# LESS IS MORE? AN ABLATION STUDY ON AUTOATTACK FOR ADVERSARIAL ROBUSTNESS EVALUATION

**LUCA MELIS[1], LUCA SCIONIS[1,2], FABIO BRAU[1], MAURA PINTOR[1], BATTISTA BIGGIO[1]**

[1]Dept. of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy
[2]Dept. of Computer Engineering, Sapienza University of Rome, Rome, Italy
E-MAIL: {luca.melis6, luca.scionis, fabio.brau, maura.pintor, battista.biggio}@unica.it

**Abstract:**

**AutoAttack is widely recognized as a standard adversarial robustness evaluation framework, yet the individual contributions of its components and mechanisms remain insufficiently explored. In this work, we present a comprehensive ablation study on the standard AutoAttack version, isolating the singular contribution of each component, focusing on the attack ensemble, random initialization, and Expectation over Transformation (EoT) optimization across four different state-of-the-art robust models. Our analysis reveals that simplified attack sequences often achieve results comparable to the complete AutoAttack sequence while requiring significantly fewer computational resources. Furthermore, our findings show that EoT generally provides modest improvements in attack success rate, while the benefits of random initialization may vary depending on the model architecture. By identifying which among the AutoAttack components has the most significant influence on the robustness evaluation, our work offers practical recommendations for designing efficient evaluation frameworks that balance thoroughness with computational cost considerations.**

**Keywords:**

**Machine Learning Security; Adversarial Attacks**

## 1. Introduction

Deep Neural Networks are vulnerable to adversarial examples [1, 2]—subtle perturbations that cause incorrect predictions with high confidence. As machine learning systems become prevalent in safety-critical applications, evaluating *adversarial robustness* has become a fundamental research direction. AutoAttack [3] has emerged as the most used evaluation framework since it combines diverse parameter-free attacks to provide a standardized baseline for assessing model robustness. By running attacks sequentially, AutoAttack generates adversarial examples by leveraging complementary strengths of different strategies to exploit various defense vulnerabilities. The framework incorporates three common techniques to improve attack effectiveness: an *attack ensemble* exploiting a sequence of different attack methods suited to various defense strategies; *random initialization* which adds small random perturbations to inputs rather than starting from zero, promoting feature space exploration; and *Expectation over Transformation (EoT)* [5] which averages adversarial loss over multiple random transformations to improve effectiveness against randomized defenses. Despite their common use in adversarial attack literature, the individual impact of these mechanisms on AutoAttack has not been studied in detail. Our work provides a systematic ablation study of AutoAttack's components. By isolating and analyzing individual attacks and mechanisms, we aim to identify the essential elements driving AutoAttack's effectiveness, highlighting the need to optimize computational efforts without losing efficacy. Our experiments use models from RobustBench [6], allowing analysis across various defenses under standardized conditions. Our goal is to improve robustness evaluation efficiency by identifying the most significant components, potentially guiding the creation of better-optimized attack ensembles. The paper is organized as follows: Section 2 analyzes the components and mechanisms of AutoAttack, Section 3 discusses our ablation study results, and Section 4 summarizes findings and suggests future research directions.

## 2. AutoAttack Structural Analysis

AutoAttack [3] is a framework designed to evaluate the adversarial robustness of machine learning models against adversarial examples, constituted by an ensemble of distinct parameter-free adversarial attacks. By combining different at-

tack strategies, AutoAttack aims to provide a more reliable and comprehensive robustness evaluation. In this section, we examine its structure, describing its principal components and mechanisms, discussing therefore possible implications of its design choices. In the following, we are going to present the basic concepts about adversarial attacks in 2.1

## 2.1. Adversarial Attacks

Adversarial attacks aim to find perturbations of input samples that cause a machine learning model to make incorrect predictions. For a classification model $f : \mathcal{X} \to \mathbb{R}^c$, given an example $\mathbf{x}$ correctly classified as $y$, an adversarial example is a perturbed input $\mathbf{x}'$ that is classified differently from the original input $\mathbf{x}$, while ensuring $\mathbf{x}'$ remains perceptually similar to $\mathbf{x}$. In detail, adversarial attacks can be categorized as either *targeted* or *untargeted*:

- *Untargeted Attacks:* Aim to cause any misclassifications. For an input $\mathbf{x}$ with true class $y$, the goal is to find $\mathbf{x}'$ such that $y \neq \arg\max_i f(\mathbf{x}')_i$.
- *Targeted Attacks:* Aim to force the model to predict a specific incorrect class $t$. The goal is to find $\mathbf{x}'$ such that $t = \arg\max_i f(\mathbf{x}')_i$, where $t \neq y$.

Even if a large number of attacks have been proposed in the literature, one of the most widely known is the *Projected Gradient Descent* (PGD) [7], that is at the base of two of the attacks proposed in the AutoAttack framework. Specifically, PGD is a first-order iterative method that optimizes an adversarial example by taking steps in the direction of the gradient of a loss function $\mathcal{L}$, followed by a projection back onto the allowed perturbation space (i.e., a $\ell_p$-norm ball around the original input $x_0$). Given an input $\mathbf{x}$, and a label $y$, the PGD update step at iteration $i$ can be described as follows,

$$\mathbf{x}_{i+1} = \Pi_{\mathcal{B}^p_\epsilon(\mathbf{x}_0)}(\mathbf{x}_i + \alpha \cdot \mathbf{g}_i), \tag{1}$$

where $\alpha$ is the step size, $\mathcal{B}^p_\epsilon(\mathbf{x}_0) = \{\mathbf{x}' \in \mathcal{X} : \|\mathbf{x}' - \mathbf{x}_0\|_p \leq \epsilon\}$ is the $\ell_p$-ball of radius $\epsilon$ centered in $\mathbf{x}_0$, and $\Pi_{\mathcal{B}^p_\epsilon(\mathbf{x}_0)}(\cdot)$ is the projection operator onto this ball. Note that, when the input data is constituted by images, the input space $\mathcal{X}$ coincides with hyper-rectangle $[0, 1]^d$, meaning that all the input pixels must satisfy $0 \leq \mathbf{x} \leq 1$, also known as box-constraint.

The direction $\mathbf{g}_i$ is the sign of the gradient when $p = \infty$, or the normalized gradient for $p = 2$. Formally, the direction step is deduced as follows,

$$\mathbf{g}_i = \begin{cases} t \cdot \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_i, y)) & \text{if} \quad p = \infty \\ t \cdot \frac{\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_i, y)}{\|\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_i, y)\|_2} & \text{if} \quad p = 2 \end{cases}, \tag{2}$$

where $t = 1$ for untargeted attacks and $t = -1$ for targeted attacks, and $y$ is the true label for untargeted attacks or the target label for targeted attacks.

## 2.2. Attacks used in AutoAttack

AutoAttack's ensemble combines three distinct attacks, whose aim is to craft adversarial examples capable of fooling the target model. Here a brief description of these attacks, specifically APGD (*Auto-PGD*) and APGD-T (*Auto-PGD-T*) [3], FAB (*Fast Adaptive Boundary*) [8] and SQUARE [9], will be discussed.

### 2.2.1. Auto-PGD

APGD is a first-order optimization attack based on the PGD attack [7]. Unlike standard PGD that operates with a fixed $\alpha$, the APGD step size is dynamically adjusted during the attack, by monitoring the optimization progress (i.e., checking if the loss is consistently decreasing/increasing). In detail, while performing the optimization, if no increase of the loss function is registered after a predefined number of iterations, then the step-size $\alpha$ is halved. This allows for faster convergence initially and finer adjustments later. If $\alpha$ becomes too small, the attack might restart or terminate. This version of APGD leverages the *Cross-Entropy* loss function to perform an untargeted attack. The CE loss operates on the model's output probabilities $\mathbf{p} \in [0, 1]^c$, which are obtained by applying the softmax function to the logits $\mathbf{z} = f(\mathbf{x})$, where $f : \mathcal{X} \to \mathbb{R}^c$ is the classification model. Formally, the CE loss is defined as follows

$$\mathcal{L}_{CE}(\mathbf{x}, y) = -\log(p_y) = -\log \frac{\exp(f(\mathbf{x})_y)}{\sum_i \exp(f(\mathbf{x})_i)}. \tag{3}$$

Observe that, finding the $\mathbf{x}'$ that maximizes this loss, by following the iteration procedure described in Eq. 1, causes the model to misclassify the adversarial input $\mathbf{x}'$.

### 2.2.2. Auto-PGD-Targeted

APGD-T produces an untargeted attack via a sequence of targeted attacks. For an input $\mathbf{x}$ with true class $y$, the attack selects the top-$k$ logits ($k = 10$) excluding the true class as potential targets. For each target class $t \neq y$, a separate targeted attack executes using a specially designed loss. By targeting the most promising classes, this approach efficiently explores the adversarial landscape around the decision boundary.

Given an input $\mathbf{x}$ with true class $y$ and target class $t$, the *Targeted Difference of Logits Ratio* DLR−T is formulated as

$$\mathcal{L}_{DLR}^{\text{targeted}}(\mathbf{x}, y) = -\frac{z_y - z_t}{z_{\pi_1} - \frac{1}{2}(z_{\pi_3} + z_{\pi_4})}, \tag{4}$$

where $\pi$ represents indices of logits sorted in descending order. The numerator contrasts the correct class logit and target class logit, driving $z_t$ to surpass $z_y$. The denominator normalizes the difference and prevents degenerate cases.

## 2.3. FAB and Square Attacks

AutoAttack integrates two complementary attack strategies: FAB and SQUARE Attack, each targeting different defense vulnerabilities.

FAB [8] finds the minimal $\ell_p$-norm perturbation ($p \in \{1, 2, \infty\}$) making correctly classified inputs adversarial. It iteratively searches for the closest misclassified point by initializing from the original point or randomly at distance $\min\{u, \epsilon\}/2$, identifying the class with minimal ratio of logit to gradient differences, computing boundary projections, updating points using convex combinations with coefficient $\alpha$, maintaining the best adversarial example $\mathbf{x}_{\text{out}}$, and performing interpolation between original and updated points using parameter $\beta$. In AutoAttack, a targeted version called FAB−T is employed.

SQUARE Attack [9] operates as a black-box, score-based approach that doesn't require gradient access, making it effective against gradient-masking defenses [10]. It applies random, structured perturbations by modifying square regions of the input, accepting or discarding perturbations based on whether they produce misclassification. The algorithm initializes a candidate adversarial example, selects squares of decreasing side length $h^{(i)}$ according to a predefined schedule, samples perturbations $\delta$ applied to square regions, projects perturbed inputs to satisfy norm constraints, and updates candidates when better loss values are found.

## 2.4. AutoAttack Pipeline

AutoAttack is structured as a sequential pipeline of adversarial attacks, each included to progressively increase the overall attack success rate. In its standard configuration, AutoAttack is constituted by four distinct attacks, in the following fixed order:

1. APGD−CE (Auto-PGD with CE loss).
2. APGD−T (Auto-PGD-T with DLR−T loss).
3. FAB−T (Fast Adaptive Boundary).
4. SQUARE Attack.

Starting from the first attack, each one is run sequentially. Unsuccessfully attacked samples are passed to the next attack, while successful ones are filtered out. The process continues until all the samples have been successfully attacked or all attacks in the ensemble have been applied.

## 2.5. Additional Heuristics

AutoAttack incorporates a few heuristics to boost the effectiveness of its ensemble. Two of the most relevant are *Random Initialization* and *Expectation over Transformation*. For *Random Initialization*, a random starting point $\mathbf{x}_0$ is used in all the attacks. For $\ell_\infty$ and $\ell_2$ norms, random vectors are sampled from uniform and Gaussian distributions respectively, then normalized and scaled. The $\ell_1$ case uses a specialized projection function, $\Pi_1(\mathbf{x}, \epsilon)$, that is the projection on $\mathcal{B}_\epsilon^1(\mathbf{x})$, the $l_1$-ball of radius $\epsilon$ used in equation 1, to ensure constraint satisfaction. Notably, APGD and SQUARE attacks utilize the full perturbation budget ($\epsilon$) when initializing adversarial examples, while FAB consistently employs a more conservative approach, using only half the perturbation budget ($0.5\epsilon$) across all norm types. Table 1 illustrates the initialization strategies employed by different adversarial attack methods across various norm constraints. *Expectation over Transformation (EoT)* is a technique that computes the expected value of the loss function over a distribution of input transformations. Formally, EoT optimizes $\mathbb{E}_{\psi\sim\Psi}[\mathcal{L}(f(\psi(\mathbf{x})), y)]$ instead of simply $\mathcal{L}(f(\mathbf{x}), y)$, where $\Psi$ represents a set of possible transformations (e.g., affine transformations, small rotations, translations, or brightness adjustments). By averaging gradients across multiple transformed versions of the input, adversarial examples become robust against defenses that employ preprocessing or data augmentation techniques. In AutoAttack, EoT is specifically implemented in the APGD component to enhance attack

**TABLE 1.** Initial perturbation strategies for different attack methods and norms.

| Attack | Norm | Initial Point |
|---|---|---|
| APGD, SQUARE | $\ell_\infty$ | $\mathbf{x}_0 = \mathbf{x} + \epsilon \cdot \mathbf{u}, \quad \mathbf{u} \sim \mathcal{U}(-1, 1)$ |
| | $\ell_2$ | $\mathbf{x}_0 = \mathbf{x} + \epsilon \cdot \frac{\mathbf{u}}{\|\mathbf{u}\|_2}, \ \mathbf{u} \sim \mathcal{N}(\mathbf{0}, 1)$ |
| | $\ell_1$ | $\mathbf{x}_0 = \Pi_1(\mathbf{v}, \epsilon), \quad \mathbf{v} \sim \mathcal{N}(\mathbf{x}, 1)$ |
| FAB | $\ell_\infty$ | $\mathbf{x}_0 = \mathbf{x} + \epsilon \cdot \mathbf{u}, \ \mathbf{u} \sim \mathcal{U}(-0.5, 0.5)$ |
| | $\ell_2$ | $\mathbf{x}_0 = \mathbf{x} + \epsilon \cdot \frac{\mathbf{u}}{\|\mathbf{u}\|_2}, \mathbf{u} \sim \mathcal{N}(\mathbf{0}, 0.5)$ |
| | $\ell_1$ | $\mathbf{x}_0 = \Pi_1(\mathbf{v}, 0.5\epsilon), \quad \mathbf{v} \sim \mathcal{N}(\mathbf{x}, 1)$ |

transferability across different defensive preprocessing methods.

## 3. Experiments

This section proposes an ablation study made up to determine which components have a relevant impact on the ensemble results, both in terms of performance and computational costs. Here, we describe how we set up our experiments and then present the results obtained.

### 3.1. Experimental Setup

All experiments have been conducted on a single NVIDIA-RTX-A6000 GPU, with a fixed seed for reproducibility. We used a subset of 1000 samples from CIFAR-10 [11] test dataset. AutoAttack has been evaluated on three state-of-the-art robust models, taken directly from RobustBench [6]: `M1`, a *WideResNet-28-10* adversarially trained with improved diffusion model data augmentation [12]; `M2`, a *PreActResNet-18* adversarially trained with diffusion-generated data augmentation [13]; `M3`, an *XCiT-L12* adversarially trained with a custom recipe for Vision Transformers [14]. Finally, `M4`, a sparse *ResNet18* whose defense relies on obfuscated gradients [15]. Each model is tested using $\ell_\infty$ and $\ell_2$ as threat models (except for `M4`, which is only tested on $L_\infty$). We focused on the standard norms $\ell_\infty$ (perturbation budget $\epsilon = 8/255$) and $\ell_2$ (perturbation budget $\epsilon = 0.5$) as perturbation constraints, since AutoAttack is considered as a reference in these two norms for the evaluation of adversarial robustness.

To properly compare the effects of the single components, we implemented a custom version of AutoAttack that enables selective activation or deactivation of the functionalities discussed in the previous section. Our main objective is to preserve the integrity of the original AutoAttack pipeline while allowing this analysis.

#### 3.1.1. Evaluation Metrics

For each experiment, we collect both per-sample and aggregated results, including: the *Attack Success Rate* (ASR), the fraction of test samples for which an adversarial example is found:

$$ASR = \frac{\#\text{ Success}}{\#\text{ Total samples}} \tag{5}$$

The number of *Queries* as the average total number of forward (classifications) and backward passes (gradients evaluation) computed per sample:

$$\text{Queries} = \#\text{Forwards} + \#\text{Backwards} \tag{6}$$

Finally, the *Execution time*, the total runtime required for each attack configuration.

For our ablation study, 32 configurations per model were considered (except for `M4`, with only 16 configurations as it's only tested on $L_\infty$). We allowed selection for Random initialization in all four attacks in the standard pipeline to consent or not to the alteration. We also considered whether to apply Expectation over Transformation (EoT). If not applied, the gradient is initialized to zero before the first attack iteration.

For each sample, we save the original and adversarial predictions, in addition to tracking metrics. Results are presented and discussed in the following sections.

### 3.2. Results

All the results are reported in Table 2. Across all the non-obfuscated-gradient models—`M1`, `M2`, and `M3`—the results reveal a clear trend: the first two attacks in the pipeline, `APGD-CE` and `APGD-T`, account for all or almost the entirety of the improvement in attack success rate (ASR) overall. As shown in Table 2, the introduction of `APGD-T` provides a noticeable increase in ASR (typically between 1 and 3 percentage points, depending on the model and norm), highlighting the combination of targeted attacks with untargeted ones. However, the addition of `FAB-T` and `SQUARE` Attack to the ensemble does not provide any further meaningful improvement. In virtually all configurations, ASR remains unchanged after their inclusion, despite a substantial increase in computational cost. This observation raises an important question about the cost-effectiveness of maintaining such a broad attack ensemble for standard robust models: does the marginal gain (often virtually zero) justify the extra computational effort? Our evidence suggests that it does not, at least for those models that do not rely on forms of gradient obfuscation [10]. Regarding the auxiliary heuristics of the pipeline, random initialization and Expectation over Transformation (EoT) appear to have only a marginal effect on ASR, as shown in Table 2. Any observed change in robustness is, in practice, negligible (typically 0.1 percentage points or less) and, crucially, not consistent across models or threat models. In some cases, as shown in Table 2 for model M3 (norm $\ell_2$), the presence of these two mechanisms seems to reduce, even if by a small amount, the overall ASR. For what concerns robustness evaluation in the presence of gradient obfuscation, the impact of the `SQUARE` Attack is evident. In particular, when attacking the M4 model, which is known to employ gradient obfuscation, the `SQUARE` Attack is able to bypass the defense and achieve a substantial increase in ASR (up to 10 percentage points over the ensemble without it). This result shows the effectiveness of black-box attacks in improv-

**TABLE 2.** Comprehensive AutoAttack ablation study across all models (M1-M4).

| Norm | R.Init | EoT | Attacks | M1 | | | M2 | | | M3 | | | M4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ASR | Queries | Time | ASR | Queries | Time | ASR | Queries | Time | ASR | Queries | Time |
| $\ell_\infty$ | F | F | apgd-ce | 0.286 | 196 | 74.1 | 0.388 | 186 | 20.0 | 0.403 | 195 | 213.4 | 0.643 | 142 | 22.1 |
| | | | apgd-ce+apgd-t | 0.322 | 1440 | 555.7 | 0.414 | 1260 | 139.4 | 0.418 | 1262 | 1392.1 | 0.756 | 629 | 192.6 |
| | | | apgd-ce+apgd-t+fab-t | 0.322 | 3290 | 1257.8 | 0.414 | 2859 | 308.3 | 0.419 | 2848 | 3110.1 | 0.756 | 1267 | 359.0 |
| | | | apgd-ce+apgd-t+fab-t+square | 0.322 | 6683 | 2449.0 | 0.414 | 5792 | *548.2* | 0.419 | *5755* | 6045.6 | 0.865 | *1818* | 671.5 |
| | F | T | apgd-ce | 0.285 | 197 | 74.9 | 0.389 | 186 | 19.0 | 0.403 | 196 | 214.7 | 0.653 | 143 | 20.4 |
| | | | apgd-ce+apgd-t | 0.322 | 1447 | 557.9 | *0.415* | 1265 | 141.1 | 0.420 | 1265 | 1395.7 | 0.765 | 604 | 199.4 |
| | | | apgd-ce+apgd-t+fab-t | 0.322 | 3297 | 1260.4 | *0.415* | 2862 | 308.9 | *0.421* | 2846 | 3103.6 | 0.765 | 1207 | 386.9 |
| | | | apgd-ce+apgd-t+fab-t+square | 0.322 | *6690* | 2450.1 | *0.415* | 5789 | 546.1 | *0.421* | 5746 | 6037.2 | 0.862 | 1771 | *706.3* |
| | T | F | apgd-ce | 0.286 | 196 | 72.6 | 0.388 | 186 | 19.2 | 0.400 | 195 | 213.8 | 0.656 | 142 | 20.8 |
| | | | apgd-ce+apgd-t | 0.322 | 1440 | 555.6 | 0.414 | 1260 | 139.7 | 0.419 | 1261 | 1392.1 | 0.767 | 610 | 196.5 |
| | | | apgd-ce+apgd-t+fab-t | 0.322 | 3290 | 1256.9 | 0.414 | 2859 | 307.2 | 0.420 | 2844 | 3109.2 | 0.767 | 1210 | 362.4 |
| | | | apgd-ce+apgd-t+fab-t+square | 0.322 | 6683 | 2451.7 | 0.414 | 5791 | 545.2 | 0.420 | 5746 | 6040.8 | 0.863 | 1766 | 693.1 |
| | T | T | apgd-ce | 0.283 | 197 | 74.8 | 0.387 | 186 | 19.2 | 0.400 | 196 | 214.8 | 0.657 | 143 | 19.9 |
| | | | apgd-ce+apgd-t | 0.322 | 1447 | 557.7 | 0.414 | 1267 | 141.4 | 0.419 | 1267 | 1402.4 | 0.760 | 606 | 187.7 |
| | | | apgd-ce+apgd-t+fab-t | *0.323* | 3295 | 1258.0 | 0.414 | 2866 | 310.0 | 0.420 | 2850 | 3110.1 | 0.760 | 1222 | 391.9 |
| | | | apgd-ce+apgd-t+fab-t+square | *0.323* | 6682 | *2452.3* | 0.414 | *5799* | 547.1 | *0.421* | 5743 | *6049.1* | *0.867* | 1784 | 675.2 |
| $\ell_2$ | F | F | apgd-ce | 0.161 | 199 | 75.8 | 0.324 | 186 | 19.4 | 0.273 | 195 | 212.5 | - | - | - |
| | | | apgd-ce+apgd-t | *0.172* | 1709 | 656.6 | *0.342* | 1388 | 155.7 | *0.285* | 1500 | 1390.8 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t | *0.172* | 3969 | 1500.6 | *0.342* | 3183 | 340.2 | *0.285* | 3451 | 3107.4 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t+square | *0.172* | 8112 | 2985.9 | *0.342* | 6476 | 642.0 | *0.285* | 7029 | 6041.2 | - | - | - |
| | F | T | apgd-ce | 0.161 | 200 | 76.2 | 0.324 | 186 | 19.1 | 0.272 | 196 | 214.3 | - | - | - |
| | | | apgd-ce+apgd-t | *0.172* | 1717 | 659.7 | 0.341 | 1397 | 158.2 | 0.284 | 1509 | 1398.6 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t | *0.172* | 3978 | 1504.7 | 0.341 | 3193 | 341.8 | 0.284 | 3463 | 3112.8 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t+square | *0.172* | *8121* | 2982.3 | 0.341 | *6485* | *646.8* | 0.284 | *7046* | *6053.7* | - | - | - |
| | T | F | apgd-ce | 0.161 | 199 | 75.9 | 0.324 | 186 | 19.5 | 0.272 | 195 | 213.1 | - | - | - |
| | | | apgd-ce+apgd-t | *0.172* | 1709 | 656.8 | *0.342* | 1388 | 155.4 | *0.285* | 1501 | 1391.5 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t | *0.172* | 3969 | 1505.4 | *0.342* | 3183 | 341.3 | *0.285* | 3451 | 3105.7 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t+square | *0.172* | 8112 | 2981.6 | *0.342* | 6476 | 643.8 | *0.285* | 7029 | 6039.8 | - | - | - |
| | T | T | apgd-ce | 0.161 | 200 | 76.2 | 0.324 | 186 | 19.3 | 0.272 | 196 | 215.2 | - | - | - |
| | | | apgd-ce+apgd-t | *0.172* | 1717 | 659.6 | 0.341 | 1397 | 157.5 | 0.284 | 1509 | 1400.3 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t | *0.172* | 3978 | 1506.5 | 0.341 | 3193 | 342.9 | 0.284 | 3463 | 3112.8 | - | - | - |
| | | | apgd-ce+apgd-t+fab-t+square | *0.172* | *8121* | *2994.2* | 0.341 | *6485* | 646.1 | 0.284 | *7046* | 6050.3 | - | - | - |

ing the attack success rate when a model's defense relies on gradient obfuscation. However, notable results are achieved even with the combination of APGD and APGD−T. From the perspective of computational efficiency, each additional attack consistently increases the computational resources required, as shown in the results. The per-sample query count grows from around 200, using APGD−CE alone, to around 7000 for the entire ensemble for model M2, and the execution time increases accordingly. However, as previously noted, this additional cost is not matched by a corresponding increase in ASR. Most of the benefit has already been obtained with the first two attacks, which means that only APGD effectively impacts the ensemble; the rest of the pipeline predominantly adds overhead.

## 4. Conclusions and Future Work

In this work, we carried out an ablation study of the AutoAttack framework, questioning the necessity and effectiveness of some of its components when applied to a set of state-of-the-art robust models. Our results reveal that the majority of the attack success rate (ASR) gains are achieved by the first two attacks in the pipeline, namely APGD−CE and APGD−T. Including further attacks—FAB−T and SQUARE Attack rarely leads to any substantial improvement in ASR for models that do not employ gradient obfuscation while consistently adding relevant computational effort. We highlight the advantage of combining an attack with its targeted version to increase ASR performance. In contrast, random initialization and Expectation over Transformation (EoT) provide marginal benefits, at least in our results.

Based on these findings, we recommend a more selective approach in adversarial robustness evaluation: for standard robust models, restricting the ensemble to APGD−CE and APGD−T offers a favorable balance between attack strength and computational efficiency, while the whole ensemble, including SQUARE Attack, should be reserved for cases where gradient obfuscation is suspected.

This study is not without limitations. Our analysis is confined to a specific set of models and attack parameters; moreover, our study was conducted only on a subset of samples from the CIFAR-10 dataset. It may be necessary to extend the analysis to other datasets. Future research may explore the development of effective ensemble methodologies exploiting more different attacks while keeping low computational costs.

## Acknowledgements

This work has been carried out while L. Scionis was enrolled in the Italian National Doctorate on AI run by the Sapienza University of Rome in collaboration with the University of Cagliari. This work has been partly supported by the EU-funded Horizon Europe projects ELSA (GA no. 101070617), Sec4AI4Sec (GA no. 101120393) and CoEvolution (GA no. 101168560); and by the projects SERICS (PE00000014) and FAIR (PE00000013) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

## References

[1] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *ECML PKDD, Part III*, ser. LNCS, vol. 8190. Springer Berlin Heidelberg, 2013, pp. 387–402.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[3] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *ICML*, 2020.

[4] C. Yao, P. Bielik, P. Tsankov, and M. Vechev, "Automated discovery of adaptive attacks on adversarial defenses," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 858–26 870, 2021.

[5] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 284–293. [Online]. Available: `https://proceedings.mlr.press/v80/athalye18b.html`

[6] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: A standardized adversarial robustness benchmark," in *NeurIPS Datasets and Benchmarks*, 2021.

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[8] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *ICML*, 2020.

[9] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," 2020.

[10] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*. PMLR, 2018, pp. 274–283.

[11] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[12] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan, "Better diffusion models further improve adversarial training," in *International conference on machine learning*. PMLR, 2023, pp. 36 246–36 263.

[13] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian, and T. A. Mann, "Improving robustness using generated data," *Advances in neural information processing systems*, vol. 34, pp. 4218–4233, 2021.

[14] E. Debenedetti, V. Sehwag, and P. Mittal, "A light recipe to train robust vision transformers," in *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE, 2023, pp. 225–253.

[15] C. Xiao, P. Zhong, and C. Zheng, "Enhancing adversarial defense by k-winners-take-all," *arXiv preprint arXiv:1905.10510*, 2019.