Proximal Splitting Adversarial Attack for Semantic Segmentation

Jérôme Rony
ÉTS Montréal

Jean-Christophe Pesquet
Centre de Vision Numérique
Université Paris-Saclay, CentraleSupélec, Inria

Ismail Ben Ayed
ÉTS Montréal

Code: https://github.com/jeromerony/alma_prox_segmentation

Abstract

Classification has been the focal point of research on adversarial attacks, but only a few works investigate methods suited to denser prediction tasks, such as semantic segmentation. The methods proposed in these works do not accurately solve the adversarial segmentation problem and, therefore, overestimate the size of the perturbations required to fool models. Here, we propose a white-box attack for these models based on a proximal splitting to produce adversarial perturbations with much smaller ℓ_∞ norms. Our attack can handle large numbers of constraints within a nonconvex minimization framework via an Augmented Lagrangian approach, coupled with adaptive constraint scaling and masking strategies. We demonstrate that our attack significantly outperforms previously proposed ones, as well as classification attacks that we adapted for segmentation, providing a first comprehensive benchmark for this dense task.

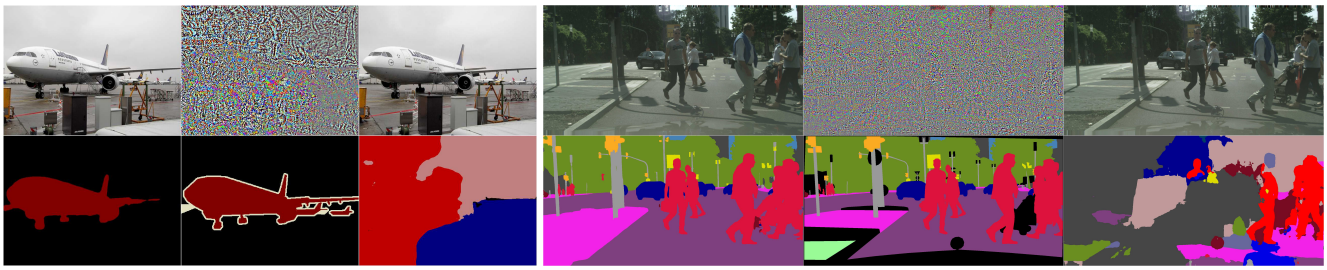
1. Introduction

Research on white-box adversarial attacks has mostly focused on classification tasks, with several methods proposed over the years for ℓ_p-norms [8, 9, 20–22, 27, 30, 32, 35, 36, 42]. In contrast, the literature on adversarial attacks for segmentation tasks has been much scarcer, with few works proposing attacks [13, 33, 39]. The lack of studies on adversarial attacks in segmentation may appear surprising because of the prominence of this computer vision task in many applications where a semantic understanding of image contents is needed. In many safety-critical application areas, it is thus crucial to assess the robustness of the employed segmentation models.

Although segmentation is treated as a per-pixel classification problem, designing adversarial attacks for semantic segmentation is much more challenging for several reasons. First, from an optimization perspective, the problem of adversarial example generation is more difficult. In a classification task, producing minimal adversarial examples is a nonconvex constrained problem with a single constraint. In a segmentation task, this optimization problem now has multiple constraints, since at least one constraint must be addressed for each pixel in the image. For a dataset such as Cityscapes [19], the images have a size of $2\,048\times1\,024$, resulting in more than 2 million constraints. Consequently, most attacks originally designed for classification cannot be directly extended to segmentation. For instance, penalty methods such as C&W [9] cannot tackle multiple constraints since they rely on a binary search of the penalty weight.

Second, the computational and memory cost of generating adversarial examples can be prohibitive. White-box adversarial attacks usually rely on computing the gradient of a loss w.r.t. the input. In segmentation tasks, the dense outputs result in high memory usage to perform a backward propagation of the gradient. For reference, computing the gradients of the logits w.r.t. the input requires $\sim$22 GiB of memory for FCN HRNetV2 W48 [38] on Cityscapes with a $2\,048\times1\,024$ image. Additionally, most recent classification adversarial attacks require between 100 and 1 000 iterations, resulting in a run-time of up to a few seconds per image [34, 36] (on GPU) depending on the dataset and model. For segmentation, this increases to tens or even hundreds of seconds per image with larger models.

In this article, we propose an adversarial attack to produce minimal adversarial perturbations w.r.t. the ℓ_∞-norm, for deep semantic segmentation models. Building on Augmented Lagrangian principles, we introduce adaptive strategies to handle a large number of constraints (*i.e.* $>10^6$). Furthermore, we tackle the nonsmooth ℓ_∞-norm minimization with a proximal splitting instead of gradient descent. In particular, we show that we can efficiently compute the proximity operator of the sum of the ℓ_∞-norm and the indicator function of the space of possible perturbations. This results in an adversarial attack that significantly outperforms the DAG attacks [39], in addition to several classification attacks that we were able to adapt for segmentation. We propose a methodology to evaluate adversarial attacks in segmentation, and compare the different approaches on the Cityscapes [19] and Pascal VOC 2012 [23] datasets with a diverse set of architectures, including well-known DeepLabV3+ models [11], the recently proposed transformer-based SegFormer [40], and the robust DeepLabV3 DDC-AT model from [41]. The



(a) Pascal VOC 2012: $\|\boldsymbol{\delta}\|_\infty = {}^{0.67}/_{255}$ (b) Cityscapes: $\|\boldsymbol{\delta}\|_\infty = {}^{0.53}/_{255}$

Figure 1. Untargeted adversarial examples for FCN HRNetV2 W48 on Pascal VOC 2012 and Cityscapes. In both cases, more than 99% of pixels are incorrectly classified. For each dataset, left is the original image and its predicted segmentation, middle is the amplified perturbation and the ground truth segmentation and right is the adversarial image with its predicted segmentation. For Pascal VOC 2012, the predicted classes are *TV monitor* (blue), *person* (beige) and *chair* (bright red).

proposed approach yields outstanding performances for all models and datasets considered in our experiments. For instance, our attack finds untargeted adversarial perturbations with ℓ_∞-norms lower than ${}^1/_{255}$ on average for all models on Cityscapes. With this attacker, we provide better means to assess the robustness of deep segmentation models, which has often been overestimated until now, as the existing attacks could only find adversarial examples with larger norms.

2. Related Works

While the literature on minimal adversarial attacks for classification is vast, the research on attacks for segmentation is much less developed. The main work on adversarial attacks for segmentation is done by Xie *et al*. [39]. It proposes a simple algorithm to generate adversarial perturbations for dense prediction tasks, including object detection and segmentation, called the Dense Adversary Generation (DAG) attack. In this attack, the rescaled gradient of the loss w.r.t. the input is added to the current perturbation, until the stopping criterion is reached, *i.e*. a given percentage of pixels is adversarial. In each iteration, the total loss is the sum of the losses over pixels that are not adversarial. This can be seen as a form of greedy algorithm. See Appendix A for the complete algorithm of the DAG attack and a discussion on the stopping criterion used. In practice, this attack is quite efficient, however, it simply accumulates gradients until the stopping criterion is reached. Therefore, it does not minimize the norm considered. Cisse *et al*. propose the Houdini attack for several tasks [13], including segmentation. The goal of this approach is to maximize a surrogate loss for a given perturbation budget (*i.e*. constraint on the ℓ_∞-norm), hence not producing minimal perturbations. More recently, Ozbulak *et al*. studied adversarial examples on a medical image segmentation task [33]. They propose a targeted attack to minimize the ℓ_2-norm, which is a regular penalty method. The weight of the penalty terms is fixed to 1, however, leading to large perturbations.

Other works study the robustness of segmentation models against adversarial attacks [1, 24, 25]. In these works, the authors use FGSM [27] or an iterative version of FGSM. However, FGSM is not a minimization attack and is known to provide rough robustness evaluations. This leads to largely overestimated robustness results on both Pascal VOC 2012 and Cityscapes in [1].

Even though most adversarial attacks were designed for classification, some may be adapted for segmentation tasks. In particular, ℓ_∞ attacks that do not rely on projections onto an estimated decision boundary can be used for segmentation (as opposed to DeepFool [32] or FAB [20]). These attacks are PGD [29], FMN [34] and PDPGD [30]. Note that PDPGD [30] relies on a proximal splitting method, but uses the AdaProx algorithm [31]; the latter is appealing but, unlike the prox-Newton algorithm it is inspired from [5], introduces a mismatch between the scaling in the computation of the proximity operator and the step-size of the gradient step. The convergence study of such an algorithm would be quite challenging in the non-convex case. It is also known that, even in the convex case, when such a mismatched algorithm converges, the asymptotic point differs from the solution to the original optimization problem [37].

3. Preliminaries

Let $\boldsymbol{x} \in \mathcal{X}$ be an input image with its corresponding label map $\boldsymbol{y} \in \mathcal{Y}$. Usually, in computer vision problems, $\mathcal{X} = [0,1]^{C\times H\times W}$ and $\mathcal{Y} = \{1,\dots,K\}^{H\times W}$, where H and W are the height and width of the image, C is the number of channels (*e.g*. 3 for RGB images), and K is the number of labelling classes. Our objective is to fool the model $f\colon \mathcal{X} \to \mathbb{R}^{K\times H\times W}$ producing *logits* $\boldsymbol{z} = f(\boldsymbol{x}) = \big(f(\boldsymbol{x})_{k,i,j}\big)_{k,i,j} \in \mathbb{R}^{K\times H\times W}$. This means that we are looking for a perturbation vector $\boldsymbol{\delta}$ such that, for every pixel $(i,j) \in \{1,\dots,H\}\times\{1,\dots,W\}$, the maximum value of $(f(\boldsymbol{x}+\boldsymbol{\delta})_{k,i,j})_{1\le k\le K}$ is reached for a label different from the true one $\boldsymbol{y}_{i,j}$. In addition, the perturbation should satisfy

the value range constraints: $\boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X}$. We denote Λ the set of admissible perturbations, so here $\Lambda = [0,1]^{C\times H\times W} - \boldsymbol{x}$. For simplicity, with a slight abuse of notation, we index the pixels with $i \in \{1,\dots,d\}$ where $d = HW$ is the total number of pixels.

Problem formulation The minimal adversarial perturbation problem for the ℓ_∞-norm can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{\delta}} \quad & \|\boldsymbol{\delta}\|_\infty \\ \text{s.t.} \quad & \operatorname*{arg\,max}_{k\in\{1,\dots,K\}} f(\boldsymbol{x}+\boldsymbol{\delta})_{k,i} \neq \boldsymbol{y}_i, \quad i=1,\dots,d, \\ & \boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X}. \end{aligned} \tag{1}$$

The d $\arg\max$ constraints correspond to the misclassification of each pixel. The $\boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X}$ constraint corresponds to producing a perturbation that results in a valid image, and can be re-written as $\boldsymbol{\delta} \in \Lambda$. Note that this constraint can also be encoded in the objective using the indicator function:

$$\iota_\Lambda(\boldsymbol{\delta}) = \begin{cases} 0 & \text{if} \quad \boldsymbol{\delta} \in \Lambda; \\ +\infty & \text{else,} \end{cases} \tag{2}$$

resulting in the minimization of $\|\boldsymbol{\delta}\|_\infty + \iota_\Lambda(\boldsymbol{\delta})$.

Equivalent problem As is common in several attacks [9, 21, 35], we replace the non-differentiable misclassification constraints by differentiable ones, to make it compatible with first-order optimization methods. Here, the $\arg\max$ constraints are replaced by constraints on the Difference of Logits Ratio (DLR) [20], or rather the DLR^+ [35]:

$$\begin{aligned} \min_{\boldsymbol{\delta}} \quad & \|\boldsymbol{\delta}\|_\infty \\ \text{s.t.} \quad & \mathrm{DLR}^+(f(\boldsymbol{x}+\boldsymbol{\delta})_i, \boldsymbol{y}_i) + \varepsilon \leq 0, \ i=1,\dots,d, \\ & \boldsymbol{\delta} \in \Lambda, \end{aligned} \tag{3}$$

where $\mathrm{DLR}^+(\boldsymbol{z}, y) = \frac{\boldsymbol{z}_y - \max_{i\neq y} \boldsymbol{z}_i}{\boldsymbol{z}_{\pi_1} - \boldsymbol{z}_{\pi_3}}$, with π_i the index of the i-th largest logit and ε a small positive constant.

Augmented Lagrangian method One way to handle the misclassification constraints is to use an Augmented Lagrangian approach [35], which in the classification setting, performs a gradient descent on the following quantity:

$$D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) + P\Big(\mathrm{DLR}^+(g(\boldsymbol{x}+\boldsymbol{\delta})), \rho, \mu\Big), \tag{4}$$

where D is a discrepancy measure (*e.g.* ℓ_2-norm, LPIPS [43]), P is a penalty-Lagrangian function parametrized by a parameter ρ and a multiplier μ, and $g : \mathcal{X} \to \mathbb{R}$ is a classification model. In [35], the penalty multiplier μ is updated after every gradient descent iteration to increase or decrease the weight of penalty, and eventually satisfy the misclassification constraint.

4. Proposed Method

Our segmentation attack is built on the general Augmented Lagrangian principle, which has led to competitive performances in the ALMA classification attack [35]. It provides an efficient solution to challenging nonconvex problems arising when designing an attacker, and we will see that it can be extended to cope with multiple constraints. This can be achieved by introducing one penalty per constraint, with its associated parameter ρ and multiplier μ. Denoting $\boldsymbol{d} = \mathrm{DLR}^+(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y}) \in \mathbb{R}^d$, we tackle problem (1) by minimizing the following objective:

$$\underbrace{\|\boldsymbol{\delta}\|_\infty + \iota_\Lambda(\boldsymbol{\delta})}_{h_1} + \underbrace{\boldsymbol{m}^\top P(\boldsymbol{d}, \boldsymbol{\rho}, \boldsymbol{\mu})}_{h_2}, \tag{5}$$

where $\boldsymbol{m} \in \{0,1\}^d$ is a binary mask (detailed in the following section), $(\boldsymbol{\rho}, \boldsymbol{\mu}) \in \mathbb{R}^d_{++} \times \mathbb{R}^d_{++}$ are the penalty parameters and multipliers associated with each constraint, and P is the penalty function applied componentwise. This formulation raises many technical challenges in the context of segmentation, when dealing with millions of constraints. Additionally, gradient based optimization does not accommodate nonsmooth functions such as the ℓ_∞-norm. In this work, we bring several modifications to make it (*i*) suitable for segmentation and (*ii*) applicable to the ℓ_∞-norm.

Our attack, called ALMA prox, consists in minimizing (5) using a proximal splitting [15] to handle the nonsmooth term h_1 and an Augmented Lagrangian method to satisfy the constraints by minimizing h_2 using a gradient descent. In Section 4.1, we introduce the adaptive constraint strategies to handle large numbers of constraints in the Augmented Lagrangian framework, and in Section 4.2, we detail the proximal splitting iteration. The complete algorithm of our attack is provided in Appendix C.

4.1. Adaptive constraints strategies

Constraint masking In segmentation tasks, some regions are unlabeled (*e.g.* object boundaries in Pascal VOC, *void* class in Cityscapes), and should be ignored during an attack. We use a binary mask $\boldsymbol{m} \in \{0,1\}^d$ to encode this, effectively reducing the number of constraints from d to $\|\boldsymbol{m}\|_1$. Given the number of constraints considered in this problem (*e.g.* $\sim 10^6$ for Cityscapes), we consider an attack as successful if it satisfies at least a fraction ν of the constraints. In this paper, we use $\nu = 99\%$. To consolidate this in our attack, we compute a mask $\tilde{\boldsymbol{m}}^{(t)} \in \{0,1\}^d$ at each iteration $t \in \{1,\dots,N\}$ such that $\nu\|\boldsymbol{m}\|_1 \leq \|\tilde{\boldsymbol{m}}^{(t)}\|_1 \leq \|\boldsymbol{m}\|_1$. This mask discards the largest constraints, aligning the optimization objective with the criterion of successful attack. The discarded constraints are less likely to be satisfied, so this avoids the continuous increase of their associated multipliers, which may result in larger perturbations. In practice, at each iteration t, we use a threshold $\xi^{(t)}$ as a percentile of

the constraints $\boldsymbol{d}^{(t)}$ to obtain $\tilde{\boldsymbol{m}}^{(t)}$, and linearly decay $\xi^{(t)}$ from 100% at the first iteration to ν at the last iteration:

$$\begin{aligned} \xi^{(t)} &= \Big(1-(1-\nu)\frac{t-1}{N-1}\Big)\text{-percentile of } \boldsymbol{d}^{(t)}, \\ \tilde{\boldsymbol{m}}^{(t)} &= [\boldsymbol{d}^{(t)} \leq \xi^{(t)}], \end{aligned} \tag{6}$$

where $[\cdot]$ is the Iverson bracket applied componentwise.

Constraint scaling When dealing with large numbers of constraints in Augmented Lagrangian methods, it is standard practice to scale them (see section 12.5 of [6]). However, there is no principled way of choosing this scale, as it is problem dependent. At iteration t, we multiply all the constraints by a scaling factor $w^{(t)} \in]0,1]$ by computing

$$h_2(\boldsymbol{\delta}) = (\tilde{\boldsymbol{m}}^{(t)})^\top P\big(w^{(t)}\boldsymbol{d}^{(t)}, \boldsymbol{\rho}^{(t)}, \boldsymbol{\mu}^{(t)}\big). \tag{7}$$

This scaling factor is adjusted during the attack with a binary decision: if the pixel success rate at the current iteration is less than the target pixel success rate ν, the scaling factor is increased, otherwise it is decreased:

$$w^{(t)} = w^{(t-1)} \times \begin{cases} \frac{1}{1-\gamma_w} & \text{if } \ \frac{\boldsymbol{m}^\top[\boldsymbol{d}^{(t)}\leq 0]}{\|\boldsymbol{m}\|_1} < \nu; \\ \frac{1}{1+\gamma_w} & \text{otherwise.} \end{cases} \tag{8}$$

Multiplying by $\frac{1}{1\pm\gamma_w}$ instead of $1\pm\gamma_w$ biases $w^{(t)}$ to increase in case of oscillations, with two iterations resulting in a multiplication of $\frac{1}{1-\gamma_w^2}$ instead of $1-\gamma_w^2$. $w^{(t)}$ is further projected on a safeguarding interval $[w_{\min}, 1]$. We use $w_{\min} = 0.1$ and set the initial scale to $w^{(0)} = 1$.

Penalty For the penalty function, we use a slightly modified version of the P_2 penalty from [6]. It corresponds to the following mapping applied componentwise defined $(\forall y \in \mathbb{R})(\forall(\rho,\mu) \in [0,+\infty[^2)$:

$$P(y,\rho,\mu) = \begin{cases} \mu y + \mu\rho y^2 + \frac{1}{6}\rho^2 y^3 & \text{if } \ y \geq 0; \\ \frac{\mu y}{1-\max(1,\rho)y} & \text{otherwise.} \end{cases} \tag{9}$$

This penalty also satisfies the requirements in [6], while allowing the penalty multipliers to decrease faster for negative values of the constraint with $\rho < 1$.

4.2. Proximal splitting

Proximal methods have been the workhorse of nonsmooth large scale optimization in the last two decades [3, 15]. Originally designed for solving convex optimization problems, they have been also successfully employed in the nonconvex case [2]. One of their main advantages is that they offer the ability to split the cost function in a sum of terms which can be addressed either by computing their proximity operator, or their gradient when they are smooth. One important point is that, generally, the limitations on the step-size arising in proximal algorithms are related to the Lipschitz-regularity of the gradient of the smooth part, whereas the proximity operators of the other functions do not introduce any restriction.

As a common practice in adversarial attacks, we are interested in minimizing the ℓ_∞-norm. To handle this term, at each iteration of the proposed algorithm, our problem is expressed as the minimization of a sum of two functions as in (5), where h_1 is convex and h_2 is smooth. With this formulation, we can solve the problem using a *forward-backward* splitting algorithm where the update reads

$$\boldsymbol{\delta}^{(t+1)} = \underbrace{\operatorname{prox}_{\lambda h_1}}_{\text{backward step}}\Big(\underbrace{\boldsymbol{\delta}^{(t)} - \lambda\nabla_{\boldsymbol{\delta}} h_2\big(\boldsymbol{\delta}^{(t)}\big)}_{\text{forward step}}\Big), \tag{10}$$

λ is a positive step-size, and $\operatorname{prox}_{\lambda h_1}$ is the proximity operator of the function λh_1. Global convergence guarantees of the forward-backward algorithm exist in the convex case [17] and local ones in the nonconvex case [2]. To carry out our attack, we thus need to find the proximity operator of h_1.

Proximity operator of h_1 The function h_1 is a sum of two functions: $\|\cdot\|_\infty$ and ι_Λ. In general, the proximity operator of a sum of functions *is neither* the sum of the proximity operators of the functions *nor* their composition. One way to solve the sub-problem of finding the prox of $h_1 = \|\cdot\|_\infty + \iota_\Lambda$ would be to resort to an iterative proximal splitting algorithm, such as the dual forward-backward splitting [15]. However, we will propose a more efficient numerical approach by going back to the definition of the proximity operator [4]:

$$\begin{aligned} \operatorname{prox}_{\lambda h_1}(\boldsymbol{\delta}) &= \operatorname{prox}_{\lambda\|\cdot\|_\infty + \iota_\Lambda}(\boldsymbol{\delta}) \\ &= \operatorname*{arg\,min}_{\boldsymbol{p}\in\mathbb{R}^{Cd}} \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{\delta}\|_2^2 + \lambda\|\boldsymbol{p}\|_\infty + \iota_\Lambda(\boldsymbol{p}). \end{aligned} \tag{11}$$

As $\mathcal{X} = [0,1]^{Cd}$, we can reformulate this problem as:

$$\begin{aligned} \min_{\boldsymbol{p},\beta} \quad & \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{\delta}\|_2^2 + \lambda\beta \quad \text{s.t.} \quad -\beta\mathbf{1}_{Cd} \leq \boldsymbol{p} \leq \beta\mathbf{1}_{Cd}, \\ & \qquad\qquad\qquad\qquad -\boldsymbol{x} \leq \boldsymbol{p} \leq \mathbf{1}_{Cd} - \boldsymbol{x}, \end{aligned} \tag{12}$$

where $\mathbf{1}_{Cd} = [1,\dots,1]^\top \in \mathbb{R}^{Cd}$. Since $\Lambda = [0,1]^{Cd} - \boldsymbol{x} \subset [-1,1]^{Cd}$, the maximum possible ℓ_∞-norm of the solution $\boldsymbol{p}^\star$ is 1, so we need to solve (12) for $\beta \in [0,1]$. For $\beta = 0$, the solution is trivially $\boldsymbol{p}^\star = 0$. For $\beta = 1$, the solution is $\boldsymbol{p}^\star = \mathcal{P}_\Lambda(\boldsymbol{\delta})$ since the problem reduces to

$$\min_{\boldsymbol{p}} \quad \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{\delta}\|_2^2 \quad \text{s.t.} \quad -\boldsymbol{x} \leq \boldsymbol{p} \leq \mathbf{1}_{Cd} - \boldsymbol{x}. \tag{13}$$

For a given $\beta \in]0,1[$, (12) becomes a projection problem, so we can express the solution for $\boldsymbol{p}$ as $\boldsymbol{p}_\beta = \mathcal{P}_{[-\beta,\beta]^{Cd}\cap\Lambda}(\boldsymbol{\delta})$. The optimal value of β has then to be found by solving

$$\min_{\beta\in[0,1]} \quad \frac{1}{2}\|\boldsymbol{p}_\beta - \boldsymbol{\delta}\|_2^2 + \lambda\beta. \tag{14}$$

Proposition 1. *Problem (14) is convex and admits a unique optimal solution $\beta^\star \in [0,1]$, for which the following inequality holds:*

$$0 \leq \beta^\star \leq \|\mathcal{P}_\Lambda(\boldsymbol{\delta})\|_\infty . \tag{15}$$

The proof of this result is provided in Appendix B. Several methods can be used to solve (14). We employ a ternary search since it offers a linear convergence rate, avoids the computation of the gradient of the objective w.r.t. β, and is numerically stable. It also yields a number of iterations depending on the required absolute precision $\Delta\beta$. Based on inequality (15), the number of iterations for the ternary search is $\log(\Delta\beta / \|\mathcal{P}_\Lambda(\boldsymbol{\delta})\|_\infty) / \log(2/3)$. Since we are performing adversarial attacks on images, the precision is usually 8-bit, or $1/255 \approx 3.9 \times 10^{-3}$, we solve the problem with $\Delta\beta = 10^{-5}$, yielding 29 iterations at worse, when $\|\mathcal{P}_\Lambda(\boldsymbol{\delta})\|_\infty = 1$. The procedure to compute $\boldsymbol{p}^\star = \text{prox}_{\lambda\|\cdot\|_\infty + \iota_\Lambda}(\boldsymbol{\delta})$ is described in Algorithm 1.

Algorithm 1 Ternary search for $\boldsymbol{p}^\star = \text{prox}_{\lambda\|\cdot\|_\infty + \iota_\Lambda}(\boldsymbol{\delta})$

Require: Input vector $\boldsymbol{\delta}$ and feasible set Λ.
Require: Absolute precision $\Delta\beta$ on the solution.
1: $\boldsymbol{\delta}_\Lambda \leftarrow \mathcal{P}_\Lambda(\boldsymbol{\delta})$
2: $l \leftarrow 0$ // Lower bound for $\beta^\star$
3: $u \leftarrow \|\boldsymbol{\delta}_\Lambda\|_\infty$ // Upper bound for $\beta^\star$
4: $n \leftarrow \lceil \log(\Delta\beta / u) / \log(2/3) \rceil$ // Number of steps
5: **for** $t \leftarrow 1, \dots, n$ **do**
6: $\quad \beta_l \leftarrow l + (u-l)/3$ // Points to evaluate the objective
7: $\quad \beta_u \leftarrow u - (u-l)/3$
8: $\quad \boldsymbol{p}_l \leftarrow \mathcal{P}_{[-\beta_l, \beta_l]}(\boldsymbol{\delta}_\Lambda)$ // Projection on interval
9: $\quad \boldsymbol{p}_u \leftarrow \mathcal{P}_{[-\beta_u, \beta_u]}(\boldsymbol{\delta}_\Lambda)$
10: $\quad f_l \leftarrow \frac{1}{2}\|\boldsymbol{p}_l - \boldsymbol{\delta}\|_2^2 + \lambda\beta_l$ // Value of objective
11: $\quad f_u \leftarrow \frac{1}{2}\|\boldsymbol{p}_u - \boldsymbol{\delta}\|_2^2 + \lambda\beta_u$
12: $\quad$ **if** $f_l \geq f_u$ **then** // Update bounds for search
13: $\quad\quad l \leftarrow \beta_l$
14: $\quad$ **else**
15: $\quad\quad u \leftarrow \beta_u$
16: $\quad$ **end if**
17: **end for**
18: $\beta^\star = (l+u)/2$ // Minimum is in $[l, u]$
19: **return** $\boldsymbol{p}^\star = \mathcal{P}_{[-\beta^\star, \beta^\star]}(\boldsymbol{\delta}_\Lambda)$ // Return prox using $\boldsymbol{\delta}_\Lambda$

Variable Metric Forward-Backward Forward-Backward algorithms can suffer from slow convergence. Therefore, we propose to use a Variable Metric Forward Backward (VMFB) [12, 16] algorithm as an acceleration.

Definition. Let H be a positive definite matrix in $\mathbb{R}^{Cd \times Cd}$. For all $\lambda > 0$, the proximity operator of λh_1 in the metric H is defined as

$$\text{prox}^{\mathsf{H}}_{\lambda h_1}(\boldsymbol{\delta}) = \underset{\boldsymbol{p} \in \mathbb{R}^{Cd}}{\arg\min} \frac{1}{2}\|\boldsymbol{p} - \boldsymbol{\delta}\|^2_{\mathsf{H}} + \lambda h_1(\boldsymbol{\delta}), \tag{16}$$

where $\|\boldsymbol{p}\|_{\mathsf{H}} = \sqrt{\boldsymbol{p}^\top \mathsf{H} \boldsymbol{p}}$ is a weighted norm.

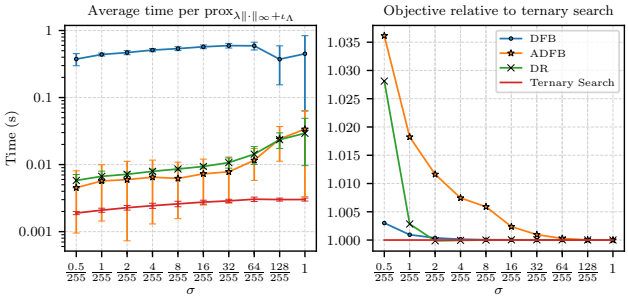


Figure 2. Comparison of average run-time and quality of solution in terms of optimization objective of DFB, ADFB, DR and Ternary Search for pseudorandom vectors $\boldsymbol{\delta} \sim \mathcal{N}(0, \sigma^2 I_d)$.

Following the definition, the update in VMFB reads

$$\boldsymbol{\delta}^{(t+1)} = \text{prox}^{\mathsf{H}}_{\lambda h_1}\left(\boldsymbol{\delta}^{(t)} - \lambda \mathsf{H}^{-1} \nabla_{\boldsymbol{\delta}} h_2(\boldsymbol{\delta}^{(t)})\right). \tag{17}$$

This method requires inverting a $Cd \times Cd$ square matrix, which may be impractical. Therefore, we use a diagonal metric $\mathsf{H} = \text{Diag}(\boldsymbol{s})$ with $\boldsymbol{s} \in]0, +\infty[^{Cd}$. At iteration t, the diagonal vector $\boldsymbol{s}^{(t)}$ is estimated from the gradient as

$$\begin{aligned} \boldsymbol{v}^{(t)} &= \alpha \boldsymbol{v}^{(t-1)} + (1-\alpha)\left(\nabla_{\boldsymbol{\delta}} h_2(\boldsymbol{\delta}^{(t)})\right)^2 \\ \boldsymbol{s}^{(t)} &= \sqrt{\frac{\boldsymbol{v}^{(t)}}{1-\alpha^t}} + \varepsilon \end{aligned} \tag{18}$$

where the square and square root operations are performed componentwise, $\alpha \in [0, 1[$ is a smoothing parameter, and $\boldsymbol{v}^{(0)} = \boldsymbol{0}_{Cd}$.

Remark. With this choice for the metric, the forward (*i.e.* gradient) step becomes equivalent to the one in the Adam algorithm [26] with $(\beta_1, \beta_2) = (0, \alpha)$.

Note that Proposition 1 still holds for a diagonal metric. Indeed, the projection on Λ w.r.t. a diagonal metric $\mathcal{P}^{\mathsf{H}}_\Lambda = \arg\min_{\boldsymbol{y} \in \Lambda} \|\cdot - \boldsymbol{y}\|^2_{\mathsf{H}}$ is equal to the Euclidean projection $\mathcal{P}_\Lambda = \arg\min_{\boldsymbol{y} \in \Lambda} \|\cdot - \boldsymbol{y}\|_2^2$. This is readily deduced from the fact that both projection problems are separable in each component, since H is diagonal and the constraint $\boldsymbol{y} \in \Lambda$ is separable. Therefore, since $\mathcal{P}^{\mathsf{H}}_\Lambda = \mathcal{P}_\Lambda$, the proof is unchanged. Additionally, the ternary search approach to compute $\boldsymbol{p}^\star$ can still be used: the norms in steps 10 and 11 are replaced by the weighted norm.

5. Experiments

5.1. Ternary Search

To evaluate the efficiency of our method to compute $\text{prox}_{\lambda h_1}$ using a ternary search, we provide a comparison with several traditional iterative splitting algorithms: Dual Forward-Backward (DFB) splitting [14], an Accelerated variant (ADFB) using Nesterov acceleration on the

dual problem [10], and a Douglas-Rachford (DR) splitting algorithm [28]. We generate pseudorandom samples $\boldsymbol{x} \sim \mathcal{U}(\mathbf{0}_d, \mathbf{1}_d)$, perturbation vectors $\boldsymbol{\delta} \sim \mathcal{N}(0, \sigma^2 I_d)$, and scale $\lambda \sim 10^{\mathcal{U}(-1,3)}$, with $d = 2^{18} = 512 \times 512$. For all methods, we use an absolute stopping criterion on the solution $\left\|\boldsymbol{p}^{(t+1)} - \boldsymbol{p}^{(t)}\right\|_\infty \leq 10^{-5}$. For each σ, we repeat the evaluation 100 times, and report the average compute time[1] and the value of the objective from (11) relative to the one obtained using the ternary search.

The results are shown in Figure 2. Overall, the Ternary Search approach is faster and provides more accurate solutions to the computation of the proximity operator of interest than DFB, ADFB and DR. In some occasions, it may happen that ADFB converges faster, but the trade-off is a worse solution in terms of objective on average.

5.2. Datasets and Models

We perform experiments on the validation set of two well-known segmentation datasets: Pascal VOC 2012 [23] and Cityscapes [19]. We evaluate the attacks on a collection of models chosen based on several criteria: widespread usage, high performance, and architectures diversity. With these criteria, we selected DeepLabV3+ ResNet-50, ResNet-101 [11], and FCN HRNetV2 W48 [38] which are CNN based models, and SegFormer MiT-B0 and MiT-B3 [40] which are transformer based models. We also consider the robust model DeepLabV3 ResNet-50 DDC-AT from [41].

For white-box attacks, most attack algorithms compute the gradient of a loss w.r.t. the input. In the context of adversarial attacks, this quantity is computed for validation images, which can be larger than the image crops used in training. For Cityscapes specifically, models are evaluated on $2\,048 \times 1\,024$ images. This leads to high memory usage, and, in particular, DeepLabV3+ ResNet-101 and larger variants of the SegFormer family (*i.e.* MiT-B4 and MiT-B5) require more than $40\,\text{GB}$ of memory per gradient computation on $2\,048 \times 1\,024$ images. Therefore, we refrain from doing experiments on models requiring more than $40\,\text{GB}$ of GPU memory[1] to ease future comparisons.

For most segmentation models, the evaluation protocol involves resizing the image to a specified size (*e.g.* such that the smallest side has a specified length, while keeping the aspect ratio), using the model to produce a segmentation, and then resizing this segmentation to the original image size and compute the performance metrics. In our context, we do not perform this resizing before and after. This slight modification of the evaluation protocol leads to marginal differences in the typical performance metrics, which we report in Appendix D. All the models weights are fetched from the MMSegmentation library [18], except for the robust model DeepLabV3 DDC-AT from [41], which was obtained from the repository associated with the paper.

[1]Experiments were run on NVIDIA A100 SXM4 40 GB GPUs.

5.3. Metrics

In segmentation, the concept of a *successful attack* is more ambiguous than in classification, where success is a binary criterion. From a security perspective in a segmentation task, *mostly* making wrong predictions, except for a few pixels, can be considered as a successful attack (or a model failure), even though all the constraints are not satisfied.

Previous works on adversarial examples in a segmentation context [1, 39, 41] measure the model robustness (or equivalently, attack performance) using the mean Intersection over Union (mIoU) over all classes. However, this metric is biased towards small regions, and does not indicate how well the adversarial optimization problem (1) is solved. Therefore, to measure the success of an attack, we simply measure the constraint satisfaction rate over all pixels in the mask $\boldsymbol{m}$, irrespective of the original class. For a given image, we call this constraint satisfaction rate the Attack Pixel Success Rate (APSR). For untargeted attacks, the APSR is defined as:

$$\text{APSR} = \frac{\boldsymbol{m}^\top [\arg\max_k f(\boldsymbol{x} + \boldsymbol{\delta})_{k,i} \neq \boldsymbol{y}_i]}{\|\boldsymbol{m}\|_1} \in [0, 1] \quad (19)$$

Although our approach has been presented in the context of untargeted attacks, it can also be straightforwardly extended to targeted attacks. In such a case, a target label $\boldsymbol{t} = (\boldsymbol{t}_i)_{1 \leq i \leq d}$ is provided and the statement becomes $\arg\max_k f(\boldsymbol{x} + \boldsymbol{\delta})_{k,i} = \boldsymbol{t}_i$.

From there, choosing a specific threshold is arbitrary. In our experiments, we use $\nu = 99\%$ as a threshold for the APSR, meaning that an attack is considered successful if $\text{APSR} \geq 99\%$. From an optimization perspective, this indicates that we satisfy at least 99% of the constraints in problem (1). A lower threshold could also result in low quality segmentations. However, given the small ℓ_∞-norms of the perturbations observed to satisfy such a high threshold, we argue that it highlights the effectiveness of our attack.

5.4. Attack objectives

The most common scenario for adversarial attacks is the untargeted setting. While it is natural for classification, this can produce unrealistic segmentations that could easily be filtered or rejected (see Figure 1). Therefore, we need to find a more natural objective for the adversarial attack depending on the context. For Pascal VOC, there is a *background* class, which can be chosen as the target class for the whole image. This means that a successful targeted attack would produce a segmentation with no object of interest.

In contrast, Cityscapes does not consider a *background* class, so we need a plausible target. A strategy explored in [24] is to *erase* a class and select the nearest neighbor that is not from the same class as the target label. This produces natural segmentation labels given the context, but it is not clear which class should be erased in general. For

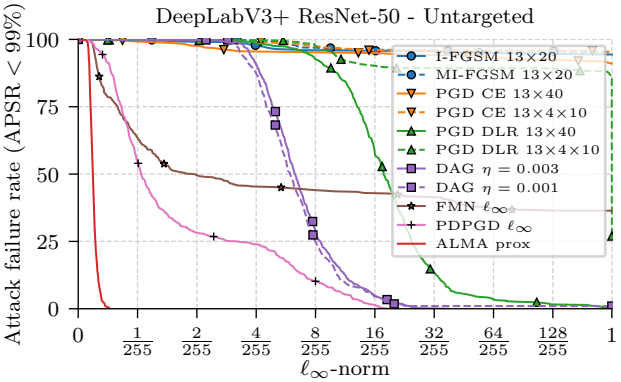


Figure 3. Percentage of unsuccessful untargeted attacks for DeepLabV3+ ResNet-50 on Cityscapes. Horizontal axis is linear on $[0, 2/255]$ and logarithmic on $[2/255, 1]$.

our experiments, we compute a target label based on the majority label for each pixel over the whole training set. This produces a more natural looking segmentation, but with high spatial frequencies, which differs from the dataset segmentation labels. Therefore, we draw a smoothed version of this target, avoiding the high frequencies, while keeping the structure. This target is provided in Appendix E.

5.5. Attacks

For the minimization attacks, we set a 500 iterations budget, corresponding to a ∼24 hours run-time to attack the entire validation set of Cityscapes with the largest model.

DAG The main baseline in our experiments is DAG [39]. We report results with two step-sizes (0.003 and 0.001) such that the larger does not fail to find adversarial perturbations, while the smaller finds smaller perturbations for some samples, but fails on others.

Other baselines As mentioned in Section 2, we can adapt some attacks originally designed for classification. We consider I-FGSM [27] and MI-FGSM [22] with 20 iterations and use a 13 steps binary search on the ℓ_∞-norm, yielding an error of $2^{-13} \approx 10^{-4}$, to find the smallest norm for which the attack succeeds. Similarly, we include four variants of the PGD attack [29] with a binary search. These variants use the Cross-Entropy (CE) or the DLR loss from [21], in combination with 40 steps or 10 steps with 3 random restarts (totaling 40 steps).[2] We also adapt the ℓ_∞ variant of FMN [34] by taking the loss as the average over the mask $\boldsymbol{m}$ and testing if $\text{APSR} \geq \nu$ as the binary decision for the projection step. We use $\alpha = 10$ in FMN (see section 3.1 of [34]). Finally, we modify the PDPGD attack [30] to take into account the constraint masking strategies. We provide the details of the modifications, as well as an ablation study on the impact of these strategies in Appendix F. We could not include the Houdini attack [13] as no public implementation is available.

ALMA prox For our attack, we use an initial step-size $\lambda^{(0)} = 10^{-3}$ and decay it to $\lambda^{(N)} = 10^{-4}$. With a budget of 500 iterations, we set $\alpha = 0.8$, compared to 0.9 for 1 000 iterations in [35]. We set the $\mu^{(0)} = \mathbf{1}_d$ and $\rho^{(0)} = 0.01 \cdot \mathbf{1}_d$, the penalty parameter increase rate $\gamma = 2$ and the constraint scale adjustment rate to $\gamma_w = 0.02$.

6. Results

Perturbation size For each dataset, model, and scenario (untargeted and targeted), we plot the attack failure rate as a function of the perturbation size. This failure rate corresponds to the fraction of samples for which an attack has *not* found an adversarial example with $\text{APSR} \geq 99\%$. This kind of plot can be interpreted in two ways: model-centric and attack-centric. In a model-centric analysis, a more robust model will require larger perturbations to be fooled, and therefore, correspond to a curve towards the upper right. In an attack-centric analysis, a stronger attack will find smaller perturbations, and have a curve towards the lower left.

Figure 3 shows this plot for untargeted attacks on Cityscapes with DeepLabV3+ ResNet-50. To improve readability, we use a linear scale on $[0, 2/255]$ and a logarithmic scale on $[2/255, 1]$. Here, DAG needs a norm of $8/255$ to successfully fool ∼75% of the samples (with $\text{APSR} \geq 99\%$), and $24/255$ to fool 99% of the samples. In contrast, ALMA prox finds adversarial perturbations with ℓ_∞-norms smaller than $0.55/255$ for *all* samples. This contradicts the robustness results in [1], where models have mIoUs of ∼50% at $1/255$, as opposed to 2.1% at $0.55/255$ here.

Table 1 reports the median and average ℓ_∞-norm (multiplied by 255 for readability) of the perturbations produced by the attacks for a subset of the regular models: DeepLabV3+ ResNet-50 and FCN HRNetV2 W46 on Pascal VOC 2012, and DeepLabV3+ ResNet-50 and SegFormer MiT-B3 on Cityscapes. For unsuccessful attacks (*i.e.* $\text{APSR} < 99\%$), the perturbation size is considered to be 1 (*i.e.* $255/255$). Overall, the ALMA prox attack outperforms all the attacks considered in our experiments. On Pascal VOC, most attacks can find small (*e.g.* $\leq 1/255$) perturbations in the targeted scenario. This is expected because the most frequent class in Pascal VOC is the background. However, the differences are much larger in the untargeted scenario, which corresponds to predicting mostly non-background classes. On Cityscapes, the scale of the problem (∼10^6 constraints) highlights the difficulty of generating adversarial perturbations for segmentation models. PDPGD handles the untargeted scenario well, with median ℓ_∞-norms of $\sim 1/255$, but produces much larger perturbations in the targeted case. Contrarily, PDG CE 13×40 finds small targeted perturbations, but fails in

[2] Given the poor performance of these attacks on the regularly trained models, we do not evaluate them on the robust model from [41].

	Attack	Pascal VOC 2012				Cityscapes			
		DeepLabV3+ ResNet-50		FCN HRNetV2 W48		DeepLabV3+ ResNet-50		SegFormer MiT-B3	
Untargeted	I-FGSM 13×20 [27]	146.32	136.55	227.11	145.69	255.00	242.15	255.00	208.97
	MI-FGSM 13×20 [22]	195.35	145.96	255.00	157.63	255.00	244.43	255.00	228.43
	PGD CE 13×40 [29]	80.10	120.93	152.08	136.05	255.00	236.77	255.00	188.15
	PGD CE 13×4×10 [29]	24.90	74.15	66.93	118.76	255.00	244.92	255.00	231.51
	PGD DLR 13×40 [29]	7.22	26.42	11.11	23.85	17.75	24.23	84.22	102.80
	PGD DLR 13×4×10 [29]	4.42	24.99	10.46	29.75	255.00	227.89	110.82	134.41
	DAG $\eta = 0.003$ [39]	5.69	6.63	8.80	10.61	6.30	7.51	8.83	9.02
	DAG $\eta = 0.001$ [39]	5.23	8.22	8.49	14.61	5.95	9.39	8.59	53.65
	FMN ℓ_∞ [34]	0.46	38.34	0.91	46.39	1.97	96.57	1.08	6.42
	PDPGD ℓ_∞ [30]	0.73	1.77	1.52	2.43	1.06	2.82	1.39	3.05
	ALMA prox	**0.32**	**0.34**	**0.51**	**0.56**	**0.24**	**0.26**	**0.33**	**0.33**
Targeted	I-FGSM 13×20 [27]	0.47	0.50	0.59	0.64	255.00	255.00	51.73	88.24
	MI-FGSM 13×20 [22]	0.59	0.66	0.77	0.85	4.26	55.35	2.54	2.60
	PGD CE 13×40 [29]	0.37	0.43	0.50	0.54	3.49	3.71	1.87	1.92
	PGD CE 13×4×10 [29]	0.50	0.51	0.62	0.65	255.00	255.00	255.00	255.00
	PGD DLR 13×40 [29]	0.62	0.68	0.81	0.94	8.37	67.86	3.98	4.09
	PGD DLR 13×4×10 [29]	0.87	1.07	1.12	1.28	255.00	255.00	255.00	255.00
	DAG $\eta = 0.003$ [39]	4.21	4.50	5.32	5.66	11.34	12.96	9.82	10.06
	DAG $\eta = 0.001$ [39]	3.92	4.21	5.07	5.36	10.96	40.28	11.53	119.47
	FMN ℓ_∞ [34]	0.42	0.45	0.47	0.49	255.00	254.36	255.00	255.00
	PDPGD ℓ_∞ [30]	0.28	0.36	0.35	0.46	14.51	14.41	19.26	19.20
	ALMA prox	**0.25**	**0.26**	**0.32**	**0.34**	**1.15**	**1.17**	**0.65**	**0.66**

Table 1. Median and average norms $\|\boldsymbol{\delta}\|_\infty \times 255$ for each adversarial attack on Pascal VOC 2012 and Cityscapes.

	Attack	Pascal VOC 2012		Cityscapes	
Untargeted	DAG $\eta = 0.003$ [39]	12.05	25.96	16.99	25.81
	DAG $\eta = 0.001$ [39]	11.71	57.93	17.02	81.30
	FMN ℓ_∞ [34]	1.28	75.66	37.54	128.67
	PDPGD ℓ_∞ [30]	3.53	11.86	43.61	58.10
	ALMA prox	**0.78**	**2.17**	**0.81**	**1.01**
Targeted	DAG $\eta = 0.003$ [39]	7.01	7.68	31.60	69.84
	DAG $\eta = 0.001$ [39]	6.78	8.78	255.00	230.29
	FMN ℓ_∞ [34]	1.31	36.10	255.00	255.00
	PDPGD ℓ_∞ [30]	0.84	1.23	255.00	255.00
	ALMA prox	**0.52**	**0.54**	**3.64**	**3.92**

Table 2. Median and average norms $\|\boldsymbol{\delta}\|_\infty \times 255$ for each adversarial attack on the robust model DeepLabV3 DDC-AT [41].

the untargeted scenario. Finally, FMN has inconsistent performance across samples: on Cityscapes in the untargeted scenario, the low median and high average indicates that it fails to find for a large portion of the samples. This can also be seen in Figure 3, where FMN finds perturbation with ℓ_∞-norms smaller than $2/255$ for $\sim$50% of the samples, but fails for $\sim$35% of them. The results for all models can be found in Appendix H with similar trends on the other models.

Finally, the median and average perturbation norms for the robust model DeepLabV3 DDC-AT are reported in Table 2. On Pascal VOC, several attacks succeed in finding small perturbations. However, Cityscapes is, again, more challenging, especially in the targeted scenario.

Attack complexity The observed average complexities in terms of number of forward and backward propagations of the model (see Section 4 of [35]) and run-times are reported in Appendix G. The number of propagations is equal to the iteration budget for all attacks except DAG, which uses an early stopping criterion. Therefore, DAG has lower complexity on average. For smaller step-sizes, it can reach the limit of 500 iterations for some samples (*e.g.* for SegFormer models on Cityscapes, see Figure 7) and fails to find adversarial perturbations. The second observation is that the proximity operator used in ALMA prox does not significantly increase the run-time compared to the other attacks with similar number of forward and backward propagations.

7. Conclusion

We proposed an adversarial attack for deep semantic segmentation models to produce minimal perturbations w.r.t. the ℓ_∞-norm. Our attack is based on an Augmented Lagrangian method, which allows us to tackle large numbers of misclassification constraints using gradient-based optimization, coupled with a proximal splitting of the objective to minimize the ℓ_∞-norm and satisfy the input space constraints with a proximity operator. Additionally, we devised an efficient method to compute this proximity operator, which is compatible with a VMFB acceleration based on a diagonal metric. Our attack offers significant improvements in terms of ℓ_∞-norm minimization for segmentation tasks, even for a robust model. One limitation of our method is that it does not produce valid digitized images, *i.e.* encoded with a reduced number of bits, such as 8. For a discussion on this issue, see [7]. Note that all the attacks considered in our experiments do not produce valid images as well.

References

[1] Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 888–897, 2018. 2, 6, 7

[2] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1):91–129, 2013. 4

[3] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012. 4

[4] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011. 4

[5] Stephen Becker and Jalal Fadili. A quasi-newton proximal splitting method. *Advances in neural information processing systems*, 25, 2012. 2

[6] Ernesto G Birgin and José Mario Martínez. *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014. 4

[7] Benoît Bonnet, Teddy Furon, and Patrick Bas. What if adversarial samples were digital images? In *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '20, page 55–66, New York, NY, USA, 2020. Association for Computing Machinery. 8

[8] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems*, 2019. 1

[9] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57. IEEE, 2017. 1, 3

[10] Antonin Chambolle and Charles H Dossal. On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm". *Journal of Optimization theory and Applications*, 166(3):968–982, 2015. 6

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision*, pages 801–818, 2018. 1, 6, 12

[12] Emilie Chouzenoux, Jean-Christophe Pesquet, and Audrey Repetti. Variable metric forward–backward algorithm for minimizing the sum of a differentiable function and a convex function. *Journal of Optimization Theory and Applications*, 162(1):107–132, 2014. 5

[13] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1, 2, 7

[14] Patrick L Combettes, Đinh Dũng, and Bằng Công Vũ. Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, 18(3):373–404, 2010. 5

[15] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011. 3, 4

[16] Patrick L Combettes and Bằng C Vũ. Variable metric forward–backward splitting with applications to monotone inclusions in duality. *Optimization*, 63(9):1289–1318, 2014. 5

[17] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 4(4):1168–1200, 2005. 4

[18] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 6

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016. 1, 6

[20] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2020. 1, 2, 3

[21] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020. 1, 3, 7

[22] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 7, 8, 14, 15

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 1, 6

[24] Volker Fischer, Mummadi Chaithanya Kumar, Jan Hendrik Metzen, and Thomas Brox. Adversarial examples for semantic image segmentation. In *International Conference on Learning Representations, Workshop Track Proceedings*, 2017. 2, 6

[25] Xu Kang, Bin Song, Xiaojiang Du, and Mohsen Guizani. Adversarial attacks for image segmentation on multiple lightweight models. *IEEE Access*, 8:31359–31370, 2020. 2

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 5

[27] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017. 1, 2, 7, 8, 14, 15

[28] Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979. 6

[29] 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. 2, 7, 8, 14, 15

[30] Alexander Matyasko and Lap-Pui Chau. Pdpgd: Primal-dual proximal gradient descent adversarial attack. *arXiv preprint arXiv:2106.01538*, 2021. 1, 2, 7, 8, 13, 14, 15

[31] Peter Melchior, Rémy Joseph, and Fred Moolekamp. Proximal adam: robust adaptive update scheme for constrained optimization. *arXiv preprint arXiv:1910.10094*, 2019. 2

[32] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2

[33] Utku Ozbulak, Arnout Van Messem, and Wesley De Neve. Impact of adversarial examples on deep learning models for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 300–308. Springer, 2019. 1, 2

[34] Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. Fast minimum-norm adversarial attacks through adaptive norm constraints. In *Advances in Neural Information Processing Systems*, 2021. 1, 2, 7, 8, 14, 15

[35] Jérôme Rony, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. Augmented lagrangian adversarial attacks. In *International Conference on Computer Vision*, 2021. 1, 3, 7, 8

[36] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[37] Marion Savanier, Emilie Chouzenoux, Jean-Christophe Pesquet, and Cyril Riddell. Unmatched preconditioning of the proximal gradient algorithm. *IEEE Signal Processing Letters*, 2022. 2

[38] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3349–3364, 2020. 1, 6, 12

[39] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *International Conference on Computer Vision*, 2017. 1, 2, 6, 7, 8, 11, 14, 15

[40] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems*, volume 34, 2021. 1, 6, 12

[41] Xiaogang Xu, Hengshuang Zhao, and Jiaya Jia. Dynamic divide-and-conquer adversarial training for robust semantic segmentation. In *International Conference on Computer Vision*, 2021. 1, 6, 7, 8, 12

[42] Zhewei Yao, Amir Gholami, Peng Xu, Kurt Keutzer, and Michael W Mahoney. Trust region based adversarial attack on neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 3