Adversarial Attacks on Neural Networks: A Survey

Izhar Ali
Rowan University
Department of Computer Science
izharali.skt@gmail.com

This paper presents a systematic survey and empirical comparison of six foundational adversarial attack strategies against convolutional neural networks (CNNs): Fast Gradient Sign Method (FGSM), Fast FGSM (FFGSM), DeepFool, Carlini & Wagner (C&W), Projected Gradient Descent (PGD), and Conjugate Gradient (CG). We frame attack generation as a constrained optimization problem under ℓ_p norms, and evaluate these approaches across five CNN architectures. Our analysis quantifies the fundamental trade-offs governing attack effectiveness (success rate), perturbation efficiency (ℓ_p magnitude, perceptual similarity), and computational requirements. Experiments reveal that computationally inexpensive gradient-sign methods (FGSM, FFGSM) achieve speed at the cost of lower success rates and larger perturbations, particularly for targeted attacks. Conversely, iterative optimization methods (C&W, DeepFool) yield highly effective attacks with minimal perturbations but incur high computational overhead. Iterative first-order methods (PGD) and the Conjugate Gradient (CG) method balance cost and potency. Our results provide a quantitative basis for understanding and selecting attack methods based on strategic objectives and resource constraints.

Keywords: adversarial attacks, optimization methods, computer vision

1 Introduction

Convolutional Neural Networks (CNNs) have demonstrated remarkable efficacy in computer vision tasks, achieving high classification accuracy on benchmarks like ImageNet [2, 7] and enabling critical applications from autonomous systems [4] to medical imaging [9]. Despite their impressive performance, CNNs remain vulnerable to adversarial attacks: *imperceptible input perturbations that induce misclassification with high confidence*. As first shown by Szegedy et al. [14], minimal, often imperceptible, perturbations δ applied to an input x can induce a trained network f to misclassify $x+\delta$ with high confidence [3, 1]. These perturbations are typically constrained by an ℓ_p norm ($\|\delta\|_p \leq \varepsilon$) to approximate perceptual similarity.

This vulnerability stems from the network's learning process and high-dimensional nature of input data. Networks often rely on predictive yet non-robust features $\phi(x)$—i.e., features highly correlated with the correct label on the training distribution but brittle under small input changes. Mathematically, this manifests in two ways. First, the decision boundaries $\mathscr{H}$ separating classes in the high-dimensional input space $\mathscr{X}$ frequently lie close to the data points themselves. Specifically, for many inputs $x \in \mathscr{X}$, the minimum ℓ_p distance to any point x' on the nearest decision boundary $\mathscr{H}$ is small [6, 14]:

$$\min_{x' \in \mathscr{H}} \|x' - x\|_p \leq \varepsilon, \quad \text{for many } x \in \mathscr{X} \tag{1}$$

where ε represents this small distance threshold. This proximity implies inherent vulnerability, as only a small perturbation is needed to push x across the boundary.

Second, the piecewise linear activation functions common in CNNs create locally linear mappings. While globally complex, the network's behavior around an input x can be approximated via its Jacobian $J(x)$:

$$f(x+\delta) \approx f(x) + J(x)\delta \tag{2}$$

The Jacobian $J(x)$ can possess large singular values, implying that perturbations δ aligned with corresponding singular vectors can be significantly amplified, causing substantial shifts in the output even for small $\|\delta\|_p$. Attack algorithms seek these amplification directions.

Consequently, generating adversarial examples is naturally framed as a constrained optimization problem: find a minimal perturbation δ that causes misclassification, subject to $\|\delta\|_p \leq \varepsilon$.

$$\min_{\delta} \|\delta\|_p \quad \text{s.t.} \quad f(x+\delta) \neq f(x), \quad \|\delta\|_p \leq \varepsilon \tag{3}$$

The strategy chosen to solve this optimization—from simple gradient ascent heuristics to more sophisticated iterative methods—influences the resulting perturbation's magnitude, the attack's success rate, and the computational resources consumed. This paper provides a systematic, empirical comparison of canonical optimization approaches (FGSM, FFGSM, DeepFool, C&W, PGD, CG) applied to this problem. We evaluate their performance across key dimensions: (1) attack success rate, (2) perturbation efficiency (ℓ_p norms, SSIM), and (3) computational cost, thereby mapping the inherent trade-offs and offering insights into the strategic selection of attack methods.

2 Problem Formulation

Let $f : \mathbb{R}^n \to \mathbb{R}^k$ represent a neural network classifier mapping an input $x \in \mathbb{R}^n$ (where $n = h \times w \times c$) to k class confidence scores. Given an input x correctly classified with label y_{true}, i.e., $\arg\max_j f_j(x) = y_{\text{true}}$, an adversarial perturbation $\delta \in \mathbb{R}^n$ is sought such that:

$$\arg\max_j f_j(x+\delta) \neq y_{\text{true}} \quad \text{and} \quad \|\delta\|_p \leq \varepsilon \tag{4}$$

Here, $\|\delta\|_p$ denotes the L_p norm of the perturbation, constrained by a budget $\varepsilon > 0$ to ensure perceptual similarity. The choice of norm p defines the geometric space in which similarity is measured. We primarily consider two standard norms:

L-infinity norm (ℓ_∞) bounds the maximum absolute change across all input dimensions:

$$\|\delta\|_\infty = \max_i |\delta_i| \tag{5}$$

This constraint enforces uniform deviation, limiting the peak perturbation intensity for any single element (pixel). For normalized images with pixel values in [0,1], an ℓ_∞ constraint of $\varepsilon = 0.03$ (approximately 8/255 in the original pixel scale) means no pixel can change by more than 3% of its possible range.

L2 norm (ℓ_2) measures the Euclidean magnitude of the perturbation vector:

$$\|\delta\|_2 = \sqrt{\sum_i \delta_i^2} \tag{6}$$

An ℓ_2 constraint restricts the overall energy of the perturbation, allowing for larger localized changes provided the total magnitude remains within ε.

While ℓ_p norms provide mathematical constraints, perceptual similarity is complex. We supplement our analysis with the Structural Similarity Index Measure (SSIM), a metric designed to better capture human visual perception of image degradation:

$$\text{SSIM}(x, x+\delta) = \frac{(2\mu_x\mu_{x+\delta} + c_1)(2\sigma_{x,x+\delta} + c_2)}{(\mu_x^2 + \mu_{x+\delta}^2 + c_1)(\sigma_x^2 + \sigma_{x+\delta}^2 + c_2)} \tag{7}$$

where μ, σ^2, and $\sigma_{x,y}$ represent means, variances, and covariance, respectively, and c_1, c_2 are stabilizing constants. SSIM values closer to 1 indicate higher structural similarity.

2.1 Optimization Formulation

The task of finding an adversarial example is fundamentally an optimization problem. We distinguish between untargeted attacks, which aim for any misclassification, and targeted attacks, which seek misclassification to a specific label $y_{\text{target}} \neq y_{\text{true}}$. Both can be formalized as minimizing the perturbation norm $\|\delta\|_p$ subject to achieving the desired misclassification outcome and respecting the norm constraint ε, as well as the valid input domain (e.g., pixel values in $[0,1]^n$). Table 1 summarizes these formulations.

Untargeted Attack	**Targeted Attack**
$\min_{\delta} \quad \|\delta\|_p$	$\min_{\delta} \quad \|\delta\|_p$
s.t. $\quad \arg\max_j f_j(x+\delta) \neq y_{\text{true}}$	s.t. $\quad \arg\max_j f_j(x+\delta) = y_{\text{target}}$
$\|\delta\|_p \leq \varepsilon$	$\|\delta\|_p \leq \varepsilon$
$x+\delta \in [0,1]^n$	$x+\delta \in [0,1]^n$

Table 1: Optimization formulations for untargeted and targeted adversarial attacks.

Solving these constrained optimization problems directly is often difficult due to the non-linearity of f and the discrete nature of the classification outcome. A common strategy is to replace the hard constraint $f(x+\delta) \neq y_{\text{true}}$ (or $= y_{\text{target}}$) with a continuous, differentiable surrogate loss function $\mathscr{L}$. The problem is then reformulated using a Lagrangian relaxation approach:

$$\min_{\delta} \quad c \cdot \mathscr{L}(f(x+\delta), y) + \|\delta\|_p \tag{8}$$

$$\text{subject to} \quad x+\delta \in [0,1]^n \tag{9}$$

where y is either y_{true} (for maximizing loss in untargeted attacks) or y_{target} (for minimizing loss towards the target), and $c > 0$ is a hyperparameter balancing the priority between minimizing the surrogate loss (achieving misclassification) and minimizing the perturbation norm $\|\delta\|_p$. The box constraint ensures the resulting image is valid.

For targeted attacks, a suitable surrogate loss $\mathscr{L}_{\text{targeted}}$ aims to maximize the confidence score of the target class relative to all other classes. A common choice is the margin loss:

$$\mathscr{L}_{\text{targeted}}(x+\delta, y_{\text{target}}) = \max(\max\{f_j(x+\delta) : j \neq y_{\text{target}}\} - f_{y_{\text{target}}}(x+\delta), -\kappa) \tag{10}$$

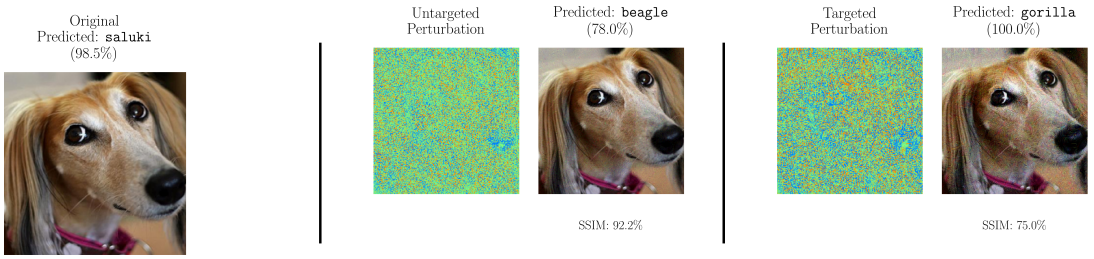

Figure 1: Illustration of untargeted vs. targeted adversarial attack outcomes on ResNet-18. **(Left)** Original image correctly classified as "saluki" (98.5% confidence). **(Center)** An untargeted attack seeks the minimal perturbation to induce *any* misclassification. Here, it results in "beagle" (48.5% confidence), a visually similar class, achieved with a relatively small perturbation (SSIM 91.9%). **(Right)** A targeted attack aims for a *specific* pre-determined class, here "gorilla". This requires a larger, more structured perturbation (SSIM 74.6%, indicating lower similarity) to force the network towards a semantically distant class, achieving high confidence (100.0%) in the incorrect target.

This loss encourages the score of the target class $f_{y_{\text{target}}}$ to exceed the highest non-target score by at least a margin $\kappa \geq 0$. Minimizing this loss (within the Lagrangian formulation Eq. 8) drives the network towards classifying the perturbed input as y_{target} with high confidence.

3 Related Work

The study of adversarial vulnerability in neural networks gained prominence with Szegedy et al. [14], who first formulated the problem as finding a minimal ℓ_2 perturbation δ causing misclassification, i.e., $f(x+\delta) \neq f(x)$. A computationally efficient alternative emerged with the Fast Gradient Sign Method (FGSM) [3], which uses a single gradient sign step ($\delta = \varepsilon \cdot \text{sign}(\nabla_x \mathscr{L}(f(x), y))$) to maximize loss under an ℓ_∞ constraint, establishing the basis for many subsequent gradient-based attacks. Iterative applications of this principle, notably Projected Gradient Descent (PGD) [10] and its variants [8], demonstrated improved attack strength by taking multiple smaller steps with projection onto the ℓ_p ball.

Geometric approaches sought minimal perturbations by analyzing decision boundaries directly. DeepFool [13] iteratively linearizes the classifier and projects the input onto the estimated decision boundary, often achieving smaller ℓ_2 perturbations than FGSM, albeit at higher computational cost. This line of inquiry highlighted the significance of the network's geometric structure in determining vulnerability, including the existence of universal perturbations [12].

Recent work has focused on improving both attack effectiveness and computational efficiency. The C&W method [1] frames the problem as Lagrangian relaxation, minimizing $\|\delta\|_p + c \cdot f(x+\delta)$ with the Adam optimizer to produce substantially smaller perturbations. Wong et al. [15] demonstrated that fast adversarial training can be more effective than free adversarial training, challenging previous assumptions about the trade-off between training speed and robustness. Mao et al. [11] conducted a large-scale empirical study of transfer attacks in real computer vision settings, providing new insights into the practical effectiveness of different attack methods. Hong et al. [5] proposed novel defense mechanisms against gradient-based attacks using gradients themselves, highlighting the ongoing arms race between attacks and defenses.

Our work builds upon these foundations by providing a systematic comparison of classical optimiza-

tion approaches for adversarial example generation. We establish a multidimensional evaluation framework that quantifies perturbation efficiency and computational cost, while also analyzing how different optimization methods navigate the non-convex landscape of neural networks.

4 Attack Methods

We evaluate six prominent adversarial attack methods, categorized by their core optimization strategy: gradient sign methods, geometric methods, and iterative optimization methods. This categorization highlights the distinct approaches employed to solve the constrained optimization problem formulated in Section 2.

4.1 Gradient Sign Methods

These methods utilize the sign of the loss function's gradient with respect to the input x to determine a perturbation direction, prioritizing computational simplicity.

4.1.1 Fast Gradient Sign Method (FGSM)

The Fast Gradient Sign Method [3], described in Algorithm 1, approximates the solution to the ℓ_∞-constrained loss maximization problem with a single linear step:

$$\max_{\delta} \mathscr{L}(f(x+\delta), y) \quad \text{s.t.} \quad \|\delta\|_\infty \leq \varepsilon \tag{11}$$

$$x_{\text{adv}} = x + \varepsilon \cdot \text{sign}(\nabla_x \mathscr{L}(f(x), y)) \tag{12}$$

FGSM identifies the direction of steepest ascent—the direction in which each pixel's intensity should change to maximize the loss function—then takes a single maximal step ε in that direction (i.e., shifts all pixels simultaneously by ε in their respective directions). Its primary advantage is computational efficiency, requiring only one gradient evaluation.

4.1.2 Fast FGSM (FFGSM)

Fast FGSM [15] introduces a small random initialization before the FGSM step to potentially improve effectiveness by escaping unfavorable local geometry as described in Algorithm 2:

$$x' = x + \alpha \cdot \text{sign}(\mathscr{N}(0,1)) \tag{13}$$

$$x_{\text{adv}} = \Pi_{\|\cdot\|_\infty \leq \varepsilon, [0,1]^n}(x' + (\varepsilon - \alpha) \cdot \text{sign}(\nabla_x \mathscr{L}(f(x'), y))) \tag{14}$$

where $\alpha \ll \varepsilon$ is a small random step size (e.g., $\alpha = 0.1\varepsilon$), and Π denotes projection onto the intersection of the ℓ_∞ ball around the original x and the valid input domain $[0,1]^n$. This adds minimal computational overhead.

4.2 Geometric Methods

This class of methods leverages the geometric properties of the network's decision boundaries.

Algorithm 1 Fast Gradient Sign Method (FGSM)

Require: Input x, label y, network f, loss $\mathscr{L}$, perturbation budget ε

$g \leftarrow \nabla_x \mathscr{L}(f(x), y)$ ▷ Compute loss gradient w.r.t. input

$\delta \leftarrow \varepsilon \cdot \text{sign}(g)$ ▷ Compute perturbation per Eq. 12

$x_{\text{adv}} \leftarrow x + \delta$ ▷ Apply perturbation to input

$x_{\text{adv}} \leftarrow \Pi_{[0,1]^n}(x_{\text{adv}})$ ▷ Clip to ensure valid image range

return x_{adv}

Algorithm 2 Fast FGSM (FFGSM)

Require: Input x, label y, network f, loss $\mathscr{L}$, budget ε, random step α

$x' \leftarrow x + \alpha \cdot \text{sign}(\mathscr{N}(0,1)^n)$ ▷ Take small random step

$x' \leftarrow \Pi_{[0,1]^n}(x')$ ▷ Clip intermediate point

$g' \leftarrow \nabla_x \mathscr{L}(f(x'), y)$ ▷ Compute gradient at x'

$x'' \leftarrow x' + (\varepsilon - \alpha) \cdot \text{sign}(g')$ ▷ Potential update (before projection)

$x_{\text{adv}} \leftarrow x + \Pi_{\|\cdot\|_\infty \leq \varepsilon}(x'' - x)$ ▷ Project total perturbation per Eq. 14

$x_{\text{adv}} \leftarrow \Pi_{[0,1]^n}(x_{\text{adv}})$ ▷ Clip final result to valid image range

return x_{adv}

4.2.1 DeepFool

DeepFool [13] seeks the minimal ℓ_2 perturbation required to cross a decision boundary. It operates by iteratively linearizing the classifier around the current point x_k and finding the smallest step r_k towards the separating hyperplane of the closest incorrect class:

$$\min_{\delta} \|\delta\|_2 \quad \text{s.t.} \quad \hat{k}(x+\delta) \neq \hat{k}(x) \tag{15}$$

where $\hat{k}(x)$ is the predicted class. The update step is:

$$x_{k+1} = x_k + r_k \tag{16}$$

with the step r_k calculated as the orthogonal projection onto the linearized boundary between the current predicted class $\hat{k}(x_k)$ and the nearest rival class j^*:

$$r_k = \frac{|f_{j^*}(x_k) - f_{\hat{k}(x_k)}(x_k)|}{\|\nabla f_{j^*}(x_k) - \nabla f_{\hat{k}(x_k)}(x_k)\|_2^2} \cdot (\nabla f_{j^*}(x_k) - \nabla f_{\hat{k}(x_k)}(x_k)) \tag{17}$$

where j^* minimizes the distance to the boundary among relevant classes. On large-scale classification problems, searching across all possible classes for the minimal distance becomes computationally prohibitive. To manage computational cost in high-dimensional output spaces, our implementation 3 considers only the top-k ($k \ll n$) most likely alternative classes.

4.3 Iterative Optimization Methods

These methods employ iterative algorithms to more carefully solve the constrained or unconstrained optimization formulations.

Algorithm 3 DeepFool

Require: Input x, network f, max iterations T_{max}, overshoot η, top-k classes K

$\delta \leftarrow 0$, $x_{adv} \leftarrow x$, $t \leftarrow 0$ ▷ Initialize perturbation, image, and counter

$y_0 \leftarrow \arg\max_j f_j(x_{adv})$ ▷ Initial prediction

while $t < T_{max}$ **and** $\arg\max_j f_j(x_{adv}) = y_0$ **do** ▷ Iterate until misclassification or max iterations

  Find K nearest class boundaries (linearized)

  $j^* \leftarrow \arg\min_{j \neq y_0} \frac{|f_j(x_{adv}) - f_{y_0}(x_{adv})|}{\|\nabla f_j(x_{adv}) - \nabla f_{y_0}(x_{adv})\|_2}$ ▷ Identify closest boundary

  $r \leftarrow \frac{|f_{j^*}(x_{adv}) - f_{y_0}(x_{adv})|}{\|\nabla f_{j^*}(x_{adv}) - \nabla f_{y_0}(x_{adv})\|_2^2} \cdot (\nabla f_{j^*}(x_{adv}) - \nabla f_{y_0}(x_{adv}))$ ▷ Perturbation step per Eq. 17

  $\delta \leftarrow \delta + r$ ▷ Accumulate perturbation

  $x_{adv} \leftarrow x + \delta$ ▷ Update image (projection often done finally)

  $t \leftarrow t + 1$

$\delta \leftarrow (1+\eta)\delta$ ▷ Apply overshoot to ensure crossing boundary

$x_{adv} \leftarrow x + \delta$ ▷ Final perturbed image before clipping

$x_{adv} \leftarrow \Pi_{[0,1]^n}(x_{adv})$ ▷ Clip to valid image range

return x_{adv}

Algorithm 4 Carlini & Wagner (C&W) (High-Level)

Require: Input x, target y_{target} (y_{true} for untargeted), network f, norm p, const c, conf κ, steps T

Initialize w via inverse tanh: $w \leftarrow \text{atanh}(2x-1)$ ▷ Initialize in unconstrained space

for $t = 0$ **to** $T-1$ **do**

  $x_{adv} \leftarrow \frac{1}{2}(\tanh(w)+1)$ ▷ Map w to valid image space

  $\delta \leftarrow x_{adv} - x$ ▷ Current perturbation

  $L_{target} \leftarrow \max(\max_{j \neq y_{\text{target}}} f_j(x_{adv}) - f_{y_{\text{target}}}(x_{adv}), -\kappa)$ ▷ Targeted margin loss per Eq. 18

  $L(w) \leftarrow \|\delta\|_p + c \cdot L_{target}$ ▷ Full C&W objective per Eq. 19

  $g \leftarrow \nabla_w L(w)$ ▷ Gradient w.r.t. unconstrained variable w

  $w \leftarrow w - \alpha \cdot \text{AdamUpdate}(g)$ ▷ Update w using Adam optimizer

$x_{adv} \leftarrow \frac{1}{2}(\tanh(w)+1)$ ▷ Final adversarial image from optimized w

return x_{adv}

4.3.1 Carlini & Wagner (C&W)

The C&W attack [1] tackles the Lagrangian formulation (Eq. 8) using a specialized loss function and a change-of-variables technique. The objective for a targeted attack is:

$$\min_{\delta} \|\delta\|_p + c \cdot \max(\max_{j \neq y_{\text{target}}} f_j(x+\delta) - f_{y_{\text{target}}}(x+\delta), -\kappa) \tag{18}$$

where $c > 0$ balances the loss and perturbation norm, and $\kappa \geq 0$ enforces a confidence margin. To handle the box constraints $x + \delta \in [0,1]^n$, C&W optimizes over an unconstrained variable w using the transformation $\delta(w) = \frac{1}{2}(\tanh(w)+1) - x$. This maps any $w \in \mathbb{R}^n$ to a valid perturbed image $x + \delta(w) \in [0,1]^n$. The optimization is then performed on w:

$$\min_{w} \left\| \frac{1}{2}(\tanh(w)+1) - x \right\|_p + c \cdot f_{\text{C\&W}}\left(\frac{1}{2}(\tanh(w)+1)\right) \tag{19}$$

where $f_{\text{C\&W}}$ denotes the C&W margin loss term (Eq. 18). This unconstrained problem is typically solved using adaptive optimizers like Adam as described in Algorithm 4.

4.3.2 Projected Gradient Descent (PGD)

PGD [10] is a fundamental iterative method for constrained optimization. It directly addresses the loss maximization problem subject to ℓ_p constraints by repeatedly taking small steps in the gradient direction (or sign direction for ℓ_∞) and projecting the result back onto the feasible ℓ_p ball centered at the original input x. The update rule at iteration t is:

$$\delta_{t+1} = \Pi_{\|\cdot\|_p \le \varepsilon}(\delta_t + \alpha \cdot d_t) \tag{20}$$

where α is the step size, d_t is the normalized ascent direction (e.g., $\text{sign}(\nabla_\delta \mathscr{L})$ for ℓ_∞ or $\nabla_\delta \mathscr{L} / \|\nabla_\delta \mathscr{L}\|_2$ for ℓ_2), and $\Pi_{\|\cdot\|_p \le \varepsilon}$ projects the intermediate perturbation onto the ℓ_p ball of radius ε. An additional projection or clipping step ensures $x + \delta_{t+1} \in [0,1]^n$. PGD iteratively refines the perturbation, often yielding stronger attacks than single-step methods. PGD effectively finds the constrained optimum by iteratively applying the gradient update followed by projection, ensuring feasibility throughout. The practical early stopping condition (line 8 of Algorithm 5) balances convergence and efficiency.

Algorithm 5 Projected Gradient Descent Attack

Require: Input image x, true/target label y, step size α, iterations T, perturbation budget ε

1: Initialize $\delta_0 \sim \mathscr{U}(-0.01, 0.01)^n$ ▷ Random initialization within small range
2: **for** $t = 0$ **to** $T-1$ **do**
3: $g_t \leftarrow \nabla_\delta \mathscr{L}(f(x+\delta_t), y)$ ▷ Compute loss gradient
4: $d_t \leftarrow \text{sign}(g_t)$ for ℓ_∞ or $d_t \leftarrow g_t / \|g_t\|_2$ for ℓ_2 ▷ Determine ascent direction d_t
5: $\delta_{t+1} \leftarrow \Pi_{\|\cdot\|_p \le \varepsilon}(\delta_t + \alpha \cdot d_t)$ ▷ Update perturbation per Eq. 20
6: $x_{t+1} \leftarrow \Pi_{[0,1]^n}(x + \delta_{t+1})$ ▷ Clip perturbed image to valid range
7: **if** $\arg\max_j f_j(x_{t+1}) \neq y_{\text{true}}$ **and** $\|\delta_{t+1} - \delta_t\|_p < 0.01 \|\delta_t\|_p$ **then**
8: **break** ▷ Early stopping if attack succeeds and converges
9: **return** δ_T

4.3.3 Conjugate Gradient (CG)

The Conjugate Gradient method described in Algorithm 6, primarily designed for efficiently minimizing quadratic functions, can be adapted for non-convex optimization. For $f(x) = \frac{1}{2}x^T Ax - b^T x$, CG generates A-orthogonal search directions p_k, allowing convergence in at most n steps. Applied to the adversarial loss $\mathscr{L}$, CG aims to accelerate convergence by constructing search directions p_k that are approximately conjugate with respect to the local Hessian $\nabla^2 \mathscr{L}$, potentially navigating the landscape more efficiently than first-order methods.

The core update involves constructing the search direction p_{k+1} from the current residual (negative gradient $r_{k+1} = -g_{k+1}$) and the previous direction p_k:

$$p_{k+1} = r_{k+1} + \beta_{k+1} p_k \tag{21}$$

where β_{k+1} maintains approximate conjugacy. The Fletcher-Reeves formula is:

$$\beta_{k+1}^{FR} = \frac{\|r_{k+1}\|_2^2}{\|r_k\|_2^2} \tag{22}$$

The optimal step size α_k for a quadratic form involves the Hessian A. For neural networks, it's approximated using the Hessian-vector product $\nabla^2 \mathscr{L} \cdot p_k$ computed efficiently via automatic differentiation:

$$\alpha_k \approx \frac{r_k^T r_k}{p_k^T (\nabla^2 \mathscr{L}(f(x+\delta_k), y) p_k)} \tag{23}$$

The algorithm iteratively updates the residual, computes the conjugate direction (Eq. 21), estimates step size using the Hessian-vector product (Eq. 23), takes a projected step, and repeats.

Algorithm 6 Projected Conjugate Gradient Attack (Fletcher-Reeves variant)

Require: Input image x, label y, iterations T, perturbation budget ε, tolerance tol

$\delta_0 \leftarrow \mathscr{U}(-0.01, 0.01)^n$ ▷ Random initialization
$g_0 \leftarrow \nabla_\delta \mathscr{L}(f(x+\delta_0), y)$ ▷ Initial gradient
$r_0 \leftarrow -g_0$ ▷ Initial residual
$p_0 \leftarrow r_0$ ▷ Initial search direction
for $k = 0$ to $T-1$ **do**
  $Ap_k \leftarrow \text{HessianVectorProduct}(p_k)$ ▷ Efficient Hessian-vector product
  $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$ ▷ Approximate step size per Eq. 23
  $\delta_{k+1} \leftarrow \Pi_{\|\cdot\|_p \le \varepsilon}(\delta_k + \alpha_k p_k)$ ▷ Update and project
  $g_{k+1} \leftarrow \nabla_\delta \mathscr{L}(f(x+\delta_{k+1}), y)$ ▷ Compute new gradient
  $r_{k+1} \leftarrow -g_{k+1}$ ▷ Update residual
  if $\|r_{k+1}\| < tol$ **or** $\arg\max_j f_j(x+\delta_{k+1}) \neq y_{\text{true}}$ **then** ▷ Early stop: convergence or success
    break
  $\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ ▷ Fletcher-Reeves update per Eq. 22
  $p_{k+1} \leftarrow r_{k+1} + \beta_{k+1} p_k$ ▷ New conjugate direction per Eq. 21
return δ_T

The primary trade-off is the higher per-iteration cost of the Hessian-vector product compared to a PGD gradient step, which may be offset by requiring fewer iterations for convergence, particularly on ill-conditioned or complex loss surfaces.

5 Experimental Setup

5.1 Dataset

Our evaluation dataset[1] consists of 1,000 images from the ImageNet validation set with one image per class. All images are preprocessed using standard ImageNet transformations:

- Resize to 256px along the shorter edge
- Center crop to 224×224px
- Convert to tensor and normalize to [0,1] range
- Normalize with ImageNet mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$

[1] `https://github.com/EliSchwartz/imagenet-sample-images`

Figure 2: Qualitative comparison of untargeted attacks on Resnet-18 generated by different methods (columns) for several input images (rows). Original images (leftmost column) are correctly classified. All attack methods induce misclassification to various incorrect labels.

5.2 Target Models

We evaluate adversarial attack methods on two residual networks (ResNet-18 and ResNet-50), a traditional convolutional architecture (VGG-16), and two efficiency-optimized networks (EfficientNet-B0 and MobileNet-V3-Large) as shown in Table 2. All models are pre-trained on ImageNet and used without modification to provide a realistic evaluation environment. All code for reproducing the experiments and analysis in this paper is available on GitHub [2].

Model	**Short Name**	**Top-1 Acc.**	**Top-5 Acc.**	**Params (M)**	**Inf. Time (ms)**
ResNet-18	RN18	79.78%	95.10%	11.69	0.53
ResNet-50	RN50	92.49%	99.00%	25.56	0.55
VGG-16	VGG	85.79%	97.10%	138.37	0.31
EfficientNet-B0	ENet	92.19%	98.89%	5.29	0.69
MobileNet-V3-Large	MNet	93.59%	99.20%	5.48	0.48

Table 2: Pre-trained model performance on our curated ImageNet dataset (1000 images, one per class). All experiments in subsequent tables use this same dataset, measured on a single NVIDIA RTX 4090 GPU with batch size of 64.

[2] `https://github.com/ali-izhar/adversarial-attacks`

5.3 Evaluation Framework

We evaluate each optimization method across three key dimensions. To assess *attack effectiveness*, we measure the success rate (percentage of inputs successfully misclassified). For *perturbation efficiency*, we quantify the L_2 norm (Euclidean distance between original and perturbed images), L_∞ norm (maximum absolute change across all pixels), and Structural Similarity Index (SSIM) to measure visual similarity. *Computational efficiency* is evaluated through the average iterations to convergence, total gradient computations required, and wall-clock runtime per successful attack.

6 Results and Analysis

6.1 Attack Effectiveness

Our analysis reveals significant variations in success rates across both gradient-based and optimization-based methods, as shown in Table 3. For the targeted attack evaluations presented here, the target class y_{target} for each input image x was selected as the class predicted by the model f to have the lowest confidence score, excluding the true class y_{true}. Specifically, $y_{\text{target}} = \arg\min_{j \neq y_{\text{true}}} f_j(x)$. This choice represents a challenging "hardest target" attack scenario.

Method	**Type**	**Config**	**Success Rate (%)**				
			RN18	**RN50**	**VGG**	**ENet**	**MNet**
Gradient Sign Methods							
FGSM	U	$\varepsilon = 4/255$	98.24	42.21	96.03	81.65	96.58
FFGSM	U	$\varepsilon = 8/255, \alpha = 0.04$	99.00	47.62	98.13	85.34	97.86
FGSM	T	$\varepsilon = 16/255$	0.00	0.00	0.00	0.11	0.00
FFGSM	T	$\varepsilon = 32/255, \alpha = 0.02$	0.13	0.43	0.12	1.09	0.32
Geometric Methods							
DeepFool	U	$steps = 50, over = 0.02$	99.87	99.78	100.00	99.78	99.79
DeepFool	T	N/A	-	-	-	-	-
Iterative Optim.							
C&W	U	$c = 1.0, \kappa = 0$	100.00	100.00	100.00	100.00	100.00
PGD	U	$\varepsilon = 8/255, steps = 40$	100.00	99.57	100.00	99.35	100.00
CG	U	$\varepsilon = 8/255, steps = 40$	100.00	99.89	99.88	99.67	99.89
C&W	T	$c = 10.0, \kappa = 5$	85.95	90.80	95.68	96.09	77.86
PGD	T	$\varepsilon = 16/255, steps = 200$	100.00	100.00	100.00	100.00	100.00
CG	T	$\varepsilon = 16/255, steps = 60$	100.00	99.89	100.00	100.00	100.00

Table 3: Attack effectiveness across models, showing success rates (%) for both untargeted (U) and targeted (T) attacks. Model accuracy under attack is (100% - success rate).

Simple gradient-based methods like FGSM and FFGSM achieve high untargeted success rates on some models (e.g., RN18, VGG, MNet). However, their effectiveness drops significantly on deeper,

more complex architectures like ResNet-50, likely because the local gradient information exploited by a single step is less indicative of a successful global perturbation direction in these more intricate loss landscapes. Furthermore, these single-step methods fail almost completely for targeted attacks across all architectures which is expected: maximizing loss locally (untargeted) is far simpler than navigating the optimization landscape to converge to a specific and potentially distant target class which will require more refined and iterative search directions.

In contrast, optimization-based methods like C&W, PGD, and CG, which iteratively refine the perturbation, achieve near-perfect success rates for both untargeted and targeted attacks across most architectures. For each attack method, we selected the configuration that provides the best balance between success rate and perturbation size, making these results directly comparable. *DeepFool does not support targeted attacks.*

To better understand the neural network's decision-making process under attack, we visualize the model's attention using Gradient-weighted Class Activation Mapping (Grad-CAM) in Figure 3.

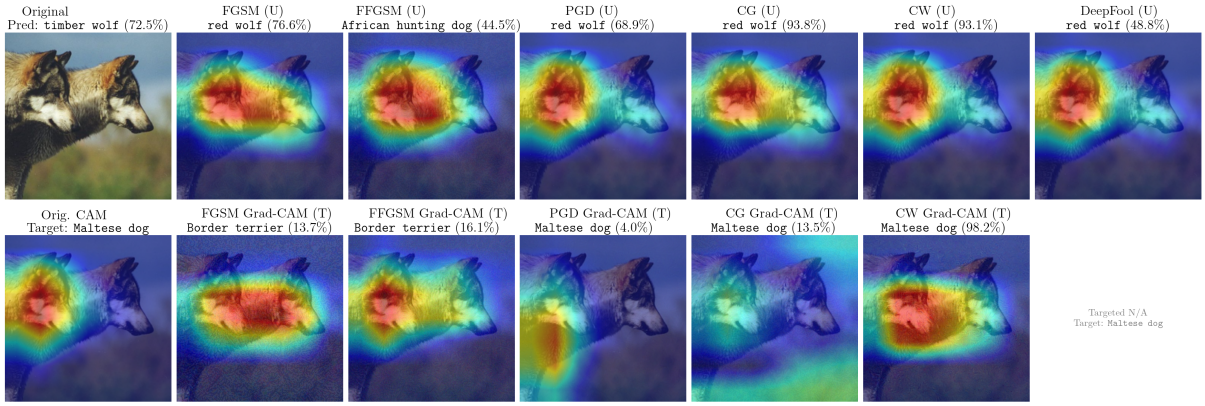

Figure 3: Comparison of adversarial attacks and their effect on model attention using Grad-CAM visualization. Note that single-iteration methods (FGSM, FFGSM) struggle with targeted attacks, producing either incorrect targets or failing entirely, while iterative methods (PGD, CG, C&W) successfully redirect the model's attention to features that lead to the target class with high confidence.

This technique highlights image regions most influential for the classification decision. The original image (left) shows Grad-CAM highlighting the wolf's features for its correct classification. **Untargeted Attacks (Top Row):** All methods successfully induce misclassification (mostly to "red wolf"). The corresponding Grad-CAM maps generally maintain focus on the original subject (the wolf), indicating the perturbations primarily exploit existing feature sensitivities near the decision boundary without drastically altering the model's focus. **Targeted Attacks (Bottom Row):** A stark difference emerges. *FGSM/FFGSM:* These methods fail to achieve the target class, instead misclassifying as "Border terrier". Crucially, their Grad-CAM maps resemble the untargeted case, with attention still largely centered on the original wolf features. This visually confirms their inability to effectively steer the model's focus towards a specific, different target with a single gradient step. *PGD/CG/C&W:* These iterative methods successfully force the classification to "Maltese dog". Their Grad-CAM maps show significant deviation from the original. PGD and CG appear to achieve this by diffusing the model's focus or highlighting less semantically relevant/potentially artifactual regions to induce the target classification, requiring moder-

ate computational effort. In contrast, C&W generates a highly localized attention pattern, suggesting it may be more aggressively altering salient features towards the target, which correlates with its higher computational cost but also its typically strong attack effectiveness and high confidence scores.

6.2 Perturbation Efficiency

The quality of adversarial attacks is often measured by how small the perturbations need to be (quantified by ℓ_p norms) and how visually similar the perturbed image remains to the original (quantified by metrics like SSIM). Table 4 compares perturbation magnitudes (ℓ_2 and ℓ_∞ norms), and Table 5 shows SSIM values across methods for untargeted and targeted attacks, respectively.

Type	**Method**	**RN18**		**RN50**		**VGG**		**ENet**		**MNet**	
		L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞	L_2	L_∞
Untargeted Attacks											
Grad. Sign	FGSM	0.0155	0.0159	0.0156	0.0159	0.0156	0.0159	0.0155	0.0159	0.0156	0.0159
	FFGSM	0.0278	0.0318	0.0278	0.0318	0.0278	0.0318	0.0278	0.0318	0.0278	0.0318
Geometric	DeepFool	0.0006	0.0080	0.0006	0.0102	0.0004	0.0068	0.0009	0.0169	0.0005	0.0101
Iterative	C&W	0.0040	0.0432	0.0044	0.0499	0.0046	0.0531	0.0028	0.0406	0.0039	0.0469
	PGD	0.0044	0.0072	0.0046	0.0072	0.0044	0.0072	0.0046	0.0072	0.0044	0.0072
	CG	0.0070	0.0072	0.0070	0.0072	0.0070	0.0072	0.0070	0.0072	0.0070	0.0072
Targeted Attacks											
Grad. Sign	FGSM	0.0615	0.0636	0.0615	0.0636	0.0615	0.0636	0.0615	0.0636	0.0615	0.0636
	FFGSM	0.0380	0.0636	0.0380	0.0636	0.0380	0.0636	0.0380	0.0636	0.0380	0.0636
Iterative	C&W	0.0277	0.2730	0.0315	0.3325	0.0275	0.3171	0.0227	0.2664	0.0223	0.2657
	PGD	0.0088	0.0144	0.0088	0.0144	0.0087	0.0144	0.0091	0.0144	0.0089	0.0144
	CG	0.0141	0.0144	0.0140	0.0144	0.0141	0.0144	0.0141	0.0144	0.0141	0.0144

Table 4: Perturbation magnitudes (L_2 and L_∞ norms) for both untargeted and targeted attacks. Lower values indicate more efficient perturbations.

DeepFool often achieves the smallest L_2 and L_∞ norms for untargeted attacks (Table 4), resulting in very high SSIM values (Table 5). C&W also produces small untargeted perturbations, particularly under the L_2 norm. For targeted attacks, PGD yields remarkably small perturbations across both norms,

Model	**Untargeted SSIM**						**Targeted SSIM**				
	FGSM	**FFGSM**	**DeepFool**	**C&W**	**PGD**	**CG**	**FGSM**	**FFGSM**	**C&W**	**PGD**	**CG**
RN18	0.9254	0.8153	0.9998	0.9920	0.9930	0.9826	0.5649	0.7250	0.8325	0.9736	0.9375
RN50	0.9249	0.8148	0.9998	0.9907	0.9926	0.9825	0.5643	0.7266	0.8109	0.9737	0.9369
VGG	0.9247	0.8142	0.9999	0.9898	0.9931	0.9824	0.5638	0.7258	0.8450	0.9742	0.9370
ENet	0.9254	0.8145	0.9995	0.9958	0.9925	0.9826	0.5657	0.7260	0.8795	0.9725	0.9371
MNet	0.9257	0.8168	0.9998	0.9921	0.9930	0.9827	0.5682	0.7268	0.8791	0.9736	0.9380

Table 5: Structural Similarity Index (SSIM) across models and attack methods. Higher values indicate better perceptual similarity. DeepFool is Untargeted only.

leading to high SSIM scores. The conjugate gradient method (CG) offers a balance, producing small perturbations, generally larger than PGD for targeted attacks but smaller than baseline methods like FFGSM. Simple methods like FGSM/FFGSM typically require larger perturbations, leading to lower SSIM scores, especially for targeted attacks where the visual distortion can become quite noticeable.

6.3 Computational Efficiency

The practical utility of an attack method hinges not only on its effectiveness and subtlety but also on its computational demands. Table 6 presents the computational profile of each method, quantifying the resources required (iterations, gradient computations, runtime).

Model	**Untargeted Runtime (ms)**						**Targeted Runtime (ms)**				
	FGSM *(1/1)*	**FFGSM** *(1/1)*	**DeepFool** *(4/45)*	**C&W** *(1k/1k)*	**PGD** *(4/7)*	**CG** *(40/40)*	**FGSM** *(1/1)*	**FFGSM** *(1/1)*	**C&W** *(1k/1k)*	**PGD** *(15/29)*	**CG** *(60/60)*
RN18	0.59	0.44	109.97	448.23	1.78	12.92	31.72	31.16	478.36	35.85	51.48
RN50	0.62	0.54	356.31	1286.17	35.36	46.16	30.31	30.81	1318.34	113.46	103.76
VGG	0.42	0.39	251.88	2754.67	11.28	81.81	29.48	28.73	2787.89	77.51	167.41
ENet	0.61	0.58	232.98	790.39	16.30	26.90	30.23	31.48	823.96	56.41	73.56
MNet	0.48	0.60	240.14	457.83	1.42	13.28	31.45	34.14	490.92	39.56	53.84

Table 6: Computational requirements (Average Runtime in ms per image) across models and attack methods (RTX 4090, batch size 64). Iteration/Gradient counts are shown in headers.

A clear trade-off emerges between attack sophistication and computational cost:

- **Gradient Sign Methods (FGSM, FFGSM):** These are exceptionally efficient, requiring only a single gradient computation. Their speed makes them suitable for scenarios demanding rapid generation, such as adversarial training, but this comes at the cost of effectiveness, particularly for targeted attacks and against more robust models.
- **Iterative First-Order Methods (PGD):** PGD offers a strong balance. Its computational cost is moderate; early stopping in untargeted scenarios further enhances its practical efficiency. It generally requires more computation than FGSM/FFGSM but significantly less than C&W, while achieving high success rates.
- **Geometric Methods (DeepFool):** DeepFool exhibits intermediate computational cost. Its iterative nature involves multiple gradient calculations per step (for finding the nearest boundary among top-k classes), making it more expensive than PGD but faster than C&W.
- **Conjugate Gradient (CG):** CG occupies a middle ground in terms of cost per iteration. While the Hessian-vector product adds overhead compared to a simple gradient step in PGD, the potentially faster convergence (fewer total iterations) can make it competitive or even more efficient than PGD for achieving a desired level of attack strength, especially on complex landscapes. Its cost is generally higher than PGD but substantially lower than C&W.
- **Optimization-Based Methods (C&W):** C&W consistently demonstrates the highest computational requirements. Its formulation often necessitates a large number of iterations using optimiz-

ers like Adam, along with the overhead of the change-of-variables transformation. This high cost is the price paid for generating highly effective attacks with often small perturbations.

Achieving higher success rates and smaller perturbations (like C&W or DeepFool) generally demands greater computational investment, while faster methods (like FGSM/FFGSM) sacrifice potency and subtlety. Methods like PGD and CG represent different compromises along this frontier.

6.4 Limitations

Our evaluation highlights several practical considerations and inherent limitations in the application of these optimization methods for adversarial attacks:

- **Non-Convexity and Local Minima:** The fundamental challenge is the highly non-convex nature of the loss landscapes of deep neural networks. All evaluated methods perform local optimization and are susceptible to converging to suboptimal local minima. There is no guarantee that the adversarial examples found are globally optimal (i.e., representing the absolute minimal perturbation). The quality of the solution can depend significantly on initialization and the specific optimization path taken.
- **Memory requirements**: Methods incorporating second-order information, even implicitly via Hessian-vector products (CG), generally have higher memory demands per iteration compared to first-order methods, which can be a constraint for very large models or limited hardware.
- **Computational scalability**: First-order methods scale better to larger batch sizes, making them more suitable for large-scale evaluation. More sophisticated optimization methods often require careful tuning of additional hyperparameters.

7 Conclusion

This survey systematically quantifies the performance of six foundational adversarial attacks: FGSM, FFGSM, DeepFool, C&W, PGD, and CG. Our empirical comparison across five CNN architectures demonstrates the critical trade-offs between attack effectiveness (success rate), perturbation efficiency (ℓ_p norms, SSIM), and computational requirements. The results show a clear spectrum: fast, single-step gradient sign methods (FGSM, FFGSM) sacrifice potency and subtlety for speed, proving insufficient for targeted attacks against complex models like ResNet-50. In contrast, iterative optimization techniques (C&W, PGD, CG) and geometric methods (DeepFool) consistently achieve higher success rates with smaller, less perceptible perturbations, but demand substantially greater computational resources. Within this group, C&W often yields the most effective attacks at the highest cost, PGD offers a compelling baseline balance, and CG presents an alternative option using approximate second-order information for efficiency gains on challenging loss surfaces.

These findings underscore that adversarial vulnerability is intrinsically linked to the network's optimization landscape and learned feature representations. The inherent non-convexity restricts current methods to finding local optima. Advancing the field requires deeper insights into the geometry of high-dimensional decision spaces and potentially new optimization strategies to navigate them, which will be crucial for developing both stronger attacks and fundamentally more robust defenses.

References

[1] Nicholas Carlini & David Wagner (2017): *Towards evaluating the robustness of neural networks*. In: *IEEE Symposium on Security and Privacy (SP)*, IEEE, pp. 39–57.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li & Li Fei-Fei (2009): *ImageNet: A large-scale hierarchical image database*. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 248–255.

[3] Ian J Goodfellow, Jonathon Shlens & Christian Szegedy (2015): *Explaining and harnessing adversarial examples*. In: *International Conference on Learning Representations*.

[4] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias & Gigel Macesanu (2020): *A survey of deep learning techniques for autonomous driving*. *Journal of Field Robotics* 37(3), pp. 362–386.

[5] H. Hong, Y. Hong & Y. Kong (2022): *An Eye for an Eye: Defending against Gradient-based Attacks with Gradients*.

[6] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran & Aleksander Madry (2019): *Adversarial examples are not bugs, they are features*. In: *Advances in Neural Information Processing Systems*, pp. 125–136.

[7] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly & Neil Houlsby (2020): *Big transfer (BiT): General visual representation learning*. In: *European Conference on Computer Vision*, Springer, pp. 491–507.

[8] Alexey Kurakin, Ian Goodfellow & Samy Bengio (2017): *Adversarial examples in the physical world*. In: *International Conference on Learning Representations Workshop*.

[9] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM Van Der Laak, Bram Van Ginneken & Clara I Sánchez (2017): *A survey on deep learning in medical image analysis*. *Medical image analysis* 42, pp. 60–88.

[10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras & Adrian Vladu (2018): *Towards deep learning models resistant to adversarial attacks*. In: *International Conference on Learning Representations*.

[11] Y. Mao, C. Fu, S. Wang, S. Ji, X. Zhang, Z. Liu & T. Wang (2022): *Transfer attacks revisited: A large-scale empirical study in real computer vision settings*. In: *2022 IEEE Symposium on Security and Privacy (SP)*, IEEE, pp. 1423–1439.

[12] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi & Pascal Frossard (2017): *Universal adversarial perturbations*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1765–1773.

[13] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi & Pascal Frossard (2016): *DeepFool: a simple and accurate method to fool deep neural networks*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582.

[14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow & Rob Fergus (2014): *Intriguing properties of neural networks*. In: *International Conference on Learning Representations*.

[15] Eric Wong, Leslie Rice & J Zico Kolter (2020): *Fast is better than free: Revisiting adversarial training*. In: *International Conference on Learning Representations*.