Single-Stage Semantic Segmentation from Image Labels
– Supplemental Material –

Nikita Araslanov Stefan Roth
Department of Computer Science, TU Darmstadt

A. On Training Stages

Table 4 provides a concise overview of previous work on semantic segmentation from image-level labels. As an important practical consideration, we highlight the number of stages used by each method. In this work, we refer to one stage as learning an independent set of model parameters with intermediate results saved *offline* as input to the next stage. For example, the approach by Ahn *et al*. [3] comprises three stages: *(1)* extracting CAMs as seed masks; *(2)* learning a pixel affinity network to refine these masks; and *(3)* training a segmentation network on the pseudo labels generated by affinity propagation. Note that three methods [26, 55, 56] also use multiple training cycles (given in parentheses) of the same model, which essentially acts as a multiplier w.r.t. the total number of stages. Finally, we note if a method relies on saliency detection, extra data, or previous frameworks. We observe that predominantly early single-stage methods stand in contrast to the more complex recent multi-stage pipelines. Our approach is single stage and relies neither on previous frameworks, nor on saliency supervision.[1]

Method	Extras	# of Stages	PSR
MIL-FCN *ICLR '15* [40]	–	1	✗
MIL-LSE *CVPR '15* [41]	–	1	✗
EM *ICCV '15* [38]	–	1	✗
TransferNet *CVPR '16* [17]	–	1	✗
SEC *ECCV '16* [28]	$\mathcal{S}$ [46]	2	✗
DCSP *ECCV '17* [5]	$\mathcal{S}$ [35]	1	✗
AdvErasing *CVPR '17* [55]	$\mathcal{S}$ [53]	2 ($\times$3)	✓
WebCrawl *CVPR '17* [18]	$\mathcal{D}$	1	✗
CRF-RNN *CVPR '17* [43]	–	1	✗
STC *TPAMI '17* [56]	$\mathcal{D}$, $\mathcal{S}$ [25]	3 ($\times$3)	✓
MCOF *CVPR '18* [54]	$\mathcal{S}$ [54]	2	✗
RDC *CVPR '18* [57]	$\mathcal{S}$ [59]	3	✓
DSRG *CVPR '18* [24]	$\mathcal{S}$ [53]	2	✓
Guided-Att *CVPR '18* [32]	SEC [28]	1+2	✗
SalientInstances *ECCV '18* [14]	$\mathcal{S}$ [13]	2	✓
Affinity *CVPR '18* [3]	–	3	✓
SeeNet *NIPS '18* [21]	$\mathcal{S}$ [20]	2	✓
FickleNet *CVPR '19* [30]	$\mathcal{S}$, DSRG [24]	1+3	✓
JointSaliency *ICCV '19* [60]	$\mathcal{S}$	1	✗
Frame-to-Frame *ICCV '19* [31]	$\mathcal{D}$, $\mathcal{S}$ [19]	2	✓
SSDD *ICCV '19* [45]	Affinity [3]	2+3	✓
IRN *CVPR '19* [2]	–	3	✓
Coarse-to-Fine *TIP '20* [26]	GrabCut [42]	2 ($\times$5)	✓
Ours	–	1	✗

Table 4. **Summary of related work.** We analyse the related methods w.r.t. external input ("Extras"), such as saliency detection ($\mathcal{S}$), additional data ($\mathcal{D}$), or their reliance on previous work. We count the number of training stages in the method and note in parentheses if the method uses multiple training cycles of the same models. We also mark methods that additionally train a standalone segmentation network in a (pseudo) fully supervised regime to refine the masks (PSR).

B. Loss Functions

In this section, we take a detailed look at the employed loss functions. Since we compute the classification scores differently from previous work, we also provide additional analysis to justify the form of this novel formulation.

B.1. Classification loss

We use the multi-label soft-margin loss function used in previous work [3, 57] as the classification loss, $\mathcal{L}_{\text{cls}}$. Given model predictions $\mathbf{y} \in \mathbb{R}^C$ (*cf*. Eq. (3) and Eq. (5); see also below) and a binary vector of ground-truth labels $\mathbf{z} \in \{0,1\}^C$, we compute the multi-label soft-margin loss [39] as

$$\mathcal{L}_{\text{cls}}(\mathbf{y},\mathbf{z}) = -\frac{1}{C}\sum_{c=1}^{C} z_c \log\left(\frac{1}{1+e^{-y_c}}\right) + \\ + (1-z_c)\log\left(\frac{e^{-y_c}}{1+e^{-y_c}}\right). \quad (12)$$

As illustrated in Fig. 9, the loss function encourages $y_c < 0$ for negative classes (*i.e*. when $z_c = 0$) and $y_c > 0$ for positive classes (*i.e*. when $z_c = 1$). This observation will be useful for our discussion below.

[1] The background cues provided by saliency detection methods give a substantial advantage, since 63% of 10K train images (VOC+SBD) have only one class.

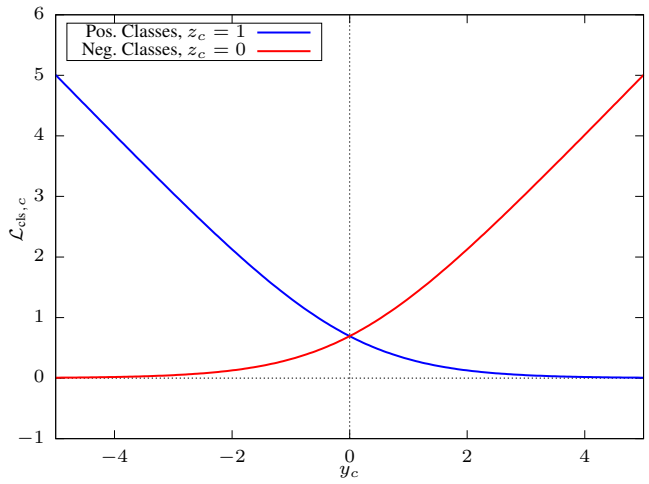

Figure 9. **Soft-margin classification loss.** The loss encourages $y_c > 0$ for positive classes and $y_c < 0$ for negative classes. The function exhibits saturation regions: as $y_c \to \infty$ for positive classes, the associated loss (shown in blue) approaches 0. Conversely, the loss for negative classes, shown in red, approaches 0, as $y_c \to -\infty$.

Recall from Eq. (3) and Eq. (5) that we define our classification score y_c as

$$y_c = y_c^{\text{nGWP}} + y_c^{\text{size-focal}}. \tag{13}$$

For convenience, we re-iterate the definition of the two terms, the normalised Global Weighted Pooling

$$y_c^{\text{nGWP}} = \frac{\sum_{i,j} m_{c,i,j} y_{c,i,j}}{\epsilon + \sum_{i',j'} m_{c,i',j'}} \tag{14}$$

and the focal penalty

$$y_c^{\text{size-focal}} = (1 - \bar{m}_c)^p \log(\lambda + \bar{m}_c), \tag{15}$$

$$\text{with} \qquad \bar{m}_c = \frac{1}{hw} \sum_{i,j} m_{c,i,j}. \tag{16}$$

Note that $y_{c,i,j}$ in Eq. (14) refers to pixel site (i,j) on the score map of class c, whereas y_c in Eq. (13) is the *aggregated* score for class c. We compute the class confidence $m_{:,i,j}$ for site (i,j) using a `softmax` on $y_{:,i,j}$, where we include an additional *background* channel $y_{0,i,j}$. We fix $y_{0,:,:} \equiv 1$ throughout our experiments.

The use of hyperparameter $\epsilon > 0$ in the definition of Eq. (14) may look rather arbitrary and redundant at first. In the analysis below, we show that in fact it serves multiple purposes:

1. **Numerical stability**. First, using $\epsilon > 0$ prevents division by zero for saturated scores, *i.e.* where $\sum_{i',j'} m_{c,i',j'} = 0$ for some c in the denominator of Eq. (14), which can happen (approximately) in the course of training for negative classes.

   Secondly, $\epsilon > 0$ resolves discontinuity issues. Observe that

   $$\lim_{m_{c,:,:} \to 0} \frac{\sum_{i,j} m_{c,i,j} y_{c,i,j}}{\epsilon + \sum_{i',j'} m_{c,i',j'}} = 0, \tag{17}$$

   i.e. $y_c^{\text{nGWP}} \approx 0$ for negative classes and positive ϵ.

   However, with $\epsilon = 0$ the nGWP term in Eq. (14) is not continuous at 0, which in practice may result in unstable training when $\sum_{i,j} m_{c,i,j} \approx 0$ for some c. One exception, unlikely to occur in practice, is when $y_{c,i,j} = y_{c,k,l} = d$ for all (i,j) and (k,l). Then,

   $$\lim_{m_{c,:,:} \to 0} \frac{\sum_{i,j} m_{c,i,j} y_{c,i,j}}{\sum_{i',j'} m_{c,i',j'}} = \\ = \lim_{m_{c,:,:} \to 0} \frac{d \sum_{i,j} m_{c,i,j}}{\sum_{i',j'} m_{c,i',j'}} = d. \tag{18}$$

   In the case of more practical relevance, *i.e.* when $y_{c,i,j} \neq y_{c,k,l}$ for some $(i,j) \neq (k,l)$, the limit does not exist, as the following lemma shows. With some abuse of notation, we here write m_i and y_i to refer to the confidence and the score values of class c at pixel site i, respectively.

   Lemma 1. Let $\epsilon = 0$ in Eq. (14) and suppose there exist k and l such that $y_k \neq y_l$. Then, the corresponding limit

   $$\lim_{m_i \to 0, \forall i} \frac{\sum_i m_i y_i}{\sum_{i'} m_{i'}} \tag{19}$$

   does not exist.

   Proof. Let $m_k(t) = t$ and $m_i(t) = 0$ for $i \neq k$. Then,

   $$\lim_{m_i \to 0, \forall i} \frac{\sum_i m_i y_i}{\sum_{i'} m_{i'}} = \lim_{t \to 0} \frac{t y_k}{t} = y_k.$$

   On the other hand, if we let $m_l(t) = t$ and $m_i(t) = 0$ for $i \neq l$, we obtain

   $$\lim_{m_i \to 0, \forall i} \frac{\sum_i m_i y_i}{\sum_{i'} m_{i'}} = \lim_{t \to 0} \frac{t y_l}{t} = y_l.$$

   Since $y_k \neq y_l$ by our assumption, we have now found two paths of the multivariable limit in Eq. (19), which evaluate to different values. Therefore, the limit in Eq. (19) is not unique, *i.e.* does not exist ([64], Ch. 14.2). □

2. **Emphasis on the focal penalty**. For negative classes, it is not sensible to give meaningful relative weightings

of the pixels for nGWP in Eq. (14); we seek such a relative weighting of different pixels only for positive classes. At training time we would thus like to minimise the class scores for negative classes *uniformly* for all pixel sites. Empirically, we observed that with $\epsilon > 0$ the focal penalty term, which encourages this behaviour, contributes more significantly to the score of the negative classes than the nGWP term, which relies on relative pixel weighting.

3. **Negative class debiasing**. Negative classes dominate in the label set of Pascal VOC, *i.e.* each image sample depicts only few categories. The gradient of the loss function from Eq. (12) vanishes for positive classes with $y_c \to \infty$ and negative classes with $y_c \to -\infty$. However, in our preliminary experiments with GAP-CAM, and later with nGWP with $\epsilon = 0$ in Eq. (14), we observed that further iterations continued to decrease the scores for the negative classes in regions in which the loss is near-saturated, while the y_c of positive classes increased only marginally. This may indicate a strong inductive bias towards negative classes, which might be undesirable for real-world deployment.

   Assuming $\epsilon > 0$ and $\sum_{i',j'} m_{c,i',j'} \neq 0$, we observe that,

$$\frac{\sum_{i,j} m_{c,i,j} y_{c,i,j}}{\epsilon + \sum_{i',j'} m_{c,i',j'}} > \frac{\sum_{i,j} m_{c,i,j} y_{c,i,j}}{\sum_{i',j'} m_{c,i',j'}} \tag{20}$$

   when $y_{c,:,:} < 0$, which is the case for saturated negative classes. Recall further that the use of the constant background score (fixed to 1) implies that $m_{c,i,j} \to 0$ as $y_{c,i,j} \to -\infty$. Since the RHS is the convex combination of $y_{c,i,j}$'s, its minimum is $\min(\{y_{c,i,j}\}_{i,j})$. Therefore, the RHS is unbounded from below as $y_{c,i,j} \to -\infty$, hence the $y_{c,i,j}$ keep getting pushed down for negative classes. By contrast, the LHS has a defined limit of 0 as $m_{c,i,j} \to 0$ (see Eq. (17)), which is undesirable as the score for a negative class. This is because a finite (*i.e.* larger) $m_{c,i,j}$ yields a negative, thus smaller value of the classification score, which we are trying to minimise in this case. Therefore, $\epsilon > 0$ will encourage SGD to pull the negative scores away from the saturation areas by pushing the $\sum_{i,j} m_{c,i,j}$ away from zero.

In summary, for negative classes ϵ improves numerical stability and emphasises the focal penalty while leveraging nGWP to alleviate the negative class bias. For positive classes, the effect of ϵ is negligible, since $\epsilon \ll \sum_{i,j} m_{c,i,j}$ in this case. We set $\epsilon = 1$ in all our experiments.

B.2. Segmentation loss

For the segmentation loss w.r.t. the pseudo ground truth, we use a *weighted* cross-entropy defined for each pixel site (i, j) as

$$\mathcal{L}_{\text{seg},i,j} = -q_c \log m_{c,i,j}, \tag{21}$$

where c is the label in the pseudo ground-truth mask. The balancing class weight q_c accounts for the fact that the pseudo ground truth may contain a different amount of pixel supervision for each class:

$$q_c = \frac{M_{b,\text{total}} - M_{b,c}}{1 + M_{b,\text{total}}}, \tag{22}$$

where $M_{b,\text{total}}$ and $M_{b,c}$ are the total and class-specific number of pixels for supervision in the pseudo ground-truth mask, respectively; b indexes the sample in a batch; and 1 in the denominator serves for numerical stability. The aggregated segmentation loss $\mathcal{L}_{\text{seg}}$ is a weighted mean over the samples in a batch, *i.e.*

$$\mathcal{L}_{\text{seg}} = \frac{1}{\sum_{b'} M_{b',\text{total}}} \frac{1}{hw} \sum_{b} M_{b,\text{total}} \sum_{i,j} \mathcal{L}_{\text{seg},i,j}. \tag{23}$$

C. Quantitative Analysis

We list the per-class segmentation accuracy on Pascal VOC validation and training in Tables 5 and 6. We first observe that none of previous methods, including the state of the art [3, 30, 45], outperforms other pipelines on *all* class categories. For example FickleNet [30], based on a ResNet-101 backbone, reaches top segmentation accuracy only for classes "bottle", "bus", "car", and "tv". SSDD [45] has the highest mean IoU, but is inferior to other methods on 10 out of 21 classes. Our single-stage method compares favourably even to the multi-stage approaches that rely on saliency supervision or additional data. For example, we improve over the more complex AffinityNet [3] that trains a deep network to predict pixel-level affinity distance from CAMs and then further trains a segmentation network in a (pseudo) fully supervised regime. The best single-stage method from previous work, CRF-RNN [43], trained using only the image-level annotation we consider in this work, reaches 52.8% and 53.7% IoU on the validation and test sets. We substantially boost this result, by 9.9% and 10.6% points, respectively, and attain new a state-of-the-art mask accuracy overall on classes "bike", "person", and "plant".

D. Ablation Study: PAMR Iterations

We empirically verify the number of iterations used in PAMR, which we set to 10 in our main ablation study. Table 7 reports the results with higher and fewer number of iterations. We observe that using only few PAMR iterations decreases the mask quality. On the other hand, the benefits of PAMR diminishes if we increase the number of iterations further. 10 iterations appears to strike a good balance between the computational expense and the obtained segmentation accuracy.

Method	bg	aero	bike	bird	boat	bot.	bus	car	cat	chair	cow	tab.	dog	horse	mbk	per.	plant	sheep	sofa	train	tv	mean
Multi stage																						
SEC* [28]	82.4	62.9	26.4	61.6	27.6	38.1	66.6	62.7	75.2	22.1	53.5	28.3	65.8	57.8	62.3	52.5	32.5	62.6	32.1	45.4	45.3	50.7
AdvErasing* [55]	83.4	71.1	30.5	72.9	41.6	55.9	63.1	60.2	74.0	18.0	66.5	32.4	71.7	56.3	64.8	52.4	37.4	69.1	31.4	58.9	43.9	55.0
RDC* [57]	89.4	85.6	34.6	75.8	61.9	65.8	67.1	73.3	80.2	15.1	69.9	8.1	75.0	68.4	70.9	71.5	32.6	74.9	24.8	73.2	50.8	60.4
FickleNet* [30]	88.1	75.0	31.3	75.7	48.8	60.1	80.0	72.7	79.6	25.7	67.3	42.2	77.1	67.5	65.4	69.2	42.2	74.1	34.2	53.7	54.7	61.2
AffinityNet [3]	88.2	68.2	30.6	81.1	49.6	61.0	77.8	66.1	75.1	29.0	66.0	40.2	80.4	62.0	70.4	73.7	42.5	70.7	42.5	68.1	51.6	61.7
FickleNet*,† [30]	89.5	76.6	32.6	74.6	51.5	71.1	83.4	74.4	83.6	24.1	73.4	47.4	78.2	74.0	68.8	73.2	47.8	79.9	37.0	57.3	64.6	64.9
SSDD [45]	89.0	62.5	28.9	83.7	52.9	59.5	77.6	73.7	87.0	34.0	83.7	47.6	84.1	77.0	73.9	69.6	29.8	84.0	43.2	68.0	53.4	64.9
Single stage																						
TransferNet* [17]	85.3	68.5	26.4	69.8	36.7	49.1	68.4	55.8	77.3	6.2	75.2	14.3	69.8	71.5	61.1	31.9	25.5	74.6	33.8	49.6	43.7	52.1
CRF-RNN [43]	85.8	65.2	29.4	63.8	31.2	37.2	69.6	64.3	76.2	21.4	56.3	29.8	68.2	60.6	66.2	55.8	30.8	66.1	34.9	48.8	47.1	52.8
WebCrawl* [18]	87.0	69.3	32.2	70.2	31.2	58.4	73.6	68.5	76.5	26.8	63.8	29.1	73.5	69.5	66.5	70.4	46.8	72.1	27.3	57.4	50.2	58.1
Ours	87.0	63.4	33.1	64.5	47.4	63.2	70.2	59.2	76.9	27.3	67.1	29.8	77.0	67.2	64.0	72.4	46.5	67.6	38.1	68.2	63.6	59.7
Ours + CRF	88.7	70.4	35.1	75.7	51.9	65.8	71.9	64.2	81.1	30.8	73.3	28.1	81.6	69.1	62.6	74.8	48.6	71.0	40.1	68.5	64.3	62.7

Methods marked with (*) use saliency detectors or additional data, or both (see Sec. 2). (†) denotes a ResNet-101 backbone.

Table 5. Per-class IoU (%) comparison on Pascal VOC 2012, validation set.

Method	bg	aero	bike	bird	boat	bot.	bus	car	cat	chair	cow	tab.	dog	horse	mbk	per.	plant	sheep	sofa	train	tv	mean
Multi stage																						
SEC* [28]	83.5	56.4	28.5	64.1	23.6	46.5	70.6	58.5	71.3	23.2	54.0	28.0	68.1	62.1	70.0	55.0	38.4	58.0	39.9	38.4	48.3	51.7
FickleNet* [30]	88.5	73.7	32.4	72.0	38.0	62.8	77.4	74.4	78.6	22.3	67.5	50.2	74.5	72.1	77.3	68.8	52.5	74.8	41.5	45.5	55.4	61.9
AffinityNet [3]	89.1	70.6	31.6	77.2	42.2	68.9	79.1	66.5	74.9	29.6	68.7	56.1	82.1	64.8	78.6	73.5	50.8	70.7	47.7	63.9	51.1	63.7
FickleNet*,† [30]	89.8	78.3	34.1	73.4	41.2	67.2	81.0	77.3	81.2	29.1	72.4	47.2	76.8	76.5	76.1	72.9	56.5	82.9	43.6	48.7	64.7	65.3
SSDD [45]	89.5	71.8	31.4	79.3	47.3	64.2	79.9	74.6	84.9	30.8	73.5	58.2	82.7	73.4	76.4	69.9	37.4	80.5	54.5	65.7	50.3	65.5
Single stage																						
TransferNet* [17]	85.7	70.1	27.8	73.7	37.3	44.8	71.4	53.8	73.0	6.7	62.9	12.4	68.4	73.7	65.9	27.9	23.5	72.3	38.9	45.9	39.2	51.2
CRF-RNN [43]	85.7	58.8	30.5	67.6	24.7	44.7	74.8	61.8	73.7	22.9	57.4	27.5	71.3	64.8	72.4	57.3	37.3	60.4	42.8	42.2	50.6	53.7
WebCrawl* [18]	87.2	63.9	32.8	72.4	26.7	64.0	72.1	70.5	77.8	23.9	63.6	32.1	77.2	75.3	76.2	71.5	45.0	68.8	35.5	46.2	49.3	58.7
Ours	87.4	63.6	34.7	59.9	40.1	63.3	70.2	56.5	71.4	29.0	71.0	38.3	76.7	73.2	70.5	71.6	55.0	66.3	47.0	63.5	60.3	60.5
Ours + CRF	89.2	73.4	37.3	68.3	45.8	68.0	72.7	64.1	74.1	32.9	74.9	39.2	81.3	74.6	72.6	75.4	58.1	71.0	48.7	67.7	60.1	64.3

Methods marked with (*) use saliency detectors or additional data, or both (see Sec. 2). (†) denotes a ResNet-101 backbone.

Table 6. Per-class IoU (%) comparison on Pascal VOC 2012, test set.

Number of iterations	IoU	IoU w/ CRF
5	59.1	59.0
10	**59.4**	**62.2**
15	59.0	61.6

Table 7. **Effect of the iteration number in PAMR.** We train our model with the iteration number in the PAMR module fixed to a pre-defined value. We report the IoU (%) with and without CRF refinement on Pascal VOC validation.

Additionally, we visualise semantic masks produced at intermediate iterations from our PAMR module in Fig. 10. The initial masks produced by our segmentation model in the early stages of training exhibit coarse boundaries. PAMR mitigates this shortcoming by virtue of exploiting visual cues with pixel-adaptive convolutions. Our model then uses the revised masks to generate pseudo ground-truth for self-supervised segmentation.

E. Pseudo Labels

The last-stage training of a segmentation network is *agnostic* to the process of pseudo-label generation; it is the quality of the pseudo labels and the ease of obtaining them that matters.

Although we intentionally omitted the common practice of training a standalone network on our pseudo labels, we show that, in fact, we can achieve state-of-the-art results in a multi-stage setting as well.

We use our pseudo labels on the train split of Pascal VOC (see Table 2) and train a segmentation model DeepLabv3+ [8] in a fully supervised fashion. Table 8 summarises the results. We observe that the resulting simple two-stage pipeline outperforms other multi-stage frameworks under the same image-level supervision. Remarkably, our method even attains the mask accuracy of Frame-to-Frame [31], which not only utilises saliency detectors, but also relies on additional data (15K extra) and sophisti-

Method	Backbone	Supervision	val	test
Fully supervised				
WideResNet38 [58]		$\mathcal{F}$	80.8	82.5
DeepLabv3+ [8]	Xception-65 [9]	$\mathcal{F}$	–	87.8
Multi stage + Saliency				
FickleNet [30]	ResNet-101	$\mathcal{S}$	64.9	65.3
Frame-to-Frame [31]	VGG-16	$\mathcal{S}, \mathcal{D}$	63.9	65.0
	ResNet-101		66.5	67.4
Single stage + Saliency				
JointSaliency [60]	DenseNet-169 [22]	$\mathcal{S}, \mathcal{D}$	63.3	64.3
Multi stage				
AffinityNet [3]	WideResNet-38	$\mathcal{I}$	61.7	63.7
IRN [2]	ResNet-50	$\mathcal{I}$	63.5	64.8
SSDD [45]	WideResNet-38	$\mathcal{I}$	64.9	65.5
Two stage				
Ours + DeepLabv3+	ResNet-101	$\mathcal{I}$	65.7	66.6
Ours + DeepLabv3+	Xception-65	$\mathcal{I}$	66.8	67.3

Table 8. **Mean IoU (%) on Pascal VOC validation and test sets.** We train DeepLabv3+ in a fully supervised regime on pseudo ground truth obtained from our method (with CRF refinement). Under equivalent level of supervision, our two-stage approach outperforms the previous state of the art, trained in three or more stages and performs on par with other multi-stage frameworks relying on additional data and saliency detection [31].

	Baseline (CAM)		Ours	
Backbone	w/o CRF	+ CRF	w/o CRF	+ CRF
VGG16	41.2	38.0	55.9	56.6
ResNet-50	43.7	43.5	60.4	64.1
ResNet-101	46.2	45.2	62.9	66.2
WideResNet-38	44.9	45.2	63.1	65.8
Mean	44.0	43.0	60.6	63.2

Table 9. **Segmentation quality (IoU, %) on Pascal VOC validation.** We use ground-truth image-level labels to remove false positive classes from the masks to decouple the segmentation quality from the accuracy of the classifier.

cated network models (*e.g.* PWC-Net [46], FickleNet [30]).

F. Exchanging Backbones

Here we confirm that our segmentation method generalises well to other backbone architectures. We choose VGG16 [47], ResNet-50 and ResNet-101 [16] – all widely used network architectures – as a drop-in alternative to WideResNet-38 [58], which we use for all other experiments. We train these models on 448×448 image crops using the same data augmentation as before. We use multi-scale inference with image sides varying by factors 1.0, 0.75, 1.25, 1.5. These scales are slightly different from the ones we used in the main experiments, which we found to slightly improve the IoU on average. We re-evaluate our WideResNet-38 on the same scales to make the results from different backbones compatible. Motivated to measure the segmentation accuracy alone, we report validation IoU on the masks with the false positives removed using ground-truth labels. Table 9 summarises the results.

The results show a clear improvement over the CAM baseline for all backbones: the average improvement without CRF post-processing is 16.6% IoU and 20.2% IoU with CRF refinement.

References

[64] James Stewart. *Calculus: Early transcendentals*. Thomson Higher Education, Belmont, CA, 6th edition, 2012. 2

Figure 10. **Visualisation of PAMR iterations.** The initial model predictions suffer from local inconsistency: mask boundaries do not align with available visual cues. Our PAMR module iteratively revises the masks to alleviate this problem. Our model uses the mask from the last iteration for self-supervision.