Towards Stable and Efficient Adversarial Training against l_1 Bounded Adversarial Attacks

Yulun Jiang [*1] **Chen Liu** [*2] **Zhichao Huang** [3] **Mathieu Salzmann** [1] **Sabine Süsstrunk** [1]

Abstract

We address the problem of stably and efficiently training a deep neural network robust to adversarial perturbations bounded by an l_1 norm. We demonstrate that achieving robustness against l_1-bounded perturbations is more challenging than in the l_2 or l_∞ cases, because adversarial training against l_1-bounded perturbations is more likely to suffer from catastrophic overfitting and yield training instabilities. Our analysis links these issues to the coordinate descent strategy used in existing methods. We address this by introducing **Fast-EG-l_1**, an efficient adversarial training algorithm based on Euclidean geometry and free of coordinate descent. Fast-EG-l_1 comes with no additional memory costs and no extra hyper-parameters to tune. Our experimental results on various datasets demonstrate that Fast-EG-l_1 yields the best and most stable robustness against l_1-bounded adversarial attacks among the methods of comparable computational complexity. Code and the checkpoints are available at https://github.com/IVRL/FastAdvL1.

1. Introduction

Comprehensive evaluations (Athalye et al., 2018; Croce & Hein, 2020b; 2021) have shown that adversarial training (Madry et al., 2018) and its variants (Alayrac et al., 2019; Zhang et al., 2019b; Carmon et al., 2019; Wu et al., 2020; Rebuffi et al., 2021) are the most effective approaches to counteract the vulnerability of deep neural networks to adversarial perturbations (Szegedy et al., 2014; Goodfellow et al., 2014; Moosavi-Dezfooli et al., 2017). At the heart of adversarial training lies the idea of generating adversarial examples by perturbing the training inputs and using these examples to update the model parameters.

While effective, adversarial training is typically not efficient. For example, the pioneering work (Madry et al., 2018) generates adversarial examples by projected gradient descent (PGD), which requires several forward and backward passes, thus significantly increasing the computational cost. Several works (Shafahi et al., 2019; Zhang et al., 2019a; Wong et al., 2020; Chen et al., 2022) have aimed to improve the efficiency of adversarial training by using weaker but computationally-cheaper adversarial examples, such as the ones generated by one-step attacks. However, such efficient adversarial training methods tend to suffer from *catastrophic overfitting*: The models overfit to the weaker attacks used during training and lose true robustness, resulting in trivial performance under stronger attacks.

Note that the above-mentioned works focus on the scenario where the perturbations applied to the model input are bounded by either an l_2 or an l_∞ adversarial budget. While such budgets have been well studied and benchmarked (Croce et al., 2021), robustness against l_1-bounded adversarial attacks remains comparatively underexplored.

In this work, we show that true robustness and efficiency can jointly be achieved under l_1 adversarial budgets. To the best of our knowledge, our work constitutes the first method designed for efficient adversarial training given l_1 bounded attacks. It is a challenging task because achieving the optimal robust accuracy under an l_1 adversarial budget (Croce & Hein, 2021) is computationally even more expensive than under the l_2 and l_∞ ones (Gowal et al., 2020). Furthermore, we evidence that adversarial training against l_1 bounded perturbations is less stable. Specifically, our empirical observations indicate that catastrophic overfitting occurs more frequently under l_1 adversarial budgets than in the l_2 and l_∞ cases, and that, in contrast with the l_2 and l_∞ cases, it occurs even when using multi-step attacks for adversarial training. In addition, we demonstrate that directly applying the existing efficient adversarial training methods to l_1 adversarial budgets yields poor performance because of instability.

To address these drawbacks, we first investigate why catas-

[*]Equal contribution [1]School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland [2]Department of Computer Science, City University of Hong Kong, Hong Kong, China [3]ByteDance. Correspondence to: Yulun Jiang <yulun.jiang@epfl.ch>, Chen Liu <cliu644@cityu.edu.hk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

trophic overfitting is more likely to occur under l_1 adversarial budgets. Our analysis links this to the coordinate descent strategy used in existing methods for l_1 robustness: coordinate descent incurs a strong bias toward generating sparse perturbations, which leaves the models vulnerable to dense ones. Therefore, we propose gradient descent based on Euclidean geometry to generate adversarial examples to mitigate catastrophic overfitting. That is to say, we use a gradient descent method similar to the l_2 case to search for stronger attacks within the l_1 adversarial budget. The step size of this gradient descent method is calculated to ensure one update in our method can cover possible updates in methods based on the coordinate descent. Furthermore, to better explore the low dimensional faces of the l_1 ball, we use the *multi-*ϵ trick, which entails a much larger adversarial budget for training. We show that our method, which uses one-step attacks and is called **Fast-EG-**l_1, enables stable and efficient adversarial training against l_1 bounded attacks. Compared with existing efficient adversarial training methods, which were developed for l_2 or l_∞ adversarial budgets, Fast-EG-l_1 yields better robust models against l_1 bounded attacks with fewer hyper-parameters, negligible overhead in both computation and memory consumption, and without exhibiting catastrophic overfitting.

To the best of our knowledge, our work is the first one to systematically study efficient adversarial training methods in the context of l_1 adversarial budgets. Our work encompasses the following contributions:

- We demonstrate that catastrophic overfitting occurs more frequently in the context of l_1 adversarial budgets. Our analysis connects this problem to the coordinate descent strategy used in existing methods. Due to its bias towards sparse perturbations, coordinate descent cannot generate strong perturbations for training without carefully tuning hyper-parameters.
- We propose **Fast-EG-**l_1, an efficient adversarial training method based on Euclidean geometry and thus free of coordinate descent. Compared with existing methods, Fast-EG-l_1 has fewer hyper-parameters, negligible overhead in both computation and memory.
- Our extensive analyses on diverse datasets demonstrate that Fast-EG-l_1 yields the best robustness against l_1-bounded attacks among the methods with similar computational complexity. Moreover, Fast-EG-l_1 is stable and does not suffer from catastrophic overfitting.

2. Background and Related Works

A deep neural network can be represented by a function $f : \Theta \times \mathbb{R}^M \to \mathbb{R}^C$, which maps an M-dimensional input $\boldsymbol{x}$ to a C-dimensional output y and is parameterized by $\theta \in \Theta$. Given a training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ and a loss function l, normal training, *i.e.*, training without attacks, aims to find the optimal θ to minimize the empirical risk $\frac{1}{N}\sum_{i=1}^N l(f(\theta, \boldsymbol{x}_i), y_i)$. Since we do not change the label of the instances in this work, we drop y_i in the formulation for notation simplicity. Furthermore, we use $\mathcal{L}$ to denote the composition function of l and f: $\mathcal{L} = l \circ f$. Then, the empirical risk is $\frac{1}{N}\sum_{i=1}^N \mathcal{L}(\theta, \boldsymbol{x}_i)$.

In robust learning, we use the *adversarial budget* to define the set of all allowable perturbations. We consider an l_p adversarial budget, parameterized by its size ϵ: $\mathcal{S}_\epsilon^{(p)} := \{\Delta \mid \|\Delta\|_p \leq \epsilon\}$. Therefore, robust learning solves the min-max optimization problem:

$$\min_\theta \frac{1}{N}\sum_{i=1}^N \max_{\Delta \in \mathcal{S}_\epsilon^{(p)}} \mathcal{L}(\theta, \boldsymbol{x}_i + \Delta)\ . \tag{1}$$

2.1. Training on Different Adversarial Budgets

The deep neural network f is high-dimensional and non-convex, and Weng et al. (2018) have proven that solving problem (1) is at least NP-complete. Despite these challenges, adversarial training (Madry et al., 2018) stands out as a reliable and popular method to obtain robust deep neural networks (Athalye et al., 2018; Croce & Hein, 2020b).

As the name indicates, adversarial training learns the model parameters using adversarial examples generated on the fly. This requires an attack algorithm to generate adversarial examples during training. Almost all existing powerful attack algorithms (Madry et al., 2018; Croce & Hein, 2020a;b; 2021) are based on gradient descent or ascent. Specifically, they consider the first-order Taylor expansion of the loss function in (1) and then maximize the linear approximation:

$$\begin{aligned}\max_{\Delta \in \mathcal{S}_\epsilon^{(p)}} \mathcal{L}(\theta, \boldsymbol{x}_i + \Delta) \simeq \mathcal{L}(\theta, \boldsymbol{x}_i) + \max_{\Delta \in \mathcal{S}_\epsilon^{(p)}} \langle \Delta, \boldsymbol{g} \rangle \\ \text{where } \boldsymbol{g} = \nabla_\Delta \mathcal{L}(\theta, \boldsymbol{x}_i + \Delta)|_{\Delta=0}\ .\end{aligned} \tag{2}$$

The maximum operator on the right-hand side of (2) is applied over an inner product, so one can easily obtain the optimal Δ. Let $\boldsymbol{g}$ represent the gradient of the input. For an l_∞ adversarial budget ($p = \infty$), the optimal Δ is $\epsilon \cdot \text{sign}(\boldsymbol{g})$; for an l_2 adversarial budget ($p = 2$), the optimal Δ is $\epsilon \cdot \boldsymbol{g}/\|\boldsymbol{g}\|_2$ (Maini et al., 2020). Equation (2) considers the Taylor expansion for the entire adversarial budget, which might be too large to obtain an accurate approximation for the nonlinear function $\mathcal{L}$. Alternatively, previous works update Δ in multiple iterations and consider the Taylor expansion within a small l_p ball of radius α, either fixed (Madry et al., 2018) or adaptive (Croce & Hein, 2020b), and centered at the current Δ. After each iteration, Δ is projected to the adversarial budget $\mathcal{S}_\epsilon^{(p)}$. This scheme is referred to as *Projected Gradient Descent* (PGD).

Appendix A.1 provides the pseudo-code and more details for adversarial training using PGD.

For an l_1 adversarial budget, the theoretical optimal Δ is a one-hot vector whose j-th element is ϵ or $-\epsilon$. Here, $j = \text{argmax}_i |\boldsymbol{g}_i|$, and the sign of the perturbation is the same as the sign of $\boldsymbol{g}_j$. That is, one can use coordinate descent to update the adversarial perturbation Δ. However, the input space $\mathcal{I}$ is typically bounded, e.g., image pixel values should be between 0 and 1. Therefore, one-hot coordinate descent not only converges slowly but also perturbs the inputs outside their valid range. As a result, instead of using one-hot coordinate descent, Tramer & Boneh (2019); Maini et al. (2020) consider coordinate descent with K-hot values, with K a hyper-parameter larger than 1. Specifically, Δ is a vector with K non-zero elements, corresponding to the K elements of the gradient $\boldsymbol{g}$ with the largest absolute values. The signs of these non-zero elements in Δ are the same as the corresponding signs of the entries in $\boldsymbol{g}$. To ensure $\|\Delta\|_1 \leq \epsilon$, the absolute values of these non-zero elements are all ϵ/K. Similar to the l_2 and l_∞ cases, we can use Taylor expansion in a small ball and run the K-hot coordinate descent for multiple iterations with a step size α. The resulting Δ after each iteration is projected to the set of allowable perturbations based on the adversarial budget $\mathcal{S}_\epsilon^{(1)}$. This algorithm is referred to as SLIDE (Tramer & Boneh, 2019). We will extensively analyze its performance and compare it with our method, so its pseudo-code is given in Algorithm 1 below. We present training on one instance in the pseudo-codes of this paper for notation simplicity, we use mini-batch training in practice.

Algorithm 1 Adversarial Training with SLIDE

Input: Step size α, number of updated coordinates K, iteration number N_{iter}, adversarial budget $\mathcal{S}_\epsilon^{(1)}$, model parameters θ.
for $(\boldsymbol{x}, y)$ in the training set **do**
  Uniformly sample Δ from $\mathcal{S}_\epsilon^{(1)}$
  for Iteration in $1, 2, ..., N_{iter}$ **do**
    Calculate the gradient $\boldsymbol{g} \leftarrow \nabla_\Delta \mathcal{L}(\theta, \boldsymbol{x} + \Delta)$
    $\mathcal{T} \leftarrow \{i | \boldsymbol{g}_i$ in the top K largest elements of $|\boldsymbol{g}|\}$
    Construct the update $\boldsymbol{d}_i \leftarrow \begin{cases} \alpha/K, & \text{if } i \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases}$
    Update $\Delta \leftarrow \Pi_{\mathcal{S}_\epsilon^{(1)}}(\Delta + \boldsymbol{d})$
  end for
  Calculate the loss $\mathcal{L}(\theta, \boldsymbol{x} + \Delta)$ and update θ
end for
Output: Updated model parameters θ

When optimizing Δ for multiple iterations, the choice of the step size α in each iteration is crucial. For l_1 adversarial budgets, the choice of K is also critical. Croce & Hein (2020b) and Croce & Hein (2021) adaptively adjust these hyper-parameters based on the loss values in the previous iterations and significantly improve the performance for l_∞, l_2 and l_1 adversarial budgets. However, these methods use more iterations than their counterparts with constant step size, which makes them computationally expensive for adversarial training. Therefore, we need efficient methods to make model robust against various attacks.

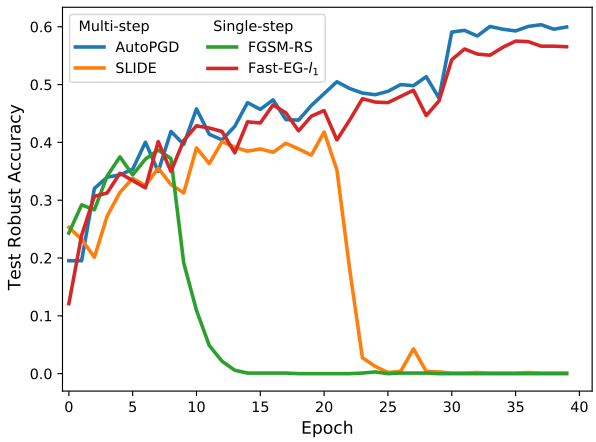

Figure 1. Learning curves of robust accuracy evaluated by 20-step AutoPGD on the CIFAR10 test set. We include both multi-step (10-step AutoPGD, 10-step SLIDE) and single-step (FGSM-RS, our proposed Fast-EG-l_1) adversarial training methods. SLIDE and FGSM-RS suffer from catastrophic overfitting, whereas our method Fast-EG-l_1 achieves competitive performance with that of the computationally much more expensive 10-step AutoPGD.

2.2. Efficient Adversarial Training

Despite being effective, adversarial training converges more slowly (Liu et al., 2020) and has a significant computational overhead caused by generating adversarial perturbations. On the CIFAR10 dataset, compared with normal training, adversarial training in Madry et al. (2018) needs 7 additional forward-backward passes in one mini-batch update. Several methods (Shafahi et al., 2019; Zhang et al., 2019a; Wong et al., 2020; Zheng et al., 2020; Sriramanan et al., 2021; de Jorge et al., 2022) propose to use weaker but faster one-step attacks to improve the efficiency of adversarial training. However, they may suffer from catastrophic overfitting even with careful tuning of hyper-parameters. When catastrophic overfitting happens, the model's robust accuracy on weaker training adversarial examples quickly and significantly increases. However, the robust accuracy on adversarial examples generated by stronger attacks quickly and significantly decreases to 0. That is, the model overfits the weaker adversarial examples used for training instead of achieving true robustness. Catastrophic overfitting is shown to arise from distorted loss landscape in the input space (Kang & Moosavi-Dezfooli, 2021).

Compared with the l_2 and l_∞ cases, the efficiency issue is even more apparent with l_1 adversarial budgets. On the CIFAR10 dataset, Croce & Hein (2021) uses at least 10 additional forward-backward passes to obtain competitive robustness against l_1-bounded attacks. By contrast, the adversarial training method introduced in Madry et al. (2018) uses 7 additional such passes against l_∞-bounded attacks. Added to that, as we show in the next section, catastrophic overfitting occurs more often with l_1 adversarial budgets, including when training against multi-iteration attacks. Therefore, stably and efficiently obtaining robust models against l_1-bounded attacks remains an important challenge.

Although efficient algorithms originally designed for l_2 or l_∞ adversarial budgets are also applicable to l_1 adversarial budgets, in the next sections, we show that applying them to the l_1 case yields sub-optimal performance or even catastrophic overfitting. Therefore, we propose a new method called **Fast-EG-l_1** to improve robustness. The learning curves in Figure 1 demonstrate the effectiveness of our method. To the best of our knowledge, this is the first efficient adversarial training method designed specifically for l_1 adversarial budgets.

3. Catastrophic Overfitting in the Context of l_1 Adversarial Budget

In this section, we study the catastrophic overfitting phenomenon in the context of an l_1 adversarial budget. We use the CIFAR10 dataset (Krizhevsky et al., 2009) and set $\epsilon = 12$, which follows the settings in (Tramer & Boneh, 2019; Croce & Hein, 2021). Following (Wong et al., 2020), we use an 18-layer pre-activation residual network (PreactResNet18) (He et al., 2016a) and train the model for 40 epochs. We use SGD with a momentum factor of 0.9. The learning rate is initialized to 0.05, and is divided by a factor of 10 after the 30th and the 35th epochs.

3.1. Adversarial Training against Multi-step Attacks

We start our investigation with a typical multi-step adversarial training method: SLIDE in Algorithm 1 (Tramer & Boneh, 2019). We use $N_{iter} = 10$ iterations to generate adversarial examples. Besides N_{iter}, SLIDE has two key hyper-parameters: the step size α and the number of coordinates updated per iteration K. Table 1 provides the results of training models under various settings of α and K. For reference, the current state-of-the-art method, training against AutoPGD (Croce & Hein, 2021), using the same number of epochs has a robust accuracy of 55.77%.

SLIDE can be considered an l_1 variant of PGD adversarial training. Compared with the l_2 and l_∞ cases, the observations in Table 1 indicate that it is more challenging to achieve robustness against l_1 adversarial budgets. In the l_2 and l_∞ cases, adversarial training against multi-step PGD does not suffer from catastrophic overfitting, while we see catastrophic overfitting occurring in the l_1 case even when using 10 steps to generate adversarial examples. Specifically, catastrophic overfitting occurs more frequently with small values of K. Furthermore, when catastrophic overfitting does not happen, the robust accuracy is very sensitive to the choice of K and α, which makes it difficult for practitioners to find the optimal hyper-parameters. To conclude, it is more challenging to achieve robustness in the l_1 case.

$\alpha \backslash K$	20	30	50	100	150	200
1.5	30.95	40.47	51.64	46.94	44.57	39.33
3.0	28.60	34.54	40.54	52.77	47.11	43.00
4.5	32.87	35.94	40.21	**53.47**	48.47	43.51
6.0	27.70	30.62	35.98	44.37	48.76	45.73

Table 1. Robust accuracy (in %) of the *best* checkpoint obtained with 10-step SLIDE under different settings of α and K. A gray background indicates that catastrophic overfitting occurs in the corresponding setting, with the robust accuracy of the *final* checkpoint being close to zero.

3.2. Existing Efficient Adversarial Training Methods

To the best of our knowledge, no existing works specifically address efficient adversarial training against l_1 adversarial attacks. In this section, we evaluate a few existing efficient adversarial training methods that were originally designed or evaluated against l_2 or l_∞ attacks. Our experiments demonstrate that directly applying them to the l_1 case does not yield satisfactory performance.

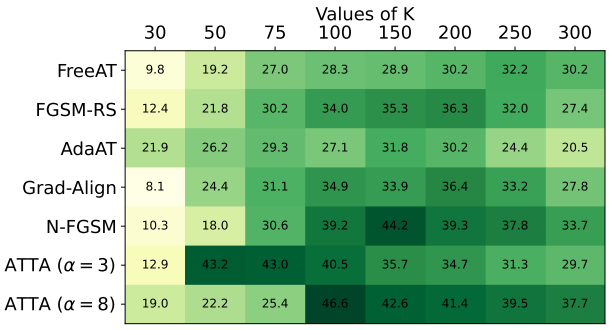

Figure 2. Robust accuracy (in %) of the *best* checkpoint of existing fast adversarial training methods with different K values. Values of α are already finetuned.

Specifically, we evaluate free adversarial training (FreeAT) (Shafahi et al., 2019), fast adversarial training (FGSM-RS) (Wong et al., 2020), adaptive single-step adversarial training (AdaAT) (Kim et al., 2021), gradient align (Grad-Align) (Andriushchenko & Flammarion, 2020), the noisy fast gradient sign method (N-FGSM) (de Jorge

et al., 2022) and adversarial training with transferable attacks (ATTA) (Zheng et al., 2020) with different values of K. We optimize other hyper-parameters of these methods and defer the setting details to the Appendix C.1. The results are summarized in Figure 2 and clearly show that all these methods suffer from catastrophic overfitting with a small K. When using a bigger K, catastrophic overfitting disappears and the performance improves. However, the robust accuracy in this case remains unsatisfactory and lower than that of SLIDE in Table 1.

For all methods in Figure 2, adversarial examples were generated by one-step attacks for updating parameters. ATTA strengthens the adversarial attacks used for training without introducing any computational overhead. It stores the adversarial perturbations for each training instance and uses them as perturbation initialization in the next epoch. Despite better stability and no computational overhead compared with FGSM-RS, ATTA has a considerable memory overhead, since it needs to store the initial perturbations for all training instances. This memory overhead prevents ATTA from scaling to large datasets, such as ImageNet (Deng et al., 2009). In Figure 2, we provide the results of ATTA using a small step size $\alpha = 3$ and a large step size $\alpha = 8$. We provide more analyses and results in Appendix C.2.

3.3. Analysis of Catastrophic Overfitting

The observations in Section 3.1 and Section 3.2 evidence the stability issues of existing training methods on l_1 adversarial budgets. That is, it is more challenging to design efficient and stable algorithms to obtain such robustness. Unlike the l_2 and l_∞ cases, existing methods use coordinate descent to train robust models in the l_1 scenario, which introduces an additional hyper-parameter K. With the other hyper-parameters, such as the step size α, fixed, the results in both Table 1 and Figure 2 indicate that catastrophic overfitting occurs more frequently when K is small.

From an optimization perspective, the number of possible perturbation updates in one iteration of K-hot coordinate descent is $N_{dir} = 2^K \binom{M}{K}$, where $\binom{M}{K}$ is the number of K-combination among M objects. Since the input dimension M is much larger than K, i.e., $M \gg K$, N_{dir} increases with K. Therefore, there are a limited number of possible update directions when K is small. This is problematic for coordinate descent: in addition to slower convergence in the general case, it may even fail to find a local minimum when optimizing a non-smooth function (Spall, 2012). We demonstrate this issue using a 2-D contour example in Figure 3. Since many neural networks use non-smooth activation functions, such as ReLU (Krizhevsky et al., 2012) and leaky ReLU (Radford et al., 2015), it would be difficult for coordinate descent to find a good adversarial example. We defer a more comprehensive analysis to Appendix B.1.

In addition to the convergence issue demonstrated in Figure 3, the coordinate descent we use to generate adversarial examples within the l_1 adversarial budget has a strong bias toward generating sparse perturbations. As a result, the model may overfit to these sparse perturbations and lose robustness against dense ones within the l_1 adversarial budget. We can demonstrate this using SLIDE in Algorithm 1. After running K-hot coordinate descent for N_{iter} iterations, we obtain the accumulated perturbation of at most KN_{iter} non-zero elements. In addition, the upper bound KN_{iter} can be reached only when the K-hot coordinates in each of the N_{iter} iterations never overlap. In practice, the gradients of the loss $\nabla_\Delta \mathcal{L}$ in different iterations are correlated, so the actual number of non-zero elements of the accumulated perturbation is smaller than the theoretical upper bound KN_{iter}. Furthermore, the following lemma indicates that the projection after each iteration makes the perturbation Δ sparser. The proof is deferred to Appendix B.2.

Lemma 3.1. *$\forall \Delta_{ori}, \epsilon$, if $\Delta_{prj} = \operatorname{argmin}_\Delta \|\Delta - \Delta_{ori}\|_2$ s.t. $\|\Delta\|_1 \leq \epsilon$, then we have $\|\Delta_{prj}\|_0 \leq \|\Delta_{ori}\|_0$.*

Although we can initialize Δ by uniform sampling from the adversarial budget to make it originally dense, such a trick turns out in practice to yield very limited improvement in terms of performance and stability. This is because we usually perturb Δ outside the adversarial budget after coordinate descent, and then the projection operator makes Δ sparse. In addition, a smaller K means a larger effective step size α/K. The following lemma indicates that, with a larger effective step size, the post-projection perturbation becomes sparser. The proof is deferred to Appendix B.3.

Lemma 3.2. *$\forall \Delta \in \mathcal{S}_\epsilon := \{\Delta | \|\Delta\|_1 \leq \epsilon\}$, $\forall \alpha_1 \geq \alpha_2 > 0$, if $\boldsymbol{v}$ satisfying $\Delta + \alpha_2 \boldsymbol{v} \notin \mathcal{S}_\epsilon$, then we have $\|\Pi_{\mathcal{S}_\epsilon}(\Delta + \alpha_1 \boldsymbol{v})\|_0 \leq \|\Pi_{\mathcal{S}_\epsilon}(\Delta + \alpha_2 \boldsymbol{v})\|_0$*

Therefore, catastrophic overfitting is more likely to occur with a small K, which is consistent with the results in Table 1 and Figure 2.

We further empirically verify the findings above. We apply AutoAttack (AA) (Croce & Hein, 2021) to the models just before and just after catastrophic overfitting when training with SLIDE. Figure 4 plots the distribution of the l_0 norm, which is the number of non-zero elements and indicates the sparsity, of the perturbations generated by AA. Figure 6 in Appendix C.2 provides a visualization of the perturbations. We can clearly see that the adversarial examples generated by AA to fool the model after catastrophic overfitting have a much larger l_0 norm than the ones before catastrophic overfitting. This observation is consistent with our analysis. The adversarial perturbations generated by SLIDE are sparse, and when the model overfits to these perturbations, it loses robustness against relatively dense ones.

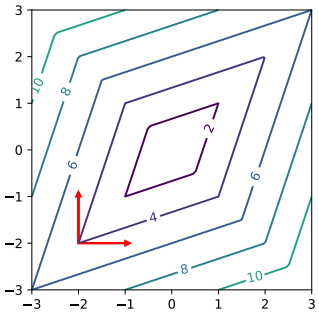

Figure 3. Example showing coordinate descent trapped in suboptimal regions with non-smooth functions. We plot the contours of the function $2 \times |x - y| + |x + y|$. At the point $(-2, -2)$, both coordinate descent directions, marked by red arrows, increase the function value.

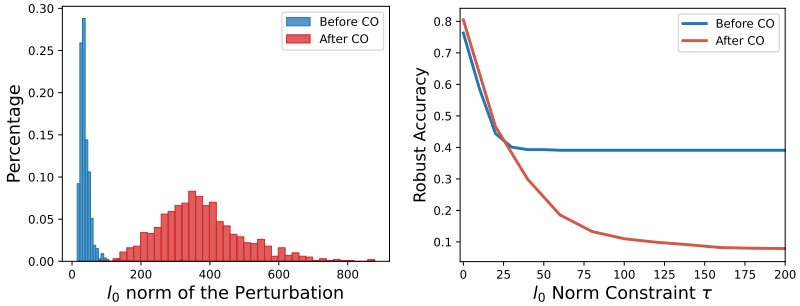

Figure 4. **Left**: Distributions of the l_0 norm of the perturbations generated by AutoAttack (AA) before and after catastrophic overfitting (CO). **Right**: Robust accuracy of the checkpoints before and after CO evaluated by varying values of l_0 norm constraint τ in the adversarial budget $\mathcal{T}_{\epsilon,\tau} := \{\Delta | \|\Delta\|_1 \leq \epsilon, \|\Delta\|_0 \leq \tau\}$ (ϵ is fixed). The model loses its robustness against less sparse attacks after CO.

To evaluate the robustness of the model against perturbations of different sparsity levels within the adversarial budget $\mathcal{S}_\epsilon^{(1)}$, we define the set of perturbations $\mathcal{T}_{\epsilon,\tau} := \{\Delta | \|\Delta\|_1 \leq \epsilon, \|\Delta\|_0 \leq \tau\}$ and calculate the robust accuracy of the model on $\mathcal{T}_{\epsilon,\tau}$ before and after catastrophic overfitting. Perturbations in $\mathcal{T}_{\epsilon,\tau}$ are not only bounded by their l_1 norm but also by their l_0 norm. Although some methods generate perturbations with small l_0 norms, they have unbounded adversarial budgets (Modas et al., 2019) or have prohibitively expensive complexity (Su et al., 2019). To tackle this issue, we run AutoPGD using the adversarial budget $\mathcal{S}_\epsilon^{(1)}$ and project the perturbation in each iteration to $\mathcal{T}_{\epsilon,\tau}$ to estimate the robust accuracy on $\mathcal{T}_{\epsilon,\tau}$. We provide more detail in Appendix A.2, including pseudo-code in Algorithm 5. With ϵ fixed, Figure 4 demonstrates the robust accuracy on $\mathcal{T}_{\epsilon,\tau}$ with varying τ. Our results indicate that the model is still robust to sparse perturbations, i.e., a small τ, even after catastrophic overfitting. For models after when catastrophic overfitting happens, the robustness becomes lower and lower when we include more and more dense attacks within the adversarial budget. This indicates that the model loses its robustness against less sparse perturbations in this phase. By contrast, the robust accuracy of the model before when catastrophic overfitting happens first decreases with the increase in τ but later becomes stable. This indicates that dense attacks do not help much when catastrophic overfitting does not happen; the most powerful perturbations for the models in this case are still sparse.

4. Efficient Adversarial Training based on Euclidean Geometry

In Section 3, we analyzed why catastrophic overfitting occurs more frequently with an l_1 adversarial budget and connected this problem to the use of coordinate descent. Therefore, we propose to stabilize adversarial training, especially efficient adversarial training, by using adversarial examples generated with another strategy.

Using an optimization method other than coordinate descent can be thought of as optimizing the inner maximization problem in a geometry other than the one based on the l_1 norm. We note that the perturbation update rule of SLIDE will degrade to the one based on l_∞ geometry when K is as large as M. However, Table 1 and Figure 2 indicate that SLIDE yields stable but sub-optimal performance with a very large K. In contrast to coordinate descent, attacks based on the l_∞ norm have a strong bias to dense perturbations, since they update Δ by the same magnitude in each dimension. Failing to generate sparse perturbations is also problematic, because Figure 4 indicates that strong perturbations are mostly sparse when catastrophic overfitting does not occur.

As a result, we choose Euclidean geometry, the geometry we use in the perturbation update rule for l_2 adversarial budgets. Note that, while we modify the descent direction used to optimize the adversarial perturbations, we still search for the optimal perturbations in the l_1 adversarial budget. Therefore, the projection operator after the perturbation update in each iteration remains the same. The pseudo-code of our method is provided in Algorithm 2, and we refer to it as **Fast-EG-l_1**. Fast-EG-l_1 uses one-step attacks based on Euclidean geometry to efficiently generate adversarial perturbations within the l_1 adversarial budget for training. Compared with SLIDE, Fast-EG-l_1 does not use coordinate descent and thus does not require the extra hyper-parameter K. Compared with ATTA, Fast-EG-l_1 randomly initializes the perturbation and thus does not yield extra memory consumption, which enables it to scale to larger datasets.

It should be pointed out that Fast-EG-l_1 is different from

Algorithm 2 Our proposed method Fast-EG-l_1

Input: Step size α, adversarial budget used for training $\mathcal{S}^{(1)}_{\epsilon_{train}}$, model parameters θ.
for $(\boldsymbol{x}, y)$ in the training set **do**
  Uniformly sample Δ from $\mathcal{S}^{(1)}_{\epsilon_{train}}$.
  Calculate the gradient $\boldsymbol{g} \leftarrow \nabla_{\Delta}\mathcal{L}(\theta, \boldsymbol{x} + \Delta)$.
  Update $\Delta \leftarrow \Pi_{\mathcal{S}^{(1)}_{\epsilon_{train}}}(\Delta + \alpha \boldsymbol{g}/\|\boldsymbol{g}\|_2)$.
  Calculate the loss $\mathcal{L}(\theta, \boldsymbol{x} + \Delta)$ and update θ.
end for
Output: Trained model parameters θ.

fast adversarial training (Wong et al., 2020) for l_2 adversarial budgets. Despite a similar update rule for perturbations, Fast-EG-l_1 projects the perturbation back to an l_1 adversarial budget. More importantly, the motivations of Fast-EG-l_1 are rather different from fast adversarial training (Wong et al., 2020). Fast adversarial training is derived from maximizing the first-order Taylor expansion of the loss objective function as demonstrated in Section 2, and we would get SLIDE under the same motivation. However, as Section 3 indicates, SLIDE is more likely to suffer from catastrophic overfitting and fails to robustify models against non-sparse attacks. Therefore, Fast-EG-l_1 is proposed to update adversarial perturbations by vectors denser than those of SLIDE to mitigate catastrophic overfitting.

4.1. Step Size Calculation

One practical challenge here relates to choosing an appropriate step size α in Algorithm 2, since we use a different geometry from the adversarial budget. When using the same geometry as the adversarial budget, we can set α proportionally to ϵ, such as $\alpha = 1.25\epsilon$ in Wong et al. (2020) and $\alpha = 0.25\epsilon$ in Madry et al. (2018). By contrast, in Fast-EG-l_1, we consider the step size α such that the l_2 ball of radius α centered on the clean input covers all allowable adversarial examples defined by the adversarial budget $\mathcal{S}^{(1)}_{\epsilon}$ and the input space $\mathcal{I}$. The following lemma formally demonstrates that $\alpha = \sqrt{\epsilon}$ is sufficiently large to satisfy this requirement under the condition that the input is bounded in $[0, 1]$. The proof is deferred to Appendix B.3.

Lemma 4.1. *Given the input $\boldsymbol{x}$, let $\mathcal{S}_{\boldsymbol{x}'}$ denote the set of allowable adversarial examples: $\{\boldsymbol{x}' | \|\boldsymbol{x}' - \boldsymbol{x}\|_1 \leq \epsilon, 0 \leq \boldsymbol{x}' \leq 1\}$, and $\mathcal{S}_{ball}$ denote the l_2 ball of radius $\sqrt{\epsilon}$ centered on $\boldsymbol{x}$: $\{\boldsymbol{x}' | \|\boldsymbol{x}' - \boldsymbol{x}\|_2 \leq \sqrt{\epsilon}\}$. Then, we have $\mathcal{S}_{\boldsymbol{x}'} \subseteq \mathcal{S}_{ball}$.*

Therefore, we use $\alpha = \sqrt{\epsilon}$ as the default step size.

4.2. Multi-ϵ Strategy

Besides adjusting the value of the step size α, we also utilize the *multi-ϵ* strategy (Croce & Hein, 2021), because the optimal adversarial perturbation can be on the low dimensional faces of the l_1 adversarial budget, i.e., the "corners" of the l_1 ball. We noticed that perturbation updates based on Euclidean geometry struggle at finding such perturbations. Therefore, we use a larger adversarial budget during training, such as $2\times$ the value of ϵ used for testing, to better explore the "corners" of the test adversarial budget. Different from Croce & Hein (2021) which uses multi-ϵ for attacks and gradually decreases the value of ϵ during the iteration, we use multi-ϵ for training and keep the size of adversarial budget fixed during training. For notation, we use ϵ_{train} to denote the size of the training adversarial budget.

5. Experiments

In this section, we demonstrate the effectiveness of our proposed method: Fast-EG-l_1. We conduct comprehensive experiments on popular benchmarks, including CIFAR10, CIFAR100 (Krizhevsky et al., 2009) and ImageNet100 (Russakovsky et al., 2015). All these datasets consist of RGB images, whose resolution is 32×32 for CIFAR10 and CIFAR100, and 224×224 for ImageNet100. The size of the l_1 adversarial budgets, i.e., ϵ, is set to 12, 6 and 72 for CIFAR10, CIFAR100, and ImageNet100, respectively. For CIFAR10 and CIFAR100, we use PreactResNet18 (He et al., 2016b) and train all the models for 40 epochs. For ImageNet100, we use ResNet34 and train all the models for 25 epochs. We evaluate the robustness of all models by AutoAttack (Croce & Hein, 2021), which is an ensemble of four different adversarial attacks and considered as the current state-of-the-art evaluation metric. We use one NVIDIA A100 GPU for the experiments on CIFAR10 and CIFAR100; and 2 NVIDIA A100 GPUs for the experiments on ImageNet100. More details are available in Appendix C.1.

5.1. Comparison with Existing Methods

We first compare our approach with the existing one-step adversarial training methods. The baselines include fast adversarial training (FGSM-RS) (Wong et al., 2020), adversarial training with transferable attacks (ATTA) (Zheng et al., 2020), adaptive one-step adversarial training (AdaAT) (Kim et al., 2021), fast adversarial training with gradient alignment (Grad-Align) (Andriushchenko & Flammarion, 2020), noise FGSM (N-FGSM) (de Jorge et al., 2022) and nuclear norm adversarial training (NuAT) (Sriramanan et al., 2021). Note that Grad-Align and NuAT are regularization methods that based on gradient alignment regularization and nuclear norm regularization, repspectively. Our algorithm, Fast-EG-l_1, uses the default values $\epsilon_{train} = 2\epsilon$ and $\alpha = \sqrt{\epsilon}$ in all settings. All baseline algorithms use coordinate descent to generate adversarial examples, so in addition to the step size α, they have an additional hyper-parameter K to tune. As shown in Section 3, the choice of hyper-parameters plays a crucial role in training for baselines. Therefore, we finetune

Method	CIFAR10 ($\epsilon = 12$)		CIFAR100 ($\epsilon = 6$)		ImageNet100 ($\epsilon = 72$)	
	AA (%)	Time (h)	AA (%)	Time (h)	AA (%)	Time (h)
AutoPGD	55.77	2.58	42.18	2.58	-	-
FGSM-RS	36.29	0.76	33.23	0.71	36.64	22.12
ATTA	46.57	0.67	33.74	0.68	-	-
AdaAT	31.84	0.88	28.64	0.84	28.62	26.96
Grad-Align	36.38	1.52	33.19	1.52	-	-
N-FGSM	44.21	0.65	35.79	0.66	30.28	23.53
NuAT	48.35	1.01	36.46	1.05	45.82	29.18
Fast-EG-l_1	**50.27**	0.67	**38.03**	0.67	**46.74**	22.11

Table 2. Robust accuracy (in %) evaluated by AutoAttack (AA) and training time in hours when we run different methods on CIFAR10, CIFAR100, and ImageNet100. Hyper-parameters of baselines are finetuned. The results of AutoPGD, ATTA and Grad-Align on ImageNet100 are not reported because of prohibitively-high computational or memory overhead.

hyper-parameters and report the highest accuracy achieved for each method. In addition, we include training against 10-step AutoPGD as a reference, since it achieves the best robustness against l_1 perturbations (Croce & Hein, 2021).

Our results are summarized in Table 2, where we report the robust accuracy evaluated by AutoAttack (AA) and the total training time. It is worth noting that training with AutoPGD is computationally expensive, as it involves 11 forward and backward passes per mini-batch update, while our proposed Fast-EG-l_1 in Algorithm 2 requires only 2 such passes. Moreover, the memory overhead of ATTA and Grad-Align on ImageNet100 is too large for 2 NVIDIA A100 GPUs. Therefore, we do not report the results of AutoPGD, ATTA and Grad-Align on ImageNet100 due to their prohibitively-high computational or memory overhead.

Robust accuracy. As observed in Table 2, our proposed algorithm Fast-EG-l_1 achieves the highest performance among the efficient adversarial training methods on all datasets. Moreover, as demonstrated in Table 1 and Figure 2 in Section 3, the performance of the baselines is sensitive to the choice of α and K, so it is not easy to quickly identify the optimal hyper-parameters. Furthermore, the optimal hyper-parameters of one method vary significantly for different datasets, especially in the presence of different input dimensions. This complicates the hyper-parameter selection process for these methods. By contrast, our proposed Fast-EG-l_1 is free of coordinate descent and does not have the hyper-parameter K. Based on Lemma 4.1, we simply set $\alpha = \sqrt{\epsilon}$ for all datasets and achieve better performance than the well-tuned baselines.

Computational and memory costs. Fast-EG-l_1 is among the fastest methods in Table 2, since it does not require calculating any regularization terms like Grad-Align and NuAT. Unlike ATTA, which stores the perturbation of each training input, and Grad-Align, which relies on second-order gradients, Fast-EG-l_1 has negligible memory overhead and is thus applicable to large datasets, such as ImageNet100, on a moderate number of GPUs.

Clean accuracy. We report the results of clean accuracy of each method in Table 6 at Appendix C.2. Fast-EG-l_1 achieves comparable clean accuracy with efficient adversarial training baselines of competitive robust accuracy (ATTA and NuAT). Compared to AutoPGD, Fast-EG-l_1 pays around 3% cost on clean accuracy but is 4 times faster. Moreover, as noted in Table 4, one could also tune the value of ϵ_{train} in Fast-EG-l_1 for better trades-off between clean and robust accuracy.

Regularization. Fast-EG-l_1 is orthogonal to any regularization schemes, such as Grad-Align and NuAT in Table 2. Therefore, one can, for example, incorporate the nuclear regularization term of NuAT into Fast-EG-l_1 to further improve the performance. In Table 3 we demonstrate the results of incorporating nuclear norm regularization into Fast-EG-l_1, where λ denotes the weight factor of nuclear norm regularization and other settings are kept the same as in Table 2. The results in Table 3 indicate that a small regularization weight is helpful to further improve the robustness of our method; while a large weight might hurt the performance. With a proper choice of the weight factor, our method yields 51.37%, 39.75%, 48.82% robust accuracy on CIFAR10, CIFAR100 and ImageNet100, respectively. Nevertheless, it's also worth noting that in this case, the method needs 70% additional training time on top of Fast-EG-l_1, which makes the total training time similar to that of NuAT.

Dataset \ λ	0	0.1	0.5	1	3
CIFAR10	50.27	51.08	51.23	**51.37**	49.43
CIFAR100	38.03	**39.75**	38.69	38.06	31.29
ImageNet100	46.74	**48.82**	48.70	-	-

Table 3. Robust accuracy (in %) when incorporating nuclear norm regularization into our proposed Fast-EG-l_1. λ denotes the weight factor of the regularization term, so $\lambda = 0$ represents the results of Fast-EG-l_1 without regularization. Other hyper-parameters are kept the same as Table 2.

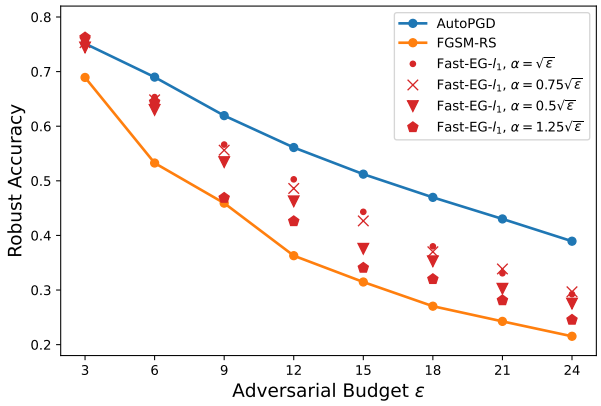

Figure 5. Robust accuracy evaluated by AutoAttack when using different values of α and ϵ on CIFAR10. We plot the results of AutoPGD and FGSM-RS for reference.

5.2. Ablation Studies

We focus on the CIFAR10 dataset for our ablation studies.

Different values of ϵ. We varied the value of ϵ from 3 to 24 while keeping $\alpha = \sqrt{\epsilon}$. Note that, ϵ is the size of the targeted adversarial budget, which is also used for testing and thus different from ϵ_{train} in Section 4.2 The results on CIFAR10 are provided in Figure 5, showing a smooth decrease in robust accuracy with an increasing ϵ. This indicates that our method and proposed hyper-parameter settings can be applied to a larger ϵ without leading to catastrophic overfitting. Results of varing ϵ on ImageNet100 are available at Table 8 in Appendix C.2.

Different step sizes α. In addition to the default step size $\alpha = \sqrt{\epsilon}$, we also run Fast-EG-l_1 using different step sizes α. Specifically, we set α to be $0.5\sqrt{\epsilon}$, $0.75\sqrt{\epsilon}$ and $1.25\sqrt{\epsilon}$. For each value of α, we also vary the value of ϵ from 3 to 24 and plot the robust accuracy in Figure 5. We also plot the robust accuracy when training against FGSM-RS and 10-step AutoPGD (Croce & Hein, 2021) for reference. The results evidence that the performance for different step sizes α is comparable. That is, the performance of Fast-EG-l_1 is not very sensitive to the choice of step size as long as the step size is close to $\sqrt{\epsilon}$, which is the theoretical result suggested by Lemma 3.2. For all choices of step size, Fast-EG-l_1 yields better performance than FGSM-RS, which needs careful hyper-parameter tuning to avoid catastrophic overfitting. It also narrows down the performance gap between the efficient one-step methods and computationally expensive multi-step methods such as 10-step AutoPGD.

With and without multi-ϵ Strategy. Finally, we compare the performance of Fast-EG-l_1 with and without the multi-ϵ trick. Specifically, we keep the size of targeted adversarial budget $\epsilon = 12$ and set ϵ_{train} to be ϵ, 1.5ϵ, 2ϵ, 2.5ϵ and 3ϵ. In addition, the step size α is always $\sqrt{\epsilon}$ based on Lemma 3.2 regardless of ϵ_{train}. The clean and robust accuracy of Fast-EG-l_1 under these settings are provided in Table 4. We can see that ϵ_{train} controls the trade-off between clean and robust accuracy. All the settings where $\epsilon_{train} > \epsilon$ have comparable and competitive performance on robust accuracy, while a large value of ϵ_{train} might hurt the performance on clean accuracy. Nevetheless, using the same ratio $\epsilon_{train}/\epsilon$ works well across different datasets. That is to say, although $\epsilon_{train} = 2\epsilon$ works the best on CIFAR10, we do not fine-tune ϵ_{train} and simply use $\epsilon_{train} = 2\epsilon$ for all other datasets in Table 2. The competitive performance of Fast-EG-l_1 in Table 2 on various datasets indicates that practitioners do not need to worry about the selection of hyper-parameters such as ϵ_{train} when using Fast-EG-l_1.

ϵ_{train}	ϵ	1.5ϵ	2ϵ	2.5ϵ	3ϵ
Clean (%)	69.70	78.35	76.14	72.77	70.05
Robust (%)	38.15	48.85	**50.27**	49.63	48.10

Table 4. Clean and robust accuracy (in %) of Fast-EG-l_1 with different sizes of the training adversarial budget ϵ_{train} on CIFAR10. The value of ϵ, which is the size of the adversarial budget for evaluation, is 12.

It is worth noting that the multi-ϵ strategy is also applicable to baselines in Table 2. In this regard, we report their results with multi-ϵ strategy in Table 7 of Appendix C.2. Still, our proposed Fast-EG-l_1 obtains the better robust accuracy than these baselines.

6. Conclusion

In this work, we have studied the design of stable and efficient adversarial training algorithms against input perturbations bounded by l_1 norms. We have demonstrated that achieving robustness against l_1-bounded perturbations is more difficult than against l_2- and l_∞-bounded ones because of more frequent catastrophic overfitting. In addition, we have shown that the cause of such more frequent catastrophic overfitting could be traced to the use of coordinate descent, as commonly done in existing methods. Based on this finding, we have designed **Fast-EG-l_1**, which uses gradient descent based on Euclidean geometry and thus without coordinate descent. Compared with existing methods on l_1 adversarial budgets, Fast-EG-l_1 has fewer hyper-parameters and does not yield additional memory consumption. Our extensive experiments on several datasets have shown that Fast-EG-l_1 yields better robust accuracy than existing efficient adversarial training methods. In future work, we will investigate how to further narrow down the gap between the one-step and multi-step adversarial training methods in the case of l_1-bounded perturbations.

References

Alayrac, J.-B., Uesato, J., Huang, P.-S., Fawzi, A., Stanforth, R., and Kohli, P. Are labels required for improving adversarial robustness? In Advances in Neural Information Processing Systems, pp. 12192–12202, 2019.

Andriushchenko, M. and Flammarion, N. Understanding and improving fast adversarial training. Advances in Neural Information Processing Systems, 33:16048–16059, 2020.

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, pp. 274–283. PMLR, 2018.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. S. Unlabeled data improves adversarial robustness. In Advances in Neural Information Processing Systems, pp. 11190–11201, 2019.

Chen, J., Cheng, Y., Gan, Z., Gu, Q., and Liu, J. Efficient robust training via backward smoothing. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pp. 6222–6230, 2022.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In International Conference on Machine Learning, pp. 2196–2205. PMLR, 2020a.

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.

Croce, F. and Hein, M. Mind the box: l_1-APGD for sparse adversarial attacks on image classifiers. In International Conference on Machine Learning, pp. 2201–2211. PMLR, 2021.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. RobustBench: a standardized adversarial robustness benchmark. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2021. URL https://openreview.net/forum?id=SSKZPJCt7B.

de Jorge, P., Bibi, A., Volpi, R., Sanyal, A., Torr, P., Rogez, G., and Dokania, P. K. Make some noise: Reliable and efficient single-step adversarial training. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=NENo__bExYu.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In International Conference on Learning Representations, 2014.

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.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In European conference on computer vision, pp. 630–645. Springer, 2016b.

Kang, P. and Moosavi-Dezfooli, S.-M. Understanding catastrophic overfitting in adversarial training. arXiv preprint arXiv:2105.02942, 2021.

Kim, H., Lee, W., and Lee, J. Understanding catastrophic overfitting in single-step adversarial training. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pp. 8119–8127, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K. (eds.), Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012.

Liu, C., Salzmann, M., Lin, T., Tomioka, R., and Süsstrunk, S. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. Advances in Neural Information Processing Systems, 33, 2020.

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. URL https://openreview.net/forum?id=rJzIBfZAb.

Maini, P., Wong, E., and Kolter, Z. Adversarial robustness against the union of multiple perturbation models. In International Conference on Machine Learning, pp. 6640–6650. PMLR, 2020.

Modas, A., Moosavi-Dezfooli, S.-M., and Frossard, P. SparseFool: a few pixels make a big difference. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9087–9096, 2019.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1765–1773, 2017.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In International Conference on Learning Representations, 2015.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. Data augmentation can improve robustness. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), Advances in Neural Information Processing Systems, 2021. URL https://openreview.net/forum?id=kgVJBBThdSZ.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. ImageNet large scale visual recognition challenge. International journal of computer vision, 115(3): 211–252, 2015.

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! Advances in Neural Information Processing Systems, 32, 2019.

Spall, J. C. Cyclic seesaw process for optimization and identification. Journal of optimization theory and applications, 154(1):187–208, 2012.

Sriramanan, G., Addepalli, S., Baburaj, A., et al. Towards efficient and effective adversarial training. Advances in Neural Information Processing Systems, 34:11821–11833, 2021.

Su, J., Vergas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation, 23(5):828–841, 2019.

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. URL http://arxiv.org/abs/1312.6199.

Tramer, F. and Boneh, D. Adversarial training and robustness for multiple perturbations. Advances in Neural Information Processing Systems, 32, 2019.

Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. Towards fast computation of certified robustness for relu networks. In International Conference on Machine Learning, pp. 5276–5285. PMLR, 2018.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. Advances in Neural Information Processing Systems, 33, 2020.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Accelerating adversarial training via maximal principle. In Advances in Neural Information Processing Systems, pp. 227–238, 2019a.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In International Conference on Machine Learning, pp. 7472–7482, 2019b.

Zheng, H., Zhang, Z., Gu, J., Lee, H., and Prakash, A. Efficient adversarial training with transferable adversarial examples. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1181–1190, 2020.

A. Algorithm Details

A.1. Training on Different Adversarial Budgets

Algorithm 3 and Algorithm 4 demonstrate the pseudo-code of adversarial training against PGD in the l_2 and l_∞ cases, respectively. They are first introduced in Madry et al. (2018), Wong et al. (2020) proposed the accelerated version (FGSM-RS) by setting $N_{iter} = 1$. Here, we use uniform sampling to randomly initialize the perturbations, because we empirically find uniform sampling helps to generate more diverse perturbations and thus yields better results.

Algorithm 3 PGD adversarial training in l_2 cases.

Input: step size α, iteration number N_{iter}, adversarial budget $\mathcal{S}_\epsilon^{(2)}$, model parameters θ.
for $(\boldsymbol{x}, y)$ in the training set **do**
  Uniformly sample Δ from $\mathcal{S}_\epsilon^{(2)}$
  for iteration in $1, 2, ..., N_{iter}$ **do**
    Calculate the gradient $\boldsymbol{g} \leftarrow \nabla_\Delta \mathcal{L}(\theta, \boldsymbol{x} + \Delta)$.
    Update $\Delta \leftarrow \Pi_{\mathcal{S}_\epsilon^{(2)}} (\Delta + \alpha \boldsymbol{g} / \|\boldsymbol{g}\|_2)$
  end for
  Calculate loss $\mathcal{L}(\theta, \boldsymbol{x} + \Delta)$ and update θ.
end for
Output: Updated model parameters θ.

Algorithm 4 PGD adversarial training in l_∞ cases.

Input: step size α, iteration number N_{iter}, adversarial budget $\mathcal{S}_\epsilon^{(\infty)}$, model parameters θ.
for $(\boldsymbol{x}, y)$ in the training set **do**
  Uniformly sample Δ from $\mathcal{S}_\epsilon^{(\infty)}$
  for iteration in $1, 2, ..., N_{iter}$ **do**
    Calculate the gradient $\boldsymbol{g} \leftarrow \nabla_\Delta \mathcal{L}(\theta, \boldsymbol{x} + \Delta)$.
    Update $\Delta \leftarrow \Pi_{\mathcal{S}_\epsilon^{(\infty)}} (\Delta + \alpha \cdot \text{sign}(\boldsymbol{g}))$
  end for
  Calculate loss $\mathcal{L}(\theta, \boldsymbol{x} + \Delta)$ and update θ.
end for
Output: Updated model parameters θ.

A.2. Evaluating Robustness Against Sparse Perturbations Within the l_1 Adversarial Budgets

In Section 3.3, we generate sparse adversarial perturbations within the l_1 adversarial budget. In this regard, the adversarial budget becomes $\mathcal{T}_{\epsilon,\tau} = \{\Delta | \|\Delta\|_1 \leq \epsilon, \|\Delta\|_0 \leq \tau\}$. Since efficiently generating l_0 bounded adversarial perturbations is challenging, we instead run the AutoPGD (Croce & Hein, 2021) algorithm to generate perturbations within the l_1 ball and project the perturbation in each iteration to $\mathcal{T}_{\epsilon,\tau}$ to estimate the robust accuracy on $\mathcal{T}_{\epsilon,\tau}$. The algorithm can be represented by the pseudo-code in Algorithm 5.

Algorithm 5 Evaluation against the Adversarial Budget $\mathcal{T}_{\epsilon,\tau}$.

Input: iteration number N_{iter}, adversarial budget $\mathcal{T}_{\epsilon,\tau}$, model parameters θ.
for $(\boldsymbol{x}, y)$ in the training set **do**
  Uniformly sample Δ from $\mathcal{S}_\epsilon^{(1)}$
  for itersation in $1, 2, ..., N_{iter}$ **do**
    Calculate the gradient $\boldsymbol{g} \leftarrow \nabla_\Delta \mathcal{L}(\theta, \boldsymbol{x} + \Delta)$.
    Update Δ based on the step size α and K calculated by AutoPGD.
    Projection $\Delta' \leftarrow \Pi_{\mathcal{T}_{\epsilon,\tau}} \Delta$
    if $\boldsymbol{x} + \Delta'$ has the incorrect output **then**
      Output: $\boldsymbol{x}$ can be successfully attacked.
      Break;
    end if
  end for
  Output: $\boldsymbol{x}$ cannot be successfully attacked.
end for

B. Theoretical Discussions & Proofs

B.1. Suboptimality of SLIDE for Non-smooth Functions

In Section 3.3 we've discussed the suboptimality of SLIDE due to the limited number of possible update directions of K-hot coordinate descent. Here we make a more general claim: there broadly exists non-smooth functions where coordinate descent can fail, depending on the initial points of the optimization. In the context of SLIDE with the number of updated coordinates K, we consider a loss objective function $f : \mathbb{R}^M \to \mathbb{R}$ and use $\mathcal{T} = \left\{\boldsymbol{v} \in \{0, 1\}^M | \|\boldsymbol{v}\|_0 \leq K\right\}$ to represent all possible "unit" coordinate updates in SLIDE. In addition, we use $\mathcal{M}(\boldsymbol{x}) = \left\{\boldsymbol{v} \in \mathbb{R}^M | \exists a > 0, s.t. f(\boldsymbol{x} + a\boldsymbol{v}) < f(\boldsymbol{x})\right\}$ to represent the directions from the input $\boldsymbol{x}$ which can decrease the function's value.

When f is a **non-smooth** function, then $\exists \boldsymbol{x} \in \mathbb{R}^M$ such that $\boldsymbol{x}$ is not a local minima of f and $\mathcal{M}(\boldsymbol{x}) \cap \mathcal{T} = \varnothing$. One example is demonstrated in Figure 3. In this case, the coordinate descent algorithm in SLIDE gets stuck at the sub-optimal point $\boldsymbol{x}$.

Furthermore, the size of the set $\mathcal{T}$: $|\mathcal{T}|$ decreases with the decrease of K, which indicates $\mathcal{M}(\boldsymbol{x}) \cap \mathcal{T} = \varnothing$ is more likely to happen when $K \ll M$. This explains why coordinate descent in SLIDE tends to perform poorly with small K in Table 1.

When f is a **smooth** function, getting stuck at the sub-optimal point will not happen. We can prove this by contradiction. We assume that $\boldsymbol{x}$ is not a local minimum and $\mathcal{M}(\boldsymbol{x}) \cap \mathcal{T} = \varnothing$. Let $\boldsymbol{v}$ be any one-hot unit vector, it is clear that $\boldsymbol{v} \in \mathcal{T}$ and $-\boldsymbol{v} \in \mathcal{T}$ since $K \geq 1$. If $\mathcal{M}(\boldsymbol{x}) \cap \mathcal{T} = \varnothing$, then we have $\boldsymbol{v} \notin \mathcal{M}(\boldsymbol{x})$ and $-\boldsymbol{v} \notin \mathcal{M}(\boldsymbol{x})$. Based on the definition of $\mathcal{M}$ and the smoothness of the function f, we can get $\langle \nabla_{\boldsymbol{x}} f(\boldsymbol{x}), \boldsymbol{v} \rangle = 0$. Furthermore, since $\boldsymbol{v}$ can be any one-hot unit vector, we can conclude $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$. That is to say, $\boldsymbol{x}$ is a local minimum in every axis, so it is a local minimum of the function f. This contradicts the original assumption, so the coordinate descent will converge to a local minimum for a smooth function if its step size is properly chosen.

In the proof for the smooth function, we utilize the gradient of the input $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. However, the gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ might not exist for non-smooth functions. As indicated in the example of Figure 3, coordinate descent might get stuck in a non-differentiable point of a non-smooth function.

As explained above, coordinate descent may get stuck at sub-optimal points for neural networks with non-smooth activations like ReLU. In general and empirically, coordinate descent converges slowly since we do not use line search to optimize the step size. This problem will become worse in the context of efficient adversarial training, where there is only a limited budget for computation.

B.2. Proof of Lemma 3.1

We prove this lemma by contradiction. We have $\Delta_{prj} = \operatorname{argmin}_{\Delta} \|\Delta - \Delta_{ori}\|_2$ s.t. $\|\Delta\|_1 \leq \epsilon$ and we assume $\|\Delta_{prj}\|_0 > \|\Delta_{ori}\|_0$, then we can construct Δ'_{prj} based on Δ_{prj} by:

$$\left[\Delta'_{prj}\right]_i = \begin{cases} 0, & \text{if } [\Delta_{ori}]_i = 0; \\ [\Delta_{prj}]_i, & \text{otherwise.} \end{cases} \tag{3}$$

Since $\|\Delta_{prj}\|_0 > \|\Delta_{ori}\|_0$, we have $\exists i$ such that $[\Delta_{ori}]_i = 0$ and $[\Delta_{prj}]_i \neq 0$. As a result, we have $\|\Delta'_{prj} - \Delta_{ori}\|_2 < \|\Delta_{prj} - \Delta_{ori}\|_2$ and $\|\Delta'_{prj}\|_1 \leq \epsilon$. The existence of Δ'_{prj} contradicts with the optimality of Δ_{prj}. Therefore, the assumption $\|\Delta_{prj}\|_0 > \|\Delta_{ori}\|_0$ does not hold, which means $\|\Delta_{prj}\|_0 \leq \|\Delta_{ori}\|_0$.

B.3. Proof of Lemma 3.2

Since $\Delta \in \mathcal{S}_\epsilon$ and $\Delta + \alpha_1 \boldsymbol{v} \notin \mathcal{S}_\epsilon$, we have $\exists 0 \leq \alpha_0 < \alpha_1$ such that $\Delta_0 := \Delta + \alpha_0 \boldsymbol{v}$ is on the surface of $\mathcal{S}_\epsilon$, which means $\|\Delta_0\|_1 = \epsilon$. For notation simplicity, we define $\Delta_1 := \Delta + \alpha_1 \boldsymbol{v}$ and $\Delta_2 := \Delta + \alpha_2 \boldsymbol{v}$. Based on the symmetry of the set $\mathcal{S}_\epsilon$, we assume $\forall i$, $[\Delta_0]_i \geq 0$ without the loss of generality.

Consider the projection operator $\Pi_{\mathcal{S}_\epsilon}$, $\forall \boldsymbol{w} \in \mathbb{R}^M$, $\boldsymbol{w}' := \Pi_{\mathcal{S}_\epsilon} \boldsymbol{w}$ is defined as the following:

$$\boldsymbol{w}'_i = sign(\boldsymbol{w}) max\left(|\boldsymbol{w}_i| - \Lambda(\boldsymbol{w}), 0\right) = \begin{cases} \boldsymbol{w}_i - \Lambda(\boldsymbol{w}), & \text{if } \boldsymbol{w}_i > \Lambda(\boldsymbol{w}) \\ 0, & \text{if } \boldsymbol{w}_i \in [-\Lambda(\boldsymbol{w}), \Lambda(\boldsymbol{w})] \\ \boldsymbol{w}_i + \Lambda(\boldsymbol{w}), & \text{if } \boldsymbol{w}_i < -\Lambda(\boldsymbol{w}) \end{cases} \tag{4}$$

where $\Lambda(\boldsymbol{w})$ is set such that $\sum_{i=1}^{M} max\left(|\boldsymbol{w}_i| - \Lambda(\boldsymbol{w}), 0\right) = \epsilon$. Based on Equation 4, we have that $\|\Pi_{\mathcal{S}_\epsilon} \boldsymbol{w}\|_0$ is M minus the size of the set $\{\boldsymbol{w}_i | - \Lambda(\boldsymbol{w}) \leq \boldsymbol{w}_i \leq \Lambda(\boldsymbol{w})\}$.

Δ_0 is on the surface of the set $\mathcal{S}_\epsilon$, so $\|\Delta_0\|_1 = \epsilon$ and $\Lambda(\Delta_0) = 0$. In addition, we have $\|\Delta_1\|_1 = \epsilon + (\alpha_1 - \alpha_0) \sum_{i=1}^{M} \boldsymbol{v}_i$ and $\|\Delta_2\|_1 = \epsilon + (\alpha_2 - \alpha_0) \sum_{i=1}^{M} \boldsymbol{v}_i$. Because both Δ_1 and Δ_2 are outside $\mathcal{S}_\epsilon$, we have $\sum_{i=1}^{M} \boldsymbol{v}_i > 0$.

Now, we define the function $f(\alpha) := \Delta_0 + \alpha \boldsymbol{v}$. Based on the definition of Λ and $\sum_{i=1}^{M}$, we have $\Lambda(f(\alpha)) \geq \alpha \cdot \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{v}_i$. In addition, the subgradient $\frac{\partial \Lambda(f(\alpha))}{\partial \alpha} \geq \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{v}_i$, and the equality can be achieved only when $\forall i$, $\boldsymbol{v}_i = 1$. That is to say, with α increasing, $\Lambda(f(\alpha))$ increases at a faster or equal rate than the average absolute value of the elements of $f(\alpha)$ for any value of α. Based on Equation (4) and that $M - \|\Pi_{\mathcal{S}_\epsilon} f(\alpha)\|_0$ is the size of the set $\{[f(\alpha)]_i \,||\, [f(\alpha)]_i| \leq \Lambda(f(\alpha))\}$, we have that $\|\Pi_{\mathcal{S}_\epsilon} f(\alpha)\|_0$ decreases or remains the same when α increases. Since $\alpha_1 \geq \alpha_2$, we have $\|\Pi_{\mathcal{S}_\epsilon} f(\alpha_1)\|_0 \leq \|\Pi_{\mathcal{S}_\epsilon} f(\alpha_2)\|_0$, the proof concludes.

B.4. Proof of Lemma 4.1

To prove Lemma 4.1, we only need to prove that $\forall \boldsymbol{x}' \in \mathcal{S}_{\boldsymbol{x}'}$, then $\|\boldsymbol{x}' - \boldsymbol{x}\|_2 \leq \sqrt{\epsilon}$. In this regard, we bound the value of $\|\boldsymbol{x}' - \boldsymbol{x}\|_2^2 = \sum_{i=1}^{M} |\boldsymbol{x}'_i - \boldsymbol{x}_i|^2$. We let $j = \text{argmax}_i |\boldsymbol{x}'_i - \boldsymbol{x}_i|$ and then we have the following:

$$\begin{aligned} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2 &= |\boldsymbol{x}'_1 - \boldsymbol{x}_1|^2 + |\boldsymbol{x}'_2 - \boldsymbol{x}_2|^2 + ... + |\boldsymbol{x}'_M - \boldsymbol{x}_M|^2 \\ &\leq |\boldsymbol{x}'_i - \boldsymbol{x}_i| \left(|\boldsymbol{x}'_1 - \boldsymbol{x}_1| + |\boldsymbol{x}'_2 - \boldsymbol{x}_2| + ... + |\boldsymbol{x}'_M - \boldsymbol{x}_M|\right) \\ &\leq |\boldsymbol{x}'_i - \boldsymbol{x}_i| \|\boldsymbol{x}' - \boldsymbol{x}\|_1 \\ &\leq 1 \times \epsilon = \epsilon \end{aligned} \tag{5}$$

The last inequality is based on the fact that $0 \leq \boldsymbol{x}' \leq 1$ and $0 \leq \boldsymbol{x} \leq 1$. Inequality (5) indicates $\|\boldsymbol{x}' - \boldsymbol{x}\|_2 \leq \sqrt{\epsilon}, \forall \boldsymbol{x}' \in \mathcal{S}_{\boldsymbol{x}'}$, then $\boldsymbol{x}' \in \mathcal{S}_{ball}$. Therefore, $\mathcal{S}_{\boldsymbol{x}'} \subseteq \mathcal{S}_{ball}$.

In addition, we point out that the upper bound of Inequality (5) can be achieved. For example, when ϵ is an integer, we let $\boldsymbol{x}'_1 = \boldsymbol{x}'_2 = ... = \boldsymbol{x}'_\epsilon = 1$, $\boldsymbol{x}'_{\epsilon+1} = ... = \boldsymbol{x}'_M = 0$ and $\boldsymbol{x}_1 = \boldsymbol{x}_2 = ... = \boldsymbol{x}_M = 0$.

C. Experimental Details

C.1. Experimental Settings

General settings. We use PreactResNet18 with softplus activation for experiments in CIFAR10 and CIFAR100, and use ResNet34 for experiments in ImageNet100. All the models are trained with an SGD optimizer, with the momentum factor being 0.9 and the weight decay factor being 5×10^{-4}. As noted in Wong et al. (2020), weight decay regularization is not performed on parameters of Batch-Normalization layers. The training batch size is 128 for CIFAR10 and CIFAR100, and 256 for ImageNet100. For CIFAR10 and CIFAR100, we train all models for 40 epochs. The learning rate is initialized to 0.05 and is divided by a factor of 10 at the 30th epoch and the 35th epoch. For ImageNet100, the models are trained for 25 epochs. The learning rate is initialized to 0.05 and is divided by a factor of 10 at the 15th epoch and 20th epoch. We split 4% of the samples in the training set for validation. During the validation phase, we randomly select 1000 instances in the validation set and evaluate the robust accuracy by 20-step AutoPGD to select the best checkpoint during training. We found that early stopping based on the validation set is important for the baseline methods that suffer from catastrophic overfitting. Our proposed Fast-EG-l_1, with $\alpha = \sqrt{\epsilon}$ and $\epsilon_{train} = 2\epsilon$, does not suffer from catastrophic overfitting and thus usually achieves the best checkpoint at the end of training. In this case, the validation phase is not necessary. The time spent in the validation phase is not counted in Table 2. The final robust accuracy of each model is evaluated by l_1-AutoAttack on the whole test set.

Hyper-parameters for baselines in Table 2. The performance of baselines is sensitive to the choice of hyper-parameters. The hyper-parameters include the step size α and the number of updated coordinates K. We thus conduct experiments on a range of settings and report the best performance among all the choices. For CIFAR10, we conduct experiments of each method for $K \in \{30, 50, 75, 100, 150, 200, 250, 300\}$ and $\alpha \in \{3, 6, 8, 10\}$. For CIFAR100, we conduct experiments of each method for $K \in \{50, 100, 150, 200\}$ and $\alpha \in \{2, 3, 4.5, 6\}$. As a result, in CIFAR10, we find the optimal choice of K is 200 for FGSM-RS, Grad-Align and NuAT, 150 for AdaAT and N-FGSM, and 100 for ATTA. The optimal choice of step size α is 8 for ATTA. In CIFAR100, the optimal choice of K is 100 for FGSM-RS, ATTA, AdaAT, Grad-Align, N-FGSM, and 200 for NuAT. The optimal choice of step size is 6 for ATTA. For ImageNet100, we set $K = 1800$ for all the baselines.

C.2. More Experimental Results

Grid-search of hyper-parameters in ATTA. We conduct grid search on hyper-parameters α and K for ATTA. The results of ATTA for CIFAR10 and CIFAR100 datasets are demonstrated in Table 5. Similar to the grid search results shown in Table 1, the robust accuracy of ATTA against l_1 bounded perturbations is very sensitive to the choice of hyper-parameters. For each step size α, a small of K will lead to catastrophic overfitting, while a large value of K results in suboptimal performance. The best choice of hyper-parameter K for each step size is just the smallest value that does not result in catastrophic overfitting.

Clean accuracy. In Table 6 we report the clean accuracy corresponding to each method in Table 2. We found FGSM-RS, AdaAT, Grad-Align and N-FGSM have a good performance on clean accuracy, due to their less effective attack strategy in training (hence these methods fail to achieve comparable robustness). In contrast, Fast-EG-l_1 achieves comparable or better clean accuracy with efficient adversarial training baselines of competitive robust accuracy (ATTA and NuAT). Compared to

$\alpha\backslash K$	30	50	75	100	150	200	250	300
3	12.90	43.19	42.95	40.47	35.67	34.73	31.33	29.74
6	21.74	22.35	45.26	42.38	40.46	38.36	36.2	25.86
8	19.00	22.25	25.40	**46.57**	42.60	41.42	39.47	37.72
10	16.07	20.95	24.32	28.62	45.03	43.30	41.77	39.89

$\alpha\backslash K$	50	100	150	200
2	32.09	29.11	26.71	24.38
3	11.34	30.81	28.65	37.06
4.5	12.75	33.02	30.73	27.06
6	15.14	**33.74**	31.97	29.64

Table 5. Robust accuracy (in %) of the *best* checkpoint of ATTA under different settings of α and K for CIFAR10 (left table) and CIFAR100 (right table). A gray background indicates that catastrophic overfitting occurs in this setting, with the robust accuracy of the *final* checkpoint close to zero.

AutoPGD, Fast-EG-l_1 pays around 3% cost on clean accuracy but is 4 times faster. Moreover, as noted in Table 4, one could also tune the value of ϵ_{train} in Fast-EG-l_1, for better trade-off between clean and robust accuracy.

Dataset	APGD	FGSM-RS	ATTA	AdaAT	Grad-Align	N-FGSM	NuAT	**Fast-EG-l_1**
CIFAR10	80.36	84.58	76.66	87.64	85.07	83.21	77.96	76.14
CIFAR100	63.34	65.91	59.14	68.00	65.25	64.54	51.15	59.43
ImageNet100	-	71.94	-	73.78	-	68.72	57.64	67.60

Table 6. Clean accuracy (in %) corresponding to each method in Table 2 on CIFAR10, CIFAR100 and ImageNet100.

Multi-ϵ strategy on other baselines. As shown in Section 5.2, multi-ϵ is an effective trick for efficient adversarial training against l_1 bounded attack. To further test its effectiveness, we applied this trick on other baseline algorithms and report the result of robust accuracy in Table 7. The results indicate that multi-ϵ strategy remains helpful to improve the performance of other baseline algorithms. Most methods have better robust accuracy under the case of $\epsilon_{train} = 2\epsilon$. Still, our proposed Fast-EG-l_1 obtains the best robust accuracy on both CIFAR10 and CIFAR100.

Dataset	ϵ_{train}	FGSM-RS	ATTA	AdaAT	Grad-Align	N-FGSM	NuAT	**Fast-EG-l_1**
CIFAR10	ϵ	36.29	46.51	31.84	36.38	44.21	48.35	38.15
	2ϵ	33.77	34.21	35.41	39.76	44.96	49.05	**50.27**
CIFAR100	ϵ	33.23	33.74	28.64	33.19	35.79	36.46	32.15
	2ϵ	34.67	13.48	28.97	34.27	36.01	37.33	**38.03**

Table 7. Robust accuracy (in %) of other baseline algorithms with and without multi-ϵ trick on CIFAR10 and CIFAR100.

Visualization of adversarial examples before and after catastrophic overfitting. In Section 3.3 we did a careful study on the phenomenon of catastrophic overfitting for adversarial training against l_1 bounded perturbations. We find model overfits to sparse attacks and lose robustness against dense attacks when catastrophic overfitting happens, thus the perturbations generated by AA to fool the model have a much larger l_0 norm after catastrophic overfitting, as shown in Figure 4. Here we demonstrate some adversarial examples generated by AA before and after catastrophic for CIFAR10 in Figure 6. It's clear that the perturbations after catastrophic overfitting are much denser for each image.

Results of different values of ϵ on ImageNet100. We report the result of FGSM-RS and Fast-EG-l_1 on ImageNet100 with different values of ϵ in Table 8. Since the resolution of images in ImageNet100 is 224×224, it allows a much larger perturbation size in l_1 norm compared with CIFAR10 and CIFAR100. Consistent with the ablation study on CIFAR10, our algorithm Fast-EG-l_1 has consistently better robust accuracy than the baseline FGSM-RS for different values of ϵ.

Method$\backslash\epsilon$	12	24	48	72	96
FGSM-RS	62.06	54.84	44.18	36.64	31.96
Fast-EG-l_1	67.62	59.00	52.46	46.74	39.50

Table 8. Robust accuracy (in %) evaluated by AutoAttack of models trained by FGSM-RS and Fast-EG-l_1 under different values of ϵ on ImageNet100. Fast-EG-l_1 consistently achieves better performance.

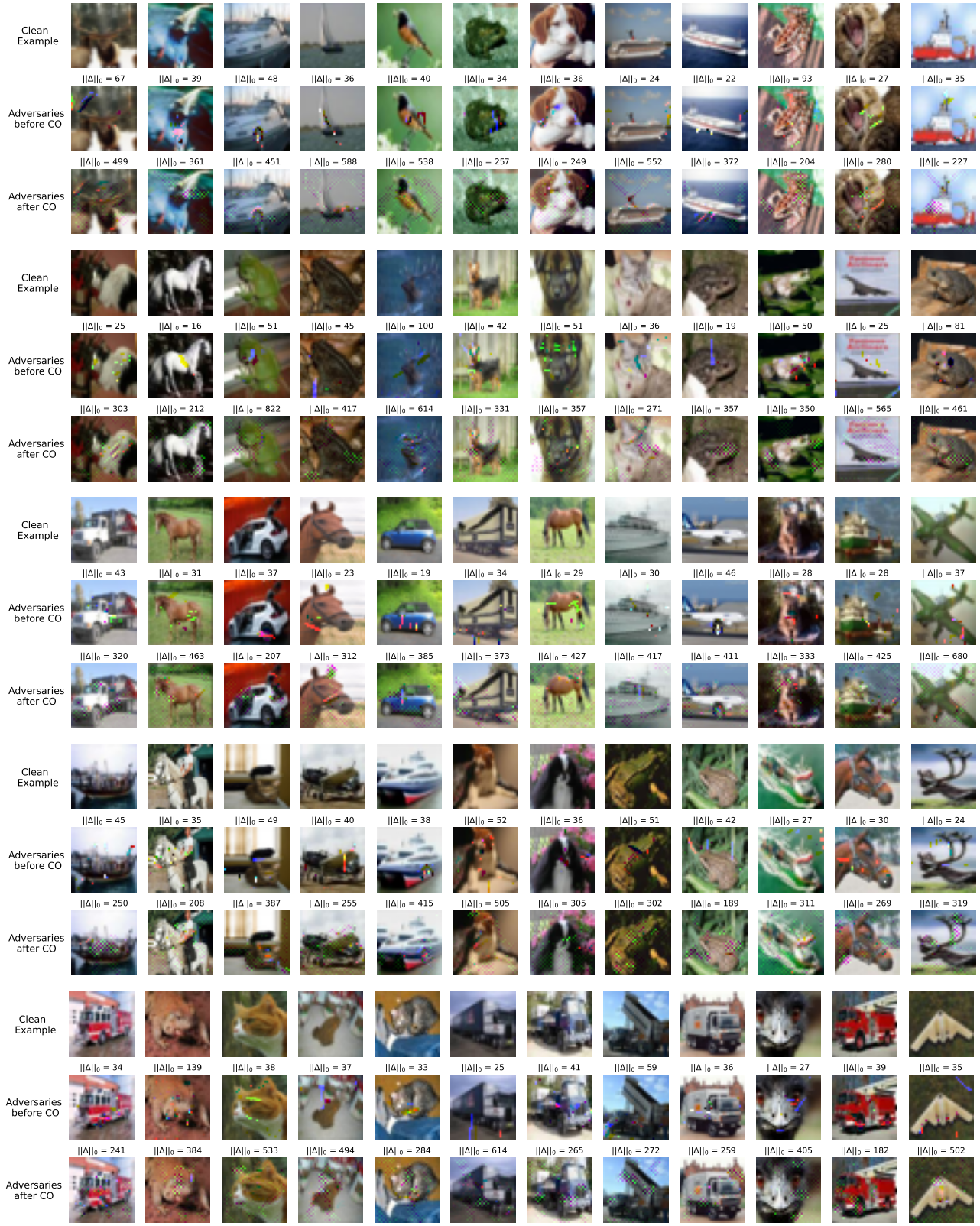

Figure 6. Visualization of adversarial examples before and after catastrophic overfitting (CO) in CIFAR10. The title of each subfigure indicates l_0 norm of the adversarial perturbation. The value of perturbation is enlarged $3\times$ for better visualization.