University of Idaho
Department of Computer Science

CS 487/587
Adversarial Machine Learning

Dr. Alex Vakanski

Lecture 4

Evasion Attacks against White-box Machine Learning Models

Lecture Outline

- Carlini and Wagner (2017) Towards Evaluating the Robustness of Neural Networks
- Papernot et al. (2016) The limitations of deep learning in adversarial settings
- Xiao et al. (2018) Spatially Transformed Adversarial Examples
- Other white-box evasion attacks
  - Elastic Net (EAD) attack
  - One-pixel attack
  - Universal perturbation attack
  - NewtonFool attack

Evasion Attacks against White-box Models

- So far we covered:
- ***Fast gradient sign method (FGSM) attack***
  - Goodfellow (2015) Explaining and Harnessing Adversarial Examples
  - $x_{adv} = x + \epsilon \cdot \mathrm{sign}\left(\nabla_x \mathcal{L}(h(x, w), y)\right)$
- ***Projected gradient descent (PGD) attack***
  - Madry (2017) Towards Deep Learning Models Resistant to Adversarial Attacks
  - $x_{adv}^{t} = x^{t-1} + \alpha \cdot \mathrm{sign}\left(\nabla_x \mathcal{L}(h(x^{t-1}), y)\right)$
- ***DeepFool attack***
  - Moosavi-Dezfooli (2015) DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks
  - Iteratively projects the perturbed image to the hyperplane of the closest class

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- *Carlini and Wagner attack*
  - Carlini and Wagner (2017) Towards Evaluating the Robustness of Neural Networks
- The paper proposed 3 targeted white-box attacks based on different norm metrics:
  - L_∞ attack
  - L_2 attack
  - L_0 attack
- These attacks are sometimes referred to as C&W attacks or C-W attacks
  - At the time of publishing, they were the strongest adversarial attacks
- Advantages of proposed approaches:
  - Low amount of perturbation
  - Resistance to defense algorithms
  - Generated adversarial images are transferrable across DL models
    - I.e., a secured model is not able to detect the adversarial examples
- Evaluated on: MNIST, CIFAR-10, and ImageNet

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Notation
  - Given an image x, a classifier F outputs a vector y, i.e., $F(x) = y$
    - The paper focuses on NN classifiers
    - The output y is treated as a probability distribution, where y_i is the probability that input x has class i
  - The assigned class by the classifier is

$$C(x) = \text{argmax}_i(y_i) = \text{argmax}_i(F(x)_i)$$

  - The correct label (true class label) of x is denoted by $C^*(x)$
  - The inputs to the softmax function (i.e., the logits) are denoted by z, where the function transforming to input x to the logits is $\mathrm{Z}(x)$, i.e.,

$$F(x) = \text{softmax}(Z(x)) = \text{softmax}(z) = y$$

  - Targeted attack: create an image x' that is similar to x, such that $C(x') = t$, where the target label t is different than the true label $C^*(x)$, i.e., $t \neq C^*(x)$
  - Untargeted attack: create an image x' that is similar to x, such that $C(x') \neq C^*(x)$
    - The paper considers only targeted attacks, as they are more challenging than untargeted attacks

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Initial problem formulation
  - Create an adversarial image x' by adding small perturbation δ to the original image x (i.e., $x' = x + \delta$), such that the distance $\mathcal{D}(x, x') = \mathcal{D}(x, x + \delta)$ is minimal
  - The classifier should assign the class label t to the adversarial image x', where t is different than the true label $C^*(x)$, i.e., $C(x') = C(x + \delta) = t \neq C^*(x)$
  - The goal is to find δ that minimizes $\mathcal{D}(x, x + \delta)$ and $C(x + \delta) = t$

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x + \delta \in [0,1]^n \end{aligned}$$

distance between *x* and *x*+δ

x+δ is classified as target class *t*

each element of *x*+δ is in [0,1] (to be a valid image)

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- This initial formulation of the optimization problem for creating adversarial attacks is difficult to solve
  - Because the constraint $C(x+\delta) = t$ is highly non-linear

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned}$$

- Carlini-Wagner propose the following reformulation of the optimization problem, which is solvable
  - The function f should be chosen such that $C(x+\delta) = t$ if and only if $f(x+\delta) \leq 0$
  - These two optimization problems are not identical: the reformulation by Carlini-Wagner just finds an approximated solution to the above problem
  - Adam optimization algorithm is used for solving the problem

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & f(x+\delta) \leq 0 \\ & x+\delta \in [0,1]^n \end{aligned}$$

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Recall the solution of constrained optimization problems from Lecture 3 using Lagrange multipliers

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) \\ &\text{subject to } c_i(\mathbf{x}) \leq 0 \end{aligned} \quad \Longrightarrow \quad \underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) + \sum_i \alpha_i c_i(\mathbf{x})$$

- The same approach can be applied to the Carlini-Wagner approach, and the optimization problem can be rewritten as shown below
  - The authors performed a grid search for the value of the parameter c (from 0.01 to 100)
  - The recommended approach is to select the value of c where $c > 0$, for which $f(x+\delta) \leq 0$ and the distance $\mathcal{D}(x, x+\delta)$ is minimal

$$\begin{aligned} &\text{minimize} \quad \mathcal{D}(x, x+\delta) \\ &\text{such that} \quad f(x+\delta) \leq 0 \end{aligned} \quad \Longrightarrow \quad \text{minimize} \quad \mathcal{D}(x, x+\delta) + c \cdot f(x+\delta)$$

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- The authors considered several variants for the function f
  - In the equations below, $\text{loss}_{F,t}(x')$ is the loss function with respect to the target class t
  - The class labels are denoted by i
  - Other notation: $(a)^+ = \max(0, a)$; $\text{softplus}(a) = \log(1 + e^a)$
- The best results were obtained by the function $f_6(x')$

$$
\begin{aligned}
f_1(x') &= -\text{loss}_{F,t}(x') + 1 \\
f_2(x') &= (\max_{i \neq t}(F(x')_i) - F(x')_t)^+ \\
f_3(x') &= \text{softplus}(\max_{i \neq t}(F(x')_i) - F(x')_t) - \log(2) \\
f_4(x') &= (0.5 - F(x')_t)^+ \\
f_5(x') &= -\log(2F(x')_t - 2) \\
f_6(x') &= (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+ \\
f_7(x') &= \text{softplus}(\max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2)
\end{aligned}
$$

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Explanation of the function $f_6(x')$
  - In f_6, $Z(x')_t$ is the logits value of the target class t for the perturbed image x'
  - Then, $\max_{i \neq t}(Z(x')_i)$ means the maximum logits values of other class i than the target class t (i.e., $i \neq t$)
  - The function calculates the difference in the logits between the target class t and the closest-to-the-target class
  - In some papers, this function is referred to as margin loss function

$$f_6(x') = (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+$$

- In the paper, a modified function f_6 is also provided
  - It introduces a confidence value k
  - The authors set $k = 0$
    - But, if k has a higher value, this will require that any other logits value exceeds the logits value of the true class $Z(x')_t$ at least by k
    - Examples with large confidence value k have enhanced transferability

$$f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$$

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- ***L_∞ attack***
  - The used distance metric is L_∞ norm, therefore $\mathcal{D}(x, x+\delta) = \|\delta\|_\infty$
  - In other words, $\|\delta\|_\infty$ means the pixel in x' with the largest change from x
- The optimization problem becomes:

$$\text{minimize} \quad \mathcal{D}(x, x+\delta) + c \cdot f(x+\delta) \quad \Rightarrow \quad \text{minimize} \quad \|\delta\|_\infty + c \cdot f(x+\delta)$$

  - However, this formulation produced poor optimization results, since the term $\|\delta\|_\infty$ penalizes only the largest component of the perturbation vector δ
- The authors proposed the following optimization method instead
  - In this case, any component of δ that exceed a threshold value τ is considered, that is, penalize all components of δ that have large values
  - The value of τ is set initially to 1, and is decreased by a factor of 0.9 after each iteration
    - I.e., $\tau \rightarrow \tau \cdot 0.9$ if all $\delta_i < \tau$, else terminate the search

$$\text{minimize} \quad \sum_i [(\delta_i - \tau)^+] + c \cdot f(x+\delta)$$

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- ***Box constraint***
  - In the optimization problem, the constraint $x + \delta \in [0, 1]^n$ requires that in the perturbed images, all pixel values are in the [0,1] range
  - I.e., $0 \leq x_i + \delta_i \leq 1$ for all i
  - This is called a box constraint
    - Or, these values can within the range [0,255] depending on how the images are scaled
- The box constraint can causes difficulties in solving the optimization problem
  - Simply clipping the values can cause that optimization to get stuck in a flat region
- The authors introduced a new variable w, such that

$$x_i + \delta_i = \frac{1}{2}(tanh(w_i) + 1) \quad \Rightarrow \quad \delta_i = \frac{1}{2}(tanh(w_i) + 1) - x_i$$

  - As we know $-1 \leq tanh(w_i) \leq 1$, therefore it follows $0 \leq x_i + \delta_i \leq 1$
  - This change of variables produced more stable optimization results

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- ***L_2 attack***
  - The used distance metric is L_2 norm, therefore $\mathcal{D}(x, x+\delta) = \|\delta\|_2$
- Using the variable w for the box-constraint, the optimization problems becomes

$$\text{minimize} \quad \|\delta\|_2^2 + c \cdot f(x+\delta) \quad \text{where} \quad \delta = \frac{1}{2}(tanh(w)+1) - x$$

$$\text{minimize} \quad \|\frac{1}{2}(\tanh(w)+1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(w)+1)$$

  - That is, search for w that minimizes the above term
- The function f is based on the $f_6(x')$ variant provided earlier

$$f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$$

- To avoid the cases when the gradient descent algorithm become stuck in a local minimum, the authors picked multiple random starting points close to the original image x

Carlini-Wagner Paper (C&W Attack)

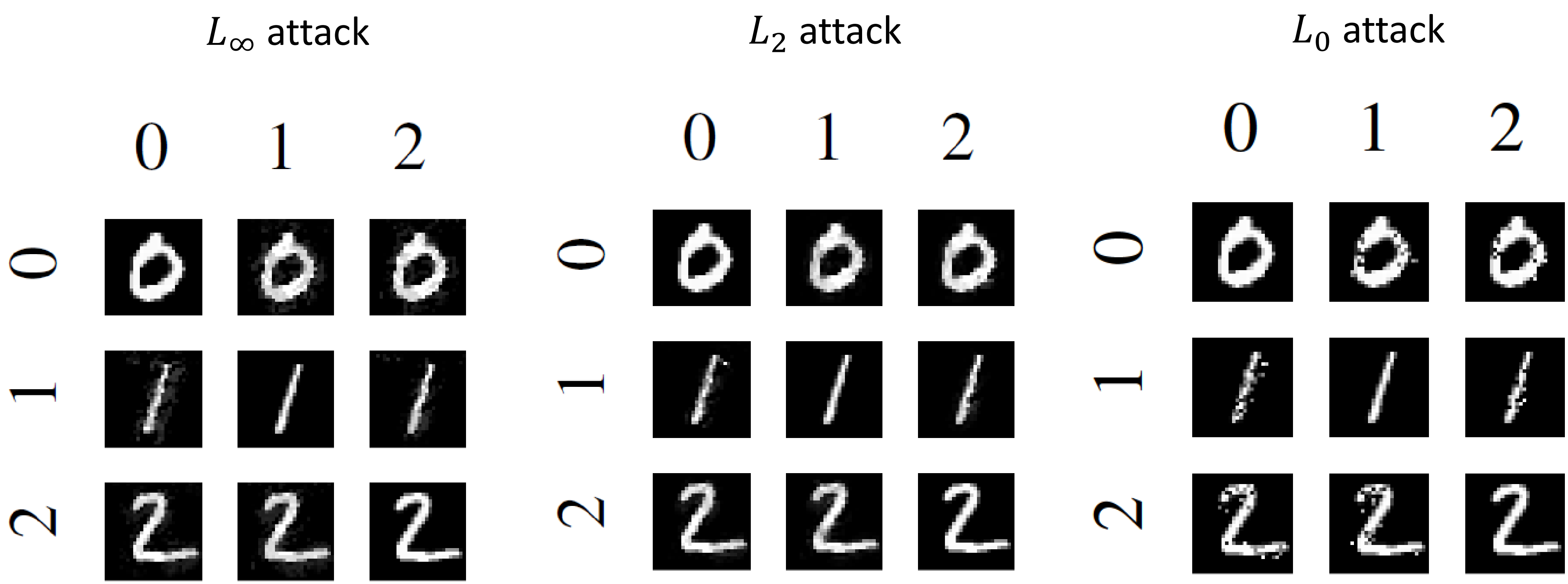
C&W Attack

- ***L_0 attack***
  - The used distance metric is L_0 norm, or, the number of non-zero pixels in δ
- The authors propose an iterative approach
  - Where the goal at each iteration is to find pixels that are not important and don't have much effect on the classifier's output
- The iterative procedure includes the following steps:
  - Initialization: the allowed set includes all pixels in the image
  - Perform L_2 attack to find an adversarial example $x + \delta$
  - Compute the gradient $g = \nabla f(x + \delta)$, where f is the objective function in the L_2 attack
  - Identify the least important pixel $i = \text{argmin}_i\, g_i \delta_i$ and remove this pixel from the allowed set
  - Iterate until the L_2 attack fails to find an adversarial example
- The approach shrinks the set of pixels that are allowed to be changed, until a minimum number of pixels is found that change the class label to the target t

Carlini-Wagner Paper (C&W Attack)

C&W Attack

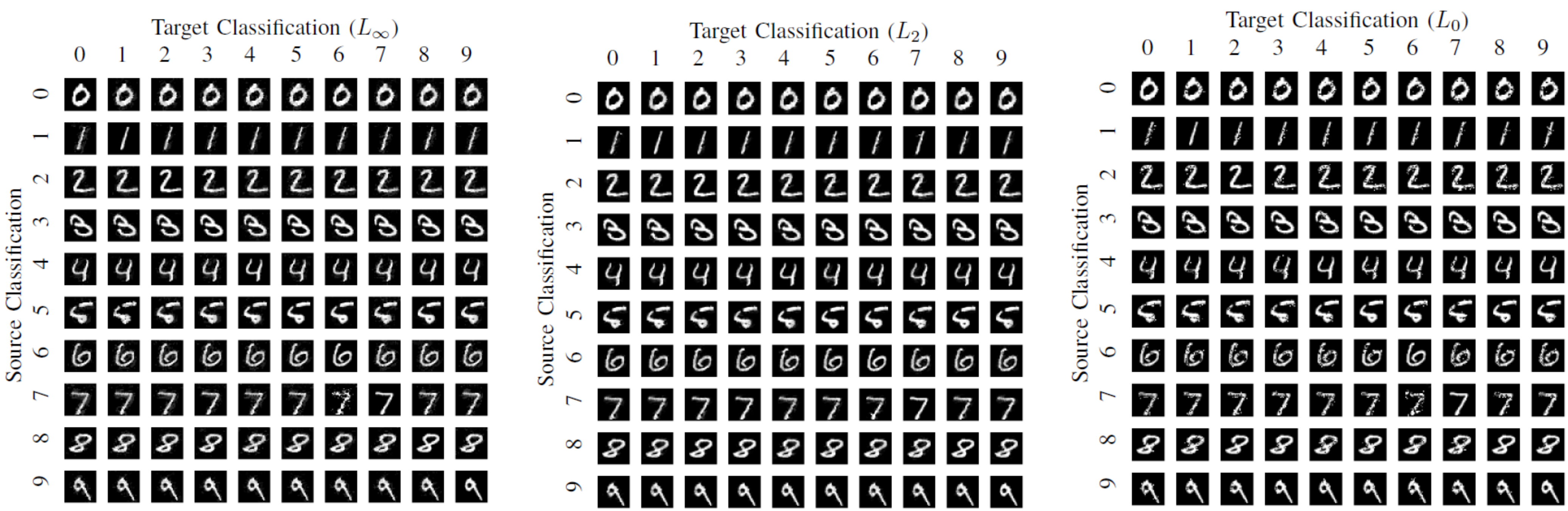
- Results on the MNIST dataset



Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Results on the MNIST dataset for all 10 digits



Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Three approaches for selecting the target class were evaluated:
  - Average Case: select the target class uniformly at random among the labels that are not the correct label
  - Best Case: perform the attack against all incorrect classes, and report the target class that was least difficult to attack
  - Worst Case: perform the attack against all incorrect classes, and report the target class that was most difficult to attack
- The used NN models for MNIST and CIFAR datasets are shown below
  - For ImageNet the paper used the Inception-v3 network

Layer Type	MNIST Model	CIFAR Model
Convolution + ReLU	3×3×32	3×3×64
Convolution + ReLU	3×3×32	3×3×64
Max Pooling	2×2	2×2
Convolution + ReLU	3×3×64	3×3×128
Convolution + ReLU	3×3×64	3×3×128
Max Pooling	2×2	2×2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Softmax	10	10

TABLE I
MODEL ARCHITECTURES FOR THE MNIST AND CIFAR MODELS. THIS ARCHITECTURE IS IDENTICAL TO THAT OF THE ORIGINAL DEFENSIVE DISTILLATION WORK. [39]

Parameter	MNIST Model	CIFAR Model
Learning Rate	0.1	0.01 (decay 0.5)
Momentum	0.9	0.9 (decay 0.5)
Delay Rate	-	10 epochs
Dropout	0.5	0.5
Batch Size	128	128
Epochs	50	50

TABLE II
MODEL PARAMETERS FOR THE MNIST AND CIFAR MODELS. THESE PARAMETERS ARE IDENTICAL TO THAT OF THE ORIGINAL DEFENSIVE DISTILLATION WORK. [39]

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Comparison to JSMA (Jacobian-based Saliency Map Attack), DeepFool, Fast Gradient Sign, and Iterative Gradient Sign attacks on the MNIST and CIFAR datasets
  - Mean is the perturbation size

	Best Case				Average Case				Worst Case			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
Our L_0	8.5	100%	5.9	100%	16	100%	13	100%	33	100%	24	100%
JSMA-Z	20	100%	20	100%	56	100%	58	100%	180	98%	150	100%
JSMA-F	17	100%	25	100%	45	100%	110	100%	100	100%	240	100%
Our L_2	1.36	100%	0.17	100%	1.76	100%	0.33	100%	2.60	100%	0.51	100%
Deepfool	2.11	100%	0.85	100%	—	-	—	-	—	-	—	-
Our L_∞	0.13	100%	0.0092	100%	0.16	100%	0.013	100%	0.23	100%	0.019	100%
Fast Gradient Sign	0.22	100%	0.015	99%	0.26	42%	0.029	51%	—	0%	0.34	1%
Iterative Gradient Sign	0.14	100%	0.0078	100%	0.19	100%	0.014	100%	0.26	100%	0.023	100%

TABLE IV
COMPARISON OF THE THREE VARIANTS OF TARGETED ATTACK TO PREVIOUS WORK FOR OUR MNIST AND CIFAR MODELS. WHEN SUCCESS RATE IS NOT 100%, THE MEAN IS ONLY OVER SUCCESSES.

Carlini-Wagner Paper (C&W Attack)

C&W Attack

- Validation on the ImageNet dataset

	Untargeted		Average Case		Least Likely	
	mean	prob	mean	prob	mean	prob
Our L_0	48	100%	410	100%	5200	100%
JSMA-Z	-	0%	-	0%	-	0%
JSMA-F	-	0%	-	0%	-	0%
Our L_2	0.32	100%	0.96	100%	2.22	100%
Deepfool	0.91	100%	-	-	-	-
Our L_∞	0.004	100%	0.006	100%	0.01	100%
FGS	0.004	100%	0.064	2%	-	0%
IGS	0.004	100%	0.01	99%	0.03	98%

TABLE V
COMPARISON OF THE THREE VARIANTS OF TARGETED ATTACK TO PREVIOUS WORK FOR THE INCEPTION V3 MODEL ON IMAGENET. WHEN SUCCESS RATE IS NOT 100%, THE MEAN IS ONLY OVER SUCCESSES.

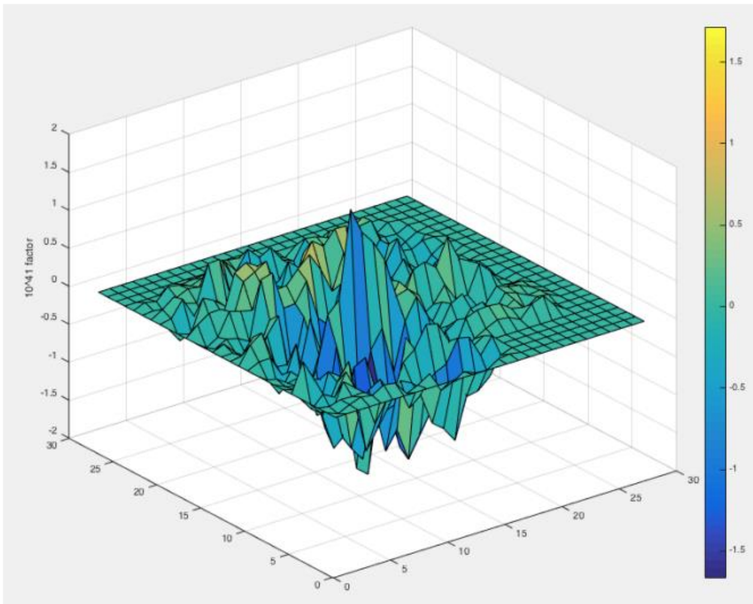
Papernot Paper (JSMA Attack)

JSMA Attack

- ***Jacobian-based Saliency Map Attack (JSMA)***
  - [Papernot et al. (2016) The limitations of deep learning in adversarial settings]
- Targeted white-box attack based on controlling the L_0 norm
  - The goal is to iteratively change each pixel until misclassification
  - The key step is calculation of a saliency map that determines which pixels to be modified, in order to increase the probability of the target class

Compute $\nabla \mathbf{F}(\mathbf{X})$ →

Jacobian matrix



→ Modify input $\mathbf{X}$

Pixels with large saliency values have large impact on the output when perturbed

Create a Saliency Map

JSMA Attack Approach

JSMA Attack

- Notation:
  - $\mathbf{X}$ – clean (benign) input sample
  - $\mathbf{Y} = \mathbf{F}(\mathbf{X})$ – output of a classification model (e.g., NN) given with a function $\mathbf{F}$
  - $\mathbf{X}^*$ – adversarial sample, obtained by manipulating the clean sample $\mathbf{X}$
  - $\mathbf{Y}^*$ – target output (class label) for the adversarial sample, i.e.,$\mathbf{Y}^* = \mathbf{F}(\mathbf{X}^*)$
  - θ – amount of perturbation that is applied to input features (i.e., pixels in images)
  - Υ – maximum distortion that is applied to the input (e.g., number of pixel in the input image that the adversary is allowed to change)
- JSMA attack steps:
  - Step 1: compute the forward derivative $\nabla\mathbf{F}(\mathbf{X}^*)$ of the NN
  - Step 2: construct an adversarial saliency map S based on the forward derivative $\nabla\mathbf{F}(\mathbf{X}^*)$
  - Step 3: modify the most impactful input features i_{max} by θ

JSMA Attack Approach

JSMA Attack

- Step 1: compute the forward derivative $\nabla\mathbf{F}(\mathbf{X}^*)$ of the NN
  - For input vectors to the NN of size M, and outputs of the NN of size N (i.e., N class classification), the function of the NN is the mapping $\mathbf{F}: \mathbb{R}^M \rightarrow \mathbb{R}^N$
  - Therefore, the forward gradient is the Jacobian matrix of the function $\mathbf{F}$, given with the first-order partial derivatives of the outputs $\mathbf{F}(\mathbf{X}^*)$ with respect to the inputs $\mathbf{X}^*$, i.e.,

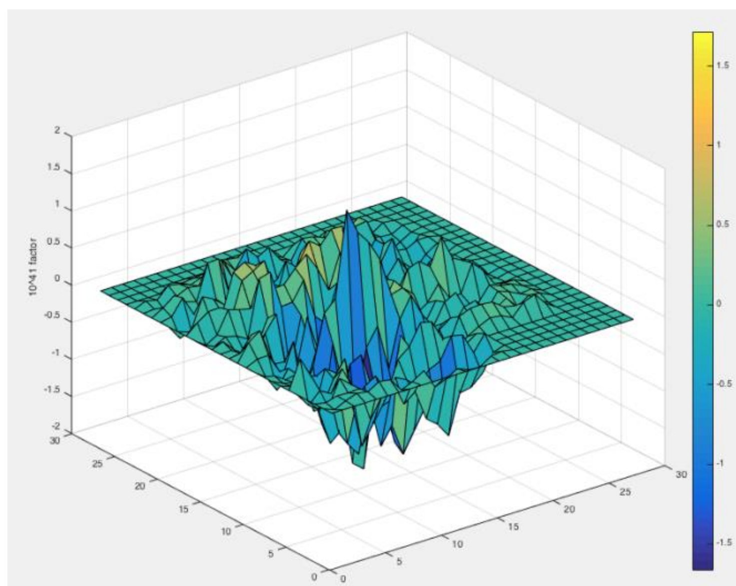
$$\nabla\mathbf{F}(\mathbf{X}^*) = \begin{bmatrix} \frac{\partial \mathbf{F}_1(\mathbf{X}^*)}{\partial x_1} & \cdots & \frac{\partial \mathbf{F}_1(\mathbf{X}^*)}{\partial x_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{F}_N(\mathbf{X}^*)}{\partial x_1} & \cdots & \frac{\partial \mathbf{F}_N(\mathbf{X}^*)}{\partial x_M} \end{bmatrix}$$

  - Each element in the Jacobian matrix (i.e., the forward derivative) $\nabla\mathbf{F}$ of an NN given with function $\mathbf{F}$ can be computed for any input $\mathbf{X}$ by successively differentiating layers, starting from the input layer until the output layer is reached

JSMA Attack Approach

JSMA Attack

- Step 2: construct an adversarial saliency maps S based on the forward derivative $\nabla\mathbf{F}(\mathbf{X}^*)$
  - Saliency maps are employed in Explainable Machine Learning, to indicate which pixels in an image contributed the most to the predicted class by a NN
  - Adversarial saliency maps can be used to indicate which pixels in an image an adversary should perturb in order to impact the predicted class by a NN
- An example of a saliency map for a 28×28 pixels image is shown below
  - The pixels with large peaks or valleys have significant impact on the predicted class



JSMA Attack Approach

- Step 3: modify the most impactful input pixels by θ
  - Once the most impactful input pixels in the saliency map have been identified, they are perturbed by θ in order to realize the adversary's goal
    - E.g., θ are discrete steps applied to change the pixel intensities
  - The algorithm perturbs 2 most impactful pixels at each step
- Afterward, all steps are repeated until:
  - The adversarial sample $\mathbf{X}^*$ is classified with the target class $\mathbf{Y}^*$, or
  - The maximum number of iterations is reached, or
  - The maximum number of pixels Υ are perturbed

JSMA Attack Approach

- The entire JSMA algorithm:

Algorithm 2 **Crafting adversarial samples for LeNet-5**

$\mathbf{X}$ is the benign image, $\mathbf{Y}^*$ is the target network output, $\mathbf{F}$ is the function learned by the network during training, Υ is the maximum distortion, and θ is the change made to pixels.

Input: $\mathbf{X}$, $\mathbf{Y}^*$, $\mathbf{F}$, Υ, θ

$\mathbf{X}^* \leftarrow \mathbf{X}$

$\Gamma = \{1 \ldots |\mathbf{X}|\}$ ▷ search domain is all pixels

$\texttt{max\_iter} = \left\lfloor \frac{784 \cdot \Upsilon}{2 \cdot 100} \right\rfloor$

$s = \arg\max_j \mathbf{F}(\mathbf{X}^*)_j$ ▷ source class

$t = \arg\max_j \mathbf{Y}^*_j$ ▷ target class

while $s \neq t$ & $\texttt{iter} < \texttt{max\_iter}$ & $\Gamma \neq \emptyset$ **do**

  Compute forward derivative $\nabla \mathbf{F}(\mathbf{X}^*)$

  $p_1, p_2 = \texttt{saliency\_map}(\nabla \mathbf{F}(\mathbf{X}^*), \Gamma, \mathbf{Y}^*)$

  Modify p_1 and p_2 in $\mathbf{X}^*$ by θ

  Remove p_1 from Γ if $p_1 == 0$ or $p_1 == 1$

  Remove p_2 from Γ if $p_2 == 0$ or $p_2 == 1$

  $s = \arg\max_j \mathbf{F}(\mathbf{X}^*)_j$

  $\texttt{iter}++$

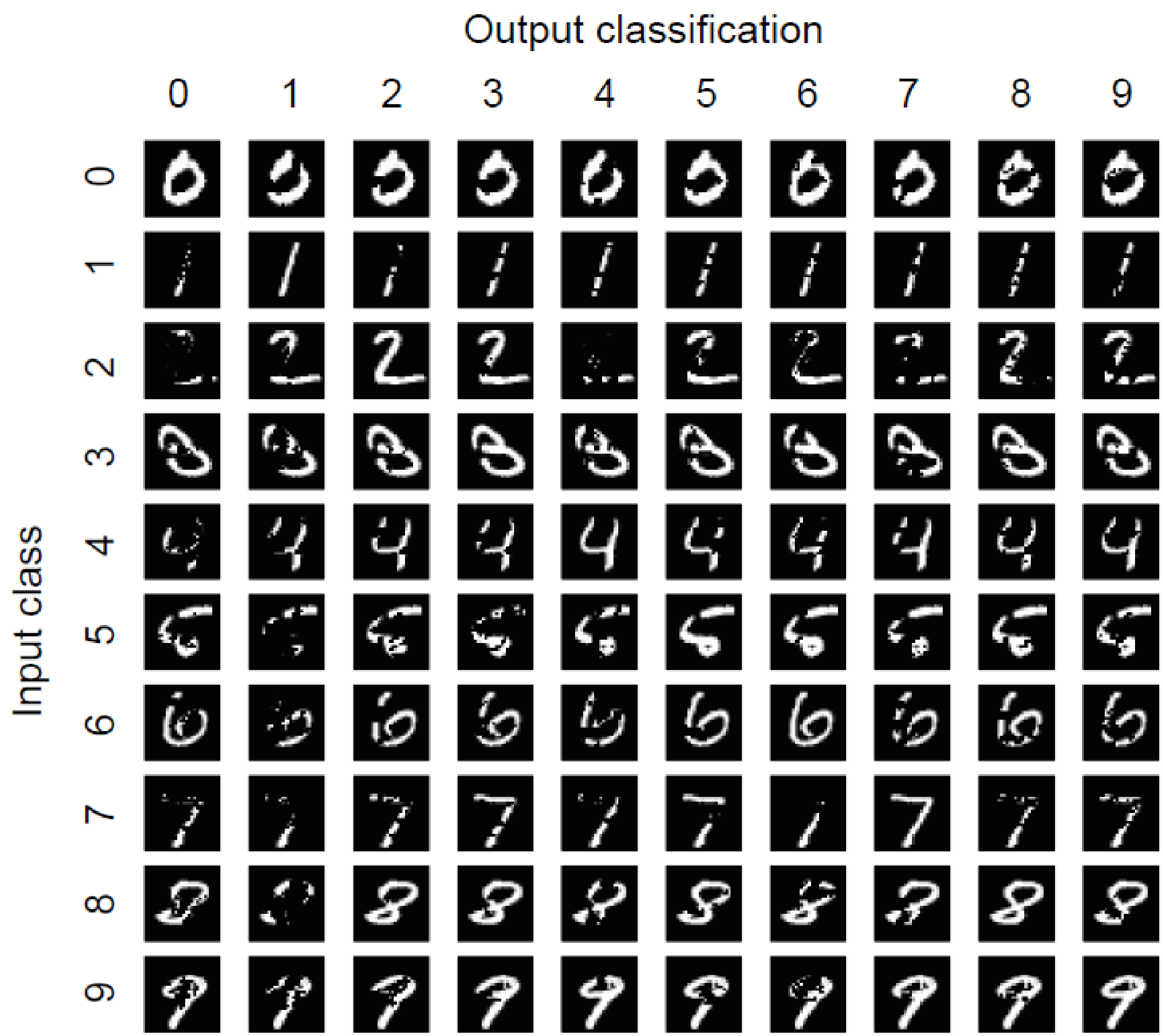
end while

return $\mathbf{X}^*$

JSMA Attack Results

JSMA Attack

- Samples of attacked MNIST images with JSMA



JSMA Attack Results

- JSMA attack was validated on the MNIST dataset using the LeNet deep model
  - The attack achieved a success rate of 97.05% while perturbing on average 4.03% of the pixels in images

Source set of 10,000 original samples	Adversarial samples successfully misclassified	Average distortion	
		All adversarial samples	Successful adversarial samples
Training	97.05%	4.45%	4.03%
Validation	97.19%	4.41%	4.01%
Test	97.05%	4.45%	4.03%

JSMA Attack

- Difference to other approaches:
  - JSMA calculates a mapping between the perturbations of input pixels and the predicted output of the model
    - JMSA uses the forward propagated derivatives of the model function with respect to the input pixels
    - The forward propagated derivatives form the Jacobian matrix $\nabla\mathbf{F}(\mathbf{X})$
  - FGSM, PGD work by calculating the mapping between the predicted output by the model and the inputs
    - These models use the backward propagated derivatives of the loss function with respect to the input pixels $\nabla\boldsymbol{\mathcal{L}}(\mathbf{X})$

Xiao, Li, Song Paper (stAdv Attack)

stAdv Attack

- ***stAdv attack***
  - Xiao, Zhu, Li, He, Liu, Song (2018) Spatially Transformed Adversarial Examples
- The paper proposes an attack that does not manipulate the pixel intensity values under an L_p norm
- Instead, the pixels are spatially moved in an image to create an adversarial example
  - Such attack can result in a large L_p distance between the original and manipulated images
  - Still, the images are perceptually realistic
  - The perturbed images are effective against defense algorithms
- The approach minimizes the local geometric distortion of images
- Validation: MNIST, CIFAR-10, and ImageNet datasets

Spatial Transformation Attack

stAdv Attack

- Example of a spatially transformed image
  - The red flow arrows indicate the local displacement of the pixels in adversarial image to the pixels in the input image

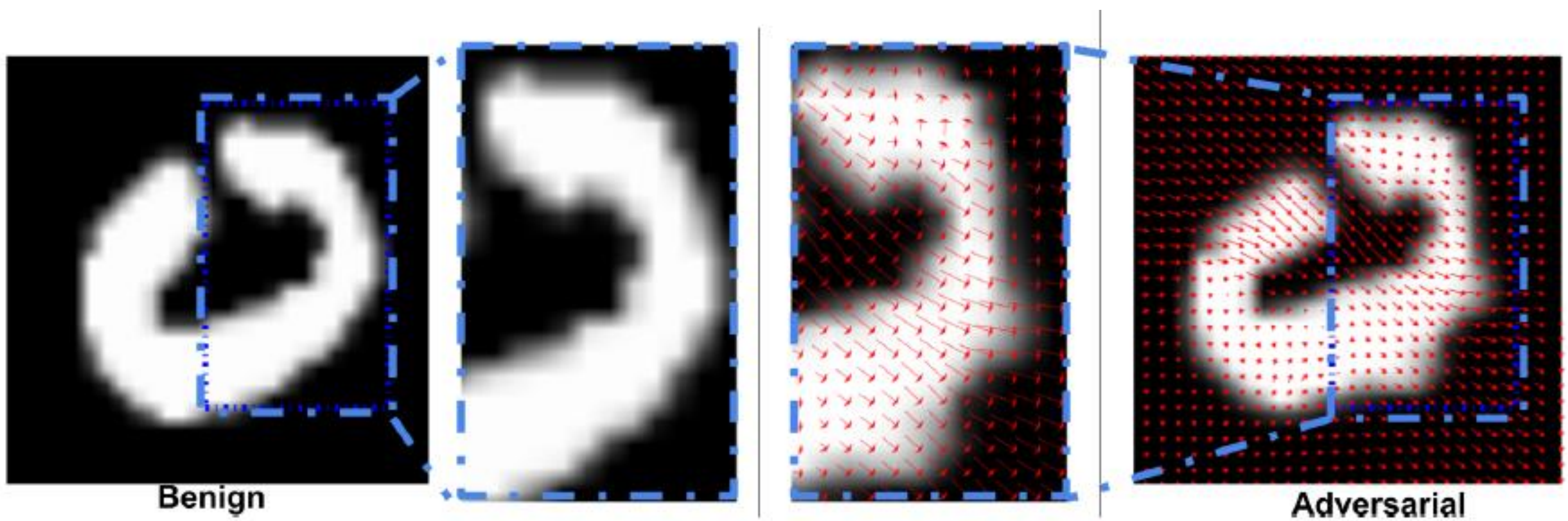
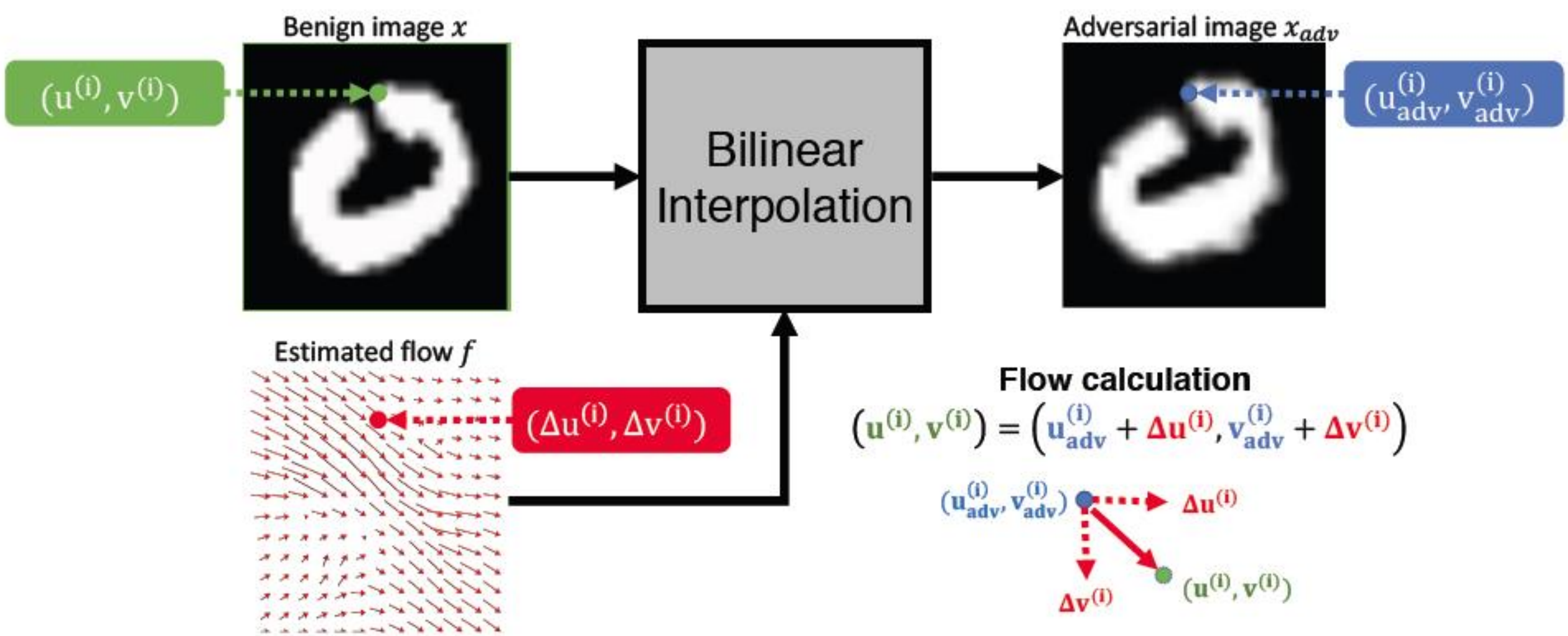


Figure 5: Flow visualization on MNIST. The digit “0" is misclassified as “2".

Spatial Transformation Attack

stAdv Attack

- Green color – the pixel i in the input (benign, clean) image
- Blue color – the spatially displaced pixel i in the adversarial image
- Red arrows – the displacement flow f: horizontal ($\Delta u^{(i)}$) and vertical ($\Delta v^{(i)}$)
  - Goal: find an adversarial image with lowest overall displacement $f_i = \left(\Delta u^{(i)}, \Delta v^{(i)}\right)$



Spatial Transformation Attack

stAdv Attack

- A targeted white-box attack is considered
- The problem is formulated as an optimization problem, that is very similar to the Carlini-Wagner paper
- For an image x, find the minimum local distortion f^*, such that

$$f^* = \underset{f}{\operatorname{argmin}} \quad \mathcal{L}_{adv}(x, f) + \tau \mathcal{L}_{flow}(f)$$

  - The term $\mathcal{L}_{adv}$ encourages the distorted image to be misclassified as the target class t
  - The term $\mathcal{L}_{flow}$ ensure that the spatial transformation is preserved
  - τ is a constant that balances the two terms (set to 0.05 for validation)
- The authors adopted the $f_6(x')$ function from Carlini-Wagner for the term $\mathcal{L}_{adv}$
  - That maximizes the logits values of the target class t with respect to other classes

$$\mathcal{L}_{adv}(x, f) = \max(\max_{i \neq t} g(\mathrm{x}_{\mathrm{adv}})_i - g(\mathrm{x}_{\mathrm{adv}})_t, \kappa)$$

Spatial Transformation Attack

stAdv Attack

- The term $\mathcal{L}_{flow}$ is calculated as the sum of spatial movement distance for any two adjacent pixels p and q
  - This makes the stAdv approach computationally expensive, because it require calculating the distances for all pairs of neighboring pixels

$$\mathcal{L}_{flow}(f) = \sum_{p}^{\text{all pixels}} \sum_{q \in \mathcal{N}(p)} \sqrt{||\Delta u^{(p)} - \Delta u^{(q)}||_2^2 + ||\Delta v^{(p)} - \Delta v^{(q)}||_2^2}.$$

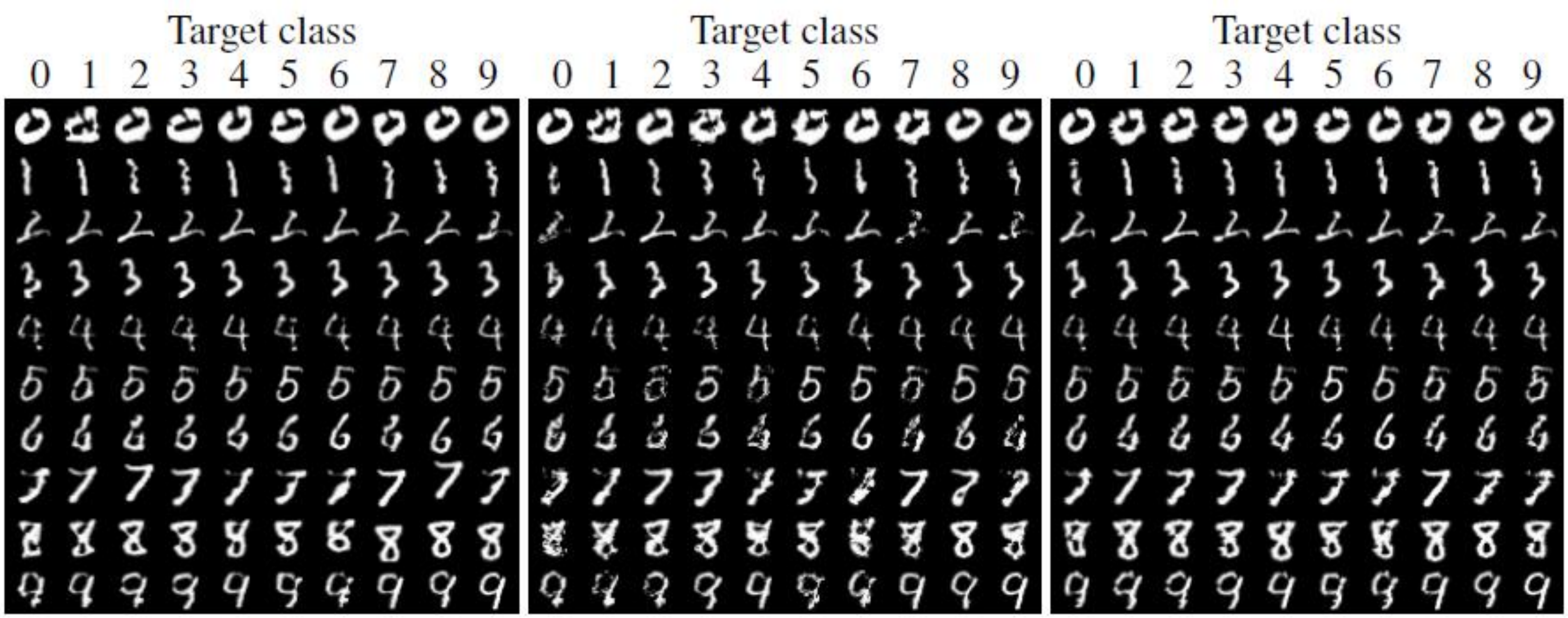
- The optimization problem is solved using the L-BFGS algorithm (Limited-memory BFGS (Broyden–Fletcher–Goldfarb–Shanno))

Spatial Transformation Attack

stAdv Attack

- Validation on MNIST for three different NN model architectures A, B, and C
  - Accuracy (p) means the model classification accuracy on pristine (original) images

Model	A	B	C
Accuracy (p)	98.58%	98.94%	99.11%
Attack Success Rate	99.95%	99.98%	100.00%



(a) Model A (b) Model B (c) Model C

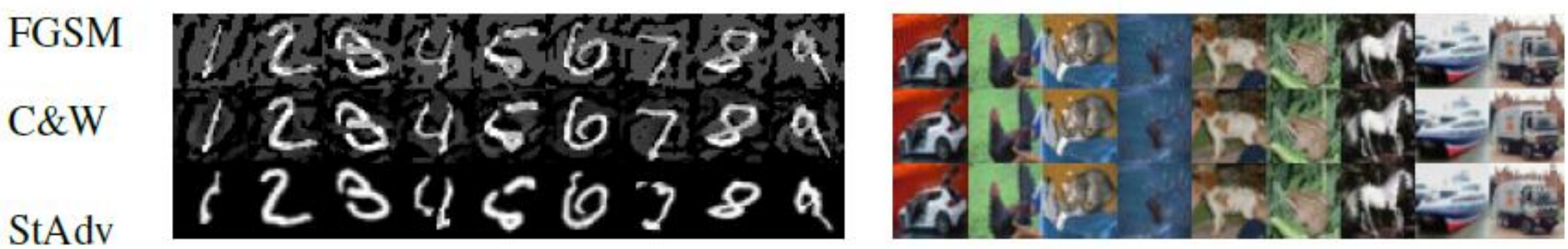
Spatial Transformation Attack

stAdv Attack

- For CIFAR-10 images, they used ResNet32 and Wide ResNet34

Model	ResNet32 (0.47M)	Wide ResNet34 (46.16M)
Accuracy (p)	93.16%	95.82%
Attack Success Rate	99.56%	98.84%

- Comparison of adversarial examples generated by FGSM, C&W, and stAdv
  - Left: MNIST, right: CIFAR-10
  - The generated images by stAdv attack have high perceptual quality



Spatial Transformation Attack

stAdv Attack

- Flow visualization on ImageNet
  - (a): the original image, (b)-(c): images are misclassified into goldfish, dog and cat
  - Although there are other objects within the image (e.g., trees), most spatial transformation flows focus on the target object – mountain bike

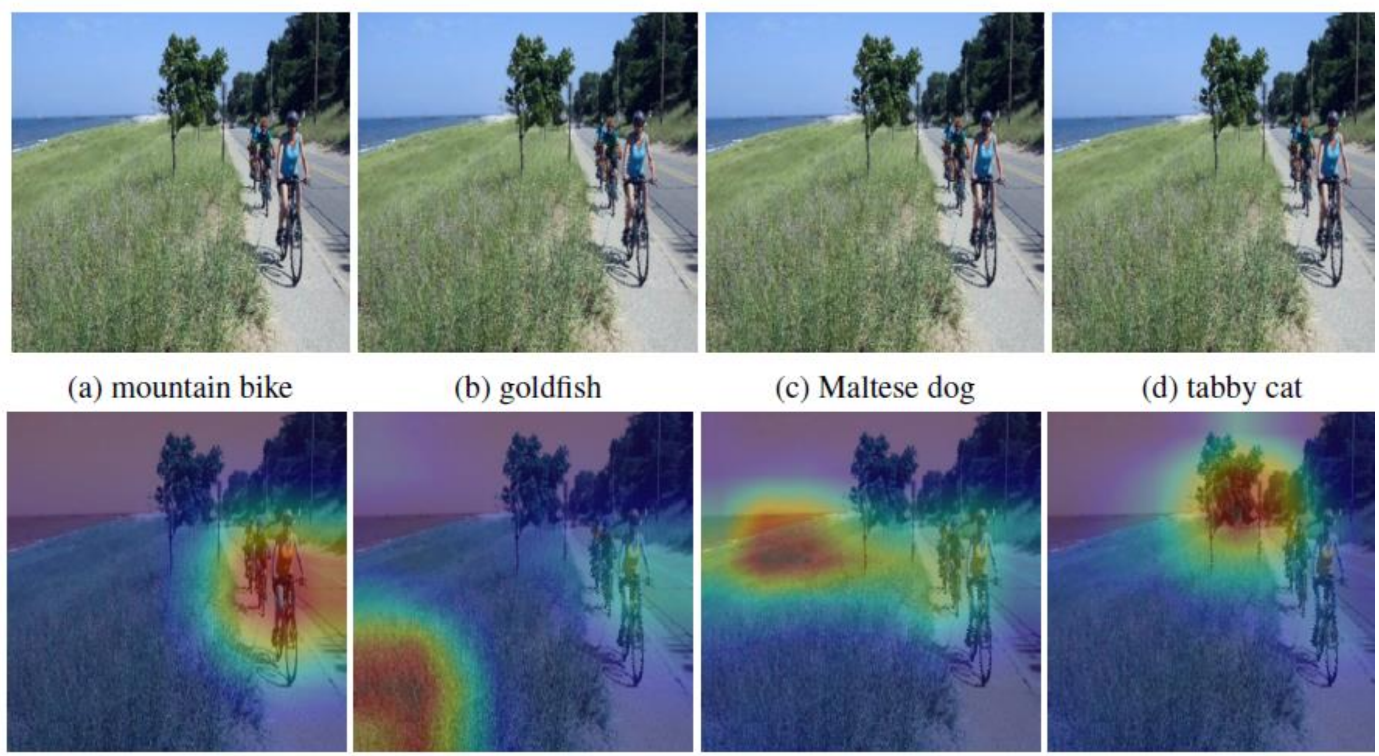
(a) mountain bike (b) goldfish (c) Maltese dog (d) tabby cat

- Human participants on Amazon Mechanical Turk (AMT) were recruited to analyze the visual perceptibility of attacked images
  - The users selected the attacked images as visually realistic

Spatial Transformation Attack

stAdv Attack

- Further analysis includes visualizing the salliency maps of images
  - I.e., find the regions in the images where the model pays the most attention for assigning a particular class to an images
  - Class Activation Mapping (CAM) was used for this purpose
  - stAdv attack misleads the model to pay attention to different regions than the bike



(a) mountain bike (b) goldfish (c) Maltese dog (d) tabby cat

(e) (f) (g) (h)

Spatial Transformation Attack

stAdv Attack

- Attack evaluation under three defense methods: FGSM adversarial training (Adv.), ensemble adversarial training (Ens.), and PGD adversarial training (PGD)

Table 3: Attack success rate of adversarial examples generated by stAdv against models A, B, and C under standard defenses on MNIST, and against ResNet and wide ResNet on CIFAR-10.

Model	Def.	FGSM	C&W.	stAdv
A	Adv.	4.3%	4.6%	**32.62%**
	Ens.	1.6%	4.2%	**48.07%**
	PGD	4.4%	2.96%	**48.38%**
B	Adv.	6.0%	4.5%	**50.17%**
	Ens.	2.7%	3.18%	**46.14%**
	PGD	9.0%	3.0%	**49.82%**
C	Adv.	3.22%	0.86%	**30.44%**
	Ens.	1.45%	0.98%	**28.82%**
	PGD	2.1%	0.98%	**28.13%**

Model	Def.	FGSM	C&W.	stAdv
ResNet32	Adv.	13.10%	11.9%	**43.36%**
	Ens.	10.00%	10.3%	**36.89%**
	PGD	22.8%	21.4%	**49.19%**
wide ResNet34	Adv.	5.04%	7.61%	**31.66%**
	Ens.	4.65%	8.43%	**29.56%**
	PGD	14.9%	13.90%	**31.6%**

Elastic-Net Attack

Other White-box Evasion Attacks

- ***Elastic Net (EAD) Attack***
  - [Chen et al. (2017) EAD: Elastic-net attacks to deep neural networks via adversarial examples]
- Modification of the C&W attack for controlling the L_1 norm of adversarial perturbations
  - Recall than C&W proposed 3 attacks for controlling the L_0, L_2, and L_∞ norms
- EAD attack produced visually plausible adversarial samples

Elastic-Net Attack

Other White-box Evasion Attacks

- EAD is based on elastic-net regularization (recall from Lecture 2, it uses both ℓ_1 and ℓ_2 penalties on the model parameters)
- The solved optimization problem is (compare to C&W):

$$\begin{aligned} &\text{minimize}_{\mathbf{x}} \;\; c \cdot f(\mathbf{x}, t) + \beta \|\mathbf{x} - \mathbf{x}_0\|_1 + \|\mathbf{x} - \mathbf{x}_0\|_2^2 \\ &\text{subject to} \;\; \mathbf{x} \in [0,1]^p, \end{aligned}$$

- EAD employs a box constraint based on Iterative Shrinkage-Thresholding Algorithm (ISTA)

Algorithm 1 Elastic-Net Attacks to DNNs (EAD)

Input: original labeled image $(\mathbf{x}_0, t_0)$, target attack class t, attack transferability parameter κ, L_1 regularization parameter β, step size α_k, # of iterations I
Output: adversarial example $\mathbf{x}$
Initialization: $\mathbf{x}^{(0)} = \mathbf{y}^{(0)} = \mathbf{x}_0$
for $k = 0$ to $I - 1$ **do**
  $\mathbf{x}^{(k+1)} = S_\beta(\mathbf{y}^{(k)} - \alpha_k \nabla g(\mathbf{y}^{(k)}))$
  $\mathbf{y}^{(k+1)} = \mathbf{x}^{(k+1)} + \frac{k}{k+3}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$
end for
Decision rule: determine $\mathbf{x}$ from successful examples in $\{\mathbf{x}^{(k)}\}_{k=1}^{I}$ (EN rule or L_1 rule).

$S_\beta : \mathbb{R}^p \mapsto \mathbb{R}^p$ is an element-wise projected shrinkage-thresholding function, which is defined as

$$[S_\beta(\mathbf{z})]_i = \begin{cases} \min\{\mathbf{z}_i - \beta, 1\}, & \text{if } \mathbf{z}_i - \mathbf{x}_{0i} > \beta; \\ \mathbf{x}_{0i}, & \text{if } |\mathbf{z}_i - \mathbf{x}_{0i}| \le \beta; \\ \max\{\mathbf{z}_i + \beta, 0\}, & \text{if } \mathbf{z}_i - \mathbf{x}_{0i} < -\beta, \end{cases}$$

Elastic-Net Attack

Other White-box Evasion Attacks

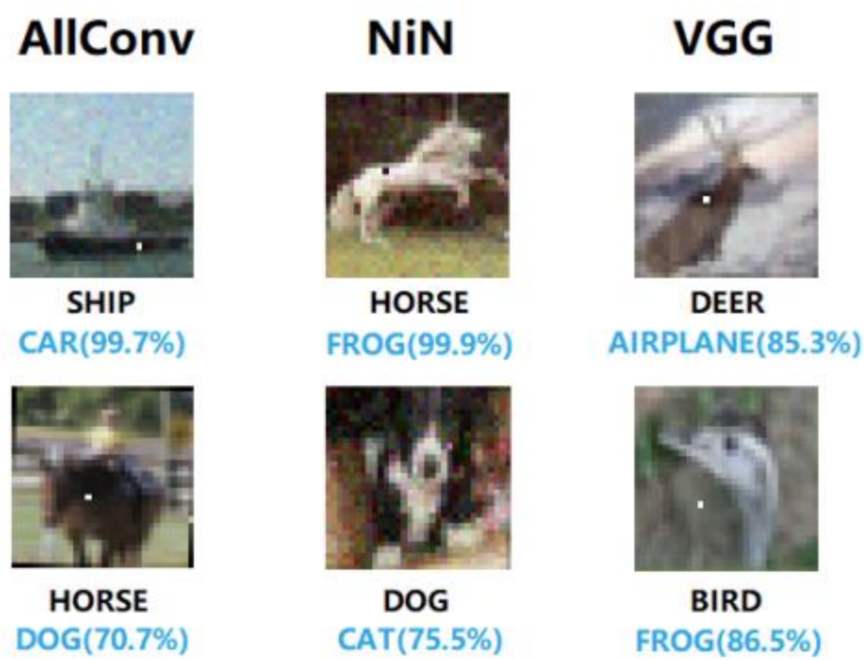
- Elastic-Net attack produces low perturbations in experimental validation on MNIST, CIFAR-10, and ImageNet
  - ASR is Attack Success Rate
  - Compared are two EAD approaches: EN rule (uses elastic net regularization) and L_1 rule (uses only L_1 regularization)
  - EAD achieved the lowest L_1 perturbation

	MNIST				CIFAR10				ImageNet			
Attack method	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞
C&W (L_2)	**100**	22.46	**1.972**	0.514	**100**	13.62	**0.392**	0.044	**100**	232.2	**0.705**	0.03
FGM-L_1	39	53.5	4.186	0.782	48.8	51.97	1.48	0.152	1	61	0.187	0.007
FGM-L_2	34.6	39.15	3.284	0.747	42.8	39.5	1.157	0.136	1	2338	6.823	0.25
FGM-L_∞	42.5	127.2	6.09	0.296	52.3	127.81	2.373	0.047	3	3655	7.102	0.014
I-FGM-L_1	**100**	32.94	2.606	0.591	**100**	17.53	0.502	0.055	77	526.4	1.609	0.054
I-FGM-L_2	**100**	30.32	2.41	0.561	**100**	17.12	0.489	0.054	**100**	774.1	2.358	0.086
I-FGM-L_∞	**100**	71.39	3.472	**0.227**	**100**	33.3	0.68	**0.018**	**100**	864.2	2.079	**0.01**
EAD (EN rule)	**100**	**17.4**	2.001	0.594	**100**	**8.18**	0.502	0.097	**100**	**69.47**	1.563	0.238
EAD (L_1 rule)	**100**	**14.11**	2.211	0.768	**100**	**6.066**	0.613	0.17	**100**	**40.9**	1.598	0.293

One-Pixel Attack

Other White-box Evasion Attacks

- ***One-pixel Attack***
  - [Su et al. (2019) One pixel attack for fooling deep neural networks]
- Attack under the L_0 norm to limit the number of pixels allowed to be changed
  - One-pixel attack employs Differential Evolution-based optimization for creating adversarial examples
- It shows that on CIFAR-10 dataset, most samples can be attacked in an untargeted manner by changing the value of only one pixel



One-Pixel Attack

Other White-box Evasion Attacks

- One-pixel attack on ImageNet
  - The modified pixels are highlighted with red circles

Cup(16.48%)
Soup Bowl(16.74%)

Bassinet(16.59%)
Paper Towel(16.21%)

Teapot(24.99%)
Joystick(37.39%)

Hamster(35.79%)
Nipple(42.36%)

One-Pixel Attack

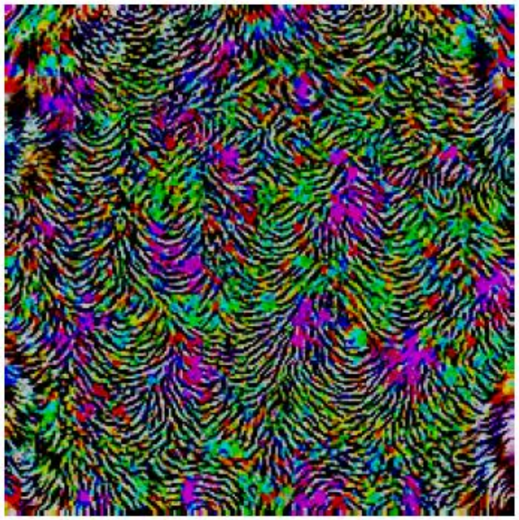
Other White-box Evasion Attacks

- Validation results on CIFAR-10 dataset using 4 type of DL models: AllConv (all convolutional network), NIN (network in network), VGG16, and BVLS AlexNet
  - OriginalAcc is accuracy on clean images
  - Targeted and Non-targeted is the accuracy for adversarial samples with target class and random class, respectively

	AllConv	NiN	VGG16	BVLC
OriginAcc	85.6%	87.2%	83.3%	57.3%
Targeted	19.82%	23.15%	16.48%	–
Non-targeted	68.71%	71.66%	63.53%	16.04%

Universal Attack

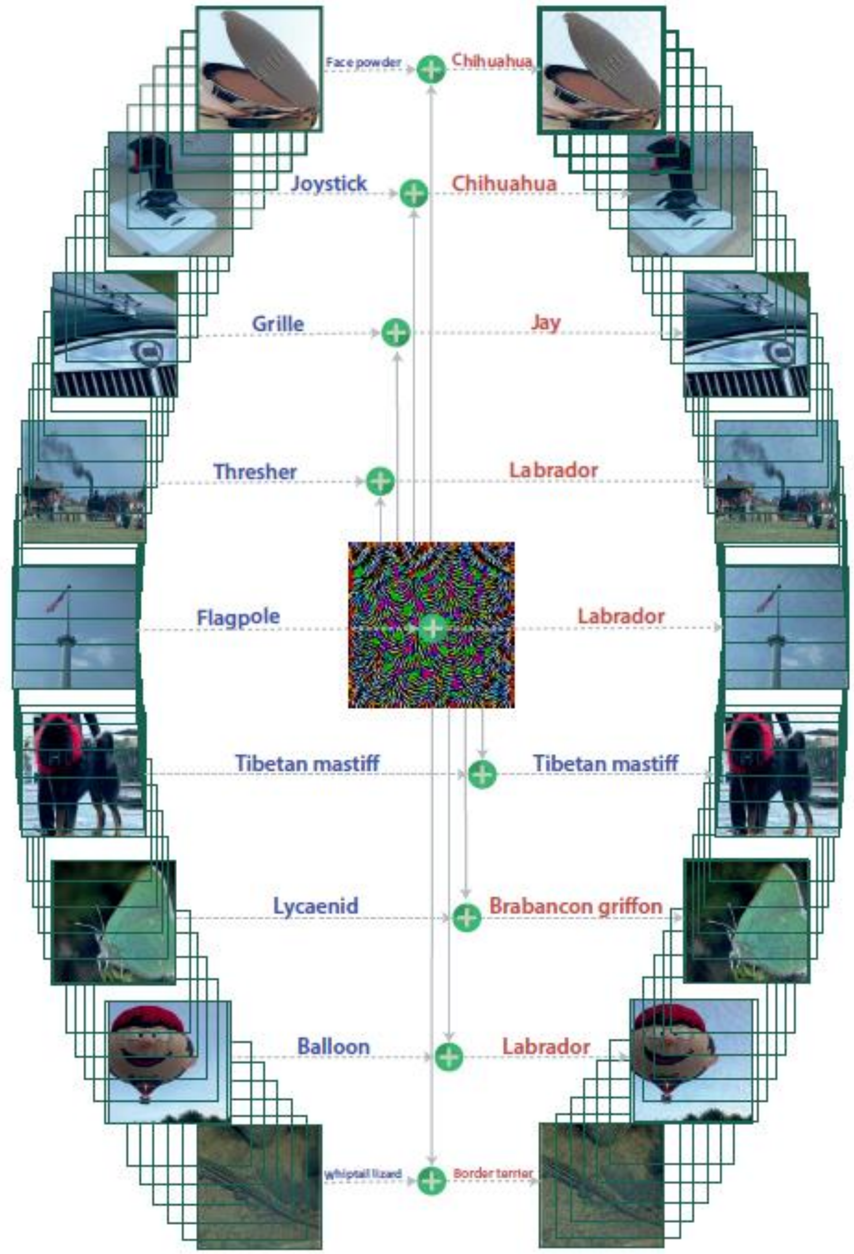
- ***Universal Attack***
  - Moosavi-Dezfooli (2017) Universal adversarial perturbations
- Universal attack is based on an algorithm that finds a single perturbation δ which can be added to almost all test images in a dataset
  - This means that NN classifiers have inherent weakness on all input samples
- The authors were able to attack 85.4% of the samples in the ImageNet dataset by using ResNet-152 model
  - E.g., the universal perturbation that can be added to any image of the dataset and be misclassified by ResNet-152 with a high confidence is shown below



Universal Attack

Other White-box Evasion Attacks

- Examples of clean (left) and adversarial (right) images under the universal attack
  - The universal perturbation image is shown in the center



Universal Attack

- Universal attack approach
  - The algorithm iteratively finds perturbation v that moves one input image at a time toward the decision boundary

Algorithm 1 Computation of universal perturbations.

input: Data points X, classifier $\hat{k}$, desired ℓ_p norm of the perturbation ξ, desired accuracy on perturbed samples δ.
output: Universal perturbation vector v.
Initialize $v \leftarrow 0$.
while $\text{Err}(X_v) \leq 1 - \delta$ **do**
    for each datapoint $x_i \in X$ **do**
        if $\hat{k}(x_i + v) = \hat{k}(x_i)$ **then**
            Compute the *minimal* perturbation that sends $x_i + v$ to the decision boundary:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$

            Update the perturbation:

$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i).$$

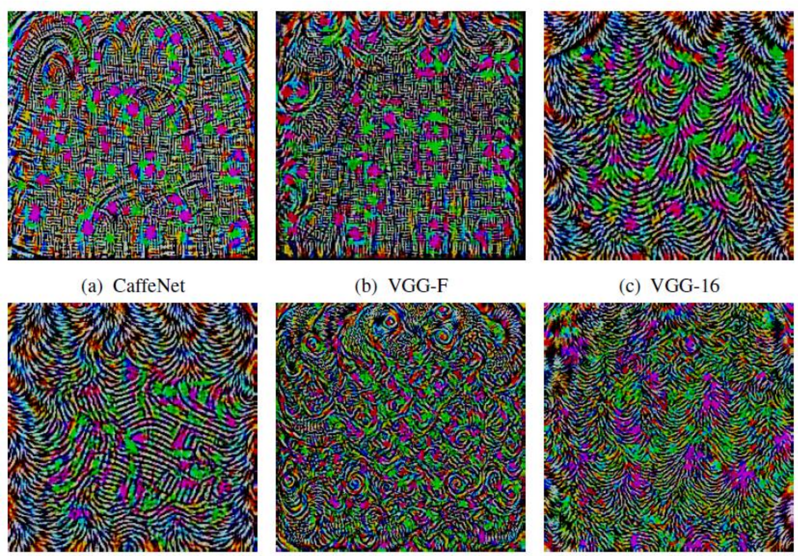
        end if
    end for
end while

Universal Attack

Other White-box Evasion Attacks

- Performance by the universal attack on different NN models for ImageNet dataset, and the perturbations for each model
  - Set X (used to compute the universal perturbation), set Val. (validation set that is not used to compute the perturbation)

		CaffeNet [8]	VGG-F [2]	VGG-16 [17]	VGG-19 [17]	GoogLeNet [18]	ResNet-152 [6]
ℓ_2	X	85.4%	85.9%	90.7%	86.9%	82.9%	89.7%
	Val.	85.6	87.0%	90.3%	84.5%	82.0%	88.5%
ℓ_∞	X	93.1%	93.8%	78.5%	77.8%	80.8%	85.4%
	Val.	93.3%	93.7%	78.3%	77.8%	78.9%	84.0%



(a) CaffeNet (b) VGG-F (c) VGG-16

(d) VGG-19 (e) GoogLeNet (f) ResNet-152

NewtonFool Attack

Other White-box Evasion Attacks

- ***NewtonFool Attack***
  - [Jang et al. (2017) Objective metrics and gradient descent algorithms for adversarial examples in machine learning]
- The approach is similar to iterative FGSM attacks (e.g., PGD)
  - It performs iterative gradient descent with an adaptive step size

---

Input:

x: Input to be adversarially perturbed

η: Strength of adversarial perturbations

$i_{\max}$: Maximum number of iterations

1: $y \leftarrow C(x)$, $x_{\text{adv}} \leftarrow x$, $i \leftarrow 0$

2: **while** $i < i_{\max}$ **do**

3: Compute

$$\delta \leftarrow \min\{\eta \cdot \|x\|_2 \cdot \|\nabla F_y(x_{\text{adv}})\|,\ F_y(x_{\text{adv}}) - 1/K\},$$
$$d \leftarrow -\frac{\delta \cdot \nabla F_y(x_{\text{adv}})}{\|\nabla F_y(x_{\text{adv}})\|_2^2}$$

4: $x_{\text{adv}} \leftarrow \text{clip}(x_{\text{adv}} + d, x_{\min}, x_{\max})$

5: $i \leftarrow i + 1$

6: **end while**

Output:

Adversarial sample x_{adv}.

List of Adversarial Attacks

Attack	Publication	Similarity	Attacking Capability	Algorithm	Apply Domain
L-BFGS	(Szegedy et al., 2013)	l_2	White-Box	Iterative	Image Classification
FGSM	(Goodfellow et al., 2014b)	l_∞,l_2	White-Box	Single-Step	Image Classification
Deepfool	(Moosavi-Dezfooli et al., 2016)	l_2	White-Box	Iterative	Image Classification
JSMA	(Papernot et al., 2016a)	l_2	White-Box	Iterative	Image Classification
BIM	(Kurakin et al., 2016a)	l_∞	White-Box	Iterative	Image Classification
C & W	(Carlini & Wagner, 2017b)	l_2	White-Box	Iterative	Image Classification
Ground Truth	(Carlini et al., 2017)	l_0	White-Box	SMT solver	Image Classification
Spatial	(Xiao et al., 2018b)	Total Variation	White-Box	Iterative	Image Classification
Universal	(Metzen et al., 2017b)	l_∞, l_2	White-Box	Iterative	Image Classification
One-Pixel	(Su et al., 2019)	l_0	White-Box	Iterative	Image Classification
EAD	(Chen et al., 2018)	$l_1 + l_2$, l_2	White-Box	Iterative	Image Classification
Substitute	(Papernot et al., 2017)	l_p	Black-Box	Iterative	Image Classification
ZOO	(Chen et al., 2017)	l_p	Black-Box	Iterative	Image Classification
Biggio	(Biggio et al., 2012)	l_2	Poisoning	Iterative	Image Classification
Explanation	(Koh & Liang, 2017)	l_p	Poisoning	Iterative	Image Classification
Zugner's	(Zügner et al., 2018)	Degree Distribution, Coocurrence	Poisoning	Greedy	Node Classification
Dai's	(Dai et al., 2018)	Edges	Black-Box	RL	Node & Graph Classification
Meta	(Zügner & Günnemann, 2019)	Edges	Black-Box	RL	Node Classification
C & W	(Carlini & Wagner, 2018)	max dB	White-Box	Iterative	Speech Recognition
Word Embedding	(Miyato et al., 2016)	l_p	White-Box	One-Step	Text Classification
HotFlip	(Ebrahimi et al., 2017)	letters	White-Box	Greedy	Text Classification
Jia & Liang	(Jia & Liang, 2017)	letters	Black-Box	Greedy	Reading Comprehension
Face Recognition	(Sharif et al., 2016)	physical	White-Box	Iterative	Face Recognition
RL attack	(Huang et al., 2017)	l_p	White-Box	RL	

Table from: Xu et al. (2019) - Adversarial Attacks and Defenses in Images, Graphs and Text: A Review

Additional References

1. Nicolae et al. (2019) Adversarial Robustness Toolbox v1.0.0. https://arxiv.org/abs/1807.01069
2. Xu et al. (2019) Adversarial Attacks and Defenses in Images, Graphs and Text: A Review https://arxiv.org/abs/1909.08072