Decoupling Direction and Norm for Efficient Gradient-Based L_2 Adversarial Attacks and Defenses

Jérôme Rony* [1] Luiz G. Hafemann* [1] Luiz S. Oliveira [2] Ismail Ben Ayed [1]
Robert Sabourin [1] Eric Granger [1]

[1] Laboratoire d'imagerie, de vision et d'intelligence artificielle (LIVIA), ÉTS Montreal, Canada
[2] Department of Informatics, Federal University of Paraná, Curitiba, Brazil

jerome.rony@gmail.com luiz.gh@mailbox.org lesoliveira@inf.ufpr.br
{ismail.benayed, robert.sabourin, eric.granger}@etsmtl.ca

Abstract

Research on adversarial examples in computer vision tasks has shown that small, often imperceptible changes to an image can induce misclassification, which has security implications for a wide range of image processing systems. Considering L_2 norm distortions, the Carlini and Wagner attack is presently the most effective white-box attack in the literature. However, this method is slow since it performs a line-search for one of the optimization terms, and often requires thousands of iterations. In this paper, an efficient approach is proposed to generate gradient-based attacks that induce misclassifications with low L_2 norm, by decoupling the direction and the norm of the adversarial perturbation that is added to the image. Experiments conducted on the MNIST, CIFAR-10 and ImageNet datasets indicate that our attack achieves comparable results to the state-of-the-art (in terms of L_2 norm) with considerably fewer iterations (as few as 100 iterations), which opens the possibility of using these attacks for adversarial training. Models trained with our attack achieve state-of-the-art robustness against white-box gradient-based L_2 attacks on the MNIST and CIFAR-10 datasets, outperforming the Madry defense when the attacks are limited to a maximum norm.

1. Introduction

Deep neural networks have achieved state-of-the-art performances on a wide variety of computer vision applications, such as image classification, object detection, tracking, and activity recognition [8]. In spite of their success in addressing these challenging tasks, they are vulnerable to active *adversaries*. Most notably, they are susceptible to *adversarial examples*[1], in which adding small perturbations to an image, often imperceptible to a human observer, causes a misclassification [2, 18].

Recent research on adversarial examples developed *attacks* that allow for evaluating the robustness of models, as well as *defenses* against these attacks. Attacks have been proposed to achieve different objectives, such as minimizing the amount of noise that induces misclassification [5, 18], or being fast enough to be incorporated into the training procedure [7, 19]. In particular, considering the case of obtaining adversarial examples with lowest perturbation (measured by its L_2 norm), the state-of-the-art attack has been proposed by Carlini and Wagner (C&W) [5]. While this attack generates adversarial examples with low L_2 noise, it also requires a high number of iterations, which makes it impractical for training a robust model to defend against such attacks. In contrast, one-step attacks are fast to generate, but using them for training does not increase model robustness on white-box scenarios, with full knowledge of the model under attack [19]. Developing an attack that finds adversarial examples with low noise in few iterations would enable adversarial training with such examples, which could potentially increase model robustness against white-box attacks.

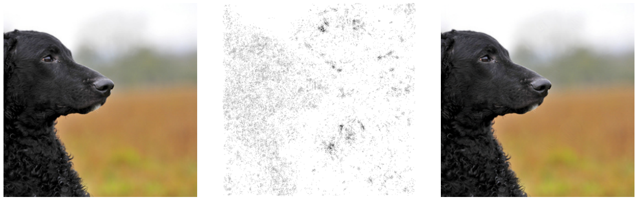
Developing attacks that minimize the norm of the adversarial perturbations requires optimizing two objectives: 1) obtaining a low L_2 norm, while 2) inducing a misclassification. With the current state-of-the-art method (C&W [5]), this is addressed by using a two-term loss function, with the weight balancing the two competing objectives found via an expensive line search, requiring a large number of iterations. This makes the evaluation of a system's robustness very slow and it is unpractical for adversarial training.

In this paper, we propose an efficient gradient-based attack called *Decoupled Direction and Norm*[2] (DDN) that induces misclassification with a low L_2 norm. This attack optimizes the cross-entropy loss, and instead of penalizing

*Equal contribution.

[1] This also affects other machine learning classifiers, but we restrict our analysis to CNNs, that are most commonly used in computer vision tasks.

[2] Code available at https://github.com/jeromerony/fast_adversarial.



x δ $\tilde{x} = x + \delta$

Figure 1: Example of an adversarial image on the ImageNet dataset. The sample x is recognized as a Curly-coated retriever. Adding a perturbation δ we obtain an adversarial image that is classified as a microwave (with $\|\delta\|_2 = 0.7$).

the norm in each iteration, projects the perturbation onto a L_2-sphere centered at the original image. The change in norm is then based on whether the sample is adversarial or not. Using this approach to decouple the direction and norm of the adversarial noise leads to an attack that needs significantly fewer iterations, achieving a level of performance comparable to state-of-the-art, while being amenable to be used for adversarial training.

A comprehensive set of experiments was conducted using the MNIST, CIFAR-10 and ImageNet datasets. Our attack obtains comparable results to the state-of-the-art while requiring much fewer iterations (~100 times less than C&W). For untargeted attacks on the ImageNet dataset, our attack achieves better performance than the C&W attack, taking less than 10 minutes to attack 1 000 images, versus over 35 hours to run the C&W attack.

Results for adversarial training on the MNIST and CIFAR-10 datasets indicate that DDN can achieve state-of-the-art robustness compared to the Madry defense [13]. These models require that attacks use a higher average L_2 norm to induce misclassifications. They also obtain a higher accuracy when the L_2 norm of the attacks is bounded. On MNIST, if the attack norm is restricted to 1.5, the model trained with the Madry defense achieves 67.3% accuracy, while our model achieves 87.2% accuracy. On CIFAR-10, for attacks restricted to a norm of 0.5, the Madry model achieves 56.1% accuracy, compared to 67.6% in our model.

2. Related Work

In this section, we formalize the problem of adversarial examples, the threat model, and review the main attack and defense methods proposed in the literature.

2.1. Problem Formulation

Let x be an sample from the input space $\mathcal{X}$, with label y_{true} from a set of possible labels $\mathcal{Y}$. Let $D(x_1, x_2)$ be a distance measure that compares two input samples (ideally capturing their perceptual similarity). $P(\mathrm{y}|x, \theta)$ is a model (classifier) parameterized by θ. An example $\tilde{x} \in \mathcal{X}$ is called *adversarial* (for non-targeted attacks) against the classifier if $\arg\max_j P(y_j|\tilde{x}, \theta) \neq y_{\text{true}}$ and $D(x, \tilde{x}) \leq \epsilon$, for a given maximum perturbation ϵ. A *targeted attack* with a given desired class y_{target} further requires that $\arg\max_j P(y_j|\tilde{x}, \theta) = y_{\text{target}}$. We denote as $J(x, y, \theta)$, the cross-entropy between the prediction of the model for an input x and a label y. Fig. 1 illustrates a targeted attack on the ImageNet dataset, against an Inception v3 model [17].

In this paper, attacks are considered to be generated by a gradient-based optimization procedure, restricting our analysis to differentiable classifiers. These attacks can be formulated either to obtain a minimum distortion $D(x, \tilde{x})$, or to obtain the worst possible loss in a region $D(x, \tilde{x}) \leq \epsilon$. As an example, consider that the distance function is a norm (e.g., L_0, L_2 or L_∞), and the inputs are images (where each pixel's value is constrained between 0 and M). In a white-box scenario, the optimization procedure to obtain an non-targeted attack with minimum distortion δ can be formulated as:

$$\begin{aligned} \min_{\delta} \|\delta\| \quad \text{subject to} \quad & \arg\max_j P(y_j|x + \delta, \theta) \neq y_{\text{true}} \\ \text{and} \quad & 0 \leq x + \delta \leq M \end{aligned} \tag{1}$$

With a similar formulation for *targeted attacks*, by changing the constraint to be equal to the target class.

If the objective is to obtain the worst possible loss for a given maximum noise of norm ϵ, the problem can be formulated as:

$$\begin{aligned} \min_{\delta} P(y_{\text{true}}|x + \delta, \theta) \quad \text{subject to} \quad & \|\delta\| \leq \epsilon \\ \text{and} \quad & 0 \leq x + \delta \leq M \end{aligned} \tag{2}$$

With a similar formulation for *targeted attacks*, by maximizing $P(y_{\text{target}}|x + \delta, \theta)$.

We focus on gradient-based attacks that optimize the L_2 norm of the distortion. While this distance does not perfectly capture perceptual similarity, it is widely used in computer vision to measure similarity between images (e.g. comparing image compression algorithms, where Peak Signal-to-Noise Ratio is used, which is directly related to the L_2 measure). A differentiable distance measure that captures perceptual similarity is still an open research problem.

2.2. Threat Model

In this paper, a *white-box* scenario is considered, also known as a Perfect Knowledge scenario [2]. In this scenario, we consider that an attacker has perfect knowledge of the system, including the neural network architecture and the learned weights θ. This threat model serves to evaluate system security under the *worst case* scenario. Other scenarios can be conceived to evaluate attacks under different assumptions on the attacker's knowledge, for instance, no access to the trained model, no access to the same training set, among others. These scenarios are referred as *black-box* or Limited-Knowledge [2].

2.3. Attacks

Several attacks were proposed in the literature, either focusing on obtaining adversarial examples with a small δ (Eq. 1) [5, 14, 18], or on obtaining adversarial examples in one (or few) steps for adversarial training [7, 12].

L-BFGS. Szegedy *et al.* [18] proposed an attack for minimally distorted examples (Eq. 1), by considering the following approximation:

$$\begin{aligned} &\min_{\delta} C \left\|\delta\right\|_2 + \log P(y_{\text{true}}|x+\delta,\theta) \\ &\quad \text{subject to} \ \ 0 \leq x+\delta \leq M \end{aligned} \tag{3}$$

where the constraint $x+\delta \in [0, M]^n$ was addressed by using a box-constrained optimizer (L-BFGS: Limited memory Broyden–Fletcher–Goldfarb–Shanno), and a line-search to find an appropriate value of C.

FGSM. Goodfellow *et al.* [7] proposed the Fast Gradient Sign Method, a one-step method that could generate adversarial examples. The original formulation was developed considering the L_∞ norm, but it has also been used to generate attacks that focus on the L_2 norm as follows:

$$\tilde{x} = x + \epsilon \frac{\nabla_x J(x,y,\theta)}{\|\nabla_x J(x,y,\theta)\|} \tag{4}$$

where the constraint $\tilde{x} \in [0, M]^n$ was addressed by simply clipping the resulting adversarial example.

DeepFool. This method considers a linear approximation of the model, and iteratively refines an adversary example by choosing the point that would cross the decision boundary under this approximation. This method was developed for untargeted attacks, and for any L_p norm [14].

C&W. Similarly to the L-BFGS method, the C&W L_2 attack [5] minimizes two criteria at the same time – the perturbation that makes the sample adversarial (e.g., misclassified by the model), and the L_2 norm of the perturbation. Instead of using a box-constrained optimization method, they propose changing variables using the $\tanh$ function, and instead of optimizing the cross-entropy of the adversarial example, they use a difference between logits. For a targeted attack aiming to obtain class t, with Z denoting the model output before the softmax activation (logits), it optimizes:

$$\begin{aligned} &\min_{\delta} \left[\|\tilde{x}-x\|_2^2 + Cf(\tilde{x})\right] \\ &\text{where} \quad f(\tilde{x}) = \max(\max_{i\neq t}\{Z(\tilde{x})_i\} - Z(\tilde{x})_t, -\kappa) \\ &\text{and} \quad \tilde{x} = \frac{1}{2}(\tanh(\operatorname{arctanh}(x)+\delta)+1) \end{aligned} \tag{5}$$

where $Z(\tilde{x})_i$ denotes the logit corresponding to the i-th class. By increasing the confidence parameter κ, the adversarial sample will be misclassified with higher confidence. To use this attack in the untargeted setting, the definition of f is modified to $f(\tilde{x}) = \max(Z(\tilde{x})_y - \max_{i\neq y}\{Z(\tilde{x})_i\}, -\kappa)$ where y is the original label.

2.4. Defenses

Developing defenses against adversarial examples is an active area of research. To some extent, there is an *arms race* on developing defenses and attacks that break them. Goodfellow *et al.* proposed a method called *adversarial training* [7], in which the training data is augmented with FGSM samples. This was later shown not to be robust against iterative white-box attacks, nor black-box single-step attacks [19]. Papernot *et al.* [15] proposed a *distillation* procedure to train robust networks, which was shown to be easily broken by iterative white-box attacks [5]. Other defenses involve *obfuscated gradients* [1], where models either incorporate non-differentiable steps (such that the gradient cannot be computed) [4, 9], or randomized elements (to induce incorrect estimations of the gradient) [6, 20]. These defenses were later shown to be ineffective when attacked with Backward Pass Differentiable Approximation (BPDA) [1], where the actual model is used for forward propagation, and the gradient in the backward-pass is approximated. The Madry defense [13], which considers a worst-case optimization, is the only defense that has been shown to be somewhat robust (on the MNIST and CIFAR-10 datasets). Below we provide more detail on the general approach of adversarial training, and the Madry defense.

Adversarial Training. This defense considers augmenting the training objective with adversarial examples [7], with the intention of improving robustness. Given a model with loss function $J(x,y,\theta)$, training is augmented as follows:

$$\tilde{J}(x,y,\theta) = \alpha J(x,y,\theta) + (1-\alpha)J(\tilde{x},y,\theta) \tag{6}$$

where $\tilde{x}$ is an adversarial sample. In [7], the FGSM is used to generate the adversarial example in a single step. Tramèr *et al.* [19] extended this method, showing that generating one-step attacks using the model under training introduced an issue. The model can converge to a degenerate solution where its gradients produce "easy" adversarial samples, causing the adversarial loss to have a limited influence on the training objective. They proposed a method in which an ensemble of models is also used to generate the adversarial examples $\tilde{x}$. This method displays some robustness against black-box attacks using surrogate models, but does not increase robustness in white-box scenarios.

Madry Defense. Madry *et al.* [13] proposed a saddle point optimization problem, optimizing for the worst case:

$$\begin{aligned} &\min_{\theta} p(\theta) \\ &\text{where} \quad p(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} J(x+\delta,y,\theta)\right] \end{aligned} \tag{7}$$

where $\mathcal{D}$ is the training set, and $\mathcal{S}$ indicates the feasible region for the attacker (e.g. $\mathcal{S} = \{\delta : \|\delta\| < \epsilon\}$). They show that Eq. 7 can be optimized by stochastic gradient descent – during each training iteration, it first finds the adversarial

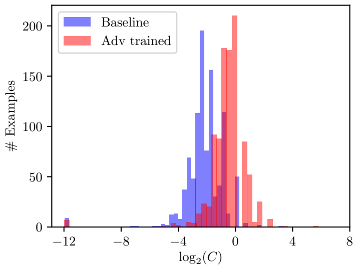


Figure 2: Histogram of the best C found by the C&W algorithm with 9 search steps on the MNIST dataset.

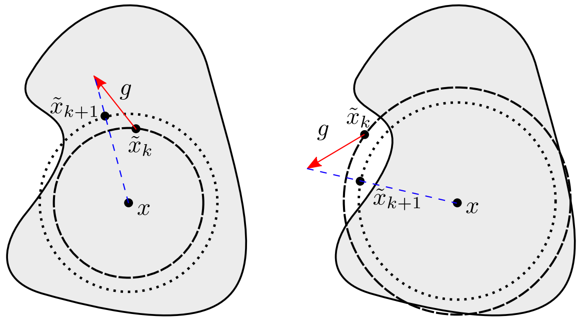
example that maximizes the loss around the current training sample x (i.e., maximizing the loss over δ, which is equivalent to minimizing the probability of the correct class as in Eq. 2), and then, it minimizes the loss over θ. Experiments in Athalye *et al.* [1] show that it was the only defense not broken under white-box attacks.

3. Decoupled Direction and Norm Attack

From the problem definition, we see that finding the worst adversary in a fixed region is an easier task. In Eq. 2, both constraints can be expressed in terms of δ, and the resulting equation can be optimized using projected gradient descent. Finding the closest adversarial example is harder: Eq. 1 has a constraint on the prediction of the model, which cannot be addressed by a simple projection. A common approach, which is used by Szegedy *et al.* [18] and in the C&W [5] attack, is to approximate the constrained problem in Eq. 1 by an unconstrained one, replacing the constraint with a *penalty*. This amounts to jointly optimizing both terms, the norm of δ and a classification term (see Eq. 3 and 5), with a sufficiently high parameter C. In the general context of constrained optimization, such a penalty-based approach is a well known general principle [11]. While tackling an unconstrained problem is convenient, penalty methods have well-known difficulties in practice. The main difficulty is that one has to choose parameter C in an *ad hoc* way. For instance, if C is too small in Eq. 5, the example will not be adversarial; if it is too large, this term will dominate, and result in an adversarial example with more noise. This can be particularly problematic when optimizing with a low number of steps (e.g. to enable its use in adversarial training). Fig. 2 plots a histogram of the values of C that were obtained by running the C&W attack on the MNIST dataset. We can see that the optimum C varies significantly among different examples, ranging from 2^{-11} to 2^5. We also see that the distribution of the best constant C changes whether we attack a model with or without adversarial training (adversarially trained models often require higher C). Furthermore, penalty methods typically result in slow convergence [11].

Algorithm 1 Decoupled Direction and Norm Attack

```
Input: $x$: original image to be attacked
Input: $y$: true label (untargeted) or target label (targeted)
Input: $K$: number of iterations
Input: $\alpha$: step size
Input: $\gamma$: factor to modify the norm in each iteration
Output: $\tilde{x}$: adversarial image
Initialize $\delta_0 \leftarrow \mathbf{0}$, $\tilde{x}_0 \leftarrow x$, $\epsilon_0 \leftarrow 1$
If targeted attack: $m \leftarrow -1$ else $m \leftarrow +1$
for $k \leftarrow 1$ to $K$ do
    $g \leftarrow m\nabla_{\tilde{x}_{k-1}} J(\tilde{x}_{k-1}, y, \theta)$
    $g \leftarrow \alpha\frac{g}{\|g\|_2}$        ▷ Step of size $\alpha$ in the direction of $g$
    $\delta_k \leftarrow \delta_{k-1} + g$
    if $\tilde{x}_{k-1}$ is adversarial then
        $\epsilon_k \leftarrow (1-\gamma)\epsilon_{k-1}$        ▷ Decrease norm
    else
        $\epsilon_k \leftarrow (1+\gamma)\epsilon_{k-1}$        ▷ Increase norm
    end if
    $\tilde{x}_k \leftarrow x + \epsilon_k\frac{\delta_k}{\|\delta_k\|_2}$        ▷ Project $\delta_k$ onto an $\epsilon_k$-sphere around $x$
    $\tilde{x}_k \leftarrow \text{clip}(\tilde{x}_k, 0, 1)$        ▷ Ensure $\tilde{x}_k \in \mathcal{X}$
end for
Return $\tilde{x}_k$ that has lowest norm $\|\tilde{x}_k - x\|_2$ and is adversarial
```



(a) $\tilde{x}_k$ not adversarial

(b) $\tilde{x}_k$ adversarial

Figure 3: Illustration of an untargeted attack. The shaded area denotes the region of the input space classified as y_{true}. In (a), $\tilde{x}_k$ is still not adversarial, and we increase the norm ϵ_{k+1} for the next iteration, otherwise it is reduced in (b). In both cases, we take a step g starting from the current point $\tilde{x}$, and project back to an ϵ_{k+1}-sphere centered at x.

Given the difficulty of finding the appropriate constant C for this optimization, we propose an algorithm that does not impose a penalty on the L_2 norm during the optimization. Instead, the norm is constrained by projecting the adversarial perturbation δ on an ϵ-sphere around the original image x. Then, the L_2 norm is modified through a binary decision. If the sample x_k is not adversarial at step k, the norm is increased for step $k+1$, otherwise it is decreased.

We also note that optimizing the cross-entropy may present two other difficulties. First, the function is not bounded, which can make it dominate in the optimization of Eq. 3. Second, when attacking trained models, often the predicted probability of the correct class for the original image is very close to 1, which causes the cross entropy to start very low and increase by several orders of magnitude during the search for an adversarial example. This affects the norm of the gradient, making it hard to find an appropriate learning rate. C&W address these issues by optimizing the difference between logits instead of the cross-entropy. In this work, the issue of it being unbounded does not affect the attack procedure, since the decision to update the norm is done on the model's prediction (not on the cross-entropy). In order to handle the issue of large changes in gradient norm, we normalize the gradient to have unit norm before taking a step in its direction.

The full procedure is described in Algorithm 1 and illustrated in Fig. 3. We start from the original image x, and iteratively refine the noise δ_k. In iteration k, if the current sample $\tilde{x}_k = x + \delta_k$ is still not adversarial, we consider a larger norm $\epsilon_{k+1} = (1+\gamma)\epsilon_k$. Otherwise, if the sample is adversarial, we consider a smaller $\epsilon_{k+1} = (1-\gamma)\epsilon_k$. In both cases, we take a step g (step 5 of Algorithm 1) from the point $\tilde{x}_k$ (red arrow in Fig. 3), and project it back onto an ϵ_{k+1}-sphere centered at x (the direction given by the dashed blue line in Fig. 3), obtaining $\tilde{x}_{k+1}$. Lastly, $\tilde{x}_{k+1}$ is projected onto the feasible region of the input space $\mathcal{X}$. In the case of images normalized to $[0,1]$, we simply clip the value of each pixel to be inside this range (step 13 of Algorithm 1). Besides this step, we can also consider quantizing the image in each iteration, to ensure the attack is a valid image.

It's worth noting that, when reaching a point where the decision boundary is tangent to the ϵ_k-sphere, g will have the same direction as δ_{k+1}. This means that δ_{k+1} will be projected on the direction of δ_k. Therefore, the norm will oscillate between the two sides of the decision boundary in this direction. Multiplying ϵ by $1+\gamma$ and $1-\gamma$ will result in a global decrease (on two steps) of the norm by $1-\gamma^2$, leading to a finer search of the best norm.

4. Attack Evaluation

Experiments were conducted on the MNIST, CIFAR-10 and ImageNet datasets, comparing the proposed attack to the state-of-the-art L_2 attacks proposed in the literature: DeepFool [14] and C&W L_2 attack [5]. We use the same model architectures and hyperparameters for training as in [5] for MNIST and CIFAR-10 (see the supplementary material for details). Our base classifiers obtain 99.44% and 85.51% accuracy on the test sets of MNIST and CIFAR-10, respectively. For the ImageNet experiments, we use a pre-trained Inception V3 [17], that achieves 22.51% top-1 error on the validation set. Inception V3 takes images of size 299×299 as input, which are cropped from images of size 342×342.

For experiments with DeepFool [14], we used the implementation from Foolbox [16], with a budget of 100 iterations. For the experiments with C&W, we ported the attack (originally implemented on TensorFlow) on PyTorch to evaluate the models in the frameworks in which they were trained. We use the same hyperparameters from [5]: 9 search steps on C with an initial constant of 0.01, with 10 000 iterations for each search step (with early stopping) - we refer to this scenario as C&W 9×10 000 in the tables. As we are interested in obtaining attacks that require few iterations, we also report experiments in a scenario where the number of iterations is limited to 100. We consider a scenario of running 100 steps with a fixed C (1×100), and a scenario of running 4 search steps on C, of 25 iterations each (4×25). Since the hyperparameters proposed in [5] were tuned for a larger number of iterations and search steps, we performed a grid search for each dataset, using learning rates in the range [0.01, 0.05, 0.1, 0.5, 1], and C in the range [0.001, 0.01, 0.1, 1, 10, 100, 1 000]. We report the results for C&W with the hyperparameters that achieve best Median L_2. Selected parameters are listed in the supplementary material.

For the experiments using DDN, we ran attacks with budgets of 100, 300 and 1 000 iterations, in all cases, using $\epsilon_0 = 1$ and $\gamma = 0.05$. The initial step size $\alpha = 1$, was reduced with cosine annealing to 0.01 in the last iteration. The choice of γ is based on the encoding of images. For any correctly classified image, the smallest possible perturbation consists in changing one pixel by $1/255$ (for images encoded in 8 bit values), corresponding to a norm of $1/255$. Since we perform quantization, the values are rounded, meaning that the algorithm must be able to achieve a norm lower than $1.5/255 = 3/510$. When using K steps, this imposes:

$$\epsilon_0(1-\gamma)^K < \frac{3}{510} \Rightarrow \gamma > 1 - \left(\frac{3}{510\,\epsilon_0}\right)^{\frac{1}{K}} \tag{8}$$

Using $\epsilon_0 = 1$ and $K = 100$ yields $\gamma \simeq 0.05$. Therefore, if there exists an adversarial example with smallest perturbation, the algorithm may find it in a fixed number of steps.

For the results with DDN, we consider quantized images (to 256 levels). The quantization step is included in each iteration (see step 13 of Algorithm 1). All results reported in the paper consider images in the $[0, 1]$ range.

Two sets of experiments were conducted: untargeted attacks and targeted attacks. As in [5], we generated attacks on the first 1 000 images of the test set for MNIST and CIFAR-10, while for ImageNet we randomly chose 1 000 images from the validation set that are correctly classified. For the untargeted attacks, we report the success rate of the attack (percentage of samples for which an attack was found), the mean L_2 norm of the adversarial noise (for successful attacks), and the median L_2 norm over all attacks while considering unsuccessful attacks as worst-case adversarial

	Attack	Budget	Success	Mean L_2	Median L_2	#Grads	Run-time (s)
MNIST	C&W	4×25	100.0	1.7382	1.7400	100	1.7
		1×100	99.4	1.5917	1.6405	100	1.7
		9×10 000	100.0	**1.3961**	1.4121	54 007	856.8
	DeepFool	100	75.4	1.9685	2.2909	98	-
	DDN	100	100.0	1.4563	1.4506	100	1.5
		300	100.0	1.4357	1.4386	300	4.5
		1 000	100.0	1.4240	1.4342	1 000	14.9
CIFAR-10	C&W	4×25	100.0	0.1924	0.1541	60	3.0
		1×100	99.8	0.1728	0.1620	91	4.6
		9×10 000	100.0	0.1543	0.1453	36 009	1 793.2
	DeepFool	100	99.7	0.1796	0.1497	25	-
	DDN	100	100.0	0.1503	0.1333	100	4.7
		300	100.0	0.1487	0.1322	300	14.2
		1 000	100.0	**0.1480**	0.1317	1 000	47.6
ImageNet	C&W	4×25	100.0	1.5812	1.3382	63	379.3
		1×100	100.0	0.9858	0.9587	48	287.1
		9×10 000	100.0	0.4692	0.3980	21 309	127 755.6
	DeepFool	100	98.5	0.3800	0.2655	41	-
	DDN	100	99.6	0.3831	0.3227	100	593.6
		300	100.0	0.3749	0.3210	300	1 779.4
		1 000	100.0	**0.3617**	0.3188	1 000	5 933.6

Table 1: Performance of our DDN attack compared to C&W [5] and DeepFool [14] attacks on MNIST, CIFAR-10 and ImageNet in the untargeted scenario.

	Average case		Least Likely	
Attack	Success	Mean L_2	Success	Mean L_2
C&W 4×25	96.11	2.8254	69.9	5.0090
C&W 1×100	86.89	2.0940	31.7	2.6062
C&W 9×10 000	100.00	1.9481	100.0	2.5370
DDN 100	100.00	1.9763	100.0	2.6008
DDN 300	100.00	1.9577	100.0	2.5503
DDN 1 000	100.00	1.9511	100.0	2.5348

Table 2: Comparison of the DDN attack to the C&W L_2 attack on MNIST.

	Average case		Least Likely	
Attack	Success	Mean L_2	Success	Mean L_2
C&W 4×25	99.78	0.3247	98.7	0.5060
C&W 1×100	99.32	0.3104	95.8	0.4159
C&W 9×10 000	100.00	0.2798	100.0	0.3905
DDN 100	100.00	0.2925	100.0	0.4170
DDN 300	100.00	0.2887	100.0	0.4090
DDN 1 000	100.00	0.2867	100.0	0.4050

Table 3: Comparison of the DDN attack to the C&W L_2 attack on CIFAR-10.

	Average case		Least Likely	
Attack	Success	Mean L_2	Success	Mean L_2
C&W 4×25	99.13	4.2826	80.6	8.7336
C&W 1×100	96.74	1.7718	66.2	2.2997
C&W 9×10 000 [5]	100.00	0.96	100.0	2.22
DDN 100	99.98	1.0260	99.5	1.7074
DDN 300	100.00	0.9021	100.0	1.3634
DDN 1 000	100.00	0.8444	100.0	1.2240

Table 4: Comparison of the DDN attack to the C&W L_2 attack on ImageNet. For C&W 9×10 000, we report the results from [5].

(distance to a uniform gray image, as in [3]). We also report the average number (for batch execution) of gradient computations and the total run-times (in seconds) on a NVIDIA GTX 1080 Ti with 11GB of memory. We did not report run-times for the DeepFool attack, since the implementation from foolbox generates adversarial examples one-by-one and is executed on CPU, leading to unrepresentative run-times. Attacks on MNIST and CIFAR-10 have been executed in a single batch of 1 000 samples, whereas attacks on ImageNet have been executed in 20 batches of 50 samples.

For the targeted attacks, following the protocol from [5], we generate attacks against all possible classes on MNIST and CIFAR-10 (9 attacks per image), and against 100 randomly chosen classes for ImageNet (10% of the number of classes). Therefore, in each targeted attack experiment, we run 9 000 attacks on MNIST and CIFAR-10, and 100 000 attacks on ImageNet. Results are reported for two scenarios: 1) average over all attacks; 2) average performance when choosing the least likely class (i.e. choosing the worst attack performance over all target classes, for each image). The reported L_2 norms are, as in the untargeted scenario, the means over successful attacks.

Table 1 reports the results of DDN compared to the C&W L_2 and DeepFool attacks on the MNIST, CIFAR-10 and ImageNet datasets. For the MNIST and CIFAR-10 datasets, results with DDN are comparable to the state-of-the-art. DDN obtains slightly worse L_2 norms on the MNIST dataset (when compared to the C&W 9×10 000), however, our attack is able to get within 5% of the norm found by C&W in only 100 iterations compared to the 54 007 iterations required for the C&W L_2 attack. When the C&W attack is restricted to use a maximum of 100 iterations, it always performed worse than DDN with 100 iterations. On the ImageNet dataset, our attack obtains better Mean L_2 norms than both other attacks. The DDN attack needs 300 iterations to reach 100% success rate. DeepFool obtains close results but fails to reach 100% success rate. It is also worth noting that DeepFool seems to performs worse against adversarially trained models (discussed in Section 6). Supplementary material reports curves of the perturbation size against accuracy of the models for the three attacks.

Tables 2, 3 and 4 present the results on targeted attacks on the MNIST, CIFAR-10 and ImageNet datasets, respectively. For the MNIST and CIFAR-10 datasets, DDN yields similar performance compared to the C&W attack with 9×10 000 iterations, and always perform better than the C&W attack when it is restricted to 100 iterations (we re-iterate that the hyperparameters for the C&W attack were tuned for each dataset, while the hyperparameters for DDN are fixed for all experiments). On the ImageNet dataset, DDN run with 100 iterations obtains superior performance than C&W. For all datasets, with the scenario restricted to 100 iterations, the C&W algorithm has a noticeable drop in success rate for finding adversarial examples to the least likely class.

5. Adversarial Training with DDN

Since the DDN attack can produce adversarial examples in relatively few iterations, it can be used for adversarial training. For this, we consider the following loss function:

$$\tilde{J}(x, y, \theta) = J(\tilde{x}, y, \theta) \tag{9}$$

where $\tilde{x}$ is an adversarial example produced by the DDN algorithm, that is projected to an ϵ-ball around x, such that the classifier is trained with adversarial examples with a maximum norm of ϵ. It is worth making a parallel of this approach with the Madry defense [13] where, in each iteration, the loss of the worst-case adversarial (see Eq. 2) in an ϵ-ball around the original sample x is used for optimization. In our proposed adversarial training procedure, we optimize the loss of the closest adversarial example (see Eq. 1). The intuition of this defense is to push the decision boundary away from x in each iteration. We do note that this method does not have the theoretical guarantees of the Madry defense. However, since in practice the Madry defense uses approximations (when searching for the global maximum of the loss around x), we argue that both methods deserve empirical comparison.

Defense	Attack	Attack Success	Mean L_2	Median L_2	Model Accuracy at $\epsilon \leq 1.5$
Baseline	C&W 9×10 000	100.0	1.3961	1.4121	42.1
	DeepFool 100	75.4	1.9685	2.2909	81.8
	DDN 1 000	100.0	1.4240	1.4342	45.2
	All	100.0	1.3778	1.3946	40.8
Madry *et al.*	C&W 9×10 000	100.0	2.0813	2.1071	73.0
	DeepFool 100	91.6	4.9585	5.2946	93.1
	DDN 1 000	99.6	1.8436	1.8994	69.9
	All	100.0	1.6917	1.8307	67.3
Ours	C&W 9×10 000	100.0	2.5181	2.6146	88.0
	DeepFool 100	94.3	3.9449	4.1754	92.7
	DDN 1 000	100.0	2.4874	2.5781	87.6
	All	100.0	**2.4497**	2.5538	**87.2**

Table 5: Evaluation of the robustness of our adversarial training on MNIST against the Madry defense.

6. Defense Evaluation

We trained models using the same architectures as [5] for MNIST, and a Wide ResNet (WRN) 28-10 [21] for CIFAR-10 (similar to [13] where they use a WRN 34-10). As described in Section 5, we augment the training images with adversarial perturbations. For each training step, we run the DDN attack with a budget of 100 iterations, and limit the norm of the perturbation to a maximum $\epsilon = 2.4$ on the MNIST experiments, and $\epsilon = 1$ for the CIFAR-10 experiments. For MNIST, we train the model for 30 epochs with a learning rate of 0.01 and then for 20 epochs with a learning rate of 0.001. To reduce the training time with CIFAR-10, we first train the model on original images for 200 epochs using the hyperparameters from [21]. Then, we continue training for 30 more epochs using Eq. 9, keeping the same final learning rate of 0.0008. Our robust MNIST model has a test accuracy of 99.01% on the clean samples, while the Madry model has an accuracy of 98.53%. On CIFAR-10, our model reaches a test accuracy of 89.0% while the model by Madry *et al.* obtains 87.3%.

We evaluate the adversarial robustness of the models using three untargeted attacks: Carlini 9×10 000, DeepFool 100 and DDN 1 000. For each sample, we consider the smallest adversarial perturbation produced by the three attacks and report it in the "**All**" row. Tables 5 and 6 report the results of this evaluation with a comparison to the defense of Madry *et al.* [13][3] and the baseline (without adversarial training) for CIFAR-10. For MNIST, the baseline corresponds

[3] Models taken from https://github.com/MadryLab

Defense	Attack	Attack Success	Mean L_2	Median L_2	Model Accuracy at $\epsilon \leq 0.5$
Baseline WRN 28-10	C&W 9×10 000	100.0	0.1343	0.1273	0.2
	DeepFool 100	99.3	0.5085	0.4241	38.3
	DDN 1 000	100.0	0.1430	0.1370	0.1
	All	100.0	0.1282	0.1222	0.1
Madry *et al.* WRN 34-10	C&W 9×10 000	100.0	0.6912	0.6050	57.1
	DeepFool 100	95.6	1.4856	0.9576	64.7
	DDN 1 000	100.0	0.6732	0.5876	56.9
	All	100.0	0.6601	0.5804	56.1
Ours WRN 28-10	C&W 9×10 000	100.0	0.8860	0.8254	67.9
	DeepFool 100	99.7	1.5298	1.1163	69.9
	DDN 1 000	100.0	0.8688	0.8177	68.0
	All	100.0	**0.8597**	0.8151	**67.6**

Table 6: Evaluation of the robustness of our adversarial training on CIFAR-10 against the Madry defense.

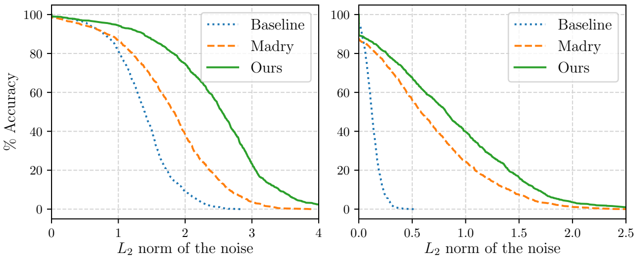


Figure 4: Models robustness on MNIST (left) and CIFAR-10 (right): impact on accuracy as we increase the maximum perturbation ϵ.

to the model used in Section 4. We observe that for attacks with unbounded norm, the attacks can successfully generate adversarial examples almost 100% of the time. However, an increased L_2 norm is required to generate attacks against the model trained with DDN.

Fig. 4 shows the robustness of the MNIST and CIFAR-10 models respectively for different attacks with increasing maximum L_2 norm. These figures can be interpreted as the expected accuracy of the systems in a scenario where the adversary is constrained to make changes with norm $L_2 \leq \epsilon$. For instance on MNIST, if the attacker is limited to a maximum norm of $\epsilon = 1.5$, the baseline performance decreases to 40.8%; Madry to 67.3% and our defense to 87.2%. At $\epsilon = 2.0$, baseline performance decreases to 9.2%, Madry to 38.6% and our defense to 74.8%. On CIFAR-10, if the attacker is limited to a maximum norm of $\epsilon = 0.5$, the baseline performance decreases to 0.1%; Madry to 56.1% and our defense to 67.6%. At $\epsilon = 1.0$, baseline performance decreases to 0%, Madry to 24.4% and our defense to 39.9%. For both datasets, the model trained with DDN outperforms the model trained with the Madry defense for all values of ϵ.

Fig. 5 shows adversarial examples produced by the DDN 1 000 attack for different models on MNIST and CIFAR-10. On MNIST, adversarial examples for the baseline are not meaningful (the still visually belong to the original class), whereas some adversarial examples obtained for the adversarially trained model (DDN) actually change classes (bottom right: 0 changes to 6). For all models, there are still some adversarial examples that are very close to the original images (first column). On CIFAR-10, while the adversarially trained models require higher norms for the attacks, most adversarial examples still perceptually resemble the original images. In few cases (bottom-right example for CIFAR-10), it could cause a confusion: it can appear as changing to class 1 - a (cropped) automobile facing right.

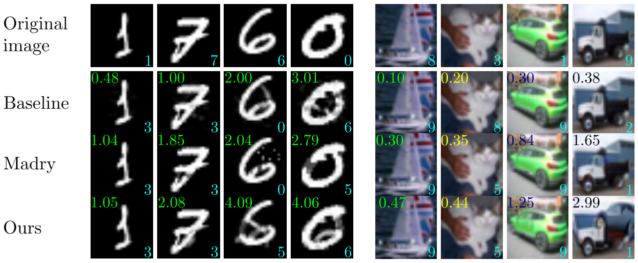


Figure 5: Adversarial examples with varied levels of noise δ against three models: baseline, Madry defense [13] and our defense. Text on top-left of each image indicate $\|\delta\|_2$; text on bottom-right indicates the predicted class[4].

7. Conclusion

We presented the *Decoupled Direction and Norm* attack, which obtains comparable results with the state-of-the-art for L_2 norm adversarial perturbations, but in much fewer iterations. Our attack allows for faster evaluation of the robustness of differentiable models, and enables a novel adversarial training, where, at each iteration, we train with examples close to the decision boundary. Our experiments with MNIST and CIFAR-10 show state-of-the-art robustness against L_2-based attacks in a white-box scenario.

The methods presented in this paper were used in NIPS 2018 Adversarial Vision Challenge [3], ranking first in untargeted attacks, and third in targeted attacks and robust models (both attacks and defense in a black-box scenario). These results highlight the effectiveness of the defense mechanism, and suggest that attacks using adversarially-trained surrogate models can be effective in black-box scenarios, which is a promising future direction.

Acknowledgements

We thank Marco Pedersoli and Christian Desrosiers for their insightful feedback. This research was supported by the Fonds de recherche du Québec - Nature et technologies, Natural Sciences and Engineering Research Council of Canada, and CNPq grant #206318/2014-6.

[4] For CIFAR-10: 1: automobile, 2: bird, 3: cat, 5: dog, 8: ship, 9: truck.

References

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 274–283, 2018.

[2] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, Dec. 2018.

[3] W. Brendel, J. Rauber, A. Kurakin, N. Papernot, B. Veliqi, M. Salathé, S. P. Mohanty, and M. Bethge. Adversarial vision challenge. *arXiv:1808.01976*, 2018.

[4] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. In *International Conference on Learning Representations*, 2018.

[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[6] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar. Stochastic Activation Pruning for Robust Adversarial Defense. In *International Conference on Learning Representations*, 2018.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

[8] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, May 2018.

[9] C. Guo, M. Rana, M. Cissé, and L. van der Maaten. Countering Adversarial Images using Input Transformations. In *International Conference on Learning Representations*, 2018.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[11] P. A. Jensen and J. F. a. Bard. *Operations Research Models and Methods*. Wiley, 2003.

[12] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations (workshop track)*, 2017.

[13] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *International Conference on Learning Representations*, 2018.

[14] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[15] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016.

[16] J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv:1707.04131*, 2017.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[19] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations*, 2018.

[20] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille. Mitigating Adversarial Effects Through Randomization. In *International Conference on Learning Representations*, 2018.

[21] S. Zagoruyko and N. Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, pages 87.1–87.12, 2016.

Supplementary material

A. Model architectures

Table 7 lists the architectures of the CNNs used in the Attack Evaluation - we used the same architecture as in [5] for a fair comparison against the C&W and DeepFool attacks. Table 8 lists the architecture used in the robust model (defense) trained on CIFAR-10. We used a Wide ResNet with 28 layers and widening factor of 10 (WRN-28-10). The residual blocks used are the “basic block" [10, 21], with stride 1 for the first group and stride 2 for the second an third groups. This architecture is slightly different from the one used by Madry *et al*. [13], where they use a modified version of Wide ResNet with 5 residual blocks instead of 4 in each group, and without convolutions in the residual connections (when the shape of the output changes, e.g. with stride=2).

B. Hyperparameters selected for the C&W attack

We considered a scenario of running the C&W attack with 100 steps and a fixed C (1×100), and a scenario of running 4 search steps on C, of 25 iterations each (4×25). Since the hyperparameters proposed in [5] were tuned for a larger number of iterations and search steps, we performed a grid search for each dataset, using learning rates in the range [0.01, 0.05, 0.1, 0.5, 1], and C in the range [0.001, 0.01, 0.1, 1, 10, 100, 1 000]. We selected the hyperparameters that resulted in targeted attacks with lowest Median L_2 for each dataset. Table 9 lists the hyperparameters found through this search procedure.

C. Examples of adversarial images

Fig. 6 plots a grid of attacks (obtained with the C&W attack) against the first 10 examples in the MNIST dataset. The rows indicate the source classification (label), and the columns indicate the target class used to generate the attack (images on the diagonal are the original samples). We can see that in the adversarially trained model, the attacks need to introduce much larger changes to the samples in order to make them adversarial, and some of the adversarial samples visually resemble another class.

Layer Type	MNIST Model	CIFAR-10 Model
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Max Pooling	2×2	2×2
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Max Pooling	2×2	2×2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Fully Connected + Softmax	10	10

Table 7: CNN architectures used for the Attack Evaluation

Layer Type	Size
Convolution	$3 \times 3 \times 16$
Residual Block	$\begin{bmatrix} 3 \times 3, 160 \\ 3 \times 3, 160 \end{bmatrix} \times 4$
Residual Block	$\begin{bmatrix} 3 \times 3, 320 \\ 3 \times 3, 320 \end{bmatrix} \times 4$
Residual Block	$\begin{bmatrix} 3 \times 3, 640 \\ 3 \times 3, 640 \end{bmatrix} \times 4$
Batch Normalization + ReLU	-
Average Pooling	8×8
Fully Connected + Softmax	10

Table 8: CIFAR-10 architecture used for the Defense evaluation

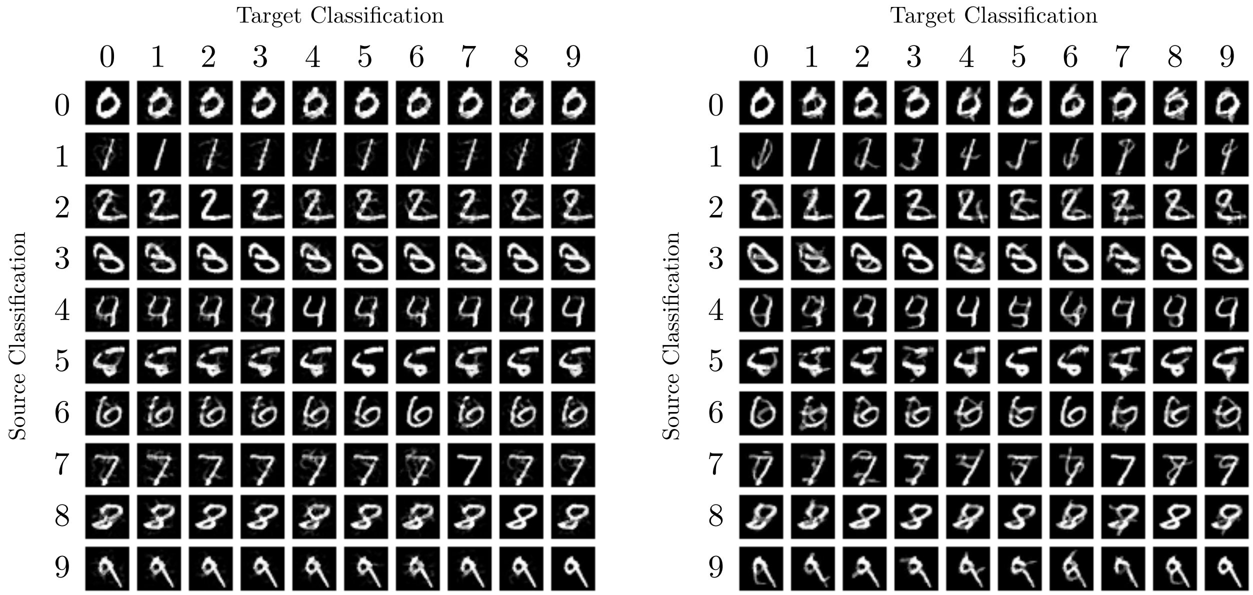
Dataset	# Iterations	Parameters
MNIST	1×100	$\alpha = 0.1, C = 1$
MNIST	4×25	$\alpha = 0.5, C = 1$
CIFAR-10	1×100	$\alpha = 0.01, C = 0.1$
CIFAR-10	4×25	$\alpha = 0.01, C = 0.1$
ImageNet	1×100	$\alpha = 0.01, C = 1$
ImageNet	4×25	$\alpha = 0.01, C = 10$

Table 9: Hyperparameters used for the C&W attack when restricted to 100 iterations.

Fig. 7 shows randomly-selected adversarial examples for the CIFAR-10 dataset, comparing the baseline model (WRN 28-10), the Madry defense and our proposed defense. For each image and model, we ran three attacks (DDN 1 000, C&W 9×10 000, DeepFool 100), and present the adversarial example with minimum L_2 perturbation among them. Fig. 8 shows cherry-picked adversarial examples on CIFAR-10, that visually resemble another class, when attacking the proposed defense. We see that on the average case (randomly-selected), adversarial examples against the defenses still require low amounts of noise (perceptually) to induce misclassification. On the other hand, we notice that on adversarially trained models, some examples do require a much larger change on the image, making it effectively resemble another class.

D. Attack performance curves

Fig. 9 reports curves of the perturbation size against accuracy of the models for three attacks: Carlini 9×10 000, DeepFool 100 and DDN 300.



(a) Baseline (without adversarial training)

(b) Adversarially trained

Figure 6: Adversarial examples obtained using the C&W L_2 attack on two models: (a) Baseline, (b) model adversarially trained with our attack.

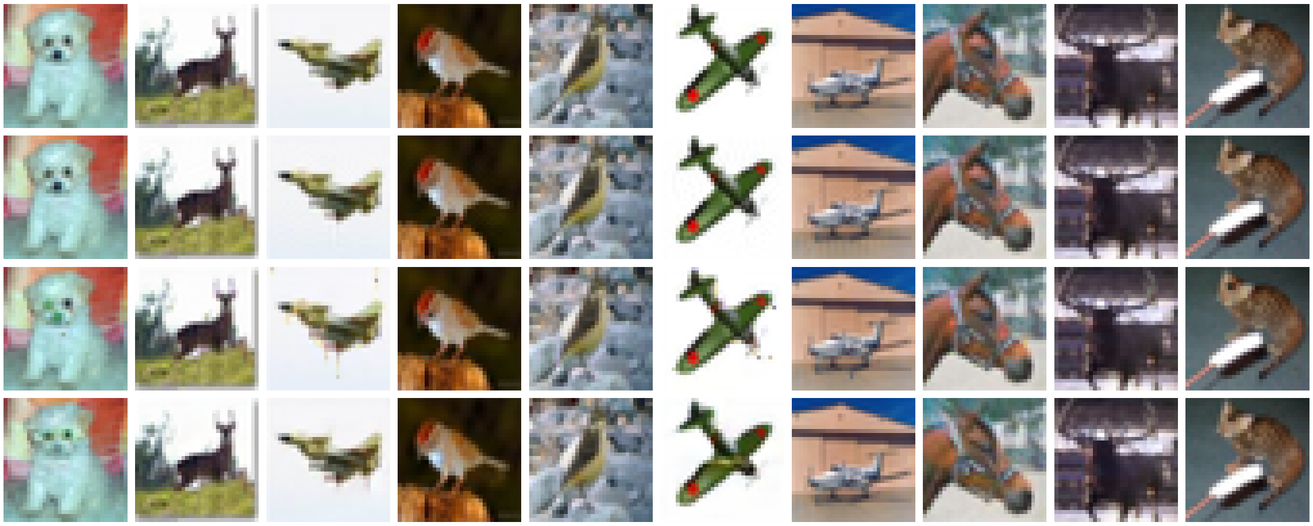


Figure 7: Randomly chosen adversarial examples on CIFAR-10 for three models. **Top row**: original images; **second row**: attacks against the baseline; **third row**: attacks against the Madry defense.

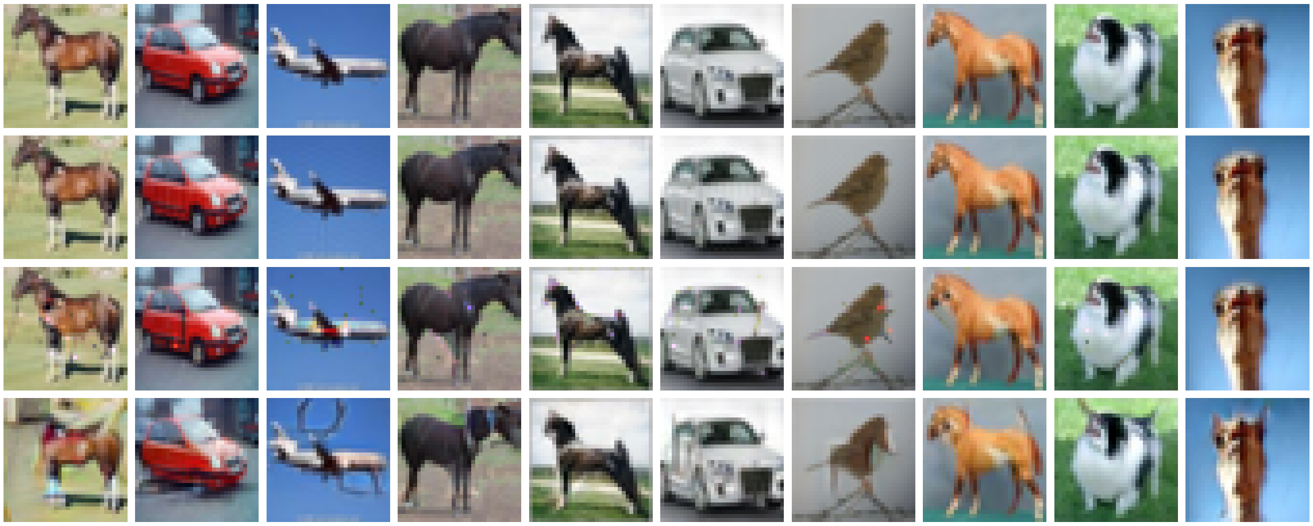
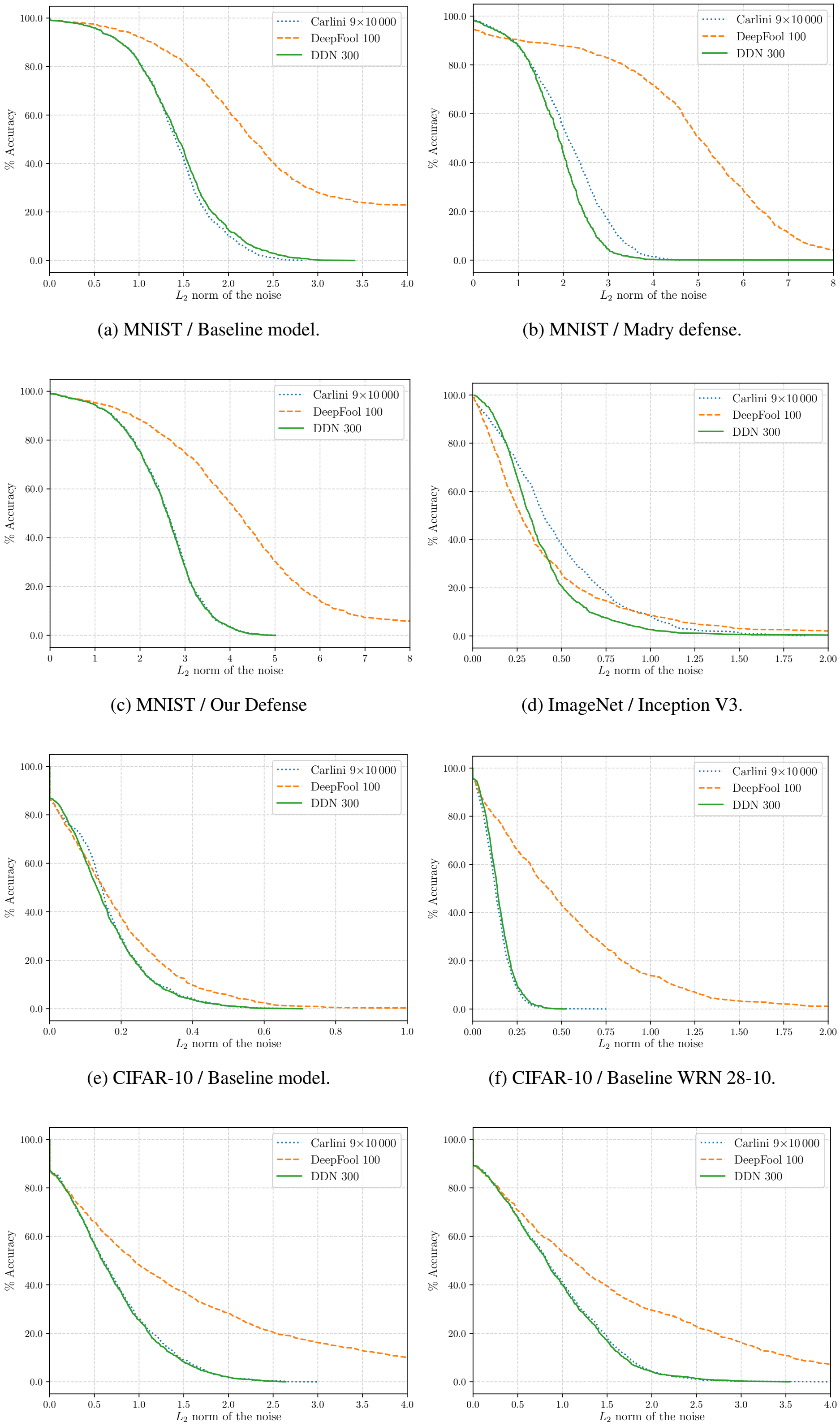


Figure 8: Cherry-picked adversarial examples on CIFAR-10 for three models. **Top row**: original images; **second row**: attacks against the baseline; **third row**: attacks against the Madry defense; **bottom row**: attacks against the proposed defense. Predicted labels for the last row are, from left to right: dog, ship, deer, dog, dog, truck, horse, dog, cat, cat.



(a) MNIST / Baseline model.

(b) MNIST / Madry defense.

(c) MNIST / Our Defense

(d) ImageNet / Inception V3.

(e) CIFAR-10 / Baseline model.

(f) CIFAR-10 / Baseline WRN 28-10.

(g) CIFAR-10 / Madry defense.

(h) CIFAR-10 / Our Defense.

Figure 9: Attacks performances on different datasets and models.