Constrained Gradient Descent: A Powerful and Principled Evasion Attack Against Neural Networks

Weiran Lin [1 2] **Keane Lucas** [1 2] **Lujo Bauer** [1 2 3] **Michael K. Reiter** [4] **Mahmood Sharif** [5]

Abstract

We propose new, more efficient targeted white-box attacks against deep neural networks. Our attacks better align with the attacker's goal: (1) tricking a model to assign higher probability to the target class than to any other class, while (2) staying within an ϵ-distance of the attacked input. First, we demonstrate a loss function that explicitly encodes (1) and show that Auto-PGD finds more attacks with it. Second, we propose a new attack method, Constrained Gradient Descent (CGD), using a refinement of our loss function that captures both (1) and (2). CGD seeks to satisfy both attacker objectives—misclassification and bounded ℓ_p-norm—in a principled manner, as part of the optimization, instead of via ad hoc post-processing techniques (e.g., projection or clipping). We show that CGD is more successful on CIFAR10 (0.9–4.2%) and ImageNet (8.6–13.6%) than state-of-the-art attacks while consuming less time (11.4–18.8%). Statistical tests confirm that our attack outperforms others against leading defenses on different datasets and values of ϵ.

1. Introduction

With the prevalence of machine learning (ML), adversarial ML techniques that slightly manipulate the inputs of an ML model to influence its functionality have also been developed (Croce & Hein, 2020b). One type of attack on classification models is the *evasion* attack, which applies a small perturbation, within a distance limit, to a classifier's input to induce misclassification during inference. The ℓ_∞ *distance* is commonly used for specifying distance limits via ϵ *boundaries*, which define the maximum change acceptable in each element of the input. In previous work it is also common for attackers to have access to all weights of the model, known as the *white-box* scenario, and the attackers aim to force a specific misclassification, performing *targeted* attacks. Previous work proposes an attack method that iteratively perturbs an input in the direction of the gradients of a loss function, which is not necessarily the same loss function used by the model, and automatically truncates the attack in each iteration to stay within the distance limits. Researchers have shown that such attacks are effective against state-of-the-art neural networks (Moosavi-Dezfooli et al., 2016; Szegedy et al., 2014).

Croce et al. show that well-tuned parameters and carefully designed loss functions can boost the performance of attacks (Croce & Hein, 2020b). For example, varying the loss function of Auto Projected Gradient Descent (Auto-PGD), a state-of-the-art attack, between the cross-entropy, Carlini and Wagner (CW), and Difference of Logits Ratio (DLR) losses has a substantial impact on the attack's performance. Guided by this observation, we define a new loss function, the *Minimal Difference loss (MD loss)*, that better aligns with the goal of a targeted attack: MD loss aims to (mis)lead the model to assign higher confidence to the target class than to *any other class*, even if by just a tiny amount. We empirically show that Auto-PGD with the MD loss finds on average 0.5–12.3% more adversarial examples, depending on the dataset and model, than Auto-PGD with other loss functions.

Although MD loss substantially improves Auto-PGD's performance, we still notice limitations that hinder the attack's effectiveness. In particular, like other attacks in the PGD family, Auto-PGD uses projection at the end of each iteration to satisfy the norm constraints, eliminating changes that fall outside a predefined ℓ_p ball. However, projecting adversarial examples back into the ℓ_p ball may work against the attacker's misclassification objective, thus harming the

[1]Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, US [2]Cylab, Carnegie Mellon University, Pittsburgh, PA, US [3]Institute for Software Research, Carnegie Mellon University, Pittsburgh, PA, US [4]Departments of Computer Science and Electrical & Computer Engineering, Duke University, Durham, NC, US [5]School of Computer Science, Tel Aviv University, Tel Aviv, Israel. Correspondence to: Weiran Lin <weiranl@andrew.cmu.edu>, Keane Lucas <kjlucas@andrew.cmu.edu>, Lujo Bauer <lbauer@cmu.edu>, Michael K. Reiter <michael.reiter@duke.edu>, Mahmood Sharif <mahmoods@cs.tau.ac.il>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

attack's success. Moreover, because projection is implemented by clipping, an *ad hoc* operation to eliminate any changes made outside the ℓ_p ball, balancing the attack's two objectives (i.e., misclassification and not leaving the ℓ_p ball) remains a challenge for Auto-PGD. To address this shortcoming, we propose the *Constrained Gradient Descent (CGD)* attack. CGD enables the attacker to balance the two attack objectives—it allows adversarial examples to lie outside the ℓ_p ball during the attack process and models the violation of the norm constraints as part of its loss function to gradually learn to stay within the ℓ_p ball while achieving misclassification. We find that, with the same ℓ_∞ distance limit, CGD finds on average 0.3–1.3% more adversarial examples than Auto-PGD using the MD loss, while consuming less time (11.41–18.76%). Furthermore, we show that CGD outperforms Auto-PGD with three previously established loss functions on the CIFAR10 and ImageNet datasets, with statistical significance.

As an example of using CGD as a framework to find other types of attacks, we also define a variant of CGD for white-box untargeted attacks. Similar to its targeted attack counterpart, this variant also gradually learns to stay within the ℓ_p ball while achieving misclassification, due to modeling the violation of the norm constraints as part of its loss function. This variant outperforms the previous best attack by 0.3–2% on the CIFAR10 dataset, on average.

In a nutshell, our contributions are:

- We improve an established targeted attack method by offering a new loss function that better captures the goal of targeted attacks (§3).
- We invent a new attack that learns to stay within the ℓ_∞ distance limit rather than using simple clipping (§4), and we empirically demonstrate that it outperforms previously established ones (§5–6).
- We demonstrate how the proposed method can be used as a framework for attacks by instantiating it to define a stronger untargeted attack (§7).

2. Background

2.1. Threat Model

Adversary goals We consider a supervised classification setting in which an ML model F is trained to map a sample x to the correct label y by minimizing a loss function $L(x, y)$ such as the cross-entropy loss (L_{CE}). At inference time, a sample x is assigned the class i with the highest logit Z_i or highest confidence P_i. To find an adversarial example x', the adversary could either launch an untargeted attack, avoiding correct classification by maximizing $L(x', y)$, or launch a targeted attack, forcing specific classification to class t by minimizing $L(x', t)$ (Papernot et al., 2016). We permit the attacker white-box access to F, i.e., so that the attacker knows the internal weights of F.

Evaluation metrics Given a specific ℓ_p distance limit ϵ, the success rate of an untargeted attack is computed as the percentage of benign inputs x from which the attack finds $x' \in \{\tilde{x} \mid F(\tilde{x}) \neq y \wedge \ell_p(\tilde{x}, x) \leq \epsilon\}$, whereas the success rate of a targeted attack is defined as the percentage of benign inputs x where the attack finds $x' \in \{\tilde{x} \mid F(\tilde{x}) = t \wedge \ell_p(\tilde{x}, x) \leq \epsilon\}$. In this paper, we primarily study targeted attacks for $\ell_\infty(x, x') = \max_{i,j,k} \left| x_{i,j,k} - x'_{i,j,k} \right|$, i.e., ℓ_∞ measures the maximum change made across all pixels and channels, where i and j are pixel coordinates, and k is the channel index.

Successful adversarial examples should also be in the same format as benign samples. Images are normally in 8-bit RGB format: every pixel consists of three bytes, three integers $\in [0, 255]$ normalized to floats $\in [0, 1]$. The value of every channel (of every pixel) should be a multiple of $1/255$. We picked ϵ values that are multiples of $1/255$ so that the ℓ_∞ distance limit is in 8-bit RGB format. We noticed that certain attacks (e.g., PGD) could perturb an adversarial examples into failed ones after quantization. Hence, in each attack iteration i from starting sample x, we produced an 8-bit RGB format copy of the current adversarial example x'_i using the formula

$$x_{test} = round(x'_i * 255)/255$$

and clipped x_{test} to be within both $[0, 1]$ and $[x - \epsilon, x + \epsilon]$ (we denote this operation by $clip(x_{test})$), projecting it onto the ℓ_∞ ball that is also bounded by the range of values of valid images. We then classified $clip(x_{test})$ using the model and compared the output with the target class. If they matched, we stopped perturbing this example and counted the attack as successful. Otherwise, the attack continued normally, with x'_i remaining in the continuous domain.

2.2. Attack Methods

We now introduce prominent established attacks.

PGD An improved version of fast gradient-sign method (Goodfellow et al., 2015), the projected gradient descent (PGD) attack (Madry et al., 2018), which iteratively calculates a projection on the imperfect ϵ ball around the benign source image so that the adversarial example is a valid image and within the max distance limit:

$$x'_{i+1} = clip(x'_i - \alpha \cdot sign(\frac{\partial L(x'_i, t)}{\partial x'_i})) \quad (1)$$

where α controls how much perturbation would be applied in each iteration and loosens the linear assumption of models for PGD. By default, PGD is configured to run for 40

iterations, and the step size of each iteration, α, is set to .01, where the values of each channel (of each pixel) in the images are normalized to $[0, 1]$. The default loss function used in PGD is the CE loss.

Auto-PGD Croce and Hein (2020b) were able to boost PGD's performance by intelligently setting its parameters (e.g., the step size α). Except for the number of iterations, the algorithm they proposed, Auto-PGD, does not require parameter tuning. Croce and Hein also showed that the loss function in PGD can markedly influence the attack's success rate. By default, Auto-PGD runs with the Difference of Logits Ratio (DLR) loss:

$$L_{DLR} = \frac{Z_t - Z_y}{Z_{\pi 1} - 0.5Z_{\pi 3} - 0.5Z_{\pi 4}} \tag{2}$$

where Z_πs are logits sorted from largest to smallest. Another commonly used loss function is the Carlini-Wagner (CW) loss (Carlini & Wagner, 2017a):

$$L_{CW} = -Z_t + max_{i \neq t} Z_i \tag{3}$$

L_{CW} and L_{DLR} both try to make the logit of the target class larger than other logits, which would cause the model to assign the highest probability to the target class.

2.3. Defenses

Researchers have proposed a variety of defenses to mitigate adversarial examples. Regarded as one of the strongest defenses (Akhtar et al., 2021), *adversarial training* augments the training process with correctly labeled adversarial examples to enhance models' robustness against them (e.g., (Goodfellow et al., 2015; Kurakin et al., 2017b; Madry et al., 2018; Shafahi et al., 2019; Wang et al., 2020)). As is common in related work (Croce & Hein, 2020b; Dong et al., 2018; Madry et al., 2018; Uesato et al., 2018; Xiao et al., 2018), we evaluate our proposed attacks against adversarially trained neural networks (see §5).

3. A Stronger Loss Function

In this section, we describe how we enhanced Auto-PGD's performance by improving its loss function. We first present an example scenario where a previous best-performing loss function falls short (§3.1). Then we describe a new loss function that mitigates the previous loss function's weaknesses (§3.2). We report on the performance of the new loss function in §6.

3.1. Investigating Established Loss Functions

To produce a baseline for comparison, we ran Auto-PGD for 100 iterations, as it is used in previous work (Croce & Hein, 2020b), with three loss functions: CW, DLR, and CE.

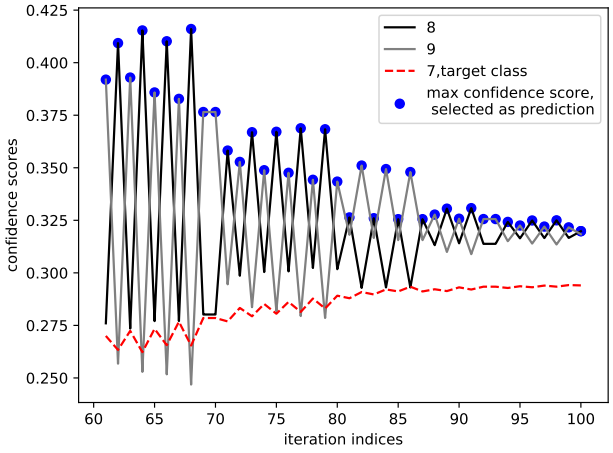

Figure 1: Confidence scores of three classes throughout 40 iterations of the Auto-PGD attack using CW loss to perturb a CIFAR10 image to class 7. Classes 8 and 9 took turns to have the highest confidence score, and the attack ultimately failed to produce a successful targeted adversarial example.

We found that CW loss performed the best against six out of seven CIFAR10 models as well as the ImageNet model we used. Still, we identified specific instances in which Auto-PGD with CW loss failed to find adversarial examples. One of them is demonstrated in Fig. 1. We identified that these failures can be often explained by the definition of CW loss. CW loss tries to increase the difference between the logit of the target class and the highest among the logits of the non-target classes; however, *which* non-target class has the highest logit may change from iteration to iteration. Namely, decreasing the value of the logit of the highest non-target class might simply cause the logit of *another* non-target class to increase and become the highest in the next iteration. Thus, the target class might never have a chance to have the highest logit and the attack might never succeed, as illustrated in Fig. 1.

We noticed that such failures can also happen with more than two non-target classes taking turns having the highest confidence scores. We used the maximum number of times that the prediction was *changed* to a certain class before the perturbation first succeeds or the attack reaches the maximum number of iterations as a metric to capture how often the behavior in Fig. 1 occurs. For 226 of 512 images sampled from CIFAR10, there is some ϵ at which the prediction was *changed* to the same class at least 10 times when Auto-PGD with CW loss attacked the DSL+20 (Ding et al., 2020) defense. We observed the same phenomenon with 265 of 512 samples on the WRK20 (Wong et al., 2020) defense. More details could be found in App. C.

3.2. A New Loss Function

The example in Fig. 1 shows how a loss function that does not completely align with the attack's intent can sometimes fail to produce successful adversarial examples. Intuitively, the attacker's goal is to produce a perturbation for which the target class' logits and probabilities are higher than those of *all* other classes, even if by only a miniscule amount, so that the target class would be selected as the prediction.

We capture this intuition by designing a new loss function containing terms representing each logit. More precisely, the loss function is the sum over all such terms (each representing a logit), where each term's value is determined as follows. The term is zero if the logit is smaller than the logit of the target class, as the adversary has no interest in directly decreasing small logits that do not influence the model's prediction. Alternately, the term is positive if the logit does not belong to the target class and is larger than or equal to the target class's logit, as the adversary has to decrease all non-target logits to make a successful perturbation. By minimizing this loss function, the adversary aims to decrease all the positive terms simultaneously and consequentially decrease *all* the logits that are higher than the logit of the target class in the same iteration.

We define this loss function, the Minimal Difference (MD) loss, as:

$$\sum_i ReLU(Z_i + \delta - Z_t) \tag{4}$$

where ReLU is the rectified linear unit function, Z_t is the logit of the target class, and Z_i are the logits of each class (as described in §2.1). δ is a minimal value introduced to mitigate the cases where non-target classes have equal logits with the logit of the target class. We set δ to $1\text{e}{-15}$ in our implementation, because $1.0 - (1\text{e}{-16}) = 1.0$ in Python due to finite arithmetic. Z_t is not excluded from Z_i as $ReLU(Z_t + \delta - Z_t) = \delta$ is always a constant and has no effect on the back-propagation gradients.

In contrast to CW loss, MD loss aims to decrease *all* logits that do not belong to the target class and are higher than the logit of the target class, rather than only the largest of the non-target-class logits. Logits that meet the following three requirements are more likely to decrease between iterations if the attacker is using MD loss instead of CW loss: (1) they do not belong to the target class; (2) are higher than the logit of the target class; and (3) are not the largest logit. Hence, with MD loss the behavior demonstrated in Fig. 1 (i.e., non-target logits alternating at being the highest logit) is unlikely to occur and, intuitively, the attacker is more likely to succeed. We found that MD loss reduced the maximum number of times that the prediction was changed to a different class at every value of ϵ (see App. C) and confirmed the statistical significance of this result with a Wilcoxon signed-rank test (Wilcoxon, 1945) (details in App. B). We show in §6 that using MD loss instead of CW loss consequently improves Auto-PGD's ability to find adversarial examples.

4. A Stronger Attack Method

In this section, we describe our new attack method, Constrained Gradient Descent (CGD). We start by explaining the intuition that drives the design and an enhancement (§4.1–4.2), and then detail our algorithm (§4.3). We report on the performance of the new attack in §6.

4.1. Learning to Stay Within the Distance Limit

PGD attacks, including Auto-PGD, enforce the ℓ_∞ distance limit by executing a $clip(\cdot)$ computation in each iteration, eliminating any perturbations made outside the limit. However, the loss function used in PGD attacks does not take into account that $clip(\cdot)$ will be used. Hence, in PGD attacks, gradients, which are derivatives of loss functions against the current perturbation, may not accurately direct the perturbation. The gradients may push the attack toward perturbations outside the ℓ_∞ distance limit, while successful perturbations within the ℓ_∞ distance limit may lie in other directions. Previous work has shown that attacks could *learn to minimize* their distance limit from the original benign image (Carlini & Wagner, 2017a; Szegedy et al., 2014). We propose a new loss function that helps attacks *learn to stay within* a fixed ℓ_∞ distance limit, and then propose a new attack that utilizes this loss function. We next describe how we include the ℓ_∞ distance limit as part of the new loss function.

Defining boundaries Before starting to create a perturbation x, we know that for an attack to be valid the upper and lower boundaries of each channel $k \in \{0, 1, 2\}$ of pixel (i, j) in the final perturbation are

$$bnd^{upper}_{i,j,k}(x) = \min(x_{i,j,k} + \epsilon, 1) \tag{5}$$

and

$$bnd^{lower}_{i,j,k}(x) = \max(x_{i,j,k} - \epsilon, 0) \tag{6}$$

These two boundaries are fixed throughout the attack process. As $x_{i,j,k} \in [0, 1]$, in each channel (of each pixel), in any iteration of an attack the updated perturbation could potentially exceed at most one boundary at a time (for any one channel of any pixel). Thus, we can compute the distance, for each channel (of each pixel), by which the current perturbation x' goes over the boundary. We call this distance $Overrun_{i,j,k}(x')$ and define it as:

$$ReLU(x'_{i,j,k} - bnd^{upper}_{i,j,k}(x)) + ReLU(bnd^{lower}_{i,j,k}(x) - x'_{i,j,k}) \tag{7}$$

$Overrun_{i,j,k}(x')$ is 0 if x' stays within the ϵ-boundary for channel k of pixel (i, j).

Penalizing exceeding boundaries We next add to the loss function the following term.

$$L_{bnd} = \sum_{i,j,k} (Overrun_{i,j,k}(x')^2) \tag{8}$$

$Overrun_{i,j,k}(x')$ is squared because when taking the derivative, $\frac{\partial L_{bnd}}{\partial x'_{i,j,k}}$ is proportional to $Overrun_{i,j,k}(x')$ and could direct the attack to learn to stay within the boundary. Our new loss function is now:

$$w * L_{cls} + (1 - w) * L_{bnd} \tag{9}$$

where L_{cls} is MD-loss (as described in §3) and $w \in [0, 1]$ is a weighting parameter. By decreasing w and weighting L_{bnd} more heavily, we guide the attack to gradually move the perturbation in a direction that leads to the target class while not crossing the boundary, ultimately creating a valid adversarial example after rounding and clipping.

Leveraging magnitude of gradients We also change the way that the attack iteratively updates the perturbation. To the best of our knowledge, previous ℓ_∞ attacks that use gradients, $\frac{\partial Loss}{\partial x'_{i,j,k}}$, only use the sign of these gradients to generate adversarial examples (e.g., (Goodfellow et al., 2015; Kurakin et al., 2017a; Madry et al., 2018)), although some use momentum along with the sign (Croce & Hein, 2020b; Gowal et al., 2019). However, the magnitude of the gradients also conveys information about the amount by which it is helpful to change a channel (of a pixel). Leveraging the magnitude of the gradients also helps avoid artificially (and unhelpfully) large step sizes that would make the perturbations step over the boundary, which we observed in, e.g., PGD attacks. Auto-PGD uses an explicit momentum term as well as gradients when computing the changes to be made to the candidate adversarial example in each iteration. In contrast, we compute these changes via an Adam optimizer (Kingma & Ba, 2015), which internally uses momentum; other optimizers may also be adequate.

4.2. Driving out of Local Minima

While L_{bnd} accounts for any channel (of any pixel) potentially crossing the ϵ-boundary, we observed that attacks could become trapped in local minima of L_{bnd} that occur when most channels are within the boundary but a small number is far beyond the boundary. To prevent this, we set a threshold distance outside the upper and lower ϵ boundaries, and we decay this threshold gradually as the attack progresses. As L_{cls} decreases, we desire the attack to also reduce $Overrun$, ultimately to zero. The decreasing threshold encourages this by increasing the relative weight of L_{bnd}. Model-specific constants, namely pre-defined fixed ratios and checkpoints, control where the threshold starts and how fast it decays. If the current perturbation is outside the threshold, we halve the weight w of L_{cls} (in Eqn. 9) to increasingly urge the attack to stay within the boundary.

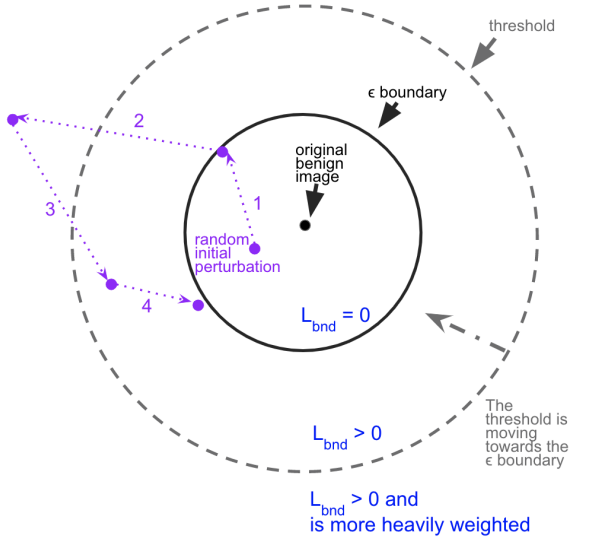

Figure 2: This is an example path of a CGD attack with a ℓ_∞ distance limit. We start with a random initial perturbation. In stage 1, we push the current perturbation to the ϵ boundary. In stage 2, the current perturbation moves beyond the threshold and $L_{bnd} > 0$. In stage 3, the current perturbation is pushed inside the threshold as L_{bnd} is more heavily weighted. In stage 4, the current perturbation moves closer to the ϵ boundary, as does the threshold.

We ran grid searches to choose the starting threshold and decay interval for each defense. We examined starting thresholds $\in [.5\epsilon, 9\epsilon]$, starting the weighting parameter of the loss function $w \in [0.01, 0.5]$, and decay intervals $\in [5, 30]$. For all defenses, the optimal starting weighting parameter w was 0.1 and the optimal decay interval was every 15 iterations. The optimal starting threshold was 8ϵ for SIE+20, 5ϵ for DSL+20, and 1.5ϵ for all other defenses (see §5).

4.3. The Constrained Gradient Descent Algorithm

Combining all the above, we define a new attack: *Constrained Gradient Descent (CGD)*. Fig. 2 illustrates an example path of CGD where the attack seeks to satisfy the ℓ_∞ distance limit.

We start the attack with a random initial perturbation to better explore the space of possible adversarial examples, as this was shown to be helpful in prior work (Croce & Hein, 2020a; Madry et al., 2018; Mosbach et al., 2018). The attack has four stages. In stage 1, we move each channel (of each pixel) by ϵ in the direction of the gradients. This is a quick way to take a substantial step in the direction of the target class. In stage 2, the candidate perturbation continues to move toward the target class, and potentially moves outside

the threshold, as the loss function is dominated by L_{cls}. If the candidate perturbation moves beyond the threshold, the algorithm moves to stage 3: since the candidate perturbation is outside the threshold, L_{bnd} is more heavily weighted, which pushes the candidate perturbation back inside the threshold. After a fixed number of iterations, the algorithm enters stage 4, in which the threshold itself moves toward the ϵ boundary, thus forcing the candidate perturbation to move closer to the ϵ boundary. The attack could succeed in any stage. Pseudocode with line-by-line descriptions can be found in App. A.

5. Evaluation Setup

Factors other than the algorithm, such as the random initialization chosen (Tashiro et al., 2020) and which classes are targeted, can also influence attacks' performance. Here we summarize how we set up experiments to enable meaningful and fair comparisons; more details can be found in App. D.

Benchmarks Because adversarial training is regarded as a strong defense, we evaluated attacks against adversarially trained models, in line with prior work (see §2.3). Specifically, we used seven pre-established adversarially trained models for CIFAR10: CRS+19 (Carmon et al., 2019), DSL+20 (Ding et al., 2020), HLM19 (Hendrycks et al., 2019), SWM+20 (Sehwag et al., 2020), WRK20 (Wong et al., 2020), WXW20 (Wu et al., 2020) and WZY+20 (Wang et al., 2020); and two versions of SIE+20 (Salman et al., 2020), pre-established and publicly available adversarially trained models on ImageNet.

Experiment setup Croce et al. found that PGD attacks find more adversarial examples the more iterations they run (Croce & Hein, 2020b). Auto-PGD declares the number of iterations as its only parameter. In this work, we ran all attacks for 100 iterations—the default configuration of Auto-PGD (Croce & Hein, 2020b)—to fairly compare the attack methods. For CIFAR10, we measured the success rate against seven defenses, using the same target, 20 random initial perturbations, and two ϵ values per image, for a total of 280 sets of 10,000 attack attempts. For ImageNet, we used five random initializations, five targets, and two ϵ values per image, thus resulting in 50 sets of 50,000 attack attempts.

6. Evaluation Results

In this section, following the setup described in §5, we compare Auto-PGD using our MD loss with Auto-PGD using previously established loss functions and also our CGD attack with Auto-PGD. We first report on raw results (§6.1) and then on the statistical tests we performed (§6.2) to demonstrate that CGD outperformed the previously best Auto-PGD with statistical significance. We also compare the time cost of attacks (§6.3) and discuss the uniqueness of the adversarial examples generated (§6.4).

6.1. Raw Results

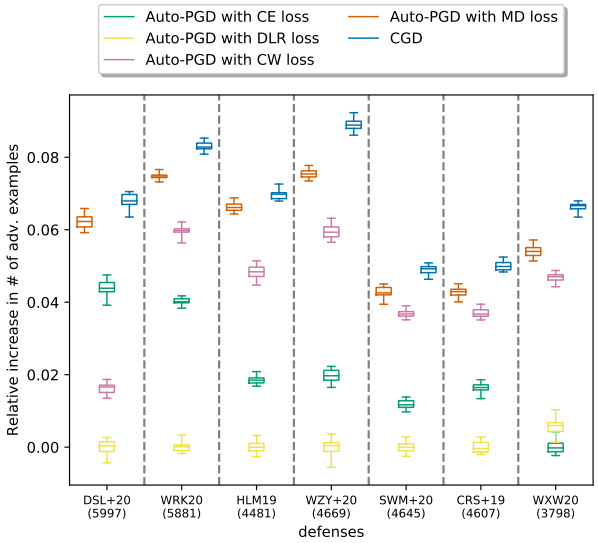

Figure 3: The relative improvement in the number of adversarial examples found by attacks on different defenses compared to the worst-performing attack. Experiments were performed using 10,000 images from the test set of CIFAR10, $\epsilon = 16/255$, 20 different random initial perturbations, and a fixed random target offset. The worst-performing attack for each defense (with a median of 0) was selected as the baseline. The y-axis denotes the improvement compared to the average performance of the baseline. For example, 0.08 on the y-axis indicates 8% more adversarial examples found compared to the baseline. The number in parentheses under each defense is the average number of times which the baseline succeeded out of 10,000 attempts.

As described in §5, we made 280 sets of 10,000 attack attempts on CIFAR10 and 50 sets of 50,000 attack attempts on ImageNet. We compared our two improvements, Auto-PGD using our MD loss (§3) and CGD (§4), to Auto-PGD using three pre-established loss functions: CE loss, DLR loss, CW loss. Implementations of Auto-PGD with CE loss and DLR loss are the ones published by the authors (Croce & Hein, 2020b). We performed this comparison on two datasets and multiple defenses and values of ϵ (see §5).

On average, Auto-PGD with MD loss found more adversarial examples than Auto-PGD with any of the three other loss functions , thus demonstrating the benefits of the MD loss compared to previous conventional loss functions. Additionally, CGD performed better than Auto-PGD with MD loss, further demonstrating the advantages of the CGD attack strategy to satisfy the ℓ_p-bound constraints compared to ad

hoc clipping. The ranking of attacks other than Auto-PGD with MD loss and CGD varied slightly depending on the defense.

Fig. 3 shows the relative number of adversarial examples these attack methods found on CIFAR10 with $\epsilon = 16/255$. Among the previously proposed attacks (Auto-PGD with one of CE, DLR, or CW loss), which performed least well and which performed best varied by defense. The best attacks were on average 1.6–6.0% better than the baseline attacks. Auto-PGD with MD loss was on average 0.6–4.5% better than the best performing Auto-PGD that did not use MD loss; and CGD was on average 1.2–5.1% better than the best performing Auto-PGD that did not use MD loss. CGD outperformed Auto-PGD with MD loss in 139 out of 140 sets of attempts and outperformed Auto-PGD (with any other loss function) in all of the sets of attempts. Depending on the defense, the ranking of the other attack methods changed, but CGD and Auto-PGD with MD loss always performed better than other attack methods. We observed similar results with different values of ϵ and when using the ImageNet dataset, as shown in App. E. When evaluated against SIE+20 on the ImageNet dataset with $\epsilon = 4/255$, CGD was 11.0% better than Auto-PGD with CW loss (the previous best Auto-PGD). When using $\epsilon = 8/255$, CGD was 13.6% better than Auto-PGD with CW loss against SIE+20 on ImageNet.

6.2. Statistical Analysis

We also performed statistical analysis to compare the performances of different attack methods. We defined a variable $ConditionalSuccess_i$ for an image i as the total number of successful perturbations made by an attack with all random initial perturbations, and all the random $target_offset$s we tried given the specific dataset, defense, and ϵ. $ConditionalSuccess_i \in [0, 20]$ on CIFAR10 while $ConditionalSuccess_i \in [0, 25]$ on ImageNet. Each $ConditionalSuccess_i$ is independent. We used the Wilcoxon signed rank test to compare $ConditionalSuccess_i$ of CGD and Auto-PGD with MD loss.

We performed the one-sided Wilcoxon signed rank test (Wilcoxon, 1945) with the null hypotheses that Auto-PGD with MD loss had equal or better performance than CGD for each combination of ϵ, dataset, and defense that we tried. Overall, we conducted 16 statistical tests, for the 16 different combinations we had. To account for the multiple tests, we used Bonferroni correction to adjust the confidence level α to $.05/16 = 0.003125$. We found the p-values are below α in 11 out of 16 tests. Namely, CGD performed statistically significantly better than Auto-PGD with MD loss across 11 combinations of ϵ, dataset, and defense that we tried.

We performed a similar one-sided Wilcoxon signed rank test (Wilcoxon, 1945) with the null hypotheses that the best performing attack among Auto-PGD using the DLR loss, CW loss, and CE loss performed equal to or better than CGD in each combination of value of ϵ, dataset, and defense that we tried. Again, we used the adjusted normal approximation of the test statistic and Bonferroni corrections. All the p-values were far smaller than $\alpha = 0.003125$, and so we reject the null hypotheses in all 16 cases, hence demonstrating that CGD significantly outperformed Auto-PGD with the losses proposed in prior work across each combination of ϵ, dataset, and defense that we tried. More details of these tests can be found in App. F.

6.3. Time Complexity

We ran all attacks for 100 iterations, as described in §5, and conducted 30 and 100 time measurements per attack-defense pair for the CIFAR10 and ImageNet datasets, respectively. The results are shown in Tab. 1.

Table 1: The average time in seconds used to perturb batches of 512 images from CIFAR10 or 10 images from ImageNet, using NVIDIA GeForce RTX 3090 GPUs. There are two versions of SIE+20, trained with $\epsilon = 4/255$ (SIE+20-4) and $\epsilon = 8/255$ (SIE+20-8).

attack methods		Auto -PGD	Auto -PGD	Auto -PGD	Auto -PGD	CGD
loss		CE	CW	DLR	MD	
dataset	*defense*	*time (seconds)*				
CIFAR10	DSL+20	26.4	26.9	26.9	26.8	22.0
	WRK20	17.0	17.7	17.6	17.5	14.5
	HLM19	115.8	117.0	115.9	116.7	100.9
	WZY+20	116.5	116.6	117.1	116.2	103.3
	SWM+20	115.0	115.2	115.6	114.8	100.1
	CRS+19	116.2	116.7	116.7	116.3	101.5
	WXW20	116.4	116.7	116.4	116.3	102.6
ImageNet	SIE+20-4	7.7	7.7	7.8	7.8	6.4
	SIE+20-8	7.7	7.7	7.8	7.8	6.4

We found that CGD was on average faster than all Auto-PGD attacks against all defenses by 11.41–18.76%. We confirmed this relationship with one-sided Wilcoxon tests (App. B). As we have nine defenses and four baseline attacks, we used Bonferroni correction to adjust the confidence level α to $.05/36 = 0.0014$.

6.4. Uniqueness of Attacks

As we observed in §6.1–6.2 that CGD found more adversarial examples than other attack methods did in most of the sets of attempts, we wondered if the adversarial examples other attacks found could be a subset of those CGD found. However, we discovered that among the attack methods we tried, each of them found a slightly different set of successful adversarial examples, as shown in Fig. 4. In addition, each attack succeeded in finding an adversarial example for

some specific input for which no other attack succeeded, as shown in Fig. 5. We observed the same phenomenon across attack methods, values of ϵ, defenses, and datasets. More details can be found in App. I.

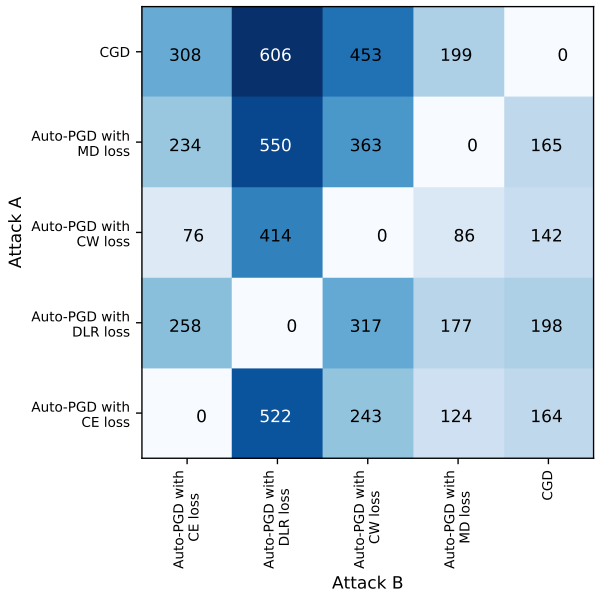

Figure 4: The average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$ against the DSL+20 (Ding et al., 2020) defense, rounded to whole numbers.

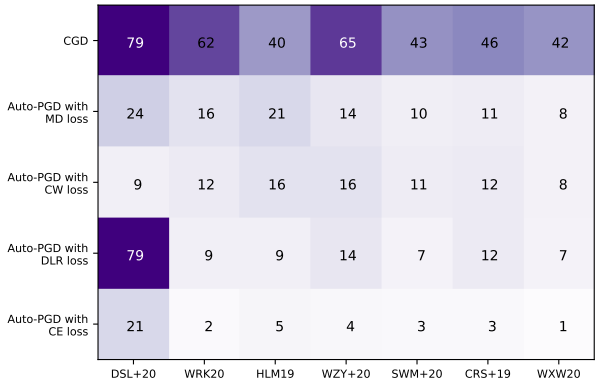

Figure 5: The average number of successful adversarial examples found by an attack, but not by any of other attack methods, against each defense in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$, rounded to whole numbers.

At the same time, when the same attack method is given different random initial perturbations, the attack always found a slightly different set of adversarial examples, within the 20 random initial perturbations we tried (an example is shown in Fig. 6). We observed the same phenomenon across attack methods, values of ϵ, and defenses on CIFAR10. More details can be found in App. I.

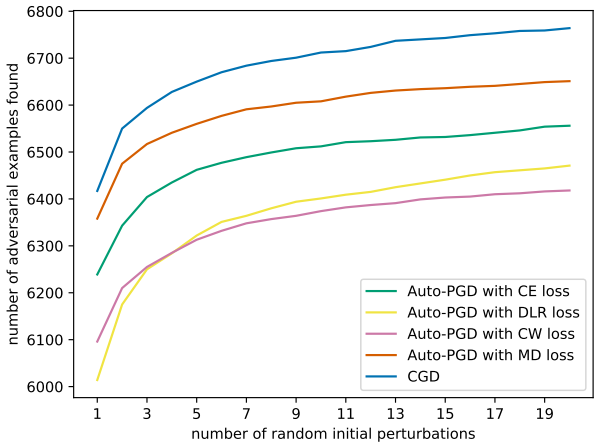

Figure 6: The image shows the number of adversarial examples each of the attacks found when using the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 16/255$, against the DSL+20 (Ding et al., 2020) defense.

7. Discussion

Here we discuss the potential to use CGD as a *framework* for attacks (§7.1), e.g., with different distance metrics and loss functions. We demonstrate one such use, where we instantiate CGD for untargeted attacks (§7.2).

7.1. CGD as a Framework

In §6 we showed that CGD outperformed the previous best attack in targeted tasks, with statistical significance and substantial effect size. Similarly to how Auto-PGD is a member of the PGD family attacks, the specific attack we explore in this paper could be viewed as member of a broader CGD family. Auto-PGD improves the performance of PGD attacks by using alternative loss functions and wisely tuning parameters; similar tweaking could also apply to CGD attacks: using alternative loss functions and wisely tuning parameters could yield a stronger attack within the CGD family. Meanwhile, as the loss function of CGD can be separated into two components, L_{bnd} and L_{cls}, each of those could be tuned. We demonstrated that CGD implemented with specific loss functions and parameters outperformed the previous best attack; other variants of CGD could perform better yet. CGD might also be extended to other distance metrics besides ℓ_∞ and also to untargeted attacks. In general, our work opens the door for future work to find stronger attacks using CGD as a framework.

7.2. Applying CGD to Untargeted Attacks

We demonstrated that CGD outperformed the previous best white-box targeted attacks in §6. Here we explore an untargeted variant of CGD as an example of extending CGD to other types of attacks, and, specifically, how other loss functions can be incorporated in the general CGD approach.

Previous works find untargeted adversarial examples by using one of several loss functions. One example is cross entropy loss, which we described in §2.1: $L_{CE} = -\log P_y = -Z_y + \log(\sum_{j=1}^{K} e^{Z_j})$. Other work proposed the untargeted attack version of the Difference of Logits Ratio loss (DLR loss):

$$L_{DLR} = \frac{Z_y - max_{i \neq y} Z_i}{Z_{\pi 1} - Z_{\pi 3}} \tag{10}$$

and showed that Auto-PGD with L_{DLR} finds more adversarial examples than Auto-PGD with L_{CE} (Croce & Hein, 2020b). Yet other work uses the untargeted version of CW loss (Carlini & Wagner, 2017a):

$$L_{CW} = -Z_y + max_{i \neq y} Z_i \tag{11}$$

Previous works show that by iteratively maximizing these loss functions the adversary can find untargeted adversarial examples (Carlini & Wagner, 2017a; Croce & Hein, 2020b).

Because the concept of staying within the ℓ_∞ distance limit is the same in targeted and untargeted attacks, the loss component for capturing the task of staying within the ℓ_∞ distance limit, L_{cls}, is the same as described in §4. L_{cls} decreases to 0 if and only if the adversarial example is within the ℓ_∞ distance limit. Hence, we design the loss component that captures the task of forcing misclassification, such as the MD loss, to also decrease to 0 when the adversary succeeds. The purpose of this design is to have the overall loss function, which is a weighted sum of the two components, decrease to 0 if and only both tasks have been successfully completed. To create the loss component that captures the task of avoiding correct classification, we define a variant of the CW loss as follows:

$$L_{CW^*} = ReLU(Z_y + \delta - max_{i \neq y} Z_i) \tag{12}$$

where δ is a minimal value, set to $1e{-}15$ (see §3.2). We replace MD loss in Alg. 1 with L_{CW^*} as L_{cls} to obtain the untargeted variant of the CGD attack, $\text{CGD}_{\text{untarg}}$.

Minimizing L_{CW^*} achieves the same result as maximizing L_{CW}: when $Z_y \geq max_{i \neq y} Z_i$, both loss functions minimize $Z_y - max_{i \neq y} Z_i$; and when $Z_y < max_{i \neq y} Z_i$, the untargeted attack has already succeeded. Hence, an Auto-PGD that minimizes L_{CW^*} and an Auto-PGD that maximizes L_{CW} behave exactly the same on each image; we report results only for the former.

We ran Auto-PGD with L_{DLR}, Auto-PGD with L_{CW^*}, and $\text{CGD}_{\text{untarg}}$ to compare their performance. Details of the setup can be found in App. G. On average, $\text{CGD}_{\text{untarg}}$ outperformed Auto-PGD with L_{DLR} and Auto-PGD with L_{CW^*}. $\text{CGD}_{\text{untarg}}$ outperformed the next best method, Auto-PGD with L_{CW^*} in 28 out of the 35 sets of attempts at $\epsilon = 4/255$, in 31 out of the 35 sets of attempts at $\epsilon = 8/255$, and in all 35 sets of attempts at $\epsilon = 16/255$. More details can be found in App. H.

8. Conclusion

In this work we improved a previously established white-box, targeted evasion attack by using a new loss function. We also proposed a yet stronger attack that learns to approach and explore the ϵ-boundary. We demonstrated the efficacy of both the new loss function and the new attack on two datasets (CIFAR10 and ImageNet), for multiple values of ϵ, and against multiple defenses; in all cases, our methods outperformed the best of the attacks we compared against, finding targeted adversarial examples more successfully while taking significantly less time to run. Finally, we showed how to use our new attack method as a general framework for attacks and demonstrated its utility by instantiating it into a stronger *untargeted* attack.

Acknowledgments

This paper was supported in part by the Department of Defense under contract FA8702-15-D-0002; by NSF grants 1801391, 2112562, and 2113345; by the National Security Agency under award H9823018D0008; by the Maof prize for excellent young faculty; and by Len Blavatnik and the Blavatnik Family foundation.

References

Akhtar, N., Mian, A. S., Kardan, N., and Shah, M. Advances in adversarial attacks and defenses in computer vision: A survey. *ArXiv*, 2021. https://arxiv.org/abs/2108.00401.

Athalye, A. and Carlini, N. On the robustness of the cvpr 2018 white-box adversarial example defenses. *ArXiv*, abs/1804.03286, 2018.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, 2018. https://arxiv.org/abs/1802.00420.

Bai Li, Shiqi Wang, S. J. and Carin, L. Towards understanding fast adversarial training. *ArXiv*, abs/2006.03089, 2020.

Bonnet, B., Furon, T., and Bas, P. What if adversarial samples were digital images? In *IH&MMSEC*, 2020.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, volume 1, pp. 39–57, 2017a. https://arxiv.org/abs/1608.04644.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security (AISec)*, 2017b.

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. Unlabeled data improves adversarial robustness. In *Conference on Neural Information Processing Systems*, 2019. https://arxiv.org/abs/1905.13736.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 2019.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2020a. https://arxiv.org/abs/1907.02044.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020b. https://arxiv.org/abs/2003.01690.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. arXiv preprint 2010.09670, June 2021. URL https://arxiv.org/abs/2010.09670.

Cureton, E. E. The normal approximation to the signed-rank sampling distribution when zero differences are present. In *Journal of the American Statistical Association*, volume 62, pp. 1068–1069, 1967. https://www.tandfonline.com/doi/abs/10.1080/01621459.1967.10500917.

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020. https://openreview.net/forum?id=HkeryxBtPB.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *Conference on Computer Vision and Pattern Recognition*, 2018. https://arxiv.org/abs/1710.06081.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. https://arxiv.org/abs/1412.6572.

Gowal, S., Uesato, J., Qin, C., Huang, P.-S., Mann, T., and Kohli, P. An alternative surrogate loss for PGD-based adversarial testing. *arXiv:1910.09338*, 2019.

Gowal, S., Qin, C., Uesato, J., Mann, T., and Kohli, P. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

Guo, C., Rana, M., Cisse, M., and Van Der Maaten, L. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

He, W., Wei, J., Chen, X., Carlini, N., and Song, D. Adversarial example defense: Ensembles of weak defenses are not strong. In *11th {USENIX} workshop on offensive technologies ({WOOT} 17)*, 2017.

Hendrycks, D., Lee, K., and Mazeika, M. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, 2019. https://arxiv.org/abs/1901.09960.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. https://arxiv.org/abs/1412.6980.

Kumar, A., Levine, A., Goldstein, T., and Feizi, S. Curse of dimensionality on randomized smoothing for certifiable robustness. In *International Conference on Machine Learning*, 2020.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. In *International Conference on Learning Representations Workshop*, 2017a. https://arxiv.org/abs/1607.02533.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017b. https://arxiv.org/abs/1611.01236.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, 2019.

Liao, F., Liang, M., Dong, Y., Pang, T., Zhu, J., and Hu, X. Defense against adversarial attacks using high-level representation guided denoiser. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787, 2018.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. https://arxiv.org/pdf/1706.06083.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017. https://arxiv.org/abs/1702.04267.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Conference on Computer Vision and Pattern Recognition*, 2016. https://arxiv.org/abs/1511.04599.

Mosbach, M., Andriushchenko, M., Trost, T. A., Hein, M., and Klakow, D. Logit pairing methods can fool gradient-based attacks. In *NeurIPS 2018 Workshop on Security in Machine Learning*, 2018. https://arxiv.org/abs/1810.12042.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pp. 372–387, 2016. doi: 10.1109/EuroSP.2016.36.

Pratt, J. W. Remarks on zeros and ties in the wilcoxon signed rank procedures. *Journal of the American Statistical Association*, 54, 1959.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018. https://arxiv.org/abs/1801.09344.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? In *Conference on Neural Information Processing Systems*, 2020. https://arxiv.org/abs/2007.08489.

Samangouei, P., Kabkab, M., and Chellappa, R. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. Hydra: Pruning adversarially robust neural networks. In *Conference on Neural Information Processing Systems*, 2020. https://arxiv.org/abs/2002.10509.

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! In *Conference on Neural Information Processing Systems*, 2019. https://arxiv.org/abs/1904.12843.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018. https://arxiv.org/abs/1710.10766.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. https://arxiv.org/abs/1312.6199.

Tashiro, Y., Song, Y., and Ermon, S. Diversity can be transferred: Output diversification for white-and black-box attacks. In *Conference on Neural Information Processing Systems*, 2020. .

Uesato, J., O'Donoghue, B., van den Oord, A., and Kohli, P. Adversarial risk and the dangers of evaluating against weak attacks. *International Conference on Machine Learning*, 2018. https://arxiv.org/abs/1802.05666.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020. https://openreview.net/forum?id=rklOg6EFwS.

Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1, 1945.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2018.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. https://arxiv.org/abs/2001.03994.

Wu, D., tao Xia, S., and Wang, Y. Adversarial weight perturbation helps robust generalization. In *Conference on Neural Information Processing Systems*, 2020. https://arxiv.org/abs/2004.05884.

Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., and Song, D. X. Generating adversarial examples with adversarial networks, 2018.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. In *The Network and Distributed System Security Symposium (NDSS)*, 2018. https://arxiv.org/abs/1704.01155.

Zhang, H., Chen, H., Xiao, C., Li, B., Boning, D. S., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020. https://arxiv.org/abs/1906.06316.

A. The CGD algorithm

Algorithm 1 CGD

Input: x, $model(\cdot)$, $n_{iterations}$, ϵ, $threshold$, $checkpoints$, $target$

1: $x' \leftarrow clip(x + 2 * rand(x.shape) - 1)$
2: $w \leftarrow 0.1$
3: $bnd^{upper}_{i,j,k}(x) \leftarrow min(1, x_{i,j,k} + \epsilon)$
4: $bnd^{lower}_{i,j,k}(x) \leftarrow max(0, x_{i,j,k} - \epsilon)$
5: **for** $iteration \leftarrow 1$ **to** $n_{iterations}$ **do**
6: **if** $iteration \in checkpoints$ **then**
7: $threshold \leftarrow threshold/2$
8: **end if**
9: $Z \leftarrow model(x')$
10: $L_{cls} \leftarrow \sum_i ReLU(Z_i + \delta - Z_t)$
11: $Overrun_{i,j,k}(x') \leftarrow ReLU(x'_{i,j,k} - bnd^{upper}_{i,j,k}(x)) + ReLU(bnd^{lower}_{i,j,k}(x) - x'_{i,j,k})$
12: $L_{bnd} \leftarrow \sum_{i,j,k}(Overrun_{i,j,k}(x')^2)$
13: **if** $\exists i,j,k \in Overrun_{i,j,k}(x') > threshold$ **then**
14: $w \leftarrow w/2$
15: **end if**
16: $loss \leftarrow w * L_{cls} + (1 - w) * L_{bnd}$
17: $gradients \leftarrow \frac{\partial loss}{\partial x'}$
18: **if** $iteration == 1$ **then**
19: $x' \leftarrow clip(x' - \epsilon * sign(gradients))$
20: **else**
21: $changes \leftarrow Adamoptimizer(gradients)$
22: $x' \leftarrow x' - changes$
23: **end if**
24: $x_{test} \leftarrow clip(round(x' * 255)/255)$
25: **if** $argmax(model(x_{test})) == target$ **then**
26: **Return** x_{test}
27: **end if**
28: **end for**
Return *failed attack*

Alg. 1 shows the pseudocode of the *Constrained Gradient Descent* (CGD) algorithm. The inputs to the algorithm are: the benign example x; the $model(\cdot)$ function that emits the logits for given inputs; the number of iterations $n_{iterations}$; the ϵ distance; a set of constants tuned per model, $threshold$ and $checkpoints$; and the target class $target$. We allow the tuning of $threshold$ and $checkpoints$ per model because we are running white-box attacks and hence the adversary has the freedom to choose the attack's parameters per model. All other parameters do not require tuning per model. In line 1, we add a random initial perturbation to the benign sample, moving it to the ϵ boundary. In lines 3 and 4 of the algorithm, before we start the iterations, we compute $bnd^{upper}(x)$ and $bnd^{lower}(x)$. From line 6 to 8, we adjust the threshold based on pre-set constants. We compute the logits Z of the model regarding the current perturbation x' in line 9 and compute the MD loss in line 10. Then

we compute the $Overrun$ in line 11 and the L_{bnd} in line 12, which motivates the adversarial example to move to be within the ℓ_∞ distance limit. From line 13 to line 15, we adjust the weight w, and then use this weight to compute the total loss as a weighted sum in line 16. We compute the gradients from the loss function in line 17 and apply changes from line 18 to line 23. In the first iteration, the candidate perturbation is pushed by a distance of ϵ to more quickly reach the boundary. In later iterations, the Adam optimizer is used to reduce fluctuation. From line 24 to 27 (highlighted in pink), we perform a normal rounding check. This rounding check enforces the successful adversarial example to be both inside the ℓ_∞ distance limit and in 8-bit RGB format. When the candidate perturbation is very close to the ϵ boundary, clipping has minimal effects on the logits, L_{cls}, and probabilities assigned to classes, meaning that an image just outside the ϵ boundry is rounded to become both within the ϵ boundary and potentially a successful attack.

B. Wilcoxon Signed-Rank Test

The Wilcoxon signed-rank test (Wilcoxon, 1945), is a non-parametric test to examine the relationship between two related paired samples. In our experiements, we ran one-sided Wilcoxon signed-rank tests to verify if one set of samples is greater than another with statistical significance. While the Wilcoxon signed-rank test assumes that the two sets differ on each pair of samples, in our case, the samples are equal in some pairs. Previous work suggests a mitigation to enable using the Wilcoxon signed-rank test in such cases: using an adjusted normal approximation of the test statistic of the Wilcoxon signed-rank test instead of using the standard test statistic to compute the p-value (Cureton, 1967), and using the pairs that are equal to produce the rankings but excluding them afterwards (Pratt, 1959). This approach requires more than 25 independent trials; our data satisfiy this requirement. In addition, to account for the multiple tests, we always used Bonferroni correction to adjust the confidence level α.

C. Measurement of Changes in Predictions

As we described in §3, we used the number of times that the prediction was changed to a certain class before the perturbation first succeeds as a metric to capture how often the undesirable behavior (mentioned in §3.1) of the CW loss occurs. We used 512 images pertaining to a diversity of classes from CIFAR10, and perturbed them while targeting randomly picked classes. We ran Auto-PGD with ϵ values from $0.1/255$ to $32/255$ by every $0.1/255$. The results for Auto-PGD with the CW loss are shown in Figs. 7–8, while the results for Auto-PGD with the MD loss are shown in Figs. 9–10. Against both DSL+20 (Ding et al., 2020) and WRK20 (Wong et al., 2020), the MD loss reduced the maximum number of times that the prediction was changed to a class. We also ran a Wilcoxon signed-rank test (see App. B) to determine whether the differences between the losses are statistically significant. Specifically, we conducted 320 statistical tests on each defense, for the 320 different values of ϵ we used. To account for the multiple tests, we used Bonferroni correction to adjust the confidence level α to $.05/320 = 0.00015625$. We found that for 203 out of 320 values of ϵ against DSL+20 and for 309 out of 320 values of ϵ against WRK20, the maximum number of times that the prediction was changed to a class is statistical significantly smaller when using the MD loss compared to the CW loss. Notably, this maximum number of changes is smaller when using the MD loss for every value of ϵ in $[0.8/255, 16.3/255]$ against DSL+20, and for every value of ϵ in $[0.7/255, 27.3/255]$ against WRK20. Both ranges include values of ϵ commonly used in ℓ_∞ attacks.

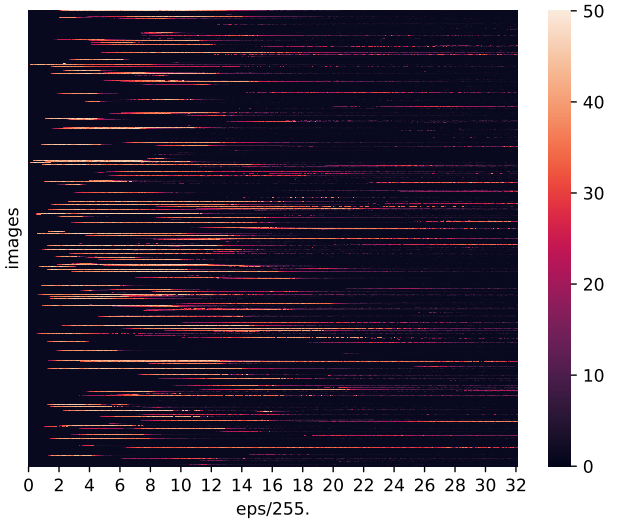

Figure 7: The maximum number of times that the prediction was changed to a class, throughout the process of perturbing each of the 512 image before success using Auto-PGD with CW loss against DSL+20.

D. More Evaluation Setup for Targeted Attacks

In this section, we provide more detail about how we executed reproducible attacks to enable meaningful and fair comparisons.

D.1. Datasets

In this work, we used the CIFAR10 and ImageNet datasets—two standard datasets that are commonly used for classification tasks. Both datasets contain colored images of objects in 8-bit RGB format as described in §2.1. Each image in

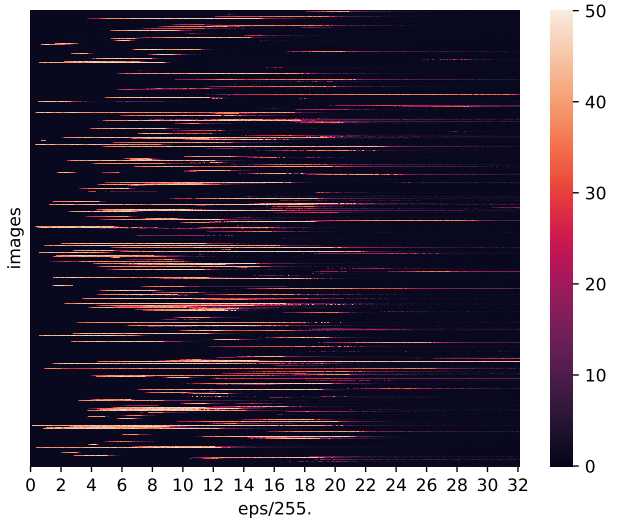

Figure 8: The maximum number of times that the prediction was changed to a class, throughout the process of perturbing each of the 512 image before success using Auto-PGD with CW loss against WRK20.

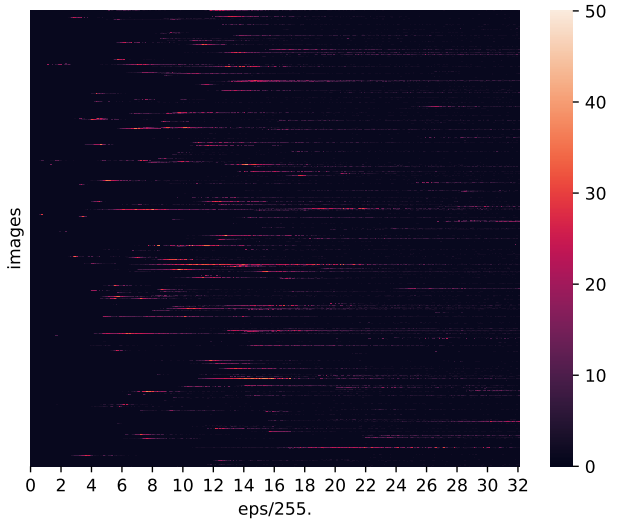

Figure 9: The maximum number of times that the prediction was changed to a class, throughout the process of perturbing each of the 512 image before success using Auto-PGD with MD loss against DSL+20.

CIFAR10 has 32×32 pixels, whereas we resized each image in ImageNet to have 224×224 pixels. We evaluated attacks on the test set of the datasets. CIFAR10 has 10,000 images in its test set and ImageNet has 50,000.

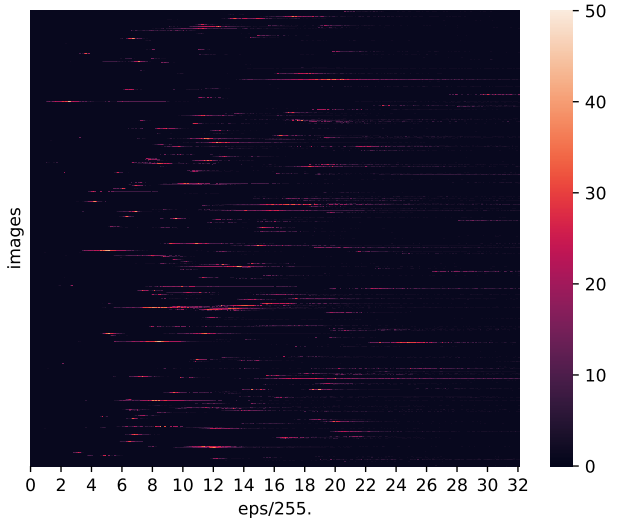

Figure 10: The maximum number of times that the prediction was changed to a class, throughout the process of perturbing each of the 512 image before success using Auto-PGD with MD loss against WRK20.

D.2. Benchmarks

Many defense strategies have been explored. For example, different input transformations have been proposed to remove adversarial perturbations from inputs prior to classification (e.g., (Guo et al., 2018; Liao et al., 2018; Samangouei et al., 2018; Xu et al., 2018)). Unfortunately, the majority of these defenses can be evaded by adaptive attacks that craft adversarial perturbations that survive the transformations (Athalye & Carlini, 2018; Athalye et al., 2018; He et al., 2017). Certain defenses attempt to detect adversarial examples (e.g., (Feinman et al., 2017; Ma et al., 2018; Metzen et al., 2017)). The majority of these defenses are unable to classify adversarial examples correctly, when detected. Moreover, researchers have also found that detection methods can often be evaded by adaptive attacks (Athalye et al., 2018; Carlini & Wagner, 2017b). Finally, certified defenses provide provable accuracy guarantees on adversarial examples (e.g., (Cohen et al., 2019; Lecuyer et al., 2019; Raghunathan et al., 2018; Wong & Kolter, 2018; Zhang et al., 2020)). However, these defenses are often effective for perturbations of smaller norms than adversarial training (Lecuyer et al., 2019), or are ineffective against the threat models we study (e.g., high-dimensional perturbations with bounded ℓ_∞-norms) (Bai Li & Carin, 2020; Kumar et al., 2020).

As we described in §2.3 and §5, adversarial training is regarded as ne of the strongest defenses (Akhtar et al., 2021), and hence we used seven pre-established adversarially trained models for CIFAR10: CRS+19 (Car-

mon et al., 2019), DSL+20 (Ding et al., 2020), HLM19 (Hendrycks et al., 2019), SWM+20 (Sehwag et al., 2020), WRK20 (Wong et al., 2020), WXW20 (Wu et al., 2020) and WZY+20 (Wang et al., 2020). All of these models were trained with $\epsilon = 8/255$ and are publicly available via `RobustBench` (Croce et al., 2021), a standard library for robustness evaluation of neural networks. We selected the models using the following criteria: they rank highly on robustness in the public `RobustBench` evaluation; they have a PyTorch implementation; and they fit into the 11GB memory of our NVIDIA GeForce RTX 2080 GPUs. These models have have also been found to be robust in evaluations other than `RobustBench`'s (Gowal et al., 2020; Rebuffi et al., 2021). The seven models can be categorized into four non-exclusive groups: some propose surrogate loss functions (Ding et al., 2020; Wang et al., 2020; Wu et al., 2020), some use pre-training or semi-supervised learning techniques (Carmon et al., 2019; Hendrycks et al., 2019; Wang et al., 2020), some try to wisely tune the training process (Wong et al., 2020), and some apply pruning to their models (Sehwag et al., 2020). Similarly to prior work (e.g., (Song et al., 2018; Wong et al., 2020)), we evaluated attacks against these models with $\epsilon = 8/255$ and $16/255$. We also found two versions of SIE+20 (Salman et al., 2020), pre-established and publicly available adversarially trained models on ImageNet—one adversarially trained with $\epsilon = 4/255$ and another with $\epsilon = 8/255$. We evaluated attacks against each version of this model with the same ϵ with which it was trained.

D.3. Experiment Setup

We observed that some PyTorch functions could yield unreproducible results in different runs. To keep our measurement results reproducible, we set `torch.backends.cudnn.benchmark` to `False` and `torch.backends.cudnn.deterministic` to `True`. In addition, we found that PGD attacks, including Auto-PGD, always start with a random perturbation and this could slightly influence the result. Hence, we ran these attacks multiple times, each time with a different random seed. We also fixed the batch size so that with the same random seed we got the same random initial perturbation. We used a batch size of 512 images for CIFAR10 and a batch size of 10 images for ImageNet. In addition to the random initialization, we also picked a random target class for each image in the testing sets of datasets which we used to evaluate attacks. Due to limited computing resources, we were not able to run attacks targeting every incorrect class, especially for ImageNet which has 1,000 classes. The target class was intentionally selected to be different from the label, the correct class of the image. We chose the difference $target\_offset$ between the target class and the correct class, using the following formula:

$$target\_offset_i = floor(rand() * (N_{classes} - 1)) + 1 \quad (13)$$

where i is the index of images in the testing set, $i \in [0, 10000)$ for CIFAR10 and $i \in [0, 50000)$ for ImageNet, and $N_{classes}$ is the number of classes, 10 for CIFAR10 and 1,000 for ImageNet. The $rand()$ function generates a float $\in [0, 1)$. The target class was

$$t_i = (y_i + target\_offset_i) \bmod N_{classes} \quad (14)$$

for $i \in [0, N_{images})$. We only used one random seed (specifically, 0) for the $target\_offset$ in CIFAR10, as on average there are $10000/(9 * 10)$ images in each source-target class pair, whereas we used five random seeds (specifically, 0–4) for the $target\_offset$ of the ImageNet dataset, as 50,000 images cannot cover all the 999×1000 source-target class pairs. We also observed that some class pairs in ImageNet (e.g., "African crocodile" and "American alligator") are closer than other class pairs (e.g., "African crocodile" and "thunder snake"), and are significantly more easy to perturb into each other. Random numbers were generated as a vector of length 10,000 for CIFAR10 and 50,000 for ImageNet. Hence, the $target\_offset$ was the same for the same image with the same random seed, but $target\_offsets$ was not the same across all images when we used the same random seed.

E. Performance of Targeted Attacks

As we demonstrated in §6.1, on average, Auto-PGD with MD loss finds more adversarial examples than Auto-PGD with all three other loss functions, and CGD attacks performs better than Auto-PGD with MD loss. We observe similar results across datasets, values of ϵ, and defenses, as we show in Figs. 11–13.

F. Statistical Tests On the Performance of Targeted Attacks

As we described in §6.2 and App. B, we used one-sided Wilcoxon signed-rank tests(Wilcoxon, 1945) to compare the performance between Auto-PGD with MD loss and CGD, whose results are shown in Tab. 2. We also compared the performance between CGD and the best performing attack among Auto-PGD using the DLR loss, CW loss, and CE loss, whose results are shown in Tab. 3. We conducted 16 statistical tests in each group, for the 16 different combinations we had. Thus, to account for the multiple tests, we used Bonferroni correction to adjust the confidence level α to $.05/16 = 0.003125$.

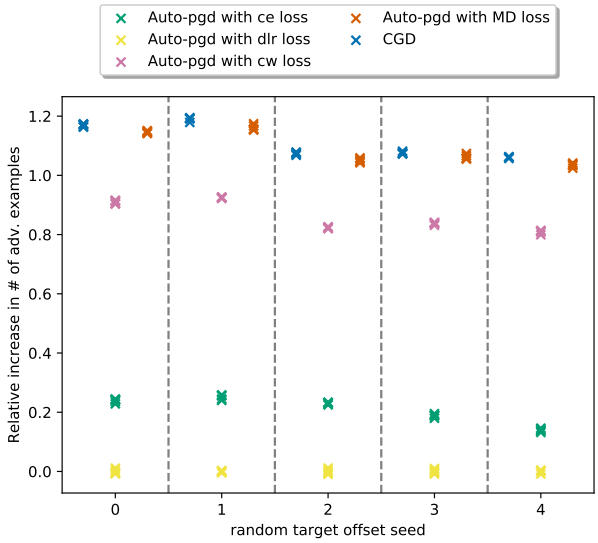

Figure 11: This figure shows the relative improvement in the number of adversarial examples found by attacks on different defenses compared to the worst-performing attack. Experiments were performed using 50,000 images from the testing set of ImageNet. We ran attacks using $\epsilon = 8/255$ against SIE+20 (Salman et al., 2020), five different random initial perturbations with seeds 0–4, and five different random target offsets with seeds 0–4. The result is normalized by the mean of the worst performing method for each target offset.

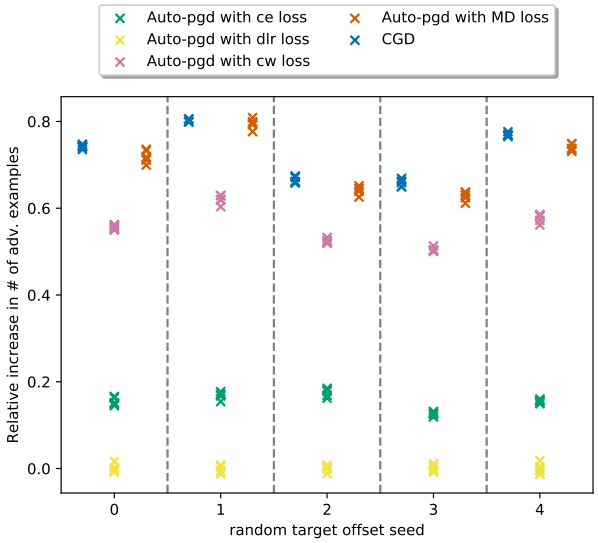

Figure 12: This figure shows the relative improvement in the number of adversarial examples found by attacks on different defenses compared to the worst-performing attack. Experiments were performed using 50,000 images from the testing set of ImageNet. We ran attacks using $\epsilon = 4/255$ against SIE+20 (Salman et al., 2020), five different random initial perturbations with seeds 0–4, and five different random target offsets with seeds 0–4. The result is normalized by the mean of the worst performing method for each target offset.

G. More Evaluation Setup for Untargeted Attacks

We evaluated untargeted attacks on the CIFAR10 dataset. We use the ℓ_∞ distance metric for these attacks as we did for targeted attacks. We use three values of ϵ: $4/255$, $8/255$, and $16/255$. We use the same seven defenses (Carmon et al., 2019; Ding et al., 2020; Hendrycks et al., 2019; Sehwag et al., 2020; Wang et al., 2020; Wong et al., 2020; Wu et al., 2020) as we did in App. D, again trained with $\epsilon = 8/255$ from robust-bench (Croce et al., 2021), as benchmarks to evaluate untargeted attacks. We evaluated untargeted attacks by the number of adversarial examples they found within the ℓ_∞ distance limit. We ran all attacks, again with 100 iterations as in its default configuration. We ran each attack with five different random initial perturbations, using seeds 0–4, and a batch size of 512.

H. Performance of Untargeted Attacks

As we introduced in §7.2, on average, $\text{CGD}_{\text{untarg}}$ performed better than Auto-PGD with L_{DLR} and Auto-PGD with L_{CW^*}. Figs. 14–16 show similar results when we use different values of ϵ.

I. Uniqueness of Attacks

As we described §6.4, among the attack methods we tried, each of them found a slightly different set of successful adversarial examples, and each attack found some adversarial examples which any of the other methods did not. We observe the same phenomena across attack methods, values of ϵ, defenses, and datasets, as shown in Figs. 17–33.

At the same time, when the same attack method is given different random initial perturbations, the attack always found a slightly different set of adversarial examples, within the 20 random initial perturbations we tried. We observe the same phenomena across attack methods, values of ϵ, and defenses on CIFAR10, as shown in Figs. 34–46.

J. Gradient Based Quantization

Bonnet et al. proposed gradient based quantization, an approach to round the current perturbation along the sign of the gradients (Bonnet et al., 2020). We ran experiments with gradient-based quantization, using the formula

$$x_{test} = round(x'_i * 255 + sign(\frac{\partial L(x'_i, t)}{\partial x'_i}) * .499999)/255$$

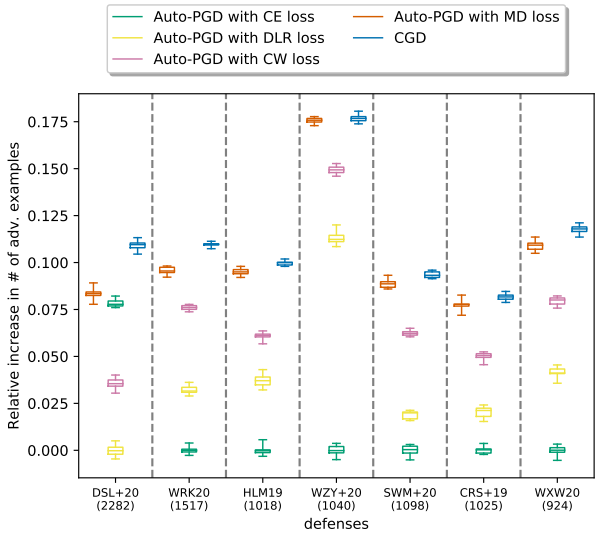

Figure 13: This figure shows the relative improvement in the number of adversarial examples found by attacks on different defenses compared to the worst-performing attack. Experiments were performed using 10,000 images from the testing set of CIFAR10. We ran attacks using $\epsilon = 8/255$, 20 different random initial perturbations with seeds 0–19, and a fixed random target offset with seed 0. The result is normalized by the mean of the worst performing method against each model.

in addition to the quantization described in §2.1, and declared an attack successful if either of the quantizations produced a perturbation that caused the model to predict the target class in any iteration. We ran all attacks at $\epsilon = 16/255$ against defenses on CIFAR10, with the same setup as described in §5. The results are shown in Fig. 47. With an additional 8.9–13.1% time cost (measured using the same approach as in §6.3), the attacks succeed only by an additional 0.00–0.04%; the relative relationship between success rates of different attacks remains the same.

Table 2: This table shows the result of the Wilcoxon signed rank test with an adjusted normal approximation for the null hypotheses that Auto-PGD using the MD loss performed equal or better than CGD in each combination of value of ϵ, dataset, and defense that we tried. The defenses we used include DSL+20 (Ding et al., 2020), WRK20 (Wong et al., 2020), HLM19 (Hendrycks et al., 2019), WZY+20 (Wang et al., 2020), SWM+20 (Sehwag et al., 2020), CRS+19 (Carmon et al., 2019), WXW20 (Wu et al., 2020), and SIE+20 (Salman et al., 2020). We follow the common practice to include the Wilcoxon statistics along with the p-values. Test results where p-values are smaller than α are shown in bold. We reject the null hypothesis in 11 of the 16 tests.

dataset	*defense*	*l_∞ distance limit*			
		$\epsilon = 8/255$		*$\epsilon = 16/255$*	
		W statistic	*p value*	*W statistic*	*p value*
CIFAR10	DSL+20	**2456346.0**	**2e-10**	5284845.5	0.09
	WRK20	**518476.5**	**1e-6**	**2539938.5**	**3e-8**
	HLM19	259521.0	0.02	2377428.0	0.02
	WZY+20	279327.0	0.07	**3297549.0**	**2e-12**
	SWM+20	259521.0	4e-3	**2247924.0**	**2e-5**
	CRS+19	**299366.5**	**3e-3**	**2526231.0**	**1e-5**
	WXW20	**249649.0**	**3e-3**	**1788273.0**	**2e-9**
ImageNet		*$\epsilon = 4/255$*		*$\epsilon = 8/255$*	
		W statistic	*p value*	*W statistic*	*p value*
	SIE+20	**9621163.5**	**7e-12**	**22184577.0**	**2e-13**

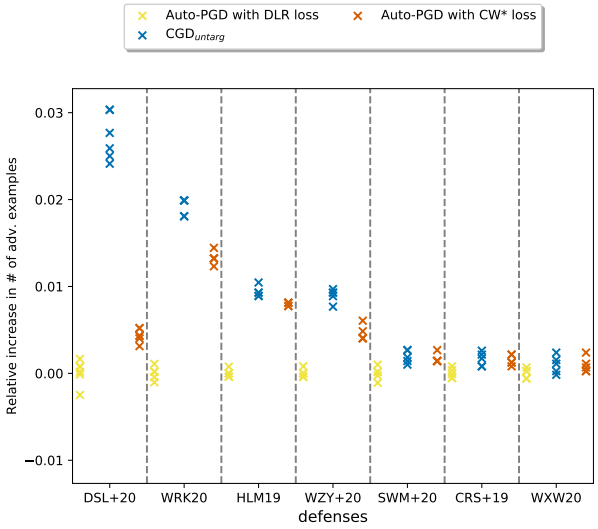

Figure 14: This figure shows the relative improvement in the number of adversarial examples found by untargeted attacks on different defenses compared to the worst-performing attack. Experiments were performed using 10,000 images from the testing set of CIFAR10. We ran attacks using $\epsilon = 4/255$, and five different random initial perturbations with seeds 0–4. The result is normalized by the mean of the worst performing method against each model.

Table 3: This table shows the result of the Wilcoxon signed rank tests with an adjusted normal approximation for the null hypotheses that the best performance among Auto-PGD using the DLR loss, CW loss and CE loss performed equal or better than the performance of CGD on each image across all values of ϵ, datasets, and defenses. The defenses we used include DSL+20 (Ding et al., 2020), WRK20 (Wong et al., 2020), HLM19 (Hendrycks et al., 2019), WZY+20 (Wang et al., 2020), SWM+20 (Sehwag et al., 2020), CRS+19 (Carmon et al., 2019), WXW20 (Wu et al., 2020), and SIE+20 (Salman et al., 2020). We follow the common practice to include the Wilcoxon statistics along with the p-values. Test results where p-values are smaller than α are shown in bold. We reject the null hypothesis in all 16 tests.

dataset	*defense*	*l_∞ distance limit*			
		$\epsilon = 8/255$		*$\epsilon = 16/255$*	
		W statistic	*p value*	*W statistic*	*p value*
CIFAR10	DSL+20	**2311330.5**	**1e-6**	**6709608.5**	**7e-10**
	WRK20	**657798.5**	**3e-8**	**3576641.5**	**7e-26**
	HLM19	**538492.0**	**2e-8**	**3271871.0**	**4e-14**
	WZY+20	**508499.5**	**2e-5**	**4021878.0**	**9e-25**
	SWM+20	**508657.5**	**1e-8**	**2540675.0**	**2e-9**
	CRS+19	**508587.0**	**1e-6**	**2731369.5**	**9e-7**
	WXW20	**498731.5**	**1e-8**	**2004237.0**	**1e-13**
		$\epsilon = 4/255$		*$\epsilon = 8/255$*	
ImageNet		*W statistic*	*p value*	*W statistic*	*p value*
	SIE+20	**19071030.0**	**3e-55**	**54798304.5**	**1e-168**

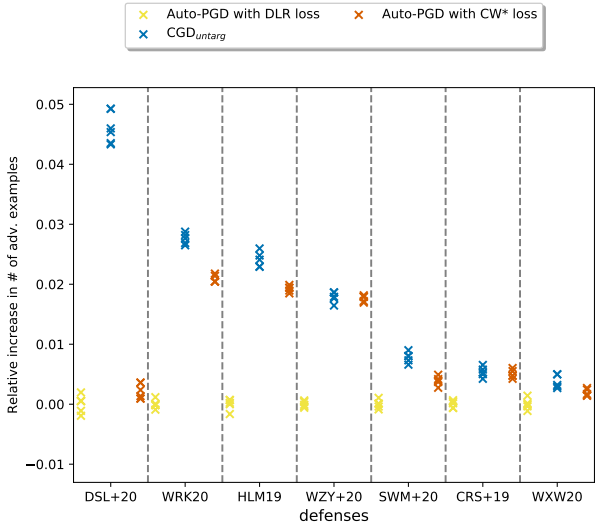

Figure 15: This figure shows the relative improvement in the number of adversarial examples found by untargeted attacks on different defenses compared to the worst-performing attack. Experiments were performed using 10,000 images from the testing set of CIFAR10. We ran attacks using $\epsilon = 8/255$, and five different random initial perturbations with seeds 0–4. The result is normalized by the mean of the worst performing method against each model.

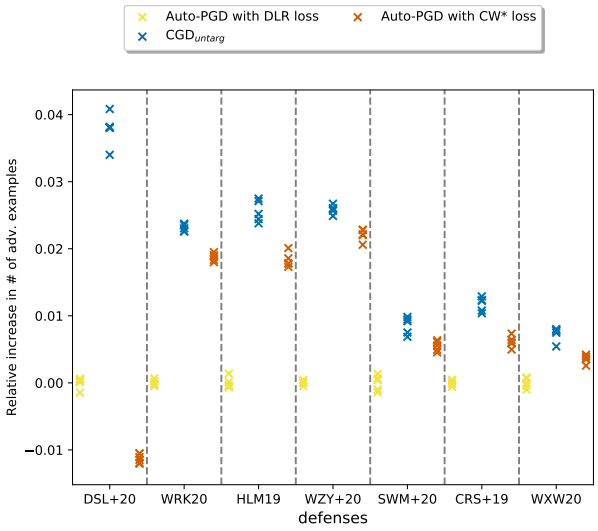

Figure 16: This figure shows the relative improvement in the number of adversarial examples found by untargeted attacks on different defenses compared to the worst-performing attack. Experiments were performed using 10,000 images from the testing set of CIFAR10. We ran attacks using $\epsilon = 16/255$, and five different random initial perturbations with seeds 0–4. The result is normalized by the mean of the worst performing method against each model.

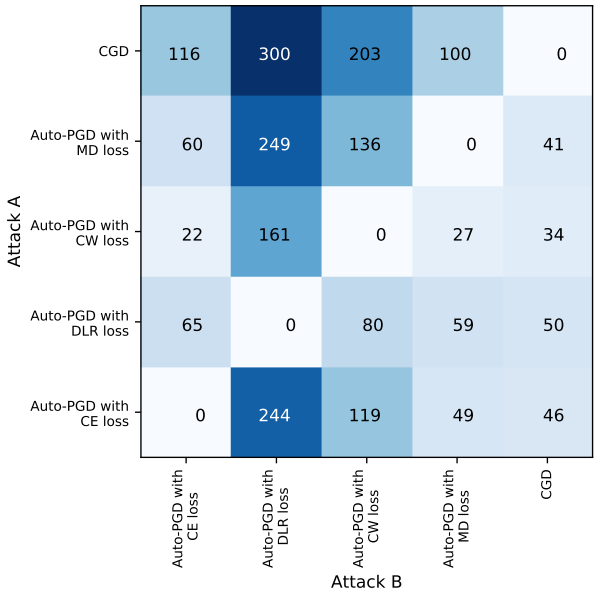

Figure 17: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$ against the DSL+20 (Ding et al., 2020) defense, rounded to whole numbers.

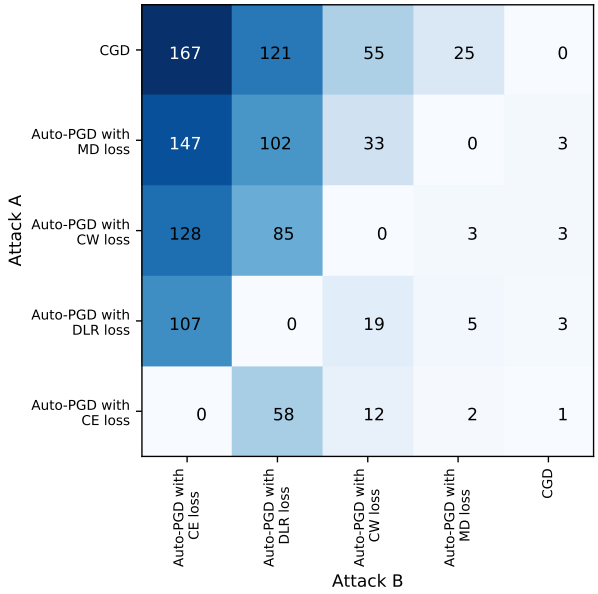

Figure 18: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$ against the WRK20 (Wong et al., 2020) defense, rounded to whole numbers.

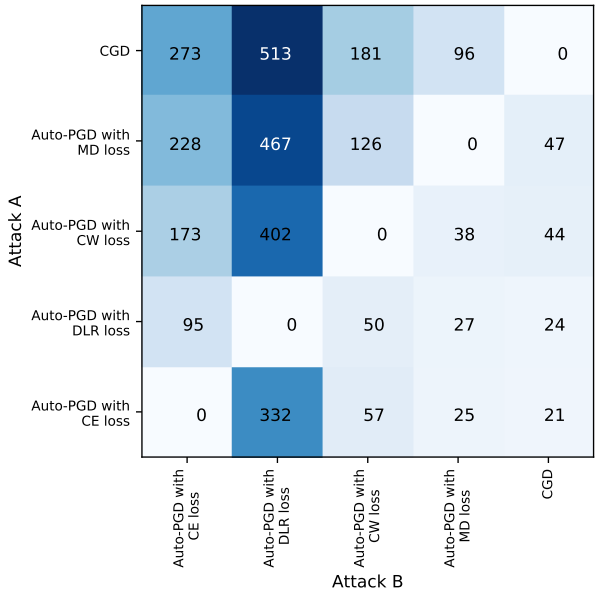

Figure 19: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$ against the WRK20 (Wong et al., 2020) defense, rounded to whole numbers

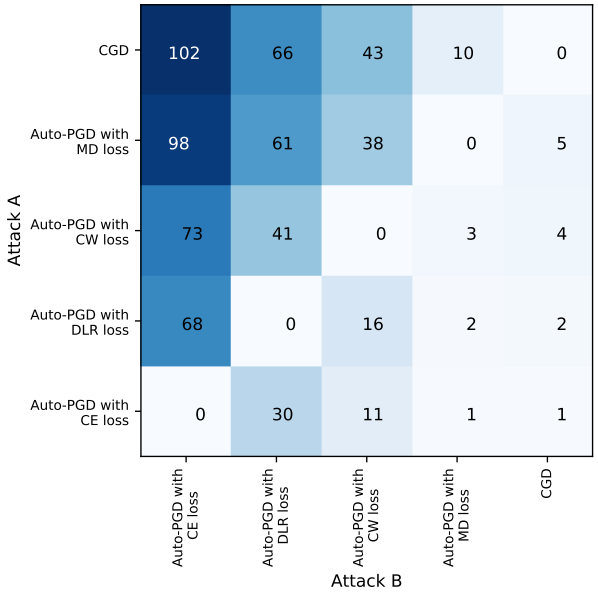

Figure 20: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$ against the HLM19 (Hendrycks et al., 2019) defense, rounded to whole numbers.

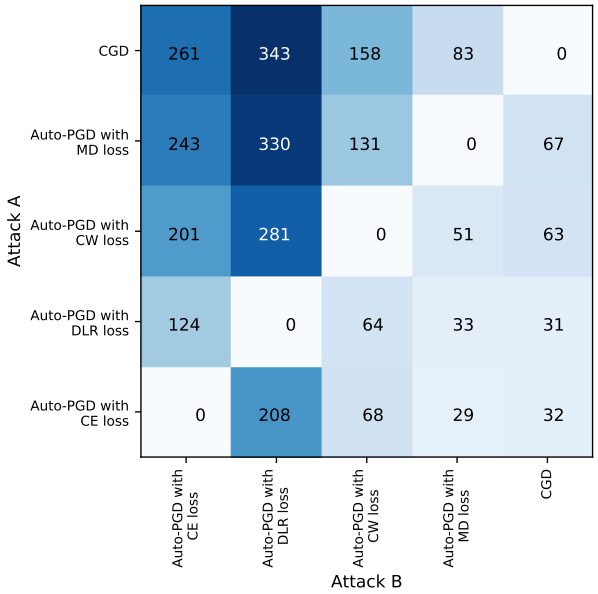

Figure 21: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$ against the HLM19 (Hendrycks et al., 2019) defense, rounded to whole numbers.

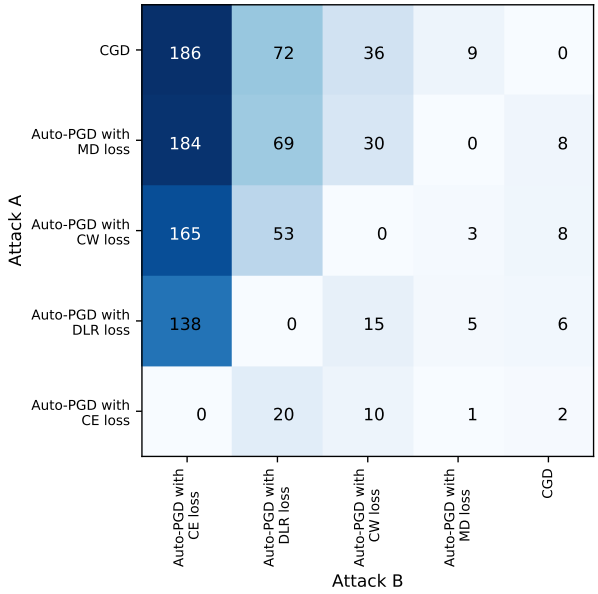

Figure 22: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$ against the WZY+20 (Wang et al., 2020) defense, rounded to whole numbers.

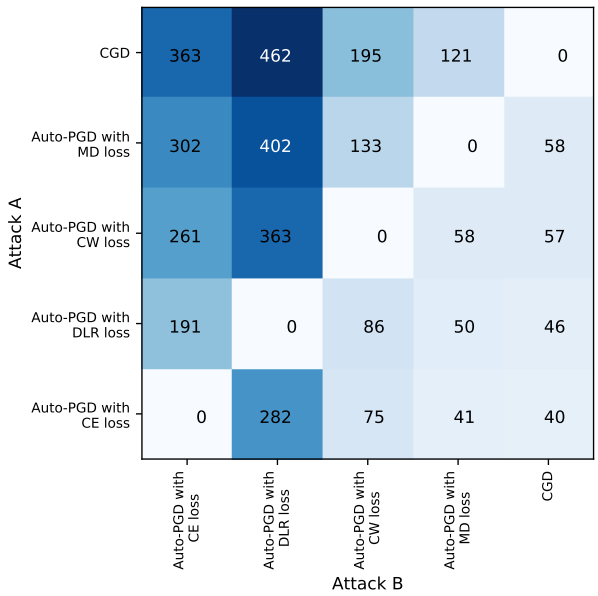

Figure 23: This image shows the average number of successful adversarial examples found by attack A but not by attack B each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$ against the WZY+20 (Wang et al., 2020) defense, rounded to whole numbers.

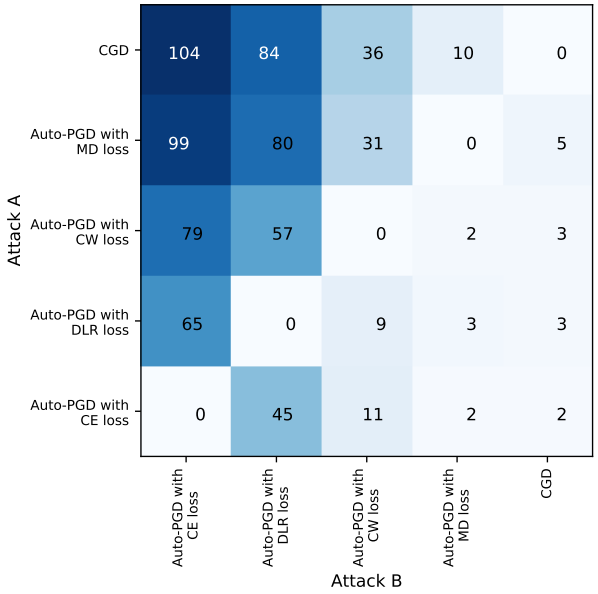

Figure 24: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$ against the SWM+20 (Sehwag et al., 2020) defense, rounded to whole numbers.

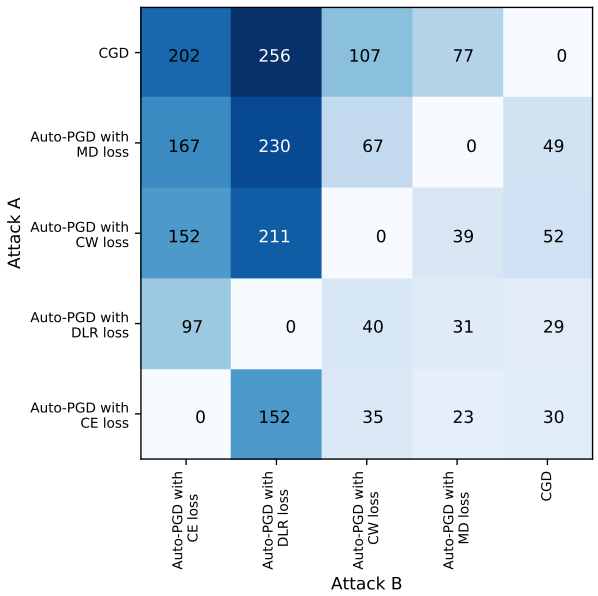

Figure 25: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$ against the SWM+20 (Sehwag et al., 2020) defense, rounded to whole numbers.

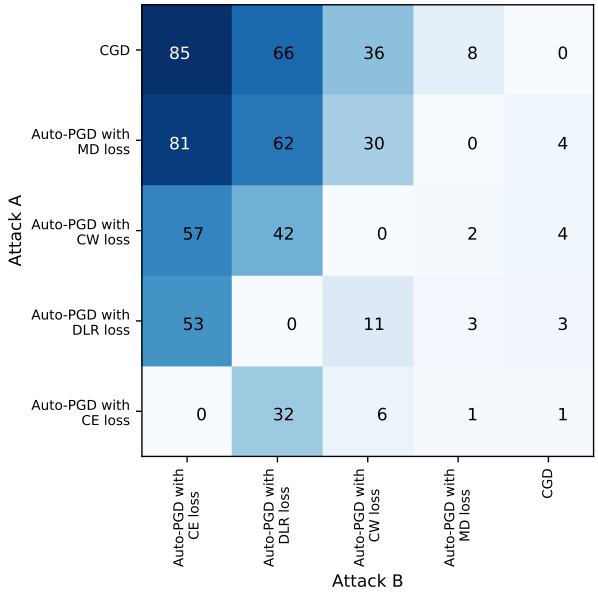

Figure 26: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$ against the CRS+19 (Carmon et al., 2019) defense, rounded to whole numbers.

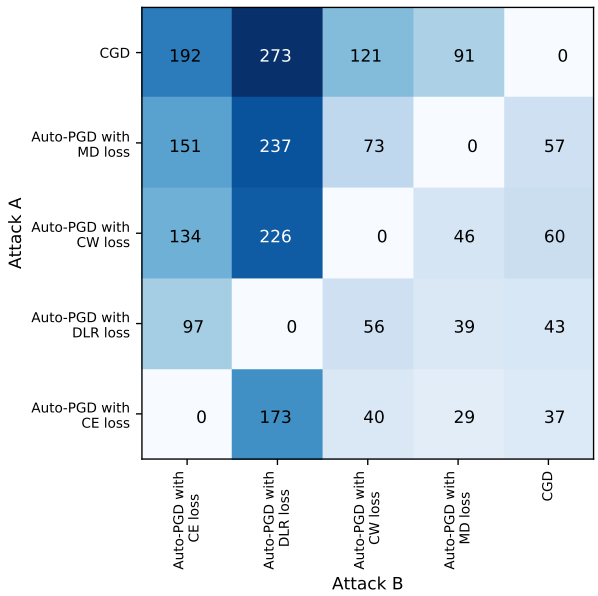

Figure 27: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$ against the CRS+19 (Carmon et al., 2019) defense, rounded to whole numbers.

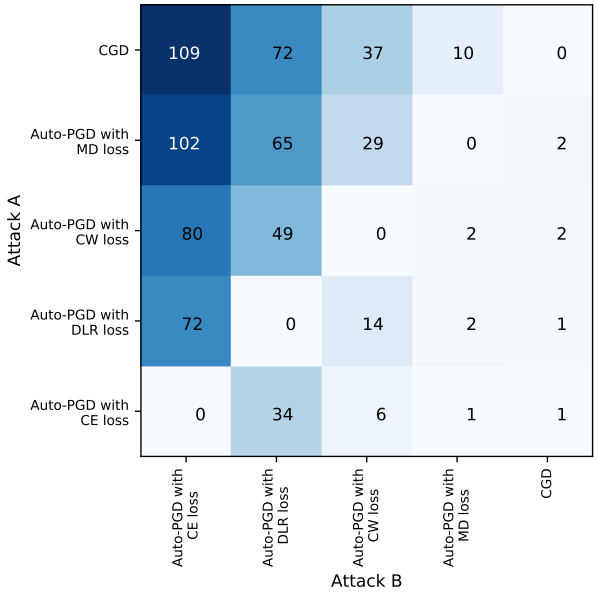

Figure 28: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$ against the WXW20 (Wu et al., 2020) defense, rounded to whole numbers.

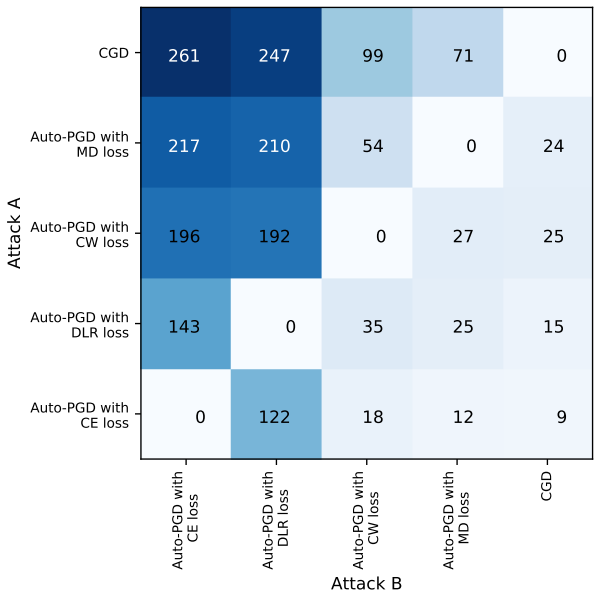

Figure 29: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 16/255$ against the WXW20 (Wu et al., 2020) defense, rounded to whole numbers.

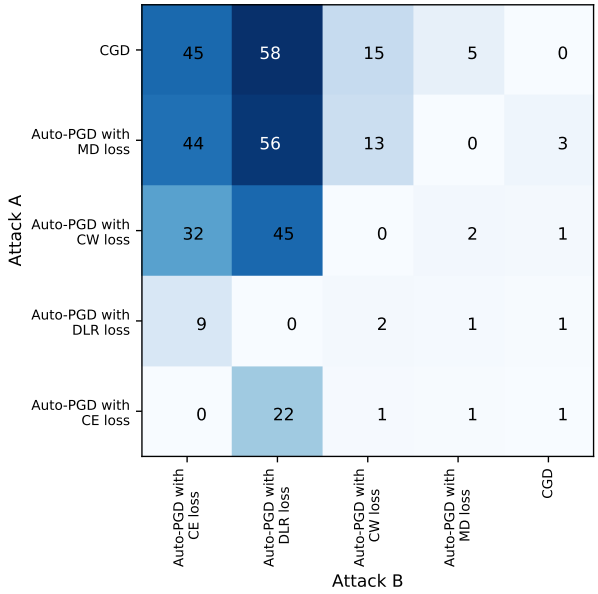

Figure 30: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 50,000 attempts on the testing set of ImageNet using $\epsilon = 4/255$ against the SIE+20 (Salman et al., 2020) defense, rounded to whole numbers.

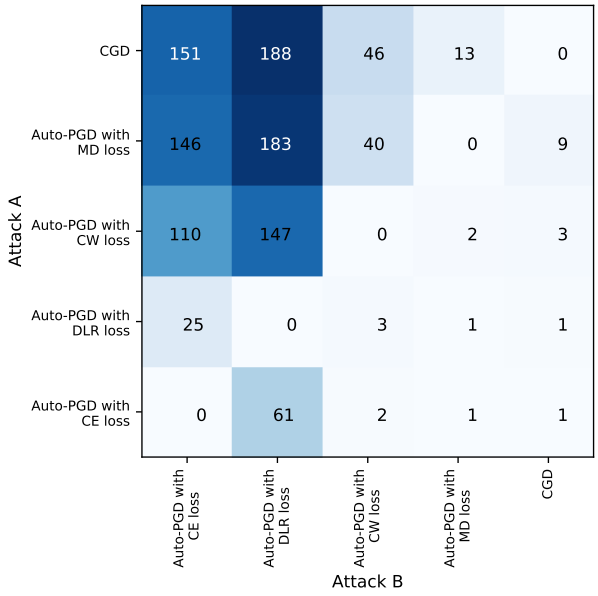

Figure 31: This image shows the average number of successful adversarial examples found by attack A but not by attack B in each set of 50,000 attempts on the testing set of ImageNet using $\epsilon = 8/255$ against the SIE+20 (Salman et al., 2020) defense, rounded to whole numbers.

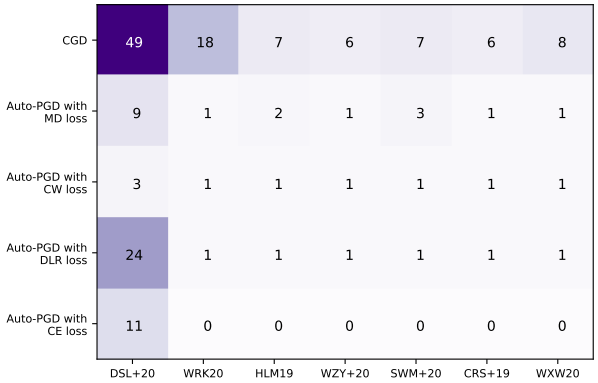

Figure 32: This is the average number of successful adversarial examples found by an attack but not by any of other attack methods, against each defense in each set of 10,000 attempts on the testing set of CIFAR10 using $\epsilon = 8/255$, rounded to whole numbers.

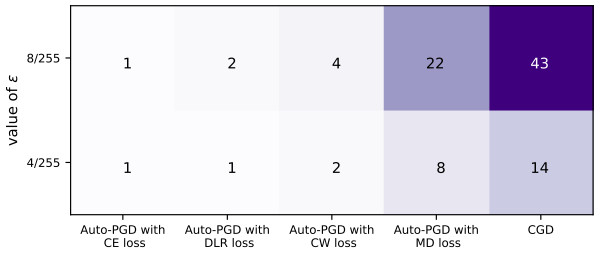

Figure 33: This is the average number of successful adversarial examples found by an attack but not by any other attack methods, against the SIE+20 (Salman et al., 2020) defense in each set of 50,000 attempts on the testing set of ImageNet, no decimals are kept.

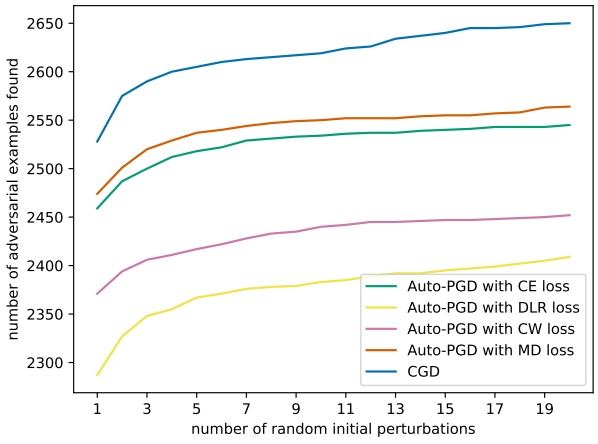

Figure 34: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 8/255$, against the DSL+20 (Ding et al., 2020) defense.

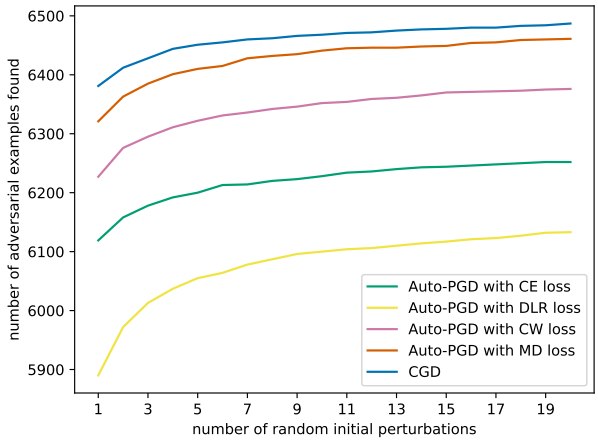

Figure 35: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 16/255$, against the WRK20 (Wong et al., 2020) defense.

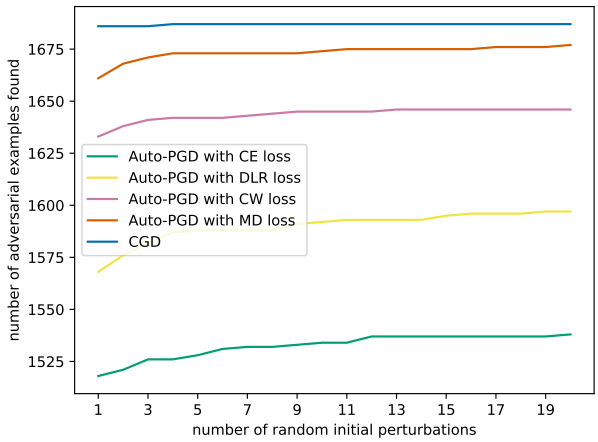

Figure 36: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 8/255$, against the WRK20 (Wong et al., 2020) defense.

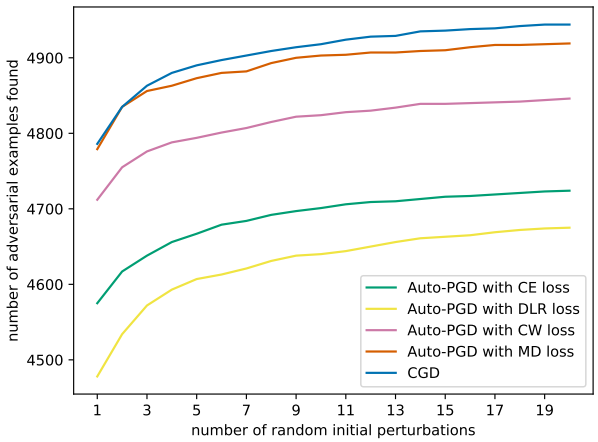

Figure 37: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 16/255$, against the HLM19 (Hendrycks et al., 2019) defense.

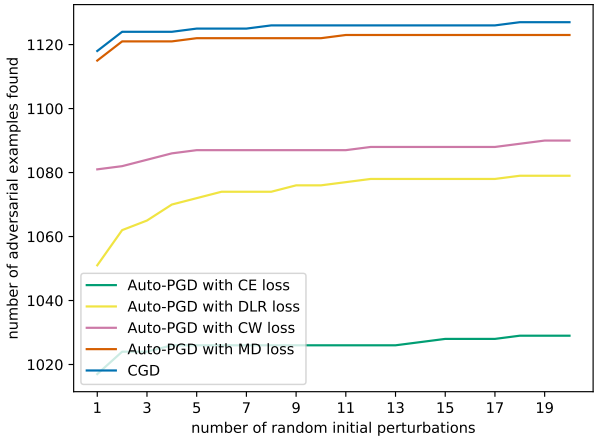

Figure 38: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 8/255$, against the HLM19 (Hendrycks et al., 2019) defense.

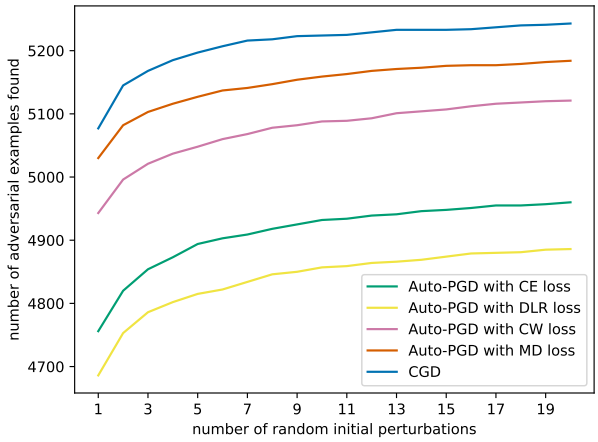

Figure 39: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 16/255$, against the WZY+20 (Wang et al., 2020) defense.

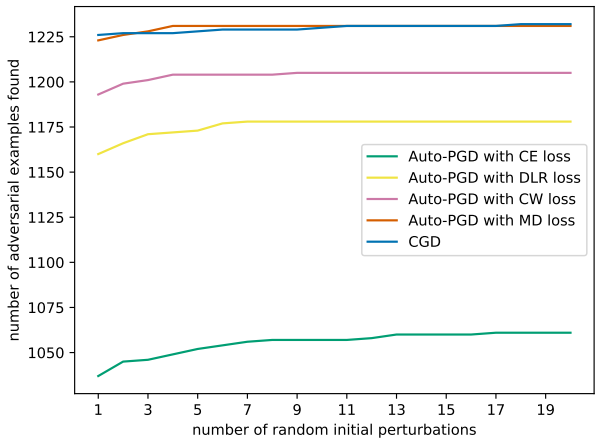

Figure 40: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 8/255$, against the WZY+20 (Wang et al., 2020) defense.

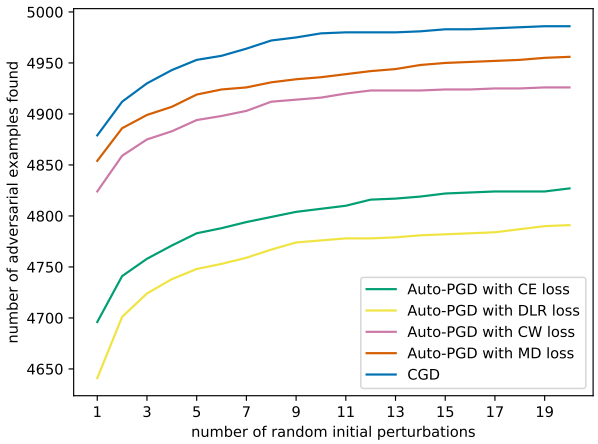

Figure 41: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 16/255$, against the SWM+20 (Sehwag et al., 2020) defense.

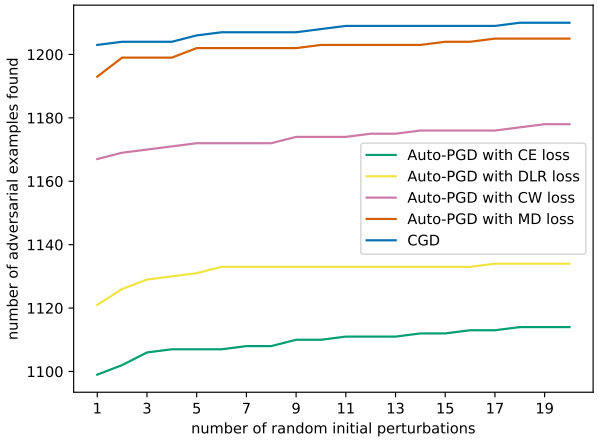

Figure 42: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 8/255$, against the SWM+20 (Sehwag et al., 2020) defense.

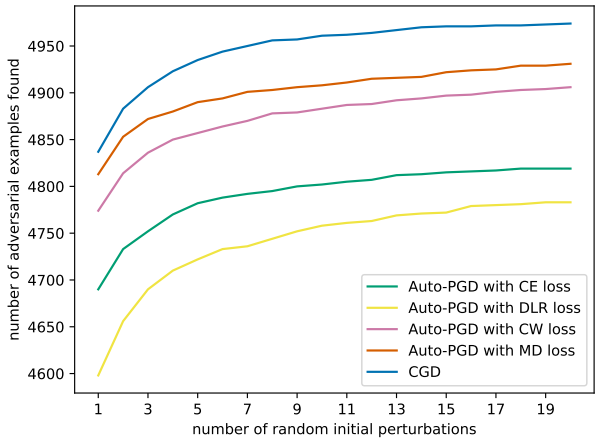

Figure 43: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 16/255$, against the CRS+19 (Carmon et al., 2019) defense.

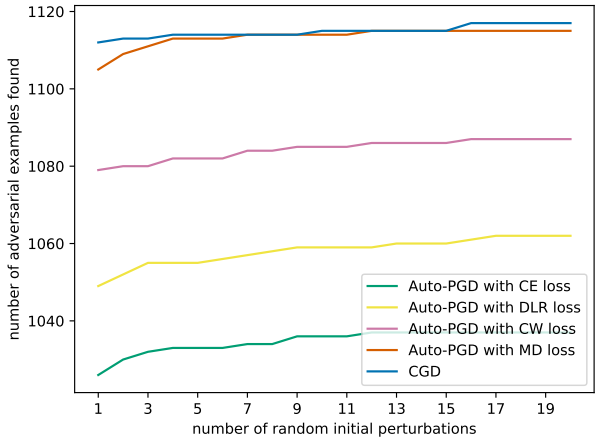

Figure 44: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 8/255$, against the CRS+19 (Carmon et al., 2019) defense.

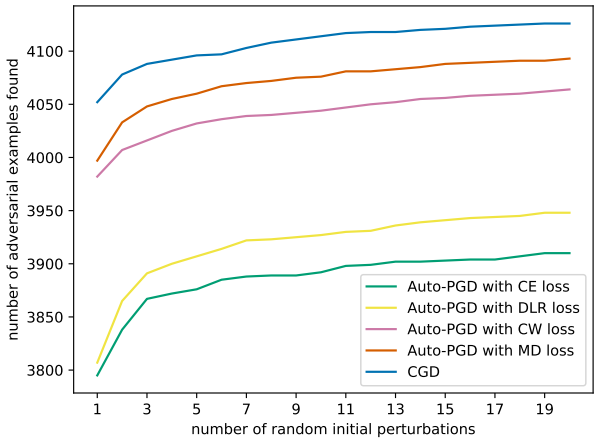

Figure 45: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 16/255$, against the WXW20 (Wu et al., 2020) defense.

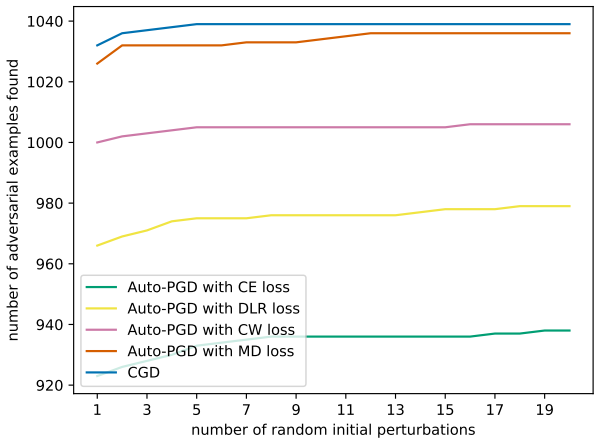

Figure 46: The image shows the number of adversarial examples each of the attacks found when they are allowed to use the specified number of random initial perturbations, on all 10,000 images from the testing set of CIFAR10, with $\epsilon = 8/255$, against the WXW20 (Wu et al., 2020) defense.

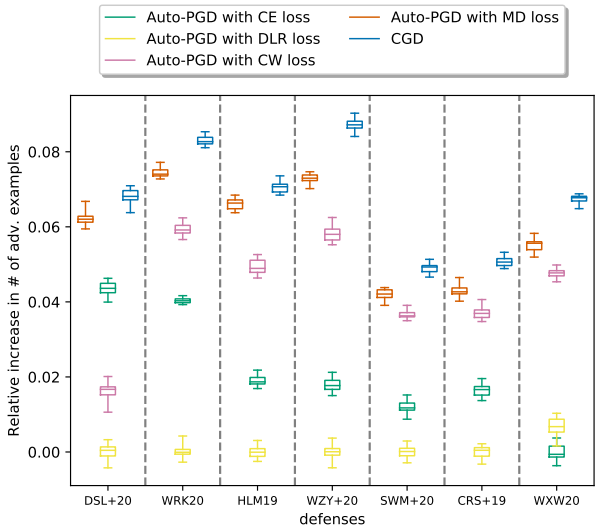

Figure 47: This figure shows the relative improvement in the number of adversarial examples found by attacks on different defenses compared to the worst-performing attack, when all attacks are allowed to perform one more forward propagation to verify if a gradient based quantized version of the current perturbation would lead to success. Experiments were performed using 10,000 images from the testing set of CIFAR10. We ran attacks using $\epsilon = 16/255$, 20 different random initial perturbations with seeds 0–19, and a fixed random target offset with seed 0. The result is normalized by the mean of the worst performing method against each model.