Lookahead adversarial learning for near real-time semantic segmentation

Jamali-Rad, Hadi; Szabó, Attila

Important note
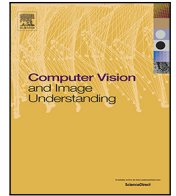To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Lookahead adversarial learning for near real-time semantic segmentation

Hadi Jamali-Rad [a,b,*], Attila Szabó [a,c]

[a] *Shell Technology Center Amsterdam (STCA), Grasweg 31, 1031HW Amsterdam, The Netherlands*
[b] *Delft University of Technology (TU Delft), 2628 XE Delft, The Netherlands*
[c] *University of Amsterdam (UvA), 1012 WX Amsterdam, The Netherlands*

ARTICLE INFO

ABSTRACT

Semantic segmentation is one of the most fundamental problems in computer vision with significant impact on a wide variety of applications. Adversarial learning is shown to be an effective approach for improving semantic segmentation quality by enforcing higher-level pixel correlations and structural information. However, state-of-the-art semantic segmentation models cannot be easily plugged into an adversarial setting because they are not designed to accommodate convergence and stability issues in adversarial networks. We bridge this gap by building a conditional adversarial network with a state-of-the-art segmentation model (DeepLabv3+) at its core. To battle the stability issues, we introduce a novel lookahead adversarial learning (LoAd) approach with an embedded label map aggregation module. We focus on semantic segmentation models that run fast at inference for near real-time field applications. Through extensive experimentation, we demonstrate that the proposed solution can alleviate divergence issues in an adversarial semantic segmentation setting and results in considerable performance improvements (+5% in some classes) on the baseline for three standard datasets.

1. Introduction and related work

Semantic segmentation is a challenging task in computer vision. It is a pivotal step towards content-based image analysis and scene understanding as it empowers machines to distinguish between different regions of an image based on its semantic context. To this aim, semantic segmentation models are trained to assign semantic labels to each and every pixel of an image as well as to cluster them into groups. Semantic segmentation has received an upsurge of attention recently owing to its wide variety of applications in medical imaging (Ronneberger et al., 2015; Rezaei et al., 2017), autonomous driving (Menze and Geiger, 2015; Cordts et al., 2016), satellite image processing (Volpi and Ferrari, 2015; Henry et al., 2018), and robotics (Geiger et al., 2013; Shvets et al., 2018), to name a few. Early segmentation methodologies are mostly developed with clustering algorithms at their core and somehow trying to incorporate contour, edge, and structural information. Examples of such algorithms are active contours (Kass et al., 1988), region-growing (Nock and Nielsen, 2004), conditional random fields (CRFs) (Plath et al., 2009), and sparse reconstruction based methods (Minaee and Wang, 2019). Recent advances in deep learning and convolutional neural networks (CNNs) revolutionized this field resulting in state-of-the-art image segmentation algorithms such as FCN (Long et al., 2015), U-Net (Ronneberger et al., 2015), PSPNet (Zhao et al., 2017), EncNet (Zhang et al., 2018a), Exfuse (Zhang et al., 2018b), DeepLabv3+ (Chen et al., 2018), PS and Panoptic DeepLab (Kirillov et al., 2019; Cheng et al., 2020), HRNet (Wang et al., 2020) and many other elegant architectures that considerably outperformed the traditional signal processing based methods addressing the same challenge.

Majority of these deep learning based methods formulate semantic segmentation as a classification problem where cross entropy (CE) with pixel independence assumption is employed as the optimization loss function. However, in practice, adjacent pixels of an image are highly correlated. These methods implicitly assume that correlation among pixels would be learned as receptive field of CNNs increases going deeper with convolutions. Recent studies challenge this assumption and argue that overlooking pixel correlations explicitly can lead to performance degradation especially with regards to capturing structural information embedded in target classes. These studies have proposed different approaches to capture pixel inter-dependencies. For instance, CRFs can be used to model pixel relationships and enforce label consistency between pixels (Liu et al., 2017; Chen et al., 2017; Shen et al., 2017; Liu et al., 2017). However, CRFs are known to be time-consuming at inference and sensitive to variations in visual appearance. An alternative approach is extracting pixel affinity information from images and fusing them back to predicted label maps (Ke et al., 2018); this comes at the cost of extra model branches and larger memory requirements. More recent studies have proposed changing the perspective and using different loss functions that encode the mutual

information or structural similarity among nearby pixels in a regional fashion (Zhao et al., 2019a; Kervadec et al., 2021; Zhao et al., 2019b) and have shown improvements. However, these losses can be derived in a sub-optimal manner by making major simplifying assumptions and by considering a small patch of pixels. To enforce preserving topology, Hu et al. (2019) proposes a new loss that enforces the ground truth and predicted label maps to have a similar topology. Inspired by perceptual losses in Johnson et al. (2016), Chen et al. (2020) proposes an edge-aware loss enabled by a proxy edge-detection network to enhance semantic segmentation precision. Along the same lines, Mosinska et al. (2018) propose a hybrid loss that is capable of capturing higher-order topological features of linear structures and helps boost the delineation of curvilinear structures.

Another avenue that has been explored to enforce structure in segmentation is employing adversarial learning (Luc et al., 2016; Souly et al., 2017; Xue et al., 2018; Hung et al., 2018). In this setup, a segmentor-discriminator pair compete to outperform each other in creating realistic label maps and distinguishing them from ground truth ones. We think a conditional adversarial approach similar to Luc et al. (2016), Souly et al. (2017) has the capacity to capture these pixel inter-dependencies and correlations in a more general (and not only local) fashion when compared to methodologies proposed in Zhao et al. (2019a,b). This is in part owing to the generic nature of the discriminator that can be trained to learn very complex structural information. Besides generality, another important aspect that makes adversarial approaches appealing for our near real-time performance criterion is the fact that the complexity of such an approach would be more tailored towards training time. As a result, our proposed solution would not impact the inference run-time performance. On the other hand, adversarial approaches are typically more difficult to analyze in behavior and are prone to stability issues, where addressing the latter is a key motivation behind the proposed work.

In medical imaging, there are studies proposing (generative) adversarial networks for image segmentation (Terzopoulos et al., 2021; Chen et al., 2018a); however, their aim is tackling domain shift in semi-supervised settings and they mostly employ U-Net type segmentors given their solid performance in this context. On the other hand, plugging state-of-the-art semantic segmentation models in an adversarial setting is prone to the well-known divergence and mode collapse issues (Arjovsky et al., 2017; Ian Goodfellow and Szegedy, 2015). Specific architecture designs for generator and discriminator networks can help to stabilize the setup, but at the cost of limiting the application domain of adversarial networks. When it comes to semantic segmentation, we are bound to architectures specifically designed to excel in doing so. When we tried to establish an adversarial network with DeepLabv3+ (Chen et al., 2018) as segmentor, we could not manage to stabilize the network regardless of remedies we pulled in. Bridging the gap between employing the state-of-the-art semantic segmentation models in adversarial settings and helping to stabilize them is the core idea of the proposed *lookahead adversarial learning* (LoAd) approach. Notably, we focus on models that run *fast* at inference time for *near real-time* field applications. That is why we opt for DeepLabv3+ base models that offer speed (*no multi-scaling (MS), no CRFs*) and performance at the same time. Nonetheless, in our ablation studies, we also demonstrate the impact of applying time-consuming post-processing steps such as MS on the performance of proposed solution.

The proposed solution (LoAd) takes inspiration from the "lookahead optimizer" (Zhang et al., 2019) and allows the semantic segmentation adversarial network to go ahead and actually diverge to some extent, then inspired by DAGGER in imitation learning (Ross et al., 2011) aggregates the degraded label maps, and steps backward to use these new sets of information and avoid further divergence. Notably, our label map aggregation strategy is different from collecting adversarial examples to retrain and stabilize generative adversarial networks (GANs) (Ian Goodfellow and Szegedy, 2015; Liu and Hsieh, 2019). We demonstrate that LoAd can alleviate divergence issues of adversarial

training in a semantic segmentation setting leading to improvement in mean-intersection-over-union (mIoU) over the baseline DeepLabv3+ on Pascal VOC 2012, Cityscapes and Stanford Background datasets. We also show that in some classes +5% improvement in IoU beyond the baseline is achieved which is quite significant. We then qualitatively demonstrate that our solution is boosting the baseline in overall segmentation performance in tackling class swap/confusion, as well as in better understanding of structure and continuity of the target classes.

The main contributions of this paper can be summarized as follows: (a) we propose a lookahead adversarial learning method (LoAd) for semantic segmentation that helps alleviate stability issues in such settings, (b) LoAd runs as fast as the baselines methods upon which it is applied, i.e., no extra delay at inference time, making it a great choice for near real-time field applications, (c) besides avoiding class confusion, LoAd improves the performance of the baseline in creating structurally more consistent label maps with significant performance boost in some classes. Enhancing state-of-the-art segmentation models that can be trained on *commodity GPUs* and run *fast at inference* time (no CRFs, no multi-scaling) is the motive of this work.

2. Conditional adversarial training for semantic segmentation

Let $\mathcal{D}_t = \{(\mathbf{X}, \mathbf{Y})_1, \ldots, (\mathbf{X}, \mathbf{Y})_M\}$ be the training dataset containing M samples with $\mathcal{X}_t = \{\mathbf{X}|(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}_t\}$ and $\mathcal{Y}_t = \{\mathbf{Y}|(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}_t\}$ respectively denoting the set of images and their corresponding label maps.[1] Here, $\mathbf{X}$ is of size $H \times W \times 3$ for RGB images with a total of $H \times W = N$ pixels. The corresponding label map $\mathbf{Y}$ is of size $H \times W$ with elements in $\mathcal{K} = \{1, \ldots, K\}$ where K is the number of classes in the segmentation task. An adaptation of conditional generative adversarial networks (CGANs) (Goodfellow et al., 2014; Mirza and Osindero, 2014; Isola et al., 2017) for semantic segmentation would not require stochastic behavior in generating semantic label maps but aims at creating the most probable map $\mathbf{Y}$ per input image $\mathbf{X}$. So, we solve a two-player min–max game to estimate $P(\mathbf{Y}|\mathbf{X})$

$$\min_G \max_D \mathcal{L}(G, D) = \mathbb{E}_{\mathbf{Y} \sim P_{\mathcal{D}_t}(\mathbf{Y})}\big[\log\big(D(\mathbf{Y}|\mathbf{X})\big)\big] + \\ \mathbb{E}_{\mathbf{Y} \sim P_g(\mathbf{Y})}\big[\log\big(1 - D(\mathbf{Y}|\mathbf{X})\big)\big], \qquad (1)$$

where $\mathcal{L}(.)$ denotes the loss function, G denotes the generator (more specifically a segmentor) parameterized with θ_g, and D stands for the discriminator parameterized with θ_d. Typically, both G and D are CNN's. This can be further simplified within a binary classification setting where the discriminator is to decide whether a sample label map is ground truth ($\mathbf{Y} \sim P_{\mathcal{D}_t}$) or generated ($\mathbf{Y} \sim P_g$) by the segmentor. Considering binary cross entropy (CE), we arrive at the following adversarial loss

$$\mathcal{L}_a(\theta_g, \theta_d) = \sum_{m=1}^{M} \big[\log\big(D(\mathbf{Y}_m|\mathbf{X}_m)\big) + \\ \log\big(1 - D(\hat{\mathbf{Y}}_m|\mathbf{X}_m)\big)\big], \qquad (2)$$

where $\hat{\mathbf{Y}}$ denotes the generated map. $\mathcal{L}_a$ should be minimized w.r.t. θ_g and maximized w.r.t θ_d. Several interesting studies such as (Luc et al., 2016; Souly et al., 2017) suggest applying a hybrid loss combining the conditional adversarial loss in (2) with a regularization or CE pixel-wise term, sometimes in a slightly different context such as weakly supervised GAN settings (Souly et al., 2017). The closest approach to our line of thought is the pioneering work in Luc et al. (2016) where the following hybrid loss $\mathcal{L}_h = \mathcal{L}_p + \lambda \mathcal{L}_a$ is proposed

$$\mathcal{L}_h = \sum_{m=1}^{M} \mathrm{CE}(\mathbf{Y}_m, \hat{\mathbf{Y}}_m) + \lambda \sum_{m=1}^{M} \log\big(D(\mathbf{Y}_m|\mathbf{X}_m)\big) + \\ \lambda \sum_{m=1}^{M} \log\big(1 - D(\hat{\mathbf{Y}}_m|\mathbf{X}_m)\big), \qquad (3)$$

---

[1] From now on, we use "map" and "label map" interchangeably.

where the pixel-wise loss $\mathcal{L}_p$ is computed using a multi-class CE between the 1-hot encoded versions of the original label map $\mathbf{Y}$ and the inferred one $\hat{\mathbf{Y}}$ using $-\sum_{i=1}^{N}\sum_{c=1}^{K} y_{i,c} \log(\hat{y}_{i,c})$, with y_i denoting the ith element of $\mathbf{Y}$. Obviously, only the second and the third terms in (3) are relevant when training the discriminator. When training the generator, Luc et al. (2016) proposes to keep the first and the third terms. Next, a standard gradient decent ascent (GDA) (Lin et al., 2020) is applied to the two-player min–max game.

We decided to take a different approach for two reasons. First, interesting findings are recently reported in Nouiehed et al. (2019), Ostrovskii et al. (2020) regarding the optimization of $\min_{x\in\mathbf{X}}\max_{y\in\mathbf{Y}} F(x,y)$ problems where $F(x,y)$ is concave in y and non-convex in x, which relates to our problem in that the loss is concave in θ_d and non-convex in θ_g in typical high dimensional settings with CNNs. Therefore, following the propositions in Nouiehed et al. (2019), Ostrovskii et al. (2020), and (to our understanding) in contrast to Luc et al. (2016), we avoid an alternating GDA in optimizing the generator and discriminator networks. Instead, in every "cycle" of the proposed adversarial approach (Algorithm 2), we keep training the discriminator with a dynamically updated dataset according to the proposed label map aggregation module (Algorithm 1) to reach sufficient accuracy before switching back to training the generator.

Second, another angle that distinguishes us from (Luc et al., 2016) is in how and when to incorporate the pixel-wise CE loss. We approach the problem in two stages as follows. Stage 1: if not pre-trained on $\mathcal{D}_t$, we first train the segmentation network using only CE pixel-wise loss up to a reasonable performance (without hard constraints). Stage 2: we then activate the adversarial loss on top the CE loss and run (LoAd) as described in Algorithm 2 to boost the performance. At this stage, when training discriminator both the second and the third terms of (3) will be active, and when training the segmentation network the first and the third terms will be used where we follow the suggestion of Luc et al. (2016), Goodfellow et al. (2014) and replace $+\lambda\sum_{m=1}^{M}\log(1-D(\hat{\mathbf{Y}}_m|\mathbf{X}_m))$ with $-\lambda\sum_{m=1}^{M}\log(D(\hat{\mathbf{Y}}_m|\mathbf{X}_m))$ to maximize the probability of $\hat{\mathbf{Y}}_m$ being the true segmentation label map of $\mathbf{X}_m$.

3. Lookahead Adversarial Learning (LoAd)

The idea. Robustness and divergence issues of GANs are not secret to anyone (Arjovsky et al., 2017; Ian Goodfellow and Szegedy, 2015; Liu and Hsieh, 2019; Roth et al., 2017; Salimans et al., 2016). Generally speaking, adversarial networks manifest the same difficulties and we had to tackle them in our semantic segmentation setup. Here is the idea behind lookahead adversarial learning (LoAd) for semantic segmentation in a nutshell. We take inspiration from "lookahead optimizer" (Zhang et al., 2019) and allow the adversarial network to *go ahead* and actually diverge (to some extent) helping us to gain new insights and construct new datasets of label maps from these divergent (or degraded) models. Inspired by the idea of DAGGER (Ross et al., 2011), we aggregate these new datasets and use them for retraining the discriminator at the end of every cycle of LoAd. Next, we go back to where the divergence started (similar to "1 step back" in lookahead optimizer) to improve our next predictions and avoid further divergence. These new datasets are not designed or generated adversarial examples but sequentially degraded label maps. Note the importance of label maps in this context. When we descend towards divergence not only our performance metric goes down (mIoU, a single score) but also destructive impacts on the generated maps provides us with new sets of information that we exploit.

The mechanics. Algorithm 1 describes the map aggregation module of LoAd and Algorithm 2 provides a pseudo code level description of LoAd itself. Let us assume a starting model $g^s = g_0$ (e.g., the ending model after Stage 1 training as explained in Section 2). First, we evaluate the model on a subset of validation data (a hold-out set) to understand our current mIoU (denoted as μ in Algorithm 2). This serves as both starting and current best mIoU (μ^s and μ^*, respectively).

We can already train our discriminator for the first time using $\mathcal{D}_t\cup g_0(\mathcal{X}_t)$, a set composed of full training data (images and maps) plus a set of generated (fake) maps. With this, we have initialized our label map aggregation buffer with $\mathcal{B} = \mathcal{B}[0] = g_0(\mathcal{X}_t)$ in Algorithm 1. Back to Algorithm 2, we then continue training adversarial until one of the following two criteria is met: (a) patience iteration counter γ reaches its maximum Γ, alerting us that it is enough looking ahead, (b) we diverge (in mIoU sense) reaching a pre-defined lower-bound ($\mu^s - \beta_l$) w.r.t. to the starting mIoU μ^s. If any of the two criteria are met, the cycle is finished, and we pick the last model of the cycle denoted by g^e. Throughout each cycle we also seek for an updated model offering a mIoU better than the staring one, and if such a new peak model g^* (above an upper-bound $\mu^s + \beta_u$) is found, the cycle would be returning two models, the best model of the cycle g^* besides its ending model g^e.

Per cycle one or both of these models $\{g^*, g^e\}$ would be passed to our map aggregation algorithm to generate new "fake" label maps which will be aggregated in $\mathcal{B}$. This dynamically updated dataset in $\mathcal{B}$ concatenated with $\mathcal{D}_t$ will then be used to retrain the discriminator before the next cycle starts. Owing to this label map aggregation and following retraining of the discriminator, we continually improve our adversarial model to avoid further divergences. At the end of each cycle, we go back and restart training adversarial from the newly found peak g^* or the old starting point g^s. Lastly, if we do more than Ψ cycles from a starting model g^s and a new peak is not found to replace it, the algorithm fully stops and returns the overall best model. Notably, the divergence patience counter γ is in practice updated in a dynamic manner (together with an auxiliary "peak finder" counter) to avoid an upward trend being stopped prematurely. Interested reader is referred to more details to the Appendix B.

Algorithm 1: Map Aggregation Buffer

Require: peak/ending models, image set $(g^p, g^e, \mathcal{X})$
if $g^p \neq 0$ (a peak is found) **then**
    $\texttt{Flush}(\mathcal{B})$
    $\mathcal{B}[0] \leftarrow g^p(\mathcal{X})$
    $\mathcal{B}[1] \leftarrow g^e(\mathcal{X})$
else
    if $\texttt{Size}(\mathcal{B}) = $ max buffer size (B_{max}) **then**
       $\texttt{Delete}(\mathcal{B}[1])$
    end
    $\mathcal{B}[\text{end}] \leftarrow g^e(\mathcal{X})$
end
Return: $\mathcal{B}$

The recap. To make this crystal clear, we use a hypothetical convergence graph in Fig. 1 (a) and corresponding dynamically updated map aggregation buffer depicted in Fig. 1 (b) to walk you through what LoAd does in action. As can be seen, starting from g_0, the first adversarial cycle immediately descends towards divergence ending with $g_{0,0}^e$. We denote the jth cycle spawned from the ith peak with $g_{i,j}$. Note that $g_{0,0}^e$ does not descend by β_l, and thus, we are assuming that the cycle is ended due to reaching patience limit of Γ propagations (or iterations) as described in Algorithm 2. This cycle also did not introduce a new peak better than g_0. Thus, only $g_{0,0}^e(\mathcal{X})$ will be added as a new set to the buffer $\mathcal{B}$. This is where we go back and restart adversarial training from g_0, but this time with a retrained discriminator. As in the figure, this helps to ascend towards g_1 after which we diverge again in the second cycle. So, the second cycle returns a new peak $g^* = g_1$ as well as the ending model $g^e = g_{0,1}^e$ for map aggregation. Since a new peak is found (better than g_0), we $\texttt{Flush}$ the buffer filling it with $\mathcal{B} = [g_1(\mathcal{X}) \mid g_{0,1}^e(\mathcal{X})]$ as shown in Fig. 1 (b). Per pseudo code in Algorithm 1, any cycle that only returns an ending model (an no new peak) would result in creating a new label map set added to the end of the buffer unless the buffer is full; i.e., it already contains B_{max} label map sets. In that case, we first $\texttt{Delete}$ the label map set corresponding to the oldest ending model and then the new label map set is added to the
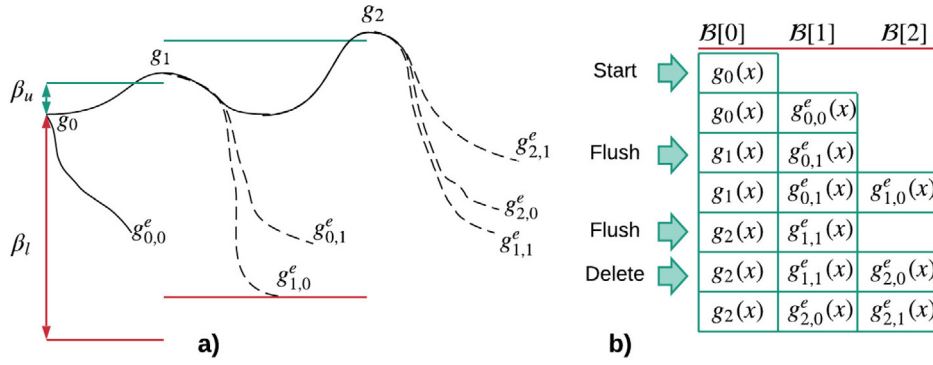
Fig. 1. (a) hypothetical convergence graph, (b) corresponding label map aggregation buffer.

Algorithm 2: LoAd for Semantic Segmentation

Initialize: $\psi = 0$, $g^s = g_0$, $\mathcal{B} = g^s(X_t)$
Input: maximum nr. of cycles: Ψ, maximum patience: Γ, β_l, β_u
$\mu^s, \mu^*, \mu \leftarrow$ evaluate mIoU
Train Discriminator$(\mathcal{D}_t \cup \mathcal{B})$
while $\psi < \Psi$ **do**
    start a divergence patience counter: $\gamma \leftarrow 0$
    while $\mu^s - \beta_l < \mu$ and $\gamma < \Gamma$ **do**
        update model: $g \leftarrow$ Train Adversarial
        $\mu \leftarrow$ evaluate mIoU
        $\mu^* \leftarrow$ best $\mu > \mu^s + \beta_u$
        update best model: $g^* \leftarrow g$
        $\gamma \leftarrow \gamma + 1$
    end
    $g^e \leftarrow$ keep the last model of the cycle
    if best model better than start **then**
        set best model as start model: $g^s \leftarrow g^*$
        reset cycle counter $\psi \leftarrow 0$
        $\mathcal{B} \leftarrow$ MapAggregation(g^*, g^e, X)
    else
        $\mathcal{B} \leftarrow$ MapAggregation$(0, g^e, X)$
        start a new cycle $\psi \leftarrow \psi + 1$
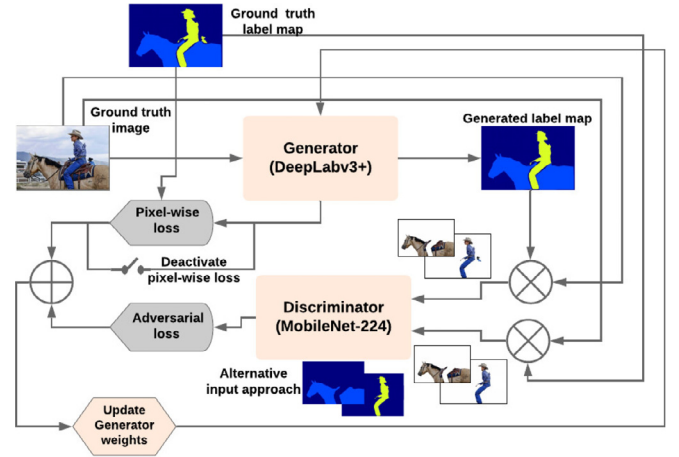   **end**
    Train Discriminator$(\mathcal{D}_t \cup \mathcal{B})$
end

---

end of $\mathcal{B}$ as described in Algorithm 1. An example of this scenario in our hypothetical setup is where the set corresponding to $g^e_{1,1}$ is deleted in favor of the newcomer set corresponding to $g^e_{2,1}$.

4. Experimental setup

Network architecture. Our experimental network architecture is shown in Fig. 2. As can be seen, we opted for DeepLabv3+ with different backbones (modified Xception-65 (Chollet, 2017) and Mobilenetv2 (Chen et al., 2018)), bearing in mind that DeepLabv3+ might not be the easiest model to simply plug into an adversarial settings. We chose Mobilenet-224 (Howard et al., 2017) as our discriminator. The figure shows our discriminator training policy being conditional on the input image split into different classes (using ground truth and generated label maps) and stacked into the input channels of the discriminator. We also considered a few other possibilities such as feeding the discriminator with split label maps themselves, which led to performance degradation. Our trainings are run separately on standard Nvidia P100 T nodes offered on Microsoft Azure each with 16 GB of memory. We focus on models that can be trained and run on standard and *commodity GPU nodes*, as many researchers do not have access to high-end TPUs. On the same note, and to re-emphasize, we

Fig. 2. Adversarial network architecture.

are particularly interested in models that *run fast at inference* time for near real-time field applications. That is why we selected DeepLabv3+ base models that offer speed (*no MS, no CRFs*) and performance at the same time.

Adversarial training. Per cycle in Algorithm 2, we train MobileNet (discriminator) until sufficient performance is reached based on an early-stopping criterion evaluated on a hold-out set explained in the following. For training the discriminator, we used a batch size of 16, and set $\alpha = 1$ with a dropout rate of 0.01 (Howard et al., 2017). We used Adagrad as optimizer with a learning rate $lr_d = 0.01$. Because of the label map aggregation, the loss function of the discriminator (binary CE) is weighted to account for the variable number of generated label maps in the map aggregation buffer. For adversarial training of the generator, we used a batch size of 5 due to the memory limitation of the GPU nodes. The adversarial learning rate was set to $lr_a = 5e - 6$, and we trained using a momentum of 0.95. Adversarial training is conducted according to LoAd (in Algorithm 2) with the following default hyperparameters: $\beta_u = 0.001$ and $\beta_l = 0.05$ corresponding to 0.1% improvement and 5% drop in mIoU in a cycle, respectively; patience counter maximum is set to $\Gamma = 50$, and maximum number of cycles allowed is set to $\Psi = 50$; maximum buffer set size is $B_{max} = 3$.

PASCAL VOC 2012 dataset. PASCAL VOC 2012 dataset (Everingham et al., 2015) is one of the most popular datasets for semantic segmentation with 20 foreground object classes and 1 background class. It contains 1464 train, 1449 validation, and 1456 test pixel-level annotated images. For the experiments on this dataset, we started from DeepLabv3+ checkpoints pre-trained on PASCAL VOC 2012 with output stride of 16 achieving mIoU = 82.2% (Xception-65 backbone) and mIoU = 75.32% (MobileNetv2 backbone) on the validation set; see,

Table 5 in Chen et al. (2018) and their model zoo on Github.[2] We then applied LoAd for adversarial training. The images have a shape of at most 512×512 pixels, so we used a crop size of 513×513 following the recommendations in Chen et al. (2018).

Cityscapes dataset. Cityscapes is one of the most commonly used large-scale datasets for semantic segmentation (Cordts et al., 2016) with 19 main target classes used for evaluation. It contains 2975 train, 500 validation, and 1525 test high quality pixel-level annotated images (a total of 5000 "fine annotation" images) used in our experimentation. For the experiments on this dataset, we started from DeepLabv3+ checkpoint pre-trained on Cityscapes with output stride of 16 achieving mIoU = 70.71% on the validation set with MobileNetv2 backbone; see their model zoo on Github referred earlier. Notably, due to considerably larger input image size (1024×2048), using Xception backbones would enforce a very small batch size for typical GPU nodes, rendering training meaningless. We applied LoAd for adversarial training with a crop size of 513×513.

Stanford Background dataset. The Stanford Background dataset (Gould et al., 2009) contains 8 classes of scene elements. It has 715 pixel-level annotated images which we have split into 400 for training, 172 for validation and 143 for testing. We chose this dataset to explore what the impact on smaller datasets (from sample size perspective) would be. For this dataset, there was no pre-trained DeepLabv3+ model available and we followed our two stage approach as explained in Section 2. We used DeepLabv3+ checkpoint from Cityscapes achieving mIoU of 78.79% (see Table 7 in Chen et al. (2018)) and trained it on Stanford Background using only pixel-wise CE loss. We set batch size to 7 and trained the model with a weighted CE loss for 4 epochs then switched to the original CE loss (all weights set to 1) and trained for another 8 epochs reaching mIoU = 74.33%. Weighted CE loss is calculated by multiplying the loss by the average number of pixels per class computed over the entire training dataset and dividing it by the number of pixel per class in the image. This model is then used as the starting point for applying and evaluating LoAd. The learning rate for the pixel-wise training was set to $lr_p = 1e - 4$ and kept constant throughout the training following (Chen et al., 2018). We did not use multi-scale training policies for DeepLabv3+ as explained earlier, and followed the recommendations of Chen et al. (2018) to upscale the logits to input image size for evaluation. Pictures in the Stanford Background dataset have a maximum pixel size of 320×320 for which we used 321×321 as crop size as suggested in Chen et al. (2018).

Augmentation, inference, and evaluation. We augmented the pictures with random flips for both pixel-wise and adversarial training stages. No special inference strategy is applied (no MS, etc.) and the evaluation metric is the mIoU score. In Section 6, we briefly reflect on and illustrate the impact of applying MS. For early stopping of the discriminator and for mIoU evaluation during adversarial training we used 30% of the validation set of each dataset as the hold-out set.

Baseline and performance comparison. We compare our performance with the non-adversarial DeepLabv3+ as our main baseline and to accentuate on the impact of the proposed conditional adversarial training (LoAd), also with the adversarial approach proposed in Luc et al. (2016). Note that Luc et al. (2016) does not employ DeepLabv3+ (but older architectures) and we had to reproduce their results by using DeepLabv3+ as segmentor. The work presented in Hung et al. (2018) could be another option; however, it is focused on *semi-supervised* adversarial segmentation and due to its different scope could not be immediately used for comparison purposes.

5. Evaluation results

In this section, we evaluate LoAd on the mentioned three datasets. For PASCAL VOC and Cityscpaes, we present performance evaluation results on their *validation* as well as *test* sets, where the latter is assessed by submitting the results to the corresponding online test server. For Stanford Background, the full dataset is available and we only present the results of the test set. Besides the baseline DeepLabv3+, we also compare the quantitative performance with the adversarial approach of Luc et al. (2016). **Remark:** for viewing the online test results, please note that PASCAL VOC server can be slow sometimes and might require patience or refreshing the page a few times.

Fig. 3 shows three examples of the adversarial training mIoU evaluated on the hold-out set of PASCAL VOC 2012, Cityscapes and Stanford Background datasets. Note the behavioral similarity to the sketch presented in Fig. 1 (a). As can be seen for PASCAL VOC, starting from the baseline DeepLabv3+ with mIoU= 82.2% (denoted by g_0), we initially diverge towards $g_{0,0}$ and $g_{0,1}$ in the first two cycles, each time going back to g_0 and applying map aggregation as discussed in Section 3. This clearly helped to stabilize the model in the third cycle to ascend from g_0 towards g_1, which is the new peak found. Starting from g_1 again the network tends to descend towards divergence (ending with $g_{1,0}$) after which applying the map aggregation helped to ascend and find the next peak g_2. This process continues following the mechanics of LoAd as described in Section 3 alleviating the unstable behavior of the adversarial network helping it to sequentially improve in all three subfigures.

5.1. Results on PASCAL VOC 2012

Table 1 summarizes the performance comparison between the baseline (DeepLabv3+, also denoted as DLv3+ for brevity), the adversarial approach of Luc et al. (2016), and the proposed boosted model after applying LoAd (DeepLabv3+ & LoAd) on the validation set. For the last two models, we also present the standard deviation (std.) of the runs. The results are interesting in that even though the overall mIoU has increased by about 0.9% (0.7% above (Luc et al., 2016)), we observe considerable improvement in several classes. Note that an increase of 0.9% *averaged over* 21 *classes* is not insignificant by itself. In some of the highlighted classes such as "aeroplane", "sofa", "diningtable", and "boat" the improvement in IoU ranges from +2% to +6% which is quite significant in this regime of performance. Obviously, we degrade in some other classes but only by a fraction of a percent.

To substantiate our understanding, we evaluated the same three models as in Table 1 on the *test* set of PASCAL VOC on its online server. Note that the online servers allow only a few trials, and thus, we do not have enough runs to report standard deviations here. The outcome in Table 2 once again corroborates our claim. The overall mIoU improves by 0.7% above both DeepLabv3+ and Luc et al. (2016) with even a larger number of classes outperforming the baseline DeepLabv3+, when compared to the validation set. However, less extreme performance jumps can be seen with a maximum of +3.6% boost for "sofa", and +1% to +2% improvement for "boat", "dining table", and so on. In some classes, we also degrade in performance, but this remains to be a fraction of a percent akin to the case of the validation set in Table 1. Another interesting observation is that on the test set the adversarial approach of Luc et al. (2016) outperforms LoAd in a couple of target classes, most notably on "car". Check out the results for the baseline DeepLabv3+,[3] the adversarial approach of Luc et al. (2016),[4] and the proposed method (DeepLabv3+ & LoAd)[5] on PASCAL VOC's online server.

---

[3] DLv3+: http://host.robots.ox.ac.uk:8080/anonymous/SPNVZZ.html.

[4] Luc et al. (2016): http://host.robots.ox.ac.uk:8080/anonymous/C2GEVB.html.

[5] DLv3+&LoAd: http://host.robots.ox.ac.uk:8080/anonymous/CNGCEP.html.

(a) PASCAL VOC 2012   (b) Cityscapes   (c) Stanford Background
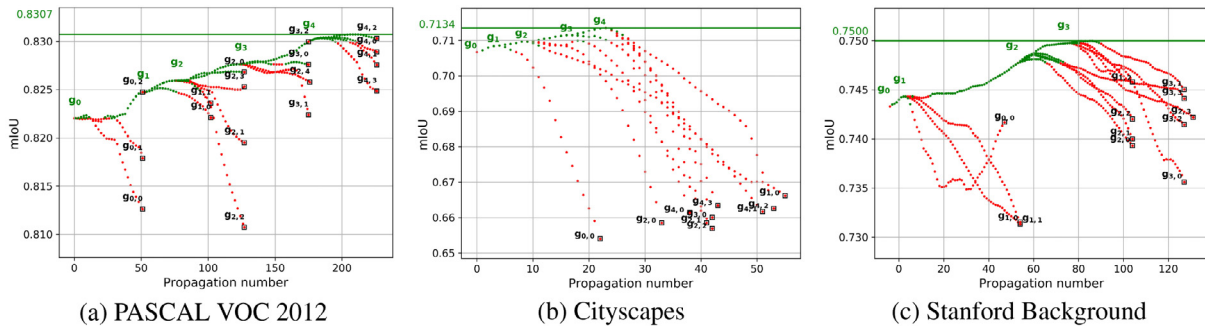
Fig. 3. Convergence graph of different datasets.

Table 1
Performance comparison (mIoU $\pm$ std.) on PASCAL VOC 2012 *validation* set.

Method	Backg.	Aero.	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
DLv3+	95.55	90.33	44.23	89.56	72.15	81.11	**96.76**	**91.37**	94.33	51.87	**96.08**
Luc et al. (2016)	95.57$_{\pm0.06}$	91.94$_{\pm0.19}$	**44.27**$_{\pm0.49}$	89.99$_{\pm0.21}$	72.83$_{\pm0.10}$	**81.48**$_{\pm0.27}$	96.70$_{\pm0.07}$	91.21$_{\pm0.17}$	94.27$_{\pm0.11}$	51.92$_{\pm0.38}$	95.75$_{\pm0.10}$
DLv3+ &LoAd	**95.71**$_{\pm0.06}$	**93.31**$_{\pm0.17}$	44.14$_{\pm0.49}$	**90.49**$_{\pm0.20}$	**74.48**$_{\pm0.10}$	81.37$_{\pm0.28}$	96.36$_{\pm0.08}$	91.34$_{\pm0.17}$	93.73$_{\pm0.11}$	**53.22**$_{\pm0.38}$	95.59$_{\pm0.10}$

Contd.	d.table	Dog	Horse	m.bike	Person	p.plant	Sheep	Sofa	Train	Tv	mIoU
DLv3+	60.14	92.63	93.33	89.23	90.18	67.19	93.75	61.26	94.81	80.27	82.20
Luc et al. (2016)	60.32$_{\pm0.25}$	**92.65**$_{\pm0.13}$	93.31$_{\pm0.14}$	**89.36**$_{\pm0.17}$	90.19$_{\pm0.17}$	67.35$_{\pm0.39}$	**93.90**$_{\pm0.13}$	61.30$_{\pm0.24}$	94.79$_{\pm0.10}$	80.36$_{\pm0.26}$	82.35$_{\pm0.09}$
DLv3+ &LoAd	**65.86**$_{\pm0.24}$	92.35$_{\pm0.13}$	93.34$_{\pm0.14}$	89.22$_{\pm0.17}$	90.20$_{\pm0.17}$	**67.42**$_{\pm0.39}$	93.41$_{\pm0.13}$	**67.20**$_{\pm0.23}$	**94.91**$_{\pm0.10}$	**80.81**$_{\pm0.25}$	**83.08**$_{\pm0.09}$

Table 2
Performance comparison on PASCAL VOC 2012 *test* set (online server).

Method	Backg.	Aero.	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
DLv3+	93.52	84.27	39.70	86.22	66.67	79.68	**92.12**	81.89	85.84	42.40	**82.91**
Luc et al. (2016)	**93.69**	**85.10**	**39.81**	86.39	**68.04**	**80.42**	92.08	**84.21**	85.34	**43.33**	80.41
DLv3+ &LoAd	93.68	84.91	39.76	**86.48**	68.02	80.32	91.98	82.70	**88.18**	42.64	82.74

Contd.	d.table	Dog	Horse	m.bike	Person	p.plant	Sheep	Sofa	Train	Tv	mIoU
DLv3+	75.20	84.10	**83.64**	**86.17**	82.68	62.54	**81.69**	63.39	81.60	**76.70**	76.81
Luc et al. (2016)	76.96	81.09	81.78	85.72	82.37	61.62	79.07	**67.24**	**83.37**	74.95	76.81
DLv3+ &LoAd	**77.65**	**86.02**	83.16	85.80	**82.87**	**62.56**	81.33	67.03	83.18	76.67	**77.51**

Table 3
Performance comparison (mIoU $\pm$ std.) on Cityscapes *validation* set.

Method	Road	Sidewalk	Building	Wall	Fence	Pole	t.light	t.sign	vegetation	Terrain
DLv3+	97.56	81.08	90.20	38.34	53.23	50.27	60.96	70.71	90.91	59.00
Luc et al. (2016)	97.54$_{\pm0.02}$	81.02$_{\pm0.12}$	90.23$_{\pm0.05}$	39.26$_{\pm0.27}$	52.77$_{\pm0.25}$	50.82$_{\pm0.30}$	61.17$_{\pm0.62}$	70.94$_{\pm0.40}$	90.96$_{\pm0.05}$	59.25$_{\pm0.28}$
DLv3+ &LoAd	**97.59**$_{\pm0.02}$	**81.21**$_{\pm0.12}$	**90.53**$_{\pm0.05}$	**47.20**$_{\pm0.27}$	**54.38**$_{\pm0.25}$	**51.81**$_{\pm0.30}$	61.54$_{\pm0.62}$	**71.11**$_{\pm0.40}$	90.92$_{\pm0.05}$	**60.66**$_{\pm0.27}$

Contd.	Sky	Person	Rider	Car	Truck	Bus	Train	m.cycle	Bicycle	mIoU
DLv3+	92.95	**76.21**	52.79	93.36	68.83	76.39	63.43	53.70	72.73	70.67
Luc et al. (2016)	**92.99**$_{\pm0.10}$	76.13$_{\pm0.25}$	52.65$_{\pm0.57}$	**93.38**$_{\pm0.07}$	**69.85**$_{\pm0.25}$	**77.30**$_{\pm0.19}$	62.72$_{\pm0.22}$	54.45$_{\pm0.56}$	72.76$_{\pm0.32}$	70.85$_{\pm0.03}$
DLv3+ &LoAd	92.94$_{\pm0.10}$	76.13$_{\pm0.25}$	**53.27**$_{\pm0.57}$	93.19$_{\pm0.07}$	67.76$_{\pm0.25}$	76.96$_{\pm0.19}$	**65.00**$_{\pm0.22}$	**54.63**$_{\pm0.56}$	**72.85**$_{\pm0.32}$	**71.57**$_{\pm0.03}$

Table 4
Performance comparison on Cityscapes *test* set (online server).

Method	Road	Sidewalk	Building	Wall	Fence	Pole	t.light	t.sign	vegetation	Terrain
DLv3+	97.86	80.91	90.14	43.73	48.73	49.14	62.11	67.82	91.69	66.42
Luc et al. (2016)	97.84	80.79	90.19	44.62	48.22	49.64	62.31	**67.84**	91.77	66.86
DLv3+ &LoAd	97.85	**80.99**	**90.25**	**50.17**	49.46	**50.24**	62.73	67.32	**91.80**	**68.32**

contd.	Sky	Person	Rider	Car	Truck	Bus	Train	m.cycle	Bicycle	mIoU
DLv3+	**93.85**	**79.43**	59.21	93.86	53.41	64.54	58.96	56.79	**68.55**	69.85
Luc et al. (2016)	93.84	79.36	59.13	**93.87**	54.28	66.72	59.22	**57.22**	68.50	70.12
DLv3+ &LoAd	93.70	79.27	**59.23**	93.75	**54.38**	**67.61**	**60.24**	56.32	68.36	**70.63**

Table 5
Performance comparison (mIoU $\pm$ std.) on Stanford Background *test* set.

Method	Sky	Tree	Road	grass	Water	Building	Mountain	Foreground	mIoU
DeepLabv3+	89.38	72.21	87.28	77.44	72.70	80.03	48.64	66.92	74.33
Luc et al. (2016)	89.35$_{\pm0.27}$	72.54$_{\pm0.41}$	87.31$_{\pm0.23}$	77.53$_{\pm0.34}$	72.78$_{\pm0.30}$	80.04$_{\pm0.25}$	49.18$_{\pm0.48}$	66.69$_{\pm0.46}$	74.43$_{\pm0.16}$
DeepLabv3+ &LoAd	**89.47**$_{\pm0.27}$	**72.89**$_{\pm0.40}$	**87.71**$_{\pm0.23}$	**77.90**$_{\pm0.33}$	**73.89**$_{\pm0.30}$	**80.87**$_{\pm0.24}$	**50.11**$_{\pm0.48}$	**67.54**$_{\pm0.45}$	**75.05**$_{\pm0.16}$

5.2. Results on cityscapes

We conducted similar extensive experimentation on Cityscapes as well. Table 3 summarizes the performance comparison between the baseline DeepLabv3+ (abbreviated as DLv3+), the adversarial approach of Luc et al. (2016), and the proposed boosted model DeepLabv3+ &LoAd. Here again we achieve an overall mIoU improvement of 0.9% (0.7% above (Luc et al., 2016)) averaged over 19 classes with considerable performance boost (up to +7%) in "wall", "fence", "pole", "train" and "terrain". To further consolidate our understanding, we evaluated the same three models as in Table 3 on the *test* set of Cityscapes on its online server and the results are summarized in Table 4. We achieve an overall performance boost of about 0.8% above the baseline DeepLabv3+ (0.5% above (Luc et al., 2016)) and a considerable improvement (up to +5%) for "wall", "terrain", "train", and so on, that once again confirm the impact of the proposed method on this commonly used dataset. Here, we consistently outperform the adversarial approach of Luc et al. (2016) on the aforementioned classes. Notably, for Cityscapes, we almost consistently outperform DeepLabv3+ in all classes across both Tables 3 and 4. Check out the results for the baseline DeepLabv3+,[6] the adversarial approach of Luc et al. (2016),[7] and the proposed method[8] on Cityscapes' online server.

5.3. Results on stanford background

Following (Luc et al., 2016), we conducted yet another set of experimentation on the Stanford Background dataset. The results are summarized in Table 5. As can be seen, applying LoAd on DeepLabv3+ boosts the overall mIoU by 0.7% (0.6% beyond (Luc et al., 2016)) averaged over 8 classes, which is slightly less than PASCAL VOC 2012 and Cityscapes. However, here the boosted model consistently outperforms the baseline DeepLabv3+ and the adversarial approach of Luc et al. (2016) on all classes without any exception with considerable improvement on "mountain" and "water".

Before we proceed with presenting some qualitative results, let us highlight the summary of our understanding on where LoAd is making the most impact. Based on our quantitative analyses so far (corroborated later on by the qualitative results), it appears that a combination of two parameters play a role in which classes are most impacted by LoAd. First, classes that tend to be confused (such as "sofa" vs. "chair", "boat" vs. "aeroplane" in PASCAL VOC, and "wall" vs. "fence" and "pole" vs. "t.light" in Cityscapes) are significantly boosted as a result of applying LoAd. Second, classes that have fluid and non-compact shapes and can be elongated across the whole image (for instance, "wall" and "terrain" in Cityscapes, and "water" and "mountain" in Stanford Background) are mostly boosted by LoAd compared to other baselines. One can also analyze this with confusion matrices. Interested reader is referred to Appendix C for more details.

5.4. Qualitative results

A selected set of qualitative results are illustrated in the next four figures. We compare the performance of LoAd with DeepLabv3+ (non-adversarial baseline) and the work of Luc et al. (2016) (adversarial baseline with DeepLabv3+ as segmentor) in predicting the ground truth label map. In Fig. 4, on the top row, the whole dining table is missed by both baselines and LoAd manages to boost DeepLabv3+ to almost fully recover that. On the second row, the confusion between "sofa" and "chair" is resolved. Notably, Luc et al. (2016) (the fourth column) is performing better than DeepLabv3+, but stays behind the proposed approach. On the third row, once again the impact is significant, a misconception of a "boat" in the middle of the water is resolved. Th

next two rows show interesting signs of avoiding class swap/confusion. Being a rather difficult case to distinguish even for human observer, LoAd helps DeeLabv3+ to resolve mistaking the second "person" on the boat with boat engine on the fourth row (a confusion which is not addressed by Luc et al. (2016)). The same can be argued about the next row where "sofa" (green) vs. "chair" (in red) confusions are resolved. The last three rows from Cityscapes demonstrate a much more consistent segmentation of "train", "truck" and "wall" compared to the other baselines. Another set of qualitative results from both PASCAL VOC and Cityscapes datasets is illustrated in Fig. 5. On the top two rows, "sofa" is segmented with better consistency and continuity compared to the other two baselines. The third row illustrates the continuity in segmenting "bird" foot. The fourth row, shows how confusion about the existence of a "chair" is resolved. The fifth row shows how LoAd outperforms in detecting the "bird" in the cluttered scene, and manages to (a good extent) resolve the confusion of tree trunk with another bird in the scene. The next three rows from Cityscapes illustrate a more consistent and continuous label map generation for "fence", "motorcycle", and "pole".

To further consolidate the qualitative improvement of DeepLabv3+ &LoAd compared to DeepLabv3+, we provide eight examples in Figs. 6 and 7 where the gradual impact of LoAd with its embedded label map aggregation on the performance of the baseline throughout its cycles (ending up with divergence or leading to finding new peaks) is at the center of attention. As can be seen, starting from the baseline DeepLabv3+, LoAd gradually learns from the intermediate degradation in its downward divergent stages at (c), (e) and (g) and resolves confusion. The first example (top two rows) in Fig. 6 is a prime example of how a complete instance of "sofa" is missed by the baseline DeepLabv3+ (b), whereas LoAd manages to completely recover it (h). The second example (next two rows) reiterate the same message, offering much better consistency compared to the baseline. The third example (third two rows) poses a challenging scenario where a "boat" and an "aeroplane" are confused due to perspective and texture complexity. LoAd (h) shows a considerable improvement in resolving this confusion. Last example in this figure depicts how the missing bottom part of the bottle is recovered owing to label map aggregation. The same story continues in Fig. 7. Top two examples highlight how LoAd helps the baseline to infill the gaps and missing pieces in segmenting a "dinning table". Last two examples (last four rows) in the figure demonstrate how class confusion ("sofa" vs "chair") is totally resolved when compared to the baseline.

6. Ablation study

The first angle to investigate is the impact of changing the core components of the adversarial network. Our initial experimentation with VGG16 as well as custom designed CNNs as discriminator did not lead to any notable performance boost or degradation. So, here we focus on the impact of modifying the generator (segmentor), currently being DeepLabv3+. Table 6 summarizes the results of our experimentation with a different backbone (MobileNetv2) for the segmentor in comparison with the main backbone used in our previous experimentation (Xception-65). The results reported in this table correspond to the best performing models (baseline and proposed) on PASCAL VOC 2012, for the sake of a fair comparison. Obviously, a weaker backbone (MobileNetv2) results in performance drop for both baseline and boosted model with LoAd on *validation* and *test* datasets. Nevertheless, what remains consistent is the improvement of DeepLabv3+ &LoAd over the baseline, even though the improvement is a bit less pronounced for both *validation* and *test* datasets.

Another angle to investigate is the impact of hyperparamter change on the performance of DeepLabv3+ &LoAd. We perturb all the important design hyperparameters of LoAd described in Section 3 and in Algorithm 2. The results summarized in Table 7 are extracted with our main choice of backbone (Xception-65). As can be seen, increasing the

---

[6] DLv3+: https://tinyurl.com/ctyscps-dlv3plus.
[7] Luc et al. (2016): https://tinyurl.com/ctyscps-dlv3plus-luc-2.
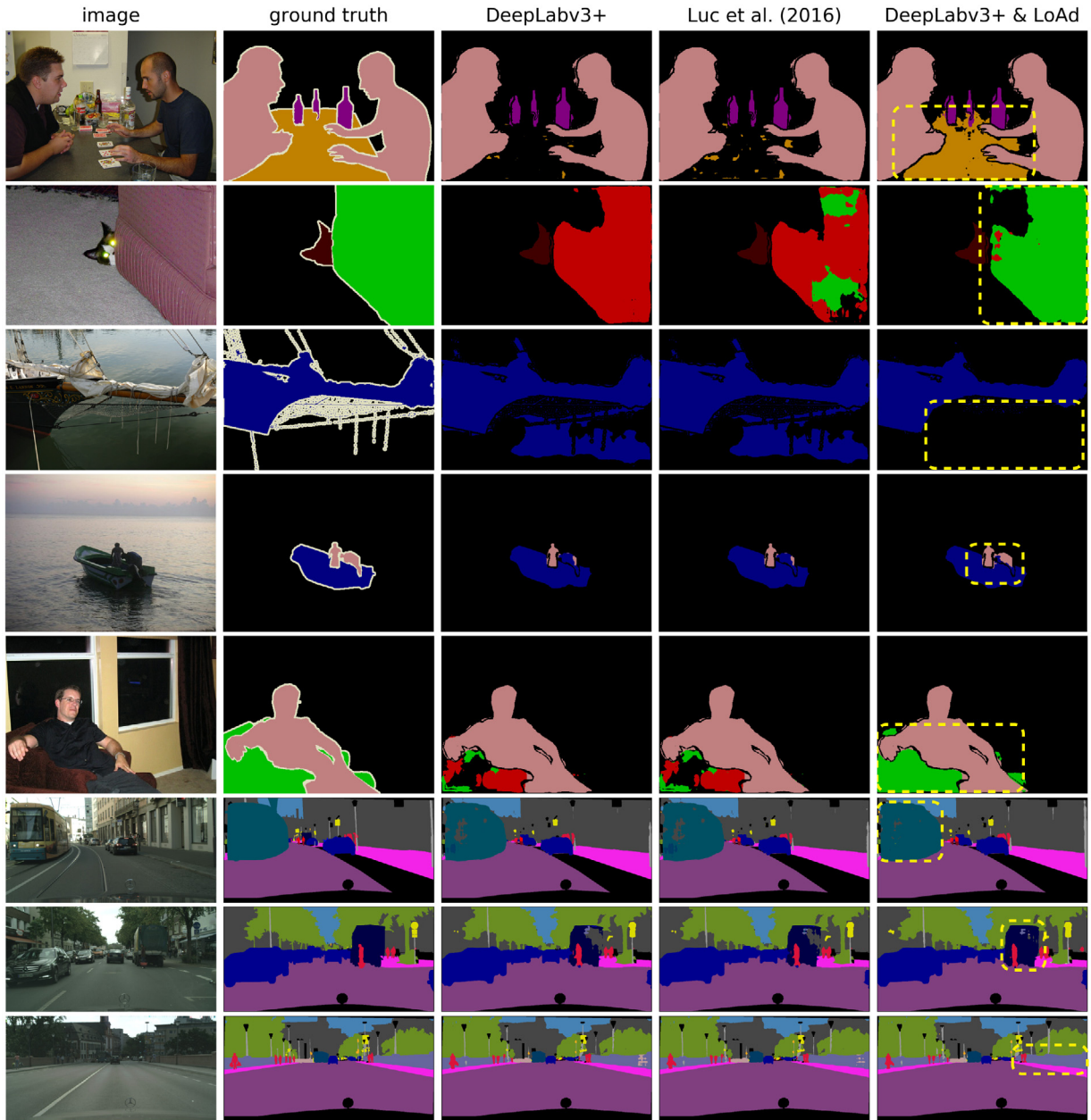[8] DLv3+&LoAd: https://tinyurl.com/ctyscps-dlv3plus-load-2.

Fig. 4. Selected qualitative results on PASCAL VOC 2012 and Cityscapes. These examples show how totally missing items and class swap/confusion are resolved or considerably improved. Ignored pixels and classes (per standard) unused for evaluation in black. Best view in color with 300% zoom. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

maximum set size of the buffer results in performance degradation of DeepLabv3+ & LoAd in comparison with the best setting with $B_{max} = 3$. This could potentially be due to the fact that we only generate and aggregate "fake" label map datasets, keeping only one ground truth label map set, which can be a source of imbalance if too many of these sets are aggregated. This is the exact reason why we have introduced this cap, B_{max}. A possible solution to remedy this issue, besides weighting the discriminator loss as discussed in Section 4, could be to generate fake label maps only for a subset of training image set, e.g., images that have been seen by the corresponding model (peak or ending) up to that stage of training.

Next experiments are concerned with perturbing β_l and β_u. As can be seen in Table 7, lowering β_l to 10% (in mIoU) does not seem to have a major impact on the overall mIoU (= 82.80%), whereas increasing β_u to 0.5% downgrades the performance (mIoU = 82.42%), suggesting a conservative upper-bound for finding a peak per cycle. To further

Table 6

Impact of generator backbone change for *val* and *test* results of PASCAL VOC 2012.

Method	Gen. backbone	mIoU (*val*)	mIoU (*test*)
DLv3+	Xception-65	82.20	76.81
DLv3+&LoAd	Xception-65	**83.08**	**77.51**
DLv3+	MobileNetv2	75.32	71.45
DLv3+&LoAd	MobileNetv2	**75.70**	**71.74**

investigate this, we conducted two more experiments. The results indicate that decreasing β_l further down to 20% seems to gradually degrade the performance (mIoU = 82.43%), even though increasing the divergence patience parameter Γ from 50 to 80 iterations helps to avoid this to some extent (mIoU = 82.65%). Overall, it seems aggregating heavily degraded label maps (corresponding to lower β_l's) does not help
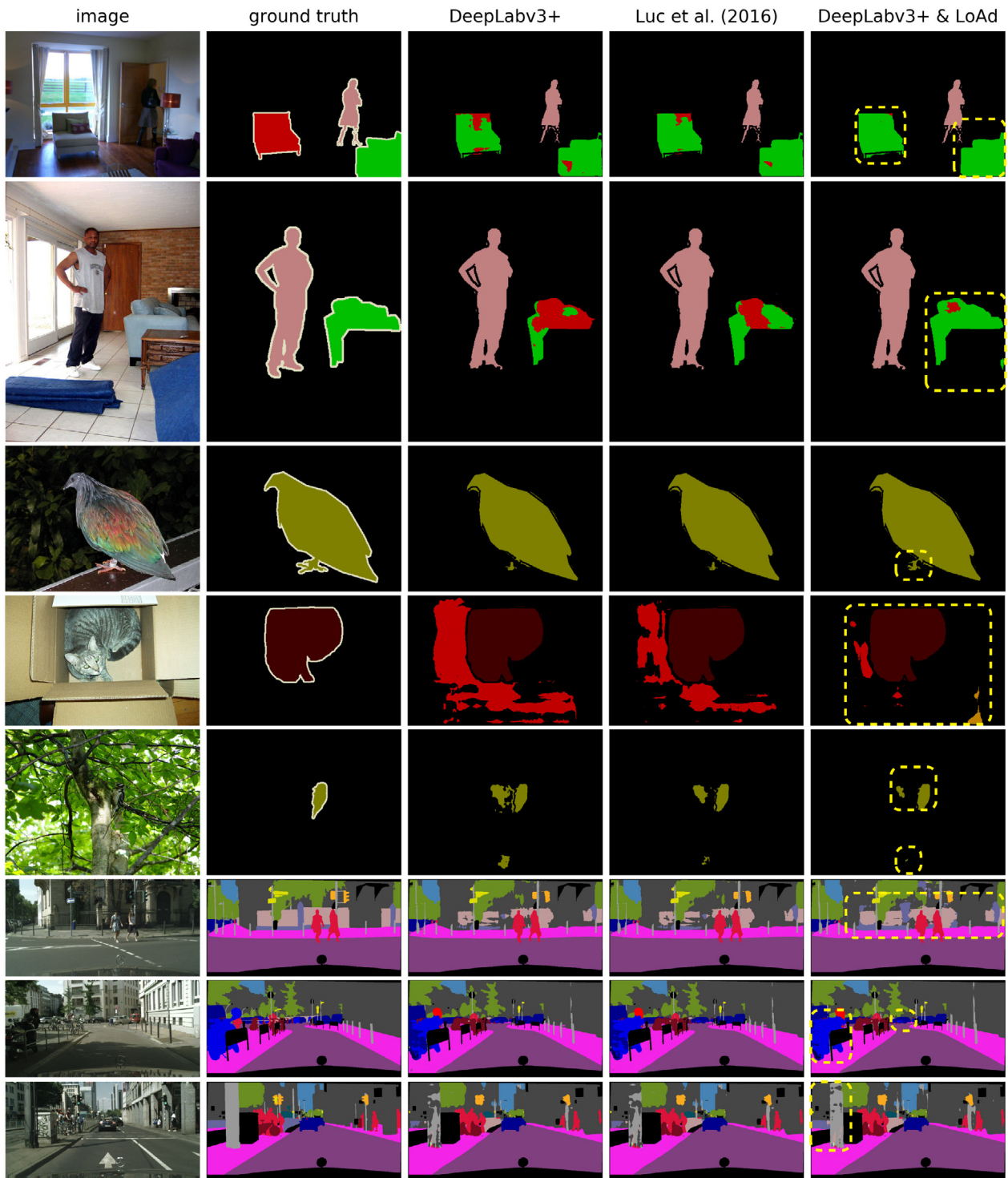
Fig. 5. Selected qualitative results on PASCAL VOC 2012 and Cityscapes. These examples demonstrate how inconsistent body/edges and class swap/confusion are either totally resolved or considerably improved. Ignored pixels and classes (per standard) unused for evaluation in black. Best view in color with 300% zoom. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the performance as this is implicitly related to how we constrain the solution space of the optimization problem per cycle of LoAd.

Next angle to investigate is the impact of the regularization parameter λ in the hybrid loss in (3). This parameter is set to 1 in the previous experiments giving an equal weight to the pixel-wise and adversarial terms. Setting $\lambda = 0$ would lead to only pixel-wise CE loss, and increasing it beyond 1 would gradually shadow the pixel-wise CE loss. Table 8 summarizes the impact of varying λ within $[0, 10]$ on LoAd for PASCAL VOC 2012 and Cityscapes datasets. As can be seen, $\lambda = 1$

seems to be the sweetspot on both datasets, even though the results are in a comparable range for $\lambda = 0.5, 1$. Decreasing λ towards 0 (leading to only pixel-wise) and increasing to 10 (resulting in adversarial loss being dominant) appears to have destructive effect.

Another aspect we would like to reflect on is LoAd's potential in stabilizing the adversarial network. Our metric to quantify this is the average number of times divergence happens throughout the adversarial training process as a result of which LoAd steps back to where it started and avoids that by retraining the discriminator with new label
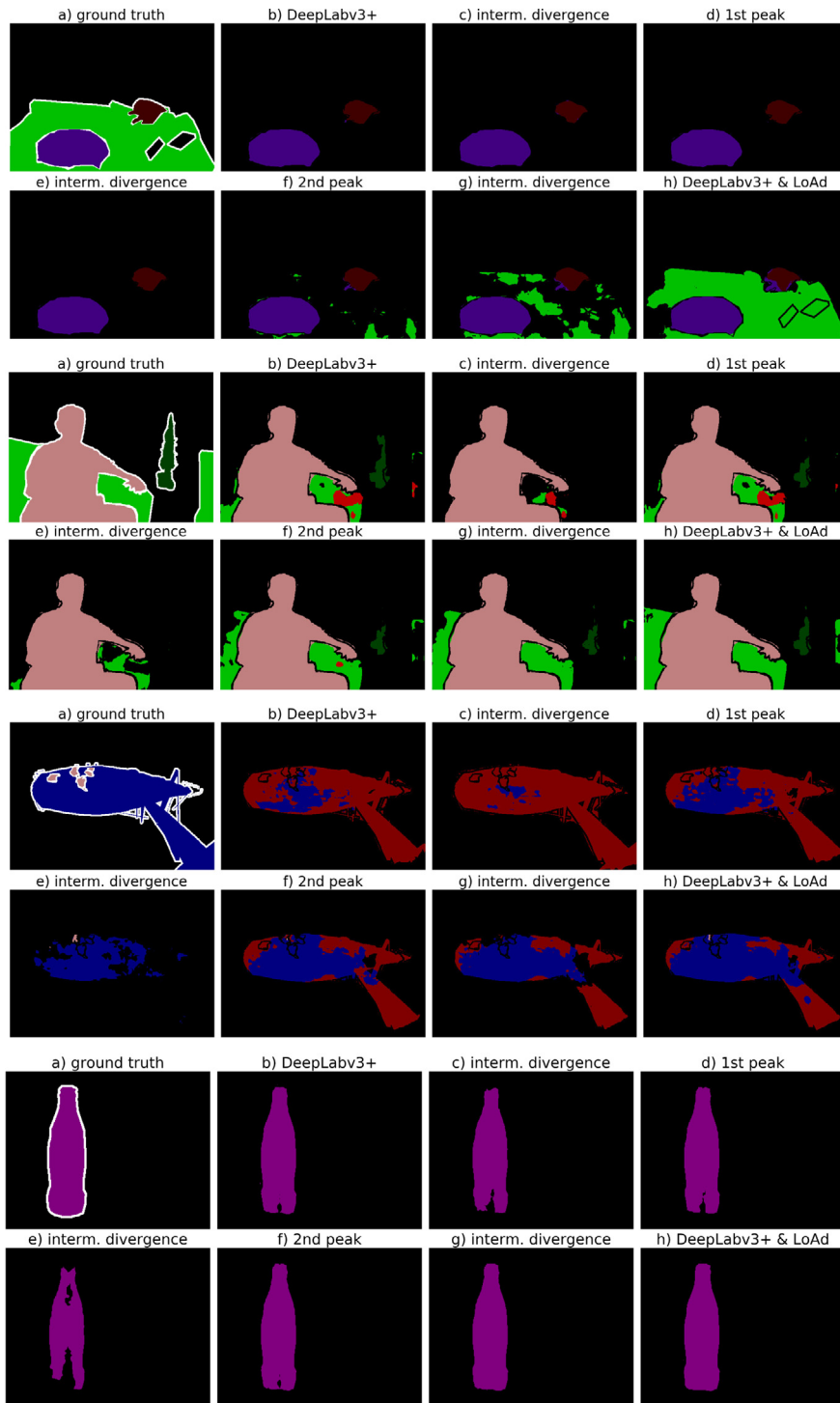
Fig. 6. This illustrates how LoAd benefits from label map aggregation throughout its convergence process to boost the performance of the baseline DeepLabv3+. Every consecutive two rows correspond to a single input image. Best view in color with 300% zoom. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

maps, as explained in Section 3. This is summarized for a total of 80 experiments on PASCAL VOC 2012 (54 runs) and Cityscapes (26 runs) in Table 9 where the average (avg.) and standard deviation (std.) of the number of treated divergences are reported. On average, regardless of dataset, LoAd handles 29 ± 9 divergences to reach the final performance. Another point that is reported in this table is the percentage of successful experiments. Here, a failed experiment is one that does not reach at least 0.1% improvement beyond the starting checkpoint. As

can be seen, 91%+ of the experiments have been successful based on this definition.

Finally, we would like to shed some light on what would happen in terms of performance if we had also employed expensive time-consuming post-processing steps at inference time. Table 10 summarizes the impact of applying multi-scale and flip (denoted by MS&F) *only for evaluation* (the last column) against already reported results without employing MS&F (the third column). We observe about 1%
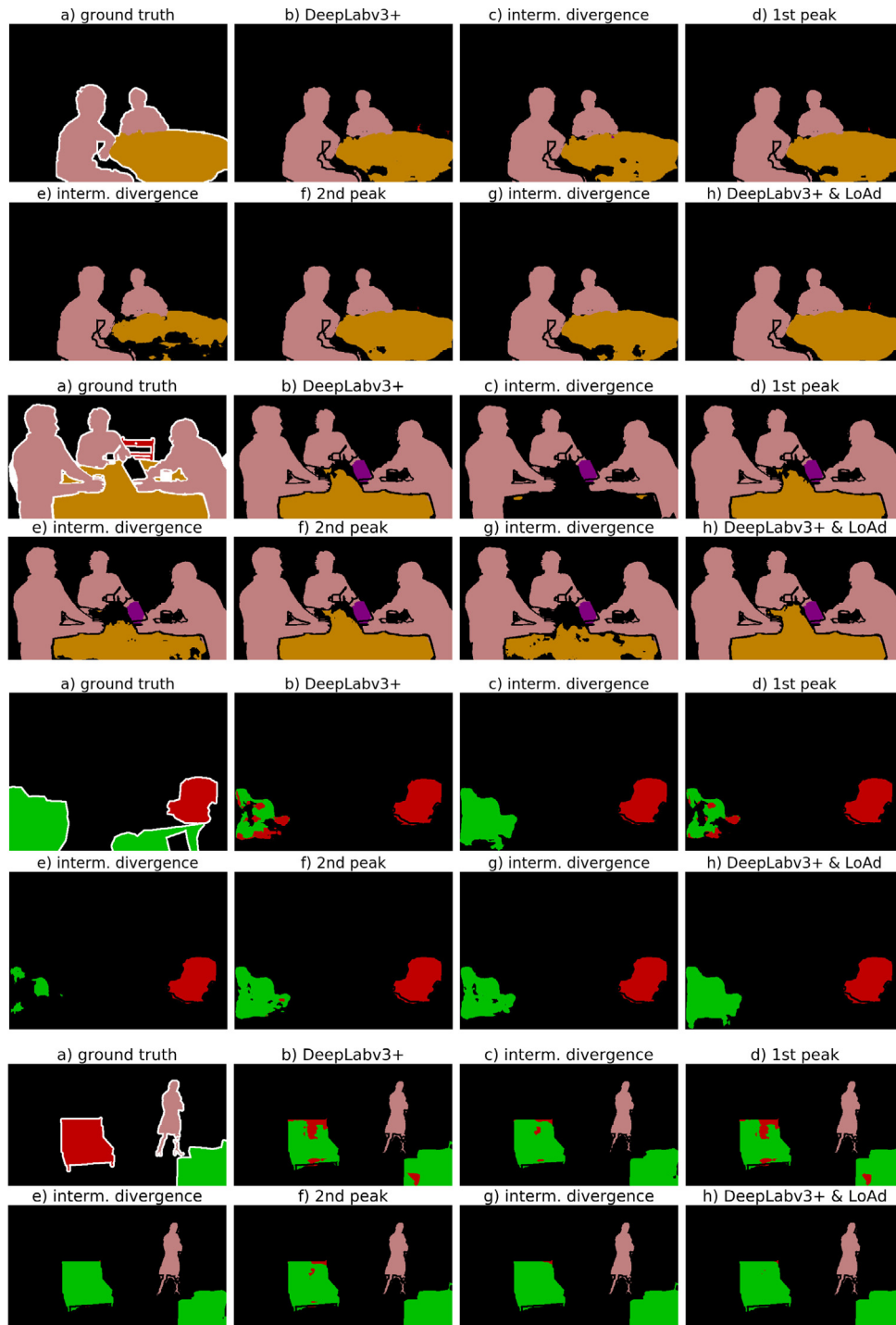
Fig. 7. This illustrates how LoAd benefits from label map aggregation throughout its convergence process to boost the performance of the baseline DeepLabv3+. Every consecutive two rows correspond to a single input image. Best view in color with 300% zoom. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to 3% performance boost in almost all models pushing their performance towards state-of-the-art results in the literature. However, this performance boost comes at the cost of scales of magnitude (roughly 100 times) larger inference time rendering the models out of real-time inference response. In summary, pushing the performance at any cost is not our focus; keeping the inference time unchanged (in practice less than 1 sec) and improving the performance is what we offer in this work.

7. Concluding remarks

Summary and impact. We proposed a novel lookahead adversarial learning (LoAd) approach with an embedded label map aggregation idea for adversarial semantic segmentation with state-of-the-art models. For experimental evaluation, we picked DeepLabv3+ (also abbreviated as DLv3+) as segmentor/generator in the adversarial setting. We further elaborated in Sections 1 and 4 that among a few available choices for DeepLabv3+ architectures and corresponding training/inference

Table 7

Influence of hyperparameter change on LoAd (Xception-65) for *val* results of PASCAL VOC 2012.

Method	β_l (%)	β_u (%)	Γ	B_{max}	mIoU (*val*)
DLv3+	–	–	–	–	82.20
DLv3+&LoAd	5	0.1	50	3	**82.88**
DLv3+&LoAd	5	0.1	50	4	82.46
DLv3+&LoAd	5	0.5	50	3	82.42
DLv3+&LoAd	10	0.1	50	3	82.80
DLv3+&LoAd	20	0.1	50	3	82.43
DLv3+&LoAd	20	0.1	80	3	82.65

Table 8

Influence of regularization parameter λ. Xception-65 is used as the generator backbone for PASCAL VOC 2012 and MobileNetv2 is used for Cityscapes.

Dataset	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 10$
VOC 2012	82.20	83.16	82.95	**83.26**	82.55
Cityscapes	70.67	71.23	71.11	**71.91**	71.14

Table 9

Summary of stability metrics.

Dataset	Backbone	Avg. div.	Std.	Succ. exps.
VOC 2012	Xception-65	30	8	95.0%
	MobileNetv2	25	2	
Cityscapes	Xception-65	23	7	84.6%
	MobileNetv2	32	11	
Overall	Any	29	9	91.9%

Table 10

Impact of multi-scale and flip (MS&F) inference strategy on validation data (unless otherwise stated) for each dataset. Xception-65 is used as the generator backbone for PASCAL VOC 2012 and Stanford Background, and MobileNetv2 is used for Cityscapes.

Method	Dataset (*val*)	mIoU	mIoU (MS&F)
DLv3+	VOC 2012	82.20	83.58
Luc et al. (2016)	VOC 2012	82.35	83.66
DLv3+&LoAd	VOC 2012	**83.08**	**84.57**
DLv3+	Cityscapes	70.67	73.88
Luc et al. (2016)	Cityscapes	70.85	73.96
DLv3+&LoAd	Cityscapes	**71.57**	**74.61**
DLv3+	Stanford (*test*)	74.33	76.33
Luc et al. (2016)	Stanford (*test*)	74.43	76.42
DLabv3+&LoAd	Stanford (*test*)	**75.05**	**76.78**

strategies, we deliberately picked the ones which can be trained on *commodity GPU nodes* (because not everyone has access to high-end TPUs) and *run fast at inference*; roughly speaking, less than a second on average per sample input image. Therefore, we did not opt for models requiring multi-scaling strategies or small output strides leading to several minutes or in some cases few tens of minutes inference time per image. Running fast at inference time for field applications is a *core motive* of this work. We demonstrated that the proposed idea can improve the performance of the baseline DeepLabv3+ by about 0.9% in mIoU sense on two of the most modern and commonly-used datasets (PASCAL VOC 12, Cityscapes) while boosting the performance in certain classes up to 7% without introducing any extra delay at inference. In other words, the proposed models run practically as fast as the baseline on top of which they are applied. The qualitative results show that our approach is helping the baseline segmentation model to resolve class confusion to a good extent as well as to produce label maps which are more consistent in terms of continuity and structure. We think the application domain of LoAd could possibly be larger than semantic segmentation. This is an interesting avenue to explore.

Complexity. LoAd adds architectural complexity by adding a (simple) discriminator network but requires no changes to the base segmentation model, as such in theory, it can be applied on top of any semantic segmentation model (not only DeepLabv3+). On the flip side,

we introduce complexity at training both in time and space slowing it down in favor of performance. Roughly speaking, in worst case scenario of large size images containing N pixels, for an evaluation (hold-out) subset of size V_e samples, we impose an extra $O(V_e N)$ time complexity per propagation for adversarial training. In practice, $V_e \ll N$. Also, with the label map aggregation module of LoAd we need to train the discriminator on $B_{max} M$ images stacked in the buffer (space complexity) instead of only M_b images in a batch with $M_b < M$. On the other hand, we train the discriminator once per LoAd cycle, i.e., orders of magnitude less often than typical GDA based adversarial training. In terms of time complexity, and for the sake of comparison, on a P100 T GPU for PASCAL VOC 2012, DeepLabv3+ takes 20:01:41 (hour/min/sec) and DLv3+&LoAd takes 53:48:30 (hour/min/sec) to reach best performance. LoAd's elapsed time is about 2.5 times larger, even though looking at typical training times in literature (sometimes taking weeks), only 1.5 days extra is not a significant overhead.

Future directions. A potential future direction is replacing DeepLabv3+ with another state-of-the-art segmentation network and analyzing the impact. We expect to obtain similar results as we have already shown by analyzing the impact of backbone change in Section 6. So far, we have also understood that slightly different map aggregation policies can be beneficial in different architectures. As an example, one could replace the Flush operation that cleans up all the previous aggregated maps with Retain for which always a certain number of previous peaks (from previous cycles) are retained. In this work, we do neither claim nor prove a generic applicability of LoAd beyond semantic segmentation. Therefore, a more general understanding of the impact of LoAd in adversarial settings requires it to be applied to other applications which is outside the scope of this work.

Declaration of competing interest

Acknowledgments

Appendices

In the following, after a brief note on the limitations of pixel-wise cross-entropy (CE) loss, we dive deep into the mechanics of LoAd providing an extended and more elaborate version of Algorithm 2 besides a process flowchart view of the proposed method. Finally some confusion matrices are presented to demonstrate how LoAd helps resolve class swap/confusion on both PASCAL VOC and Cityscapes.

Appendix A. Dimensionality of the problem

Let us briefly reflect on why employing only a pixel-wise cross-entropy (CE) optimization loss for semantic segmentation and relying on gradual increase in receptive field of CNNs might not be sufficient for explicitly promoting a large set of candidate solutions for label maps. A good portion of these solutions might not be valid, bearing in mind that missing out on the rest could also lead to sub-optimal solutions for the segmentation problem. This is in line with literature (Liu et al., 2017; Chen et al., 2017; Shen et al., 2017; Liu et al., 2017; Ke et al., 2018; Zhao et al., 2019a,b) proposing different approaches to address this challenge. Following our notation in Section 2, let $\mathcal{D}_t = \{(\mathbf{X}, \mathbf{Y})_1, \ldots, (\mathbf{X}, \mathbf{Y})_M\}$ be training dataset with M samples so that for all $(\mathbf{X}, \mathbf{Y})_i \in \mathcal{D}_t, (\mathbf{X}, \mathbf{Y})_i \sim P(\mathbf{X}, \mathbf{Y})$ where $\mathcal{X}_t = \{\mathbf{X} | (\mathbf{X}, \mathbf{Y}) \in \mathcal{D}_t\}$ and $\mathcal{Y}_t = \{\mathbf{Y} | (\mathbf{X}, \mathbf{Y}) \in \mathcal{D}_t\}$ respectively denote the set of images and their corresponding label maps in the training dataset. As discussed earlier, $\mathbf{X}$ is of size $H \times W \times 3$ for RGB images with a total of N pixels and the corresponding label map $\mathbf{Y}$ is of size $H \times W$ with elements in $\mathcal{K} = \{1, \ldots, K\}$ where K is the number of classes in segmentation task. Let $P(\mathbf{X})$, $P(\mathbf{Y})$ and $P(\mathbf{X}, \mathbf{Y})$ denote probability mass functions of corresponding categorical distributions. Now, with pixel independence assumption:

$$P(\mathbf{Y}) = \prod_{i=1}^{N} P(y_i), \tag{A.1}$$

where y_i denotes the ith pixel in $\mathbf{Y}$. This obviously ignores the potential correlation among the pixels. $P(\mathbf{Y})$ is a function from the set $E_{\mathbf{Y}}$, comprised of all possible values $\mathbf{Y}$ can take, to a probability in $\mathbb{R}$. As such, learning $P(\mathbf{Y})$ mandates exploring $E_{\mathbf{Y}} \times \mathbb{R}$. But how do all possible maps in $E_{\mathbf{Y}}$ look like? To answer this, let us consider a toy setup where $\mathbf{Y}$ is a 2×1 label map $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ with $K = 3$ classes in $\mathcal{K} = \{1, 2, 3\}$. The set of all possible label maps is simply:

$$E_{\mathbf{Y}} = \left\{ \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}3\\1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}2\\2\end{bmatrix}, \begin{bmatrix}3\\2\end{bmatrix}, \begin{bmatrix}1\\3\end{bmatrix}, \begin{bmatrix}2\\3\end{bmatrix}, \begin{bmatrix}3\\3\end{bmatrix} \right\},$$

whose size is equal to the permutations with replacement of 2 pixels from a set of 3 classes, i.e., $3^2 = 9$. One can straightforwardly generalize this toy example to any arbitrary number of pixels N and classes K and show that $|E_{\mathbf{Y}}| = K^N$ where $|\mathcal{A}|$ denotes the cardinality of set $\mathcal{A}$. Following the same approach, this time with pixel independence assumption in (A.1) for y_1 and y_2, the set of permutations would be 6, counting 3 possible values per y_i summing them up due to independence. Again, it is straightforward to show $|E_{\mathbf{Y}}| = KN$. This toy example illustrates how the solution space of the problem with pixel independence assumption is significantly smaller than the one considering all possible correlations between pixels. However, the latter can easily become intractable for large size images, and that is the reason why a pixel-wise loss has been so popular. We think a conditional adversarial approach has the capacity to partially explore the larger solution space and to capture these pixel correlations in a general (and not only local) fashion.

Appendix B. Deeper dive in LoAd

As discussed in Section 3, there are subtle details we skipped or described on a high level to convey the core idea of the proposed method (LoAd) and not to steal the readers attention away from the main message. In this section, we dive deeper into the mechanics of LoAd using Fig. B.8, a flow chart that breaks down its process flow in clear terms, together with a more detailed version of Algorithm 2. For the sake of clarity, we are using line numbers in Algorithm 3 to be able to refer to specific lines throughout the following explanations. Except for more details, the most important difference between Algorithm 3 and its simplified version, Algorithm 2, is that the search process to find a new peak per cycle is conducted in a dynamic fashion by introducing a new parameter, ω, that we call peak finder patience counter. The flowchart in Fig. B.8 does not go down into the smallest

details of every single line in Algorithm 3 yet it reflects on the most important steps to clarify the process. As can be seen in Fig. B.8, there are three components: (a) initialization (a one-off process), (b) label map aggregation (corresponding to Algorithm 1), and (c) the main body of LoAd for semantic segmentation (corresponding to Algorithm 3). The process starts from initialization depicted on the top left (part a), which corresponds to the preamble of Algorithm 3. Provided a starting model, the cycles of LoAd can start and iterate between its main body (part c) and its label map aggregation module (part b).

Algorithm 3: LoAd for Semantic Segmentation — Detailed Version

---

Initialize: $\psi = 0$, $g^s = g_0$, $\mathcal{B} = g^s(\mathcal{X}_t)$
Input: maximum cycle: Ψ, maximum divergence: Γ, maximum peak finder: Ω, β_l, β_u

1   $\mu^s, \mu, \mu^* \leftarrow$ Evaluate mIoU
2   Train Discriminator$(\mathcal{D}_t \cup \mathcal{B})$
3   **while** $\psi < \Psi$ **do**
4   keep the very first starting point: $\mu^0 \leftarrow \mu^s$
5   start a divergence patience counter: $\gamma \leftarrow 0$
6   start a peak finder patience counter: $\omega \leftarrow 0$
7   **while** $\mu^s - \beta_l < \mu$ and $\gamma < \Gamma$ **do**
8   update the model: $g \leftarrow$ Train Adversarial
9       $\mu \leftarrow$ Evaluate mIoU
10   **if** $\mu > \mu^s + \beta_u$ **then**
11   **if** new peak is found: $\mu > \mu^*$ **then**
12   update best performance: $\mu^* \leftarrow \mu$
13   update best model: $g^* \leftarrow g$
14   **end**
15   **if** $\omega > \Omega$ **then**
16   update starting performance: $\mu^s \leftarrow \mu$
17   update starting model: $g^s \leftarrow g$
18   reset peak finder counter: $\omega \leftarrow 0$
19   reset divergence counter: $\gamma \leftarrow 0$
20   **end**
21        $\omega \leftarrow \omega + 1$
22   **end**
23      $\gamma \leftarrow \gamma + 1$
24   **end**
25     $g^e \leftarrow$ keep last model of the cycle
26   **if** best peak is better than very first start: $\mu^* > \mu^0$ **then**
27   set best model as start model: $g^s \leftarrow g^*$
28   set best mIoU as start mIoU: $\mu^s \leftarrow \mu^*$
29       $\mathcal{B} \leftarrow$ Lookahead Map Aggregation$(g^*, g^e, \mathcal{X})$
30   reset cycle counter $\psi \leftarrow 0$
31   **else**
32       $\mathcal{B} \leftarrow$ Lookahead Map Aggregation$(0, g^e, \mathcal{X})$
33   start a new cycle $\psi \leftarrow \psi + 1$
34   **end**
35   Train Discriminator$(\mathcal{D}_t \cup \mathcal{B})$
36   **end**

Let us delve deeper into this process by zooming into Algorithm 3. Lines 3 to 35 outline what happens in a single cycle of LoAd. A cycle starts by initializing the divergence patience counter $\gamma \in [0, \ldots, \Gamma]$ and the peak finder patience counter $\omega \in [0, \ldots, \Omega]$. At the beginning of every cycle, we mark the very initial model performance (in mIoU) as μ^0 which will only be used in line 26 to decide if throughout the cycle we actually found a new model (a new peak in the convergence graph) that is better than the very initial model of the cycle in mIoU sense. Lines 7 to 24 embody the main while loop of the cycle that keeps training the adversarial network until one of the following conditions is met: (1) the current mIoU (μ) touches the bottom line we defined for downward/divergence trend ($\mu^s - \beta_l < \mu$) or γ meets its limit in the number of iterations. Inside this while loop, we watch for finding a new peak (model) in lines 10 to 22, and such a peak is only valid if the current mIoU goes beyond the starting mIoU plus a minimum increment
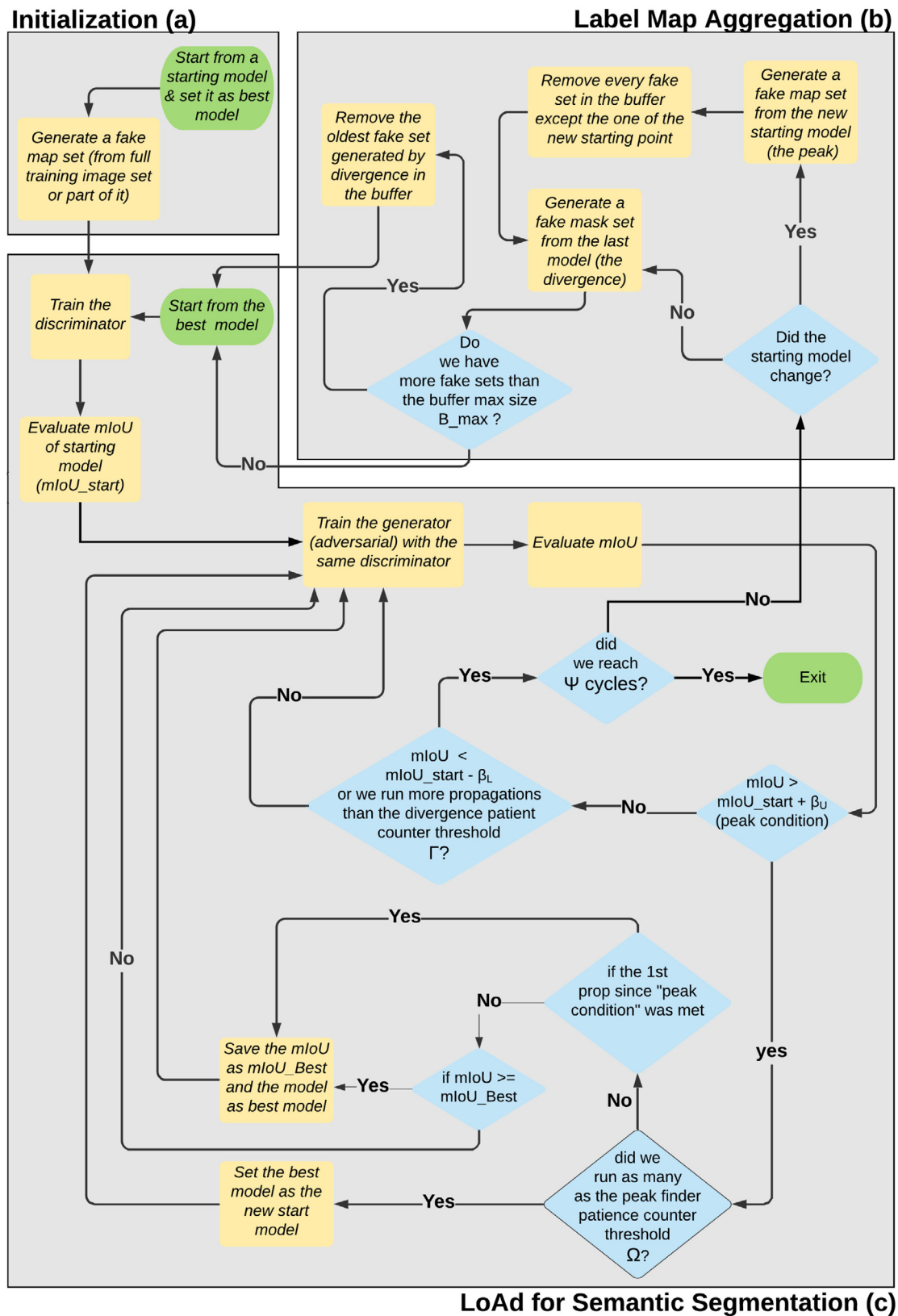
Fig. B.8. Process flow of LoAd with label map aggregation for semantic segmentation.

$(\mu > \mu^s + \beta_u)$. We came to understand by extensive experimentation that if the starting model (g^s with mIoU μ^s) is kept unchanged in this while loop (e.g., kept as μ^0 as in line 4), and thus, we select a peak based on a static peak finder patience counter ω, we can prematurely kill an

upward trend leading to a higher peak. This is the rationale behind the dynamic peak finding process in which we dynamically update the starting model (g^s, μ^s) in lines 16 and 17. To do so, every time we reach the end of a peak finding process (ω reaches Ω), if the mIoU is

(a) DLv3+: recall

(b) DLv3+ & LoAd: recall
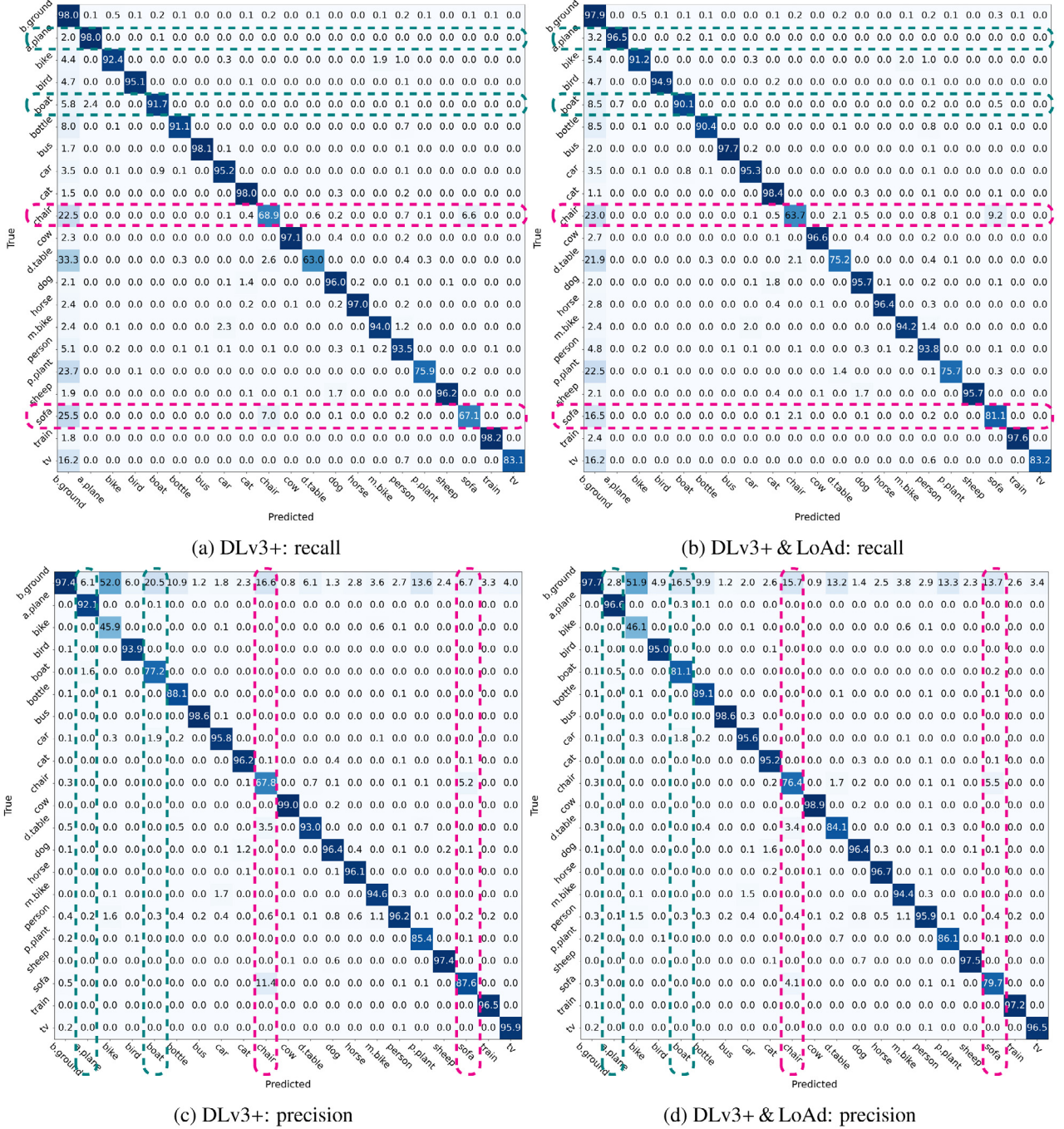
(c) DLv3+: precision

(d) DLv3+ & LoAd: precision

Fig. C.9. Confusion matrices (CM) on PASCAL VOC 2012.

beyond the current starting model mIoU by at least β_u in mIoU sense, we reset both γ and ω and allow this upward trend to continue and keep updating the best model. This is delineated in lines 15 to 20.

At the end of this inner while loop (lines 7 to 24), we are expected to mark one or two models: an ending model and possibly a new peak. The ending model of the cycle g^e (line 25) will be passed to the label map aggregation module regardless of whether we find a new peak or not. In case the dynamic peak finding process discovers a new peak that is better than the initial model of the cycle ($\mu^* > \mu^0$), then we pass both the new peak model g^* and the ending model g^e to our label map aggregation module. Lines 29 and 32 highlight the two ways in which the label map aggregation module can be invoked, with and without a new peak found. Based on that the map aggregation buffer $\mathcal{B}$ will be

updated at the end of each cycle. The containment of the updated buffer will be concatenated with the full training dataset ($\mathcal{D}_t \cup \mathcal{B}$) and will be used to retrain the discriminator (line 35). As discussed in Section 2, we do not follow a standard GDA approach and every time train the discriminator with this dynamically updated dataset until a sufficient accuracy is reached. Lastly, a new peak serves as a new starting model based on which counting the cycles will be restarted (lines 27, 28 and 30). On the other hand, if no new peak is found, we go back to where we started and continue (lines 32 and 33) hoping for finding a new peak in the next cycle. We fully stop the algorithm if we run Ψ cycles and a new peak is not found. This process is also sketched in Fig. B.8.

(a) DLv3+: recall

(b) DLv3+ & LoAd: recall

(c) DLv3+: precision
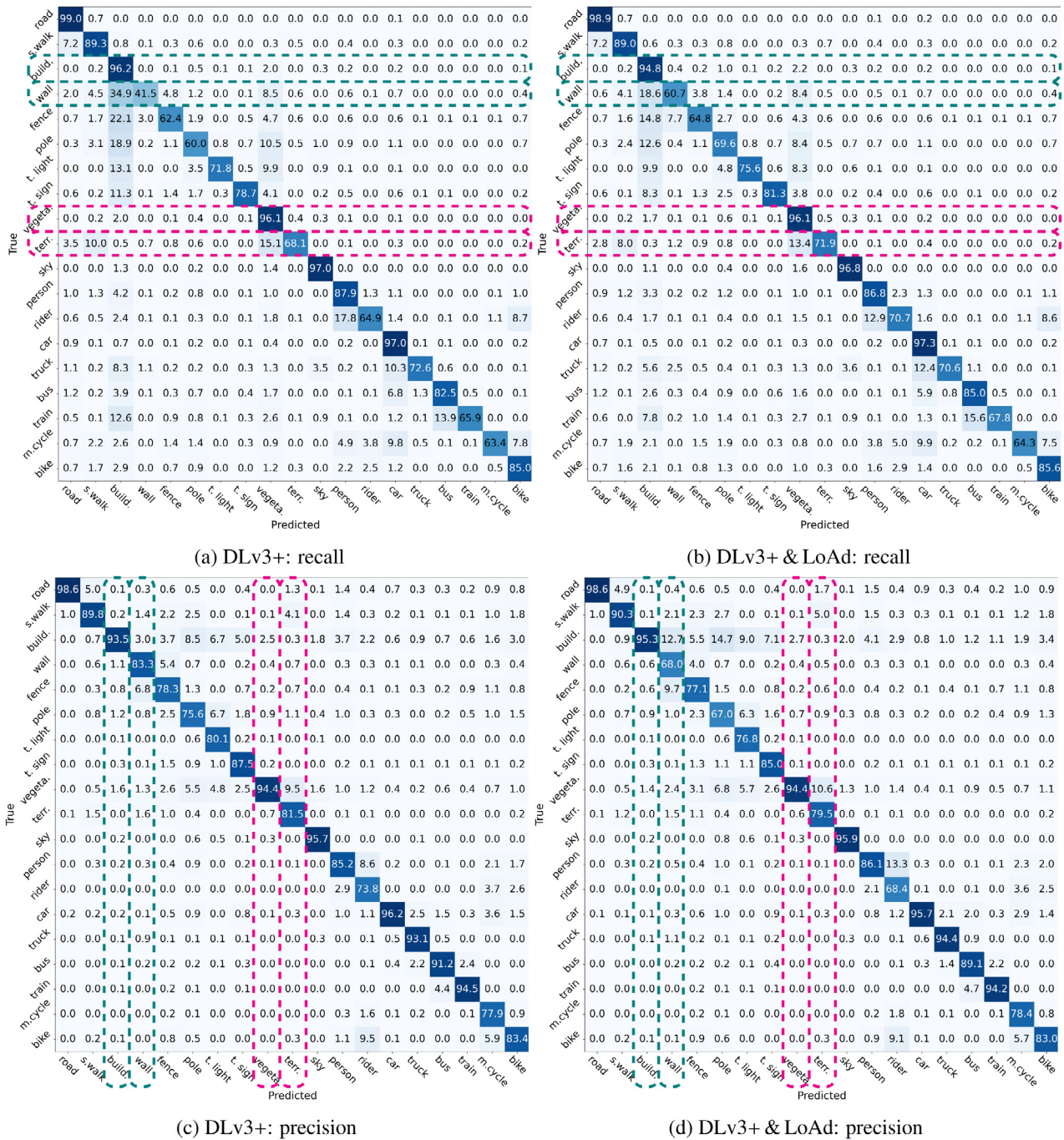
(d) DLv3+ & LoAd: precision

Fig. C.10. Confusion matrices (CM) on Cityscapes.

Appendix C. More quantitative results

Here, we delve deeper into the impact of LoAd on resolving class swap/confusion in comparison with DLv3+. To this aim, row/column-normalized confusion matrices are presented. Note that confusion matrices (in normalized pixel counts) are calculated across the whole validation set of the corresponding datasets. When normalized by row, these represent recall values on the diagonal, and in the case of normalization by column, they represent precision values. The selected classes are highlighted with dashed lines of the same color. As an example, "sofa" vs. "chair" in Fig. C.9. From recall perspective, comparing (a) and (b), chair is more confused with "sofa" by LoAd (9.2% vs. 6.6%), but "sofa" is significantly less confused with "chair" by LoAd (2.1% vs 7%). In line with this, the recall increase of "sofa" (67.1% → 81.1%)

is larger than the decrease in "chair" (68.9% → 63.7%). The opposite trend can be seen in precision sense, comparing (c) and (d), but again the impact on the confusion of "chair" with "sofa" (11.4% → 4.1%) is considerably larger than the other way around (5.2% → 5.5%). Thus, overall, the confusion of these two classes has been alleviated to some extent as a result of applying LoAd. Another way to look at this in a combined fashion (incorporating both precision and recall) is the IoU of these classes in Table 1, which are both increased when comparing LoAd and baseline DLv3+. A similar analysis can be applied to other highlighted class pairs in Figs. C.9 and C.10.

References

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein generative adversarial networks. In: International Conference on Machine Learning. PMLR, pp. 214–223.

Chen, Y., Dapogny, A., Cord, M., 2020. SEMEDA: Enhancing segmentation precision with semantic edge aware loss. Pattern Recognit. 108, 107557.

Chen, C., Dou, Q., Chen, H., Heng, P.-A., 2018a. Semantic-aware generative adversarial nets for unsupervised domain adaptation in chest x-ray segmentation. In: International Workshop on Machine Learning in Medical Imaging. Springer, pp. 143–151.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell. 40 (4), 834–848.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 801–818.

Cheng, B., Collins, M.D., Zhu, Y., Liu, T., Huang, T.S., Adam, H., Chen, L.-C., 2020. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12475–12485.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3213–3223.

Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A., 2015. The pascal visual object classes challenge: A retrospective. Int. J. Comput. Vis. 111 (1), 98–136.

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The kitti dataset. Int. J. Robot. Res. 32 (11), 1231–1237.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680.

Gould, S., Fulton, R., Koller, D., 2009. Decomposing a scene into geometric and semantically consistent regions. In: 2009 IEEE 12th International Conference on Computer Vision. IEEE, pp. 1–8.

Henry, C., Azimi, S.M., Merkle, N., 2018. Road segmentation in SAR satellite images with deep fully convolutional neural networks. IEEE Geosci. Remote Sens. Lett. 15 (12), 1867–1871.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Hu, X., Fuxin, L., Samaras, D., Chen, C., 2019. Topology-preserving deep image segmentation. In: Conference on Neural Information Processing Systems (NeurIPS).

Hung, W.-C., Tsai, Y.-H., Liou, Y.-T., Lin, Y.-Y., Yang, M.-H., 2018. Adversarial learning for semi-supervised semantic segmentation. In: British Machine Vision Conference (BMVC).

Ian Goodfellow, J.S., Szegedy, C., 2015. Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR).

Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A., 2017. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134.

Johnson, J., Alahi, A., Fei-Fei, L., 2016. Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. Springer, pp. 694–711.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. Int. J. Comput. Vis. 1 (4), 321–331.

Ke, T.-W., Hwang, J.-J., Liu, Z., Yu, S.X., 2018. Adaptive affinity fields for semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 587–602.

Kervadec, H., Bouchtiba, J., Desrosiers, C., Granger, E., Dolz, J., Ayed, I.B., 2021. Boundary loss for highly unbalanced segmentation. Med. Image Anal. 67, 101851.

Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P., 2019. Panoptic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9404–9413.

Lin, T., Jin, C., Jordan, M., 2020. On gradient descent ascent for nonconvex-concave minimax problems. In: International Conference on Machine Learning. PMLR, pp. 6083–6093.

Liu, X., Hsieh, C.-J., 2019. Rob-GAN: Generator, discriminator, and adversarial attacker. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11234–11243.

Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X., 2017. Deep learning markov random field for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 40 (8), 1814–1828.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431–3440.

Luc, P., Couprie, C., Chintala, S., Verbeek, J., 2016. Semantic segmentation using adversarial networks. Proc. Neural Inf. Process. (NeurIPS).

Menze, M., Geiger, A., 2015. Object scene flow for autonomous vehicles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3061–3070.

Minaee, S., Wang, Y., 2019. An ADMM approach to masked signal decomposition using subspace representation. IEEE Trans. Image Process. 28 (7), 3192–3204.

Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Mosinska, A., Marquez-Neila, P., Koziński, M., Fua, P., 2018. Beyond the pixel-wise loss for topology-aware delineation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3136–3145.

Nock, R., Nielsen, F., 2004. Statistical region merging. IEEE Trans. Pattern Anal. Mach. Intell. 26 (11), 1452–1458.

Nouiehed, M., Sanjabi, M., Huang, T., Lee, J.D., Razaviyayn, M., 2019. Solving a class of non-convex min-max games using iterative first order methods. In: Advances in Neural Information Processing Systems. pp. 14905–14916.

Ostrovskii, D.M., Lowy, A., Razaviyayn, M., 2020. Efficient search of first-order nash equilibria in nonconvex-concave smooth min-max problems. arXiv preprint arXiv: 2002.07919.

Plath, N., Toussaint, M., Nakajima, S., 2009. Multi-class image segmentation using conditional random fields and global classification. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML), pp. 817–824.

Rezaei, M., Harmuth, K., Gierke, W., Kellermeier, T., Fischer, M., Yang, H., Meinel, C., 2017. A conditional adversarial network for semantic segmentation of brain tumor. In: International MICCAI Brainlesion Workshop. Springer, pp. 241–252.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 234–241.

Ross, S., Gordon, G., Bagnell, D., 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 627–635.

Roth, K., Lucchi, A., Nowozin, S., Hofmann, T., 2017. Stabilizing training of generative adversarial networks through regularization. In: Advances in Neural Information Processing Systems. pp. 2018–2028.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., 2016. Improved techniques for training gans. In: Advances in Neural Information Processing Systems. pp. 2234–2242.

Shen, F., Gan, R., Yan, S., Zeng, G., 2017. Semantic segmentation via structured patch prediction, context crf and guidance crf. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1953–1961.

Shvets, A.A., Rakhlin, A., Kalinin, A.A., Iglovikov, V.I., 2018. Automatic instrument segmentation in robot-assisted surgery using deep learning. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, pp. 624–628.

Souly, N., Spampinato, C., Shah, M., 2017. Semi supervised semantic segmentation using generative adversarial network. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5688–5696.

Terzopoulos, D., et al., 2021. Progressive adversarial semantic segmentation. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, pp. 4910–4917.

Volpi, M., Ferrari, V., 2015. Semantic segmentation of urban scenes by learning local class interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 1–9.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al., 2020. Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell..

Xue, Y., Xu, T., Zhang, H., Long, L.R., Huang, X., 2018. SegAN: Adversarial network with multi-scale L1 loss for medical image segmentation. Neuroinformatics 16 (3–4), 383–392.

Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., Agrawal, A., 2018. Context encoding for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7151–7160.

Zhang, M., Lucas, J., Ba, J., Hinton, G.E., 2019. Lookahead optimizer: k steps forward, 1 step back. In: Advances in Neural Information Processing Systems. pp. 9593–9604.

Zhang, Z., Zhang, X., Peng, C., Xue, X., Sun, J., 2018. Exfuse: Enhancing feature fusion for semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 269–284.

Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., 2017. Pyramid scene parsing network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2881–2890.

Zhao, S., Wang, Y., Yang, Z., Cai, D., 2019a. Region mutual information loss for semantic segmentation. In: Advances in Neural Information Processing Systems. pp. 11115–11125.

Zhao, S., Wu, B., Chu, W., Hu, Y., Cai, D., 2019b. Correlation maximized structural similarity loss for semantic segmentation. arXiv preprint arXiv:1910.08711.