MALT Powers Up Adversarial Attacks

Odelia Melamed [*]   Gilad Yehudai[†]   Adi Shamir [‡]

Abstract

Current adversarial attacks for multi-class classifiers choose the target class for a given input naively, based on the classifier's confidence levels for various target classes. We present a novel adversarial targeting method, *MALT - Mesoscopic Almost Linearity Targeting*, based on medium-scale almost linearity assumptions. Our attack wins over the current state of the art AutoAttack on the standard benchmark datasets CIFAR-100 and ImageNet and for a variety of robust models. In particular, our attack is *five times faster* than AutoAttack, while successfully matching all of AutoAttack's successes and attacking additional samples that were previously out of reach. We then prove formally and demonstrate empirically that our targeting method, although inspired by linear predictors, also applies to standard non-linear models.

1 Introduction

Neural networks are widely known to be susceptible to adversarial perturbations (Szegedy et al. [2013]), which are typically imperceptible by humans. Many different papers have shown how to construct such attacks, where adding a small perturbation to the input significantly changes the output of the model (Carlini and Wagner [2017], Papernot et al. [2017], Athalye et al. [2018]). To protect from these attacks, researchers have tried to develop more robust models using several different techniques, such as adversarial training using different attacks (Madry et al. [2017], Papernot et al. [2016], Liu et al. [2023], Wang et al. [2023]).

The current state of the art adversarial attack, known as AutoAttack (Croce and Hein [2020b]), combines several different parameter-free attacks, some targeted and some untargeted. AutoAttack currently leads the RobustBench benchmark (Croce et al. [2020]), which is the standard benchmark for adversarial robustness. Notably, the targeted attacks used in AutoAttack pick the adversarial target classes according to the model's confidence levels and attack the top nine classes, even though CIFAR-100 and ImageNet have many more possible target classes. The reason for attacking only a limited number of classes, rather than all possible classes, is computational, as each such attack has a significant running time.

The reason that adversarial examples exist remains a hot topic of debate, specifically, whether it is due to the highly non-linear landscape of neural networks or rather to their local linearity properties. In Goodfellow et al. [2014], the authors provide several strong arguments for the local linearity hypothesis while presenting FGSM, a one-step gradient attack. Following Bubeck et al. [2021], we consider three distance scales around each fixed data point where linearity properties should be studied separately. On the *macroscopic* scale, neural networks are highly non-linear functions and have very complicated decision boundaries. On the other hand, neural networks with ReLU activation are piecewise linear, and thus at the *microscopic* scale (in which no ReLU input changes signs), they can be seen as completely linear functions. The third is the intermediate *mesoscopic* scale, which characterizes the typical distances between inputs to their nearest adversarial examples.

---

[*]Weizmann Institute of Science, Israel, `odelia.melamed@weizmann.ac.il`

[†]Center for Data Science, New York University, `gy2219@nyu.edu`

[‡]Weizmann Institute of Science, Israel, `adi.shamir@weizmann.ac.il`.

At this mesoscopic scale, Bubeck et al. [2021] proved that random networks are *locally almost linear*, hinting that for adversarial examples, we can use prediction techniques that are motivated by the behavior of linear functions.

In this paper, we present a novel adversarial attack, *MALT* (Mesoscopic Almost Linearity Targeting), which is five times faster than the current SOTA attack while exceeding its success rate. It uses the common fast APGD attack (Croce and Hein [2020b]) with a new targeting algorithm. The basic idea of MALT is that instead of sorting the target classes only by the confidence levels of the attacked model, *MALT* normalizes the class's confidence by the norm of the row of the Jacobian corresponding to it. This ordering is motivated by the case of linear classifiers that can be fully analyzed, and makes use of the almost linearity of the model at the mesoscopic scale.

Our attack wins over AutoAttack on both CIFAR-100 and ImageNet datasets for different robust models from the standard RobustBench benchmark. In particular, in all of our experiments, we show that the MALT targeting algorithm eliminates the need to use any attacks other than APGD: for every image that AutoAttack successfully attacks, MALT also succeeds while reaching several additional images on which AutoAttacks fails. A major benefit of MALT is that on SOTA robust models, our attack runs *five times faster* than AutoAttack on the ImageNet attack test dataset.

To further support the mesoscopic almost linearity hypothesis presented in Goodfellow et al. [2014] and Bubeck et al. [2021], we demonstrate both theoretically and empirically that our targeting method, though inspired by linearity considerations, also applies to non-linear models. On the theoretical side, we consider the setting of Melamed et al. [2023], Haldar et al. [2024] where the data resides on a low dimensional manifold, and prove that the network remains "almost linear" at the mesoscopic scale around data points, deducing that the ordering provided by our targeting method is preserved in that scale. Empirically, we demonstrate that models tend to be almost linear close to data points by showing that a linear approximation successfully predicts the model's output when taking random or adversarial steps away from a data point.

2   Related Works

Linearity and adversarial attacks   The local linearity of non-linear classifier in the context of adversarial example was first hypothesized for explaining the adversarial examples' existence (Goodfellow et al. [2014]), presenting the successful FGSM attack. Later, multi-step attacks with higher success rates have shown that the standard classifiers, indeed, are not entirely linear in the mesoscopic scale (Madry et al. [2017], Carlini and Wagner [2017]). Local linearity was also researched in the robustness context, finding linearity constraints increases the classifier robustness (Uesato et al. [2018], Sarkar and Iyengar [2020], Qin et al. [2019]) and the attack transferability (Papernot et al. [2017], Guo et al. [2020]).

Targeting methods   As a non-naive targeting method has yet to be proposed, the FAB attack (Croce and Hein [2020a]) and DeepFool attack (Moosavi-Dezfooli et al. [2016]) are using a step-wise targeting method to boost the untargeted version of the attack. The step-wise point of view, looking at the microscopic scale, allows assuming linearity, and estimating the best target becomes an easy linear problem. Sarkar and Iyengar [2020] uses the same simple method for linearity-enforced classifiers. Our targeting method implements the same linearity-based targeting idea but is calculated once at the starting point.

Theory of adversarial attacks   The source of adversarial examples has been extensively researched in recent years. Building on Shamir et al. [2019], it was shown in Daniely and Shacham [2020] that adversarial perturbations appear in random networks. This result was extended to a larger family of networks with random weights (Bartlett et al. [2021], Bubeck et al. [2019, 2021], Montanari and Wu [2022]). In Vardi et al. [2022], Frei et al. [2024], the authors show that the gradient method is implicitly biased towards non-robust networks. The hypothesis of the data lies in a low-dimensional manifold, which the adversarial examples are perpendicular to was suggested in both theoretical (Gilmer et al. [2018], Tanay and Griffin [2016], Melamed et al. [2023]) and applied (Song et al. [2017], Stutz et al. [2019], Shamir et al. [2021]) settings. Melamed et al. [2023] proved that for data that lies on a linear subspace, there exist adversarial perturbations. Their result is extended in Haldar et al. [2024] to clustered data.

3 MALT – Mesoscopic Almost Linearity Targeting

The targeted attacks in AutoAttack choose nine target classes to attack, while ImageNet, for example, contains 1000 classes. This is done since the running time of each such targeted attack is long, and attacking each image in the entire ImageNet test dataset for a large number of classes will be cumbersome. Hence, it is extremely important to choose the right nine target classes. Otherwise, the attack may fail on those target classes, while it may succeed on other classes that are not chosen.

The naive targeting in AutoAttack chooses the top nine classes sorted by the output of the model, which corresponds to the confidence the model gives to each class. In Figure 1, we show two examples of images from the ImageNet dataset on which AutoAttack fails, both with targeted and untargeted attacks, while MALT succeeds. For each image $\mathbf{x}$, after finding the adversarial perturbation $\mathbf{v}$ we divide it into 100 equal parts, namely $\mathbf{v}_i = \frac{i}{100}\mathbf{v}$ for $i = 0, \ldots, 100$. We plot the network's confidence levels for each class $N(\mathbf{x} + \mathbf{v}_i)$ (y-axis) w.r.t the interval $i = 0, ..., 100$ (x-axis). The advantage of MALT here is that it allows targeting classes that are overlooked by AutoAttack. Before presenting the MALT attack, we motivate it by analyzing adversarial attacks in linear models. Additional examples where MALT finds adversarial examples while AutoAttack fails can be found in Appendix A.
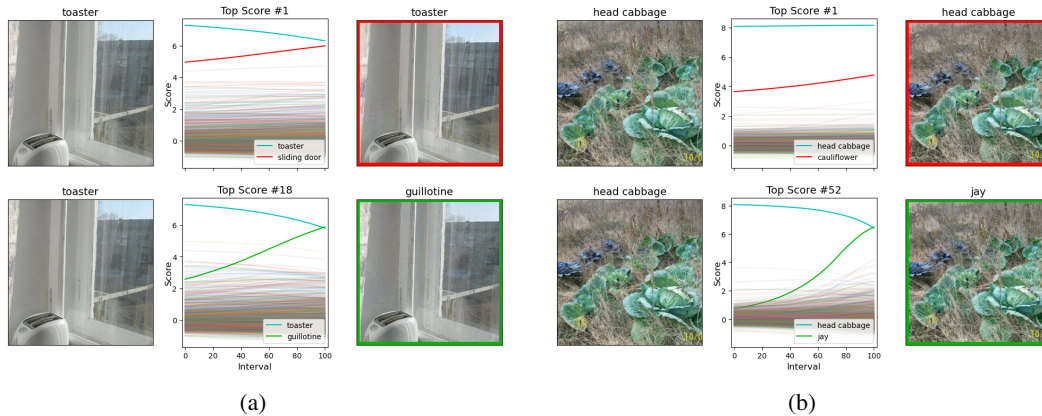

Figure 1: Examples of images from the ImageNet dataset that AutoAttack fails to attack while MALT succeeds. The top row shows an APGD attack on the target class with the highest logit, and the bottom row shows an APGD attack on the class which MALT finds and succeeds, corresponding to the (a) 18th and (b) 52nd classes with the highest logits. The images are shown before and after the attack, and the change in logits is presented in the middle column.

3.1 Motivation – The best target in a linear function

Naive targeting methods consider the top-c classes with the largest output of the model, i.e., the top-c logits (often, for $c = 9$), which corresponds to the classes that receive the highest confidence from the model. However, our goal is to find the target class for which its adversarial perturbation is the smallest from a given data point. In fact, it is not the case in general that the targets with the highest confidence will also have a small adversarial perturbation. For linear predictors, it is possible to fully analyze and find the target class with the smallest adversarial perturbation, as shown in the following:

Lemma 3.1. *Consider a linear predictor over k classes of the form $F(\mathbf{x}) = W\mathbf{x} + \mathbf{b}$ where $\mathbf{x} \in \mathbb{R}^d, W \in \mathbb{R}^{k \times d}$ and $\mathbf{b} \in \mathbb{R}^k$. Denote the i-th row of W by $\mathbf{w}_i$ and by $F_i(\mathbf{x}) = \langle \mathbf{w}_i, \mathbf{x} \rangle + b_i$ the i-th output of F for every $i \in [k]$. Let $\mathbf{x}_0 \in \mathbb{R}^d$ with $\arg\max_i F_i(\mathbf{x}_0) = \ell$ and denote by $\epsilon_i := \frac{\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle}{\|\mathbf{w}_i - \mathbf{w}_\ell\|^2}$. Then, the adversarial perturbation $\mathbf{z}$ that changes the label of $\mathbf{x}_0$ with the smallest L_2 norm is equal to $\mathbf{z} := \epsilon_j(\mathbf{w}_j - \mathbf{w}_\ell)$ for the target class $j := \arg\min_i \frac{\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle}{\|\mathbf{w}_i - \mathbf{w}_\ell\|}$.*

The full proof is deferred to Appendix B. The above lemma states that, for a data point $\mathbf{x}_0$ classified by the linear predictor as class ℓ, the class i with the smallest linear perturbation will minimize the term $\frac{\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle}{\|\mathbf{w}_i - \mathbf{w}_\ell\|}$. Note that the term $\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle$ represents the distance between the output of

the model on class ℓ and class i, which are the logits that are commonly used in selecting the targets for adversarial attacks. The lemma states that this term should be divided by $\|\mathbf{w}_i - \mathbf{w}_\ell\|$, which corresponds to the norm of the difference between the gradients.

There is an intuitive explanation as to why this targeting method is optimal for linear predictors. While the term $\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle$ represents the "distance" that the adversarial perturbation should move to change the prediction class, the term $\|\mathbf{w}_i - \mathbf{w}_\ell\|$ represents the "speed" at which this change happens. Thus, the fastest way to change the prediction class is to consider the target for which the ratio between the distance and the speed is the smallest.

3.2 Targeting method

We now formalize MALT, which is an adversarial attack that is applicable to any model, not only linear. The basic idea is to re-order the target classes using our targeting method, and then employ a fast targeted adversarial attack towards those targets.

Consider a classification model over k-classes (e.g., a neural network) $N : \mathbb{R}^d \to \mathbb{R}^k$. Suppose we are given a data point $\mathbf{x}_0 \in \mathbb{R}^d$ classified in class ℓ by N, meaning that $\ell = \arg\max_{j \in \{1, \dots, k\}} (N(\mathbf{x}_0))_j$. We look at the top $c \in \mathbb{N}$ ($c \leq k$) classes ordered by their output on $\mathbf{x}_0$, and for each such class j calculate a score of the form $\frac{(N(\mathbf{x}_0))_j - (N(\mathbf{x}_0))_\ell}{\|(\nabla N(\mathbf{x}_0))_j - (\nabla N(\mathbf{x}_0))_\ell\|}$. We now pick the top $a \in \mathbb{N}$ ($a \leq c \leq k$) classes re-ordered by our score and perform a targeted adversarial attack towards it. The two hyperparameters in our algorithm are c, which represents the number of candidates for which we calculate the score, and a, which represents the number of classes we perform a targeted attack towards. The full attack algorithm is presented in Algorithm 1.

Note that to calculate a score for some target class, we need to calculate the gradient of the model on $\mathbf{x}_0$ w.r.t this class. For datasets with many possible classes (e.g. ImageNet) we would like to limit these calculations. In all of our experiments, we used $c = 100$ and $a = 9$.

We note that our targeting method is compatible with any targeted attack. We empirically found that APGD (Croce and Hein [2020b]) with the default DLR loss performs well across datasets and models, improving the current state of the art – AutoAttack (Croce and Hein [2020b]). Full empirical results are in Section 5.

Algorithm 1 MALT attack algorithm

Input: Trained classification model $N : \mathbb{R}^d \to \mathbb{R}^k$, data point $\mathbf{x}_0 \in \mathbb{R}^d$, hyperparameters $a, c \in \mathbb{N}$.

set $\ell = \arg\max_{j \in \{1, \dots, k\}} (N(\mathbf{x}_0))_j$
Set CandidateClasses $= [0, \dots, 0]$, a list of size c
for top c classes $i \in \{1 \dots, k\} \setminus \{\ell\}$ ordered by $(N(\mathbf{x}_0))_i$ **do**
    Set AttackScore$_i = \frac{(N(\mathbf{x}_0))_i - (N(\mathbf{x}_0))_\ell}{\|(\nabla N(\mathbf{x}_0))_i - (\nabla N(\mathbf{x}_0))_\ell\|}$
    Append $(i, \text{AttackScore}_i)$ to the CandidateClasses list
end for
Sort CandidateClasses list by AttackScore
for class i in the top a classes of the CandidateClasses list **do**
    Run TargetedAttack$(N, \mathbf{x}_0, i)$
    if Adversarial perturbation found **then**
        return the adversarial perturbation
    end if
end for

3.3 Complexity analysis

We now calculate the time complexity of MALT with APGD, and compare it with the time complexity of AutoAttack. We calculate the complexity in terms of forward and backward passes (i.e., gradient calculations). We also consider the default hyperparameters of each adversarial attack, e.g., for APGD, we perform an attack with 100 iterations.

MALT with APGD. Calculating the targeting order takes c backward passes, since the corresponding row of the Jacobian is calculated. For the top a classes in this ordering, we perform an APGD attack which takes $a \cdot 100$ backward passes and $a \cdot 100$ forward passes.

AutoAttack. We calculate the complexity of each attack separately: (1) Untargeted APGD with CE loss takes 100 forward and 100 backward passes; (2) Targeted APGD with DLR loss attack the top 9 targets, which takes 900 forward and 900 backward passes; (3) Targeted FAB also attack the top 9 targets, runs for 100 iterations and takes one backward, two forward passes for each iteration, in addition to three forward passes for each target. In total, it takes 900 backward and 1827 forward passes; and (4) Untargeted Square black-box attack for 5000 steps, which takes 1 forward pass each for a total of 5000 forward passes.

Finally, we sum the total complexity, and use the hyperparameters $c = 100$, $a = 9$ for MALT (which are used in our experimental results). MALT takes 1000 backward and 900 forward passes, while AutoAttack takes 1900 backward and 7827 forward passes. Since each forward pass takes approximately the same time as a backward pass, we conclude that MALT requires 1900 passes, while AutoAttack takes 9727 passes, which is more than *five times* faster. In Section 5, we complement this analysis with experiments, showing that also in practice, MALT is on average five times faster than AutoAttack on the ImageNet test dataset.

4 Mesoscopic Almost Linearity in Neural Networks

In the previous section, we defined MALT, which relies on normalizing the output of the model by the magnitude of the gradients. This method was motivated by analyzing a linear predictor and finding the target class with the closest adversarial example. However, neural networks are highly non-linear, and the gradient can potentially change significantly when traversing the trajectory from a data point to an adversarial example. In this section, we claim that neural networks behave as though they are almost linear in the mesoscopic scale, in the sense that the norm of the gradient does not change very much when moving from a data point towards an adversarial example. This means that the target classes which MALT chooses to attack remain good target classes also when moving away from the attacked data point.

4.1 Almost Linearity for Data Residing on a Low-Dimensional Subspace

In this subsection, we prove theoretically that 2-layer neural networks are almost linear in the mesoscopic scale and in certain directions under the setting where the high-dimensional data lies on a low-dimensional manifold. This setting was studied in several works such as Fawzi et al. [2018], Khoury and Hadfield-Menell [2018], Shamir et al. [2021], Melamed et al. [2023]. In particular, we consider the setting from Melamed et al. [2023], where the authors analyzed two-layer networks, and the data lies on a low-dimensional linear subspace. All the proofs can be found in Appendix C.

Model. Our model is a two-layer fully-connected ReLU network $N : \mathbb{R}^d \to \mathbb{R}$ with input dimension d and hidden dimension m: $N(\mathbf{x}, \mathbf{w}_{1:m}) = \sum_{i=1}^m u_i \sigma(\mathbf{w}_i^\top \mathbf{x})$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation, and $\mathbf{w}_{1:m} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$. We additionally assume that σ is L-smooth and there exists $\beta > 0$ such that $\sigma'(z) \geq \beta$ for every $z \in \mathbb{R}$. An example of such an activation is a smoothed version of Leaky ReLU. We initialize the first layer using standard Kaiming initialization He et al. [2015], i.e. $\mathbf{w}_i \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d}I\right)$, and the output layer as $u_i \sim \mathcal{U}\left(\left\{\pm \frac{1}{\sqrt{m}}\right\}\right)$. Note that in standard Kaiming initialization, each u_i would be initialized normally with a standard deviation of $\frac{1}{\sqrt{m}}$. For ease of analysis, we fix these weights to their standard deviation and only randomize the sign (this was also done in Melamed et al. [2023], Bubeck et al. [2021]).

Data. Following Melamed et al. [2023], our main assumption on the data is that it lies on a low dimensional linear subspace. Namely, we consider a binary classification dataset $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_r) \in \mathbb{R}^d \times \{\pm 1\}$. We assume there exists a linear subspace P of dimension $\ell < d$ such that $\mathbf{x}_i \in P$ for every i.

Loss and training. Given a loss function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ (e.g. MSE, BCE), we consider the optimization problem: $\min_{\mathbf{w}_{1:m}} \sum_{i=1}^r L(y_i, N(\mathbf{x}_i, \mathbf{w}_{1:m}))$, which is optimized using standard gradient descent. For ease of analysis, we assume only the weights of the first layer are trained (i.e. the $\mathbf{w}_i$'s) while the weights of the second layer (i.e. the u_i's) are fixed at their initial values, this was

also done in Melamed et al. [2023]. Since we consider trained networks in our results, we will write $N(\mathbf{x})$, and omit the $\mathbf{w}_{1:m}$ as an input to the network.

Mesoscopic local linearity. Following Bubeck et al. [2021], we say that the network N is *mesoscopic locally linear* at a point $\mathbf{x}_0$ with $\|\mathbf{x}\| \leq \sqrt{d}$ if for every $\mathbf{v}$ with $\|\mathbf{v}\| = o(\sqrt{d})$ we have:

$$\|\nabla N(\mathbf{x}_0) - \nabla N(\mathbf{x}_0 + \mathbf{v})\| = o(\|\nabla N(\mathbf{x}_0)\|) . \tag{1}$$

We now show an upper bound on the l.h.s of Eq. (1) and a lower bound on the r.h.s of Eq. (1) when projecting the gradients on the orthogonal subspace on which the data lies on. In the following theorems, $\Pi_{P^\perp}$ means an orthogonal projection on the subspace $P^\perp$.

Theorem 4.1. *Suppose that the network $N(\mathbf{x}, \mathbf{w}_{1:m}) = \sum_{i=1}^m u_i \sigma(\mathbf{w}_i^\top \mathbf{x})$ is trained on a dataset which lies on a linear subspace P of dimension ℓ. Then, w.p $> 1 - \delta$ over the initialization for every $\mathbf{v} \in P^\perp$ with $\|\mathbf{v}\| \leq R$ and every $\mathbf{x} \in P$ we have that:*

$$\|\Pi_{P^\perp} (\nabla N(\mathbf{x}) - \nabla N(\mathbf{x} + \mathbf{v}))\| \leq 20LR \left(\sqrt{\frac{\log\left(\frac{m}{\delta}\right)}{d - \ell}} + \frac{\log\left(\frac{1}{\delta}\right)}{m} \right)$$

Theorem 4.2. *Under the same assumptions as in Theorem 4.1 and that $d \geq 2 \log\left(\frac{1}{\delta}\right)$, w.p $> 1 - \delta$ over the initialization we have that:*

$$\|\Pi_{P^\perp} (\nabla_{\mathbf{x}} N(\mathbf{x}))\| \geq \beta \cdot \sqrt{1 - 2\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{d}}}$$

Using both theorems, we can prove that the network is mesoscopic almost linear in directions orthogonal to the data subspace:

Corollary 4.1. *Suppose that the network $N(\mathbf{x}) = \sum_{i=1}^m u_i \sigma(\mathbf{w}_i^\top \mathbf{x})$ is trained on a dataset which lies on a linear subspace P of dimensions ℓ. Let $\mathbf{v} \in P^\perp$ with $\|\mathbf{v}\| \leq R$ and $\mathbf{x} \in P$. Assume that $\beta = \Omega(1)$, $d - \ell = \Omega(d)$, $R = o(\sqrt{d - \ell})$ and $m = \Omega(d - \ell)$, then we have that:*

$$\|\Pi_{P^\perp}(\nabla N(\mathbf{x}_0) - \nabla N(\mathbf{x}_0 + \mathbf{v}))\| = o(\|\Pi_{P^\perp}(\nabla N(\mathbf{x}_0))\|) .$$

The reason that the directions orthogonal to the data subspace are interesting, is because it is proven in previous works that in those directions, there exist adversarial perturbations, see for example Melamed et al. [2023], Haldar et al. [2024]. This means that if we consider an adversarial perturbation in those directions, the ordering of the target classes from our targeting method should remain approximately the same throughout the trajectory of the perturbation. This is because the norm of the difference between the gradients is not too big, at least compared to the norm of the gradient itself.

Remark 4.1 (Assumption in the theoretical part). *For the formal statements to work, we make several simplifying assumptions. In Appendix C.4, we provide further explanation on the necessity of those assumptions and whether they could be relaxed. In a nutshell, most assumption (especially the ones on the activation) are made since we consider a very general setting, where our only assumption on the data is that it lies on a low dimensional manifold, note that we don't assume a more intricate structure or limit the number of training points. By introducing more assumptions on the data, we could relax our other assumptions.*

4.2 Empirical local linearity

In this section, we study mesoscopic almost linearity empirically using state of the art robust networks from RobustBench (Croce et al. [2020]), and for both the CIFAR100 and ImageNet datasets. In the first experiment, for each image $\mathbf{x}_0$ in the test dataset, we consider an ϵ-norm ball in L_∞ around it (with $\epsilon = 8/255$ and $4/255$ for CIFAR100 and ImageNet respectively). Our goal is to study how the gradient changes when moving from $\mathbf{x}_0$ to a $\mathbf{x}_0 + \mathbf{v}$, where $\mathbf{v}$ is some direction inside the ϵ-ball.

We use two strategies to choose the direction $\mathbf{v}$. The first is drawing a random direction for each image, and the second is the direction of the gradient at each image, which corresponds to an adversarial direction. After choosing $\mathbf{v}$ we divide it into 100 equal parts, namely $\mathbf{v}_1 = \frac{1}{100}\mathbf{v}, \mathbf{v}_2 = \frac{2}{100}\mathbf{v}, \ldots, \mathbf{v}_{100} = \mathbf{v}$. We will consider two measures to study the mesoscopic linearity:

$$\alpha = \frac{\|\nabla N(\mathbf{x}_0) - \nabla N(\mathbf{x}_0 + \mathbf{v}_i)\|}{\|\nabla N(\mathbf{x}_0)\|}, \quad \alpha_{\text{part}} = \frac{\|\nabla N(\mathbf{x}_0 + \mathbf{v}_{i+1}) - \nabla N(\mathbf{x}_0 + \mathbf{v}_i)\|}{\|\nabla N(\mathbf{x}_0)\|}.$$

Namely, α corresponds to Eq. (1) and measures the total change in gradient norm from $\mathbf{x}_0$ until $\mathbf{v}_i$, while α_{part} measure this change but only for consecutive steps. In Figure 2a and Figure 2b we plot α and α_{part} for all $i \in [100]$ for both adversarial and random step $\mathbf{v}$, respectively, for the ImageNet Swin-L [Liu et al., 2023] classifier and the CIFAR100 WRN-28-10 [Wang et al., 2023] classifier. The experiment details are in Appendix D.1.
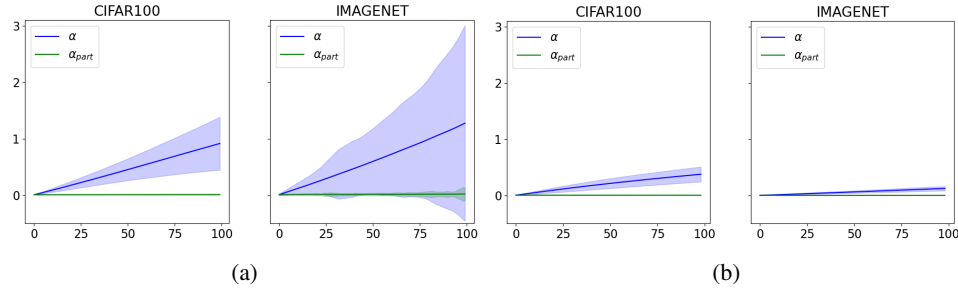

Figure 2: Measurement of **mesoscopic almost linearity** experimentally when taking a step $\mathbf{v}$ away from test image x_0 for CIFAR100 and ImageNet. The results are averaged over all the images in the test set, where (a) random step; and (b) Direction of the gradient (adversarial step).

It can be seen that for random directions (Figure 2b), both α and α_{part} are very small for both datasets, and with a very small variance. For adversarial directions (Figure 2a), α_{part} is still very small, while α is larger since it accumulates small deviations from linearity when moving further away from $\mathbf{x}_0$. We emphasize that by Eq. (1), mesoscopic almost linearity means that $\alpha = o_d(1)$, where d is the dimension of the input. In other words, a larger input dimension should decrease, or at least not increase, the value of α. In this experiment, the CIFAR100 and ImageNet datasets have very different input dimensions, namely, $d = 3072$ and $d = 150528$. We see that although the input dimension is almost 50 times larger for ImageNet compared to CIFAR100, the average value of α is nearly the same. This implies that mesoscopic almost linearity also happens in adversarial directions, and it would be interesting to further study how α changes when having even more drastic changes in the input dimension.

In our second experiment, we take a small adversarial step and compare the change of the logits of the output of the network and its linear approximation. In Figure 3, for each image $\mathbf{x}_0$, we find an adversarial direction $\mathbf{v}$, which we split into 100 equal parts similarly to the previous experiment. For a network N we calculate its linear approximation L at $\mathbf{x}_0$ using Taylor's expansion, and compare $N(\mathbf{x}_0 + \mathbf{v}_i)$ to $L(\mathbf{x}_0 + \mathbf{v}_i)$. Note that since L is a linear function, $L(\mathbf{x}_0 + \mathbf{v}_i)$ is always a straight line when plotting the output for every $i = 1, \ldots, 100$. It can be seen that although $N(\mathbf{x}_0 + \mathbf{v}_i)$ is not necessarily linear, it does closely resemble the linear approximation, which suggests that almost linearity happens in the mesoscopic scale. We note that the logits from Figure 1b look less linear than in the figures below. We conjecture that this is because it is difficult to find an adversarial example for this image (and indeed, AutoAttack fails to do so). Hence, the network is less linear in those adversarial directions, and such examples are what causes the high variance in Figure 2a. This may require additional investigation, which we leave for future research.

5 Experiments

In this section, we compare the MALT attack (using a standard APGD attack) to the current state of the art AutoAttack. Our comparison is made to be compatible with RobustBench (Croce et al. [2020]), which is the standard benchmark for testing adversarial robustness. Namely, we consider attack on CIFAR-100 (Krizhevsky et al. [2009]) with an ℓ_∞ budget of $\epsilon = 8/255$ and on ImageNet (Deng et al. [2009]) with an ℓ_∞ budget of $\epsilon = 4/255$. We compare the two attacks on several robust models from the top RobustBench benchmark.

For MALT, we consider calculating the score for the $c = 100$ classes with the highest model's confidence and attacking the top $a = 9$ classes according to this score. This corresponds to the
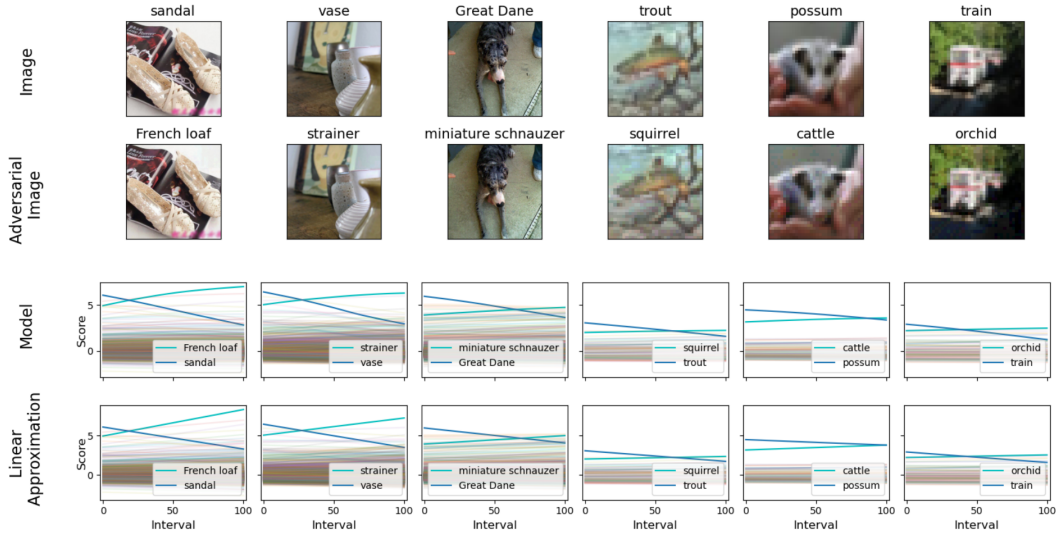
Figure 3: Empirical **mesoscopic almost linearity**: demonstrating the logits changes from an image x_0 to its adversarial example. In the third row, we plot the model's output logits changes, and in the bottom row are the results of the linear approximation of the model at x_0.

targeted attacks in AutoAttack, which attempt to attack the top 9 classes according to the model's confidence. All the hyperparameters of APGD and the other attacks used in AutoAttack are set to their default values. Full details for all experiments in this section are in Appendix D.

In Table 1 (CIFAR-100) and Table 2 (ImageNet), we present the results of the experiments. Note that MALT is able to attack more images than AutoAttack across all robust models and for both datasets. Also, there is an inclusion of successful attacks in the sense that for every image that AutoAttack successfully attacks, MALT also succeeds. Thus, the improvement contains only images that AutoAttack fails on. Note that in Table 2, the robust and clean accuracy is not exactly equal to those reported on the RobustBench website since newer versions of the Python libraries in use give slightly different results.

Table 1: **CIFAR100 -** L_∞ robust accuracy *(lower is better)*, comparing MALT and AutoAttack, which is the current state of the art.

MODEL		ROBUSTNESS			
	ACC.	MALT	SOTA	DIFF	SPEED-UP
WRN-28-10 [WANG ET AL., 2023]	72.58%	38.79%	38.83%	−0.04%	× 3.36 ±0.18
WRN-70-16 [WANG ET AL., 2023]	75.22%	42.66%	42.67%	−0.01%	× 3.87 ±0.08
WRN-28-10 [CUI ET AL., 2023]	73.83%	39.18%	39.18%	0%	× 3.43 ±0.08
WRN-70-16 [GOWAL ET AL., 2020]	69.15%	36.81%	36.88%	−0.07%	× 3.42 ±0.09

Table 2: **ImageNet -** L_∞ robust accuracy *(lower is better)*, comparing MALT and AutoAttack, which is the current state of the art.

MODEL		ROBUSTNESS			
	ACC.	MALT	SOTA	DIFF.	SPEED-UP
SWIN-L [LIU ET AL., 2023]	79.18%	59.84%	59.90%	−0.06%	× 5.18 ±0.04
CONVNEXT-L [LIU ET AL., 2023]	78.20%	58.82%	58.88%	−0.06%	× 5.22 ±0.1
CONVNEXT-L+ [SINGH ET AL., 2024]	77.02%	57.94%	57.96%	−0.02%	× 4.86 ±0.06
SWIN-B [LIU ET AL., 2023]	76.22%	56.54%	56.56%	−0.02%	× 5.02 ±0.03
CONVNEXT-B+ [SINGH ET AL., 2024]	76.00%	56.48%	56.52%	−0.04%	× 5.00 ±0.07

Attack Running Time For the speed-up column, we first sampled uniformly from the test dataset five batches of size 200 for ImageNet and 400 for CIFAR100, which corresponds to $5 \times 4\%$ of the entire test set. We timed both MALT and AutoAttack, when running on the exact same GPU chip and on the same samples, we present the mean and variance of these experiments for each robust model. On the ImageNet dataset there is on average a five times speed up, while for CIFAR100 it is 3.5 times on average. The lower rate can be explained by the higher attack success rates for CIFAR100. In other words, the complexity analysis from Section 3 is of the worst case, and since fewer test examples pass through all four different attacks in AutoAttack, the running time is lower.

Integrating MALT with different attacks Our experiments found that MALT operates well across datasets and models with the standard and fast APGD attack. We also integrated MALT with the targeted FAB attack (Croce and Hein [2020a]) as well as with the APGD attack with the CE loss, on the Swin-L robust model (Liu et al. [2023]). Both attacks achieved a worse results of 60.64% for the FAB attack and 60.52% for the APGD attack with CE loss, compared to the superior 59.84% robust accuracy using APGD with the DLR loss.

5.1 Targeting analysis

In this subsection, we analyze the targeting mechanism of MALT and whether other targeting techniques could have improved it. In Figure 4, we show how the successful attacks are distributed according to the score given either by MALT or the naive targeting (i.e., according to the model's confidence). It is evident that the score provided by MALT leans much more towards the top -1 than naive targeting. This indicates that the score provided by MALT has a higher correlation with successful attacks than the naive targeting method. Also, since MALT successfully attacks more images for the top 9 targets (achieving robust accuracy of 59.84%, compared to 59.94% for naive targeting), the sum over the columns of MALT attacks is larger than that of the naive targeting.

Finally, we test whether the choice of $c = 100$ is enough, namely calculating the score for MALT only for the top 100 classes sorted by the confidence level of the model. To this end, we ran MALT with $c = 1000$ for the ImageNet test dataset on the Swin-L robust network. Note that the number of gradient calculations for each test image has increased by a factor of more than two. This attack reached a robust accuracy of 59.84%, *exactly the same as* $c = 100$. This means that calculating the score for the top 100 classes is enough to find the top classes to attack. It may
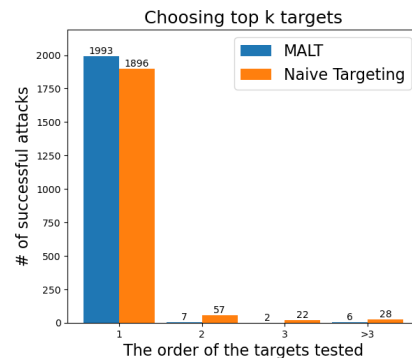

Figure 4: Comparing targeting methods for Liu et al. [2023] SOTA model: The number of successful attacks for each target order by two targeting methods: In blue, we use MALT targeting and APGD, and in orange, we compare to APGD with top logits targeting performed in AutoAttack.

be possible to optimize this hyperparameter even more, although the benefit in running time will be negligible.

6 Conclusions

In this paper we present MALT, an adversarial attack which is based on a targeting method that assumes almost linearity in the mesoscopic scale. MALT wins over the current state of the art adversarial attack AutoAttack on several robust models, and for both CIFAR100 and ImageNet, while also speeding up the runtime by more than *five times*. We also present theoretical and empirical evidence that our almost linearity assumption is applied to neural network, in the mesoscopic scale where adversarial examples exist.

There are several future research directions that we think are interesting. First, it would be interesting to further study the mesoscopic almost linearity property of neural networks. In particular, whether this property is affected by (or affecting) the robustness of the network to adversarial perturbations, and the perturbation's transferability capabilities. Theoretically, it would be interesting to extend our

analysis to deeper networks. As for the applied results, a good future direction would be to conduct a more extensive empirical study of MALT on more robust models and datasets, even beyond the scope of the RobustBench benchmark. Finally, it would be interesting to understand whether there exists a better targeting method to find adversarial classes, e.g., using a second-order approximation of the network.

References

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

P. Bartlett, S. Bubeck, and Y. Cherapanamjeri. Adversarial examples in multi-layer random relu networks. *Advances in Neural Information Processing Systems*, 34, 2021.

S. Bubeck and M. Sellke. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34:28811–28822, 2021.

S. Bubeck, Y. T. Lee, E. Price, and I. Razenshteyn. Adversarial examples from computational constraints. In *International Conference on Machine Learning*, pages 831–840. PMLR, 2019.

S. Bubeck, Y. Cherapanamjeri, G. Gidel, and R. T. d. Combes. A single gradient step finds adversarial examples on random two-layers neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

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

F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020b.

F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

J. Cui, Z. Tian, Z. Zhong, X. Qi, B. Yu, and H. Zhang. Decoupled kullback-leibler divergence loss. *arXiv preprint arXiv:2305.13948*, 2023.

A. Daniely and H. Shacham. Most relu networks suffer from ℓ_2 adversarial perturbations. *Advances in Neural Information Processing Systems*, 33, 2020.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

A. Fawzi, H. Fawzi, and O. Fawzi. Adversarial vulnerability for any classifier. *Advances in neural information processing systems*, 31, 2018.

S. Frei, G. Vardi, P. Bartlett, and N. Srebro. The double-edged sword of implicit bias: Generalization vs. robustness in relu networks. *Advances in Neural Information Processing Systems*, 36, 2024.

J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *Preprint, arXiv:1412.6572*, 2014.

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

Y. Guo, Q. Li, and H. Chen. Backpropagating linearly improves transferability of adversarial examples. *Advances in neural information processing systems*, 33:85–95, 2020.

R. Haldar, Y. Xing, and Q. Song. Effect of ambient-intrinsic dimension gap on adversarial vulnerability. In *International Conference on Artificial Intelligence and Statistics*, pages 1090–1098. PMLR, 2024.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

M. Khoury and D. Hadfield-Menell. On the geometry of adversarial examples. *Preprint, arXiv:1811.00525*, 2018.

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

B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28(5):1302 – 1338, 2000. doi: 10.1214/aos/1015957395. URL https://doi.org/10.1214/aos/1015957395.

C. Liu, Y. Dong, W. Xiang, X. Yang, H. Su, J. Zhu, Y. Chen, Y. He, H. Xue, and S. Zheng. A comprehensive study on robustness of image classification models: Benchmarking and rethinking. *arXiv preprint arXiv:2302.14301*, 2023.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *Preprint, arXiv:1706.06083*, 2017.

O. Melamed, G. Yehudai, and G. Vardi. Adversarial examples exist in two-layer relu networks for low dimensional data manifolds. *arXiv preprint arXiv:2303.00783*, 2023.

A. Montanari and Y. Wu. Adversarial examples in random neural networks with general activations. *arXiv preprint arXiv:2203.17209*, 2022.

S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

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

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli. Adversarial robustness through local linearization. *Advances in neural information processing systems*, 32, 2019.

A. Sarkar and R. Iyengar. Enforcing linearity in dnn succours robustness and adversarial image generation. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part I 29*, pages 52–64. Springer, 2020.

A. Shamir, I. Safran, E. Ronen, and O. Dunkelman. A simple explanation for the existence of adversarial examples with small hamming distance. *Preprint, arXiv:1901.10861*, 2019.

A. Shamir, O. Melamed, and O. BenShmuel. The dimpled manifold model of adversarial examples in machine learning. *Preprint, arXiv:2106.10151*, 2021.

N. D. Singh, F. Croce, and M. Hein. Revisiting adversarial training for imagenet: Architectures, training and generalization across threat models. *Advances in Neural Information Processing Systems*, 36, 2024.

Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

D. Stutz, M. Hein, and B. Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *Preprint, arXiv:1312.6199*, 2013.

T. Tanay and L. Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

J. Uesato, B. O'donoghue, P. Kohli, and A. Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*, pages 5025–5034. PMLR, 2018.

G. Vardi, G. Yehudai, and O. Shamir. Gradient methods provably converge to non-robust networks. *Preprint, arXiv:2202.04347*, 2022.

Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan. Better diffusion models further improve adversarial training. In *International Conference on Machine Learning*, pages 36246–36263. PMLR, 2023.

A   Additional MALT Targeting Examples

Following Section 3, to further present the performance of the MALT targeting method, we add more examples of images on which AutoAttack fails and MALT succeeds. We present such images from the CIFAR100 and the ImageNet datasets in Figure 6 and Figure 5 respectively. As in the examples found in Section 3, Figure 1, we present two attacks on each image - the naively targeted APGD and the MALT attack using APGD.

MALT and APGD   For all experiments, for the MALT attack, we used $a = 9$ and $c = 100$, and the default hyperparameters of the attack APGD adapted from the official AutoAttack git repository.[4] Note that we used the APGD attack with the given default DLR loss.

In the upper row, for each image, one can see a failed APGD attack toward the class with the model's second-best confidence level. On the bottom row, we present our successful attack - a successful APGD attack toward a target found using the MALT method. The original image is on the left, and the perturbed image is on the right. Between them, we present the change in the model's output logits, starting from the original image and ending in the perturbed one.

Note that in the CIFAR100 dataset experiments, the APGD attack gets random errors due to randomization. We count both here in Figure 6 and in Section 5 only images that MALT targets to class **not within** the naive top nine logits, therefore a clear advantage to MALT, regardless of these randomization-resulted errors.
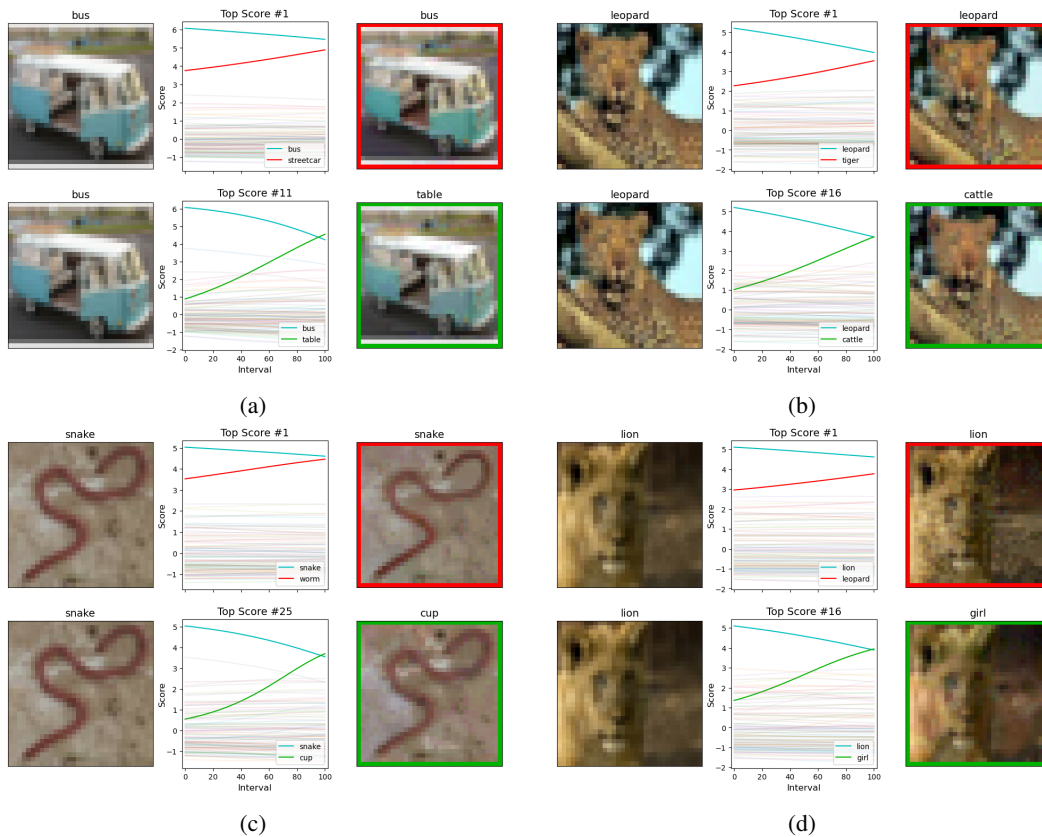

(a)

(b)

(c)

(d)

---

[4] https://github.com/fra31/auto-attack
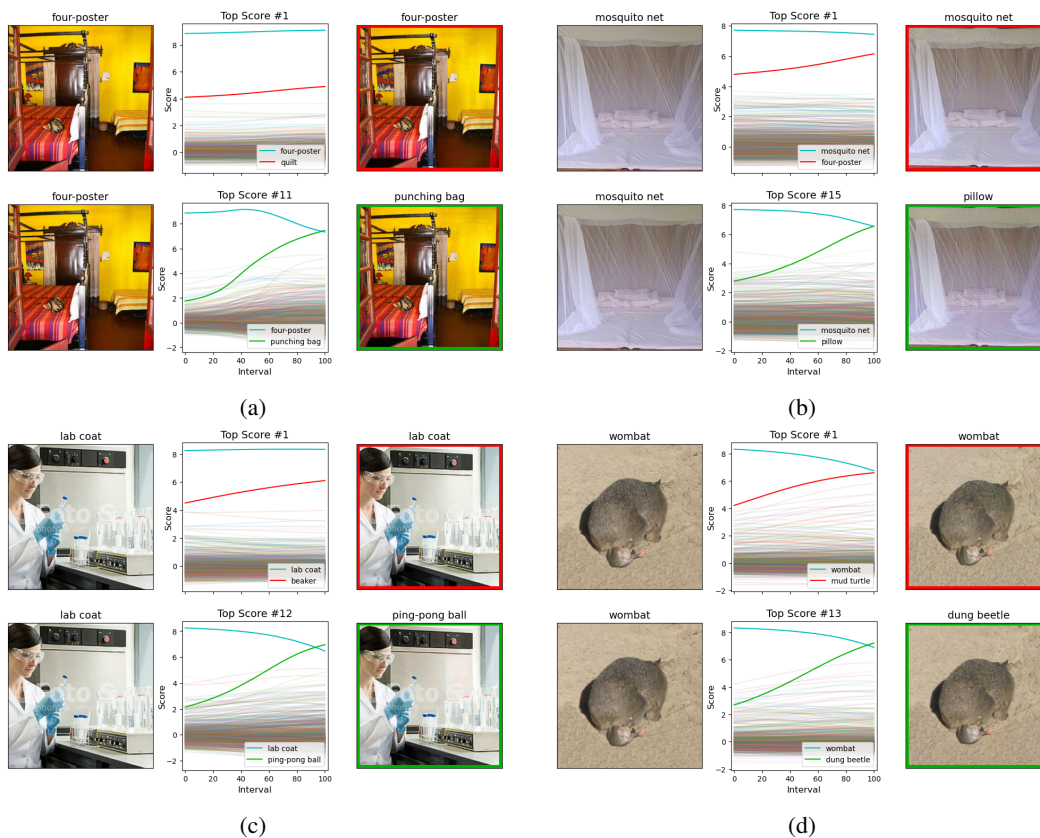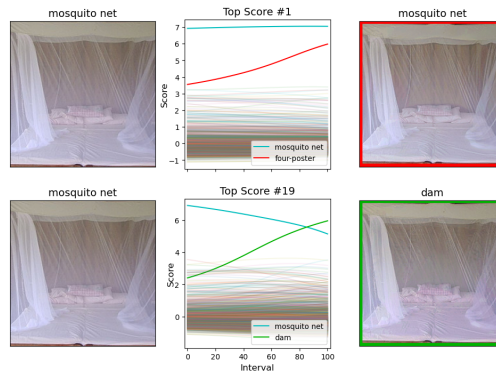
(e)

Figure 5: Additional examples of images from the ImageNet dataset that AutoAttack fails to attack while MALT succeeds. The top row shows an APGD attack on the target class with the highest logit, and the bottom row shows an APGD attack on the class that MALT finds and succeeds. (a) and (b) examples from Swin-L [Liu et al., 2023] network. (c) through (e) are from ConvNext-L [Liu et al., 2023] network. The images are shown before and after the attack, and the change in logits is presented in the middle column.

(a)

(b)

(c)
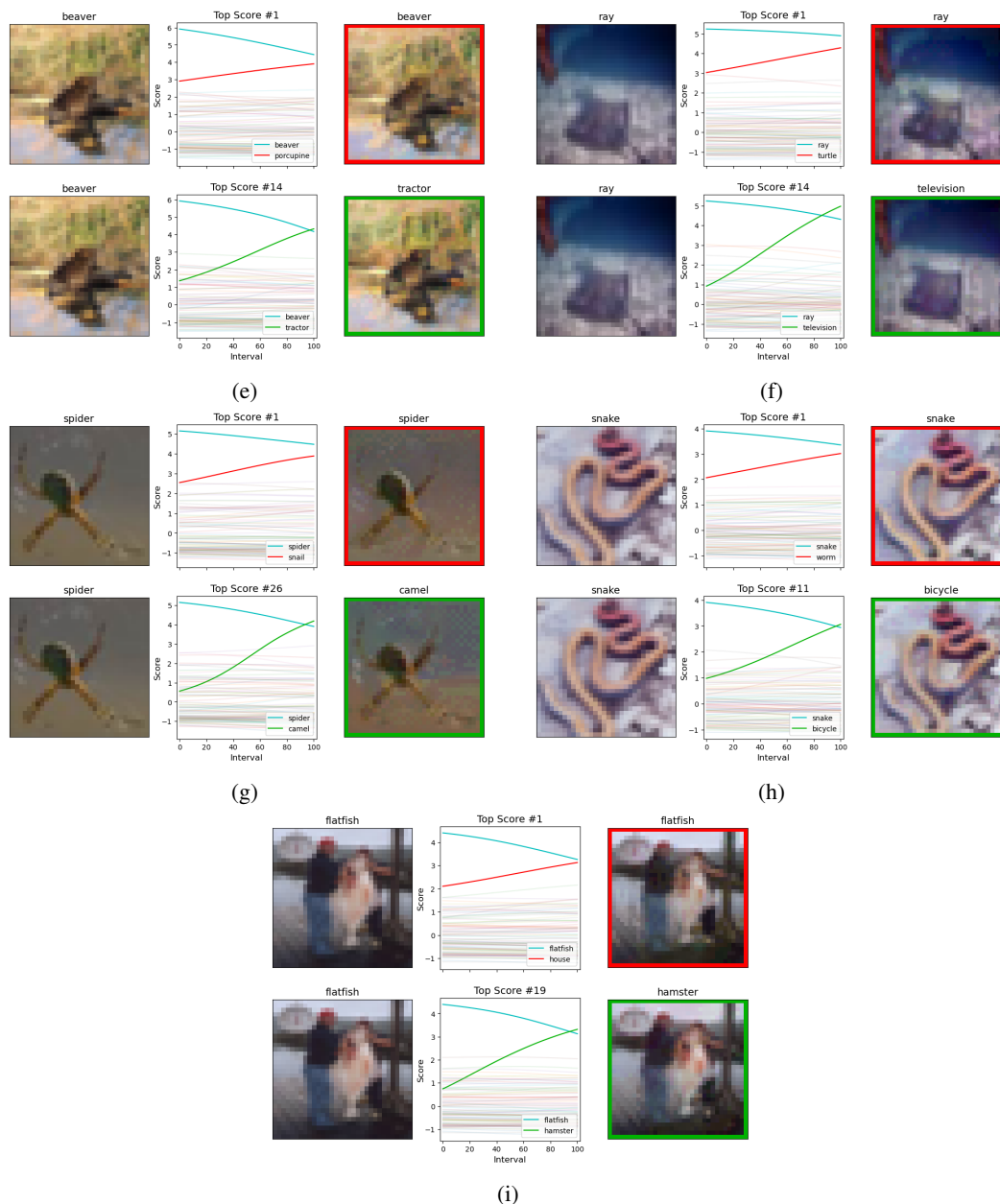

(d)

(e)

(f)

(g)

(h)

(i)

Figure 6: Additional examples of images from the CIFAR100 dataset that AutoAttack fails to attack while MALT succeeds. The top row shows an APGD attack on the target class with the highest logit, and the bottom row shows an APGD attack on the class that MALT finds and succeeds. (a) through (g) examples from WRN-70-16 [Gowal et al., 2020] network. (h) is from WRN-28-10 [Wang et al., 2023] network. (i) is from WRN-70-16 [Wang et al., 2023] network. The images are shown before and after the attack, and the change in logits is presented in the middle column.

B   Proofs from Section 3

Proof of Lemma 3.1. Consider finding the smallest adversarial perturbation $\mathbf{z}_i$ for $\mathbf{x}_0$ where the i-th output of F is larger than the ℓ-th output of F. That is, we want to find:

$$\min_{\mathbf{z}} \|\mathbf{z}\|^2 \text{ s.t. } F_i(\mathbf{x}_0 + \mathbf{z}_i) - F_\ell(\mathbf{x}_0 + \mathbf{z}_i) > 0 \tag{2}$$

We have that: $F_i(\mathbf{x}_0 + \mathbf{z}_i) - F_\ell(\mathbf{x}_0 + \mathbf{z}_i) = \langle \mathbf{w}_i, \mathbf{x}_0 + \mathbf{z}_i \rangle - \langle \mathbf{w}_\ell, \mathbf{x}_0 + \mathbf{z}_i \rangle$. Hence, we get that Eq. (2) is a constraint minimization problem with a single linear constraint. The solution to this problem is given by the direction $\mathbf{z}_i \propto \mathbf{w}_i - \mathbf{w}_\ell$. We can write $\mathbf{z}_i = \epsilon_i(\mathbf{w}_i - \mathbf{w}_\ell)$ for $\epsilon_i \in \mathbb{R}$ which will be determined later, then we have that:

$$\begin{aligned}
F_i(\mathbf{x}_0 + \mathbf{z}_i) - F_\ell(\mathbf{x}_0 + \mathbf{z}_i) &= \langle \mathbf{w}_i, \mathbf{x}_0 + \mathbf{z}_i \rangle - \langle \mathbf{w}_\ell, \mathbf{x}_0 + \mathbf{z}_i \rangle \\
&= \langle \mathbf{w}_i, \mathbf{x}_0 + \epsilon_i(\mathbf{w}_i - \mathbf{w}_\ell) \rangle - \langle \mathbf{w}_\ell, \mathbf{x}_0 + \epsilon_i(\mathbf{w}_i - \mathbf{w}_\ell) \rangle \\
&= \langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle + \epsilon_i \|\mathbf{w}_i - \mathbf{w}_\ell\|^2 .
\end{aligned}$$

That is, for $\epsilon_i = \frac{\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle}{\|\mathbf{w}_i - \mathbf{w}_\ell\|^2}$ and $\mathbf{z}_i = \epsilon_i(\mathbf{w}_i - \mathbf{w}_\ell)$ we have that $F_i(\mathbf{x}_0 + \mathbf{z}_i) = F_\ell(\mathbf{x}_0 + \mathbf{z}_i)$. Note that $\|\mathbf{z}_i\| = \frac{\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle}{\|\mathbf{w}_i - \mathbf{w}_\ell\|}$, we omit the absolute value on the nominator since the constraint in Eq. (2) is that it is positive. Finally, finding $\arg\min_i \|\mathbf{z}_i\| = \arg\min_i \frac{\langle \mathbf{w}_i - \mathbf{w}_\ell, \mathbf{x}_0 \rangle}{\|\mathbf{w}_i - \mathbf{w}_\ell\|}$ provides the class index with the smallest norm adversarial perturbation. $\qquad\square$

C   Proofs from Section 4

C.1   Proof of Theorem 4.1

Proof. Using Lemma C.1 we can assume w.l.o.g that $P = \text{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_{d-\ell}\}$. Denote by $\hat{\mathbf{w}} := \Pi_{P^\perp}(\mathbf{w})$ the projection of $\mathbf{w}$ on the subspace $P^\perp$. By Lemma C.2 we get that the weights $\hat{\mathbf{w}}_i$ did not change from their initial value for any $i \in \{1, \ldots, m\}$.

We will now calculate the gradient of the network w.r.t the data. For ease of notations we drop the $\mathbf{w}_{1:m}$ as the input of the network.

$$\nabla_{\mathbf{x}} N(\mathbf{x}) = \sum_{i=1}^m u_i \mathbf{w}_i \sigma'(\mathbf{w}_i^\top \mathbf{x})$$

Thus, we want to bound the following:

$$\left\| \Pi_{P^\perp} \left(\sum_{i=1}^m u_i \mathbf{w}_i \sigma'(\mathbf{w}_i^\top \mathbf{x}) - \sum_{i=1}^m u_i \mathbf{w}_i \sigma'(\mathbf{w}_i^\top (\mathbf{x} + \mathbf{v})) \right) \right\|$$

$$\leq \left\| \sum_{i=1}^m u_i \hat{\mathbf{w}}_i \left(\sigma'(\mathbf{w}_i^\top \mathbf{x}) - \sigma'(\mathbf{w}_i^\top \mathbf{x} + \mathbf{v}) \right) \right\|$$

Using that $\|\mathbf{w}\| = \sup_{\mathbf{u} \in \mathbb{S}^{d-1}} \mathbf{w}^\top \mathbf{u}$, it is enough to bound the following for any $\mathbf{u} \in \mathbb{S}^{d-1}$:

$$\sum_{i=1}^m u_i \hat{\mathbf{w}}_i^\top \mathbf{u} \left(\sigma'(\mathbf{w}_i^\top \mathbf{x}) - \sigma'(\mathbf{w}_i^\top \mathbf{x} + \mathbf{v}) \right) . \tag{3}$$

We will now use Bernstein inequality on the above sum. Denote $X_i = u_i \hat{\mathbf{w}}_i^\top \mathbf{u} \left(\sigma'(\mathbf{w}_i^\top \mathbf{x}) - \sigma'(\mathbf{w}_i^\top \mathbf{x} + \mathbf{v}) \right)$, then we have that:

$$\begin{aligned}
\mathbb{E}[|X_i|^q] &\leq \frac{L^q}{m^{q/2}} \mathbb{E}\left[|\hat{\mathbf{w}}^\top \mathbf{u}| \cdot |\hat{\mathbf{w}}^\top \mathbf{v}| \right] = \frac{L^q}{m^{q/2}} \sqrt{\mathbb{E}[|\hat{\mathbf{w}}^\top \mathbf{u}|]^{2q} \cdot \mathbb{E}[|\hat{\mathbf{w}}^\top \mathbf{v}|]^{2q}} \\
&\leq \frac{(LR)^q}{(d-\ell)^q m^{q/2}} \mathbb{E}_{Y \sim N(0,1)}[|Y|^{2q}] \leq \frac{(LR)^q}{(d-\ell)^q m^{q/2}} (2q-1)!! \leq \frac{q!}{2} \cdot \frac{(LR)^q}{(d-\ell)^q m^{q/2}} ,
\end{aligned}$$

where we used that σ is L-smooth, and the assumption on the initialization of $\mathbf{w}_i$ and u_i. Note that the above is true for every $\mathbf{u} \in \mathbb{S}^{d-1}$, hence by Bernstein inequality with $c = \frac{LR}{(d-\ell)\sqrt{m}}$ we have w.p $> 1 - \delta$ over the initialization:

$$\left\| \sum_{i=1}^{m} u_i \hat{\mathbf{w}}_i \left(\sigma'(\mathbf{w}_i^\top \mathbf{x}) - \sigma'(\mathbf{w}_i^\top \mathbf{x} + \mathbf{v}) \right) \right\| \leq \frac{LR}{d-\ell} \sqrt{\log\left(\frac{1}{\delta}\right)} \left(1 + \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{m}} \right) . \quad (4)$$

Denote by $\Omega := \{(\mathbf{u}, \mathbf{v}) : \mathbf{u} \in \mathbb{S}^{d-1}, \|\mathbf{v}\| \leq R, \mathbf{v}, \mathbf{u} \in P^\perp, \}$, we will now bound Eq. (3) uniformly over Ω. Denote by $\Phi(\mathbf{u}, \mathbf{v}) := \sum_{i=1}^{m} u_i \hat{\mathbf{w}}_i^\top \mathbf{u} \left(\sigma'(\mathbf{w}_i^\top \mathbf{x}) - \sigma'(\mathbf{w}_i^\top \mathbf{x} + \mathbf{v}) \right)$. We define an ϵ-net over Ω for ϵ to be chosen later, denote this net by M_ϵ. The size of the M_ϵ is at most $\left(\frac{10R}{\epsilon}\right)^{d-\ell}$, and we have that:

$$\sup_{(\mathbf{u},\mathbf{v})\in\Omega} \Phi(\mathbf{u},\mathbf{v}) \leq \sup_{(\mathbf{u},\mathbf{v})\in M_\epsilon} \Phi(\mathbf{u},\mathbf{v}) + \sup_{(\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}')\in\Omega,\|\mathbf{u}+\mathbf{u}'\|+\|\mathbf{v}-\mathbf{v}'\|\leq\epsilon} |\Phi(\mathbf{u},\mathbf{v}) - \Phi(\mathbf{u}',\mathbf{v}')| . \quad (5)$$

To bound the first term of Eq. (5) we use a union bound over all of M_ϵ, and by Eq. (4) we have w.p $> 1 - \delta$:

$$\sup_{(\mathbf{u},\mathbf{v})\in M_\epsilon} \Phi(\mathbf{u},\mathbf{v}) \leq \frac{LR}{d-\ell} \sqrt{(d-\ell)\log\left(\frac{1}{\epsilon}\right) + \log\left(\frac{1}{\delta}\right)} \left(1 + \sqrt{\frac{(d-\ell)\log\left(\frac{1}{\epsilon}\right) + \log\left(\frac{1}{\delta}\right)}{m}} \right) . \quad (6)$$

For the second term in Eq. (5), note that for any $\mathbf{u}, \mathbf{u}', \mathbf{v}, \mathbf{v}'$:

$$|\Phi(\mathbf{u},\mathbf{v}) - \Phi(\mathbf{u}',\mathbf{v})| \leq \frac{L\|\mathbf{u}-\mathbf{u}'\|}{\sqrt{m}} \sum_{i=1}^{m} \|\hat{\mathbf{w}}_i\|^2$$

$$|\Phi(\mathbf{u},\mathbf{v}) - \Phi(\mathbf{u},\mathbf{v}')| \leq \frac{LR\|\mathbf{v}-\mathbf{v}'\|}{\sqrt{m}} \sum_{i=1}^{m} \|\hat{\mathbf{w}}_i\|^2$$

Note that $\sum_{i=1}^{m} \|\hat{\mathbf{w}}_i\|^2$ is a scaled chi-squared distribution with $m \cdot d$ degrees of freedom. By Lemma 1 in Laurent and Massart [2000] we have w.p $> 1 - \delta$ that:

$$\sum_{i=1}^{m} \|\hat{\mathbf{w}}_i\|^2 \leq m + 2\sqrt{\frac{m\log\left(\frac{1}{\delta}\right)}{d-\ell}} + \frac{2\log\left(\frac{1}{\delta}\right)}{d-\ell} .$$

Combining the above, and taking $\epsilon = \frac{1}{m}$ we get:

$$\sup_{(\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}')\in\Omega,\|\mathbf{u}+\mathbf{u}'\|+\|\mathbf{v}-\mathbf{v}'\|\leq\frac{1}{m}} |\Phi(\mathbf{u},\mathbf{v}) - \Phi(\mathbf{u}',\mathbf{v}')| \leq LR \left(\frac{1}{\sqrt{m}} + 2\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{d-\ell}} + \frac{2\log\left(\frac{1}{\delta}\right)}{(d-\ell)\sqrt{m}} \right)$$

Plugging this into Eq. (5), while choosing $\delta' = \frac{\delta}{2}$ finishes the proof $\qquad \square$

C.2 Proof of Theorem 4.2

Proof. Denote by $\hat{\mathbf{w}} := \Pi_{P^\perp}(\mathbf{w})$ the projection of $\mathbf{w}$ on the subspace $P^\perp$. We have that:

$$\|\Pi_{P^\perp}(\nabla_{\mathbf{x}} N(\mathbf{x}))\| = \left\| \sum_{i=1}^{m} u_i \hat{\mathbf{w}}_i \sigma'(\mathbf{w}_i^\top \mathbf{x}) \right\| \geq \frac{\beta}{\sqrt{m}} \left\| \sum_{i=1}^{m} \hat{\mathbf{w}}_i \right\| .$$

Using Lemma C.1 we can assume w.l.o.g that $P = \text{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_{d-\ell}\}$. By Lemma C.2 we get that the weights $\hat{\mathbf{w}}_i$ did not change from their initial value for any $i \in \{1, \ldots, m\}$. Hence, $\hat{\mathbf{w}}_i \sim N\left(0, \frac{1}{d}I\right)$ for every i, then $\sum_{i=1}^{m} \hat{\mathbf{w}}_i \sim N\left(0, \frac{m}{d}I\right)$. By Lemma 1 in Laurent and Massart [2000] w.p $> 1 - \delta$ we have that:

$$\left\| \sum_{i=1}^{m} \hat{\mathbf{w}}_i \right\| \geq \sqrt{m - 2\sqrt{\frac{m\log\left(\frac{1}{\delta}\right)}{d}}} .$$

Combining the two displayed equations yields:

$$\|\Pi_{P^{\perp}}(\nabla_{\mathbf{x}} N(\mathbf{x}))\| \geq \beta \cdot \sqrt{1 - 2\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{d}}} \; .$$

$\square$

C.3 Additional Lemmas

The following the lemmas are taken from Melamed et al. [2023], and used to assume w.l.o.g that $P = \text{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_{d-\ell}\}$, and that the weights of the neurons projected on $P^{\perp}$ do not change during training. We provide them here for completeness[5].

Lemma C.1. *[Theorem A.1 from Melamed et al. [2023]] Let $P \subseteq \mathbb{R}^d$ be a subspace of dimension $d - \ell$, and let $M = \text{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_{d-\ell}\}$. Let R be an orthogonal matrix such that $R \cdot P = M$, let $X \subseteq P$ be a training dataset and let $X_R = \{R \cdot \mathbf{x} : \mathbf{x} \in X\}$. Assume we train a neural network $N(\mathbf{x}) = \sum_{i=1}^{m} u_i \sigma(\mathbf{w}_i^{\top} \mathbf{x})$ as explained in Section 4, and denote by N^X and N^{X_R} the network trained on X and X_R respectively for the same number of iterations. Let $\mathbf{x}_1, \mathbf{x}_2 \in P$, then we have:*

1. *W.p. p (over the initialization) we have $\left\|\Pi_{P^{\perp}}\left(\frac{\partial N^X(\mathbf{x}_1)}{\partial \mathbf{x}}\right)\right\| \geq c$ (resp. $\leq c$) for some $c \in \mathbb{R}$, iff w.p. p also $\left\|\Pi_{M^{\perp}}\left(\frac{\partial N^{X_R}(R\mathbf{x}_1)}{\partial \mathbf{x}}\right)\right\| \geq c$ (resp. $\leq c$).*

2. *W.p. p (over the initialization) we have $\left\|\Pi_{P^{\perp}}\left(\frac{\partial N^X(\mathbf{x}_1)}{\partial \mathbf{x}}\right) - \Pi_{P^{\perp}}\left(\frac{\partial N^X(\mathbf{x}_2)}{\partial \mathbf{x}}\right)\right\| \geq c$ (resp. $\leq c$) for some $c \in \mathbb{R}$, iff w.p. p also $\left\|\Pi_{M^{\perp}}\left(\frac{\partial N^{X_R}(R\mathbf{x}_1)}{\partial \mathbf{x}}\right) - \Pi_{M^{\perp}}\left(\frac{\partial N^{X_R}(R\mathbf{x}_2)}{\partial \mathbf{x}}\right)\right\| \geq c$ (resp. $\leq c$).*

Lemma C.2 (Theorem A.2 from Melamed et al. [2023]). *Let $M = \text{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_{d-\ell}\}$. Assume we train a neural network $N(\mathbf{x}, \mathbf{w}_{1:m}) := \sum_{i=1}^{m} u_i \sigma(\mathbf{w}_i^{\top} x)$ as explained in Section 4 (where $\mathbf{w}_{1:m} = (w_1, \ldots, w_m)$). Denote by $\hat{\mathbf{w}} := \Pi_{M^{\perp}}(w)$ for $\mathbf{w} \in \mathbb{R}^d$, then after training, for each $i \in [m]$, $\hat{\mathbf{w}}_i$ did not change from their initial value.*

The following is Bernstein lemma, adapted from the phrasing of Theorem 3 in Bubeck et al. [2021].

Lemma C.3. *Let X_i for $i = 1, \ldots, m$ be i.i.d random variables with zero mean, such that there exists $c > 0$ that for all integers $q \geq 2$ we have:*

$$\mathbb{E}[|X_i|^q] \leq \frac{q! c^q}{2}$$

Then, w.p $> 1 - \delta$ we have that:

$$\sum_{i=1}^{m} X_i \leq \sqrt{2c^2 k \log\left(\frac{1}{\delta}\right)} + c \log\left(\frac{1}{\delta}\right)$$

C.4 Assumption on the theorems

Here we discuss the different assumptions made in the formal proofs, and where they are used. We separate between the assumption on the activation, and the assumptions on the parameters made in Corollary 4.1.

Assumptions on the activation. The assumption that the activation is L-smooth is used in the proof Theorem 4.1 to bound the difference between the gradient at different points. Bubeck and Sellke [2021] generalize this result to the ReLU activation, however they assume that the data point are drawn randomly from a Gaussian. The reason for such an assumption is to bound the number of neurons which change the sign for ReLU networks. In our case, since we don't assume random data points, the number of neurons that change sign depends on the data, and the neurons $\mathbf{w}_i$, which during

---

[5]The second item of Lemma C.1 does not appear as is in Melamed et al. [2023], but proved in the exact same way as first item.

training with gradient descent also become dependent on the data points. Thus, the generalization from Bubeck and Sellke [2021] is not applicable in our case.

The assumption that the derivative is lower bounded by some constant is used in the proof of Theorem 4.2. There, the norm of the gradient can be bounded more generally by the term $\sum_{i=1}^{m} \sigma'(\mathbf{w}_i^\top \mathbf{x})^2$. Again, note that each neuron $\mathbf{w}_i$ depends on the data that the network trained on. Hence, if σ' is not lower bounded, even at a single point, then $\sigma'(\mathbf{w}_i^\top \mathbf{x})^2$ can be very close to zero for every i, and thus the bound becomes vacuous.

Both assumptions can be relaxed by adding more assumptions on the data. For example, if we assume that the data is drawn randomly from a Gaussian, then we could use similar arguments to those used in Bubeck et al. [2021] to generalize these results. Another possible direction is to assume over-parameterization, i.e. the number of neurons is much larger than the number of samples. In this case, it is possible to analyze these results in the NTK regime Jacot et al. [2018].

Assumptions in Corollary 4.1. Here, besides the assumption on the activation, we additionally have assumptions on the parameters of the problem. We explain in details the interpretation of each assumption:

1. $d - \ell = \Omega(d)$. This assumption means that the dimension of the orthogonal subspace to P is large enough. Such an assumption is also needed in Melamed et al. [2023], Haldar et al. [2024], since the analysis of adversarial perturbation is made inside the subspace $P^\perp$, which needs to be large enough.

2. $R = o(\sqrt{d - \ell})$. This just means that the perturbation is not too large. This is equivalent to saying that we focus on the mesoscopic scale, i.e. where the network is non-linear, but still behaves as if it is almost linear.

3. $m = \Omega(d - \ell)$. This assumption is used since the bound on the gradient depends both on the input dimension, and the number of neurons. This assumption can be interpreted as if the number of neurons cannot be much smaller than the dimension of the subspace $P^\perp$ on which the adversarial perturbations exist.

D Experimental Details

D.1 Experiments from Section 4.2

In Figure 2, for an image $\mathbf{x}_0$ in the test dataset, we create a perturbation vector $\mathbf{v}$ of norm $\epsilon = 8/255$ or $\epsilon = 4/255$ for CIFAR100 or ImageNet datasets respectively, with respect to the L_∞ norm. The models considered are the ImageNet Swin-L [Liu et al., 2023] classifier and the CIFAR100 WRN-28-10 [Wang et al., 2023] classifier.

We consider two different choices of such $\mathbf{v}$:

1. Random direction: we start from a random $\mathbf{v} \sim \mathcal{U}\left(\{\pm\epsilon\}\right)^d$. This randomization is common for adversarial attacks.

2. Gradient direction: we calculate the gradient of the network w.r.t. the input at $\mathbf{x}_0$, $\nabla_{\mathbf{x}} N(\mathbf{x}_0)$, and normalize it to be of norm ϵ.

In both cases, we then truncate $\mathbf{v}$ to the input domain so that $\mathbf{x}_0 + \mathbf{v}$ would not exceed ± 1 in each coordinate.

We divide $\mathbf{v}$ to 100 equal parts, namely $\mathbf{v}_1 = \frac{1}{100}\mathbf{v}, \mathbf{v}_2 = \frac{2}{100}\mathbf{v}, \ldots, \mathbf{v}_{100} = \mathbf{v}$. For each i, we calculate:

$$\alpha = \frac{\|\nabla N(\mathbf{x}_0) - \nabla N(\mathbf{x}_0 + \mathbf{v}_i)\|}{\|\nabla N(\mathbf{x}_0)\|}, \quad \alpha_{\text{part}} = \frac{\|\nabla N(\mathbf{x}_0 + \mathbf{v}_{i+1}) - \nabla N(\mathbf{x}_0 + \mathbf{v}_i)\|}{\|\nabla N(\mathbf{x}_0)\|}.$$

For each i, we take an average and a mean over all the test datasets, viewed in Figure 2.

D.2 Experiments from Section 5

We use the same ϵ budget for all experiments - $8/255$ and $4/255$ in L_∞ norm for CIFAR100 and ImageNet datasets, respectively.

D.2.1 Robust Accuracy Experiment Details

AutoAttack   For all experiments, if needed, we run AutoAttack with its default hyperparameters taken from the RobustBench git repository.[6] For the CIFAR100 dataset, we took the clean and robust accuracy results directly from the RobustBench website. For the ImageNet dataset, we noticed that even the clean accuracy of the networks changes when using different versions of the Pytorch library. This clearly affects the robust accuracy of the networks since misclassified test images are considered their own adversarial examples. Therefore, we re-calculate the clean and robust accuracy for all considered networks, presenting the results of our execution.

MALT and APGD   For all experiments, for the MALT targeting method, we used $a = 9$ and $c = 100$, and the default hyperparameters of the APGD attack, adapted from the official AutoAttack git repository.[7] Note that we used the APGD attack with the given default DLR loss.

Our attack uses an APGD attack within it; therefore, it suffers from a common issue of randomization in the CIFAR100 executions (also mentioned in Appendix A). Consequently, we ran these experiments five times for each model examined and took the best value to avoid these randomization-resulted errors.

General execution details   For the CIFAR100 robust classifiers, we execute the attack in batches of 200 samples each and 50 sample batches for the ImageNet classifiers. All the weights for all the classifiers were downloaded directly from the RobustBench git repository. All experiments were done using a GPU Tesla V-100, 16GB.

D.2.2 Running Time Experiment Details

In this experiment presented in the rightmost column of Table 1 and Table 2, we sample five independent image subsets for each dataset, consisting of 400 and 200 images from CIFAR100 and ImageNet datasets, respectively. For each sample, we run both AutoAttack and our attack on the entire sample and measure the attack running time. Later, for each network, we calculate the mean and standard deviation over all five samples. The CIFAR100 attacks ran all on 2 GPUs of Tesla V-100, 16GB. The ImageNet attacks ran on 3 GPUs of Tesla V-100, 16GB. Of course, for each network, both attacks were executed one after the other using the exact same computing power.

D.2.3 MALT and FAB Details

For this experiment, we take the same $a = 9$ and $c = 100$ hyperparameters for MALT and take FAB implementation from the official AutoAttack git repository, with all its default hyperparameters.

D.2.4 Targeting Analysis Details

For the analysis of the targeting capabilities in Figure 4, we ran two attacks:

1. MALT and APGD, with the same $a = 9$, $c = 100$ hyperparameters
2. APGD attack with naive targeting to top 9 targets, as it performed in the targeted version of APGD within AutoAttack.

We observe the number of successful attacks for each targeting method and for each ordered target suggested. For naive targeting columns, the targets are ordered by the corresponding class confidence level given by the model. For the MALT columns, the targets are ordered according to the MALT algorithm.

---

[6]https://github.com/RobustBench/robustbench
[7]https://github.com/fra31/auto-attack

44385

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We present our attack algorithm in Section 3, its theoretical justification and applied evidence in Section 4, and its winning experimental results in Section 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: For the teoretical limitation, we include the theoretical limitations and assumptions in Remark 4.1, and elaborate about it in Appendix C.4. For the applied results limitation, we mention the subset of classifier examined, particularly not being the full set. We thoroughly discuss the time complexity in worst-case in Section 3, and empirically in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We include the theoretical limitations and assumptions in Remark 4.1, and elaborate about it in Appendix C.4. Full proof of Lemma 3.1 are in Appendix B. Full proof for the theorems 4.1, 4.2, 4.1 are in Appendix C.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Experiment presented in Section 3 are fully detailed in Appendix A. The experiments from Section 4 and Section 5 are fully detailed in Appendix D. We provide links to all external code, attacks, and models we use, full description of all hyperparametes chosen, and a full description of our method, so it can be easily reproduced. In the final version we will link to a (non-anonymized) GitHub page with an API for our attack.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We detail all the commands and environment details required to faithfully reproduce the main experimental results in Appendix D, including relevant GitHub references and links.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Our paper doesn't include any training, but test details are supplies fully in Appendix D, and the important datails are within the experimental Sections 5, 4.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The statistical significance are extremely important at Section 4.2, Figure 2, and plotted as a blurred colored area around the reported mean, to demonstrate its significance. In experiments in Section 5, we denote it using the $\pm$ sign next to the running time mean in Tables 1 and 2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Computing details are in Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: Read and conform.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Our paper presents a faster attack that overcomes few of the state of the art attack limitations. As any other paper regarding adversarial attacks, it may have both positive and negative societal impacts. We suggest a faster attack that can be used to attack neural networks, yet we believe reviling those weakness of neural networks will encourage researchers to develop networks more robust to these attacks.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite all the papers and sources from which we took the datasets, models and attack codes, including URLs.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.