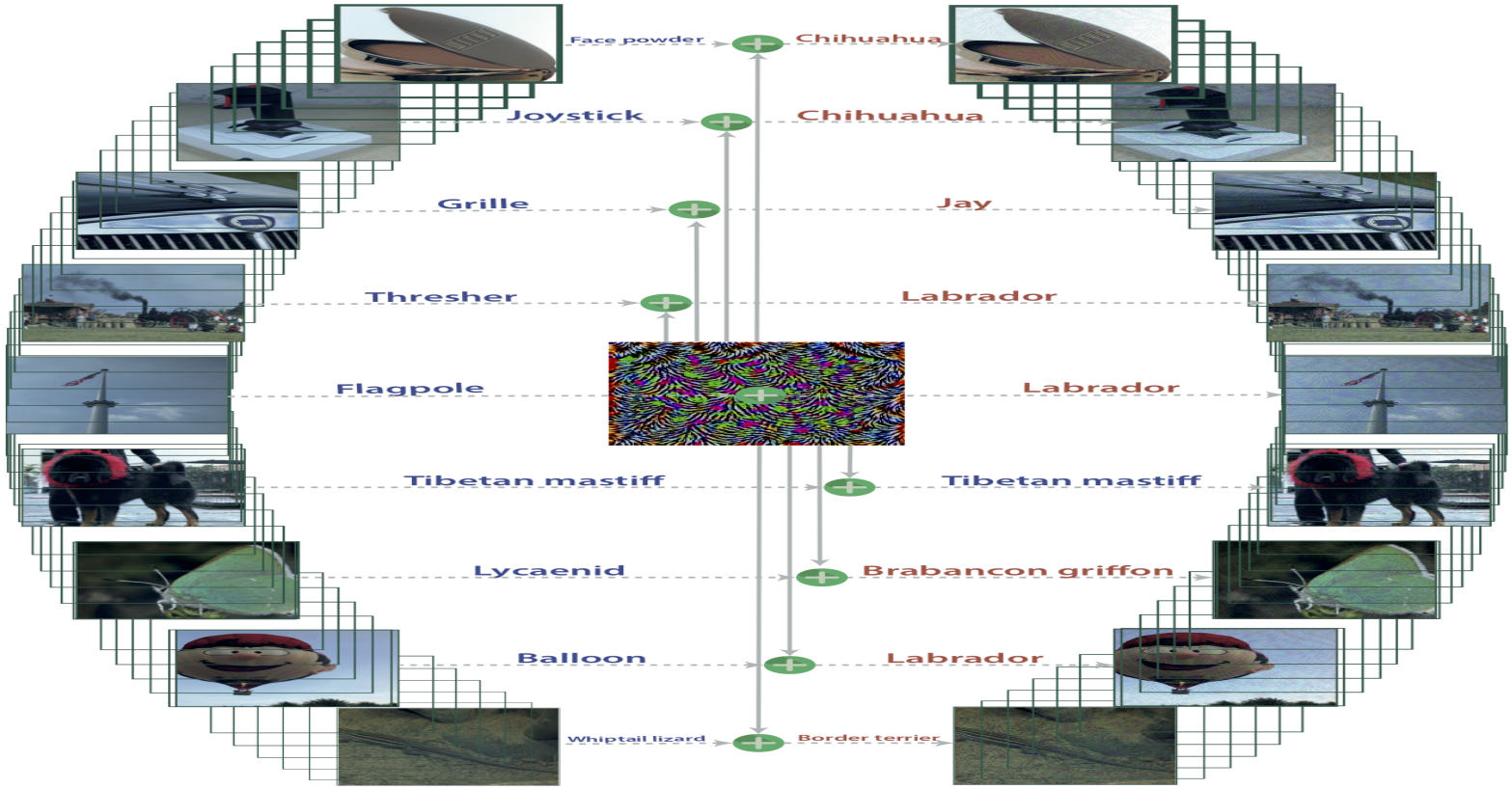


Figure 1: When added to a natural image, a universal perturbation image causes the image to be misclassified by the deep neural network with high probability. *Left images:* Original natural images. The labels are shown on top of each arrow. *Central image:* Universal perturbation. *Right images:* Perturbed images. The estimated labels of the perturbed images are shown on top of each arrow.

[11] https://arxiv.org/pdf/1610.08401.pdf

Transferability between nets (Liu et al. 16[12])

	RMSD	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
-ResNet-152	17.17	0%	0%	0%	0%	0%
-ResNet-101	17.25	0%	1%	0%	0%	0%
-ResNet-50	17.25	0%	0%	2%	0%	0%
-VGG-16	17.80	0%	0%	0%	6%	0%
-GoogLeNet	17.41	0%	0%	0%	0%	5%

Table 4: Accuracy of non-targeted adversarial images generated using the optimization-based approach. The first column indicates the average RMSD of the generated adversarial images. Cell (i, j) corresponds to the accuracy of the attack generated using four models except model i (row) when evaluated over model j (column). In each row, the minus sign "–" indicates that the model of the row is not used when generating the attacks. Results of top-5 accuracy can be found in the appendix (Table 14).

[12] https://arxiv.org/pdf/1611.02770.pdf

Transferability between nets (Liu et al. 16[13])

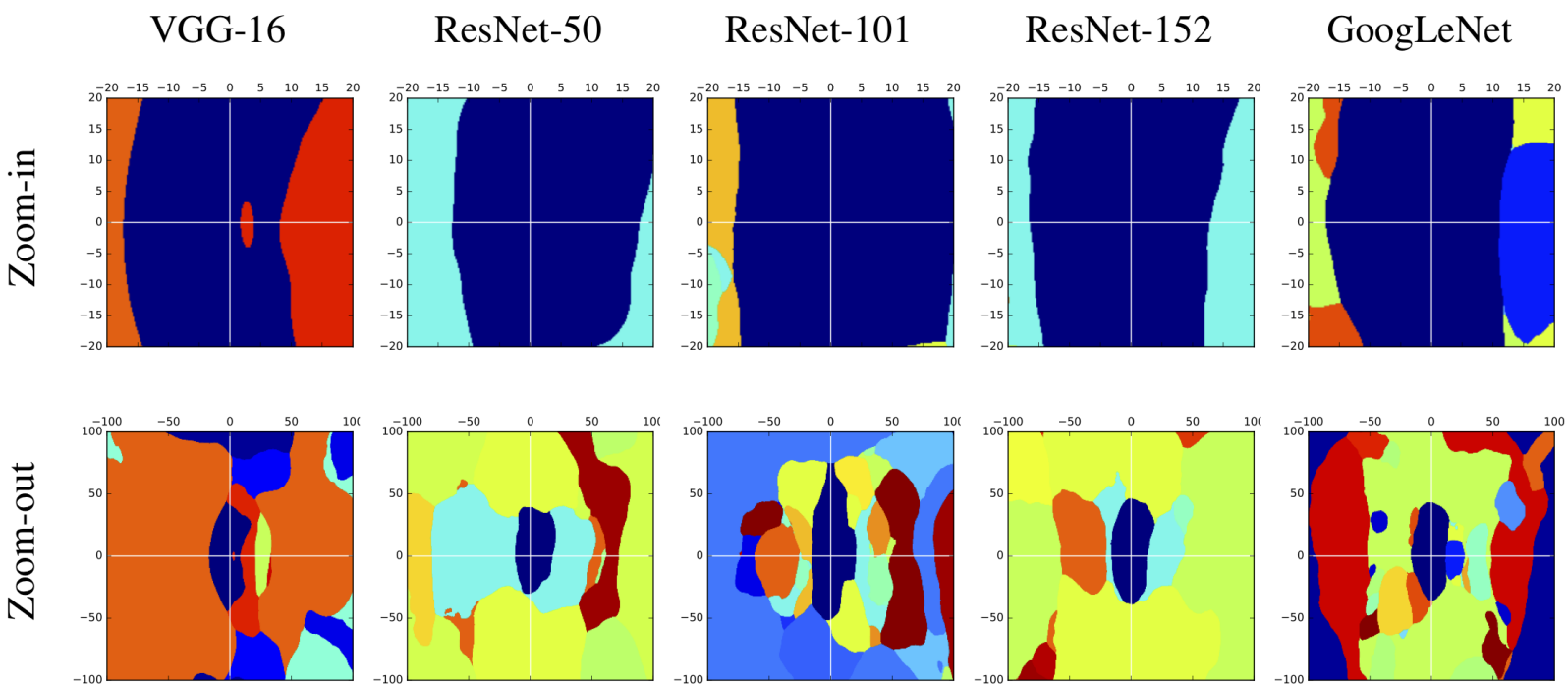


Figure 3: Decision regions of different models. We pick the same two directions for all plots: one is the gradient direction of VGG-16 (x-axis), and the other is a random orthogonal direction (y-axis). Each point in the span plane shows the predicted label of the image generated by adding a noise to the original image (e.g., the origin corresponds to the predicted label of the original image). The units of both axises are 1 pixel values. All sub-figure plots the regions on the span plane using the same color for the same label. The image is in Figure 2.

[13] https://arxiv.org/pdf/1611.02770.pdf

Adversarial physical object: Turtle (Athalye et al. 17[14])

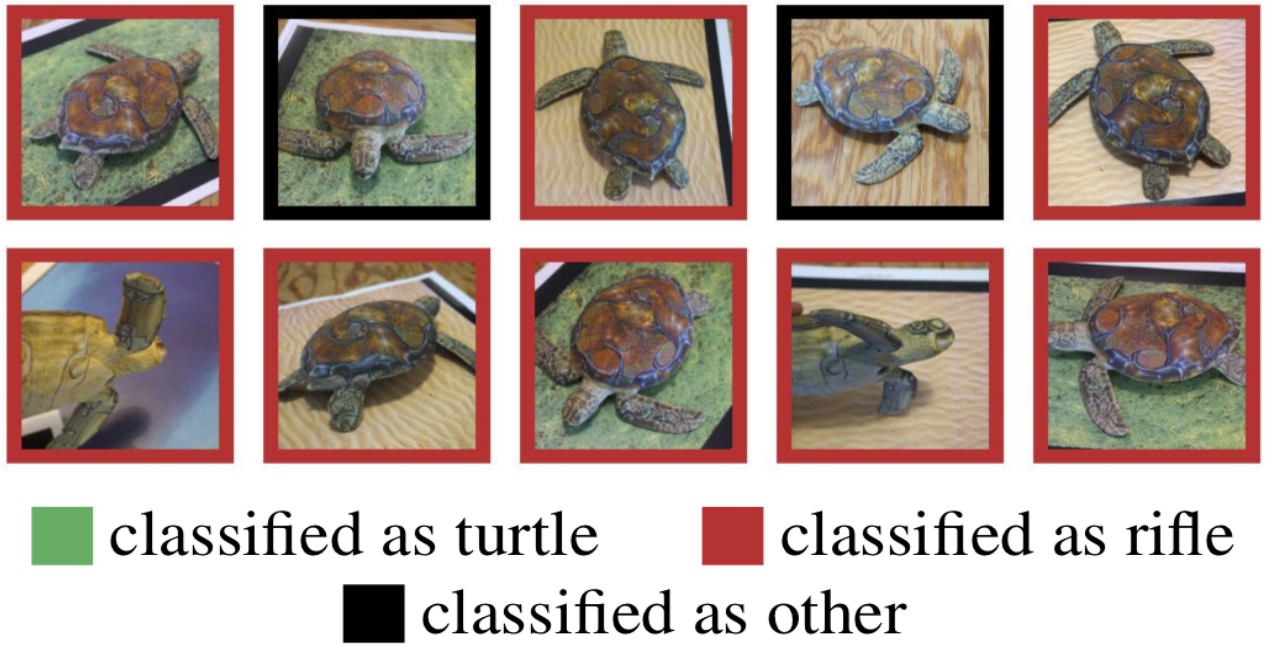


Figure 1. Randomly sampled poses of a 3D-printed turtle adversarially perturbed to classify as a rifle at every viewpoint[2]. An unperturbed model is classified correctly as a turtle nearly 100% of the time.

[14] https://arxiv.org/pdf/1707.07397.pdf

Adversarial graffiti (Eykholt et al. 17[15])

Figure 1: The left image shows real graffiti on a Stop sign, something that most humans would not think is suspicious. The right image shows our a physical perturbation applied to a Stop sign. We design our perturbations to mimic graffiti, and thus "hide in the human psyche."

Table 5: A camouflage art attack on GTSRB-CNN. See example images in Table 1. The targeted-attack success rate is 80% (true class label: Stop, target: Speed Limit 80).

Distance & Angle	Top Class (Confid.)	Second Class (Confid.)
5′ 0°	Speed Limit 80 (0.88)	Speed Limit 70 (0.07)
5′ 15°	Speed Limit 80 (0.94)	Stop (0.03)
5′ 30°	Speed Limit 80 (0.86)	Keep Right (0.03)
5′ 45°	Keep Right (0.82)	Speed Limit 80 (0.12)
5′ 60°	Speed Limit 80 (0.55)	Stop (0.31)
10′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.006)
10′ 15°	Stop (0.75)	Speed Limit 80 (0.20)
10′ 30°	Speed Limit 80 (0.77)	Speed Limit 100 (0.11)
15′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.01)
15′ 15°	Stop (0.90)	Speed Limit 80 (0.06)
20′ 0°	Speed Limit 80 (0.95)	Speed Limit 100 (0.03)
20′ 15°	Speed Limit 80 (0.97)	Speed Limit 100 (0.01)
25′ 0°	Speed Limit 80 (0.99)	Speed Limit 70 (0.0008)
30′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)
40′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)

[15] https://arxiv.org/pdf/1707.08945.pdf

Provable defense: convex polytope pt. 1 (Wong et al. 17[16])

Possible output of net $f_\theta(\cdot)$ from bounded perturbation is a non-convex set, say $\mathcal{Z}_\epsilon(x) = \{f_\theta(x+\delta) : \|\delta\|_\infty \leq \epsilon\}$. A convex outer-polytope of $\mathcal{Z}_\epsilon(x)$, say $\mathcal{Z}_\epsilon^{conv}(x)$, can be computed by replacing the input to each activation with a two dimensional convex set:

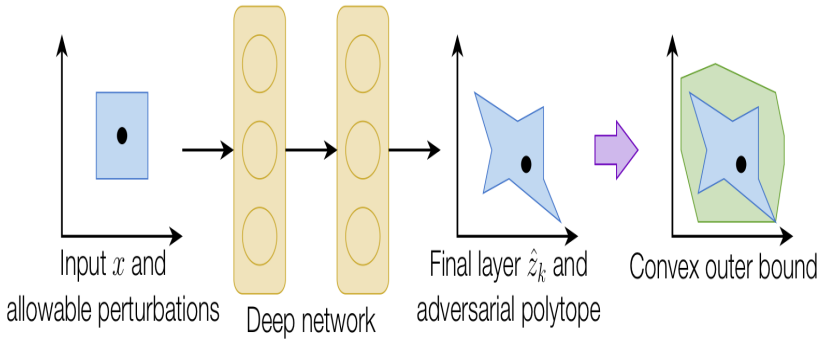


Figure 1. Conceptual illustration of the (non-convex) adversarial polytope, and an outer convex bound.

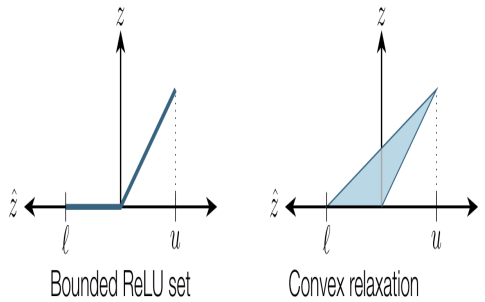


Figure 2. Illustration of the convex ReLU relaxation over the bounded set $[\ell, u]$.

Requires knowledge of lower and upper bound for each input to a nonlinear activation. Let $c = e_\ell - e_{\ell'}$ or $c = 2e_\ell - 1_{|class|}$ and solve:

$$\min_{\hat{z}_\ell \in \hat{\mathcal{Z}}_\epsilon(x)} c^T \hat{z}_\ell$$ and if nonnegative then robust to ϵ perturbation.

[16] https://arxiv.org/pdf/1711.00851.pdf

Provable defense: convex polytope pt. 2 (Wong et al. 17[17])

Algorithm 1 Computing Activation Bounds

input: Network parameters $\{W_i, b_i\}_{i=1}^{k-1}$, data point x, ball size ϵ

// initialization

$\hat{\nu}_1 := W_1^T$

$\gamma_1 := b_1^T$

$\ell_2 := x^T W_1^T + b_1^T - \epsilon \|W_1^T\|_{1,:}$

$u_2 := x^T W_1^T + b_1^T + \epsilon \|W_1^T\|_{1,:}$

// $\|\cdot\|_{1,:}$ for a matrix here denotes ℓ_1 norm of all columns

for $i = 2, \ldots, k-1$ **do**

  form $\mathcal{I}_i^-$, $\mathcal{I}_i^+$, $\mathcal{I}_i$; form D_i as in (10)

  // initialize new terms

  $\nu_{i,\mathcal{I}_i} := (D_i)_{\mathcal{I}_i} W_i^T$

  $\gamma_i := b_i^T$

  // propagate existing terms

  $\nu_{j,\mathcal{I}_j} := \nu_{j,\mathcal{I}_j} D_i W_i^T, \quad j = 2, \ldots, i-1$

  $\gamma_j := \gamma_j D_i W_i^T, \quad j = 1, \ldots, i-1$

  $\hat{\nu}_1 := \hat{\nu}_1 D_i W_i^T$

  // compute bounds

  $\psi_i := x^T \hat{\nu}_1 + \sum_{j=1}^{i} \gamma_j$

  $\ell_{i+1} := \psi_i - \epsilon \|\hat{\nu}_1\|_{1,:} + \sum_{j=2}^{i} \sum_{i' \in \mathcal{I}_i} \ell_{j,i'} [-\nu_{j,i'}]_+$

  $u_{i+1} := \psi_i + \epsilon \|\hat{\nu}_1\|_{1,:} - \sum_{j=2}^{i} \sum_{i' \in \mathcal{I}_i} \ell_{j,i'} [\nu_{j,i'}]_+$

end for

output: bounds $\{\ell_i, u_i\}_{i=2}^{k}$

[17] https://arxiv.org/pdf/1711.00851.pdf

Provable defense: convex polytope pt. 3 (Wong et al. 17[18])

Table 1. Error rates for various problems and attacks, and our robust bound for baseline and robust models.

PROBLEM	ROBUST	ϵ	TEST ERROR	FGSM ERROR	PGD ERROR	ROBUST ERROR BOUND
MNIST	X	0.1	1.07%	50.01%	81.68%	100%
MNIST	√	0.1	1.80%	3.93%	4.11%	5.82%
FASHION-MNIST	X	0.1	9.36%	77.98%	81.85%	100%
FASHION-MNIST	√	0.1	21.73%	31.25%	31.63%	34.53%
HAR	X	0.05	4.95%	60.57%	63.82%	81.56%
HAR	√	0.05	7.80%	21.49%	21.52%	21.90%
SVHN	X	0.01	16.01%	62.21%	83.43%	100%
SVHN	√	0.01	20.38%	33.28%	33.74%	40.67%

[18] `https://arxiv.org/pdf/1711.00851.pdf`

Robustness via sparsification (Gopalakrishnan et al. 18[19])

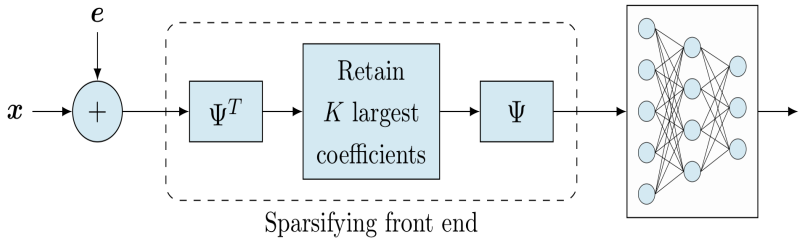


Figure 1: Sparsifying front end defense: For a basis in which the input is sparse, the input is projected onto the subspace spanned by the K largest basis coefficients. This attenuates the impact of the attack by K/N, where N is the input dimension.

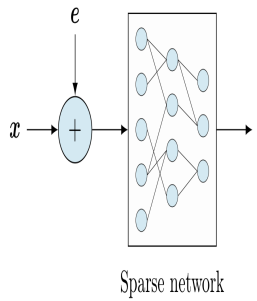


Figure 2: Network sparsity defense: Imposing sparsity *within* the neural network attenuates the worst-case growth of the attack as it flows up the network.

Theorem 2. *Consider an ℓ_∞-constrained input perturbation $\boldsymbol{e}_0 = \boldsymbol{e}$, with $\|\boldsymbol{e}\|_\infty \leq \epsilon$. Suppose that we impose ℓ_1 constraints on the weights at each layer as follows:*

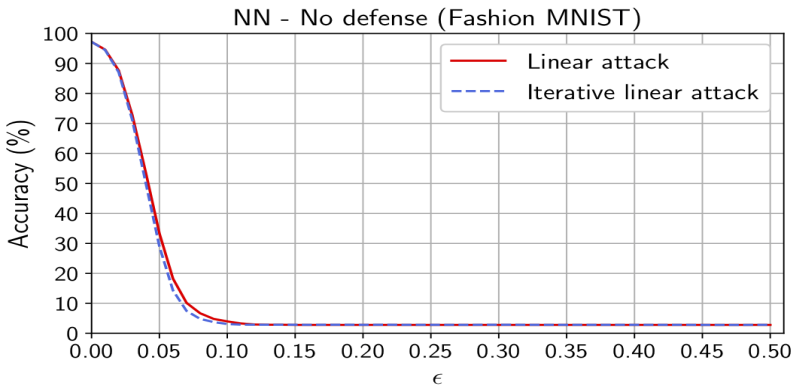
$$\|\boldsymbol{w}_{ij}\|_1 \leq \gamma_j \quad \forall\ i$$

Then the effect of the perturbation is ℓ_∞-bounded at each layer:

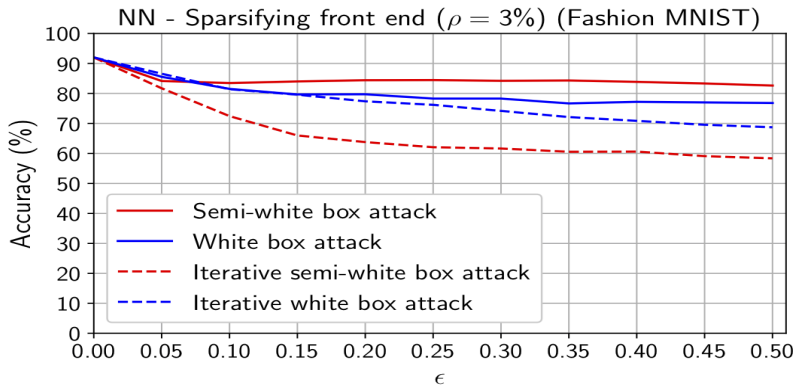
$$\|\boldsymbol{e}_j\|_\infty \leq \epsilon \prod_{l=1}^{j} \gamma_l \tag{2}$$

[19] https://arxiv.org/abs/1810.10625

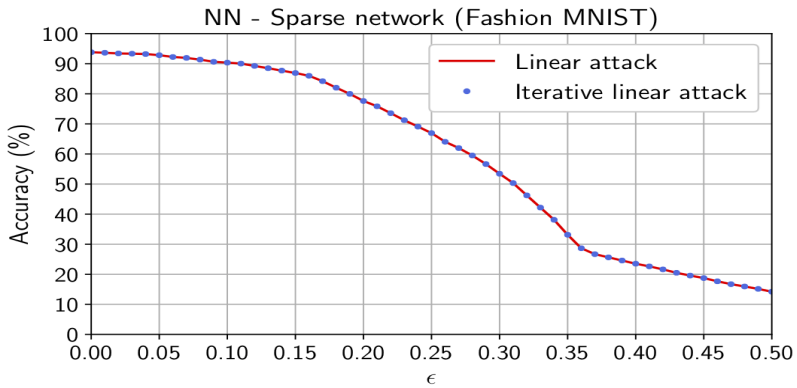
Robustness via sparsification (Gopalakrishnan et al. 18[20])



(a) No defense



(b) With sparsifying front end



(c) With network sparsity

Figure 6: Fashion-MNIST: Binary classification accuracies as a function of ϵ

[20] https://arxiv.org/abs/1810.10625