DYNAMIC DIVIDE-AND-CONQUER ADVERSARIAL TRAINING FOR ROBUST SEMANTIC SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial training is promising for improving the robustness of deep neural networks towards adversarial perturbations, especially on the classification task. The effect of this type of training on semantic segmentation, contrarily, just commences. We make the initial attempt to explore the defense strategy on semantic segmentation by formulating a general adversarial training procedure that can perform decently on both adversarial and clean samples. We propose a dynamic divide-and-conquer adversarial training (DDC-AT) strategy to enhance the defense effect, by setting additional branches in the target model during training, and dealing with pixels with diverse properties towards adversarial perturbation. Our dynamical division mechanism divides pixels into multiple branches automatically, achieved by unsupervised learning. Note all these additional branches can be abandoned during inference and thus leave no extra parameter and computation cost. Extensive experiments with various segmentation models are conducted on PASCAL VOC 2012 and Cityscapes datasets, in which DDC-AT yields satisfying performance under both white- and black-box attacks.

1 INTRODUCTION

Recent work has revealed that deep learning models, especially in the classification task, are often vulnerable to adversarial samples (Szegedy et al., 2013; Goodfellow et al., 2014; Papernot et al., 2016b). Adversarial attack can deceive the target model by generating crafted adversarial perturbations on original clean samples. Such perturbations are often imperceptible. Meanwhile, such threat also exists in semantic segmentation (Xie et al., 2017b; Metzen et al., 2017; Arnab et al., 2018), as shown in Fig. 1. However, there is seldom work to improve the robustness of semantic segmentation networks. As a universal approach, *adversarial training* (Goodfellow et al., 2014; Kurakin et al., 2016b; Madry et al., 2017) is effective to enhance the target model in classification by training models with adversarial samples. In this paper, we study the effect of adversarial training on the semantic segmentation task. We find that adversarial training impedes convergence on clean samples, which also happens in classification. Thus we set our goal as making networks perform well on adversarial examples and meanwhile maintaining good performance on clean samples.

For the semantic segmentation task, each pixel has one classification output. Thus the property of every pixel in one image toward adversarial perturbations might be different. Based on this motivation, we design a dynamic divide-and-conquer adversarial training (DDC-AT) strategy. We propose to use multiple branches in the target model during training, each handling pixels with a set of properties. During training, a "main branch" is adopted to deal with pixels from adversarial samples and pixels from clean samples that are not likely to be perturbed; an "auxiliary branch" is utilized to deal with pixels from clean samples that are sensitive to perturbations.

(a) Image (b) No Defense (c) With Our Defense (d) Image (e) No Defense (f) With Our Defense

Figure 1: For each image in (a)/(d), the left side is the normal data while the right side is perturbed by adversarial noise. (b)/(e) shows that adversarial attack could fail existing segmentation models. We provide an effective defense strategy shown in (c)/(f).

Moreover, such divide-and-conquer setting is dynamical. During training, pixels stay near the decision boundary from clean samples are initially set to the "auxiliary branch". They become more insensitive to perturbations in the network, and finally move back to the "main branch" for processing. Such dynamical procedure is implemented by training a "mask branch" via unsupervised learning. With this mechanism, our method reduces performance deterioration over clean samples. Experiments manifest that such mechanism also improves robustness towards adversarial samples. Another notable advantage of our proposed DDC-AT is that branches apart from the main one can be abandoned during inference. Thus parameters and computation cost remain almost the same. We conduct extensive experiments with various segmentation models on both PASCAL VOC 2012 (Everingham et al., 2010) and Cityscapes (Cordts et al., 2016) datasets. It is validated that our standard adversarial training strategy is effective to improve the robustness of semantic segmentation networks, and our new DDC-AT strategy further boost the effectiveness of defense. It yields superior performance under both white- and blackbox attack. Our main contribution is threefold.

- It is the first attempt to have comprehensive exploration on the effect of adversarial training for semantic segmentation. Our standard adversarial training can be treated as a strong baseline to evaluate defense strategies for semantic segmentation networks.
- We propose the DDC-AT to notably improve the defense performance of segmentation networks on both clean and adversarial samples.
- We conduct experiments with various model structures on different datasets, which manifest the effectiveness and generality of DDC-AT.

2 Related Work

Adversarial Attack Adversarial attack can be divided into two categories of white-box attack (Athalye & Carlini, 2018; Goodfellow et al., 2014), where attackers have complete knowledge of the target model, and black-box attack (Papernot et al., 2017; 2016a), where attackers have almost no knowledge of the target model. Existing adversarial attack methods focus on solving image classification problems. Such attack is normally achieved by computing or simulating the gradient information of target models (Goodfellow et al., 2014; Tramèr et al., 2017; Dong et al., 2018; Kurakin et al., 2016a). Meanwhile, as indicated by recent works (Xie et al., 2017b; Metzen et al., 2017; Arnab et al., 2018), semantic segmentation networks are also vulnerable to adversarial samples.

Adversarial Defense Current defense methods for the classification task can be divided into four kinds. 1) Changing the input of networks to remove perturbation (Jia et al., 2019). 2) Adopting random strategy to obtain correct output (Xie et al., 2017a). 3) Designing robust structures for different tasks (Xie et al., 2019). 4) Adversarial training, which adds adversarial samples into training procedure (Kurakin et al., 2016b; Tramèr et al., 2017; Song et al., 2018) and can improve robustness of networks to a certain degree. Goodfellow et al. (Goodfellow et al., 2014) first increased the robustness by feeding the model with both original and adversarial samples, and many researchers proposed follow-up work (Tramèr et al., 2017; Cai et al., 2018; Kannan et al., 2018; Wang & Zhang, 2019; Zhang & Wang, 2019).

On the other hand, no literature exists yet to improve robustness of semantic segmentation networks against various types of adversarial perturbations, without extra data during training and extra model parameters during testing. e.g., Xiao et al. (Xiao et al., 2018) proposed defense methods that aim at the detection of adversarial regions while detection only is not enough since the model still gives incorrect prediction; several methods improve the robustness of networks with extra data (Klingner et al., 2020; Mao et al., 2020; Bar et al., 2019). We advocate that models should accomplish correct outputs for adversarial samples during inference without extra training data and model parameters, and adversarial training is such a universal method while there is still no comprehensive exploration for its effects on the semantic segmentation task.

3 Standard Adversarial Attack and Training

Given a semantic segmentation network f and an input x, the segmentation output is $o = f(x)$, where $x \in \mathbb{R}^{H\times W\times 3}$ and $o \in \mathbb{R}^{H\times W\times K}$ – H, W and K are the height, width and number of

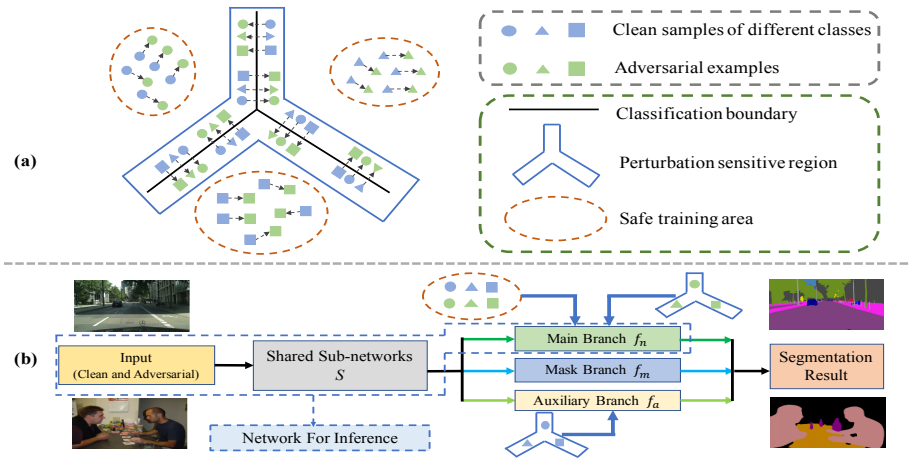

Figure 2: Motivation and overall framework of DDC-AT. (a) Clean pixels in the output space are divided into two categories by divide-and-conquer strategy. (b) Main branch f_n is utilized to conquer adversarial pixels, and clean pixels stay far away from classification boundary. The auxiliary branch f_a is employed to conquer clean pixels that are sensitive to perturbation. The mask branch f_m divides pixels into these two branches dynamically. The final output o is combined from the division during training. In testing, both f_a and f_m are abandoned, and only f_n is utilized to output o_n.

classes respectively. For a clean sample x^{clean}, pixel $x^{clean}(i,j)$ is called "clean pixel"; for the adversarial sample x^{adv}, which is obtained by adding perturbation on x^{clean}, pixel $x^{adv}(i,j)$ is called "adversarial pixel", paired with $x^{clean}(i,j)$. Moreover, the standard cross-entropy loss is denoted as $\mathcal{L}(f(x),y)$, where y is the one-hot label of x. Adversarial sample for f can be generated by computing the gradient information of f (Arnab et al., 2018; Goodfellow et al., 2014). Further, adversarial attack is often iteratively implemented and BIM (Kurakin et al., 2016a) is such an approach – it has parameters for perturbation range ϵ, step range α, and start with $x^{adv_0} = x^{clean}$ – as:

$$x^{adv_{t+1}} = clip^{\epsilon}(x^{adv_t} + \alpha \times sign(\bigtriangledown_{x^{adv_t}}(\mathcal{L}(f(x^{adv_t}),y))), \tag{1}$$

where x^{adv_t} is the adversarial sample after the t-th attack step, function $clip^{\epsilon}()$ forces its output to reside in the range of $[x^{clean}-\epsilon, x^{clean}+\epsilon]$, $sign()$ is the sign function and $\bigtriangledown_a(b)$ is the matrix derivative of b with respect to a.

We first design our standard adversarial training (SAT) on semantic segmentation task. We find that models trained with adversarial samples only largely drop performance on clean samples, and this leads to the phenomenon that the results on adversarial samples are better than those on clean samples. This phenomenon is called "label leaking" (Kurakin et al., 2016b). Models with "label leaking" are not suitable for the evaluation of robustness. Thus, to ensure the performance on both clean/adversarial samples and avoid label leaking, we use mixed data where clean and adversarial samples are equally included in each batch during training. This mixed strategy can scale up adversarial training to large models and datasets in classification (Kurakin et al., 2016b). It also works for semantic segmentation. SAT yields reasonable defense effects on various datasets as in the experimental part and the detailed procedure of SAT is listed in the Sec. A.2 of appendix.

4 Dynamic Divide-and-Conquer Adversarial Training

To further boost the robustness of semantic segmentation networks, we then propose a novel and much more effective strategy named dynamic divide-and-conquer adversarial training (DDC-AT).

4.1 Divide-and-Conquer Procedure

DDC-AT adopts divide-and-conquer procedure during training, as shown in Fig. 2 (b) and explained as the following. 1) Divide: for an input image x, DDC-AT divides its pixels into two sub-tasks for two branches respectively. 2) Conquer: each branch predicts labels for the pixels assigned to it. 3) Merge: predictions from two branches are merged into the final prediction of image x.

Algorithm 1 Dynamic divide-and-conquer adversarial training for semantic segmentation networks

Parameter: clean training set $\mathbf{X}$, shared backbone S, main branch f_n, auxiliary branch f_a, mask branch f_m, training batch size m, and maximum training iteration T_{max}, Number of iterations $T \leftarrow 0$

1: **while** $T \neq T_{max}$ **do**
2: **Load** a mini-batch of data $\mathbf{D}_b = \{x_1^{clean}, ..., x_b^{clean}\}$ from $\mathbf{X}$ with one-hot labels $\mathbf{Y}_b = \{y_1, ..., y_b\}$.
3: **Use** the current state of network $\{S, f_n, f_a, f_m\}$, $\mathbf{D}_b$, and $\mathbf{Y}_b$ to generate adversarial examples as $\mathbf{A}_b = \{x_1^{adv}, ..., x_b^{adv}\}$, and obtain "mask label" for $\mathbf{D}_b$ and $\mathbf{A}_b$ as $\mathbf{M}_b^{clean} = \{M_1^{clean}, ..., M_b^{clean}\}$ and $\mathbf{M}_b^{adv} = \{M_1^{adv}, ..., M_b^{adv}\}$.
4: **Compute** output from f_m for $\mathbf{D}_b$, and obtain the label map $\{p_1^{clean}, ..., p_b^{clean}\}$.
5: **Compute** output from f_m for $\mathbf{A}_b$, and obtain the label map $\{p_1^{adv}, ..., p_b^{adv}\}$.
6: **Compute** $\{q_1^{clean}, ..., q_b^{clean}\}$ and $\{q_1^{adv}, ..., q_b^{adv}\}$, where $q_i^{clean} = 1 - p_i^{clean}$, $q_i^{adv} = 1 - p_i^{adv}$.
7: $\mathbf{T}_b = \{x_1^{clean}, ..., x_{\lfloor b/2 \rfloor}^{clean}, x_{\lfloor b/2 \rfloor+1}^{adv}, ..., x_b^{adv}\}$, $\mathbf{M}_b = \{M_1^{clean}, ..., M_{\lfloor b/2 \rfloor}^{clean}, M_{\lfloor b/2 \rfloor+1}^{adv}, ..., M_b^{adv}\}$, $\mathbf{P}_b = \{p_1^{clean}, ..., p_{\lfloor b/2 \rfloor}^{clean}, p_{\lfloor b/2 \rfloor+1}^{adv}, ..., p_b^{adv}\}$, $\mathbf{Q}_b = \{q_1^{clean}, ..., q_{\lfloor b/2 \rfloor}^{clean}, q_{\lfloor b/2 \rfloor+1}^{adv}, ..., q_b^{adv}\}$.
8: **Compute** loss by equation 2 with $\mathbf{T}_b$, $\mathbf{Y}_b$, $\mathbf{P}_b$ and $\mathbf{Q}_b$. Update weights of network $\{S, f_n, f_a\}$.
9: **Compute** loss by equation 3 using $\mathbf{T}_b$ and $\mathbf{M}_b$. Update weights of $\{S, f_m\}$. $T \leftarrow T + 1$
10: **end while**

Dividing Pixels As shown in Fig. 2 (a), clean pixels in the output space can be divided into two types during training, according to their "boundary property".

1) Pixels $\mathcal{A}$ without "boundary property": clean pixels and their paired adversarial pixels are in the same classification space (in the "Safe Training Area"). The properties of clean and adversarial pixels are similar in this output space. They are likely to stay far away from the boundary. Their distribution can be aligned in an identical branch with adversarial training.

2) Pixels $\mathcal{B}$ with "boundary property": clean pixels and their paired adversarial pixels are in diverse classification spaces. Such clean pixels are likely to stay near the classification boundary (in the "Perturbation Sensitive Region"). They have "boundary property" since they are easy to be perturbed through the boundary. Directly aligning them with the adversarial pixels in the identical branch is difficult, since the distribution is complex. Thus, we propose to first use two different branches to train them respectively. Once the clean and their adversarial pixels stay in the same space, we use an identical branch to align them.

In short, we divide pixels in one clean image into different kinds according to whether they have "boundary property". The "boundary property" describes whether clean pixels and the corresponding adversarial pixels have different predictions or not. For semantic segmentation, normally not all pixels in a clean sample will be perturbed to have wrong predictions after adding adversarial noise. Thus, some pixels in a clean sample will have boundary property while others not.

Conquering Pixels Based on the above division setting, we set our framework as shown in Fig. 2 (b), which consists of three branches. They are "main branch", "auxiliary branch" and "mask branch", denoted by f_n, f_a and f_m respectively. f_n and f_a can be utilized to *conquer* pixels, i.e., predicting labels for pixels through forwarding the corresponding networks. We use "main branch" to conquer $\mathcal{A}$, as well as all adversarial pixels, and use "auxiliary branch" to conquer $\mathcal{B}$. In this way, clean pixels in one image after division can be processed by different branches. In additional, f_n and f_a have shared backbone, which means they help each other in the feature level. It is noteworthy that only the main branch is used for inference.

Merging Pixels As shown in Fig. 2 (b), divided pixels after conquering can be merged. This is because all pixels in one clean image are divided into f_n and f_a, and there is no overlap between the pixels assigned to f_n and f_a. Thus they can be merged into the final prediction of the input image to compute loss, according to the division. Moreover, this also indicates that the output space to decide the division should be the combination of f_n and f_a during training.

4.2 Dynamical Division and Implementation

In this section, we illustrate the dynamical property of division setting in DDC-AT, and explain how such division is achieved through unsupervised learning.

Dynamical Division During training, $\mathcal{B}$ is first set to the auxiliary branch for training. Once such clean pixels turn into $\mathcal{A}$, they move to the main branch. Since the auxiliary branch is specially designed for the training of $\mathcal{B}$, it can remove boundary property for such clean pixels effectively to ensure that more clean pixels gradually move into the main branch. In this design, the main branch finally trains all clean pixels, which stay far away from the boundary. Such mechanism effectively helps to avoid the decrease of performance on clean samples. Moreover, training adversarial pixels with $\mathcal{A}$ can improve the robustness towards adversarial perturbations for the main branch.

The Implementation DDC-AT requires to distribute all adversarial pixels into f_n, and adopts dynamical division for clean pixels. Such division can be implemented with a "mask branch" f_m.

First, f_m predicts the division for pixels as shown in Fig. 2 (b). For an input x, output from f_n, f_a, and f_m is o_n, o_a, and $o_m \in \mathbb{R}^{H\times W\times 2}$ respectively. The label map of o_m is $p \in \mathbb{R}^{H\times W}$, which is a binary matrix to decide division. $p(i,j) = 1$ means pixel $x(i,j)$ is sent to f_a. Otherwise, it moves to f_n. This operation yields the combined output for x as $o = o_a \odot p + o_n \odot (1-p)$, as shown in Fig. 2 (b). Here $\odot$ is the Hadamard product. If x^{clean} is perturbed to x^{adv}, we denote the combined output as o^{clean} and o^{adv} respectively, which are obtained in the same way.

Next, the ideal division scheme is based on these combined outputs. This scheme has a "mask label" notation $M \in \mathbb{R}^{H\times W}$. $M(i,j) = 1$ means the pixel in (i,j) location is "divided into f_a". Otherwise, it is "divided into f_n". We set the mask label for x^{clean} as M^{clean}, and denote the label map of o^{clean} and o^{adv} as B^{clean} and B^{adv} respectively. For pixel $x^{clean}(i,j)$, if $B^{clean}(i,j) \neq B^{adv}(i,j)$, it should set into f_a since it has the boundary property. In this case, we set $M^{clean}(i,j) = 1$. Otherwise, it should set into f_n, making $M^{clean}(i,j) = 0$. Besides, all adversarial pixels should be sent to f_n, and we set the mask label for x^{adv} as $M^{adv} = \mathbf{0}$ which is the matrix with all elements as zero.

M^{adv} and M^{clean} are obtained according to the ideal division rule in DDC-AT. We can use them as the ground truth to train f_m. Repeating the whole process makes f_m learn how to achieve ideal division for pixels automatically. The algorithm pipeline to obtain M^{adv} and M^{clean} is listed in the Sec. A.2 of appendix. Such training is unsupervised where the learning of f_m does not need external supervised information. In addition, since $\mathcal{B}$ will be turned into $\mathcal{A}$, they will be assigned into f_n progressively during training. Thus, finally all pixels are assigned into f_n, and the predicted mask has almost all zero values.

4.3 Overall Loss Function

For the training data x (x^{clean} or x^{adv}), its label map obtained from the mask branch is $p \in \mathbb{R}^{H\times W}$, and we set q as $q = 1 - p$. The loss of x for f_n and f_a can be written as

$$\mathcal{L}_n = \mathbb{E}\left(-\sum_{i=0}^{K-1}[y(:,:,i)\cdot\iota(f_n(x)(:,:,i))]\odot q\right), \mathcal{L}_a = \mathbb{E}\left(-\sum_{i=0}^{K-1}[y(:,:,i)\cdot\iota(f_a(x)(:,:,i))]\odot p\right), \quad (2)$$

where $\mathbb{E}$ is the operation to compute the mean value, $\iota()$ is the function of computing logarithm, $y(:,:,i)$, $f_n(x)(:,:,i)$ and $f_a(x)(:,:,i)$ are score maps with shape as $\mathbb{R}^{H\times W}$. Turn the mask label M for x into one-hot form $\widetilde{M} \in \mathbb{R}^{H\times W\times 2}$, the loss for f_m is

$$\mathcal{L}_m = \mathbb{E}\left(-\sum_{i=0}^{1}[\widetilde{M}(:,:,i)\cdot\iota(f_m(x)(:,:,i))]\right). \quad (3)$$

Combined with equation 2 and equation 3, the overall loss term is

$$\mathcal{L}_{all} = \lambda_1\mathcal{L}_n + \lambda_2\mathcal{L}_a + \lambda_3\mathcal{L}_m, \quad (4)$$

where λ_1, λ_2 and λ_3 are set to 1 in experiments. Overall training procedure is concluded in Alg. 1.

5 Experiments

The newly proposed standard adversarial training (SAT) and dynamic divide-and-conquer adversarial training (DDC-AT) are effective for robust semantic segmentation. We evaluate our method on challenging PASCAL VOC 2012 (Everingham et al., 2010) and Cityscapes (Cordts et al., 2016) datasets, with popular semantic segmentation architectures PSPNet (Zhao et al., 2017) and DeepLabv3 (Chen et al., 2017). In the following, we first show the implementation details related to training strategy and hyper-parameters, then we exhibit results on corresponding datasets.

Table 1: Evaluation under white-box attack on VOC with the mean value of mIoU reported. "No Defense" means normal training without adversarial samples. "clean" means mIoU on clean samples. Results in columns "1" to "7" are mIoUs under BIM attack (L_∞ constraint) with attack iteration number ranging from 1-7, results in the column"DeepFool" are mIoUs under DeepFool attack, results in the column "C&W" are mIoUs under C&W attack, results in the column "BIM L_2" are mIoUs under BIM attack (L_2 constraint).

	clean	Model: PSPNet									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.769	0.371	0.189	0.111	0.078	0.062	0.054	0.048	0.403	0.033	0.157
SAT	0.743	0.521	0.681	0.707	0.445	0.404	0.279	0.264	0.590	0.655	0.364
DDC-AT	0.760	**0.535**	**0.756**	**0.723**	**0.479**	**0.470**	**0.338**	**0.332**	**0.612**	**0.674**	**0.371**
	clean	Model: DeepLabv3									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.775	0.374	0.196	0.119	0.081	0.064	0.055	0.048	0.393	0.039	0.167
SAT	0.727	0.507	0.624	0.645	0.431	0.385	0.288	0.266	0.590	0.660	0.370
DDC-AT	0.752	**0.518**	**0.699**	**0.678**	**0.436**	**0.447**	**0.323**	**0.326**	**0.604**	**0.671**	**0.378**

5.1 Experimental Dataset

PASCAL VOC 2012 (with abbreviation as VOC) (Everingham et al., 2010) focuses on object segmentation. It contains 20 object classes and one class for background, with 1,464, 1,499 and 1,456 images for training, validation and testing respectively. The training set is augmented to 10,582 images in (Hariharan et al., 2015), which is also adopted. The Cityscapes (Cordts et al., 2016) dataset is collected for urban scene understanding with 19 categories. It contains high quality pixel-level annotations with 2,975, 500 and 1,525 images for training, validation and testing respectively.

5.2 Implementation Details

We choose white-box BIM attacker (by L_∞ constraint) (Kurakin et al., 2016a) to generate adversarial samples during training, since single-step attack (e.g. FGSM) is more likely to introduce "label leaking" (Kurakin et al., 2016b). The maximum perturbation value is set to $\epsilon = 0.03 \times 255$. The consideration is that perturbation can be visually noticed by human (Arnab et al., 2018) with larger values. The attack step size and number of attack iterations are set as $\alpha = 0.01 \times 255$ and $n = 3$ for training respectively. We use the *mean of class-wise intersection over union* (mIoU) as our evaluation metric. The parameters ϵ, α and n are kept constant during training. For each training mini-batch, half of the input includes adversarial samples that are dynamically decided by current model states, resulting in variance of results. For both SAT and DDC-AT, we train for one more time and report the average results as well as their standard deviations (in the Sec. A.3 of appendix).

5.3 PASCAL VOC 2012

White-Box Attack White-box attackers utilize the exact gradient information of the target model (Athalye & Carlini, 2018). Specifically, for the evaluation, we consider untargeted BIM attack (L_∞ constraint, $\epsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$) with n ranging from 1 to 7, untargeted DeepFool (L_∞ constraint, $\epsilon = 0.03 \times 255$) (Moosavi-Dezfooli et al., 2016), C&W (L_∞ constraint, $\epsilon = 0.03 \times 255$) (Carlini & Wagner, 2017), and BIM attack (L_2 constraint, n=3, $\epsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$).

The results of different defense methods on VOC are shown in Table 1. In this table, we compare our methods with the baseline (model trained with clean samples only, *i.e.*, no defense). Notably, without defense, all untargeted attacks yield sharp performance decrease. Especially, under untargeted BIM attack (L_∞ constraint), the results approach zeros when the attack iteration number is large.

For BIM attack (L_∞ constraint), Table 1 basically indicates that results on adversarial samples with large attack iteration number represent the lower bound of each method on adversarial perturbations, since the corresponding performance drops with the increase of n, and converges when n is large (actually, the mean value of mIoU does not change more than 0.025 when n is 10 or 20, compared with the results when n=7 for No Defense, SAT and DDC-AT). This leads to the conclusion that SAT is already reasonable: it improves results from 0.048 to 0.264 on PSPNet and 0.048 to 0.266 on DeepLabv3 when $n = 7$. Moreover, SAT also improves results on other different types of attacks.

Table 2: Evaluation under black-box attack on VOC. Symbolic representations are same as Table 1.

	clean	Model: PSPNet									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.769	0.433	0.240	0.148	0.106	0.078	0.060	0.058	0.466	0.156	0.209
SAT	0.743	0.584	0.565	0.535	0.513	0.471	0.449	0.415	0.640	0.685	0.587
DDC-AT	0.760	**0.596**	**0.615**	**0.564**	**0.534**	**0.486**	**0.461**	**0.437**	**0.684**	**0.705**	**0.596**
	clean	Model: DeepLabv3									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.775	0.445	0.246	0.148	0.105	0.083	0.070	0.064	0.491	0.196	0.209
SAT	0.727	0.567	0.518	0.518	0.510	0.470	0.450	0.431	0.644	0.685	0.645
DDC-AT	0.752	**0.583**	**0.604**	**0.547**	**0.526**	**0.483**	**0.460**	**0.436**	**0.687**	**0.706**	**0.659**

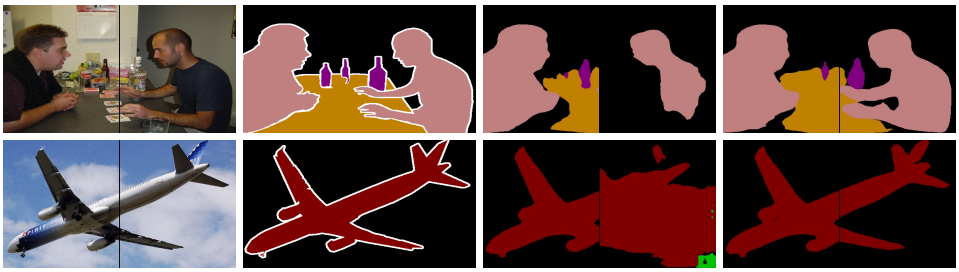

Image Ground Truth No Defense DDC-AT

Figure 3: Visual comparison on VOC. Top row is obtained from models with PSPNet, and bottom row is derived from models with DeepLabv3.

In addition, training with only adversarial samples for SAT will lead to "label leaking" phenomenon and heavily reduce performance on clean samples. For example, on the evaluation of VOC for such setting, the mean value of mIoU on clean samples is 0.722 (lower than AT with 0.743), while the result on adversarial samples with BIM attack (L_∞ constraint, n=3, $\epsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$) is 0.801 (higher than that on clean samples).

DDC-AT in Table 1 gives results of our final framework. Performance of DDC-AT on clean samples increases compared with SAT (by 0.017 and 0.025 on PSPNet and DeepLabv3 respectively), consistent with our design motivation. Further, the performance of DDC-AT is higher than SAT notably under each attacker iteration on average for BIM attack (L_∞ constraint). Intriguingly, the best case of SAT under every attack iteration is almost the worst case of DDC-AT. For example, when the attack iteration $n = 3$, we have $0.707 + 0.008 < 0.723 - 0.005$ on PSPNet and $0.645 + 0.010 < 0.678 - 0.011$ on DeepLabv3. More interestingly, for unseen attacks, DDC-AT also clearly improves robustness over SAT. We also provide visual comparison on VOC in Fig. 3 for illustrating the effectiveness of DDC-AT.

Black-Box Attack Black-box attackers cannot utilize the exact gradient information of the target model. Instead, gradient information from a substitute network, which is defensively trained on the same dataset (Papernot et al., 2017; 2016a; Liu et al., 2016), can be adopted. In our evaluation setting, the perturbation for trained PSPNet models is generated by DeepLabv3, trained on the same dataset and enhanced with adversarial training, and vice versa. For SAT and DDC-AT, the substitute networks are the same. As described in Sec. 5.2, models trained with the same method and dataset may have diverse behavior. To reduce evaluation bias from training randomness, we evaluate SAT and DDC-AT on dataset $\widehat{D}$ in the following way. Using training strategy $\widehat{S}$ (SAT or DDC-AT) with a model structure $\widehat{f}$ on $\widehat{D}$, we obtain model set $\widehat{M}_1$. Then using adversarial training with a model structure different from $\widehat{f}$ on $\widehat{D}$, we obtain model set $\widehat{M}_2$ as substitute defensive networks. Finally, for each model in $\widehat{M}_1$, we use attack generated from each model in $\widehat{M}_2$ for evaluation.

The results under black-box evaluation on VOC are included in Table 2. The performance of clean models also decreases along with the increase of attack iteration for BIM attack (L_∞ constraint), like the white-box situation. This phenomenon suggests that there is strong transferability for adversarial samples in the semantic segmentation task. It is therefore meaningful to evaluate robustness under

Table 3: Evaluation of our method and the baseline under white-box attack on Cityscapes. Symbolic representations are same as Table 1.

	clean	Model: PSPNet									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.746	0.454	0.262	0.120	0.055	0.031	0.021	0.016	0.358	0.138	0.227
SAT	0.690	0.521	0.467	0.376	0.329	0.276	0.258	0.252	0.560	0.491	0.458
DDC-AT	0.717	**0.546**	**0.502**	**0.401**	**0.347**	**0.306**	**0.287**	**0.270**	**0.572**	**0.508**	**0.467**
	clean	Model: DeepLabv3									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.748	0.458	0.260	0.122	0.057	0.033	0.023	0.018	0.315	0.138	0.226
SAT	0.694	0.520	0.461	0.365	0.318	0.279	0.262	0.246	0.567	0.484	0.450
DDC-AT	0.713	**0.546**	**0.509**	**0.403**	**0.349**	**0.309**	**0.290**	**0.273**	**0.574**	**0.505**	**0.468**

Table 4: Evaluation of our method and the baseline under black-box attack on Cityscapes. Symbolic representations are same as Table 1.

	clean	Model: PSPNet									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.746	0.476	0.280	0.141	0.069	0.039	0.033	0.022	0.356	0.211	0.253
SAT	0.690	0.511	0.444	0.399	0.367	0.320	0.308	0.291	0.577	0.578	0.566
DDC-AT	0.717	**0.561**	**0.506**	**0.425**	**0.379**	**0.339**	**0.323**	**0.306**	**0.586**	**0.584**	**0.574**
	clean	Model: DeepLabv3									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
No Defense	0.748	0.482	0.299	0.153	0.076	0.044	0.031	0.024	0.358	0.273	0.273
SAT	0.694	0.507	0.432	0.394	0.361	0.328	0.316	0.297	0.584	0.583	0.574
DDC-AT	0.713	**0.523**	**0.478**	**0.416**	**0.378**	**0.341**	**0.328**	**0.311**	**0.596**	**0.597**	**0.592**

this black-box setting. In comparison between DDC-AT and SAT, we use the same hyper-parameters as white-box attacks. From Table 2, it is clear that SAT also improves the defense effect under black-box attacks. The final performance of DDC-AT is consistently higher than SAT for BIM attack (L_∞ constraint) with attack iteration number ranging from 1 to 7, as well as other attacks.

5.4 CITYSCAPES

White-Box Attack The results of different methods on clean samples are included in Table 3. DDC-AT effectively reduces the drop of performance on clean samples, compared with SAT: DDC-AT improves mIoU on clean samples by 0.027 and 0.023, which are significant with the setting of PSPNet and DeepLabv3, compared with SAT. And we show results under white-box attack on Cityscapes dataset in Table 3. Obviously, clean models get worse with the increase of attack iterations for BIM attack (L_∞ constraint), like the case in VOC, which proves the general effect of adversarial attack on different datasets. The results of DDC-AT and SAT under various attack iterations for BIM attack (L_∞ constraint) are like we observe before – they also improve the robustness of the models on this large dataset. DDC-AT outperforms SAT in Table 3 where the best cases of SAT under every attack iteration are actually worse than the worst cases of DDC-AT. Furthermore, DDC-AT also outperforms SAT on other types of attacks in this dataset.

Black-Box Attack The results under the evaluation of black-box attack for the Cityscapes dataset are shown in Table 4. DDC-AT also outperforms SAT on average with various types of attacks.

6 CONCLUSION

In this paper, we have explored the property of adversarial training on the semantic segmentation task. Our defense strategy can consistently enhance the robustness of target models under adversarial attack. Besides proposing the standard adversarial training (SAT) process, we propose a new strategy to improve the performance of adversarial training in this task, with no extra parameter and computation cost introduced during inference. The extensive experimental results with different model structures on two representative benchmark datasets suggest that the proposed method achieves significantly better generalization and stability on unseen adversarial examples and clean samples, compared with standard adversarial training.

REFERENCES

Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. In *CVPR*, 2018.

Anish Athalye and Nicholas Carlini. On the robustness of the cvpr 2018 white-box adversarial example defenses. *arXiv:1804.03286*, 2018.

Andreas Bar, Fabian Huger, Peter Schlicht, and Tim Fingscheidt. On the robustness of redundant teacher-student frameworks for semantic segmentation. In *CVPRW*, 2019.

Qi-Zhi Cai, Min Du, Chang Liu, and Dawn Song. Curriculum adversarial training. *IJCAI*, 2018.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE symposium on security and privacy*, 2017.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

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 *CVPR*, 2016.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, 2018.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2014.

Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Hassan Foroosh. Comdefend: An efficient image compression model to defend adversarial examples. In *CVPR*, 2019.

Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv:1803.06373*, 2018.

Marvin Klingner, Andreas Bar, and Tim Fingscheidt. Improved noise and attack robustness for semantic segmentation by using multi-task training with self-supervised depth estimation. In *CVPRW*, 2020.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR*, 2016a.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ICLR*, 2016b.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv:1611.02770*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

Chengzhi Mao, Amogh Gupta, Vikram Nitin, Baishakhi Ray, Shuran Song, Junfeng Yang, and Carl Vondrick. Multitask learning strengthens adversarial robustness. *arXiv:2007.07236*, 2020.

Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *ICCV*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2016.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv:1605.07277*, 2016a.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy*, 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017.

Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Improving the generalization of adversarial training with domain adaptation. *ICLR*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *ICLR*, 2017.

Jianyu Wang and Haichao Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *ICCV*, 2019.

Chaowei Xiao, Ruizhi Deng, Bo Li, Fisher Yu, Mingyan Liu, and Dawn Song. Characterizing adversarial examples based on spatial consistency information for semantic segmentation. In *ECCV*, 2018.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *ICLR*, 2017a.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *CVPR*, 2017b.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *CVPR*, 2019.

Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *NIPS*, 2019.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

A APPENDIX

A.1 SUPERIORITY OF DIVIDE-AND-CONQUER

DDC-AT is designed to be superior than SAT and this is proved in the experiments part. It gets much better performance than classical SAT on both clean as well as adversisal pixels, and we list the detailed explanation as below.

1) For segmentation, usually not all pixels in a clean sample will be perturbed to have wrong predictions after adding adversarial noises. Thus, some pixels in a clean sample will have boundary property while others not, and mask branch f_m can divide pixels into f_n and f_a based on their boundary properties. This is the motivation for our divide strategy.

2) First, for the training of clean pixels in the main branch f_n, training with $\mathcal{A}$ only (setting of DDC-AT) is much easier than mixed training with both $\mathcal{A}$ and $\mathcal{B}$ (setting of SAT). The introduced auxiliary branch f_a in DDC-AT can turn $\mathcal{B}$ into $\mathcal{A}$ gradually and effectively. Thus, the main branch f_n that is adopted for inference can better handle clean pixels and improve accuracy over SAT.

3) Second, to obtain decent results on adversarial pixels, SAT trains adversarial pixels with both $\mathcal{A}$ and $\mathcal{B}$, while DDC-AT trains adversarial pixels with only $\mathcal{A}$ for the main branch f_n. Obviously, training with both $\mathcal{A}$ and $\mathcal{B}$ causes higher difficulty for the learning of adversarial pixels than training with $\mathcal{A}$ only. Thus, DDC-AT yields higher accuracy on adversarial pixels.

A.2 DETAILS

Details of Algorithm –The Algorithm of SAT and Obtaining Mask Label The detailed procedure of SAT is listed in Alg. 2. This algorithm yields reasonable defense effect on various datasets and meets part of our requirement as the standard adversarial training. Moreover, the pipeline to obtain the adversarial sample x^{adv}, mask label M^{adv} and M^{clean} from a clean sample x^{clean} is summarized in Alg. 3.

Algorithm 2 Standard adversarial training

Parameter: clean training set $\mathbf{X}$, segmentation network f, maximum number of training iterations T_{max}
Number of iteration $T \leftarrow 0$

1: **while** $T \neq T_{max}$ **do**
2: **Load** a mini-batch of data $\mathbf{D}_b = \{x_1^{clean}, ..., x_b^{clean}\}$ from $\mathbf{X}$ with one-hot labels $\mathbf{Y}_b = \{y_1, ..., y_b\}$.
3: **Use** the network f and the chosen attack to obtain adversarial samples $\mathbf{A}_b = \{x_1^{adv}, ..., x_b^{adv}\}$.
4: **Design** the training batch as $\mathbf{T}_b = \{x_1^{clean}, ..., x_{\lfloor b/2 \rfloor}^{clean}, x_{\lfloor b/2 \rfloor+1}^{adv}, ..., x_b^{adv}\}$ from $\mathbf{D}_b$ and $\mathbf{A}_b$.
5: **Forward** $\mathbf{T}_b$ through f to obtain outputs, compute cross-entropy loss with the outputs and $\mathbf{Y}_b$.
6: **Update** parameters of the network f by back propagation.
7: $T \leftarrow T + 1$
8: **end while**

Algorithm 3 Algorithm to obtain ground truth (mask label) for training of mask branch f_m

Parameter: clean data x^{clean}, the corresponding one-hot label y, all-zero matrix $\mathbf{0}$, matrix function $\mathcal{F} = \mathbf{1}[\mathcal{N}]$ ($\mathcal{F}(i,j) = 1$ if $\mathcal{N}(i,j)$ is True)

1: **Obtain** output o_n^{clean}, o_a^{clean}, and o_m^{clean} for x^{clean} from f_n, f_a, and f_m. Label map of o_m^{clean} is p^{clean}.
2: **Compute** combined output $o^{clean} = o_a^{clean} \odot p^{clean} + o_n^{clean} \odot (1 - p^{clean})$, its label map is B^{clean}, $B^{clean}(i,j) \in \{0, 1, ...K-1\}$.
3: **Use** loss $\mathcal{L}(o_n^{clean}, y)$ to generate adversarial examples x^{adv}.
4: **Obtain** output o_n^{adv}, o_a^{adv}, and o_m^{adv} for x^{adv} from f_n, f_a, and f_m. The label map of o_m^{adv} is p^{adv}.
5: **Compute** combined output $o^{adv} = o_a^{adv} \odot p^{adv} + o_n^{adv} \odot (1 - p^{adv})$ with label map B^{adv}, where $B^{adv}(i,j) \in \{0, 1, ...K-1\}$.
6: **Generate** "mask label" for x^{clean} as $M^{clean} = \mathbf{1}[B^{clean} \neq B^{adv}]$, $M^{clean} \in \mathbb{R}^{H \times W}$.
7: **Generate** "mask label" for x^{adv} as $M^{adv} = \mathbf{0}$ with the same shape of M^{clean}.
8: **return** M^{clean}, M^{adv}, x^{clean}, and x^{adv}.

Table 5: The architecture and initial learning rate of the model (PSPNet/DeepLabv3) for the method with no defense and SAT. The visual illustration can be seen from Fig. 4 and 5. "Base LR" denotes initial learning rate of different modules.

Module Name	**Base LR**
Backbone	-
ResNet	0.01
PPM/ASPP	0.01×10
Branch	-
Convolution	0.01×10
Batch Normalization	0.01×10
ReLU	-
Convolution	0.01×10

Table 6: The architecture and initial learning rate of the model (PSPNet/DeepLabv3), which is employed in DDC-AT. "Base LR" denotes initial learning rate of different modules. "BN" means Batch Normalization.

Module Name			**Base LR**
Shared Backbone			-
ResNet			0.01
PPM/ASPP			0.01×10
Main branch	***Auxiliary branch***	***Mask branch***	-
Convolution	Convolution	Convolution	0.01×10
BN	BN	BN	0.01×10
ReLU	ReLU	ReLU	-
Convolution	Convolution	Convolution	0.01×10

Details of Model Structures The model structures for the method with no defense, which are PSPNet (Zhao et al., 2017) and DeepLabv3 (Chen et al., 2017) here, are shown in Table 5. Different modules in corresponding models have different initial learning rates. Meanwhile, the model structure of SAT is the same as the model trained with no defense. On the other hand, we add extra segmentation branches to target models in DDC-AT during training. One important problem is where to set main branch, auxiliary branch and mask branch. In DDC-AT, these three branches are separated from the same location for both VOC and Cityscapes: for PSPNet, they are separated after the PPM module; for DeepLabv3, they are separated after the ASPP module. Thus, the structures for PSPNet and DeepLabv3 in DDC-AT are shown as Table 6. Moreover, note that the split point is close to the output of network, which is convenient to choose for models with other structures.

A.3 Experiments

Experiments–Analysis of Standard Deviation Value We report the standard deviation values for the experiments in Table 1, 2, 3, 4 here. The standard deviation values for the experiments of white-box attack on VOC are shown in Table 7. It is obvious that the standard deviation of SAT is low, which means SAT is stable. Further, smaller standard deviation for DDC-AT indicates that its results are more stable. Moreover, the standard deviation values for the experiments of black-box attack on VOC are also exhibited in Table 7. It shoud be note that the standard deviation of SAT is larger than the results by white-box attacks because black-box perturbations for each model are obtained from a set of substitute networks, which have different adversarial behaviors. Meanwhile, the standard deviation of DDC-AT is lower in all cases, especially under unseen attacks. It proves that DDC-AT is more stable and effective than SAT by all types of attack. Furthermore, the standard deviation values for the experiments of white-box attack and black-box attack on Cityscapes are simultaneously displayed in Table 7 and these results also support that both DDC-AT and SAT are stable while DDC-AT is even better.

Table 7: Evaluation under white-box attack as well as black-box attack on the datasets of VOC and Cityscapes, which reports the standard deviation value of mIoU. Symbolic representation is the same as that of Table 1.

	clean	Model: PSPNet, Dataset: VOC, Setting: White-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.005	0.042	0.018	0.008	0.029	0.032	0.032	0.032	0.014	0.012	0.041
DDC-AT	0.001	0.005	0.005	0.005	0.022	0.040	0.040	0.046	0.011	0.011	0.018
	clean	Model: DeepLabv3, Dataset: VOC, Setting: White-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.010	0.040	0.006	0.010	0.019	0.020	0.020	0.010	0.004	0.012	0.011
DDC-AT	0.001	0.006	0.013	0.011	0.005	0.030	0.012	0.020	0.004	0.004	0.001
	clean	Model: PSPNet, Dataset: VOC, Setting: Black-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.005	0.032	0.029	0.027	0.028	0.027	0.042	0.041	0.021	0.013	0.030
DDC-AT	0.001	0.021	0.017	0.017	0.018	0.031	0.039	0.040	0.003	0.001	0.008
	clean	Model: DeepLabv3, Dataset: VOC, Setting: Black-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.010	0.023	0.038	0.033	0.037	0.040	0.041	0.050	0.018	0.016	0.004
DDC-AT	0.001	0.020	0.051	0.035	0.018	0.017	0.016	0.016	0.010	0.005	0.010
	clean	Model: PSPNet, Dataset: Cityscapes, Setting: White-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.010	0.002	0.010	0.003	0.003	0.010	0.010	0.005	0.030	0.015	0.018
DDC-AT	0.001	0.001	0.002	0.002	0.002	0.003	0.003	0.005	0.001	0.001	0.001
	clean	Model: DeepLabv3, Dataset: Cityscapes, Setting: White-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.010	0.005	0.010	0.007	0.006	0.005	0.004	0.004	0.017	0.013	0.009
DDC-AT	0.003	0.003	0.004	0.004	0.002	0.002	0.002	0.002	0.002	0.015	0.004
	clean	Model: PSPNet, Dataset: Cityscapes, Setting: Black-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.010	0.005	0.030	0.026	0.033	0.023	0.026	0.025	0.024	0.020	0.025
DDC-AT	0.001	0.004	0.010	0.010	0.010	0.002	0.002	0.002	0.001	0.001	0.003
	clean	Model: DeepLabv3, Dataset: Cityscapes, Setting: Black-box									
		1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
SAT	0.010	0.008	0.030	0.028	0.030	0.022	0.023	0.025	0.011	0.018	0.023
DDC-AT	0.003	0.006	0.019	0.010	0.010	0.004	0.003	0.004	0.007	0.005	0.002

Experiments–Ablation Study The motivation of DDC-AT is to dynamically divide pixels with/without boundary property into diverse branches during training. We prove our division setting is better than other alternatives by adjusting the division setting for pixels with boundary property. Especially, the common alternatives are the following.

- Use "main branch" to deal with pixels from clean and adversarial samples without boundary property. Use "auxiliary branch" to process pixels from clean and adversarial samples with boundary property. We name this setting DDC-AT-M. Its visual explanation is shown in Fig. 6, where we also implement dynamical division by training a "mask branch".
- Use "main branch" to deal with pixels from clean samples, pixels from adversarial samples without boundary property. Use "auxiliary branch" to solve pixels from adversarial samples with boundary property. We name this setting DDC-AT-N. Its visual explanation is shown in Fig. 7, where the dynamical division is also completed by training a "mask branch".
- Use only "main branch" to deal with pixels from either clean or adversarial samples. This is what SAT does, thus we do not train the mask branch. Moreover, its visual explanation is shown in Fig. 5.
- Especially, the visual explanation for "the model trained with no defense" is shown in 4, where we only involve clean samples into training.

For all these methods, only main branch is utilized during testing. We evaluate the performances of these alternatives and list results in Table 8. For PSPNet model, the performance of DDC-AT-N is higher than SAT and lower than DDC-AT. Their standard deviations are in the same scale. The average results of DDC-AT-M are comparable with SAT and are worse than DDC-AT. Also, compared

with DDC-AT, the standard deviation increases clearly by DDC-AT-M. This is because the adversarial samples during the training are different at every training iteration, and the dynamical distribution enhances such randomness. Similarly, for DeepLabv3 model, the average results of DDC-AT-M and DDC-AT-N are lower than DDC-AT, and higher than SAT consistently. The standard deviation increases compared with DDC-AT and SAT. In summary, the division setting of DDC-AT is optimal among these common alternatives in terms of both average performance and stability measurement.

Table 8: Performance comparison of defense setting in ablation study on VOC. Symbolic representation is the same as that of Table 1. We record the mean value ("Mean") and standard deviation ("Std") of results through several repeated training process.

	Methods	clean	Model: PSPNet						
			1	2	3	4	5	6	7
Mean	SAT	0.743	0.521	0.681	0.707	0.445	0.404	0.279	0.264
	DDC-AT-M	0.751	0.511	0.690	0.690	0.441	0.463	0.304	0.318
	DDC-AT-N	0.748	0.528	0.737	0.694	0.456	0.460	0.318	0.330
	DDC-AT	0.760	**0.535**	**0.756**	**0.723**	**0.479**	**0.470**	**0.338**	**0.332**
Std	SAT	0.005	0.042	0.018	0.008	0.029	0.032	0.032	0.033
	DDC-AT-M	0.001	0.025	0.031	0.022	0.009	0.035	0.040	0.041
	DDC-AT-N	0.003	0.013	0.003	0.007	0.003	0.044	0.034	0.045
	DDC-AT	0.001	0.005	0.005	0.005	0.022	0.040	0.040	0.046
	Methods	clean	Model: DeepLabv3						
			1	2	3	4	5	6	7
Mean	SAT	0.727	0.507	0.624	0.645	0.431	0.385	0.288	0.266
	DDC-AT-M	0.741	0.505	**0.720**	0.666	**0.451**	0.435	0.314	0.304
	DDC-AT-N	0.741	0.506	0.683	0.665	0.426	0.415	0.312	0.302
	DDC-AT	0.752	**0.518**	0.699	**0.678**	0.436	**0.447**	**0.323**	**0.326**
Std	SAT	0.010	0.040	0.006	0.010	0.019	0.020	0.020	0.010
	DDC-AT-M	0.001	0.023	0.046	0.029	0.047	0.034	0.038	0.041
	DDC-AT-N	0.001	0.014	0.044	0.023	0.024	0.035	0.017	0.023
	DDC-AT	0.001	0.006	0.013	0.011	0.005	0.030	0.012	0.020

Experiments–Evaluation with Different Backbones All results in the manuscript are obtained from models with **Resnet50** (He et al., 2016) backbone. Here, we provide the comparison, when all defense methods are trained with the backbone of **Resnet101** (He et al., 2016). Similar to the situation of ResNet50 in the manuscript, we evaluate their performances on the clean samples and adversarial samples, under white-box attacks as well as black-box attacks. The results under white-box/black-box attack on VOC/Cityscapes are recorded in Table 9. From this table, we can see the performance on clean samples is improved for all defense methods, since we use ResNet101 to replace ResNet50. Meanwhile, the results of DDC-AT indicate that DDC-AT can still further improve the defense effect on clean and adversarial samples, compared with SAT.

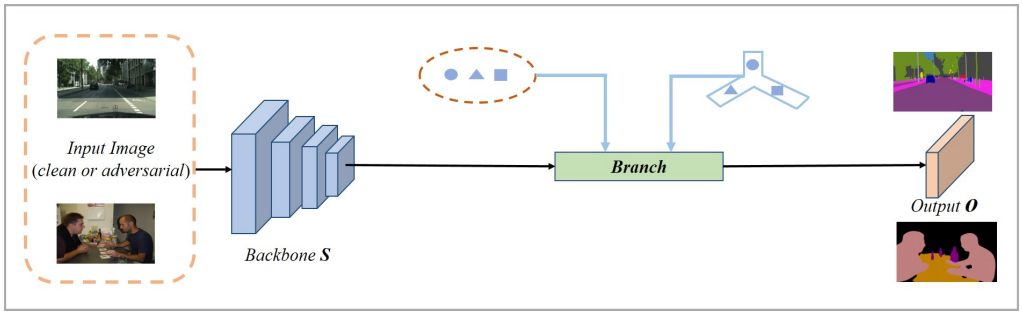

Figure 4: The training framework for the method **without defense**. Such training scheme does not add adversarial samples into training.

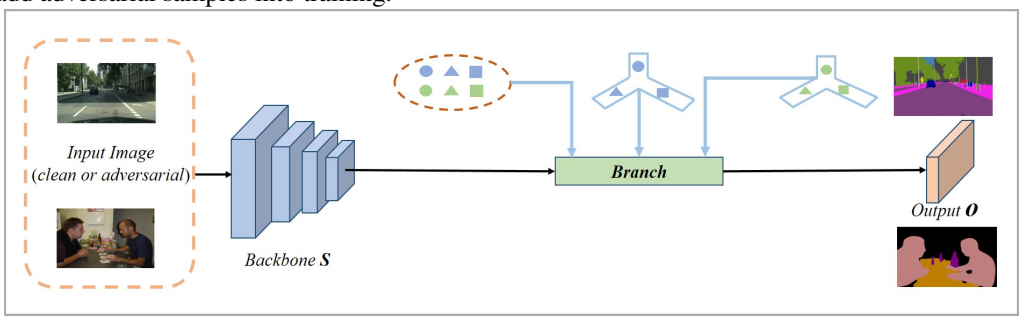

Figure 5: The training framework for standard adversarial training **SAT**. Such training scheme always uses one identical branch to align clean and adversarial samples.

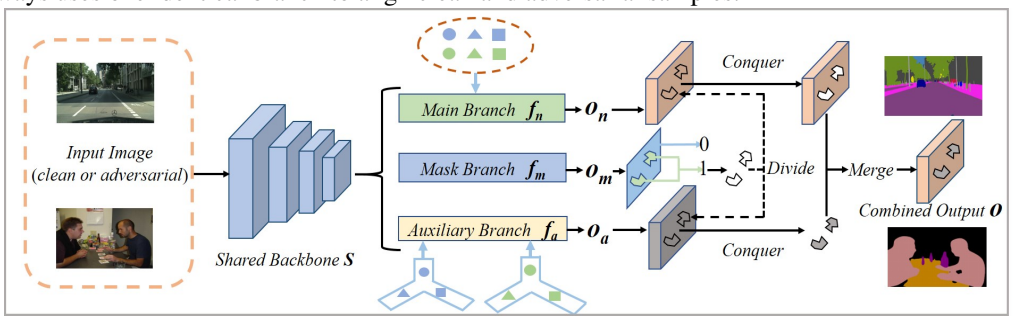

Figure 6: The training framework for **DDC-AT-M**

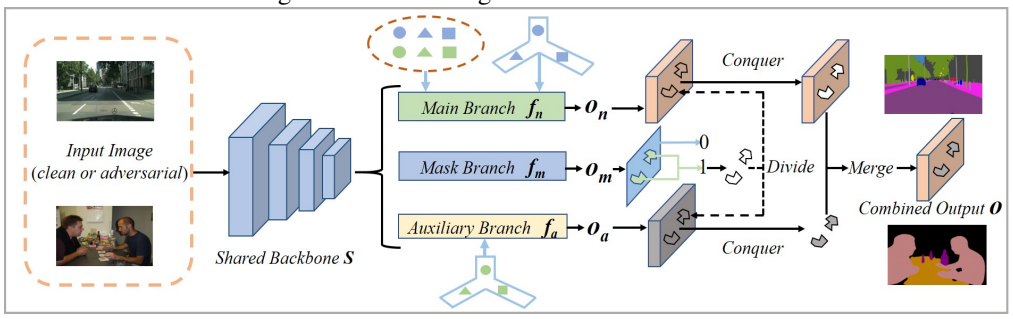

Figure 7: The training framework for **DDC-AT-N**

Table 9: The evluation of our method and the baseline on white-box/black-box attack, experiments are executed on the dataset of VOC/Cityscapes. We report the mIoU for different defense methods with various model structures. Especially, we record the mean value ("Mean") and standard deviation ("Std") of results through several repeated training process. The bold indicates the higher performance between SAT and DDC-AT. Symbolic representation is the same as that of Table 1.

Model: PSPNet, Dataset: VOC, Setting: White-box												
	Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.789	0.439	0.255	0.161	0.108	0.079	0.062	0.053	0.455	0.092	0.196
	SAT	0.761	0.535	0.710	0.715	0.457	0.490	0.366	0.341	0.609	0.648	0.370
	DDC-AT	0.772	**0.569**	**0.736**	**0.746**	**0.511**	**0.539**	**0.395**	**0.402**	**0.617**	**0.654**	**0.382**
Std	SAT	0.007	0.018	0.007	0.031	0.005	0.038	0.003	0.036	0.006	0.021	0.014
	DDC-AT	0.002	0.011	0.023	0.021	0.001	0.037	0.013	0.031	0.001	0.005	0.005
Model: DeepLabv3, Dataset: VOC, Setting: White-box												
	Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.788	0.435	0.258	0.165	0.113	0.079	0.063	0.052	0.445	0.087	0.197
	SAT	0.754	0.557	0.704	0.737	0.459	0.494	0.328	0.333	0.536	0.632	0.358
	DDC-AT	0.764	**0.588**	**0.754**	**0.772**	**0.517**	**0.512**	**0.364**	**0.374**	**0.550**	**0.640**	**0.365**
Std	SAT	0.009	0.003	0.013	0.008	0.006	0.000	0.003	0.002	0.036	0.015	0.009
	DDC-AT	0.002	0.001	0.015	0.008	0.014	0.035	0.033	0.032	0.042	0.025	0.006
Model: PSPNet, Dataset: Cityscapes, Setting: White-box												
	Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.762	0.489	0.319	0.197	0.120	0.074	0.048	0.033	0.425	0.158	0.266
	SAT	0.715	0.563	0.563	0.644	0.450	0.394	0.333	0.306	0.594	0.606	0.436
	DDC-AT	0.729	**0.581**	**0.578**	**0.695**	**0.477**	**0.420**	**0.354**	**0.330**	**0.605**	**0.613**	**0.450**
Std	SAT	0.005	0.004	0.032	0.011	0.021	0.021	0.012	0.009	0.020	0.031	0.018
	DDC-AT	0.003	0.003	0.016	0.004	0.001	0.007	0.005	0.004	0.004	0.003	0.003
Model: DeepLabv3, Dataset: Cityscapes, Setting: White-box												
	Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.765	0.484	0.317	0.189	0.114	0.070	0.046	0.032	0.395	0.145	0.261
	SAT	0.713	0.544	0.635	0.614	0.444	0.364	0.337	0.275	0.578	0.631	0.420
	DDC-AT	0.727	**0.564**	**0.694**	**0.701**	**0.502**	**0.455**	**0.376**	**0.361**	**0.585**	**0.637**	**0.430**
Std	SAT	0.010	0.023	0.014	0.014	0.020	0.010	0.021	0.024	0.030	0.006	0.030
	DDC-AT	0.007	0.002	0.001	0.007	0.004	0.004	0.006	0.004	0.004	0.001	0.001
Model: PSPNet, Dataset: VOC, Setting: Black-box												
	Defense Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.789	0.493	0.305	0.203	0.150	0.115	0.093	0.086	0.388	0.235	0.196
	SAT	0.761	0.552	0.498	0.497	0.482	0.458	0.450	0.437	0.637	0.578	0.541
	DDC-AT	0.772	**0.593**	**0.562**	**0.521**	**0.501**	**0.467**	**0.460**	**0.446**	**0.642**	**0.590**	**0.554**
Std	SAT	0.007	0.029	0.032	0.045	0.044	0.044	0.043	0.041	0.003	0.025	0.004
	DDC-AT	0.002	0.006	0.030	0.035	0.031	0.021	0.021	0.015	0.009	0.014	0.004
Model: DeepLabv3, Dataset: VOC, Setting: Black-box												
	Defense Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.788	0.515	0.318	0.215	0.164	0.129	0.106	0.094	0.388	0.229	0.199
	SAT	0.754	0.621	0.607	0.587	0.541	0.531	0.498	0.487	0.621	0.594	0.552
	DDC-AT	0.764	**0.649**	**0.650**	**0.638**	**0.574**	**0.559**	**0.525**	**0.512**	**0.630**	**0.603**	**0.578**
Std	SAT	0.009	0.006	0.020	0.031	0.026	0.028	0.032	0.032	0.012	0.024	0.004
	DDC-AT	0.002	0.006	0.022	0.026	0.011	0.011	0.016	0.007	0.001	0.007	0.004
Model: PSPNet, Dataset: Cityscapes, Setting: Black-box												
	Defense Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.762	0.506	0.345	0.216	0.139	0.097	0.064	0.057	0.387	0.129	0.184
	SAT	0.715	0.662	0.583	0.585	0.580	0.572	0.543	0.534	**0.616**	0.550	0.534
	DDC-AT	0.729	**0.704**	**0.694**	**0.699**	**0.673**	**0.654**	**0.650**	**0.631**	0.610	**0.556**	**0.546**
Std	SAT	0.005	0.022	0.021	0.019	0.017	0.018	0.019	0.018	0.002	0.029	0.003
	DDC-AT	0.003	0.003	0.005	0.003	0.006	0.010	0.002	0.007	0.002	0.010	0.003
Model: DeepLabv3, Dataset: Cityscapes, Setting: Black-box												
	Defense Method	clean	1	2	3	4	5	6	7	DeepFool	C&W	BIM L_2
Mean	No Defense	0.765	0.513	0.351	0.229	0.150	0.099	0.070	0.054	0.399	0.149	0.190
	SAT	0.713	0.573	0.547	0.544	0.513	0.466	0.445	0.416	0.632	0.606	0.557
	DDC-AT	0.727	**0.619**	**0.651**	**0.609**	**0.549**	**0.494**	**0.464**	**0.435**	**0.637**	**0.624**	**0.575**
Std	SAT	0.010	0.020	0.010	0.014	0.013	0.018	0.014	0.020	0.001	0.023	0.009
	DDC-AT	0.007	0.009	0.003	0.005	0.006	0.005	0.006	0.007	0.003	0.003	0.008

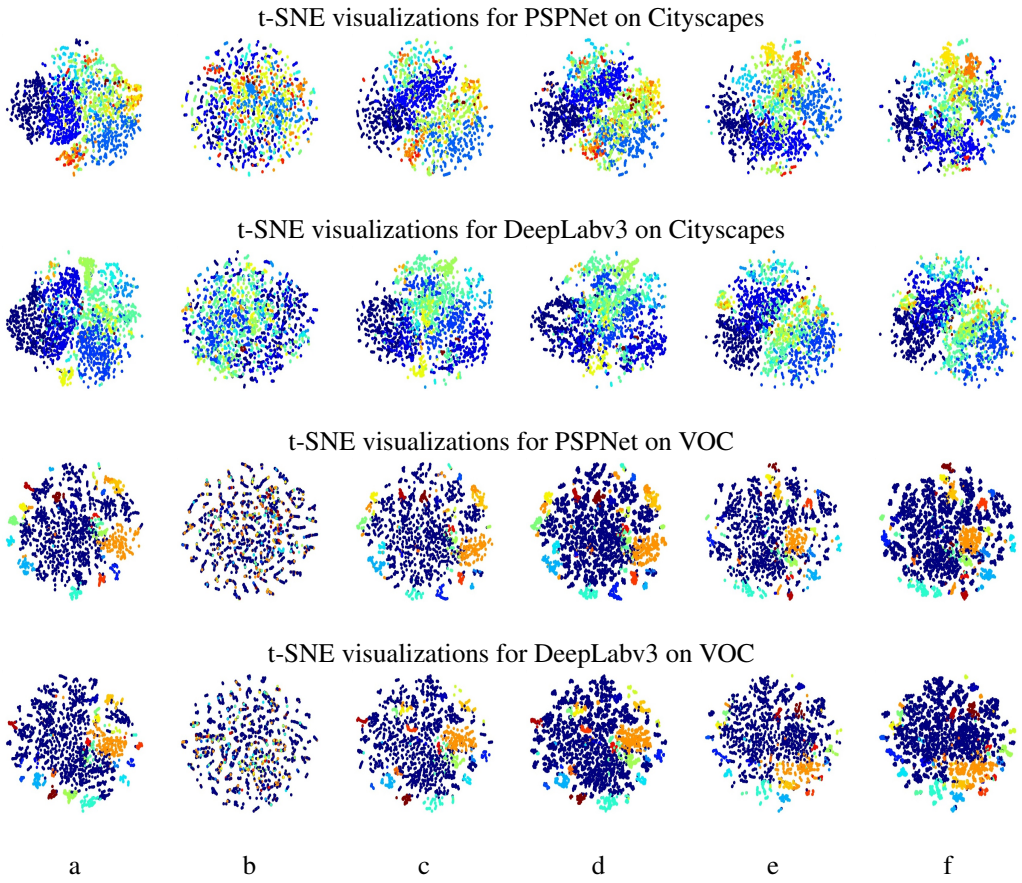

Figure 8: The t-SNE analysis for VOC and Cityscapes. a: clean samples in the model with no defense, b: adversarial samples in the model with no defense, c: clean samples in the SAT model, d: adversarial samples in the SAT model, e: clean samples in the DDC-AT model, f: adversarial samples in the DDC-AT model. The adversarial samples are generated with white-box BIM attack (L_∞ constraint, n=2, $\epsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$)

A.4 Analysis

t-SNE Analysis for SAT and DDC-AT To further analyze the defense effect of SAT and DDC-AT, we use t-SNE to visualize the corresponding feature distribution of trained models. As displayed in Fig. 8, we find: for the distribution of clean samples in models with no defense, features from different classes are separated severally. This is helpful for segmentation. However, for the distribution of adversarial samples, features from different categories are mixed. For SAT and DDC-AT, features of adversarial samples and clean samples from different classes are both separated. Thus, SAT and DDC-AT can improve the robustness of models and keep great performance on clean samples.

Visual Analysis for SAT and DDC-AT We provide visual cases from the results of models trained with No Defense, SAT and DDC-AT on VOC and Cityscapes dataset. They are shown in Fig. 9, 10, 11 and 12. All corresponding models are trained with ResNet50 as backbone, and evaluated under white-box attacks. The attacks are executed with the hyper-parameters as $\epsilon = 0.03 \times 255, \alpha = 0.01 \times 255, n = 3$. As we can see, both SAT and DDC-AT can improve the effect of defense on these datasets, while the performance of DDC-AT is better than SAT.

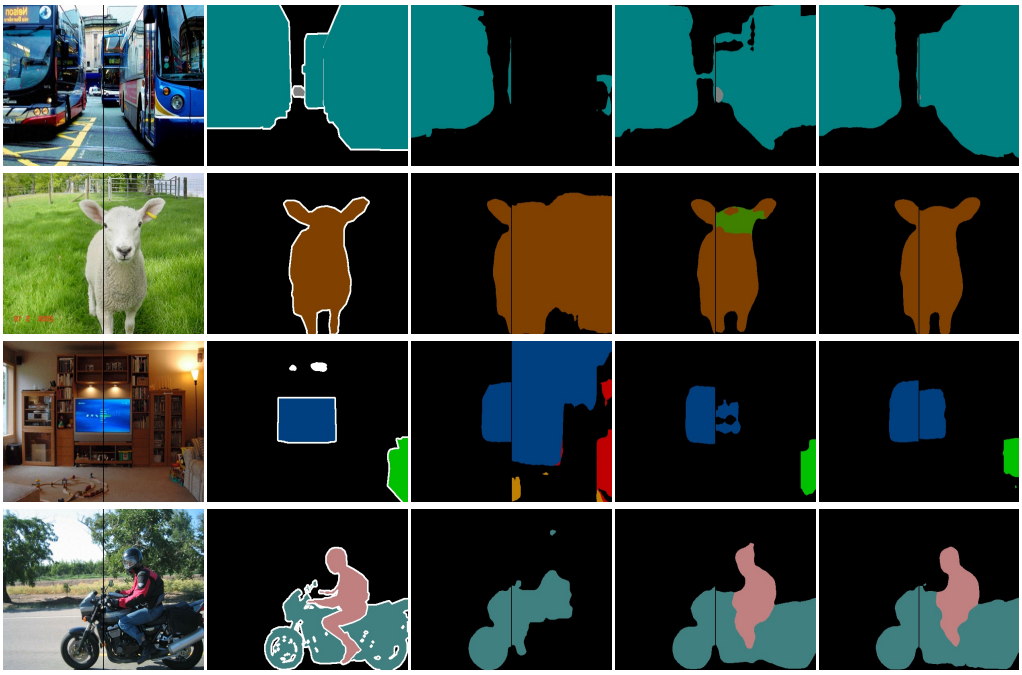

Figure 9: The visual comparison for different defense methods, which are executed on VOC dataset with model structure as PSPNet.

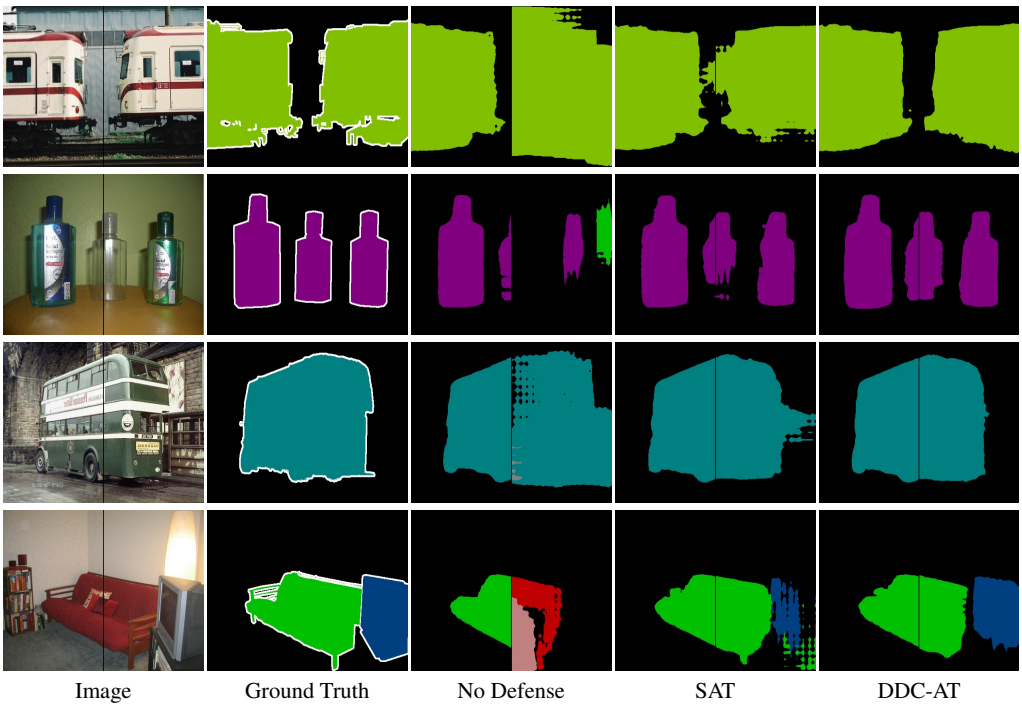

Figure 10: The visual comparison for different defense methods, which are executed on VOC dataset with model structure as DeepLabv3.

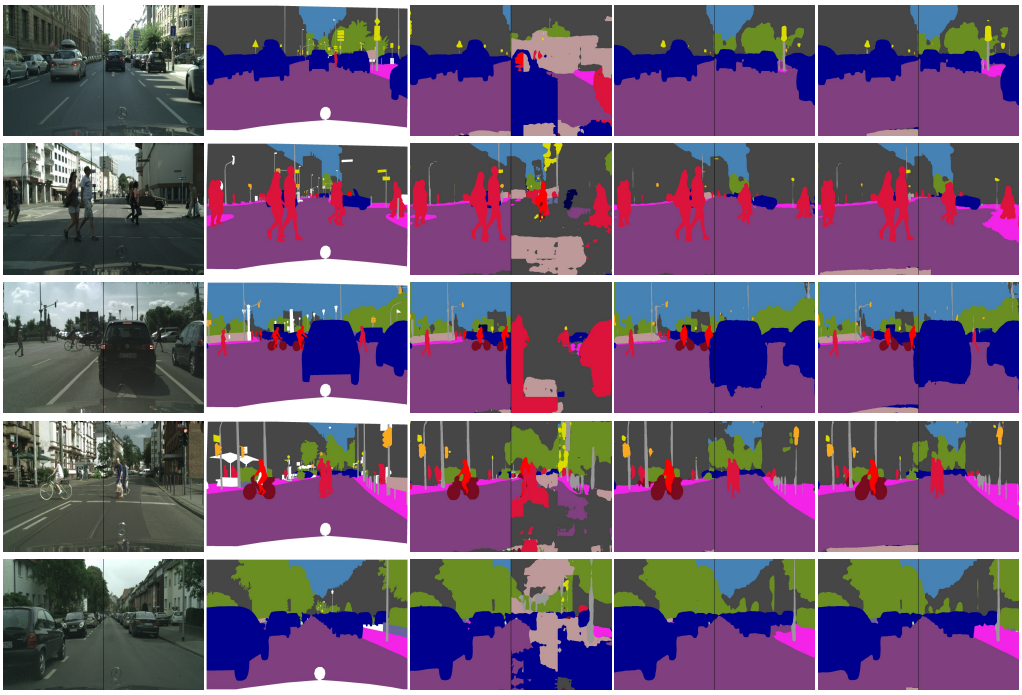

Figure 11: The visual comparison for different defense methods, which are executed on Cityscapes dataset with model structure as PSPNet.

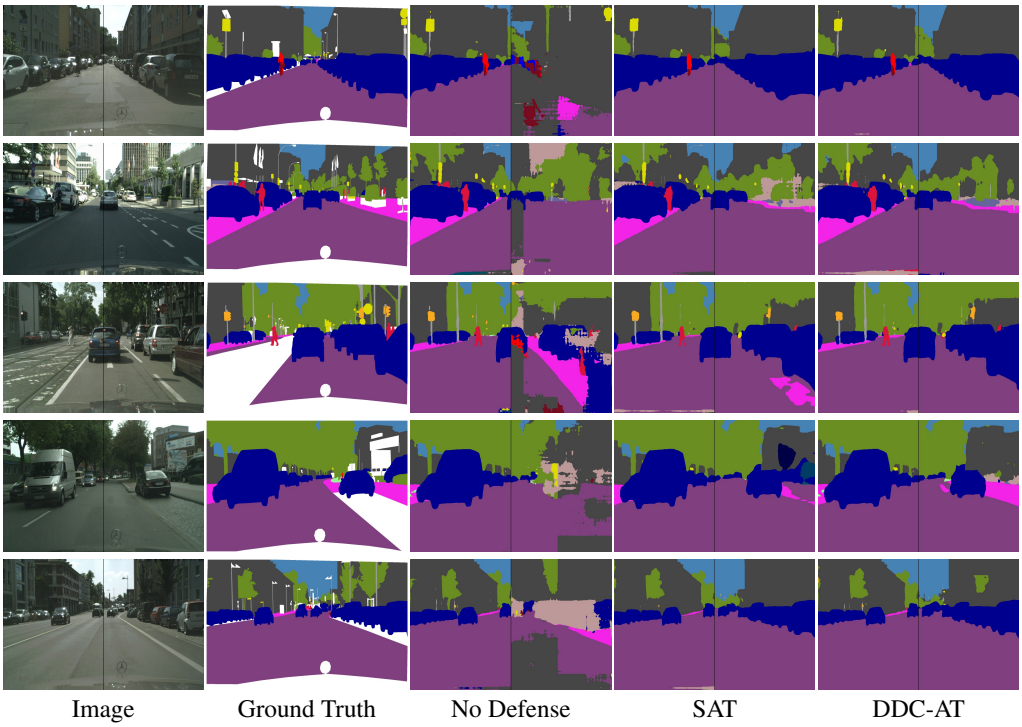

Figure 12: The visual comparison for different defense methods, which are executed on Cityscapes dataset with model structure as DeepLabv3.