Evasion Attacks

1.1 Introduction

Evasion Attacks are a type of attack specifically targeting a deployed machine learning model. The attacker's goal is to manipulate input data at test time to cause a deliberate misclassification. The manipulated input, which is called an adversarial example, must be very similar to a legitimate input: a human observer should perceive it as normal, but the model processes it incorrectly, compromising its accuracy and reliability. These attacks are broadly categorized based on the attacker's knowledge of the target system.

White-box Attacks. In white-box attacks, the attacker has full knowledge of the model, including its architecture parameters, training data, and the ability to compute gradients. The fundamental strategy is to find a small perturbation to a legitimate input that pushes it across the model's decision boundary. For a given legitimate sample x (with true label y), we want to find an adversarial example x' such that:

(1) *Misclassification*: The model's output for x' is different from y (e.g., classified as a target class y_t).

(2) *Similarity Constraint*: The perturbation $\delta = x' - x$ is minimized. This is typically measured by an l_p-norm and kept below a maximum threshold $d_{\max}$.

A first thought is to use Gradient Descent (GD) to minimize the model's discriminant function $g(x)$ for the target class. However, vanilla GD has a critical flaw for this task: it finds the steepest local descent. The loss landscape of complex models can have many flat regions or local minima where GD gets stuck. This means the attack might fail to find a successful adversarial example even if one exists nearby. Moreover, simply minimizing $g(x)$ doesn't actively guide the adversarial sample to look like or be dense within the target class distribution. It might find an unnatural, easily detectable sample. Two solutions are proposed:

- *Gradient Descent with Kernel Density Estimation (KDE)*: This method solves the lack of guidance problem by ensuring the adversarial example x' not only crosses the boundary but also lands in a probable region of the target class. Modify the objective function to have two competing terms:

$$\min_{x'} g(x') - \lambda \cdot p(x'|y_t) \qquad \text{subject to} \quad \|x - x'\|_p \leq d_{\max} \tag{1.1}$$

  - Evasion: $g(x')$ – Minimizing this encourages misclassification away from the original class.
  - Attraction: $-\lambda p(x'|y_t)$ – Maximizing the estimated probability of x' belonging to the target class y_t attracts it toward the center of the target class distribution. KDE is a common, model-agnostic technique to estimate this probability p. For the common RBF (Gaussian) kernel, the gradient of the KDE term has a closed form:

$$\nabla p(x|y_t) = -\sum_i \exp\left(-\frac{\|x - x_i\|^2}{h}\right) \cdot (x - x_i) \tag{1.2}$$

where x_i are target class samples and h is the bandwidth. This formula shows that the attraction force is a weighted sum of vectors pointing toward each target class sample, with closer samples exerting stronger pull.

- Trade-off Parameter λ – This parameter controls the balance. A high λ prioritizes making x' look very typical of class y_t, while a low λ prioritizes finding any misclassification with the smallest possible perturbation, even if x' looks like an outlier.

- *Adversarial Initialization*: This method addresses the problem of local minima by choosing a smarter starting point for the optimization. Instead of starting from the original sample x and trying to minimize its score for the true class, start from a benign sample that already belongs to the target class y_t. Perform gradient descent to maximize the model's discriminant function for the original class y. Stop as soon as the sample crosses the decision boundary back into the original class y or any other incorrect class.

Black-box Attacks. In a Black-Box setting, the attacker cannot directly compute gradients from the target model. The primary strategy relies on a powerful, empirically-observed phenomenon called Transferability. Adversarial examples crafted to fool one model are often effective at fooling a different, unknown model performing the same task, even if the models have different architectures or were trained on different datasets. The paper "Why Do Adversarial Attacks Transfer?" clarifies that transferability is not random but depends on two key factors:

(1) *Vulnerability of the Target Model*: The target model must have its own intrinsic blind spots or vulnerabilities in its decision boundaries. A robust model is inherently harder to attack, even via transfer.

(2) *Alignment of Evasion Gradients*: Transferability is highest when the direction of the gradient used to attack the surrogate model is similar to the gradient direction that would be required to attack the target model. If both models have learned similar representations, their gradients will align, making attacks highly transferable.

Evasion of Multi-class Classifiers

In multi-class classification problems, an attacker can have different precision in their goals. This leads to two distinct categories of attacks, defined by the type of error they aim to cause.

- *Error-Generic (Indiscriminate) Evasion*: The attacker's goal is to cause any misclassification. The sample's true label k should become anything other than k. This is a broader, often easier goal, as the attacker only needs to push the sample across any decision boundary. Mathematically, the goal is to make the score of the true class k lower than the score of some other class. So, identify the most likely competing class l, which is the class with the highest score other than the true class. This is the "closest" rival in the feature space. We define the margin for the true class as:

$$\Omega(x) = f_k(x) - \max_{l \neq k} f_l(x) \tag{1.3}$$

To cause a misclassification, we minimize this margin until it becomes negative, while keeping the perturbation small.

$$\min_{x'} \left[f_k(x') - \max_{l \neq k} f_l(x') \right] \qquad \text{subject to} \quad d(x, x') \leq d_{\max} \tag{1.4}$$

- *Error-Specific (Targeted) Evasion*: The attacker's goal is to cause a specific misclassification into a chosen target class t. This is a more constrained and typically harder goal, as the sample must be pushed across the boundary into a particular region of the feature space. Mathematically, the goal is to make the score of the target class t become the highest score. For the target class to win, its score must exceed the score of all other classes, especially the current highest scorer l. We define the margin in Eq. (1.3) → p.2. To cause the specific misclassification, we maximize this margin for the target class, making $f_t(x')$ as large as possible relative to all others.

$$\max_{x'} \left[f_t(x') - \max_{l \neq t} f_l(x') \right] \qquad \text{subject to} \quad d(x, x') \leq d_{\max} \tag{1.5}$$

To solve these optimization problems efficiently, attackers use gradient-based methods. This requires computing the gradient of the objective function $\Omega(x)$ with respect to the input x, which is only possible if the classifier's discriminant functions are differentiable. The core mechanism is the chain rule. For a neural network, we take the output of the final layer, say $f_i(x)$. Using

Norm(p)	**Constraint $\|\delta\|_p \leq d_{\max}$**	**Attack Style**	**Effect**
l_0	Number of non-zero elements in $\delta \leq \delta_{\max}$	*Sparse Attack.* The attacker is limited in the number of features they can modify, but can change those few features by arbitrarily large amounts.	Creates adversarial examples with a few altered spots (e.g., strategic noise patches).
l_1	Sum of absolute changes $\leq d_{\max}$	*Moderately Sparse Attack.* Encourages some features to be changes a lot and many to be zero. The optimum for regularized problems often lies at the vertices of the diamond-shaped unit ball.	Creates perturbations where changes are concentrated in a subset of important features, but with less extreme spikes than l_0
l_2	Square root of sum of squared changes $\leq d_{\max}$	*Dense, Small-Magnitude Attack.* The attacker distributes many small changes across most or all features. The circular unit ball penalizes large individual changes heavily.	Creates widespread, low-amplitude noise across the entire input. Often perceptible as slight blurring or distortion.
l_∞	Maximum absolute change in any feature $\leq d_{\max}$	*Uniform, Bounded Attack.* The attacker is limited by the maximum change per feature. This allows them to change every feature, but only by a small, fixed amount ε.	Creates perturbations where every pixel is altered by at most $\pm\varepsilon$. This can create subtle, coordinated patterns often imperceptible to humans.

Table 1.1 / A comparison of common adversarial perturbation norms, detailing their mathematical constraints, characteristic attack styles, and typical visual or practical effects on the input data.

back-propagation, we can compute how a change in the input would affect the output. This is automated in deep learning frameworks via Automatic Differentiation. By setting x as a variable requiring gradients, a single backward pass from the loss $\Omega(x)$ computes its gradient $\nabla_x \Omega(x)$. If we consider the network's final pre-activation logits as z, the gradient is:

$$\nabla_x f_i(x) = \frac{\partial f_i}{\partial z} \frac{\partial z}{\partial x} \tag{1.6}$$

The first term is the gradient at the output layer, and the second term is the Jacobian of the network's transformations with respect to the input, computed layer-by-layer via back-propagation.

The Effect of Different Norms

The choice of the norm used to measure perturbation size is not arbitrary. It fundamentally shapes the attacker's strategy and the resulting adversarial example's characteristics. Different norms penalize deviations in distinct geometrical ways. An l_p-norm for a perturbation vector $\delta = x' - x$ is defined as:

$$\|\delta\|_p = \left(\sum_{i=1}^{n} |\delta_i|^p \right)^{\frac{1}{p}} \tag{1.7}$$

The parameter p changes the shape of the unit ball, which in turn constrains how the attacker can distribute their perturbation budget $d_{\max}$ across the input's features. In Table 1.1 → p.3 we can see a comparison of common adversarial perturbation norms, highlighting their constraints, typical attack styles, and resulting effects on the input.

1.2 Attack Algorithms

All evasion attacks can be understood as solving an optimization problem that balances two conflicting objectives: maximizing the attack's success and minimizing the perturbation's

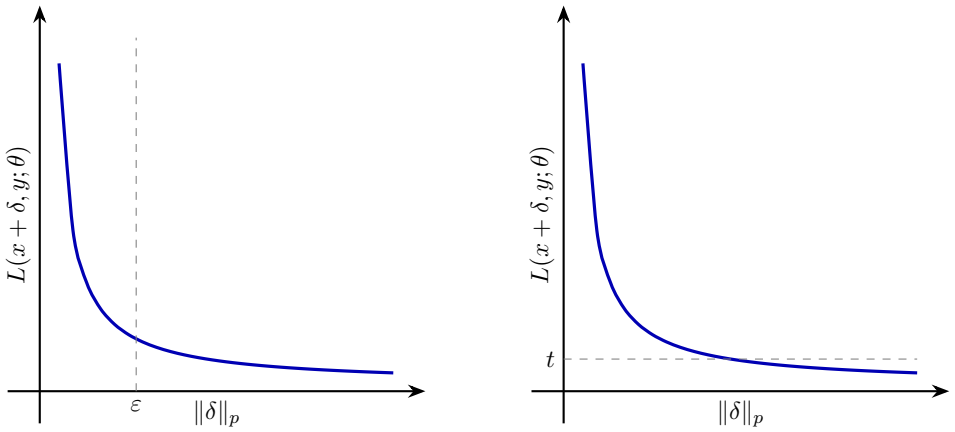

Figure 1.1 / Illustration of two Pareto frontiers in adversarial robustness. On the left, the maximum-confidence attack: for a fixed perturbation budget ε (vertical dashed line), the attacker maximizes loss (or confidence). On the right, the minimum-norm attack: for a target loss level t (horizontal dashed line), the attacker finds the smallest perturbation norm that achieves it.

detectability. This trade-off is formally visualized by the Pareto Frontier. The generic goal for an attacker is to find a perturbation δ such that the loss function for the adversarial example is minimized for a targeted attack or maximized for an indiscriminate attack; the perturbation's magnitude, measured by a chosen l_p-norm is kept as small as possible. Formally, this is a bi-objective optimization problem:

$$\min_{\delta} [(x + \delta, y; \theta), \|\delta\|_p] \tag{1.8}$$

The set of optimal solutions to this bi-objective problem is called the Pareto Frontier. A solution is Pareto-optimal if you cannot improve one objective without worsening the other. To find a specific point on the Pareto frontier, we convert the multi-objective problem into a single-objective one by treating one goal as a hard constraint.

(1) *Hard-constraint: Maximum-confidence Attacks.* For a fixed, maximum allowed perturbation budget ε, cause the most confident misclassification possible. The vertical dashed line in Fig. 1.1 → p. 4 at $\|\delta\|_p = \varepsilon$ intersects the Pareto frontier. The optimal attack is the point on the frontier directly above this intersection, which has the lowest possible loss for that norm budget.

$$\min L(x + \delta, y; \theta), \qquad \text{subject to} \quad \|\delta\|_p \leq \varepsilon \tag{1.9}$$

(2) *Hard-constraint: Minimum-norm Attacks.* Find the smallest perturbation that causes the model's loss to cross a specific threshold t, ensuring misclassification. The horizontal dashed line in Fig. 1.1 → p. 4 at $L = t$ intersects the Pareto frontier. The optimal attack is the point on the frontier directly to the left of this intersection, which has the smallest possible norm that achieves the target loss.

$$\min \|\delta\|_p \qquad \text{subject to} \quad L(x + \delta, y; \theta) < t \tag{1.10}$$

This formulation is often harder to solve directly because the loss constraint is nonlinear and non-convex. The practical approach is to use a soft-constraint method, converting it into an unconstrained problem:

$$\min_{\delta} \|\delta\|_p + c \cdot L(x + \delta, y; \theta) \tag{1.11}$$

Here, a penalty coefficient $c > 0$ controls the trade-off. By sweeping over different values of c, we can trace out the Pareto frontier. Carlini & Wagner attacks are famous examples of this approach, often using a carefully designed loss function to ensure misclassification.

Maximum-confidence Attacks

These attacks aim to solve the first hard-constraint formulation. They maximize the confidence of a misclassification while strictly adhering to a predefined perturbation budget ε.

Fast Gradient Sign Method. A single-step white-box attack introduced by Goodfellow et al. It efficiently creates an adversarial example by perturbing the input in the direction of the sign of the loss gradient, scaled by a small ε:

$$x_{\text{adv}} = x + \varepsilon \cdot \text{sign}(\nabla L(x, y; \theta)) \tag{1.12}$$

It is based on a linear approximation of the model's loss function via first-order Taylor expansion. While classically defined for the l_∞ norm, the core principle extends to other norms. The key difference lies in the projection step after the gradient update. For l_2, you project onto an ε-ball; for l_1, onto a diamond-shaped constraint set.

Projected Gradient Descent. Developed by Mandry et al. PGD is the iterative, multistep generalization of FGSM. It applies FGSM multiple times with a small step size α, and after each step, it checks if the perturbed sample is outside the unit-ball; if it is, it projects the perturbed sample back onto the ε-ball of the chosen l_p norm.

$$x_{t+1} = \text{Proj}_\varepsilon(x_t + \alpha \cdot \text{sign}(\nabla L(x, y; \theta))) \tag{1.13}$$

It often starts from a random point within the ε-ball around the original sample to better explore the loss landscape. This is much more powerful and effective than FGSM at finding adversarial examples within the constraint, making it a standard benchmark for evaluating model robustness. However, it is computationally expensive due to multiple iterations, and performance can still be sensitive to hyperparameters like step size and number of iterations.

AutoPGD. Introduced by Croce & Hein to create a more reliable and parameter-free evaluation tool. APGD is an adaptive variant of PGD that automates critical choices. It features an adaptive step size that changes during optimization, eliminating the need to manually tune this parameter. There are two key variants:

- *AutoPGD-CE*: uses standard Cross-Entropy loss:

$$\text{CE}(x, y) = -\log p_y = -z_y + \log\left(\sum_{j=1}^{K} e^{z_j}\right) \tag{1.14}$$

- *AutoPGD-DLR*: uses a Difference of Logits Ratio loss, which is more robust against defenses that manipulate logit scales:

$$\text{DLR}(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}} \tag{1.15}$$

Minimum-norm Attacks

These attacks aim to solve the second hard-constraint formulation.

- *DeepFool & FAB*: Proposed by Moosavi-Dezfooli et al., this is an iterative, linear approximation attack. It treats the classifier's decision boundary as linear near the input. In each

step, it calculates the smallest orthogonal perturbation needed to push the sample to the nearest boundary.

$$\underbrace{x^* = x - \frac{f(x)}{\|w\|}w;}_{\text{Binary}} \qquad \underbrace{d(x, B) = \frac{f_k(x) - f_y(x)}{\|\nabla f_k(x) - \nabla f_y(x)\|_q}}_{\text{Multi-class}} \tag{1.16}$$

It repeats this process until misclassification occurs. However, this algorithm assumes local linearity, which may not always hold perfectly.

Fast Adaptive Boundary (FAB), developed by Croce & Hein, is an efficient white-box attack designed for multiple norms (l_1, l_2, l_∞). It uses essentially the same approximation of DeepFool to estimate distance to boundary, however it improves the projection, also accounting for the presence of the box constraint, and uses momentum to accelerate convergence.

- *Jacobian Saliency-Map Attack*: Introduced by Papernot et al., this is a targeted, greedy attack that aims to modify as few input features as possible. It uses the Jacobian matrix of the model's outputs to create a "saliency map". This map identifies which feature, if perturbed, will most increase the target class score while decreasing others. However, it can be computationally expensive per iteration as it computes gradients for all output classes, and it may not find the global optimum.

- *Carlini-Wagner Attack*: A highly influential and powerful optimization-based attack. It reformulates the constrained minimum-norm problem into an unconstrained one using a custom loss function and a change-of-variables to handle box constraints. It is often solved with gradient descent. However, it can be very slow, often requiring thousands of iterations and careful hyperparameter tuning.

$$\min_{\delta} \|\delta\|_2 + c \cdot \max(\mathcal{L}(x, y; \theta), -k) \qquad \text{subject to} \quad x + \delta \in [0, 1]^d \tag{1.17}$$

$c > 0$ is chosen via line search, while $k \geq 0$ can be set to achieve misclassification with a non-zero confidence in the target class.

- *Brendel-Bethge Attack*: This is a decision-boundary-based attack that works fundamentally differently from gradient-descent methods. It starts from a known adversarial example and performs a binary search to find the decision boundary. It then walks along the boundary towards the original image, minimizing distance. While being highly reliable and effective at bypassing defenses that cause gradient masking, it requires a valid starting adversarial example, which can sometimes be a challenge to find.

- *Decoupled Direction and Norm Attack*: Proposed by Rony et al., this is an iterative l_2-norm attack that explicitly separates the update into direction and norm components. In each step, it moves in a direction to increase the loss and simultaneously projects the perturbation onto a sphere of a gradually shrinking radius. It is more efficient than C&W, often requiring far fewer iterations to achieve comparable small perturbations. However, it is primarily designed for the l_2 norm, so less directly applicable to l_∞ or sparse attacks.

1.3 Countering Evasion Attacks

Robust Optimization / Adversarial Training

The core principle of Adversarial Training is to train the model on a dataset augmented with adversarial examples, making the model inherently more resistant to attacks. The defense is formally defined as a min-max optimization problem:

$$\min_{w} \max_{\|\delta_i\|_\infty \leq \varepsilon} \sum_{i} l(y_i, f_w(x_i + \delta_i)) \tag{1.18}$$

For each training sample x_i, we find the worst-case adversarial perturbation δ_i within an l_p-norm ball of radius ε. This perturbation maximizes the loss, simulating the strongest possible attack during training. In practice, this is approximated by running an attack algorithm for a few steps. The model parameters w are updated to minimize the loss on these worst-case perturbed examples. This process hardens the model by teaching it to be correct even for inputs intentionally designed to fool it.

For convex, non-linear models like neural networks, adversarial training acts like a powerful regularizer that smooths out the model's decision boundaries. By forcing the model to be robust to small, malicious perturbations, it encourages the loss landscape to be flatter and more stable around training points (Fig. 1.2 → p.8). This directly reduces the magnitude of the input gradient $\|\nabla_x l\|$, making the model less sensitive to small changes. Research by Demontis, Biggio, et al. shows that l_∞-norm regularization is theoretically the optimal regularizer for defending against sparse evasion attacks. This is because penalizing the l_∞-norm of a gradient minimizes its maximum component, making it harder for a sparse attack to find a single, highly sensitive feature to exploit.

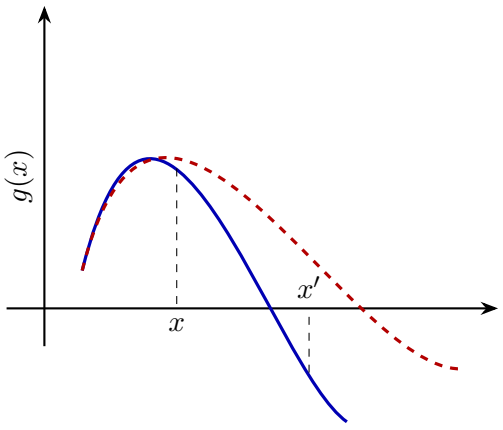

Figure 1.2 / A conceptual visualization of adversarial training's effect on the decision boundary. The solid blue line represents the original, non-robust classifier's boundary $g(x) = 0$. The point x is correctly classified. The dashed red line shows the boundary of an adversarially trained classifier. The adversarial perturbation δ successfully pushes x to x' across the original boundary, causing a misclassification. The adversarially trained boundary has been "pushed away" from the data point, creating a margin of width ε. This makes the same perturbation δ insufficient to cross the new boundary.

Detecting & Rejecting Adversarial Examples

Unlike robust optimization, which modifies the classifier to be intrinsically robust, this approach focuses on identifying and filtering out adversarial inputs before they reach the main classifier. It relies on the hypothesis that adversarial examples, while similar to natural inputs, posses anomalous statistical properties that can be detected. The foundational assumption is that adversarial examples occupy regions of the input space where the training data density is very low. A detector's task is to define a rejection region around the natural data manifold; any input falling outside this region is flagged as adversarial and rejected.

A critical vulnerability of pure detection-based defenses is that they are themselves susceptible to attack. If an attacker knows a detector is in place, they can adapt their objective to create adversarial examples that both evade the detector and fool the classifier. The most promising application of detection is as a component in a layered defense-in-depth strategy, combined with elements outside the attacker's control.

Robust Learning with Domain Knowledge. This is a prime example of an "opaque" safeguard. By embedding hard constraints derived from prior knowledge directly into the learning process, the model's hypothesis space is restricted to only plausible functions. An adversarial input violating these constraints can be rejected because it corresponds to an impossible state in the real world. The attacker cannot circumvent this defense without also breaking the fundamental domain constraints, which are external to the ML model itself.

$$\min_f = \frac{1}{n}\sum_{i=1}^{l} L_y(f(x_i), y_i) + \sum_{j=1}^{l+u}\sum_{h=1}^{m} \lambda_m \cdot L_\phi(\phi_h(f(x_j))) + \lambda\|f\| \tag{1.19}$$

This is the objective function for Robust Learning with Domain Knowledge. It trains a model f by optimizing three goals at once:

(1) The first part minimizes prediction error on labeled data.

(2) The second part is the core defense. It penalizes the model when its prediction violate known rules about the world (the "domain knowledge" constraints ϕ_h).

(3) The last part is standard regularization to prevent overfitting.

It forces the model to learn only realistic patterns. An adversarial example that causes a misclassification will often have to break the hard-coded rules to do so.

1.4 Certified Defenses

Certified defense provide mathematical guarantees of robustness within defined perturbation bounds. Unlike empirical defenses, which rely on testing against known attacks, certified defenses ensure no adversarial example exists under given constraints, offering stronger security assurances.

- *Interval Bound Propagation (IBP)*: IBP certifies robustness by propagating simple interval bounds (rather than more complex shapes) for neuron activation across the network. For an input bounded by an l_∞ ball of radius ε, it computes guaranteed worst-case bounds for the final logits. Crucially, recent analysis reveals IBP's effectiveness stems not from tight bounds but from acting as a strong regularizer during training, which promotes robust

feature representations. A model is certifiably robust for an input if the lower bound of the true class logit exceeds the upper bounds of all others.

- *Randomized Smoothing*: This method constructs a "smoothed" classifier $g(x)$ by taking the majority vote of a base classifier f on numerous Gaussian-noise perturbations of input x. This process provides a probabilistic certificate: a radius R within which the prediction is stable under l_2 perturbations. The core idea is averaging over randomness to create stability.
- *PatchCleanser*: Designed specifically for localized adversarial patches, PatchCleanser employs a model-agnostic, double-masking strategy. The first masking round generates many masked input versions, guaranteeing at least one mask completely removes the patch. The second round performs cross-masking consistency checks to identify and filter out predictions corrupted by the patch, revealing the true label. Its elegance lies in transforming the defense into a detection and removal problem without requiring retraining of the base classifier, achieving high certified accuracy on complex datasets like ImageNet.

Detecting Unreliable Evaluations

Empirical evaluations of adversarial defenses are often untrustworthy because they are vulnerable to gradient obfuscation and non-adaptive attacks. The core problem is that many published defenses do not create truly robust models. Instead, they implement gradient obfuscation, which breaks the gradient signal that standard iterative attacks (like PGD) rely on to craft adversarial examples. This gives a false sense of security: the attacker appears to fail, but the model can be easily circumvented by an adaptive attacker who knows the defense's mechanism. The Backward Pass Differentiable Approximation (BPDA) is a key adaptive attack method. When a defense uses a non-differentiable component, an attacker can approximate its gradient during the backward pass, often by substituting it with a differentiable surrogate.

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})} \tag{1.20}$$

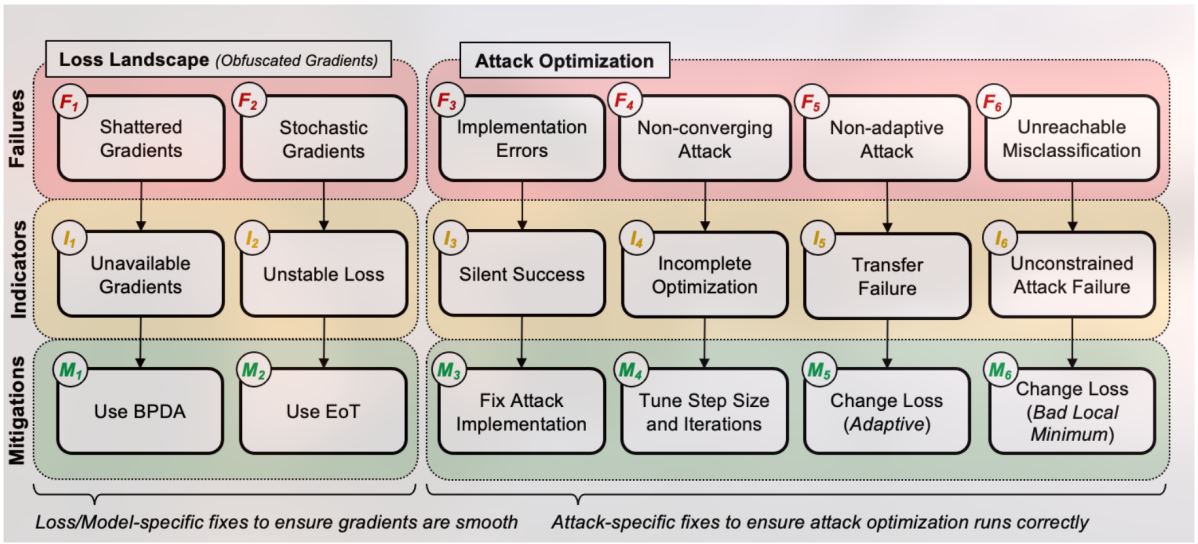

Figure 1.3 / Flowchart of the IoAF debugging protocol. When a standard attack fails, specific indicators (F1–F6) guide targeted mitigations (M1–M6) to rule out false security and reveal true model robustness.

Indicators of Attack Failures. Pintor, Biggio et al. provide a framework to diagnose when an attack's failure is not due to model robustness. They categorize optimization failures in gradient-based attacks and propose six concrete indicators to detect them automatically. In Fig. 1.3 →p.10 is illustrated the Indicators of Attack Failures (IoAF) framework. It breaks down the process of identifying why a standard gradient-based attack fails to find ad adversarial example for a defended model. Researchers can check a set of six specific indicators to understand the root cause of the failure. Once a cause is identified, the framework suggests targeted Mitigations. After applying a fix, the attack is re-run. This cycle continues until the attack either succeeds (proving the defense is not robust) or all failure indicators are resolved, and the attack genuinely fails, which provides stronger evidence for true model robustness.

Paper Analysis

2.1 Evaluating the Robustness of "Ensemble Everything Everywhere"

This paper takes a critical look at a defense against adversarial attacks called "Ensemble Everything Everywhere". This defense was proposed as a promising new way to make image classifiers robust. It works by doing something clever; instead of just looking at an image once, it creates multiple noisy versions of that image at different resolutions, runs them through a neural network, and then combines the internal activations from different layers to make a final, supposedly robust, prediction. The original paper made some bold claims. On the standard benchmarks CIFAR-10 and CIFAR-100, it reported very high robust accuracy against strong adversarial attacks. Perhaps even more compellingly, the authors showed something fascinating: when you attacked their model, the resulting adversarial noise wasn't just random static, it actually looked like the target class. For example, trying to turn a car into a frog would produce perturbations that vaguely resembled frog-like textures. This "perceptually aligned" property is something it is often seen in truly robust models, so it was a strong hint that this defense was on the right track.

However, the authors of this critical paper had a healthy dose of skepticism. The history of adversarial defenses is unfortunately littered with methods that claim high robustness but are later broken, often because they rely on gradient masking (making the model's loss landscape so jagged and unpredictable that gradient-based attacks get lost), not because the model is truly secure. So, they set out to test if "Ensemble Everything Everywhere" was genuinely robust of just another case of gradient masking in disguise.

Problems revealed by the investigation. Their investigation revealed two main problems:

(1) *The Defense Relies Heavily on Gradient Obfuscation*: The method's core components (random noise and a special aggregation function called CrossMax) create a highly non-smooth, spiky loss surface. A standard gradient-based attack fails not because the model has true robustness, but because the loss landscape reveals masked gradients.

(2) *The Original Evaluation Wasn't Strong Enough*: The defense was only tested with a standard, off-the-shelf attack suite called AutoAttack. For a defense this complex and randomized, you need a custom, adaptive attack that specifically accounts for its tricks.

Breaking the Defense. They built a simpler, more effective attack. The key insight was to bypass the problematic parts of the defense during the attack generation. They attacked a source version of the model that replaced the tricky CrossMax aggregation with a simple average, which provided usable gradients. They used a very long 500-step gradient attack and averaged over many random transformations to smooth out the noise. Finally, they simply transferred the adversarial perturbations found on this source model to the original, defended model. The claimed robust accuracy of 72% on CIFAR-10 plummeted to just 11%. On CIFAR-100, it fell from 48% to 14%. Their adaptive attack achieved a success rate of over 85%, whereas the previously-used AutoAttack maxed out at around 52%.

2.2 Ensemble Everything Everywhere

The authors show that strong robustness can be achieved without adversarial training. Fine-tuning a pre-trained model on a multi-resolution input stack with a very small learning rate is sufficient to induce robustness. When training from scratch, techniques like mix-up further improve results. Notably, robustness peaks early during training and then decays, suggesting a trade-off between clean accuracy and adversarial robustness that can be tuned via early stopping or learning rate schedulers. The method shows favorable scaling with dataset difficulty: gains on CIFAR-100 are more pronounced than on CIFAR-10. When combined with light adversarial training, performance improves further, achieving significant improvement over previous SOTA.

Multi-Resolution Prior. Inspired by the biological functioning of the human visual system, the multi-resolution prior is a novel training approach designed to enhance a classifier's robustness without explicit adversarial training. The core idea is to present multiple versions of the same image simultaneously, each at a different resolution and then stack these versions channel-wise to create an augmented input. Alongside resolution reduction, each variant is also subjected to controlled augmentations: small random noise, spatial jitter, contrast adjustments, and color-to-grayscale shifts. This mimics how the human eye perceives objects across varying distances, blurs, and lighting conditions, especially through natural eye movements like microsaccades and macrosaccades, which constantly change the retinal image while preserving semantic content. By forcing the model to classify this multi-resolution, multi-perturbed stack as a single input, the network learns a more transformation-invariant representation of objects. This not only improves generalization but also inherently boosts adversarial robustness by reducing the effective attackable input space and making it harder for an adversary to craft perturbations that deceive all resolution layers at once. In experiments, this method yielded measurable gains in robustness on datasets like CIFAR-10 and CIFAR-100, even before any adversarial training was applied.

A central claim of the paper is the Interpretability-Robustness hypothesis: models that produce human-interpretable adversarial examples tend to be more adversarially robust. Under standard attacks, brittle models yield noise-like perturbations that fool the model but are meaningless to humans. In contrast, attacks against the author's multi-resolution models produce semantically meaningful changes. This suggests that robust representation align more closely with human visual perception, and that interpretability can serve as a proxy for robustness.

$$x_k = \text{UPSCALE}\left(\text{DOWNSCALE}(x, d_i) + \mathcal{N}(0, \delta_1^2 \cdot I_{d_i \times d_i})\right) + \mathcal{N}(0, \delta_2^2 \cdot I_{d \times d}) \tag{2.1}$$

CrossMax Robust Ensembling. When ensembling multiple neural networks, the standard approach is to average their output logits. While this improves clean accuracy and uncertainty estimation, it creates a single point of failure in adversarial settings. An attacker can deploy a targeted gradient-based attack against just one vulnerable model in the ensemble. By maximizing that single model's logit for the target class, that outlier prediction can dominate the ensemble's mean, effectively hijacking the entire decision. The authors propose: instead of using the most confident prediction, use the k^{th}-highest confidence per class.

Given logits Z of shape `[Batch, N_models, C_classes]` :

(1) *Per-Model Max Subtraction*: Compute $\hat{Z} = Z - \max(Z, \text{axis} = 2)$. This removes each model's intrinsic bias scale. A model that outputs generally higher numbers can no longer

dominate. It neutralizes attacks that work by simply boosting all logits of one model.

(2) *Per-Class Max Subtraction*: Compute $\hat{Z} = \hat{Z} - \max(\hat{Z}, \text{axis} = 1)$. This eliminates any single model's extreme confidence for any particular class. Now, for a class to win, it needs consistent high support across multiple models.

(3) *Robust Aggregation*: Finally compute $Y = \text{median}(\hat{Z}, \text{axis} = 1)$. Take the median across models for each class. The median is highly resistant to outliers. An attacker must now manipulate enough models to shift the median, not just create one extreme value.

The paper's most powerful application is creating a self-ensemble from just one model. They discovered that adversarial attacks don't affect all layers of a network equally. An image fooling the final layer might still be correctly identified by the middle or early layers. They attach simple probe classifiers to multiple intermediate layers of a single network. When an input comes in, you get multiple independent predictions. These predictions from different layers of the same model are treated as votes from a committee, and are then aggregated using the CrossMax algorithm. This single-model self-ensemble achieves significant adversarial robustness, because an attack designed to fool the final layer fails to fool the earlier layers, and CrossMax robustly combines these differing opinions.

A key empirical finding is that adversarial susceptibility is not uniform across a network's layers. Attacks crafted to fool the final classification layer do not reliably fool intermediate layers; early and middle layers often retain features of the original class. This layer-wise decorrelation enables the self-ensemble strategy: by aggregating predictions from multiple layers via CrossMax, the model can "notice" when an attack is inconsistent across its own internal representations, thereby resisting manipulation.

Generative Byproduct. An unexpected outcome of the multi-resolution prior is that it allows pre-trained classifiers and CLIP models to act as controllable image generators. By starting from uniform noise and performing gradient ascent toward a target class using the multi-resolution perturbation parameterization, the optim8izer produces human-interpretable images rather than adversarial noise. This demonstrates that the model's internal representations are semantically structured and that adversarial robustness can coincide with generative interpretability. The multi-resolution attack formulation also serves as a strong transfer prior. The authors successfully craft adversarial images that fool closed-source vision-language models (GPT-4, Gemini, Claude) by optimizing against open-source CLIP variants.