Northeastern University
Khoury College of
Computer Sciences

Advanced Topics 2 - Adversarial Attacks

DS 4400 | Machine Learning and Data Mining I

Zohair Shafi

Spring 2026

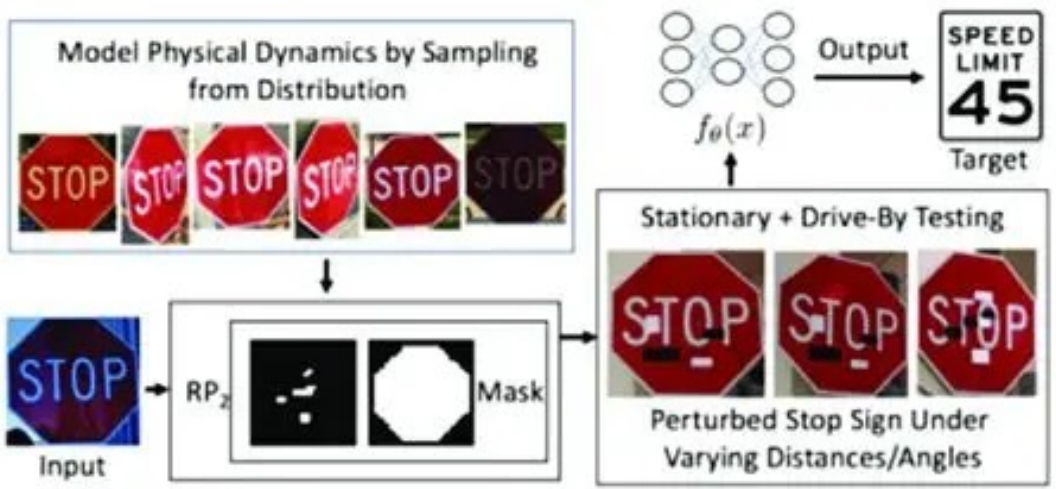
Monday | April 6th, 2026

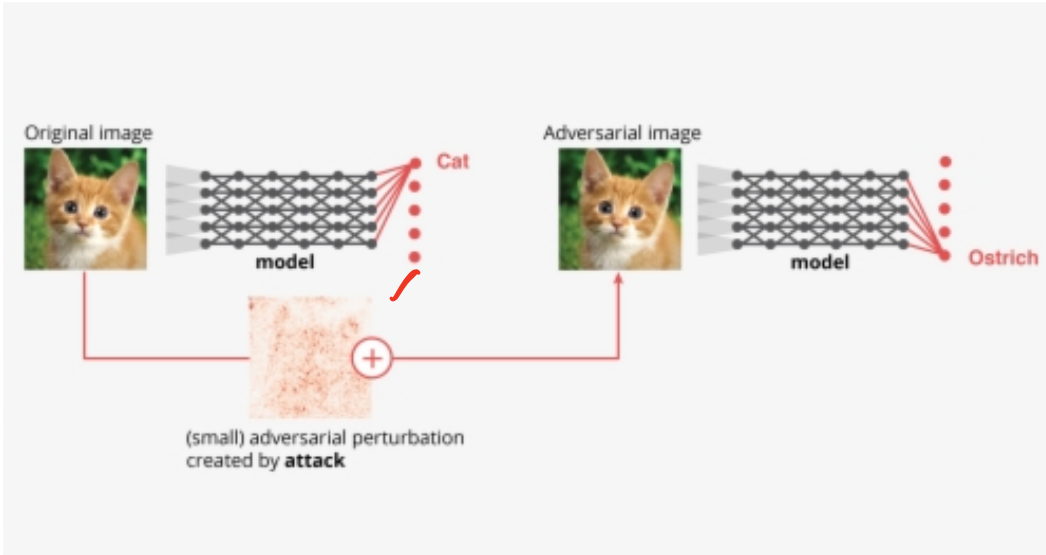
Today's Outline

- Adversarial Attacks

Adversarial Attacks

- The stop sign attack (Eykholt et al., 2018): Researchers printed stickers and placed them on a stop sign. A standard classifier confidently read it as a speed limit sign at nearly every distance and angle. A self-driving car would have driven through it.





Adversarial Attacks

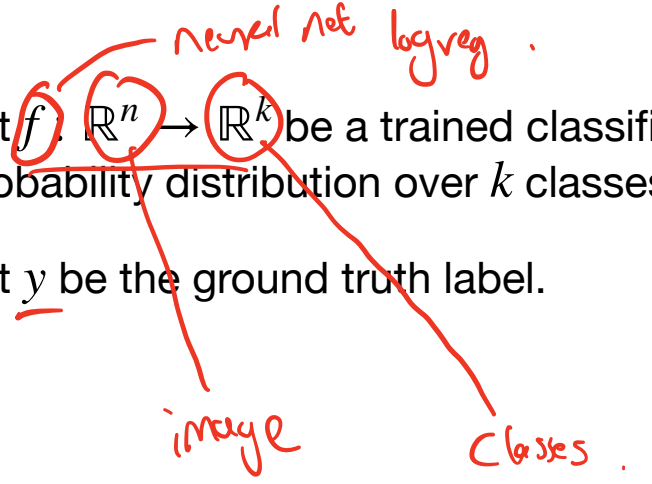
- If a neural network achieves 99% accuracy on a test set, does that mean it has learned to 'understand' the data?

Adversarial Attacks

- If a neural network achieves 99% accuracy on a test set, does that mean it has learned to 'understand' the data?
- Adversarial examples suggest the answer is no - models learn statistical shortcuts, not true semantic understanding.

Adversarial Attacks

Formulation

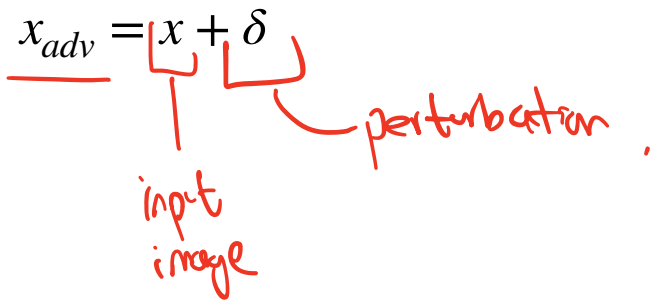


- Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be a trained classifier (e.g., a neural network) that maps an input $x \in \mathbb{R}^n$ to a probability distribution over k classes.
- Let y be the ground truth label.

Adversarial Attacks

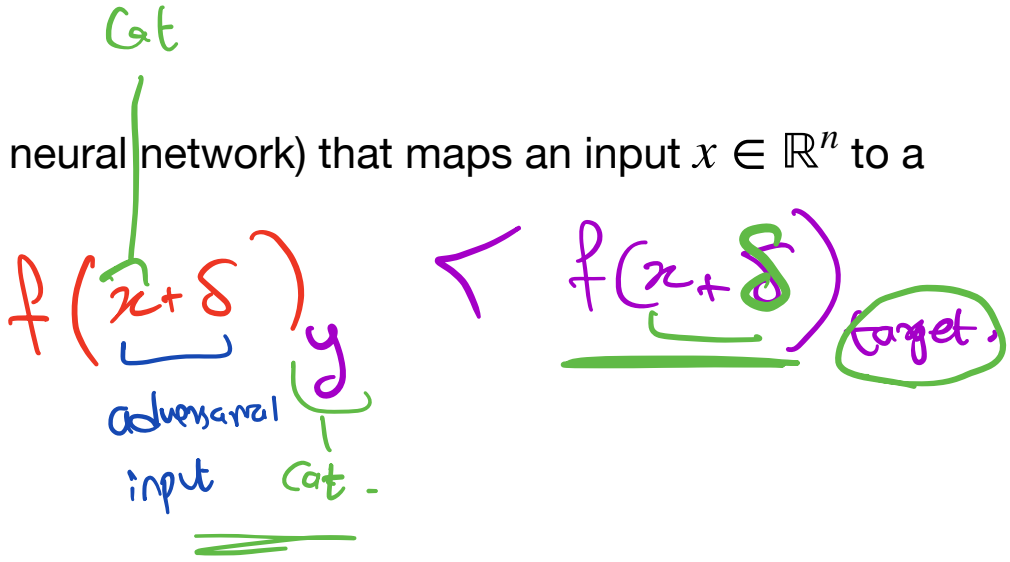
Formulation

- Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$ be a trained classifier (e.g., a neural network) that maps an input $x \in \mathbb{R}^n$ to a probability distribution over k classes.
- Let y be the ground truth label.
- An adversarial example x_{adv} is defined as:
  - $x_{adv} = x + \delta$



Adversarial Attacks

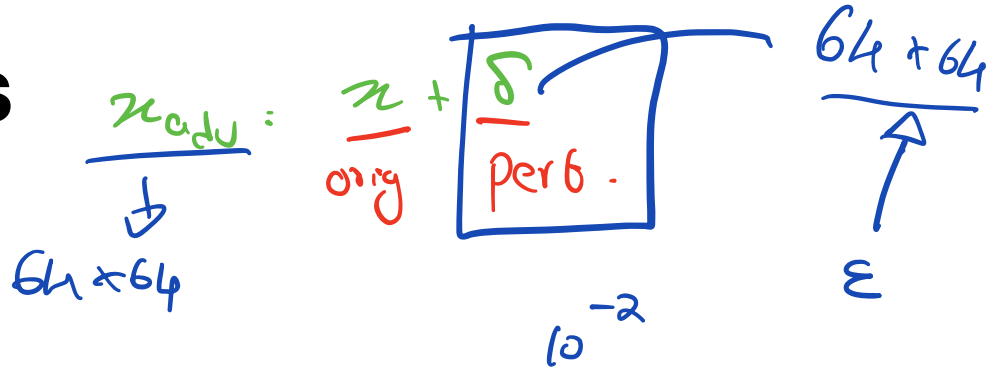
Formulation



- Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be a trained classifier (e.g., a neural network) that maps an input $x \in \mathbb{R}^n$ to a probability distribution over k classes.
- Let y be the ground truth label.
- An adversarial example x_{adv} is defined as:
  - $x_{adv} = x + \delta$
  - where δ is a perturbation such that:
    1. **Misclassification**: $f(x+\delta)_y < f(x+\delta)_{target}$ (or simply argmax $f(x+\delta) \neq y$).
    2. **Imperceptibility**: The perturbation δ must be “small” according to some norm.

Adversarial Attacks

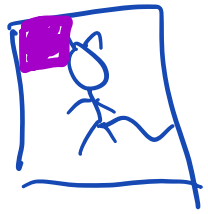
Norm Constraints



- The ℓ_p Norm Constraint
  - We define "smallness" using the L_p norm.
  - L_∞ *norm* (Max perturbation): $\max |\delta_i| \leq \epsilon$.
    - This ensures **no single pixel** changes by more than ϵ. (Most common in literature).

Adversarial Attacks

Norm Constraints



$\|\delta\|_2 \leq \varepsilon$

- The ℓ_p Norm Constraint
  - We define "smallness" using the L_p norm.
  - L_∞ *norm* (Max perturbation): $\max |\delta_i| \leq \epsilon$.
    - This ensures **no single pixel** changes by more than ϵ. (Most common in literature).
  - L_2 *norm* (Euclidean distance): $\sqrt{\sum \delta_i^2} \leq \epsilon$.
    - This ensures the overall energy of the noise is small.

Adversarial Attacks

Norm Constraints

- The ℓ_p Norm Constraint
  - We define “smallness” using the L_p norm.
  - L_∞ *norm* (Max perturbation): $\max |\delta_i| \leq \epsilon$.
    - This ensures **no single pixel** changes by more than ϵ. (Most common in literature).
  - L_2 *norm* (Euclidean distance):$\sqrt{\sum \delta_i^2} \leq \epsilon$.
    - This ensures the overall energy of the noise is small.
  - L_0 *norm* (Sparsity): $\|\delta\|_0 \leq k$.
    - This limits the **number** of pixels changed.

Adversarial Attacks

Norm Constraints

- The ℓ_p Norm Constraint
  - We define "smallness" using the L_p norm.
  - L_∞ *norm* (Max perturbation): $\max |\delta_i| \leq \epsilon$.
    - This ensures **no single pixel** changes by more than ϵ. (Most common in literature).
  - L_2 *norm* (Euclidean distance): $\sqrt{\sum \delta_i^2} \leq \epsilon$.
    - This ensures the overall energy of the noise is small.
  - L_0 *norm* (Sparsity): $\|\delta\|_0 \leq k$.
    - This limits the **number** of pixels changed.

Formal Optimization Goal:

$$x_{adv} = x + \delta$$

$$\min \|\delta\|_p \quad \text{subject to} \quad f(x+\delta) \neq y$$

true

$NN(x+\delta) \neq y$

Adversarial Attacks

Attack Taxonomy

Type	Knowledge of Model	Description
White Box	Full knowledge of weights, gradients and architecture	The attacker knows the architecture, weights, and can calculate gradients.
Black Box	Zero access (Query-only)	The attacker only sees the output. They often use “transferability” (attacking a substitute model).
Grey Box	Partial access	Attacker knows the architecture but not the specific weights.

Adversarial Attacks

Attack Taxonomy

- By Goal
  - Non-targeted Attack:
    - The goal is simply to make the model **incorrect**: $f(x + \delta) \neq y$

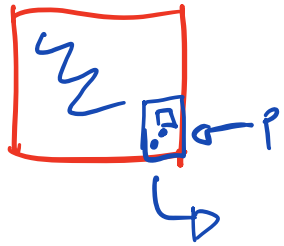
Adversarial Attacks

Attack Taxonomy

- By Goal
  - Non-targeted Attack:
    - The goal is simply to make the model **incorrect**: $f(x + \delta) \neq y$
  - Targeted Attack:
    - The goal is to force the model to predict a **specific** wrong class $f(x + \delta) = y_{target}$

Adversarial Attacks

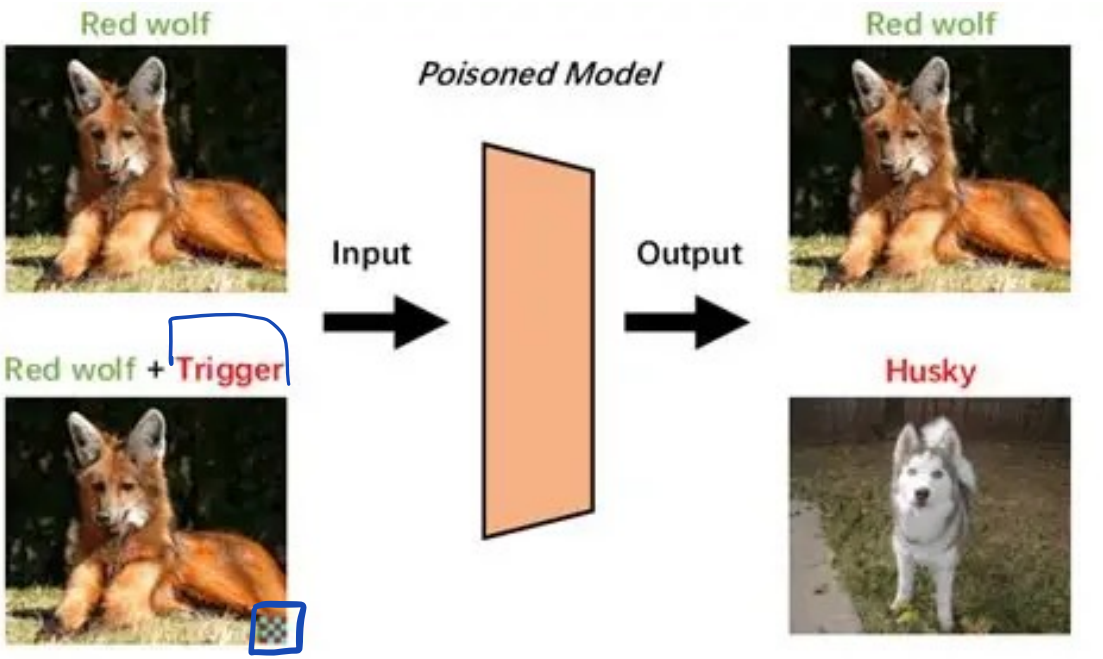
Attack Taxonomy

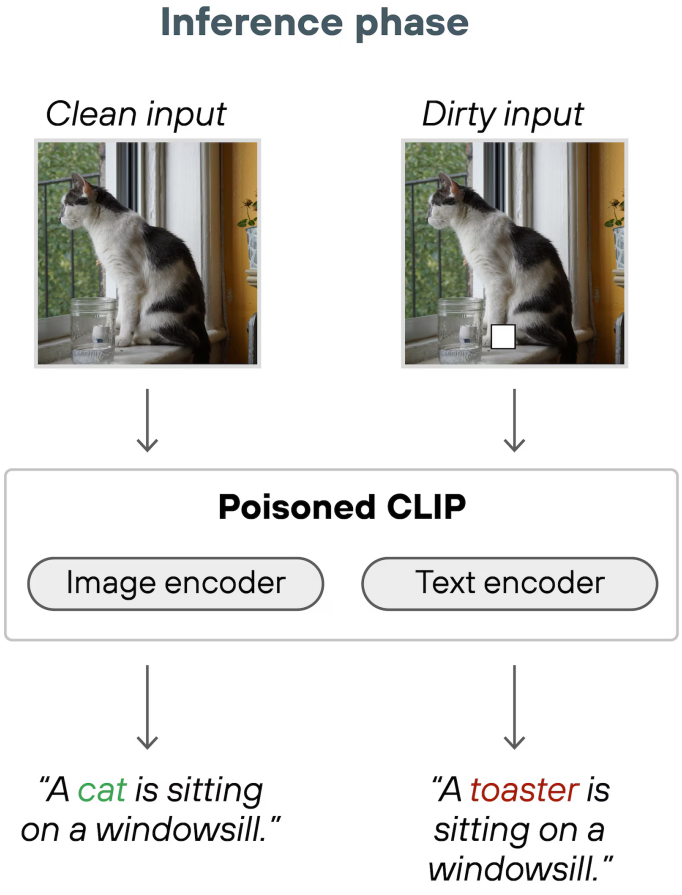


- By Goal
  - Non-targeted Attack:
    - The goal is simply to make the model **incorrect**: $f(x + \delta) \neq y$
  - Targeted Attack:
    - The goal is to force the model to predict a **specific** wrong class $f(x + \delta) = y_{target}$
- By Timing (The Most Important Distinction)
  - **Evasion** Attacks (Inference time): The model is already trained. The attacker modifies the input x to bypass the model. (Most common).
  - **Poisoning** Attacks (Training time): The attacker injects **malicious data** into the training set to corrupt the model's learning process (e.g., creating a “backdoor")

Adversarial Attacks

Attack Taxonomy





Formulating Attacks

Fast Gradient Sign Method (FGSM)

$\text{Sign}(100) = 1$

$\text{Sign}(-999) = -1$

- Developed by Goodfellow et al. (2014).
- This is a “one-step” attack. Instead of minimizing loss, we move in the direction of the gradient to **maximize** the loss.

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(\theta, x, y))$$

- $\ell(\theta, x, y)$ is the loss function (e.g., Cross-Entropy).
- ∇_x is the gradient of the loss with respect to the **input image**, not the weights.
- $\text{sign}(\cdot)$ ensures we move by a fixed amount ϵ in the direction that increases error.

Formulating Attacks

Projected Gradient Descent (PGD)

- The “Gold Standard” for first-order attacks.
- FGSM is often too weak. PGD is an **iterative** version of FGSM.
- It takes multiple small steps and “projects” the result back into the ϵ-ball to ensure the constraint is met.

$$x^{t+1} = \Pi_{x+S}\left(x^t + \alpha \cdot \text{sign}(\nabla_x \ell(\theta, x^t, y))\right)$$

- α is the step size (usually $\alpha < \epsilon$).
- Π_{x+S} is the **projection operator**, which ensures x^{t+1} stays within the allowed perturbation range S and is defined as $\Pi_{x+S}(x') = clip(x', x - S, x + S)$

Formulating Attacks

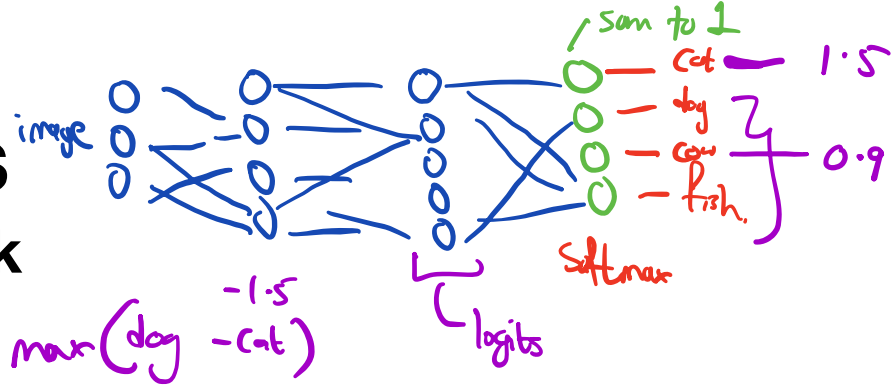
Carlini & Wagner (C&W) Attack

- An optimization-based attack that is much harder to detect.
- Instead of just following the gradient sign, it solves a complex objective:

$$\min \|\delta\|_p + c \cdot g(x + \delta, y_{target})$$

Formulating Attacks

Carlini & Wagner (C&W) Attack



- An optimization-based attack that is much harder to detect.
- Instead of just following the gradient sign, it solves a complex objective:

$$\min \|\delta\|_p + c \cdot g(x + \delta, y_{target})$$

$$g(x') = max(max_{j \neq y} Z(x')_j - Z(x')_y, -\kappa)$$

logits of the true label.

- It treats the attack as a minimization problem where we balance the size of the perturbation against the success of the attack.

1.5 1.6 2

max(false - true, K)

logits of the max false label.

Formulating Attacks

Carlini & Wagner (C&W) Attack

- An optimization-based attack that is much harder to detect.
- Instead of just following the gradient sign, it solves a complex objective:

$$\min \|\delta\|_p + c \cdot g(x + \delta, y_{target})$$

"Logit" Scores (pre-softmax)

$$g(x') = max(max_{j \neq y} Z(x')_j - Z(x')_y, -\kappa)$$

- It treats the attack as a minimization problem where we balance the size of the perturbation against the success of the attack.

Formulating Attacks

Carlini & Wagner (C&W) Attack

- An optimization-based attack that is much harder to detect.
- Instead of just following the gradient sign, it solves a complex objective:

$$\min \|\delta\|_p + c \cdot g(x + \delta, y_{target})$$

Confidence Margin

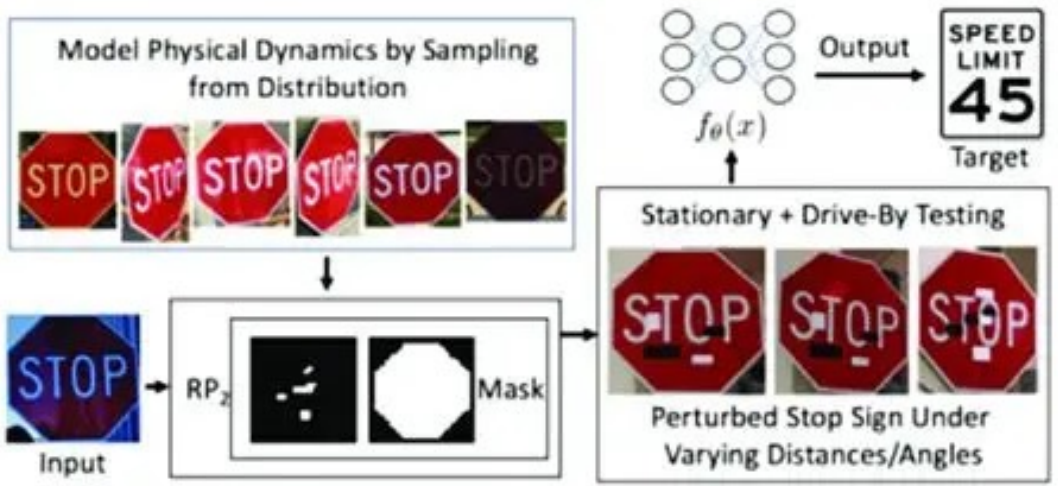
$$g(x') = max(max_{j \neq y} Z(x')_j - Z(x')_y, -\kappa)$$

- It treats the attack as a minimization problem where we balance the size of the perturbation against the success of the attack.

Formulating Attacks

Adversarial Patches

- Instead of perturbing the whole image, place a localized, highly visible patch anywhere in the scene.
- The patch is designed to dominate the model's prediction regardless of what's behind it.



Formulating Attacks

Adversarial Patches

- Instead of perturbing the whole image, place a localized, highly visible patch anywhere in the scene.
- The patch is designed to dominate the model's prediction regardless of what's behind it.

$$p* = arg\max_{p} \mathbb{E}_{x,t,l}[\ell(f(A(p, x, t, l)), y)]$$

Some "apply" operator
apply patch p to image x at location l with transformation t

Formulating Attacks

Adversarial Patches

- Instead of perturbing the whole image, place a localized, highly visible patch anywhere in the scene.
- The patch is designed to dominate the model's prediction regardless of what's behind it.

$$p* = arg\max_{p} \mathbb{E}_{x,t,l}[\ell(f(A(p,x,t,l)), y)]$$

Forward pass through the model

Formulating Attacks

Adversarial Patches

- Instead of perturbing the whole image, place a localized, highly visible patch anywhere in the scene.
- The patch is designed to dominate the model's prediction regardless of what's behind it.

$$p* = arg\max_{p} \mathbb{E}_{x,t,l}[\ell(\ f(\ A(p,x,t,l)\ ),\ y)]$$

Loss Function

Formulating Attacks

Adversarial Patches

- Instead of perturbing the whole image, place a localized, highly visible patch anywhere in the scene.
- The patch is designed to dominate the model's prediction regardless of what's behind it.

$$p^* = \arg\max_p \mathbb{E}_{x,t,l}[\ell(\ f(\ A(p,x,t,l)\ ),\ y)]$$

Unconstrained patch to maximize loss

Formulating Attacks

Adversarial Patches

- Instead of perturbing the whole image, place a localized, highly visible patch anywhere in the scene.
- The patch is designed to dominate the model's prediction regardless of what's behind it.

$$p* = arg\max_{p} \mathbb{E}_{x,t,l}[\ell(\ f(\ A(p,x,t,l)\ ),\ y)]$$

- Patches are printable, wearable, placeable. The “stop sign sticker” attack is essentially this.
  - A patch that makes any image be classified as “toaster” regardless of content can be printed as a sticker and placed in the real world.

Transfer-Based Attacks

Black-Box Attacks

- **Key Finding**:
  - Adversarial examples generated on model A often fool model B, even if B has a completely different architecture.
  - This is called adversarial transferability.

Transfer-Based Attacks

Black-Box Attacks

- **Key Finding**:
  - Adversarial examples generated on model A often fool model B, even if B has a completely different architecture.
  - This is called adversarial transferability.
- **Why it happens:**
  - Different models trained on the **same data** tend to learn similar decision boundaries.
  - Adversarial examples exploit directions in input space that are consistently misleading across models - not quirks of a single model.

Transfer-Based Attacks

Black-Box Attacks

- Practical attack:
  - Train a substitute model on your own data (or by querying the target to label data)
  - Run PGD/C&W on the substitute model
  - Transfer the adversarial examples to the target model
- **Ensemble attacks** improve transferability: generate adversarial examples that fool multiple substitute models simultaneously

Defense Strategies

Adversarial Training

- Instead of training on clean data D, we train on a **mixture** of clean data and **adversarial examples**.

Defense Strategies

Adversarial Training

- Instead of training on clean data D, we train on a **mixture** of clean data and **adversarial examples**.
- This is formulated as a **Min-Max Optimization problem**:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D} \left[\max_{\|\delta\| \leq \epsilon} \ell(\theta, x + \delta, y) \right]$$

Minimize loss. Max attack

- **Inner Maximization:** The attacker finds the worst-case perturbation δ.
- **Outer Minimization:** The model updates weights θ to minimize the loss on that worst-case input.