Track 1:
Sparse patches adversarial attacks via extrapolating point-wise information

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Sparse and patch adversarial attacks were previously shown to be applicable in realistic settings and are considered a security risk to autonomous systems. Sparse adversarial perturbations constitute a setting in which the adversarial perturbations are limited to affecting a relatively small number of points in the input. Patch adversarial attacks denote the setting where the sparse attacks are limited to a given structure, i.e., sparse patches with a given shape and number. However, previous patch adversarial attacks do not simultaneously optimize multiple patches' locations and perturbations. This work suggests a novel approach for sparse patches adversarial attacks via point-wise trimming of dense adversarial perturbations. Our approach enables simultaneous optimization of multiple sparse patches' locations and perturbations for any given number and shape. Moreover, our approach is also applicable for standard sparse adversarial attacks, where we show that it significantly improves the state-of-the-art over multiple extensive settings. A reference implementation of the proposed method and the reported experiments is provided at `https://anonymous.4open.science/r/sparse-patches-adversarial-attacks-3CF3`.

1 Introduction

Adversarial perturbations were first discovered in the context of deep neural networks (DNNs), where the networks' gradients were used to produce small bounded-norm perturbations of the input that significantly altered their output Szegedy et al. [2013]. Methods for optimizing such perturbations and the resulting perturbed inputs are denoted as adversarial attacks and adversarial inputs. Such attacks target the increase of the model's loss or the decrease of its accuracy and were shown to undermine the impressive performance of DNNs in multiple fields. The norm bounds on adversarial perturbations are usually discussed in either the L_∞ or L_2 norms Szegedy et al. [2013], Goodfellow et al. [2014], Madry et al. [2018]. Sparse adversarial attacks, in contrast, are a setting where L_0 norm bounds are applied and limit the perturbations to affect a relatively small number of points in the input. Sparsity L_0 norm bounds can also be applied in addition to the usually considered norms of L_∞, L_2 but we consider such out of the scope of the current work. Croce and Hein [2019], Fan et al. [2020], Croce and Hein [2021], Dong et al. [2020]. Patch adversarial attacks are a sub-setting of sparse attacks, where the perturbed points are constrained to constitute patches of a given shape and number. Patch adversarial attacks are highly realistic and were shown to be applicable in multiple real-world settings Nemcovsky et al. [2022], Xu et al. [2019], Zolfi et al. [2021], Wei et al. [2022a], Chen et al. [2019]. However, the optimization of sparse adversarial patches is computationally complex and entails the simultaneous optimization of the patches' locations and corresponding perturbations. Moreover,

Submitted to AdvML-Frontiers'24: The 3nd Workshop on New Frontiers in Adversarial Machine Learning@NeurIPS'24, Vancouver, CA. Do not distribute.

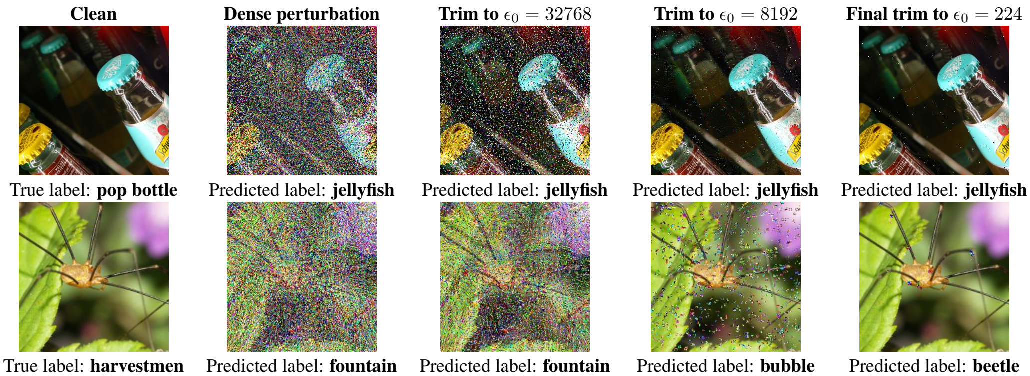

Figure 1: Flowchart of our sparse (top) and 2×2 patch (bottom) adversarial attacks trim process on Imagenet standard $Resnet50$ model, for attacks bounded to $\epsilon_0 = 224$. We present the adversarial inputs produced for distinct ϵ_0 bounds during the process and the predicted label for each, compared to the true label.

the locations' optimization is not directly differentiable and mandates a search over combinatorial spaces that grow exponentially with the number of patches. Previous patch attacks do not solve this optimization but rather a problem relaxation. Such attacks either optimize the perturbations over fixed locations Nemcovsky et al. [2022], Chen et al. [2019], optimize the locations of fixed patches Wei et al. [2022b], Zolfi et al. [2021], or limit the optimization to be over a single patch Wei et al. [2022a]. In contrast, previous sparse attacks that do not discuss patches suggest several approaches for simultaneously optimizing the selection of points to perturb and point-wise perturbations. To solve this complex optimization problem, Modas et al. [2019] first suggested approximating the non-convex L_0 norm by the convex L_1 norm, proposing the $SparseFool(SF)$ attack. Following this, Croce and Hein [2019] suggested to utilize binary optimization and presented the PGD_{L0} PGD-based Madry et al. [2018] attack. Goodfellow et al. [2020] then suggested first increasing the number of perturbed points, then reducing any unnecessary, presenting the $GreedyFool(GF)$ attack. Lastly, Zhu et al. [2021] suggested a homotopy algorithm and the $Homotopy$ attack.

In the present work, we suggest a novel approach for simultaneously optimizing multiple sparse patches' locations and perturbations. Our approach is based on point-wise trimming of dense adversarial perturbations and enables the optimization of patches for any given number and shape. To the best of our knowledge, this is the first direct solution to the complex optimization problem of adversarial patches. Moreover, our solution does not require differentiability during the trimming process and is therefore applicable to all the real-world settings presented in previous works Nemcovsky et al. [2022], Xu et al. [2019], Zolfi et al. [2021], Wei et al. [2022a], Chen et al. [2019]. In all these settings, our solution enables the optimization to be over a more extensive scope of patch adversarial attacks. In addition, our approach applies to standard sparse adversarial attacks, and we compare it to previous works on the $ImageNet$ classification task over various models. We consider ϵ_0 bounds up to the common sparse representation bound of root input size Candès et al. [2006] and show that we significantly outperform the state-of-the-art for all the considered settings.

2 Background

Let $\mathcal{X} \in [0,1]^n$ be some normalized data space comprising N data points, and we denote $[N] \equiv \{i\}_{i=1}^{N}$. Let $x \in \mathcal{X}$ be a data sample and let $\delta \in \mathcal{X}$ be a perturbation, for δ to be applicable on x it must be limited s.t. the perturbed data sample remains in the data space $x_\delta = x + \delta \in \mathcal{X}$. Let $GT : \mathcal{X} \to \mathcal{Y}$ be a ground truth function over $\mathcal{X}$ and target space $\mathcal{Y}$, and let $M : \mathcal{X} \to \mathcal{Y}$ be a model aiming to predict GT. Given a data sample $(x, y) \in \mathcal{X} \times \mathcal{Y}$, a criterion over the model prediction $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathcal{R}^+$, and L_0 norm bound $\epsilon_0 \in [N]$, a sparse adversarial attack $A_s : \mathcal{X} \times \mathcal{Y} \times [N] \to \mathcal{X}$ targets the maximization of the criterion over the data sample and bound:

$$A_s(x, y, \epsilon_0) = \arg \max_{\{\delta | x+\delta \in \mathcal{X}, \|\delta\|_0 \leq \epsilon_0\}} \ell(M(x + \delta), y) \tag{1}$$

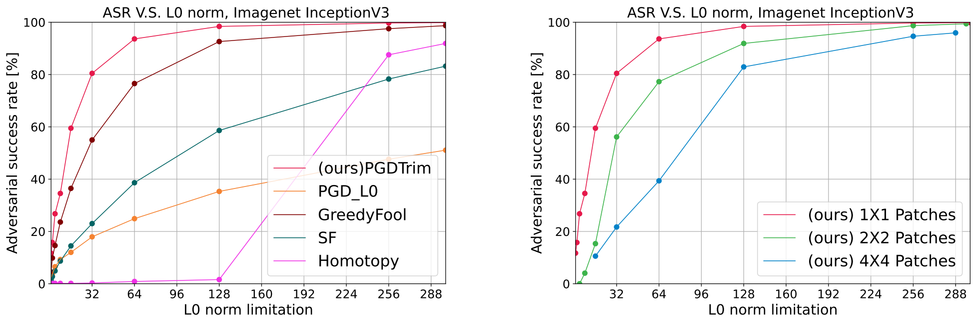

Figure 2: We compare our method to previous sparse attack works(left) and with various patch sizes (right) on the Imagenet dataset $InceptionV3$ model. We report the ASR as a function of l_0 for all attacks.

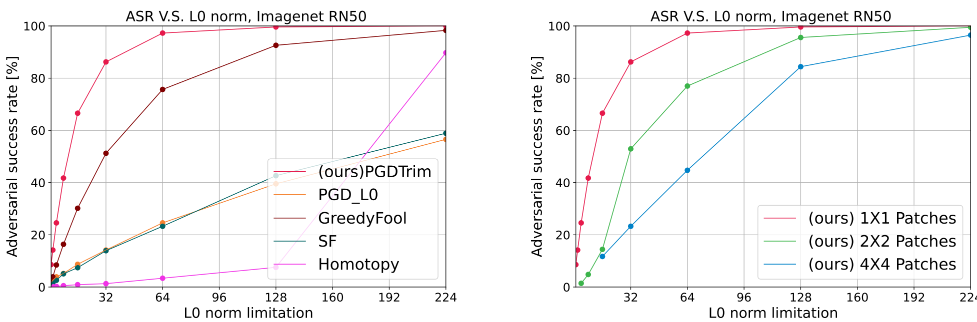

Figure 3: We compare our method to previous sparse attack works(left) and with various patch sizes (right) on the Imagenet dataset Resnet50 standard model. We report the ASR as a function of l_0 for all attacks.

For a given choice of points and corresponding binary mask $B \in \{0,1\}^N$, the point-wise multiplication $\delta_s = B \odot \delta$ defines a projection onto the L_0 norm-bound space. We denote the set of binary masks with exactly ϵ_0 ones as $C_{N,\epsilon_0} \subset \{0,1\}^N$ and, for $B \in C_{N,\epsilon_0}$, the L_0 norm of the resulting sparse perturbation δ_s is bound by $\|\delta_s\|_0 \leq \epsilon_0$. Sparse adversarial perturbations can be optimized using such projections Fan et al. [2020]. For an RGB normalized data space, we define the mask according to the pixels, i.e., $\mathcal{X} \in [0,1]^{H \times W \times 3}, N \equiv H \cdot W, C_{N,\epsilon_0} \subset \{0,1\}^{H \times W}$. Given an additional patch constraint with kernel $K \equiv (K_h, K_w) \in [H] \times [W]$, the perturbed points are limited to form exactly $\frac{\epsilon_0}{K_h \cdot K_w}$ patches of K's shape, where we only consider accordingly divisible parameters. We denote the corresponding set of binary masks as $C_{N,\epsilon_0}^{K_h \times K_w}$. We allow for partial overlapping patches, as for sufficiently large kernels and ϵ_0 bounds, most and then all of the binary masks $B \in C_{N,\epsilon_0}^{K_h \times K_w}$ will contain such. The patch adversarial attack is then denoted as $A_p : \mathcal{X} \times \mathcal{Y} \times [N] \times [H] \times [W] \rightarrow \mathcal{X}$ and we formulate the attacks targets as:

$$\delta_s \equiv A_s(x, y, \epsilon_0) = \arg \max_{\{\delta_s = B \odot \delta | x + \delta \in \mathcal{X}, B \in C_{N,\epsilon_0}\}} \ell(M(x + \delta_s), y) \tag{2}$$

$$\delta_p \equiv A_p(x, y, \epsilon_0, K_h, K_w) = \arg \max_{\{\delta_s = B \odot \delta | x + \delta \in \mathcal{X}, B \in C_{N,\epsilon_0}^{K_h \times K_w}\}} \ell(M(x + \delta_s), y) \tag{3}$$

3 Method

We define our approach for point-wise evaluation and corresponding trimming of adversarial perturbations. We first present the process we denote as $TrimStep$ and discuss its optimization target and the point-wise evaluation criterion it utilizes. We discuss the underlying assumptions under which

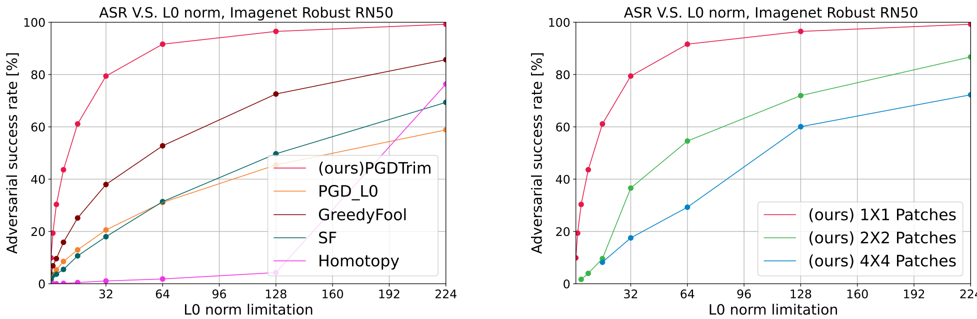

Figure 4: We compare our method to previous sparse attack works(left) and with various patch sizes (right) on the Imagenet dataset Resnet50 robust model. We report the ASR as a function of l_0 for all attacks.

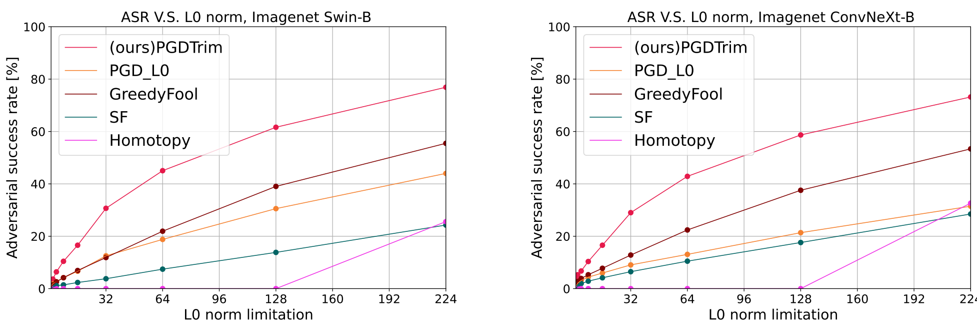

Figure 5: We compare our method to previous works on the Imagenet dataset, visual transformer-based SwinB model (left), and ConvNextB model (right). We report the ASR as a function of l_0 over sparse adversarial attacks.

this process is most accurate. We then discuss our suggested sparse and patch adversarial attacks, which utilize the $TrimStep$ while aiming to fulfill the underlying assumptions.

3.1 $TrimStep$

Let $x, y, M, \ell, \epsilon_0$ be defined as in Eq. (1), and let δ be a somewhat denser pre-optimized adversarial perturbation $\|\delta\|_0 > \epsilon_0$. In this process, we aim to extrapolate a binary mask $B \in C_{N,\epsilon_0}$ from δ while targeting the proceeding optimization of a sparse perturbation under the fixed binary mask:

$$\begin{aligned} \delta_s^B &= \arg \max_{\{\delta_s = B \odot \delta | x+\delta \in \mathcal{X}\}} \ell(M(x+\delta_s), y) \\ B &= \arg \max_{B \in C_{N,\epsilon_0}} \delta_s^B \end{aligned} \tag{4}$$

For this purpose, we consider the distributions of binary masks $B \in C_{N,\epsilon_0}$ and criteria $\ell(M(x + B \odot \delta), y), \ell(M(x + \delta_s^B)$ as prior and posterior distributions. We then define a point-wise evaluation criterion over the distributions and approximate the point-wise evaluation of the posterior by the prior. We denote the point-wise criterion over δ_s^B as $L_{\delta_s} \in \mathbb{R}^N$, and formally define the evaluation and its approximation:

$$L_{\delta_s} = \mathbb{E}_{B \in C_{N,\epsilon_0}} \ell(M(x+\delta_s), y) \cdot B \tag{5}$$

$$\approx \mathbb{E}_{B \in C_{N,\epsilon_0}} \ell(M(x + B \odot \delta), y) \cdot B \tag{6}$$

While computing L_{δ_s} directly is infeasible, we can efficiently compute the approximation given δ. As the number of possible masks $|C_{N,\epsilon_0}|$ may be infeasible to compute, we further approximate this evaluation via Monte Carlo sampling. For each point in the data sample, the point-wise value of L_{δ_s} is the expectation of the attack target over binary masks that indicate the perturbation of the corresponding point. Accordingly, this evaluation estimates the expected benefit of each point selection to the attack target in Eq. (4). We, therefore, extrapolate the binary mask B to perturb the top evaluated points, according to Eq. (6). Similarly, given an additional patch kernel constraint K defined as in Eq. (3), the same process applies over the corresponding set of binary masks. Formally:

$$B_s = \arg\max_{B \in C_{N,\epsilon_0}} L_{\delta_s}^T \cdot B \tag{7}$$

$$B_p = \arg\max_{B \in C_{N,\epsilon_0}^{K_h \times K_w}} L_{\delta_s}^T \cdot B \tag{8}$$

The maximization in Eq. (7) can be implemented directly as the top evaluated points in L_{δ_s}; however, for Eq. (8), we need to account for overlapping patches. We, therefore, use a max-out scheme when choosing the best patches, where the best patch in each step is chosen according to the sum of L_{δ_s} over the corresponding points. We then zero the L_{δ_s} values for the chosen patch to eliminate their benefit when considering overlapping patches. We can employ a similar process while applying a binary mask over the points in the kernel K to allow for optimization of patches of any given shape. However, we consider this out of the scope of the current work.

There are two approximations in the $TrimStep$ process. The first of which is approximating the best mask in Eq. (4) as in Eq. (7), and the second is approximating the posterior in Eq. (5) via the prior in Eq. (6). We consider several assumptions for which these approximations should be most accurate. We first assume that attack criterion ℓ mainly depends on selecting significant points in the dense perturbation rather than a well-correlated group. Secondly, we assume that δ is sufficiently robust to the projections $B \odot \delta$, s.t., the decrease in the criterion for top evaluated points in L_{δ_s}, $\ell(M(x+\delta), y) \to \ell(M(x + B \odot \delta), y)$ is mainly due to trimming less significant points. Finally, we assume that the L_0 gap between the perturbations $\Delta\epsilon_0 \equiv \|\delta\|_0 - \epsilon_0$ is sufficiently small as it aids our previous assumptions. This entails that the point-wise significance should remain relatively unaltered between perturbations and limits the effect of the projections $B \odot \delta$. Under these assumptions, the top evaluated points according to L_{δ_s} should correlate well with the optimal mask selection in Eq. (4), and more so for sufficiently small $\Delta\epsilon_0$. Moreover, the top evaluated points in both Eq. (6) and Eq. (5) should correlate to the points' importance in the $dense$ perturbation and, therefore, to each other. Thereby indicating the accuracy of the approximations in the $TrimStep$.

3.2 $PGDTrim$ and $PGDTrimKernel$

We continue to present our suggested sparse and patches adversarial attack based on the PGD iterative optimization scheme Madry et al. [2018]. Both attacks use the same optimization scheme and differ only in utilizing the corresponding $TrimStep$. This optimization scheme aims to mitigate the inaccuracy of $TrimStep$ by fulfilling the underlying assumptions. The assumption on the attack criterion cannot be directly mitigated as it depends on the task; however, the other assumptions of small $\Delta\epsilon_0$ and robust δ are highly dependent on the optimization scheme. To fulfill the small $\Delta\epsilon_0$ assumption, we use a trimming schedule containing several applications of $TrimStep$ to gradually decrease the L_0 norm of the optimized perturbations until reaching the ϵ_0 bound. We consider a logarithmic trimming schedule with up to $n_{trim} = \lceil log_2(N) \rceil - \lfloor log_2(\epsilon_0) \rfloor$ trim steps, where N is the input size and ϵ_0 is the L_0 norm bound. In addition, before each application of $TrimStep$, we optimize the current dense perturbation δ via the PGD scheme. To improve the robustness of δ to the perturbations $B \odot \delta$ we employ a corresponding dropout scheme. The dropout we consider in training the perturbations depends on the distributions of binary masks in the proceeding trim step. For a given current and following L_0 norms L_0^{curr}, L_0^{next}, the binary masks in the proceeding trim step are sampled from the set $B \in C_{L_0^{curr}, L_0^{next}}$. We consider Bernoulli dropout from the corresponding distribution $Bernoulli(L_0^{next}/L_0^{curr})$, as it best simulates the binary mask projection. We present a flowchart of our attacks in Fig. 1, and in the supplementary material, we continue to discuss our optimization scheme and provide an entire algorithm of the resulting attacks.

4 Experiments

Experimental settings. We now present an empirical evaluation of the proposed method. We compare our method to previous sparse attacks on the $ImageNet$ classification task Deng et al.

[2009] over various models. We present each attack's adversarial success rate (ASR), dependent on the L_0 norm bound, and show the result of our proposed method for both sparse and patch attacks. The L_0 norm bounds we consider are all values up to root input size $\epsilon_0 = \sqrt{N}$, and we present the performance of the compared attacks for powers of 2 in this range. The considered models are then the $InceptionV3$ Szegedy et al. [2016], standardly trained $Resnet50$ model Koonce and Koonce [2021], adversarially robust $Resnet50$ model, and the visual transformer-based Swin-B Liu et al. [2021] and ConvNeXt-B models Liu et al. [2022]. We use the pre-trained models made available by Croce et al. [2020], and the adversarially robust $Resnet50$ we consider is the corresponding state-of-the-art adversarial defense suggested by Salman et al. [2020], which we denote as robust $Resnet50$. The input size for the $InceptionV3$ model is then $N = 299$, and $N = 224$ for all other models.

In our method, for all the presented settings, we use $K = 100$ PGD iterations for optimizing perturbations and $MC = 1000$ Monte Carlo samples in our trim steps, where if these samples are sufficient, we compute the expression in Eq. (6) directly. We compute the attacks for $n_{trim} = 11$ trim steps and $n_{restarts} = 11$ restarts; we use the PGD restarts optimization scheme to re-initiate the attack with fewer trim steps, as doing so will result in different perturbations and allow for re-evaluation of points trimmed in the extra steps. We use the default settings suggested by the authors for all the compared attacks for all the presented settings. In addition, as GF, SF, and $Homotopy$ attacks minimize the L_0 for each sparse adversarial perturbation instead of utilizing ϵ_0 bounds, we report their ASR for each L_0 limitation as the rate of produced adversarial perturbations with correspondingly bounded L_0 norms.

4.1 Experimental results

In Fig. 1, we show the trimming process of our sparse and patch attacks. We see that the perturbed points are gradually trimmed until reaching the ϵ_0 bounds with the most significant points remaining. In Fig. 2, we compare the ASR of previous sparse attacks to our sparse and patch attacks on the $InceptionV3$ model. In this setting, our sparse attack achieves the best ASR on all the presented attacks and 100% ASR starting from $\epsilon_0 = 128$. The second best sparse attack is GF, which shows comparable results to our patch attack over 2×2 patches. Our patch attacks over 4×4 achieve somewhat lower results, possibly due to the attacks' scope being more limited under this patch constraint. In Fig. 3, we compare the ASR of previous sparse attacks to our sparse and patch attacks on the standard $Resnet50$ model. Similarly, our sparse attack achieves the best ASR and 100% starting from $\epsilon_0 = 128$. Our results for 2×2 and 4×4 patches are again somewhat lower than our sparse attack, with the 2×2 setting comparable to the second-best sparse attack, GF. In Fig. 4, we compare the ASR of previous sparse attacks to our sparse and patch attacks on the robust $Resnet50$ model. Our sparse attack again achieves the best ASR with 100% achieved at $\epsilon_0 = 224$, corresponding to the model's robustness. Moreover, these results significantly outperform all other sparse attacks, which may entail that our method performs relatively better in robust settings. Our results for 2×2 and 4×4 patches are significantly lower than those of our sparse attack, yet the 2×2 setting is still comparable to the second-best sparse attack, GF. In Fig. 5, we compare the ASR of previous sparse attacks to our sparse attack on the $Swin - B$ and $ConvNeXt$ VIT models. Similarly, our sparse attack achieves the best ASR on all the compared settings and significantly outperforms other sparse attacks.

5 Discussion

This paper proposes novel sparse and patch adversarial attacks based on point-wise trimming of dense adversarial perturbations. For that purpose, we suggest ranking the points based on their average significance over potential resulting perturbations. We then approximate this significance based on the dense perturbation and choose the most significant points for our attacks under the corresponding constraints. Our sparse attack achieves state-of-the-art results for all the considered L_0 bounds. Moreover, our 2×2 patch attack shows results comparable to previous sparse attacks. The success of our method suggests that our point-wise evaluation may correspond to the significance of points in the input sample and not only in the adversarial perturbation. Therefore, our trimming-based approach is an efficient optimization method for sparse and patch attacks. In addition, our approach is the first to enable simultaneous optimization of multiple patches' locations and perturbations. Our approach does not require differentiability during trimming and applies to various real-world settings.

References

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. URL http://arxiv.org/abs/1312.6199.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. URL http://arxiv.org/abs/1412.6572.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4724–4732, 2019.

Yanbo Fan, Baoyuan Wu, Tuanhui Li, Yong Zhang, Mingyang Li, Zhifeng Li, and Yujiu Yang. Sparse adversarial attack via perturbation factorization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 35–50. Springer, 2020.

Francesco Croce and Matthias Hein. Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers. In *International Conference on Machine Learning*, pages 2201–2211. PMLR, 2021.

Xiaoyi Dong, Dongdong Chen, Jianmin Bao, Chuan Qin, Lu Yuan, Weiming Zhang, Nenghai Yu, and Dong Chen. Greedyfool: Distortion-aware sparse adversarial attack. *Advances in Neural Information Processing Systems*, 33:11226–11236, 2020.

Yaniv Nemcovsky, Matan Jacoby, Alex M Bronstein, and Chaim Baskin. Physical passive patch adversarial attacks on visual odometry systems. In *Proceedings of the Asian Conference on Computer Vision*, pages 1795–1811, 2022.

Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Evading real-time person detectors by adversarial t-shirt. *arXiv preprint arXiv:1910.11099*, 2019. URL http://arxiv.org/abs/1910.11099.

Alon Zolfi, Moshe Kravchik, Yuval Elovici, and Asaf Shabtai. The translucent patch: A physical and universal attack on object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15232–15241, 2021.

Xingxing Wei, Ying Guo, Jie Yu, and Bo Zhang. Simultaneously optimizing perturbations and positions for black-box adversarial patch attacks. *IEEE transactions on pattern analysis and machine intelligence*, 2022a.

Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*, pages 52–68. Springer, 2019.

Xingxing Wei, Ying Guo, and Jie Yu. Adversarial sticker: A stealthy attack method in the physical world. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2711–2725, 2022b.

Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: a few pixels make a big difference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9087–9096, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Mingkang Zhu, Tianlong Chen, and Zhangyang Wang. Sparse and imperceptible adversarial attack via a homotopy algorithm. In *International Conference on Machine Learning*, pages 12868–12877. PMLR, 2021.

Emmanuel J Candès et al. Compressive sampling. In *Proceedings of the international congress of mathematicians*, volume 3, pages 1433–1452. Madrid, Spain, 2006.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Brett Koonce and Brett Koonce. Resnet 50. *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, pages 63–72, 2021.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.

A Adversarial attacks

A.1 Optimization scheme

We continue to discuss the optimization scheme we use in the attack as described in Section 3.2. We continue the discussion on the trimming schedule and offer continuous alternatives to the Bernoulli dropout. We have previously defined the number of trimming steps n_{trim}, and we now detail the logarithmic trimming schedule we consider. We first define the L_0 norm values to which we trim the perturbation in each step. The first perturbation we train is always whole $\|\delta_{init}\|_0 = N$, and the last is always constrained to ϵ_0. For the maximal number of trim steps, the L_0 norms to which we trim and train perturbations are:

$$N, 2^{\lceil log_2(N) \rceil - 1}, 2^{\lceil log_2(N) \rceil - 2}, \dots, 2^{\lfloor log_2(\epsilon_0) \rfloor + 1}, \epsilon_0 \tag{9}$$

For fewer trim steps, we skip a corresponding number of L_0 norms, where we attempt to keep the L_0 decrease ratio relatively fixed and otherwise slightly lower for the initial trim steps. In addition, we use the PGD restarts optimization scheme to re-initiate the attack with fewer trim steps, as doing so will result in different perturbations and allow for re-evaluation of points trimmed in the extra steps.

Concerning the continuous alternatives to the Bernoulli dropout, we consider the continuous Bernoulli and Gaussian dropouts, for which we preserve the mean as in the Bernoulli dropout and, when possible, the standard deviation.

A.2 Attacks Algorithms

We introduce algorithms for our sparse adversarial attack (Algorithm 2), our patch adversarial attack (Algorithm 3), and the PGD-based optimization scheme they make use of (Algorithm 1). We first present the optimization scheme, which we denote as $Dropout - PGD(DPGD)$, then continue to present our sparse and patch attacks while using $DPGD$ as a procedure. Given a binary projection, dropout distribution, and initial perturbation, $DPGD$ optimizes a corresponding perturbation for maximized attack criterion. Our sparse and patch attacks then use $DPGD$ to optimize perturbations and then trim them using our point-wise evaluation. Given trim steps L_0 norms and dropout distribution class, our sparse attack utilizes $DPGD$ to optimize a corresponding perturbation in each trim step, then trim it to be the initial perturbation for the next step. Given an additional kernel constraint $K \equiv (K_h, K_w)$, our patch attack similarly optimizes and trims the perturbation but limits the resulting perturbation to consist of patches of K's shape. Once the trimming process is finished, it returns the final binary mask, and an additional $DPGD$ procedure maximizes a corresponding perturbation. The L_0 bound is thereby specified in the norm of the last trim step.

Algorithm 1 $Dropout - PGD(DPGD)$

Input M: attacked model
Input (x, y): input sample
Input ℓ: attack criterion
Input B: Binary projection
Input δ_{init}: perturbation initialization
Input D: dropout distribution
Input $Iter$: number PGD iterations
Input α: Step size for the attack

initialize perturbation:
$\delta_{\text{best}} \leftarrow \delta_{\text{init}}$
$\text{Loss}_{\text{best}} \leftarrow \ell(M(x + \delta_{\text{best}}), y)$
for $k = 1$ to $Iter$ **do**
  optimization step:
  $g \leftarrow \nabla_\delta \ell(M(x + D(\delta)), y)$
  $\delta \leftarrow \delta + \alpha \cdot B \odot \text{sign}(g)$
  $\delta \leftarrow clip(\delta, -x, 1 - x)$
  evaluate perturbation:
  $\text{Loss} \leftarrow \ell(M(x + \delta), y)$
  if $\text{Loss} > \text{Loss}_{\text{best}}$ **then**
    $\delta_{\text{best}} \leftarrow \delta$
    $\text{Loss}_{\text{best}} \leftarrow \text{Loss}$
  end if
end for
return δ_{best}

Algorithm 2 $PGDTrim$ sparse adversarial attack

Input M: attacked model
Input N: input size
Input (x, y): input sample
Input ℓ: attack criterion
Input $TrimSteps$: trim steps l_0^{curr}, l_0^{next} norms
Input $Dropout$: dropout distribution class
Input MC: number Monte Carlo samples
Input $Iter$: number PGD iterations
Input α: Step size for the attack

initialize perturbation:
$B_{\text{trim}} \leftarrow \{1\}^N$
$\delta_{\text{best}} \leftarrow \text{Uniform}(-1, 1)^N$
$\text{Loss}_{\text{best}} \leftarrow \ell(M(x + \delta_{\text{best}}), y)$
for l_0^{curr}, l_0^{next} in $TrimSteps$ **do**
  perturbation optimization:
  $D \leftarrow Dropout(l_0^{next}/l_0^{curr})$
  $\delta_{\text{best}} \leftarrow \text{DPGD}(M, (x, y), \ell, B_{\text{trim}}, \delta_{\text{best}}, D, Iter, \alpha)$
  point-wise evaluation:
  $\text{BLoss} \leftarrow \{0\}^N$
  $\text{BCount} \leftarrow \{0\}^N$
  for $i = 1$ to MC **do**
    $B \leftarrow \text{Multinomial}(l_0^{next}, B_{\text{trim}})$
    $\text{BLoss} \leftarrow \text{BLoss} + \ell(M(x + B \odot \delta_{\text{best}}), y) \cdot B$
    $\text{BCount} \leftarrow \text{BCount} + B$
  end for
  $\text{BLoss} \leftarrow \text{BLoss}/\text{BCount}$
  trim step:
  $B_{\text{trim}} \leftarrow \{0\}^N + B_{\text{trim}}[\text{TopK}(l_0^{next}, \text{BLoss})]$
  $\delta_{\text{best}} \leftarrow B_{\text{trim}} \odot \delta_{\text{best}}$
  $\text{Loss}_{\text{best}} \leftarrow \ell(M(x + \delta_{\text{best}}), y)$
end for
final perturbation optimization:
$D \leftarrow \text{Identity}$
$\delta_{\text{best}} \leftarrow \text{DPGD}(M, (x, y), \ell, B_{\text{trim}}, \delta_{\text{best}}, D, Iter, \alpha)$
return δ_{best}

Algorithm 3 $PGDTrimKernel$ patch adversarial attack

Input M: attacked model
Input N: input size
Input (x, y): input sample
Input ℓ: attack criterion
Input $TrimSteps$: trim steps l_0^{curr}, l_0^{next} norms
Input $K = (K_h, K_w)$: Kernel patch constraint
Input $Dropout$: dropout distribution class
Input MC: number Monte Carlo samples
Input $Iter$: number PGD iterations
Input α: Step size for the attack

initialize perturbation:
$B_{\text{trim}} \leftarrow \{1\}^N$
$K_{\text{size}} \leftarrow K_h \cdot K_w$
$\delta_{\text{best}} \leftarrow \text{Uniform}(-1, 1)^N$
$\text{Loss}_{\text{best}} \leftarrow \ell(M(x + \delta_{\text{best}}), y)$
for l_0^{curr}, l_0^{next} in $TrimSteps$ **do**
  perturbation optimization:
  $D \leftarrow Dropout(l_0^{next}/l_0^{curr})$
  $\delta_{\text{best}} \leftarrow \text{DPGD}(M, (x, y), \ell, B_{\text{trim}}, \delta_{\text{best}}, D, Iter, \alpha)$
  point-wise evaluation:
  $\text{BLoss} \leftarrow \{0\}^N$
  $\text{BCount} \leftarrow \{0\}^N$
  $B_{\text{kernel}} \leftarrow \text{MaxPool}(B_{\text{trim}}, K)$
  for $i = 1$ to MC **do**
    $B \leftarrow \text{Multinomial}(l_0^{next}/K_{\text{size}}, B_{\text{kernel}})$
    $B \leftarrow \text{MaxPool}(\text{Pad}(B, ((K_h - 1, K_h - 1), (K_w - 1, K_w - 1))), K)$
    $\text{BLoss} \leftarrow \text{BLoss} + \ell(M(x + B \odot \delta_{\text{best}}), y) \cdot B$
    $\text{BCount} \leftarrow \text{BCount} + B$
  end for
  $\text{BLoss} \leftarrow \text{BLoss}/\text{BCount}$
  trim step:
  $B_{\text{trim}} \leftarrow \{0\}^N$
  for $i = 1$ to l_0^{next} **do**
    $B_{\text{Max}} \leftarrow \text{OneHot}(\text{ArgMax}(\text{SumPool}(\text{BLoss}, K)))$
    $B_{\text{MaxKernel}} \leftarrow \text{MaxPool}(\text{Pad}(B_{\text{Max}}, ((K_h - 1, 0), (K_w - 1, 0)), K)$
    $B_{\text{trim}} \leftarrow B_{\text{trim}} + B_{\text{MaxKernel}}$
    $\text{BLoss} \leftarrow \text{BLoss} \odot (1 - B_{\text{MaxKernel}})$
  end for
  $\delta_{\text{best}} \leftarrow B_{\text{trim}} \odot \delta_{\text{best}}$
  $\text{Loss}_{\text{best}} \leftarrow \ell(M(x + \delta_{\text{best}}), y)$
end for
final perturbation optimization:
$D \leftarrow \text{Identity}$
$\delta_{\text{best}} \leftarrow \text{DPGD}(M, (x, y), \ell, B_{\text{trim}}, \delta_{\text{best}}, D, Iter, \alpha)$
return δ_{best}