International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Adversarial Attacks and Defenses: A Survey

Racherla Swapna
Research Scholar, Department of Computer Science and Engineering,
Jawaharlal Nehru Technological University, Hyderabad, Telangana.

Dr. R. Sridevi
Professor, Department of Computer Science and Engineering,
Jawaharlal Nehru Technological University, Hyderabad, Telangana.

Abstract - Deep learning is a subset of machine learning. It uses advanced techniques such as neural networks, which have many hidden layers that mimic the human brain, and it is used in a wide range of applications. Deep learning techniques are prone to adversarial attacks. Adversarial attacks use adversarial samples, which are injected into the system to fool the model, so that the system fails to produce correct results. These samples can be injected during the training phase or the testing phase. Hence, it is crucial to provide robustness to deep learning applications against these adversaries. In this paper, we provide a detailed discussion of various types of adversarial attacks and of the defense techniques against them.

Keywords— Deep Learning, Adversarial Attacks, Adversaries.

1 INTRODUCTION

Deep learning powers a wide array of modern technological domains, ranging from visual perception tasks like image classification, object detection, and face recognition to auditory and linguistic systems including speech recognition, voice synthesis, and natural language processing. These applications rely on diverse architectures and learning paradigms—such as Deep Neural Networks (DNNs), Deep Reinforcement Learning (DRL), Federated Learning (FL), and Transfer Learning (TL) [1].Despite their remarkable advancements, many of these implementations operate in safety- and security-critical environments where operational failures carry severe consequences. Furthermore, foundational techniques across DNNs, DRL, FL, and TL remain susceptible to adversarial manipulation, representing a significant hurdle for real-world deployment. Specifically, deep networks can be deceived by carefully crafted perturbations that are virtually imperceptible to human observers. For example, slight, intentional alterations can cause autonomous vehicles, biometric scanners, and automated recognition systems to produce erroneous outputs.To mitigate these vulnerabilities, the research community actively develops robust defense methodologies aimed at safeguarding neural networks. Strategies such as adversarial training, input preprocessing, and architectural hardening are essential for maintaining model reliability and ensuring that critical systems continue to operate securely under adversarial pressure.This study surveys adversarial attacks against deep neural networks and the main defense strategies developed against them. It considers attack techniques under different threat assumptions, attacker capabilities, and operational scenarios, and it presents the formulations of representative attacks together with the training-based and denoising-based

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

defenses that respond to them. The discussion centers on image classification, where most of these methods were developed and evaluated. Attacks on deep reinforcement learning, federated learning, and transfer learning are outside its scope, and those techniques are introduced in Section 2 as background only.

This paper presents a structured survey of adversarial attacks and corresponding defense mechanisms for deep neural networks. To establish a systematic understanding of the research domain, relevant concepts, terminology, and key terms are organized into a structured taxonomy. Furthermore, adversarial threat models are classified based on their characteristics, attacker capabilities, objectives, and underlying attack settings, providing a unified framework for analyzing the security and robustness of deep learning systems. The rest of the paper is organized as follows. Section 2 gives the necessary background, Section 3 covers adversarial attacks, Section 4 covers adversarial defenses, and Section 5 concludes.

2 BACKGROUND

Taxonomy of Deep Learning Techniques and Adversarial Model:

Deep Neural Network. The concept of a deep neural network is inspired by biological neural systems. A neural network consists of an interconnected collection of artificial neurons. An individual neuron receives an input vector x through multiple incoming channels, computes an inner product with a weight vector w, and passes the result through a non-linear activation function $f(\cdot)$. The output of the neuron is given by $y = f(w^T x + b)$. A neural network is formed through the layered interconnection of these processing units.

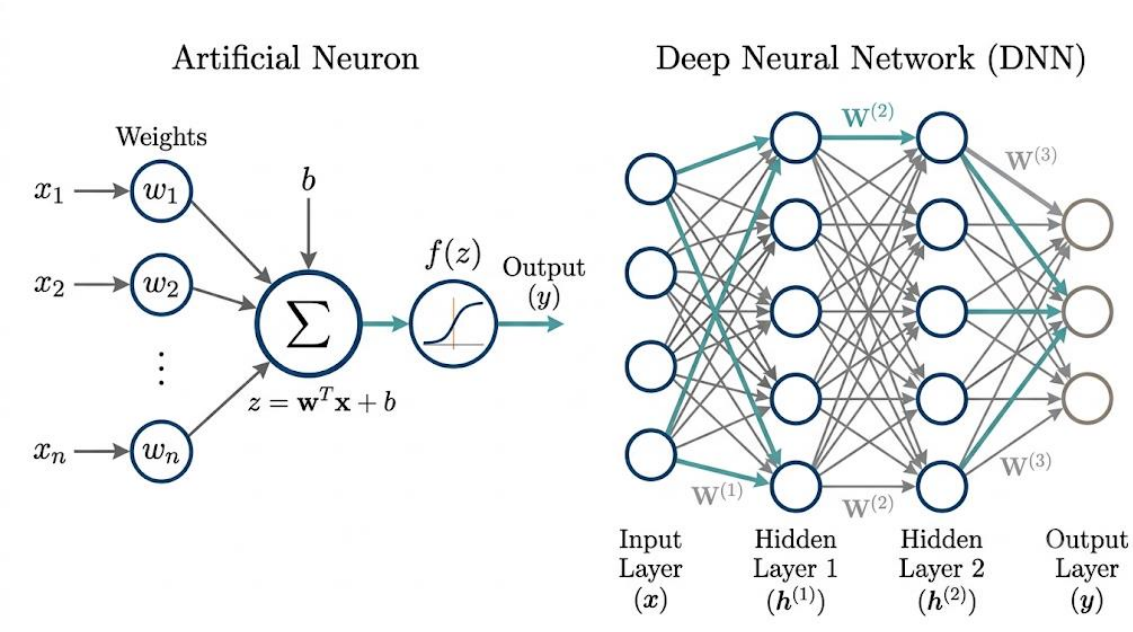


Fig. 1. Artificial Neuron and Deep Neural Network.

A deep neural network (DNN) takes raw data as input. A DNN comprises an input layer, multiple hidden layers, and an output layer; these intermediate hidden layers hierarchically extract latent features from the raw data. Through these

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

successive layer-wise transformations, the network computes complex mappings to solve challenging real-world problems.

Convolutional Neural Networks (CNNs) are primarily designed to process structured grid data, such as images. A standard CNN architecture comprises convolutional layers, pooling layers, non-linear activation functions, and fully connected layers. Convolutional layers extract hierarchical spatial features from input images, ranging from low-level edges and textures to complex semantic patterns. Pooling layers apply aggregation functions over local spatial regions to downsample feature maps and retain salient activations. Activation functions introduce non-linearity, enabling the network to model complex relationships among extracted features. Finally, one or more fully connected layers map these representations to target predictions, such as classification logits.

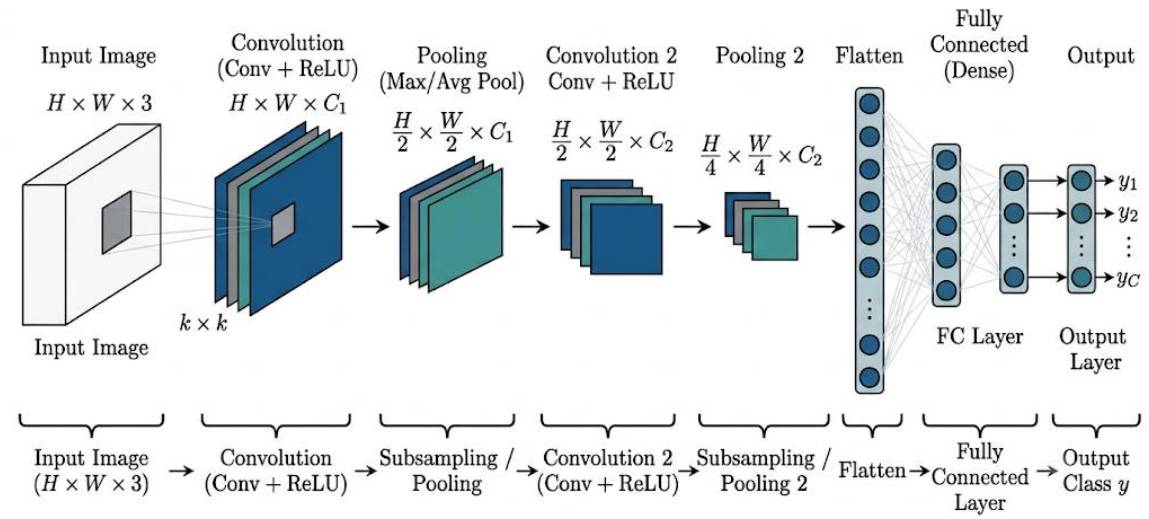


Fig. 2. Convolutional Neural Network.

Federated Learning (FL) and Transfer Learning (TL) represent two essential paradigms designed to optimize resource usage, data privacy, and model adaptability.

Federated Learning (FL): FL introduces a decentralized training framework wherein distributed participants—such as edge devices or independent institutions—jointly optimize a shared global model coordinated by a central aggregator. By keeping sensitive raw data locally on client hardware, FL adheres to strict data minimization standards. This decentralized approach effectively addresses major privacy concerns and logistical overheads inherent in conventional centralized data aggregation pipelines [2].

Transfer Learning (TL): TL is a methodology focused on knowledge reuse, where insights gained from solving a source task are applied to enhance learning efficiency on a distinct yet related target problem. Rather than initiating training from random initialization, practitioners adapt pre-trained parameters or feature embeddings to the new domain. This strategy proves especially advantageous in data-constrained scenarios, simultaneously lowering computational

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

resource demands, accelerating convergence speed, and boosting overall generalization capabilities by leveraging universally applicable feature representations.

Deep Reinforcement Learning (DRL). Deep Reinforcement Learning (DRL) is an AI technique that combines reinforcement learning with deep neural networks. It learns to take actions based on feedback in the form of a reward or a penalty signal. This feedback about reward and penalty defines the policy that maximizes the reward function. This policy is represented by a neural network whose input is the current state and output is the probability distribution of possible actions. Reinforcement learning is inherently well suited to applications without a training dataset. In deep reinforcement learning, the decision is made based on feedback from the environment.

3 ADVERSARIAL ATTACKS

An adversarial attack on deep learning is a technique designed to fool a neural network into making incorrect predictions by applying small, intentionally crafted perturbations to the input data. These attacks significantly decrease the model's performance. Adversarial attacks are intended to undermine the reliability of the model. Adversarial attacks can be systematically categorized along multiple foundational axes, including the adversary's knowledge, operational objectives, and execution timing.

In adversarial machine learning, a threat model defines the precise boundaries, assumptions, and capabilities governing the adversary's interaction with the target system. Standardizing the threat model is critical to ensuring sound evaluation integrity and avoiding artificial claims of robustness [3].

Formally, an adversarial threat model is parameterized along four orthogonal operational axes: Adversary Knowledge, Adversary Goals, Adversary Capabilities and Perturbation Budgets, and Attack Timing.

3.1 Adversary's Goal

Based on the adversary's operational objective, adversarial attacks are broadly classified into confidence reduction, untargeted misclassification, targeted misclassification, and misclassification.

Untargeted Attacks. The objective of an untargeted attack is to force the model into any incorrect classification, regardless of the resulting output label. For instance, given an image of a cat, the adversary introduces a subtle perturbation such that the model fails to predict the class 'cat' and instead outputs an arbitrary incorrect class.

Targeted Attacks. Targeted attacks aim to force the model to misclassify an input into a specific, predetermined target class different from its true label. For example, the adversary crafts a perturbation such that an image of a cat is predicted not merely as an arbitrary wrong class, but explicitly as a chosen label, such as 'dog'.

Confidence Reduction. Confidence reduction attacks are adversarial perturbations engineered to suppress the prediction confidence of deployed deep learning systems. For instance, in image classification, an adversary crafts subtle input perturbations that drive down the confidence score of the true class or distribute probability mass across alternative categories, significantly impairing the model's decision certainty.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Misclassification. Misclassification attacks are adversarial perturbations engineered to push input representations across a model's learned decision boundaries, thereby compelling the classifier to assign incorrect output labels to legitimate samples. The objective is to alter the model's prediction to any incorrect class distinct from the original label.

3.2 Based on Distance Metrics

In this category of adversarial attacks, the perturbations introduced into the input data are constrained to remain as small as possible while still causing the model to produce incorrect predictions. The magnitude of these perturbations is quantified using different L_p norms, such as the L_0, L_2, and L_∞ norms. These metrics measure different aspects of the perturbation, including the number of modified features, the overall magnitude of the changes, and the maximum change applied to any individual feature, respectively.

L_0 norm. The L_0 norm quantifies sparsity by measuring the total number of input features or pixels altered between the clean and adversarial samples, regardless of the magnitude of each change.

L_2 norm. The L_2 norm quantifies the Euclidean distance between the clean and adversarial samples by measuring the square root of the sum of squared differences across all pixel values.

L_∞ norm. Under the L_∞ norm, the perturbation budget is determined by the maximum absolute deviation across all pixels between the original input and its adversarial counterpart.

3.3 Adversary's Timing

3.3.1 Training-time adversarial capabilities

Attacks that exploit the training phase, commonly termed causative attacks, are distinguished by the degree of control an adversary exerts over the learning pipeline [4, 5]. Following the capability-based taxonomy of Papernot et al. [6], three levels are generally recognized, in order of increasing strength.

Data Injection. At the weakest level, an adversary with data injection capability can contribute new samples to the training corpus but has no means of modifying or removing existing records. This situation arises in systems that continually retrain on user-contributed or crawled data, where an attacker may submit maliciously crafted inputs through legitimate channels [7].

Data Modification. A stronger adversary possesses data modification capability, meaning direct write access to the existing training data or its annotations. Such an attacker may corrupt labels, perturb feature values, or otherwise alter records already in the dataset, which affords finer control over the resulting decision boundary than injection alone [8, 9].

Logic Corruption. The strongest threat model assumes logic corruption, in which the adversary can interfere with the learning procedure itself, for example by tampering with the training code, the optimization process, or the model parameters. This capability is most relevant to supply-chain settings, such as the use of third-party pretrained models or outsourced training, where the integrity of the learning process cannot be verified by the end user [6, 10].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Poisoning attacks (Training Time). Poisoning attacks are executed during the training phase of machine learning models to corrupt the underlying learning process. In this threat model, the adversary injects carefully crafted malicious samples into the training dataset, thereby manipulating the learned parameter set and degrading the generalization performance or integrity of the resulting classifier.

Evasion attacks (Testing Time). Evasion attacks represent the most prevalent adversarial attack vector and are executed strictly during the inference (testing) phase. In this regime, the adversary crafts deceptive input samples designed to mislead the target model into producing erroneous outputs at runtime. Crucially, the adversary leaves the model parameters, architecture, and underlying training pipeline entirely unaltered, exploiting pre-existing vulnerabilities in the learned decision boundaries.

3.4 Adversary's Knowledge

3.4.1 White Box

In a white-box attack, the adversary possesses complete knowledge of the target system, including the network architecture, trained weights, and training hyperparameters. By exploiting direct access to the model's loss gradients, the attacker crafts subtle perturbations to input samples that reliably force the classifier into producing erroneous outputs.

Fast Gradient Sign Method (FGSM). FGSM generates an adversarial image by adding a pixel-wise perturbation of magnitude ϵ in the direction of the sign of the loss gradient with respect to the input [11]. Operating as a single-step technique, FGSM achieves high computational efficiency by requiring only a single backward pass to craft the perturbed input.

Method. Let x be a clean input with true label y, θ the model parameters, and $J(\theta, x, y)$ the training loss. FGSM takes a single step in the direction that increases the loss, using only the sign of the gradient:

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

Here ϵ controls the perturbation magnitude. Because the sign operation gives every input dimension a change of exactly $\pm\epsilon$, the perturbation satisfies an L_∞ constraint, $\|x_{adv} - x\|_\infty \leq \epsilon$ [11]. The adversarial example is then typically clipped to the valid input range (e.g., [0,1] for images).

Key properties.

- **Efficiency.** Only one gradient computation (one backward pass) is needed, making FGSM much cheaper than iterative optimization-based attacks such as the L-BFGS formulation in [12].
- **White-box requirement.** The attacker needs access to the model's gradients, so FGSM is a white-box attack.
- **Untargeted by default.** The basic form maximizes the loss on the true label. A targeted variant instead minimizes the loss with respect to a chosen target label, taking a step in the negative gradient direction.
- **Transferability.** Adversarial examples crafted on one model often fool others, which enables black-box attacks using a surrogate model [11, 13].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

- **Use in defense.** Goodfellow et al. also proposed using FGSM examples during training (adversarial training) as a regularizer to improve robustness [11].

For a targeted attack, the same method is applied, with the key difference being that the perturbation is generated in the direction of the negative gradient of the loss with respect to the target class, encouraging the model to classify the input as the desired target class [14]:

$$x_{adv} = x - \epsilon \cdot sign(\nabla_x J(\theta, x, y_t))$$

where:

- x_{adv} — generated adversarial example.
- x — original input.
- ϵ — perturbation magnitude.
- $J(\theta, x, y_t)$ — loss function evaluated with respect to the target class y_t.
- $\nabla_x J(\theta, x, y_t)$ — gradient of the loss with respect to the input.
- $sign(\cdot)$ — sign function applied to each gradient component.
- y_t — desired target class.

Projected Gradient Descent (PGD). PGD is one of the most popular and widely used white-box attacks [15]. It is an iterative extension of FGSM: at each step, the input is perturbed in the direction of the sign of the loss gradient, and the result is then projected back onto the allowed perturbation set (e.g., an L_∞ ball of radius ϵ around the original input) so that the perturbation budget is never exceeded [15, 16]. PGD is typically initialized at a random point within this ball. Because it is not restricted to a single step, it can spend as much computation as needed, and it is therefore considered a much stronger attack than FGSM. The process is repeated until the attack succeeds (i.e., the model's predicted class changes) or a fixed number of iterations is reached.

PGD update equation. For an untargeted PGD attack under an L_∞ constraint, the iterative update is expressed as:

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

where:

- x is the original, clean input.
- $x^{(t)}$ is the adversarial input at iteration t.
- $x^{(t+1)}$ is the updated adversarial input.
- α is the step size used at each iteration.
- $J(\theta, x^{(t)}, y)$ is the loss function evaluated at the current input.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

- $\nabla_{x^{(t)}} J$ is the gradient of the loss with respect to the current input.
- Π_{x+S} denotes projection onto the allowed perturbation region centered at the original input.
- $sign(\cdot)$ determines the direction of each input update.

The equation above follows the standard L_∞-constrained PGD formulation described by Madry et al. [15].

DeepFool. DeepFool is an untargeted, iterative white-box attack proposed by Moosavi-Dezfooli et al. [17] that generates minimal adversarial perturbations based on the geometry of the classifier's decision boundaries. Instead of maximizing loss within a pre-allocated perturbation budget like FGSM or PGD, DeepFool computes the minimal perturbation required to push an input across the closest decision boundary.

To formalize this concept, consider an affine binary classifier:

$$f(x) = w^T x + b$$

where the decision boundary corresponds to the hyperplane $F = \{x: w^T x + b = 0\}$. The exact minimal perturbation $\delta^*(x)$ required to alter the classification corresponds to the orthogonal projection of x onto F:

$$\delta^*(x) = -\frac{f(x)}{\|w\|_2^2} w$$

Because deep neural networks are non-linear, their decision boundaries are represented by complex, non-linear hypersurfaces. DeepFool addresses this by iteratively linearizing the boundary around the current iterate x_i using a first-order Taylor approximation:

$$f(x) \approx f(x_i) + \nabla f(x_i)^T (x - x_i)$$

In a multi-class setting with C classes, the algorithm computes the projection onto the closest separating hyperplane among all alternative classes $k \neq \hat{k}(x)$.

The perturbation vector is accumulated iteratively until the input crosses the boundary, yielding tighter L_2 and L_∞ perturbation bounds and serving as an accurate empirical proxy for classifier robustness [17].

Carlini and Wagner (C&W) Attacks. Carlini and Wagner introduced a suite of optimization-based adversarial attacks designed to evaluate the limits of adversarial robustness [18]. Specifically developed to defeat defensive distillation [19] and other heuristic defenses that rely on gradient masking, the C&W framework formulates adversarial example generation as a constrained, continuous optimization problem rather than a greedy gradient-ascent heuristic.

Optimization Formulation. Given an input $x \in R^d$, the objective is to synthesize a perturbation δ of minimum norm that forces a neural network $f(\cdot)$ to predict a chosen target class $t \neq y$, while satisfying the box constraint that the perturbed sample remains a valid image ($x + \delta \in [0,1]^d$). Carlini and Wagner expressed this constrained problem as:

$$\min_{\delta} \|\delta\|_p + c \cdot g(x + \delta) \quad subject\ to \quad x + \delta \in [0,1]^d$$

where:

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

- $\| \cdot \|_p$ denotes the chosen L_p distance metric, predominantly instantiated under the L_0, L_2, or L_∞ norm.
- $c > 0$ is a dynamically tuned regularization constant determined via binary search.
- $g(\cdot)$ is an objective loss function engineered such that $g(x + \delta) \leq 0$ if and only if the model predicts the target class t.

Logit-Based Objective Function (g). To prevent the vanishing gradient issues caused by standard saturated softmax cross-entropy losses—which defensive distillation inherently exploits—Carlini and Wagner evaluated several candidate objective functions and identified the logit-difference loss as the most effective:

$$g(x') = max\left(\underset{i \neq t}{max} Z(x')_i - Z(x')_t,\ -\kappa\right)$$

where:

- $Z(x')_i$ represents the unnormalized logit score corresponding to class i prior to the softmax activation layer.
- $\kappa \geq 0$ denotes a confidence parameter. Setting $\kappa = 0$ yields a minimal perturbation that marginally flips the prediction, whereas $\kappa > 0$ crafts high-confidence adversarial examples that exhibit enhanced transferability across distinct architectures.

Box-Constraint Handling: Change of Variables. To eliminate the need for projected gradient descent or heuristic clipping when enforcing $x + \delta \in [0,1]^d$, C&W introduced a change-of-variables formulation using the hyperbolic tangent function:

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

Under this transformation, any unconstrained real-valued parameter vector $w \in R^d$ inherently satisfies:

$$0 \leq x_i + \delta_i = \frac{1}{2}(tanh(w_i) + 1) \leq 1$$

This allows the optimization problem to be solved smoothly using standard unconstrained gradient-based optimizers, such as Adam [20].

Attack Variants Across Norms. Carlini and Wagner instantiated their framework across three standard distance metrics:

- L_2 **Attack:** Optimizes $\|\delta\|_2^2 + c \cdot g(x + \delta)$ directly using the Adam optimizer, generating perturbations with significantly lower Euclidean distortion than FGSM or DeepFool.
- L_∞ **Attack:** Employs an iterative penalty method to minimize $max_i|\delta_i|$, penalizing only the dimensions that exceed a progressively decaying threshold τ.
- L_0 **Attack:** An iterative greedy algorithm that uses the L_2 attack to identify non-critical coordinates, iteratively masking out pixels that contribute least to the adversarial objective until a minimal subset remains.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Jacobian-based Saliency Map Attack (JSMA). Introduced by Papernot et al. [21], JSMA is a targeted, white-box attack optimized under the L_0 norm to alter a minimal number of input features. Rather than computing loss gradients via backpropagation, JSMA evaluates the forward derivative of the network $F(x)$ via its Jacobian matrix $J_F(x) \in R^{K \times d}$:

$$J_F(x) = \left[\frac{\partial F_j(x)}{\partial x_i}\right]_{j \in \{1,\ldots,K\},\ i \in \{1,\ldots,d\}}$$

Saliency Map and Feature Selection. To drive the network toward a designated target class t, an adversarial saliency map $S(x,t)$ scores each feature i based on its ability to increase the target probability while suppressing competing outputs:

$$S(x,t)[i] = \{0, if\ \frac{\partial F_t(x)}{\partial x_i} < 0\ \ or\ \sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i} > 0\ \frac{\partial F_t(x)}{\partial x_i} \left| \sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i} \right|, otherwise$$

The algorithm iteratively selects the highest-scoring pixel pair, saturating their values until $arg\,max_j F_j(x_{adv}) = t$ or the sparsity budget $\|\delta\|_0 \leq \gamma \cdot d$ is reached. Due to its $O(d^2)$ pairwise Jacobian complexity, JSMA scales poorly to high-dimensional datasets [6, 21].

Universal Adversarial Perturbations (UAP). Proposed by Moosavi-Dezfooli et al. [22], a Universal Adversarial Perturbation (UAP) is a single, input-agnostic noise vector $v \in R^d$ that misleads a classifier across almost all inputs sampled from a data distribution μ.

Mathematical Formulation. The adversary seeks a single perturbation v bounded by an L_p budget ξ that achieves a target fooling rate $1 - \delta$:

$$\|v\|_p \leq \xi \quad such\ that \quad P_{x \sim \mu}\left(\hat{k}(x+v) \neq \hat{k}(x)\right) \geq 1 - \delta$$

where $\hat{k}(x)$ denotes the model's predicted label.

Algorithmic Mechanism. Initialized at $v = 0$, the algorithm iteratively loops over training samples $x_i \in X$:

1. If $\hat{k}(x_i + v) = \hat{k}(x_i)$, it computes the minimal additional perturbation Δv_i required to cross the decision boundary (typically via DeepFool [17]):

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

2. The vector is updated and projected back into the L_p ball of radius ξ:

$$v \leftarrow \Pi_{p,\xi}(v + \Delta v_i)$$

This loop repeats until the empirical fooling rate on X meets or exceeds $1 - \delta$. UAPs generalize well across training subsets and exhibit strong cross-architecture transferability [22].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

3.4.2 Black Box: Operational Taxonomy of Black-Box Attacks

In adversarial machine learning, black-box threat models are defined along two primary axes: the temporal nature of adversary-target interaction (open-loop vs. closed-loop) and the degree of defense awareness and adaptability exhibited by the adversary [23, 3]. While early surveys categorized black-box attacks primarily by output granularity (score-based vs. decision-based) [23], practical security audits require a taxonomy organized by operational dynamics.

We formalize three operational regimes: Non-Adaptive Black-Box, Adaptive Black-Box, and Strict Black-Box (No-Box) attacks.

Comparative Operational Taxonomy.

Table 1. Comparative operational taxonomy of black-box attacks.

Attack Paradigm	Target Interaction & Query Budget	Defense Awareness	Optimization Mechanism	Canonical References
Non-Adaptive Black-Box	Open-loop; zero online queries ($Q = 0$)	Oblivious to runtime defenses	Transferability via surrogate landscapes; input transformations; momentum	Papernot et al. [24], Dong et al. [25], Xie et al. [26]
Adaptive Black-Box	Closed-loop; sequential queries ($Q \in [10^2, 10^5]$)	Actively circumvents runtime mitigations	Zeroth-order gradient estimation (NES); boundary tracking; score-based random search	Carlini et al. [3], Athalye et al. [27], Andriushchenko et al. [28]
Strict Black-Box (No-Box)	Extreme query constraints ($Q \to 0$); hard labels	Zero parameter, architectural, or distribution knowledge	Synthetic dataset distillation; unsupervised proxy surrogates; topological graph probing	Gao et al. [29], Li et al. [30]

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Non-Adaptive Black-Box Attacks. A non-adaptive black-box attack executes under an open-loop regime: candidate perturbations are generated entirely offline without runtime feedback or iterative refinement against the victim model f_θ [24].

Operational Characteristics.

- **Surrogate-Driven Optimization:** The adversary optimizes a bounded perturbation δ against a locally trained substitute model $\hat{f}_\Phi$ (or an ensemble of heterogeneous architectures $\sum_{m=1}^{M} w_m \hat{f}_{\Phi_m}$):

$$\delta^* = arg\, max_{\|\delta\|_p \le \epsilon} L\big(\hat{f}_\Phi(x+\delta), y\big)$$

  The resulting adversarial sample $x_{adv} = x + \delta^*$ is submitted directly to the target system f_θ under the assumption of cross-model transferability [24, 25].

- **Landscape Stabilization:** Because the adversary cannot verify whether an attack settles into narrow, surrogate-specific loss minima, non-adaptive attacks apply momentum accumulation to stabilize update trajectories across iterations. Dong et al. [25] introduced the Momentum Iterative Fast Gradient Sign Method (MI-FGSM):

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x L(\hat{f}_\Phi(x_t), y)}{\|\nabla_x L(\hat{f}_\Phi(x_t), y)\|_1}, \qquad x_{t+1} = \Pi_{x+S}(x_t + \alpha \cdot sign(g_{t+1}))$$

- **Input Diversity Transformations:** To prevent spatial overfitting to the surrogate architecture, Xie et al. [26] introduced the Diverse Inputs Method (DIM), which applies stochastic image transformations $T(x; p)$ (random resizing and padding with probability p) during backpropagation:

$$\delta^* = arg\, max_{\|\delta\|_p \le \epsilon} E_T\big[L\big(\hat{f}_\Phi(T(x+\delta)), y\big)\big]$$

- **Absence of Runtime Compensation:** Non-adaptive perturbations cannot dynamically compensate for non-deterministic or reactive defense mechanisms (e.g., stochastic input transformation, feature quantization, or test-time noise injection) deployed by the target model.

Adaptive Black-Box Attacks. In an adaptive black-box attack, the adversary interacts with the victim in a closed-loop feedback system. The perturbation evolves sequentially based on observed oracle responses, and the optimization strategy explicitly accounts for defensive countermeasures deployed on the target model [3, 27].

Operational Characteristics.

- **Interactive Query Feedback:** The attacker formulates subsequent input queries conditioned on previous model outputs:

$$x^{(t+1)} = x^{(t)} + u(\{x^{(\tau)}, f_\theta(x^{(\tau)})\}_{\tau=1}^{t}), \quad s.t. \quad \|x^{(t+1)} - x\|_p \le \epsilon$$

  where $u(\cdot)$ represents a zeroth-order update rule, evolutionary strategy, or geometric boundary adjustment.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

- **Circumvention of Obfuscated Gradients:** Athalye et al. [27] and Carlini et al. [3] demonstrated that defenses relying on gradient masking (shattered gradients, stochastic transformations, or numerical saturation) give a false sense of security. Adaptive adversaries tailor their probing mechanism to bypass these defenses:
    - **Stochastic Defenses:** Circumvented by combining Natural Evolution Strategies (NES) with Expectation Over Transformation (EOT), taking expectations over runtime defense transformations:

$$\nabla_x E_{t\sim T}[L(f_\theta(t(x)), y)] \approx \frac{1}{B\sigma}\sum_{i=1}^{B} \delta_i \, L(f_\theta(t(x+\delta_i)), y)$$

    - **Shattered / Discontinuous Surfaces:** Mitigated by gradient-free exploration methods. Andriushchenko et al. [28] designed Square Attack, a score-based search algorithm that updates localized, square-shaped coordinate slices via randomized sampling:

$$x^{(t+1)} = \Pi_{x+S}\left(x^{(t)} + \Delta^{(t)}\right)$$

- Because Square Attack does not rely on local loss gradients or their numerical approximations, it renders gradient obfuscation defenses ineffective [3, 28].
- **High Query Consumption:** Adaptive optimization requires query budgets typically ranging from $Q \sim 10^2$ (for random-search score attacks [28]) to $Q \sim 10^5$ (for hard-label decision-boundary tracking).

Strict Black-Box Attacks (No-Box / Zero-Knowledge). A strict black-box attack (often designated as a No-Box or Zero-Knowledge threat model [29, 30]) represents the most constrained operational setting. It strips away conventional academic relaxations—such as access to identical training data distributions, unlimited query budgets, and continuous logit vectors.

Operational Characteristics.

- **Severe Query and Interface Constraints ($Q \to 0$):** The adversary faces aggressive query rate-limiting, CAPTCHAs, or active stateful query analyzers that prevent the multi-thousand query iterations required by zeroth-order methods [3, 28]. The model provides only hard, discrete top-1 predictions:

$$\hat{y} = arg\, max_{c\in\{1,\ldots,K\}} f_\theta(x)_c$$

- **Distributional and Architectural Disparity:** Unlike the assumptions in Papernot et al. [24], the adversary possesses neither the target model's training dataset D_{train} nor architectural knowledge (e.g., whether the victim is a ResNet, a Vision Transformer, or an ensemble). Li et al. [30] formalized practical no-box attacks where the attacker trains generative networks or surrogate models using uncurated, out-of-domain public data or self-supervised feature reconstructors without target feedback.
- **Non-Euclidean and Domain Extensions:** Strict black-box threat models extend beyond Euclidean continuous manifolds into discrete, structured topologies. Gao et al. [29] demonstrated strict black-box adversarial attacks

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

on Graph Neural Networks (GNNs), where attackers modify graph structures (node features and edge connectivity) with zero victim query feedback and restricted topological visibility:

$$\max_{A' \in V_{budget}} L_{GNN}(f_\theta(A', X), Y)$$

These results show that topological vulnerabilities can be exploited without model access or interactive feedback.

4 ADVERSARIAL DEFENSES

Defenses against evasion attacks fall into two broad groups. The first changes how the classifier is trained, so that its decision boundary is harder to cross. The second places a component in front of the classifier, or inside it, whose job is to remove the adversarial part of an input. This section describes representative methods from each group. Whichever group a method belongs to, the protection it offers holds only for the threat model it was evaluated under, so claimed robustness has to be re-tested against adaptive attacks [3, 27].

4.1 Adversarial Training

Adversarial training feeds the model adversarial examples produced against its current parameters, so that it learns to classify correctly the very inputs an attacker would submit [12, 11]. In its general form it is a saddle-point problem [15]:

$$\min_\theta E_{(x,y)\sim D}\left[\max_{\|\delta\|_p \le \epsilon} J(\theta, x+\delta, y)\right]$$

The inner maximization plays the attacker, and the outer minimization is ordinary training on the attacker's output. The variants below differ mainly in how the inner problem is solved, which determines both their cost and their strength.

4.1.1 FGSM Adversarial Training

The original proposal solves the inner problem with a single FGSM step, because one extra gradient evaluation per example keeps training close to the cost of standard training [11]. The training loss mixes the clean and the perturbed objective:

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J(\theta,\ x + \epsilon \cdot sign(\nabla_x J(\theta, x, y)),\ y)$$

where an equal weighting, $\alpha = 0.5$, was used in the original work. Single-step training has two known weaknesses. First, because the perturbation is computed from the true label, the network can learn to recognize the perturbation pattern and read the label from it. This effect, called label leaking, can make accuracy on FGSM examples exceed clean accuracy without improving robustness to other attacks [31]. Second, training can suffer catastrophic overfitting, in which robustness to multi-step attacks collapses abruptly while accuracy on FGSM examples stays high. Starting the single step from a random point and choosing the step size with care avoids this and makes single-step training competitive [32].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

4.1.2 PGD Adversarial Training

PGD adversarial training replaces the single step by the iterative attack described in Section 3.4.1: starting from a random point in the ϵ-ball, the inner loop takes several signed-gradient steps and projects back onto the ball after each one [15]. Treating PGD as a strong first-order adversary turns training into a robust-optimization problem, and networks trained this way resist a wide range of first-order attacks within the chosen budget, which made the method the usual baseline for later work [15]. The price is computational: an update needs roughly as many forward–backward passes as the inner loop has steps, plus one, so training is several times slower than FGSM training. Robust models also tend to lose some accuracy on clean inputs and benefit from higher network capacity [15].

4.1.3 Ensemble Adversarial Training

Single-step training on a model's own gradients can lead the model to distort its loss surface near the data points. The linear approximation behind FGSM then becomes misleading, so the model looks robust to the attack it trains against, yet it remains weak against perturbations transferred from other models and against a one-step attack that begins with a small random move [33]. Ensemble adversarial training breaks this dependence between the perturbation and the model being trained. At each iteration the adversarial examples are crafted on a model drawn from a pool made up of the current network and several fixed, pre-trained networks, so the training perturbations no longer depend only on the current weights [33]. The method improves robustness to black-box attacks built from transferred examples. It does not claim protection against an iterative white-box adversary, for which PGD training remains the stronger option.

4.1.4 Generative Adversarial Training

A generative adversarial network (GAN) pits a generator against a discriminator in a two-player game [34]. Two defenses borrow this idea. In the generative adversarial trainer, a generator learns to produce perturbations that fool the classifier while the classifier is trained to withstand them, so the attack distribution is learned instead of being fixed by hand, and new perturbations are cheap to produce once the generator is trained [35]. Defense-GAN takes a different route: it trains a GAN on clean data only and, at test time, replaces the input x by the generator output $G(z^*)$, where

$$z^* = arg\,min_z \, \|G(z) - x\|_2^2$$

is found by a few gradient steps from several random starting points [36]. The classifier then sees only points close to the learned data manifold. Because the procedure does not depend on a particular attack, it can protect an unmodified classifier, but it adds an optimization to every query, and its quality depends on how well the generator covers the data. Since the projection is itself an optimization, a fair evaluation needs an attacker that differentiates through it or approximates it [27].

4.2 Denoising

Denoising defenses treat the adversarial perturbation as noise to be removed before classification, or suppressed inside the network. Early work placed a denoising autoencoder in front of the classifier. The authors found that the combined network could still be attacked end to end, and proposed penalizing the network's sensitivity to small input changes

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

instead [37]. A later pixel-space denoiser, the high-level representation guided denoiser, observed that a small residual error left by the denoiser is amplified as it passes through the layers. It is therefore trained with a loss on the difference between the target network's high-level features for the clean image and for the denoised adversarial image, not on pixel error [38]. Feature denoising moves the operation inside the classifier: denoising blocks, such as non-local means, are inserted into intermediate feature maps and trained jointly with adversarial training, which suppresses the perturbation's effect at the feature level [39]. As with other pre-processing defenses, the reported robustness of a denoiser should be measured with an attacker who includes it in the gradient computation or approximates it [27].

5 CONCLUSION

This paper surveyed adversarial attacks on deep neural networks and the defenses developed against them. On the attack side, we organized the literature by the adversary's goal, the distance metric that bounds the perturbation, the timing of the attack, and the adversary's knowledge. Within that framework we covered white-box attacks (FGSM, PGD, DeepFool, the Carlini and Wagner attacks, JSMA and universal perturbations) and black-box attacks, which we divided into non-adaptive, adaptive and strict no-box regimes according to how the adversary interacts with the target. We also covered training-time attacks, namely data injection, data modification and logic corruption, alongside evasion attacks at test time.

On the defense side, we described adversarial training in its FGSM, PGD, ensemble and generative forms, together with denoising defenses that act on inputs or on intermediate features. The methods trade protection against cost: single-step training is cheap but can overfit to the attack it uses, PGD training is stronger but several times slower and often lowers clean accuracy, and pre-processing defenses can be defeated by an attacker who accounts for them. The recurring lesson is that a defense is only as credible as its evaluation. Robustness claims should state the threat model and be tested against adaptive attacks that know the defense.

Open problems follow from these limits: making robust training affordable at scale, extending robustness beyond the perturbation budgets used in training, and building evaluation practices that do not overstate protection. We hope the taxonomy given here helps readers place new attacks and defenses within a common framework.

REFERENCES

[1] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019, doi: 10.1109/TNNLS.2018.2886017.
[2] P. Kairouz et al., "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, nos. 1–2, pp. 1–210, 2021, doi: 10.1561/2200000083.
[3] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," arXiv preprint arXiv:1902.06705, 2019.
[4] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proc. 2006 ACM Symp. on Information, Computer and Communications Security (ASIACCS '06)*, Taipei, Taiwan, Mar. 2006, pp. 16–25, doi: 10.1145/1128817.1128824.
[5] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Secur. Artif. Intell. (AISec)*, 2011, pp. 43–58.
[6] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "SoK: Security and privacy in machine learning," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, 2018, pp. 399–414.
[7] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.
[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, 2012.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

[9] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, vol. 160, pp. 53–62, Jul. 2015.
[10] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," arXiv:1708.06733, 2017.
[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015. [Online]. Available: https://arxiv.org/abs/1412.6572
[12] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014. arXiv:1312.6199.
[13] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," arXiv:1605.07277, 2016.
[14] Y. Li and Y. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, Mar. 2020, doi: 10.1016/j.eng.2019.12.012.
[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1706.06083
[16] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Represent. (ICLR) Workshop*, 2017. [Online]. Available: https://arxiv.org/abs/1607.02533
[17] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 2574–2582.
[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, 2017, pp. 39–57.
[19] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, 2016, pp. 582–597.
[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.
[21] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. 1st IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, 2016, pp. 372–387.
[22] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1765–1773.
[23] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," arXiv preprint arXiv:1810.00069, 2018.
[24] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIACCS)*, 2017, pp. 506–519.
[25] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 9185–9193.
[26] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 2730–2739.
[27] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 274–283.
[28] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square Attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 484–501.
[29] J. Gao, M. Ding, B. Du, and H. Shen, "Strict black-box adversarial attacks on graphs," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 36, no. 1, 2022, pp. 62–70.
[30] Q. Li, Y. Guo, H. Zha, and H. Chen, "Practical no-box adversarial attacks against deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2020.
[31] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1611.01236
[32] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020. [Online]. Available: https://arxiv.org/abs/2001.03994
[33] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1705.07204
[34] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 27, 2014, pp. 2672–2680.
[35] H. Lee, S. Han, and J. Lee, "Generative adversarial trainer: Defense to adversarial perturbations with GAN," arXiv:1705.03387, 2017.
[36] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1805.06605
[37] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR) Workshop*, 2015. [Online]. Available: https://arxiv.org/abs/1412.5068
[38] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 1778–1787.
[39] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 501–509.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)