Self-Regulation for Semantic Segmentation

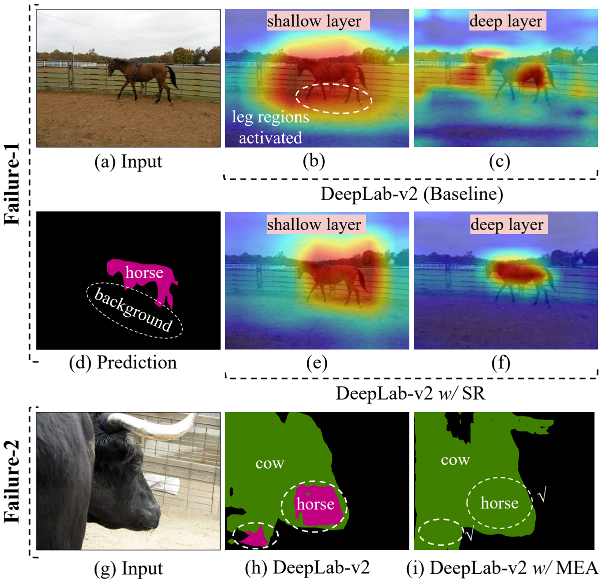
Dong Zhang[1] Hanwang Zhang[2] Jinhui Tang[1]* Xian-Sheng Hua[3] Qianru Sun[4]
[1]School of Computer Science and Engineering, Nanjing University of Science and Technology
[2]Nanyang Technological University [3]Damo Academy, Alibaba Group [4]Singapore Management University

Abstract

In this paper, we seek reasons for the two major failure cases in Semantic Segmentation (SS): 1) missing small objects or minor object parts, and 2) mislabeling minor parts of large objects as wrong classes. We have an interesting finding that Failure-1 is due to the underuse of detailed features and Failure-2 is due to the underuse of visual contexts. To help the model learn a better trade-off, we introduce several Self-Regulation (SR) losses for training SS neural networks. By "self", we mean that the losses are from the model per se without using any additional data or supervision. By applying the SR losses, the deep layer features are regulated by the shallow ones to preserve more details; meanwhile, shallow layer classification logits are regulated by the deep ones to capture more semantics. We conduct extensive experiments on both weakly and fully supervised SS tasks, and the results show that our approach consistently surpasses the baselines. We also validate that SR losses are easy to implement in various state-of-the-art SS models, e.g., *SPGNet [7] and OCRNet [62], incurring little computational overhead during training and none for testing*[1].

1. Introduction

Semantic Segmentation (SS) aims to label each image pixel with its corresponding semantic class [37]. It is an indispensable computer vision building block in the scene understanding systems, *e.g.*, autonomous driving [49] and medical imaging [18]. Thanks to the development of deep convolutional neural networks [20, 52] and the labour input in pixel-level annotations [8, 39], the research for SS has experienced a great progress, *e.g.*, top-performing models can segment about 85% objects in the complex natural scenes [33]. Intriguingly, we particularly seek reasons for the 15% failure cases which generally fall into two categories: 1) missing small objects or minor object parts, *e.g.*, the "horse legs" in Figure 1 (d), and 2) mislabeling minor parts of large objects as wrong classes, *e.g.*, a part of "cow" is wrongly marked as "horse" in Figure 1 (h).



(a) Input (b) (c)

(d) Prediction (e) (f)

(g) Input (h) DeepLab-v2 (i) DeepLab-v2 *w/* MEA

Figure 1. Our inspections on two failure cases of the baseline method DeepLab-v2 [5] in (d) and (h), for which (a) and (g) are input images, respectively. The case in (d) has the flaw of missing "horse legs". The visualization of shallow-layer features in (b) shows that the model indeed attends to "horse legs" but also to background noises. Applying our SR, in (e), reduces noises while maintaining the desirable attention on "horse legs". In addition, deep-layer features in (c) and (f) also prove the effectiveness of SR. The case in (h) confuses between object classes (*i.e.*, "horse" and "cow"). The revision in (i) is obtained by adding class-level cross-entropy losses (*i.e.*, using our MEA loss on DeepLab-v2).

To this end, we present our empirical inspections in Figure 1. We find in Figure 1 (b) that the minor object parts are clearly visible in the shallow-layers of the model, making use of which could address the issue in Failure-1, as in Figure 1 (e). In the literature, there is indeed a popular solution — reshaping and then combining feature maps from different layers for prediction, *e.g.*, Hourglass [40], SegNet [2], U-Net [45] and HRNet [52]. Such implementation typically relies on one of the three specific operations: pixel-wise addition [2, 36, 45], map-wise concatenation [5, 52, 71] and

*Corresponding author.

[1]The code is available at: https://github.com/dongzhang89/SR-SS

pixel-wise transformation [53, 65, 72]. However, they are expensive for deep backbones, *e.g.*, SPGNet [7] and HRNet [52] take 2.04× and 1.87× of model parameters more than their baselines, respectively.

Besides, what we observe from the "cow" example in Figure 1 is that penalizing the model with the image-level classification loss can mitigate Failure-2 [67, 70]: pixel-level confusion between the foreground objects (*i.e.*, the "cow") and the local pixel cues (*i.e.*, the "horse" part). The intuitive reason is that this loss penalizes the prediction of an unseen class-level context (*i.e.*, "cow" can not have a "horse" mouth). In this paper, we make use of this intuition and introduce a formal loss function to enhance the contextual encoding for each image.

Now, we present our overall approach, called *Self-Regulation* (SR), which has three major advantages: 1) generic to be implemented in any deep backbones, 2) without using any auxiliary data or supervision, and 3) with little overhead for training and none for testing. As shown in Figure 2, given a backbone network, we first add a pair of classifier and segmenter at each layer (*e.g.*, head networks for multi-label classification and semantic segmentation), and feed them with the corresponding feature maps. Then, we apply the proposed Self-Regulation for the features and logits by introducing the following three operations: **I**) Regulating the predictions of each pair with ground-truth labels, *i.e.*, the image-level classification labels and the pixel-level mask labels, respectively; **II**) Using the feature map of the *shallowest* layer — Shallow Teacher — to regulate *all the subsequent* deeper-layer feature maps — Deep Students; **III**) Using the prediction of the *deepest* layer classification logits — Deep Teacher — to regulate *all the previous* shallow-layer classification logits — Shallow Students. Here is our punchline: for pixel-level feature maps, we use one Shallow Teacher to teach many Deep Students; for image-level classification logits, we use one Deep Teacher to teach many Shallow Students.

Operation **I** is the standard supervised training that spreads over all the layers. This computation overhead (*e.g.*, multiple-exit network [44]) allows the ground-truth to supervise each layer at its earliest convenience. Operation **II** and Operation **III** aim to balance the trade-off among these supervisions. To implement them, we introduce a self-regulation method inspired by knowledge distillation (KD) [21, 35, 42]. Different from the conventional KD that needs to train a teacher model in prior, our "distillation" occurs in the same model during training. The shallowest layer retains the most details, so its segmentation behavior (*e.g.*, encoded in its feature maps and then decoded by the segmentation logits) can teach the subsequent deeper layers not to forget the details to avoid feature underuse (*e.g.*, Failure-1 in Figure 1); while the deepest layer retain the highest-level contextual semantics, so its classification behavior (*i.e.*, class logits) can teach the previous shallower layers not to ignore the contexts to avoid semantic underuse (*e.g.*, Failure-2 in Figure 1).

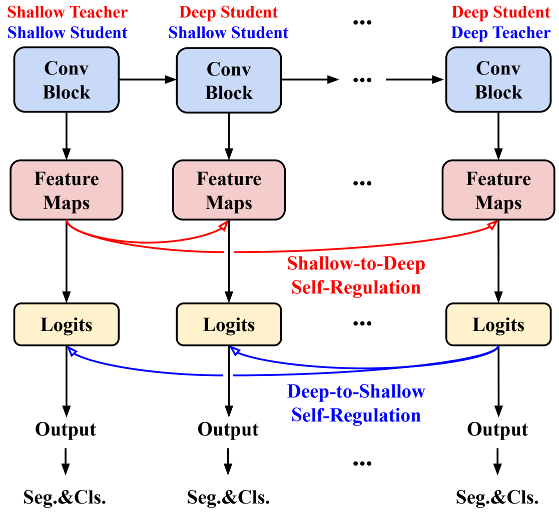


Figure 2. An illustrative example of implementing our proposed self-regulation (SR) to a plain Conv network. It is first converted to a multiple-exit architecture (MEA) where "Seg.&Cls." are a pair of classifier and segmenter for each layer. Shallow-to-deep SR means shallow-layer features regulate deep layers. Deep-to-shallow SR means deep-layer predictions regulate shallow layers.

Note that Operation **I** at each layer also protects any overuse. Specifically, if Operation **II** unnecessarily imposes shallow details on deep contexts, the image-level classification logits at deep layers will penalize it; similarly, if Operation **III** unnecessarily imposes deep contexts on shallow details, the pixel-level segmentation logits at shallow layers will discourage it. Such "shallow to deep and back" regulation collaborate with each other to improve the overall segmentation. Our empirical results on different baselines, *e.g.*, DeepLab-v2 [5], SegNet [2], SPGNet [7], and OCRNet [62], show that our SR is helpful to balance the use of low-/high-level semantics among different layers.

In summary, our contributions are two-fold: 1) A novel set of SR operations that tackle the two key issues in SS tasks while introducing little overhead for training and none for testing; and 2) Through extensive experiments in both fully-supervised and weakly-supervised SS settings, we validate that the proposed SR operations can be easily plugged-and-play and consistently improve various baseline models by a large margin.

2. Related Work

Semantic Segmentation (SS). The mainstream SS models are based on Fully Convolutional Networks (FCN) [37]. However, deeper layers of FCN often suffer from two problems: underusing detailed spatial information and insufficient receptive fields. Recent methods proposed to solve these issues can be classified into two camps: 1) encoder-decoder [41, 2, 7] based methods, and 2) atrous/dilated convolution [61] based methods. In the first camp, there are SegNet [2], De-convNet [41], and SPGNet [7]. Their common idea is to progressively aggregate high-resolution feature maps using a learnable decoder. In this way, detailed spatial information is embedded to deeper layers without any refinement on its semantic meaning, *i.e.*, such details may bring background noises to deep layers. In the second camp, there are DeepLab-v2 [5] and PSP [71], and their main idea is to use atrous/dilated convolutions or pooling operations to enable deeper layers to cover large receptive fields and to capture more contextual information [5, 67, 68, 71]. These methods fix the size of output feature maps to be $1/8$ or $1/16$ of the input image, so their output are too compressed to contain sufficient spatial details. In contrast to these two camps, our approach enriches the model with both contextual semantics and spatial details by leveraging optimization-based interactions between deep and shallow layers in FCN.

Multiple-Exit Architecture (MEA). MEA was firstly proposed in [23] for training image recognition models, aiming to improve the inference speed of the models [44]. Recently, MEA has been also successfully applied to boost the model performance [38, 48] in a variety of visual tasks, *e.g.*, image classification [10, 63], object detection [60], and semantic segmentation [66, 70]. Our differences with SS-related works [66, 70] lie in that our implementation of MAE integrates two visual tasks, *i.e.*, multi-label classification and semantic segmentation, in a single deep model.

Knowledge Distillation (KD). KD uses a pre-trained teacher model to produce soft labels as ground truth to train a student model [21]. It was originally introduced to compress a big network (teacher) to a small network (student), *i.e.*, network compression [25, 50]. It computes a distillation loss between teacher and student, for which the implementation methods can be logits-based [26, 44], feature-based [34, 57], and hybrid [28, 56]. In recent works, a self learning version called self KD, *i.e.*, teacher and student networks have the same architecture, achieved impressive results in several visual tasks, such as image classification [51, 57], object detection [28, 34], and pedestrian re-identification [6, 13]. Our approach is based on layer-wise self KD [69, 58]. We introduce a novel approach of bi-directional layer-wise teaching & learning to particularly enhance SS models. Our model is learned on a single network and our KD supervisions are applied across different layers. Compared to the existing KD methods, our approach is different in two aspects. First, it is self-regulated and does not require any pre-trained model as teacher. Second, it consists of two distillation directions each for a specific purpose of learning, *i.e.*, the shallow-to-deep regulation is to enrich the details in the deeper feature maps and the deep-to-shallow one is to force shallow layers to pay more attention to contexts.

3. Preliminaries

Our approach essentially encourages layer-wise interactions within a SS model. For the convenience of comparison, we make a brief review of closely related methods (which also focus on layer-wise interactions). We divide these methods into three types according to their operations. 1) **Type 1**: Pixel-wise Addition, 2) **Type 2**: Map-wise Concatenation, and 3) **Type 3**: Pixel-wise Transformation.

Type 1 sums up feature maps element-wisely, which are from two layers. It is an intuitive feature interaction method, and has been widely deployed in the U-Shape networks, *e.g.*, UNet [45], SegNet [2], and SPGNet [7]. Please note that before the addition, one set of feature maps must be resized to match the size of the other sets as in [20, 31, 41, 42]. Popular resizing methods include bi-linear interpolation [46] (or deconvolution [41]) for up-sampling and stride convolution (or pooling) for down-sampling.

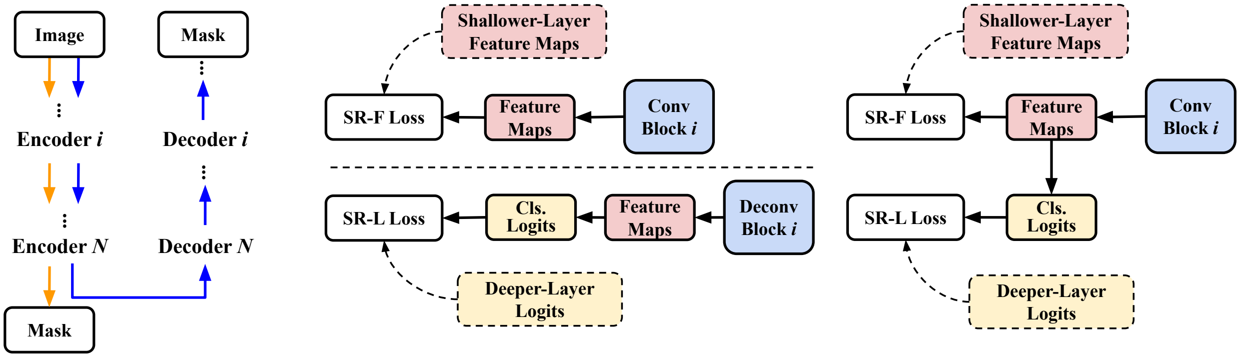
Type 2 concatenates feature maps (before which one set of feature maps are resized) along the channel dimension, *e.g.*, ASPP [5], PSP [71], HRNet [52], and Res2Net [16]. There is an additional convolution applied to the stacked feature maps for channel size adjustment. It is indeed through this convolution that the feature maps can interact with each other within local regions.

Type 3 applies the advanced non-local interaction [53] densely between every pixel from one feature map and all pixels from another feature map. This category has been validated of high effectiveness on several tasks, *i.e.*, image classification [30, 59], object detection [3, 72], and semantic segmentation [65, 73]. However, it is expensive to apply in deep neural networks.

Our difference from the above methods. We highlight that the above methods are all based on the direct operations on feature maps. In contrast, our self-regulation approach is indirect as it is embodied in the loss function. Specifically, it is realized by applying additional loss terms to the loss function. Therefore, our approach is orthogonal to them. Our experiments show that it can play as an easy and cheap plug-and-play to them, while bringing consistent and significant performance boosts.

4. Self-Regulation (SR)

Our proposed SR applies three sets (kinds) of losses for training semantic segmentation models: 1) Multi-Exit Architecture losses [23, 44] (**MEA**)—image-level multi-label classification loss and pixel-level segmentation loss for every block (layer); 2) Self-Regulation losses using Feature maps (**SR-F**)—distillation losses between the features output by Teacher and Student blocks; and 3) Self-



(a) Conv arch. & U-Shape arch. **(b) Encoder *i* (up) & Decoder *i* (down) for U-Shape arch.** **(c) Encoder *i* for Conv arch.**

Figure 3. Two example network architectures (arch) in (a) and their respective ways of applying SR in (b) and (c). "Encoder (Decoder) i" denotes the i-th Conv (Deconv) block and its related computation in the Encoder (Decoder) network of U-Shape arch, or in the Conv arch (where there is no Decoder). Both architectures have N Conv blocks, and U-Shape arch additionally contains N Deconv blocks. "Cls. Logits" denotes the classification Logits.

Regulation losses using classification Logits (**SR-L**)— distillation losses between the classification logits of Teacher and Student blocks. For the details of MEA losses, please kindly refer to the paragraph headed with **MEA** in Section 2. For SR-F and SR-L losses, we demonstrate their computation pipelines in Figure 3 based on two popular semantic segmentation backbones: a fully convolutional architecture (Conv arch) [37] and a U-Shape Encoder-Decoder architecture (U-Shape arch) [7].

In the standard U-Shape arch, each Encoder has a sequence of Conv blocks and every Decoder has the same number of Deconv blocks. In its Encoder, feature maps output by the shallowest block are taken as the "ground truth features" to teach deeper blocks by computing SR-F losses. For its Decoder (or the Encoder of Conv arch), the classification logits output by the deepest block are taken as the "ground truth logits" to teach shallower blocks by computing SR-L losses. When back-propagating these losses, the model is optimized to capture more pixel-level (taught by the shallowest block) as well as semantic-level (taught by the deepest block) information in each layer, thus derives a more powerful image representation.

4.1. SR-F Loss: Shallow to Deep

As illustrated in Figure 3, each Conv block or Deconv block can be regarded as an individual feature extractor. The ending (the shallowest or the deepest) block can be either teacher or student in different situations, as our self-regulation is bidirectional. The middle blocks are always students. We denote the transformation functions of Teacher and Student as $\mathcal{T}_\theta(\mathbf{x})$ and $\mathcal{S}_\phi(\mathbf{x})$, respectively, where θ and ϕ represent their corresponding network parameters, and $\mathbf{x}$ is the input image. For simplicity, we omit $\mathbf{x}$ in the following formulations.

SR-F loss aims to encourage deep blocks to preserve the detailed information that is retained in shallow blocks. Our method simply deploys the shallowest (the first) block as the Teacher (called Shallow Teacher) to regulate deep blocks (called Deep Students). Below, we elaborate the computation pipeline of SR-F loss. We use a cross-entropy function to measure the distance between Shallow Teacher's feature maps $\mathbf{T}_\theta^{[1]}$ to the feature maps of the i-th Deep Student (*i.e.*, the i-th block) $\mathbf{S}_\phi^{[i]}$, and obtain the SR-F loss as follows,

$$\mathcal{L}_{\text{ce}}\big(\mathbf{T}_\theta^{[1]}, \mathbf{S}_\phi^{[i]}\big) = -\frac{1}{M}\sum_{j=1}^{M}\sigma(\mathbf{t}_j)\log\left(\sigma(\mathbf{s}_j)\right), \tag{1}$$

where $i \in [2, N]$, N denotes the number of Conv blocks in a Conv arch (or a U-Shape arch); $\mathbf{t}_j$ is the vector on the j-th pixel position in $\mathbf{T}_\theta^{[1]}$, and its dimension is the same as the channel size; $\mathbf{s}_j$ is the corresponding vector in $\mathbf{S}_\phi^{[i]}$; σ denotes the softmax normalization along channels. and M is the spatial size ($=$ width $\times$ height) of the feature map.

In general KD [21], an advanced technique to improve efficiency is to leverage the temperature scaling [17]. If applying this technique in our case, we can rewrite Eq. 1 as:

$$\mathcal{L}_{\text{ce}}^{\tau}\big(\mathbf{T}_\theta^{[1]}, \mathbf{S}_\phi^{[i]}\big) = -\tau^2\frac{1}{M}\sum_{j=1}^{M}\sigma(\mathbf{t}_j)^{1/\tau}\log\left(\sigma(\mathbf{s}_j)^{1/\tau}\right), \tag{2}$$

where we note that the value of $\mathbf{t}_j^{1/\tau}$ (or $\mathbf{s}_j^{1/\tau}$) is normalized within each feature map. $\tau \in \mathbb{R}_+$ denotes the temperature and its effect is spatial smoothness, *e.g.*, a larger value of τ results a greater suppression on the difference between the maximum and minimum pixel values on the feature map.

The Shallow Teacher is "teaching" many Deep Students, so the overall SR-F loss $\mathcal{L}_{\text{SR-F}}$ can be derived as follows,

$$\mathcal{L}_{\text{SR-F}} = \sum_{i=2}^{N}\mathcal{L}_{\text{ce}}^{\tau}\big(\mathbf{T}_\theta^{[1]}, \mathbf{S}_\phi^{[i]}\big). \tag{3}$$

4.2. SR-L Loss: Deep to Shallow

SR-L aims to encourage shallow blocks (layers) to capture more global contextual information, *i.e.*, the shape of a "horse", such that their feature can be more robust against background noises. Our method simply deploys the deepest (*i.e.*, the last) layer as the Teacher (called Deep Teacher) to regulate all shallow layers (called Shallow Students). We understand that the classification logits contain the high-level semantic information of object classes, so we use those logits of Deep Teacher as "teaching materials".

Given Deep Teacher's logits $\mathbf{T}_\theta^{[N]}$ and all Shallow Students's logits $\{\mathbf{S}_\phi^{[k]}\}_{k=1}^{N-1}$, we just follow methods as in Eq. 1 to Eq. 3. *The difference is that we compute cross-entropy loss between two sets of logits rather than feature maps.* Therefore, we obtain the SR-L loss as follows,

$$\mathcal{L}_{\text{SR-L}} = \sum_{k=1}^{N-1} \mathcal{L}_{\text{ce}}^{\tau}\big(\mathbf{T}_\theta^{[N]}, \mathbf{S}_\phi^{[k]}\big). \tag{4}$$

4.3. Overall Loss Function

Below we incorporate the introduced SR losses with the layer-wise loss terms for multi-label classification as well as semantic segmentation (computed from multiple pairs of classifier and segmenter as illustrated in Figure 2), and derive the overall objective function as follows,

$$\mathcal{L}_{\text{SR}} = \underbrace{\lambda_1 \mathcal{L}_{\text{SR-cls}} + \lambda_2 \mathcal{L}_{\text{SR-seg}}}_{\text{MEA loss}} + \lambda_3 \mathcal{L}_{\text{SR-F}} + \mathcal{L}_{\text{SR-L}}, \tag{5}$$

where $\mathcal{L}_{\text{SR-seg}}$ and $\mathcal{L}_{\text{SR-cls}}$ are cross-entropy losses using ground truth one-hot labels. *This pair of loss terms are called MEA loss in the following.* On the top of them, $\mathcal{L}_{\text{SR-F}}$ and $\mathcal{L}_{\text{SR-L}}$ bring further performance improvement to the model because they encourage the layer in the model to learn from the "soft knowledge" acquired by another layer which is beyond the knowledge of one-hot labels. λ_1, λ_2, and λ_3 denote the weights used to balance these losses.

We highlight that our $\mathcal{L}_{\text{SR}}$ is a general loss function, which is easy to incorporate into any FCN-based SS model. In the following, we introduce how to implement $\mathcal{L}_{\text{SR}}$ on two representative as well as state-of-the-art SS backbones, *i.e.*, a U-Shape arch and a Conv arch.

4.4. SR on Specific SS Backbones

In Figure 3 (a), we demonstrate an U-Shape arch on the Blue path and a Conv arch on the Orange path. To better elaborate the implementation of SR on these two types of backbones, we take the state-of-the-art SPGNet [7] and HRNet [52] as examples, respectively.

SPGNet. It is a two-stage U-Shape arch (double-sized compared to the U-Shape arch in Figure 3 (a)), each stage is a standard U-Shape arch consisted of 5 Conv and 5 Deconv residual blocks [7]. In Figure 3 (b), we illustrate our method of applying SR-F and SF-L losses to an example pair of Conv and Deconv blocks.

In the i-th Conv block ($1 < i \le N$), a 3×3 convolution is first deployed to reduce the channel size to 256. Another 3×3 convolutional layer followed by a batch normalization layer is then used to generate the feature maps $\mathbf{S}_\phi^{[i]}$ which have the same channel size with the feature maps $\mathbf{T}_\theta^{[1]}$ of Shallow Teacher. Next, we compute the i-th SR-F loss $\mathcal{L}_{\text{SR-F}}^{[i]}$ following Eq. 2. Finally, we can derive the overall SR-F loss, *i.e.*, $\mathcal{L}_{\text{SR-F}}$ by summing up the SR-F losses of all Deep Students $\big\{\mathbf{S}_\phi^{[i]}\big\}_{i=2}^{N}$ as in Eq. 3.

In the i-th Deconv block ($1 \le i < N$), we follow the same method to generate feature maps, *i.e.*, through two convolutional layers and one batch normalization layer. Then, we feed these maps into a pair of classifier and segmenter. We use the resulted predictions to respectively compare to image-level and pixel-level one-hot labels, yielding two standard cross-entropy losses, *i.e.*, $\mathcal{L}_{\text{SR-cls}}^{[i]}$ and $\mathcal{L}_{\text{SR-seg}}^{[i]}$. Computing these losses through all Deconv blocks, we can get the overall MEA loss (consisting of $\mathcal{L}_{\text{SR-cls}}$ and $\mathcal{L}_{\text{SR-seg}}$).

Besides, we denote the classification logits of Deep Teacher as $\mathbf{T}_\theta^{[N]}$, and use them to regulate each Shallow Student $\mathbf{S}_\phi^{[i]}$, *i.e.*, to compute the SF-L loss $\mathcal{L}_{\text{SR-F}}^{[i]}$. When getting all SF-L losses (through all Shallow Students) $\big\{\mathbf{S}_\phi^{[i]}\big\}_{i=1}^{N-1}$, we can derive the overall SR-L loss $\mathcal{L}_{\text{SR-L}}$ using Eq. 4.

HRNet. It is a Conv arch consisting of 4 Conv blocks, *i.e.*, N=4 [52]. Its block 1 produces the highest resolution feature maps. These feature maps are fed into two paths: path 1 goes to block 2 to continue producing high-resolution feature maps; and path 2 is down-sampling first via a stride 3×3 convolution and then goes to block 2 to produce low-resolution feature maps. Repeating these two paths on block 3 and block 4 yields four outputs with different resolutions.

We take block i as an example and illustrate the computation flow in Figure 3 (c). First, we apply a 3×3 convolution and a batch normalization layer to reduce the channel size of feature maps to be 256. Then, we resize those maps to be $1/8$ spatial size of the input image, and apply another 3×3 convolution to produce the final feature maps (named as "Feature Maps" in Figure 3 (c)) of block i. We take the highest-resolution feature (after block 1) as Shallow Teacher $\mathbf{T}_\theta^{[1]}$. The SR-F loss is thus the summation of its distances to all Deep Students $\big\{\mathbf{S}_\phi^{[j]}\big\}_{k=2}^{N}$ as in Eq. 3. We take the classification logits of block 4 as Deep Teacher $\mathbf{T}_\theta^{[N]}$. The SR-L loss is thus the summation of its distances to all Shallow Students $\big\{\mathbf{S}_\phi^{[j]}\big\}_{k=1}^{N-1}$ as in Eq. 4. The MEA loss consisting of $\mathcal{L}_{\text{SR-seg}}$ and $\mathcal{L}_{\text{SR-cls}}$ can be derived in the same way as in the U-Shape arch. When using these losses as in Eq. 5, we apply different combination weights and will show the details in the experiment section.

5. Experiments

Our SR was evaluated on two SS tasks: *i.e.*, weakly-supervised SS (WSSS) with image-level class labels, and fully-supervised SS (FSSS) with pixel-level class labels.

Common Settings. All experiments were implemented on PyTorch [43]. All models were pre-trained on ImageNet [9], and then fine-tuned on the *training* sets of SS as in [5, 62]. For evaluation[2], the mean Intersection of Union (mIoU) was used as the primary evaluation metric. In addition, the number of model parameters and FLOPs were used as efficiency metrics. Following [66, 67], the combination weights of classification loss and segmentation loss were respectively set to 0.2 and 0.8 in MEA. The weights of SR-L and SR-F were set equal, *i.e.*, $\lambda_1 = 0.2, \lambda_2 = 0.8, \lambda_3 = 1$. The results of using other weights are given in the supplementary materials. For the hyperparameter τ (temperature), we applied an adaptive annealing scheme, *i.e.*, initializing τ as 1 and multiplying it by a factor of 1.05 every time when the difference between the minimum and maximum values (on any feature map in the minibatch) exceeds 0.5.

5.1. WSSS Settings

Datasets. Our WSSS experiments were carried out on two widely-used benchmarks: PASCAL VOC 2012 (PC) [11] and MS-COCO 2014 (MC) [32]. PC dataset consists of 21 classes (20 for objects and 1 for background) and splits $1,464$ images for *training*, $1,449$ for *val*, and $1,456$ for *test*. Following related works [1, 24, 27, 64], we used an enlarged *training* set including $10,582$ images. MC dataset consists of 81 classes (80 for objects and 1 for background) with $80k$ and $40k$ images respectively for *training* and *val*. In the training phase of WSSS, **only the image-level class labels were used as ground truth**. For data augmentation, we followed the same strategy as in [66], including gaussian blur, color augmentation, random horizontal flip, randomly rotate (from $-10°$ to $+10°$), and random scaling (using the scaling rates between $0.5\times$ and $2\times$).

Baseline Models. The WSSS framework includes two steps: pseudo-mask generation and semantic segmentation training with pseudo-masks. For pseudo-mask generation, we deployed two popular ones, namely IRNet [1] and SEAM [54], and the state-of-the-art (SOTA) CONTA [64]. For semantic segmentation, we deployed DeepLab-v2 [5] with ResNet-38 [55], SegNet [2] with ResNet-101 [20] and the SOTA SPGNet [7] with ResNet-50 [20], respectively. Baseline models are trained on the conventional arch and using only the cross-entropy loss. Baseline+SR models are ours which apply our proposed SR loss terms (see Eq. 5) and train the model on the MEA arch (see Figure 2).

Training Details. Our major settings followed close related works [1, 7, 54, 64]. The input images were cropped into fixed sizes as 512×512 and 448×448 for PC and MC, respectively. Zero padding was used if needed. The mini-batch SGD momentum optimizer was used to train all SS models with batch size as 16, momentum as 0.9 and weight decay as 0.0001. The initial learning rates (LR) were 0.01 and 0.04 for PC and MC datasets, respectively. A "poly" schedule for LR was deployed, *i.e.*, updating LR as $(1 - \frac{iter}{iter_{max}})^{0.9}$. All our implemented models were trained for 80 epochs on PC and 50 epochs on MC.

5.2. FSSS Settings

Datasets. Our FSSS experiments were carried out on two challenging benchmarks: Cityscapes (CS) [8], and PASCAL-Context (PAC) [39]. CS dataset consists of 19 classes with $2,975$ images for *training*, 500 for *val* and $1,525$ for *test*. Please note that we only used these finely annotated images, although this dataset offers $20,000$ coarsely annotated images. For the PAC dataset, we followed [66, 62, 19] and used the most popular version consisted of 60 classes (including the background class). There are $4,998$ and $5,015$ images for *training* and *test*, respectively. For data augmentation, we followed [62, 67, 65] to use random horizontal flipping, random scaling (using the the scaling rates between $0.5\times$ and $2\times$), and random brightness jittering (in the range from $-10°$ to $+10°$).

Baseline Models. We implemented our SR onto a popular method DBES [29] with ResNet-101 [20], and the SOTA method OCRNet [62] with HRNetV2-W48 [52]. Please note that HRNetV2-W48 is a backbone network.

Training Details. Our major settings were the same as those in baseline methods [52, 62]. The input images were cropped into 969×969 and 520×520 on CS and PAC datasets, respectively. The mini-batch SGD momentum optimizer was used in the training phase with batch size as 8 and momentum as 0.9. The initial LRs were set to 0.01 and 0.001 for CS and PAC, respectively. The same "poly" schedule was deployed. The L2 regularization term weights were set to 0.0005 and 0.0001 for CS and PAC, respectively. All our implemented models were trained from 580 epochs on CS and 100 epochs on PAC.

5.3. Results and Analyses

Ablation Study. We conducted the ablation study on the val sets of PC, MC and CS datasets, and reported the results in Table 1. "MEA" denotes using the MEA loss consisting of $\mathcal{L}_{\text{SR-seg}}$ and $\mathcal{L}_{\text{SR-cls}}$. We used the SOTA WSSS method, namely CONTA [64]+SPGNet [7], as the baseline here and show its results in row 1. Comparing row 5 to row 1, we can see the proposed SR loss brought clear performance gains, *e.g.*, 1.4% on PC dataset. Intriguingly, comparing row 5 to row 2 (and row 1), we find that distillation-based loss terms ($\mathcal{L}_{\text{SR-L}}$ and $\mathcal{L}_{\text{SR-F}}$) brought higher improvement mar-

[2] Please kindly refer to more details about hyperparameters and evaluation results in supplementary materials.

MEA	SR-F	SR-L	PC	MC	CS
✗	✗	✗	67.1	33.6	80.7$^\flat$
✓	✗	✗	67.7$_{+0.6}$	34.0$_{+0.4}$	81.1$_{+0.4}$
✓	✗	✓	68.1$_{+1.0}$	34.3$_{+0.7}$	81.6$_{+0.9}$
✓	✓	✗	68.2$_{+1.1}$	34.3$_{+0.7}$	81.7$_{+1.0}$
✓	✓	✓	68.5$_{+1.4}$	34.5$_{+0.9}$	82.1$_{+1.4}$

Table 1. Ablation study results (mIoU, %) on the *val* sets of three datasets: PASCAL VOC 2012 (PC) [11], MS-COCO 2014 (MC) [32] with image-level class labels (WSSS), and Cityscapes (CS) with pixel-level class labels (FSSS). The WSSS baseline is CONTA [64]+SPGNet [7], and the FSSS baseline is OCRNet [62]. Their results are shown in the first row. "MEA" indicates the MEA loss. "SR-F" and "SR-L" indicate the other two losses respectively (see Eq. 5). $\flat$ means the result is produced by us using public code.

	Baseline	+MEA	+(MEA,SR-L)	+(MEA,SR-F)	+SR
WSSS; SPGNet [7] as Baseline					
Params.	55.6**M**	56.7$_{+1.1}$**M**	56.7$_{+0.0}$**M**	56.9$_{+0.2}$**M**	56.9**M**
FLOPs	467.6**B**	467.9$_{+0.3}$**B**	467.9$_{+0.0}$**B**	467.9$_{+0.0}$**B**	467.9**B**
FSSS; OCRNet [62] as Baseline					
Params.	76.4**M**	78.9$_{+2.5}$**M**	78.9$_{+0.0}$**M**	79.5$_{+0.6}$**M**	79.5**M**
FLOPs	1,087.3**G**	1,095.5$_{+8.2}$**G**	1,095.5$_{+0.0}$**G**	1,095.5$_{+0.0}$**G**	1,095.5**G**

Table 2. Model efficiency analysis for different components of SR. "+SR" means including all terms (MEA, SR-L, and SR-F).

gins than using only MEA loss. As we mentioned under Eq. 5, this is because our SR-L and SR-F terms encourage each individual layer to learn the "soft knowledge" (from a superior layer) which is richer than the "hard knowledge" in one-hot labels (used for computing the MEA loss). This phenomenon is consistent across all datasets.

Model Efficiency. Our SR brings performance improvements without increasing much computational costs. To validate this, we show the statistics of Params, *i.e.*, the number of network parameters, and FLOPs, *i.e.*, the speed of training, in Table 2. It is clearly shown that SR introduced very little computational overheads for both SS tasks. For example in WSSS, applying MEA, SR-L and SR-F losses increased 1.1**M**, 0.0**M**, and 0.2**M** Params, respectively. Using MEA increased only 0.3**B** FLOPs, and using SR-L and SR-F for 0. This overhead is mainly caused by using additional convolutional layers in constructing MEA.

Comparing to SOTA in WSSS. To compare with SOTA methods, we implemented three pseudo-mask generation approaches, *i.e.*, IRNet [1], SEAM [54], and CONTA [64], and used three baseline SS models, *i.e.*, DeepLab-v2 [5], SegNet [2], and SPGNet [7]. We plugged our SR in all of them. We report the results on the *val* and *test* sets of the PC dataset, and the *val* set of the MC dataset in Table 3. It is shown that using our SR consistently improved the performance of all implemented methods and achieved the top performance on two datasets. For example, it boosted DeepLab-v2 on the *val* set of PC by 1.1%, 1.1%, and 0.7% mIoU improvements when using IRNet, SEAM and CONTA for generating pseudo masks, respectively. It also boosted SegNet and SPGNet on PC respectively by 1.0% and 1.4% when using CONTA. On the *test* set of PC, it achieved the best performance which is higher than the SOTA method (CONTA w/ SPGNet) by 1.2% mIoU. Its superiority is also obvious when comparing its best results to those of the methods in the top block, *e.g.*, it surpassed EME by 1.3% on PC *val* set and MCS by 2.3% on PC *test* set.

Methods	Backbone	PC *val*	PC *test*	MC *val*
SCE [4]	ResNet-101	66.1	65.9	–
EME [12]	ResNet-101	67.2	66.7	–
MCS [47]	ResNet-101	66.2	66.8	–
IRNet [1]	Deeplab-v2 [5]	63.5	64.8	32.6
IRNet+SR	Deeplab-v2 [5]	64.6$_{+1.1}$	65.8$_{+1.0}$	33.4$_{+0.8}$
SEAM [54]	Deeplab-v2 [5]	64.5	65.7	31.9
SEAM+SR	Deeplab-v2 [5]	65.6$_{+1.1}$	66.5$_{+0.8}$	32.6$_{+0.7}$
CONTA [64]	Deeplab-v2 [5]	66.1	66.7	33.4
CONTA+SR	Deeplab-v2 [5]	66.8$_{+0.7}$	67.2$_{+0.5}$	34.0$_{+0.6}$
CONTA [64]	SegNet [2]	66.9	67.7	33.7
CONTA+SR	SegNet [2]	67.9$_{+1.0}$	68.4$_{+0.7}$	34.4$_{+0.7}$
CONTA [64]	SPGNet [7]	67.1	67.9	33.6
CONTA+SR	SPGNet [7]	68.5$_{+1.4}$	69.1$_{+1.2}$	34.5$_{+0.9}$

Table 3. Comparing to the state-of-the-arts on the *val* and *test* sets of PASCAL VOC 2012 (PC) [11], and the *val* set of MS-COCO (MC) 2014 [32] using image-level class labels. "–" denotes there are no reported results in the original papers. "+SR" means applying our SR loss (Eq. 5) function to train the models.

Methods	Backbone	*val* (%)	*test* (%)
DANet [14]	ResNet-101	81.5	81.5
CDGCNet [22]	ResNet-101	81.9	–
HRNet [52]	HRNetV2-W48	81.1	81.6
DBES [29]	ResNet-101	81.3$^\flat$	81.5$^\flat$
DBES+SR	ResNet-101	82.0$_{+0.7}$	82.1$_{+0.6}$
OCRNet [62]	HRNetV2-W48	80.7$^\flat$	81.8$^\flat$
OCRNet+SR	HRNetV2-W48	82.1$_{+1.4}$	82.7$_{+0.9}$

Table 4. Result comparison (mIoU) with the state-of-the-arts on Cityscapes [8] using pixel-level labels. "–" denotes there are no reported results in the original papers. $\flat$ means that this is our re-implemented result. "+SR" means applying SR loss (Eq. 5) function to train the models.

Comparing to SOTA in FSSS. In the task of FSSS, SOTA methods include DBES [29] on ResNet-101 [20] (backbone) and OCRNet [62] on HRNet [52] (backbone), on both CS and PAC datasets. We present their original results and also show our results (by plugging SR in these methods) in Table 4 and Table 5 respectively for two datasets. We can see from both tables that our SR becomes the new state-of-the-art. Impressively, our SR with little computing overhead boosted the large-scale network OCRNet by a clear margin of 1.4% mIoU on the *val* set and 0.9% mIoU on the *test* set of CS, and 0.8% on the more challenging PAC.

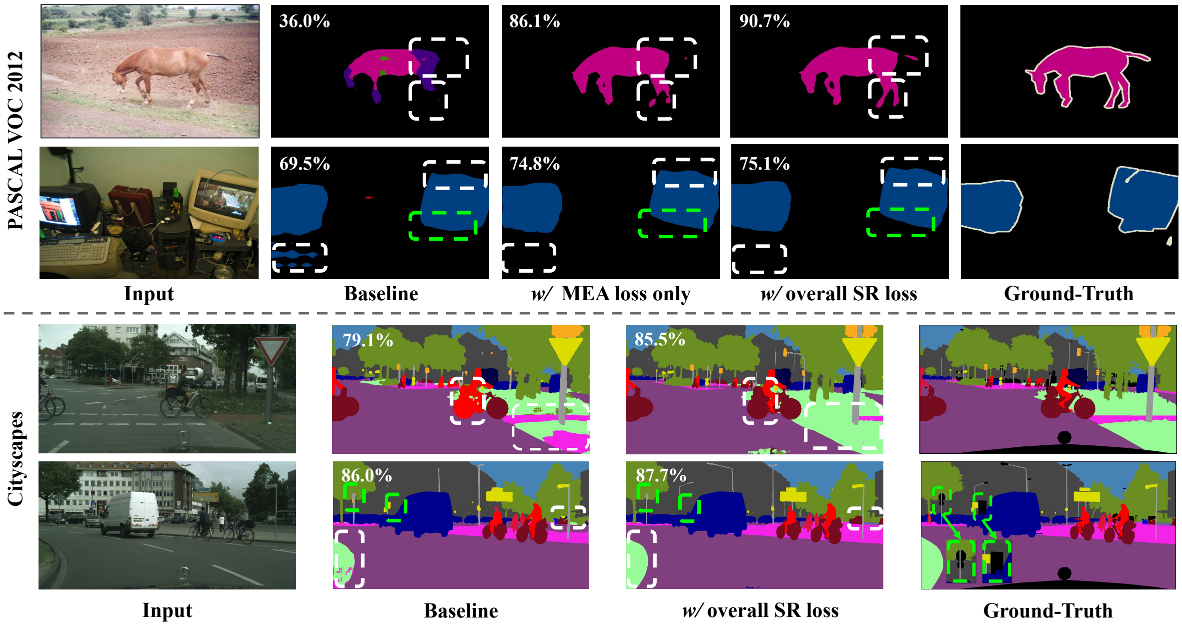


Figure 4. Visualization results for WSSS on the *val* set of PASCAL VOC 2012 (PC) [11] using CONTA [64]+SPGNet [7] as Baseline, and FSSS on the *val* set of Cityscapes (CS) [8] using OCRNet [62] as Baseline. "*w/* overall SR loss" means applying MEA loss, SR-F loss, and SR-L loss on the Baseline. The mIoU is shown on each result image. The white dashed boxes highlight the revised regions by our methods. The green dashed boxes highlight the failure cases of both baseline and our models.

Methods	Backbone	*test* (%)
CFNet [67]	ResNet-101	54.0
ACNet [15]	ResNet-101	54.1
APCNet [19]	ResNet-101	54.7
DBES [29]	ResNet-101	54.3
DBES+SR [29]	ResNet-101	55.3$_{+1.0}$
OCRNet [62]	HRNetV2-W48	54.9$^{\flat}$
OCRNet+SR	HRNetV2-W48	55.7$_{+0.8}$

Table 5. Result comparison (mIoU, %) with the state-of-the-arts on PASCAL-Context [39] using pixel-level labels.

Visualizations. In Figure 4, we visualize four segmentation samples on PC (val) and CS (val) datasets. The top two show the results of WSSS models, and the bottom two for FSSS models. From PC samples, we can see that many of the failure regions (highlighted with white dash boxes) using baseline models were corrected by adding MEA losses. This is a gain of strengthening the semantics and details on every individual layer of the model. On the top of it, using the overall SR loss makes the models more effective to mark out minor object parts, *e.g.*, "horse legs" and "horse tail". On both datasets, we also saw some failure cases. For example when segmenting "monitors", all models are missing "neck" regions (see green dashed boxes). We think the reason is that the pseudo mask module (which is basically a classification model) in WSSS rarely attends to "monitor neck" when training the classifier of "monitor". Another failure case is on the second row of CS: tiny objects such as distant traffic signs are missing. We believe those signs are difficult for human eyes, not to mention for machine models trained on 969 × 969 images.

6. Conclusion

We started by seeking reasons for two major failure cases in SS. We found that it is either the overuse or underuse of the semantics and visual details across different layers. To this end, we proposed three "shallow to deep and back" regulation operations: MEA — regulating the predictions of each pair with ground-truth labels; SR-F — regulating deeper-layer segmenters by the shallowest layer segmenter; and SR-L — regulating shallow-layer classifiers by the deepest layer classifier. Experimental results on both WSSS and FSSS demonstrated that our SR loss can bring continuous performance gains with little computational overhead. In the future, we will further investigate some new regulation methods on SS, as well as how to apply existing regulations to solve similar problems in other tasks, such as video event classification and action recognition.

Acknowledgements

This work was partly supported by the National Key Research and Development Program of China under Grant 2018AAA0102002, the National Natural Science Foundation of China under Grant 61732007, A*STAR under its AME YIRG Grant (Project No. A20E6c0101), and the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 2 Grant.

References

[1] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *CVPR*, 2019.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *TPAMI*, 2017.

[3] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV*, 2019.

[4] Yu-Ting Chang, Qiaosong Wang, Wei-Chih Hung, Robinson Piramuthu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Weakly-supervised semantic segmentation via sub-category exploration. In *CVPR*, 2020.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *TPAMI*, 2017.

[6] Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. In *ECCV*, 2018.

[7] Bowen Cheng, Liang-Chieh Chen, Yunchao Wei, Yukun Zhu, Zilong Huang, Jinjun Xiong, Thomas S Huang, Wen-Mei Hwu, and Honghui Shi. Spgnet: Semantic prediction guidance for scene parsing. In *ICCV*, 2019.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] Rahul Duggal, Scott Freitas, Sunny Dhamnani, Duen Horng, Jimeng Sun, et al. Elf: An early-exiting framework for long-tailed classification. In *arXiv*, 2020.

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

[12] Junsong Fan, Zhaoxiang Zhang, and Tieniu Tan. Employing multi-estimations for weakly-supervised semantic segmentation. In *ECCV*, 2020.

[13] Xing Fan, Wei Jiang, Hao Luo, and Mengjuan Fei. Spherereid: Deep hypersphere manifold embedding for person reidentification. In *Journal of Visual Communication and Image Representation*, 2019.

[14] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019.

[15] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for scene parsing. In *ICCV*, 2019.

[16] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. In *TPAMI*, 2019.

[17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[18] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. In *Medical Image Analysis*, 2017.

[19] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang, and Yu Qiao. Adaptive pyramid context network for semantic segmentation. In *CVPR*, 2019.

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

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2014.

[22] Hanzhe Hu, Deyi Ji, Weihao Gan, Shuai Bai, Wei Wu, and Junjie Yan. Class-wise dynamic graph convolution for semantic segmentation. In *ECCV*, 2020.

[23] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018.

[24] Zilong Huang, Xinggang Wang, Jiasi Wang, Wenyu Liu, and Jingdong Wang. Weakly-supervised semantic segmentation network with deep seeded region growing. In *CVPR*, 2018.

[25] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network. In *CVPR*, 2018.

[26] Lei Jiang, Wengang Zhou, and Houqiang Li. Knowledge distillation with category-aware attention and discriminant logit losses. In *ICME*, 2019.

[27] Alexander Kolesnikov and Christoph H Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *ECCV*, 2016.

[28] Quanquan Li, Shengying Jin, and Junjie Yan. Mimicking very efficient network for object detection. In *CVPR*, 2017.

[29] Xiangtai Li, Xia Li, Li Zhang, Guangliang Cheng, Jianping Shi, Zhouchen Lin, Shaohua Tan, and Yunhai Tong. Improving semantic segmentation via decoupled body and edge supervision. In *ECCV*, 2020.

[30] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L Yuille. Neural architecture search for lightweight non-local networks. In *CVPR*, 2020.

[31] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[33] Jianbo Liu, Junjun He, Jiawei Zhang, Jimmy S Ren, and Hongsheng Li. Efficientfcn: Holistically-guided decoding for semantic segmentation. In *ECCV*, 2020.

[34] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. In *IJCV*, 2020.

[35] Yifan Liu, Changyong Shu, Jingdong Wang, and Chunhua Shen. Structured knowledge distillation for dense prediction. In *TPAMI*, 2020.

[36] Zhenguang Liu, Haoming Chen, Runyang Feng, Shuang Wu, Shouling Ji, Bailin Yang, and Xun Wang. Deep dual consecutive network for human pose estimation. In *CVPR*, 2021.

[37] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[38] Yunteng Luan, Hanyu Zhao, Zhi Yang, and Yafei Dai. Msd: Multi-self-distillation learning via multi-classifiers within deep neural networks. In *arXiv*, 2019.

[39] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[40] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[41] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[42] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas. Heterogeneous knowledge distillation using information flow modeling. In *CVPR*, 2020.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[44] Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *ICCV*, 2019.

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[46] PR Smith. Bilinear interpolation of digital images. *Ultramicroscopy*, 6(2):201–204, 1981.

[47] Guolei Sun, Wenguan Wang, Jifeng Dai, and Luc Van Gool. Mining cross-image semantics for weakly supervised semantic segmentation. In *ECCV*, 2020.

[48] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, 2016.

[49] Michael Treml, José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, et al. Speeding up semantic segmentation for autonomous driving. In *NeurIPS*, 2016.

[50] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *ICCV*, 2019.

[51] Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? In *ICLR*, 2016.

[52] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. In *TPAMI*, 2020.

[53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[54] Yude Wang, Jie Zhang, Meina Kan, Shiguang Shan, and Xilin Chen. Self-supervised equivariant attention mechanism for weakly supervised semantic segmentation. In *CVPR*, 2020.

[55] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. In *PR*, 2019.

[56] Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision. In *ECCV*, 2020.

[57] Kunran Xu, Lai Rui, Yishi Li, and Lin Gu. Feature normalized knowledge distillation for image classification. In *ECCV*, 2020.

[58] Ting-Bing Xu and Cheng-Lin Liu. Data-distortion guided self-distillation for deep neural networks. In *AAAI*, 2019.

[59] Yang Xu, Zebin Wu, Jocelyn Chanussot, and Zhihui Wei. Nonlocal patch tensor sparse representation for hyperspectral image super-resolution. In *TIP*, 2019.

[60] Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, and Andrew Willis. Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In *ECCV*, 2020.

[61] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[62] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020.

[63] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *ICLR*, 2019.

[64] Dong Zhang, Hanwang Zhang, Jinhui Tang, Xiansheng Hua, and Qianru Sun. Causal intervention for weakly-supervised semantic segmentation. In *NeurIPS*, 2020.

[65] Dong Zhang, Hanwang Zhang, Jinhui Tang, Meng Wang, Xiansheng Hua, and Qianru Sun. Feature pyramid transformer. In *ECCV*, 2020.

[66] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.

[67] Hang Zhang, Han Zhang, Chenguang Wang, and Junyuan Xie. Co-occurrent features in semantic segmentation. In *CVPR*, 2019.

[68] Li Zhang, Xiangtai Li, Anurag Arnab, Kuiyuan Yang, Yunhai Tong, and Philip HS Torr. Dual graph convolutional network for semantic segmentation. In *BMVC*, 2019.

[69] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*, 2019.

[70] Zhenli Zhang, Xiangyu Zhang, Chao Peng, Xiangyang Xue, and Jian Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *ECCV*, 2018.

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

[72] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.

[73] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *ICCV*, 2019.