Introduction to Adversarial in Deep Learning

Huang, Zhichao

Hong Kong University of Science and Technology

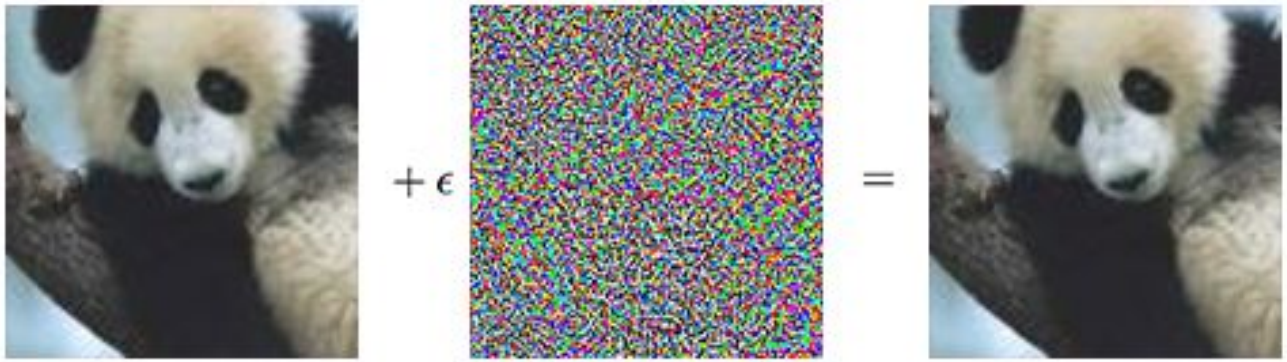
huangzhichao95@gmail.com

November 25, 2018

Overview

Adversarial Examples

- When adding small artificially chosen noise, the classification result will be totally different from the original result.



- The difference is hard to notice and the noise is measured in L_p norm. If the noise is small in L_p norm, it is hard for human to notice the difference.

Category

- White Box, Black Box
- Target Attack, Non-target Attack

White Box

- White box means you will have the knowledge of all the model, which means you can get the derivatives of the model with respect to the input.
- Optimization Formulation

$$\max_{||\mathbf{x}'-\mathbf{x}||<\epsilon} L(x'; y)$$

Fast Gradient Signed Method(FGSM)

- The neural network is linear in a small region
- Taylor Expansion

$$L(x'; y) = L(x; y) + \nabla_{\mathbf{x}} L(x; y)^T (\mathbf{x}' - \mathbf{x})$$

- The maximum Loss is to set

$$\mathbf{x}_{adv} = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} L(x; y))$$

Iterative Method

- Use FGSM is used in an iterative way

$$x_{adv}^{0} = \mathbf{x}$$

$$x_{adv}^{n} = \mathrm{Clip}_{\mathbf{x},\epsilon}(\mathbf{x}_{adv}^{n-1} + \alpha \mathrm{sign}(\nabla_{\mathbf{x}} L(x; y)))$$

- α can be set to $\frac{\epsilon}{N}$ and no clip is needed

Projected Gradient Descent(PGD)

- No Signed Anymore

$$x_{adv}^{0} = \mathbf{x}$$

$$x_{adv}^{n} = \mathrm{Clip}_{\mathbf{x},\epsilon}(\mathbf{x}_{adv}^{n-1} + \alpha \nabla_{\mathbf{x}} L(x; y))$$

Carlini & Wagner Attack(C&W)

- A new loss function:

$$L(\mathbf{x}; y) = \max(\max_{i \neq y} \mathbf{z}_i - z_y, -k)$$

  $\mathbf{z}$ is the output before the softmax layer. k is the margin, usually set to zero.

- A unconstrained optimization problem:

$$\max(L(\mathbf{x}; y) + \lambda ||\mathbf{x}' - \mathbf{x}||^2)$$

  λ is the penalty for getting too far from the origin image.

Some result

MNIST		CIFAR	
mean	prob	mean	prob
16	100%	13	100%
56	100%	58	100%
45	100%	110	100%
1.76	100%	0.33	100%
–	-	–	-
0.16	100%	0.013	100%
0.26	42%	0.029	51%
0.19	100%	0.014	100%

Backward Pass Differentiable Approximation(BPDA)

Another White Box method. Introduce Later ...

Substitute Model

- It is find that If we train one model and use gradient ascent to attack this model. The adversarial can attack another model in the same domain.
- Closer architecture and closer training data leads to higher success rate of attack.
- The reason may be similar decision boundary of different network. But it is still unknown.

Substitute Model

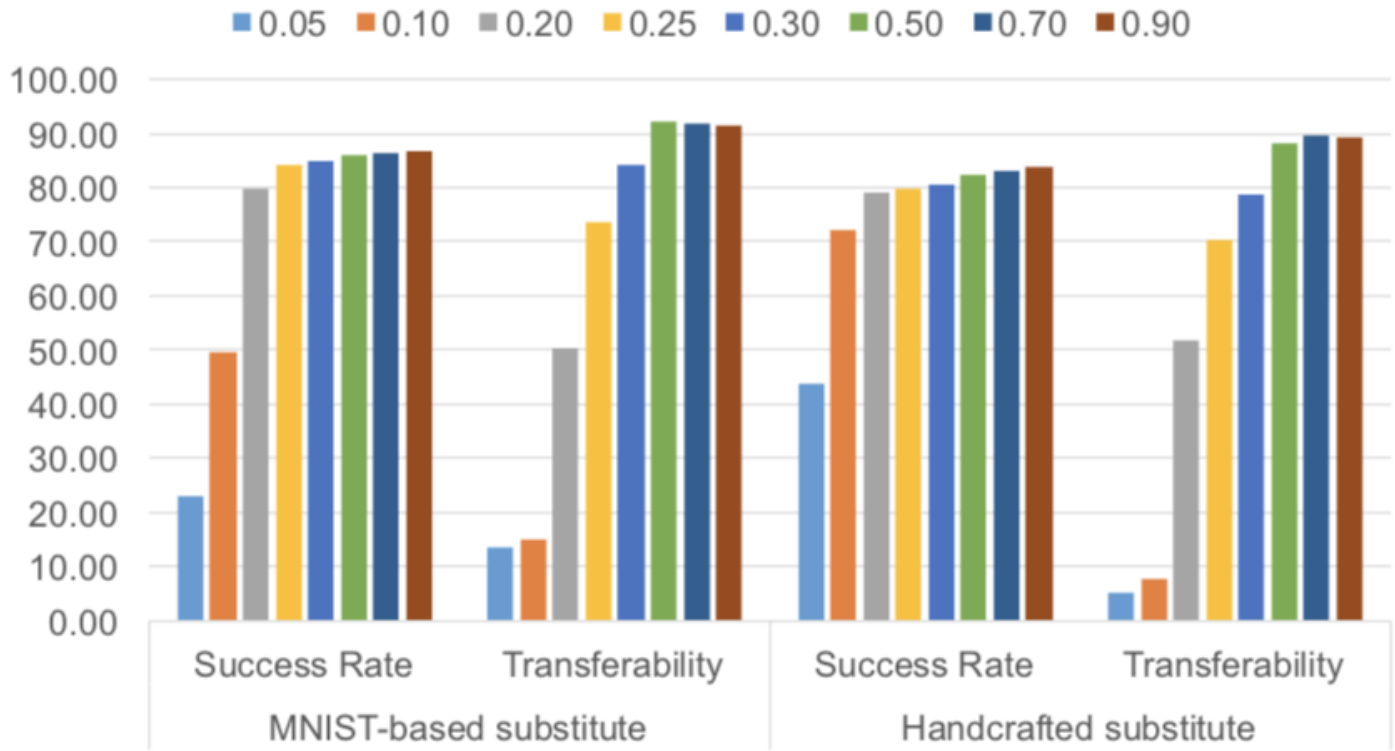
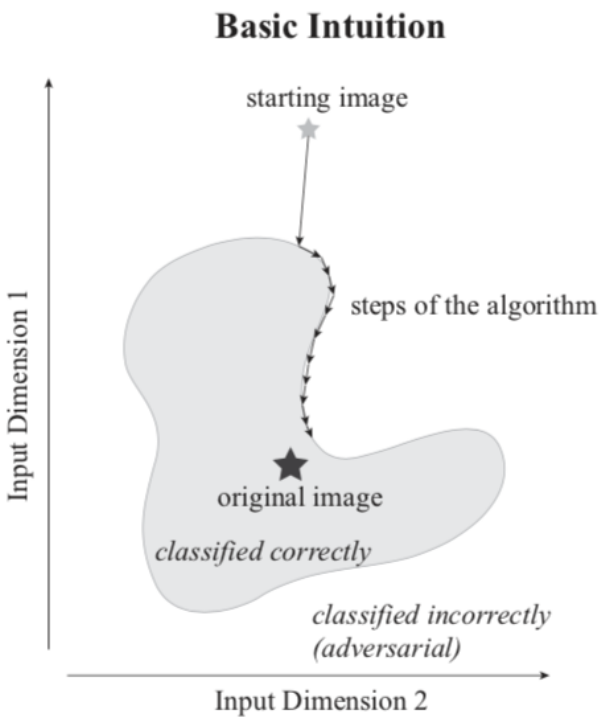
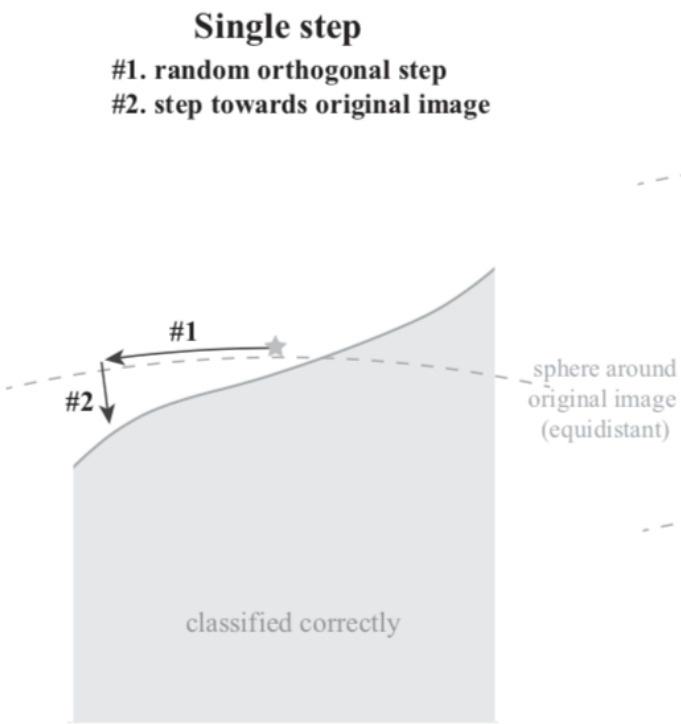
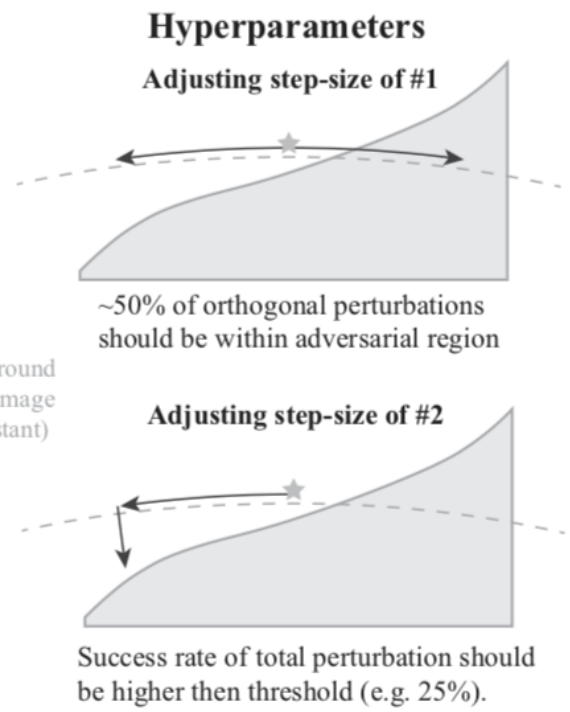


Figure 5: **Success Rate and Transferability of Adversarial Samples for the MetaMind attacks:** performed using MNIST-based and handcrafted substitutes: each bar corresponds to a different perturbation input variation.

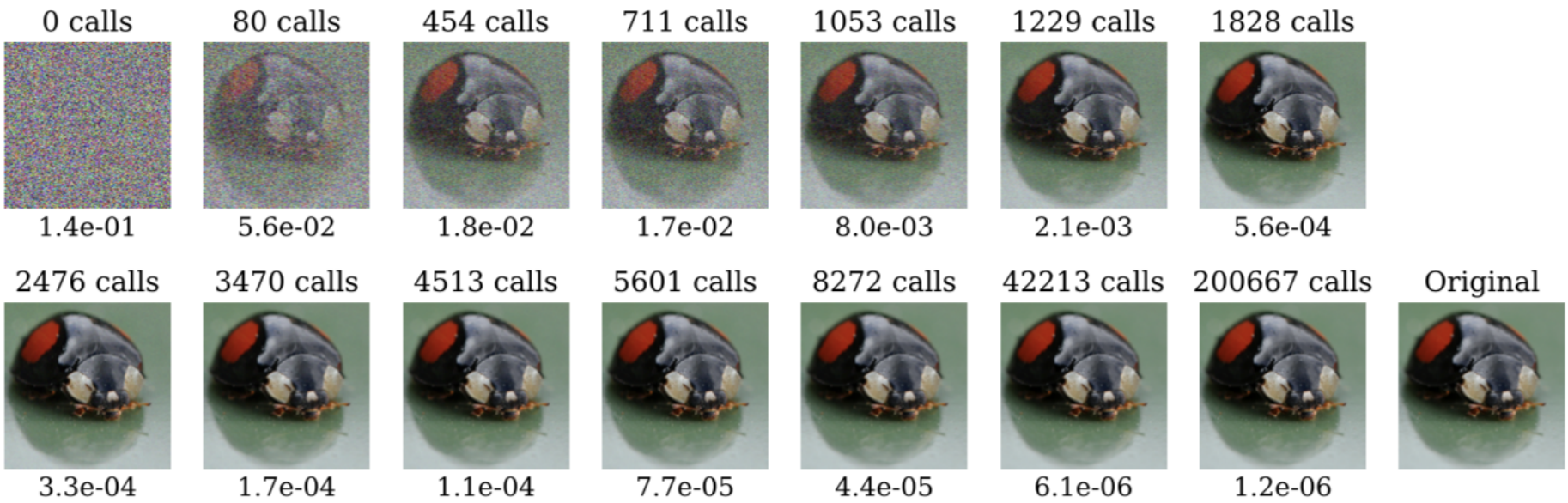
Boundary Attack







Boundary Attack



Zero Order Optimization(ZOO)

- Estimate gradient using

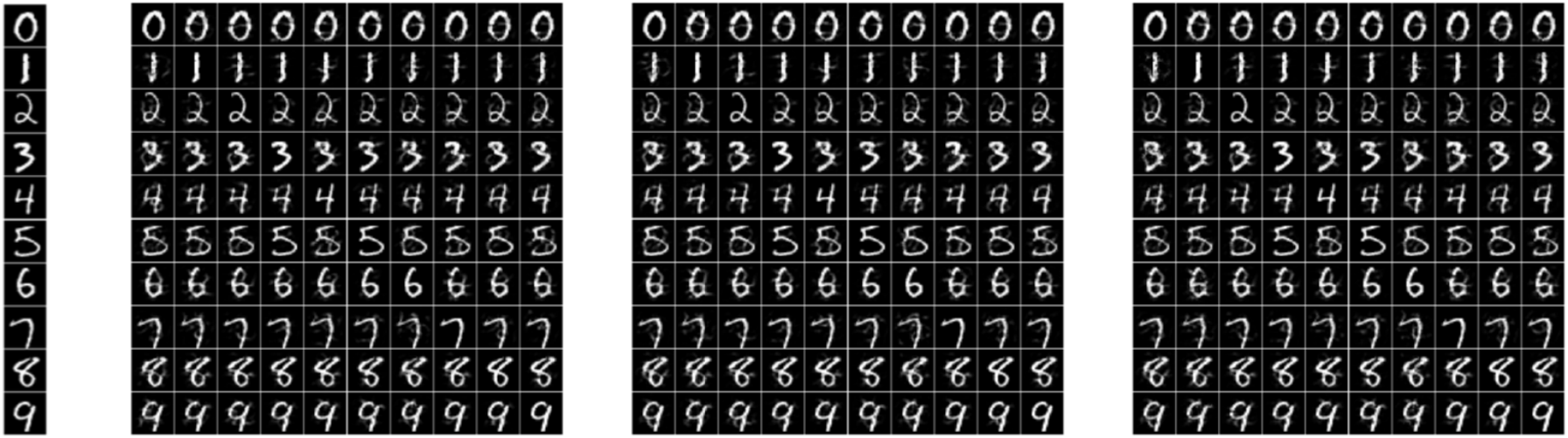
$$\hat{f'(x)} = \frac{f(x + he_i) - f(x - he_i)}{2h}$$

- At each time, only estimate the gradient of one pixel (or a batch of pixels) and update it (them);
- Hierarchical attack
- Optimize the important pixels first

Zero Order Optimization(ZOO)

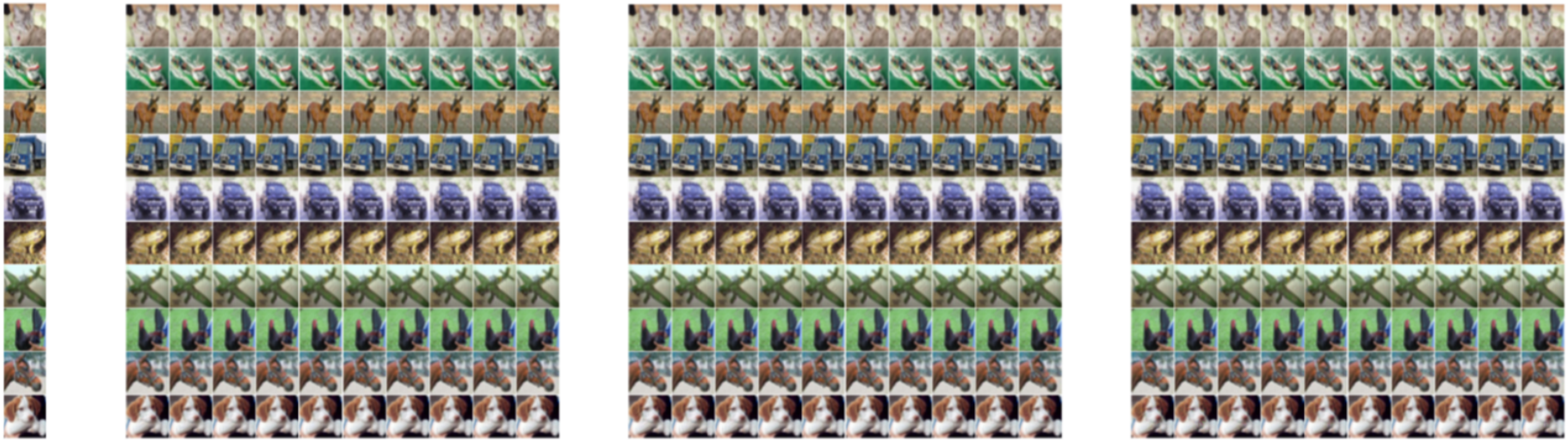
	MNIST					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	1.48066	0.48 min	100 %	2.00661	0.53 min
Black-box (Substitute Model + FGSM)	40.6 %	-	0.002 sec (+ 6.16 min)	7.48 %	-	0.002 sec (+ 6.16 min)
Black-box (Substitute Model + C&W)	33.3 %	3.6111	0.76 min (+ 6.16 min)	26.74 %	5.272	0.80 min (+ 6.16 min)
Proposed black-box (ZOO-ADAM)	100 %	1.49550	1.38 min	98.9 %	1.987068	1.62 min
Proposed black-box (ZOO-Newton)	100 %	1.51502	2.75 min	98.9 %	2.057264	2.06 min
	CIFAR10					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	0.17980	0.20 min	100 %	0.37974	0.16 min
Black-box (Substitute Model + FGSM)	76.1 %	-	0.005 sec (+ 7.81 min)	11.48 %	-	0.005 sec (+ 7.81 min)
Black-box (Substitute Model + C&W)	25.3 %	2.9708	0.47 min (+ 7.81 min)	5.3 %	5.7439	0.49 min (+ 7.81 min)
Proposed Black-box (ZOO-ADAM)	100 %	0.19973	3.43 min	96.8 %	0.39879	3.95 min
Proposed Black-box (ZOO-Newton)	100 %	0.23554	4.41 min	97.0 %	0.54226	4.40 min

Zero Order Optimization(ZOO)



(a) (b) White-box C&W attack (c) ZOO-ADAM black-box attack (d) ZOO-Newton black-box attack

Zero Order Optimization(ZOO)



(a) (b) White-box C&W attack (c) ZOO-ADAM black-box attack (d) ZOO-Newton black-box attack

Evolutional Black-box Attack(Nattack)

- Consider adversarial $\mathbf{x}' \sim \mathcal{N}(\theta(\mathbf{x}), \sigma^2 I)$
- Gradient Estimation can be transformed into sampling

$$\nabla_\theta E_{\mathbf{x}' \sim \theta} L(\mathbf{x}') = E_{\mathbf{x}' \sim \theta}[L(\mathbf{x}') \nabla_\theta \log P_\theta(\mathbf{x}')]$$

Algorithm 1 Evolutional black-box adversarial attack

Input: DNN classifier $F(\cdot)$, input x and its label y, initial mean θ_0, standard deviation σ, learning rate η, sample size N, and the maximum number of iterations T

Output: θ_T, mean of the adversarial population

1: **for** $t = 0, 1, ..., T-1$ **do**
2: Sample ϵ_1,...,$\epsilon_N \sim \mathcal{N}(0, I)$
3: Compute losses $J_i = J(\theta_t + \sigma\epsilon_i)$ for $i = 1, \cdots, N$
4: Set $\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{N\sigma} \sum_{i=1}^{N} J_i \epsilon_i$
5: **end for**

Evolutional Black-box Attack(Nattack)

Defense technique	Dataset	Classification Accuracy %	Threshold & Distance	Attack success rate % BPDA	ZOO	$\mathcal{N}$ATTACK
THERM-ADV (Madry et al., 2017)	CIFAR10	88.5	0.031 (L_∞)	76.1*	0.0*	90.0*
LID (Ma et al., 2018)	CIFAR10	66.9	0.031 (L_∞)	95.0	92.9	100.0
THERM (Buckman et al., 2018)	CIFAR10	92.8	0.031 (L_∞)	100.0	0.0	100.0
SAP (Dhillon et al., 2018)	CIFAR10	93.3	0.031 (L_∞)	100.0	5.9	100.0
RSE (Liu et al., 2017)	CIFAR10	91.4	0.031 (L_∞)	–	–	100.0
CAS-ADV (Na et al., 2018)	CIFAR10	75.6	0.015 (L_∞)	85.0**	96.1	*97.7*
GUIDED DENOISER (Liao et al., 2018)	ImageNet	79.1	0.031 (L_∞)	100.0	–	95.5
RANDOMIZATION (Xie et al., 2018)	ImageNet	77.8	0.031 (L_∞)	100.0	6.7	96.5
INPUT-TRANS (Guo et al., 2018)	ImageNet	77.6	0.05 (L_2)	100.0	36.8	100.0
PIXEL DEFLECTION (Prakash et al., 2018)	ImageNet	69.1	0.015 (L_∞)	97.0	–	100.0

Overview

- The objective of adversarial training is a minimax problem:

$$\min_{\theta} E_{(x,y)\sim\mathcal{D}}\left[\max_{||\mathbf{x}'-\mathbf{x}||<\epsilon} L(x', y; \theta)\right]$$

- A very difficult problem. Neither min or max is very hard
- Current Approach is to approximate the max example, and minimize the loss of this example
- Recall the method we have introduce to find the adversarial

Fast Gradient Signed Method(FGSM)

- The neural network is linear in a small region
- Taylor Expansion

$$L(x'; y) = L(x; y) + \nabla_{\mathbf{x}} L(x; y)^T (\mathbf{x}' - \mathbf{x})$$

- The maximum Loss is to set

$$\mathbf{x}_{adv} = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} L(x; y))$$

Ensemble Adversarial Training

- Replace the maximum part by the example found by FGSM

$$\min_{\theta} E_{(x,y)\sim\mathcal{D}}[\mathbf{x}_{FGSM}]$$

- The $\mathbf{x}_{FGSM}$ is a weighted sum of $\mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} L(x, y; \theta))$ and adversarial examples coming from other network

Cascade Adversarial Training

- Training a network with adversarial training.
- Attack the network with some method
- Combine these inputs to do adversarial training

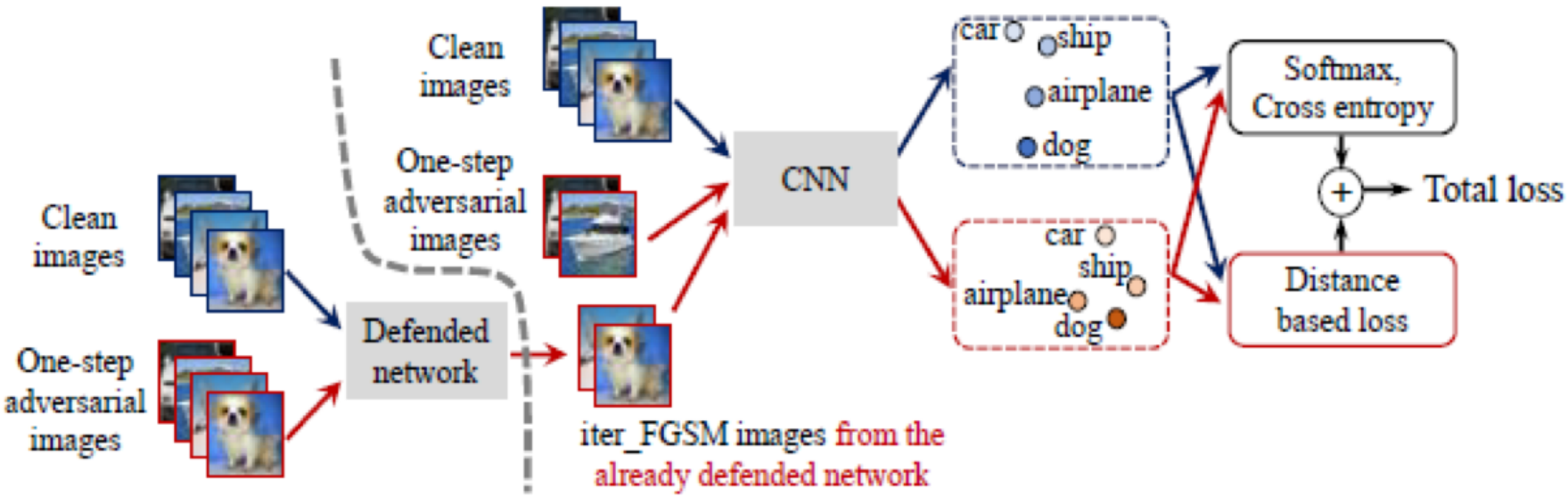


Figure 2: Cascade adversarial training regularized with a unified embedding.

Projected Gradient Descent(PGD)

- $$x_{adv}^{0} = \mathbf{x}$$
$$x_{adv}^{n} = \mathrm{Clip}_{\mathbf{x},\epsilon}(\mathbf{x}_{adv}^{n-1} + \alpha \nabla_{\mathbf{x}} L(x; y))$$

Towards Deep Learning Models Resistant To Adversarial Attacks

- Replace the maximum part by the example found by PGD

$$\min_{\theta} E_{(x,y)\sim\mathcal{D}}[\mathbf{x}_{PGD}]$$

- More robust while more time consuming
- Model complexity strongly affect the performance

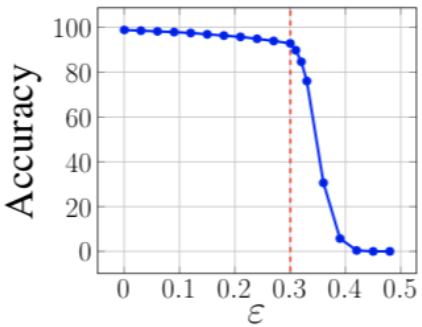
Towards Deep Learning Models Resistant To Adversarial Attacks

Method	Steps	Restarts	Source	Accuracy
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%
FGSM	-	-	A'	96.8%
PGD	40	1	A'	96.0%
PGD	100	20	A'	**95.7%**
CW	40	1	A'	97.0%
CW+	40	1	A'	96.4%
FGSM	-	-	B	**95.4%**
PGD	40	1	B	96.4%
CW+	-	-	B	95.7%

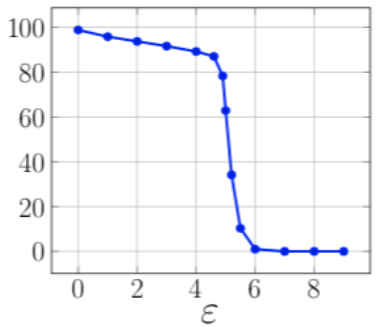
Towards Deep Learning Models Resistant To Adversarial Attacks

Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%
FGSM	-	A'	67.0%
PGD	7	A'	**64.2%**
CW	30	A'	78.7%
FGSM	-	A_{nat}	85.6%
PGD	7	A_{nat}	86.0%

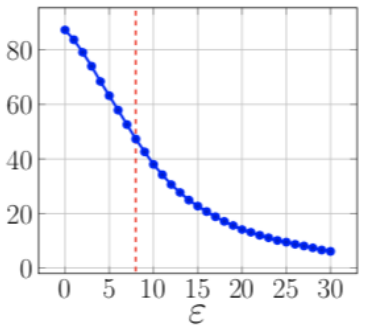
Towards Deep Learning Models Resistant To Adversarial Attacks



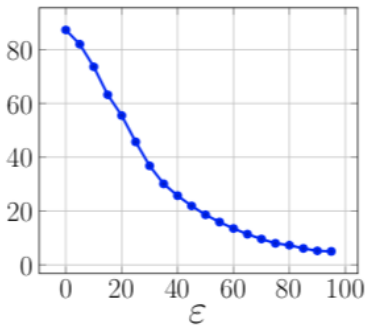
(a) MNIST, ℓ_∞ norm



(b) MNIST, ℓ_2 norm



(c) CIFAR10, ℓ_∞ norm



(d) CIFAR10, ℓ_2 norm

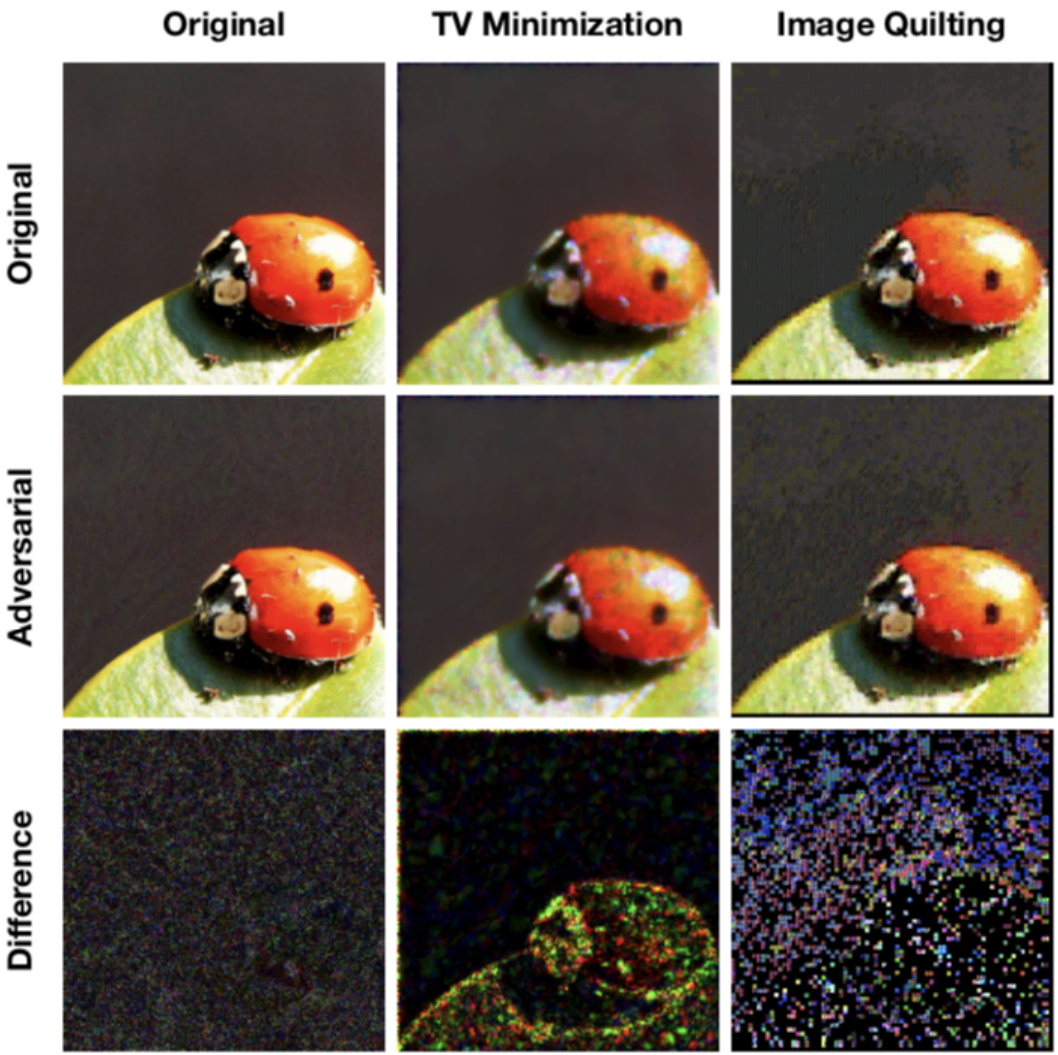
C&W Attack?

A paper: *Certifying Some Distributional Robustness With Principled Adversarial Training* link the C&W Attack with Distributional Robustness under Wasserstein distance.
One finding of the paper is that the penalty term of C&W Attack can make the optimization problem to converge to a station point very quickly if the penalty term is very large.
I am not totally understand the paper...

Input Transformation

- Image cropping and rescaling
- Bit-depth reduction
- JPEG compression
- Total variance minimization: Dropout some pixel and restore them with total variance minimization
- Image quilting: Replacing some patches with clean patches from database. The patches is chosen based on KNN. Minimum graph cut is applied in surrounding region.

Input Transformation



Input Transformation

	Quilting				TVM + Quilting				Cropping + TVM + Quilting			
	RN50	**RN101**	**DN169**	**Iv4**	**RN50**	**RN101**	**DN169**	**Iv4**	**RN50**	**RN101**	**DN169**	**Iv4**
No Attack	70.07	72.56	70.18	73.01	72.38	74.74	73.10	**75.55**	72.14	74.53	72.92	75.10
FGSM	65.45	68.50	65.96	67.53	65.70	68.77	67.09	69.19	66.65	69.75	67.86	**70.37**
I-FGSM	65.59	68.72	66.16	69.29	65.84	69.10	67.32	71.05	67.03	70.14	68.20	**71.52**
DeepFool	65.20	68.73	65.86	68.70	65.80	69.34	67.40	71.03	67.11	70.49	68.62	**71.47**
CW-L2	64.11	67.72	65.00	68.14	63.99	68.20	66.08	70.13	65.31	69.14	66.96	**70.50**

Input Transformation

	Cropping	TVM	Quilting	Ensemble Training (Tramèr et al., 2017)
No Attack	65.41	66.29	69.66	**80.3**
FGSM	49.52	31.37	39.55	**69.15**
I-FGSM	**43.89**	40.99	33.22	5.07
DeepFool	**44.92**	44.69	34.54	1.84
CW-L2	41.06	**48.41**	30.51	22.23

BPDA

- Replace some layer with new approximation: use $f(x)$ as forward pass while the replacing function $g(x)$ in the backward pass
- Run multiple time to get expectation (EOT)
- Reparameter to elinminate vanishing/exploding gradient

BPDA to Input Transformation

- Image cropping and rescaling: EOT
- Bit-depth reduction and JPEG compression: BPDA
- Total variance minimization and Image quilting: EOT and BPDA

Accuracy reduce to 0% while $\epsilon = 0.05(l_2)$

PixelCNN

PixelCNN is a generative model and we can get joint probability of the images. Clean images have higher probability, it can be used to classified adversarials.

PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training is faster than PixelRNN
(can parallelize convolutions since context region values known from training images)

Generation must still proceed sequentially
=> still slow

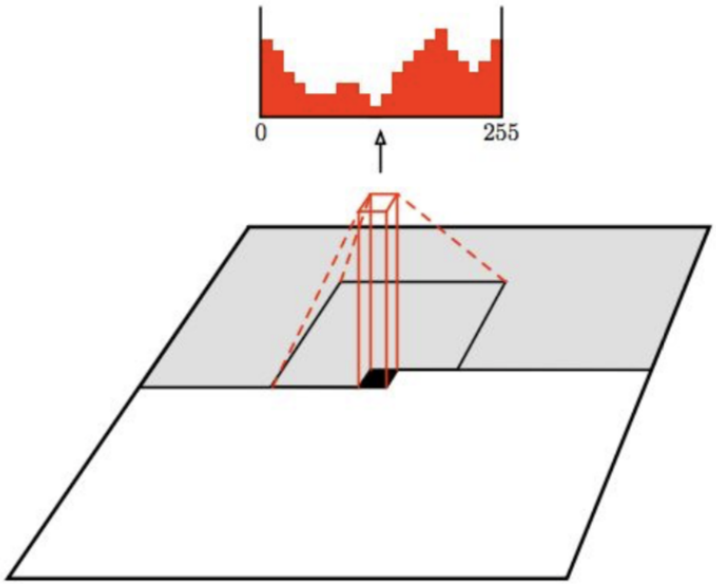
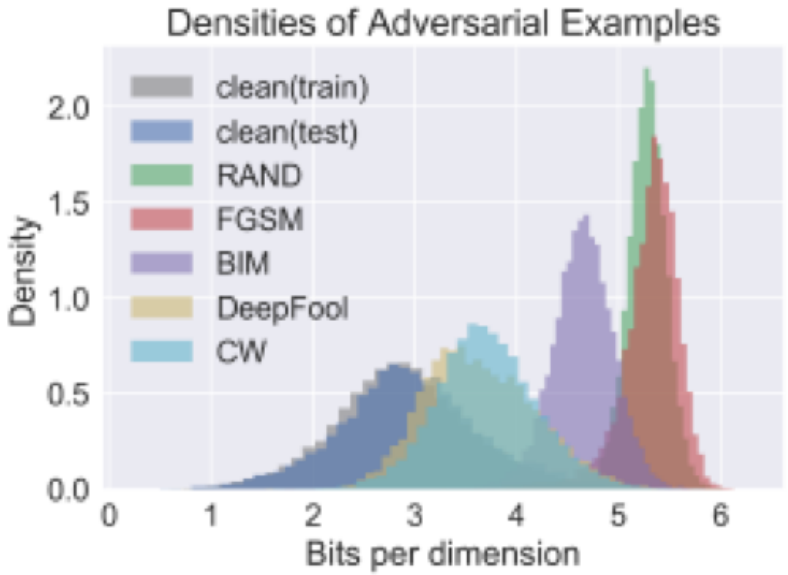


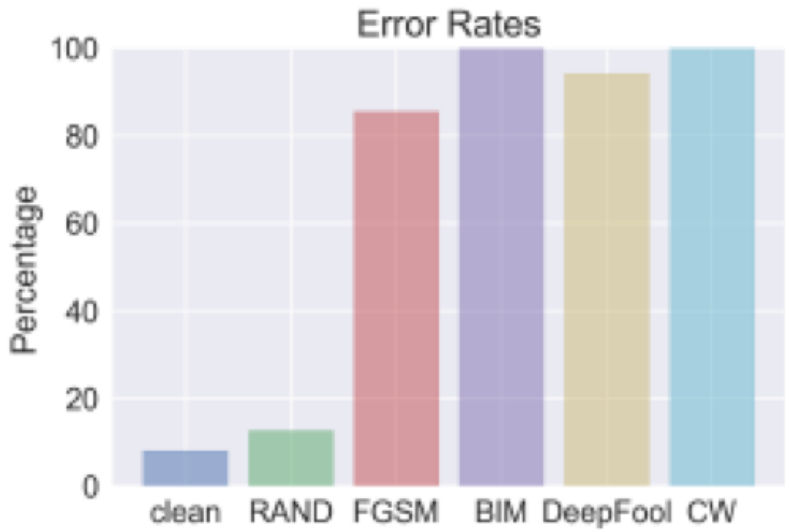
Figure copyright van der Oord et al., 2016. Reproduced with permission.

PixelCNN

PixelCNN is a generative model and we can get joint probability of the images. Clean images have higher probability, it can be used to classified adversarials.



(a)



(b)

PixelCNN

Purifying adversarial samples means maximize the joint probability

Algorithm 1 PixelDefend

Input: Image $\mathbf{X}$, Defense parameter ϵ_{defend}, Pre-trained PixelCNN model p_{CNN}
Output: Purified Image $\mathbf{X}^*$

$\mathbf{X}^* \leftarrow \mathbf{X}$
for each row i **do**
  for each column j **do**
    for each channel k **do**
      $x \leftarrow \mathbf{X}[i, j, k]$
      Set feasible range $R \leftarrow [\max(x - \epsilon_{\text{defend}}, 0), \min(x + \epsilon_{\text{defend}}, 255)]$
      Compute the 256-way softmax $p_{\text{CNN}}(\mathbf{X}^*)$.
      Update $\mathbf{X}^*[i, j, k] \leftarrow \arg\max_{z \in R} p_{\text{CNN}}[i, j, k, z]$
    end for
  end for
end for

PixelCNN

Table 2: **CIFAR-10** ($\epsilon_{\text{attack}} = 2/8/16$, $\epsilon_{\text{defend}} = 16$)

NETWORK	TRAINING TECHNIQUE	CLEAN	RAND	FGSM	BIM	DEEP FOOL	CW	STRONGEST ATTACK
ResNet	Normal	**92/92/92**	**92**/87/76	33/15/11	10/00/00	12/06/06	07/00/00	07/00/00
VGG	Normal	89/89/89	89/88/80	60/46/30	44/02/00	57/25/11	37/00/00	37/00/00
ResNet	Adversarial FGSM	91/91/91	90/**88**/84	**88/91/91**	24/07/00	45/00/00	20/00/07	20/00/00
	Adversarial BIM	87/87/87	87/87/86	80/52/34	74/32/06	79/48/25	76/42/08	74/32/06
	Label Smoothing	**92/92/92**	91/**88**/77	73/54/28	59/08/01	56/20/10	30/02/02	30/02/01
	Feature Squeezing	84/84/84	83/82/76	31/20/18	13/00/00	75/75/75	78/78/78	13/00/00
	Adversarial FGSM + Feature Squeezing	86/86/86	85/84/81	73/67/55	55/02/00	**85/85/85**	83/83/83	55/02/00
ResNet	Normal + ***PixelDefend***	85/85/88	82/83/84	73/46/24	71/46/25	80/80/80	78/78/78	71/46/24
VGG	Normal + ***PixelDefend***	82/82/82	82/82/84	80/62/52	80/61/48	81/76/76	81/79/79	80/61/48
ResNet	Adversarial FGSM + ***PixelDefend***	88/88/86	86/86/**87**	81/68/67	**81**/69/**56**	**85/85/85**	**84/84/84**	**81**/69/**56**
	Adversarial FGSM + ***Adaptive PixelDefend***	90/90/90	86/87/**87**	81/70/67	**81/70/56**	82/81/82	81/80/81	**81/70/56**

- It is almost impossible to compute the gradients through an unrolled version of PixelDefend.
- But it could be bypassed by BPDA : 9% Accuracy with $\epsilon = 0.031$

Defense GAN

- Motivation: Generator G is pre-trained in natural data. It could project adversarial sample into natural manifold.
- Optimization: Gradient Descent with multiple random seed

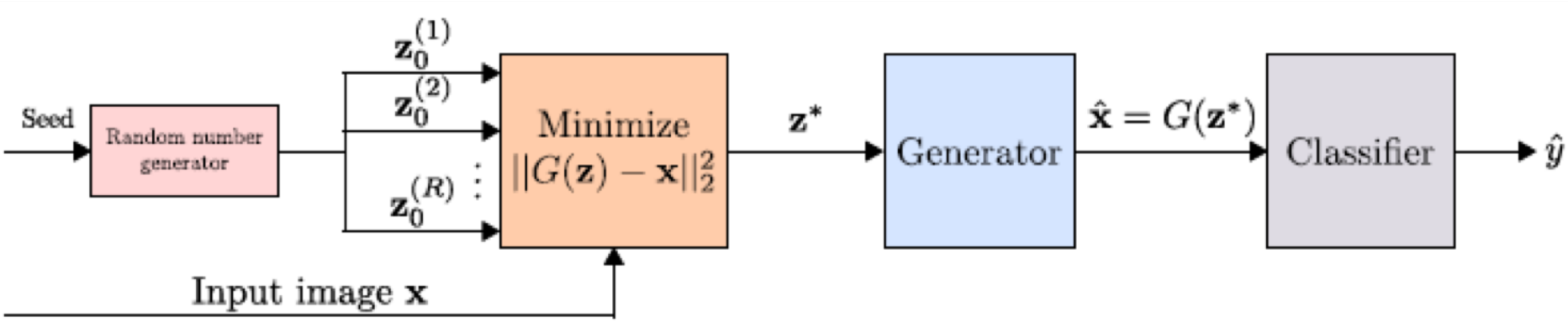


Figure 1: Overview of the Defense-GAN algorithm.

Defense GAN

Attack	Classifier Model	No Attack	No Defense	**Defense-GAN-Rec**	MagNet	Adv. Tr. $\epsilon = 0.3$
FGSM $\epsilon = 0.3$	A	0.997	0.217	**0.988**	0.191	0.651
	B	0.962	0.022	**0.956**	0.082	0.060
	C	0.996	0.331	**0.989**	0.163	0.786
	D	0.992	0.038	**0.980**	0.094	0.732
RAND+FGSM $\epsilon = 0.3, \alpha = 0.05$	A	0.997	0.179	**0.988**	0.171	0.774
	B	0.962	0.017	**0.944**	0.091	0.138
	C	0.996	0.103	**0.985**	0.151	0.907
	D	0.992	0.050	**0.980**	0.115	0.539
CW ℓ_2 norm	A	0.997	0.141	**0.989**	0.038	0.077
	B	0.962	0.032	**0.916**	0.034	0.280
	C	0.996	0.126	**0.989**	0.025	0.031
	D	0.992	0.032	**0.983**	0.021	0.010

- Adversarial samples are on the generator manifold
- It can be attacked be BPDA as well: 45% Accuracy with $\epsilon = 0.05(l_2)$

Concentration

- The author claims that the images are not compact in $[0,1]^n$, there will be an adversarial example within ϵ distance for large proportion of the images.
- ϵ Expansion of $\mathcal{A}$: $\mathcal{A}(\epsilon, d) = \{x | d(x, x_0) < \epsilon,\ x_0 \in \mathcal{A}\}$

Isoperimetric inequality on a unit cube

Let $\Phi(z) = \frac{1}{2\pi}\int_{-\infty}^{z} e^{-t^2/2}dt$ and choose α satisfes $\Phi(\alpha) = \mathrm{vol}[\mathcal{A}]$, Then

$$\mathrm{vol}[\mathcal{A}(\epsilon)] \geq \Phi(\alpha + \frac{\sqrt{2\pi n}}{n^{1/p*}}\epsilon)$$

where $p* = \min(p, 2)$. Further more, if $\mathrm{vol}[\mathcal{A}] \geq 1/2$, then

$$\mathrm{vol}[\mathcal{A}(\epsilon)] \geq 1 - \frac{\exp(-\pi n^{1-2/p*}\epsilon^2)}{2\pi n^{1-2/p*}}$$

Adversarial Examples on a Cube

Adversarial Examples on a Cube

Let f_c be the fraction of hypercube partitioned into class c, and U_c be the supreme of the density. choose some $f_c \leq 1/2$, then with probability at least

$$1 - U_c \frac{\exp(-\pi n^{1-2/p*} \epsilon^2)}{2\pi n^{1-2/p*}}$$

either condition holds

- x is misclassified;
- x has an adversarial examples within ϵ distance.

Condition for Existence of Adversarial Examples

$$\text{vol}[\text{supp}(\rho_c)] \geq \frac{1}{2}\exp(-\pi n^{1-2/p*} \epsilon^2)$$

Discussion

- **Compact distribution of images**: adding small noise for each pixel
- **Adding a don't know class**
- **Feature squeezing**: decreasing the dimension of the image
- **Computational hardness**

The End