Building Detail-Sensitive Semantic Segmentation Networks with Polynomial Pooling

Zhen Wei*[2,3], Jingyi Zhang*[1,3], Li Liu[3], Fan Zhu[3], Fumin Shen†[1], Yi Zhou[3], Si Liu[4]
Yao Sun[2], Ling Shao[3]
1. Center for Future Media and School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu, China
2. Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
3. Inception Institute of Artificial Intelligence, Abu Dhabi, UAE
4. Beijing Key Laboratory of Digital Media,
School of Computer Science and Engineering, Beihang University, Beijing, China
{weizhen, sunyao}@iie.ac.cn, {jingyi.zhang1995, liuli1213, fanzhu1987, fumin.shen}@gmail.com,
yi.zhou@inceptioniai.org, liusi@buaa.edu.cn, ling.shao@ieee.org

Abstract

Semantic segmentation is an important computer vision task, which aims to allocate a semantic label to each pixel in an image. When training a segmentation model, it is common to fine-tune a classification network pre-trained on a large-scale dataset. However, as an intrinsic property of the classification model, invariance to spatial perturbation resulting from the lose of detail-sensitivity prevents segmentation networks from achieving high performance. The use of standard poolings is one of the key factors for this invariance. The most common standard poolings are max and average pooling. Max pooling can increase both the invariance to spatial perturbations and the non-linearity of the networks. Average pooling, on the other hand, is sensitive to spatial perturbations, but is a linear function. For semantic segmentation, we prefer both the preservation of detailed cues within a local feature region and non-linearity that increases a network's functional complexity. In this work, we propose a polynomial pooling ($\mathcal{P}$-pooling) function that finds an intermediate form between max and average pooling to provide an optimally balanced and self-adjusted pooling strategy for semantic segmentation. The $\mathcal{P}$-pooling is differentiable and can be applied into a variety of pre-trained networks. Extensive studies on the PASCAL VOC, Cityscapes and ADE20k datasets demonstrate the superiority of $\mathcal{P}$-pooling over other poolings. Experiments on various network architectures and state-of-the-art training strategies also show that models with $\mathcal{P}$-pooling layers consistently outperform those directly fine-tuned using pre-trained classification models.

1. Introduction

Semantic segmentation is a critically important and highly challenging computer vision task, which has a wide range of practical industrial use cases, such as medical imaging and autonomous driving. Benefiting from advancements in deep learning, deep segmentation networks have enabled dramatic improvement in performance for this task. The essence of semantic segmentation is to perform pixel-wise classification for an input image. For this purpose a deep segmentation network needs to be detail-sensitive. Additionally, for a segmentation network to be both a robust feature extractor and classifier, it must be highly non-linear and complex.

Compared to image-level labels, pixel-wise annotations require much more human effort and, as such, the amount of pixel-wise annotated data is limited. Thus, to obtain improved performance, it is common to fine-tune classification networks that have been pre-trained on large-scale datasets. However, classification and segmentation networks are designed with contradicting purposes, where the former requires the pre-trained models to be invariant to small spatial perturbations on input images (e.g., small translations, scalings or rotations), and the latter requires the network to be aware of detailed variance within local regions. Previous approaches tackled this contradiction mainly with new network components [3, 37, 22, 1, 27], modified convolutions [33, 30, 4, 9], different network architectures [23, 24, 21, 11, 25, 36] or learned optimal receptive fields [32, 35, 9, 6, 34].

However, improving pre-trained classification models by

*the authors contribute equally
†corresponding author

effectively modifying the pooling layers of segmentation networks has been largely overlooked in previous works. Among existing pooling layers, max pooling and average pooling are the two most commonly applied mechanisms in convolutional neural networks (CNN). The max pooling contributes significantly to the spatial perturbation invariance and non-linearity of a classification network, while the average pooling is linear and sensitive to spatial perturbations. For semantic segmentation, we favor a pooling mechanism that is: (a) sensitive to trivial spatial perturbations and (b) highly non-linear so as to extend the network's capacity. To this end, we propose a *polynomial pooling* ($\mathcal{P}$*-pooling*) mechanism that has **(i)** an optimal balance between non-linearity and detail-sensitivity, **(ii)** a differentiable form that allows end-to-end training, **(iii)** sufficient flexibility and the potential to be dynamically adjusted for various data, and **(iv)** compatibility with any pre-trained classification model. We highlight our three main contributions as follows:

- To our best knowledge, this paper is the first work to propose a novel pooling function for *semantic segmentation*. The proposed $\mathcal{P}$*-pooling* enhances the detail-sensitivity of a segmentation network while maintaining its high non-linearity.
- $\mathcal{P}$*-pooling* is a learnable function with a dynamically adjustable mechanism that offers a tradeoff between the detail-sensitivity and the non-linearity with arbitrary degrees. $\mathcal{P}$*-pooling* is also differentiable to allow end-to-end training.
- $\mathcal{P}$*-pooling* method consistently improves standard fine-tuned segmentation models with various network architectures, datasets and different training strategies.

2. Related Works

Improving pooling in CNN models has been long studied. In the literature on parsing, [16, 12] proposed to use dynamic kernel sizes or strides for input features with various sizes in order to obtain results at a fixed length. [9] proposed a deformable sampling scheme to select input features in a data-driven way. Although [32, 35] were originally proposed to adaptively regularize receptive fields in convolutional layers, they can also be directly transferred to determine pooling kernel sizes.

Other works that focused on pooling kernel functions mainly performed their studies in the context of the image classification task. Among them, [13] used L_p norm (or Minkowski norm) to extend max pooling. The authors manually selected intermediate pooling functions between max and average pooling to better fit the distribution of input data. [20] generalized pooling methods by using a learned linear combination of max and average pooling. Detail-Preserving Pooling (or DPP) [26] borrows ideas from Detail-Preserving Image Downscaling (DPID) method in [31] which was originally proposed for the graphics task. DPP learns weighted summations of pixels over different pooling regions, where more importance is given to salient pixels in order to achieve higher visual satisfaction on pooled results. Offering an alternative to pooling methods, [29] replaced all max pooling layers with strided convolutions and activation functions in a small classification model that was trained from scratch and achieved better performance. Recent classification networks, such as [17], have also used strided convolution layers for feature down-sampling.

However, the related works in question are either limited to improving the usage of pooling layers rather than the kernel functions, or focus primarily on classification models that are trained from scratch rather than during the training process of segmentation networks. More discussions on the aforementioned methods will be provided in Section 3.6.

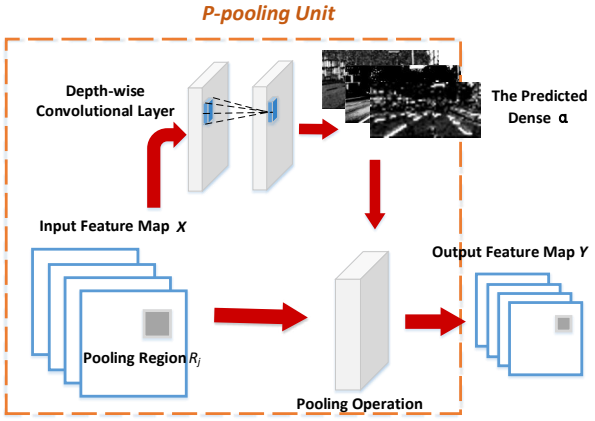

Figure 1. The structure of a $\mathcal{P}$*-pooling* unit. A side-branch net takes the features to be pooled as input and generates α values that are used in $\mathcal{P}$*-pooling*.

3. Polynomial Pooling ($\mathcal{P}$*-pooling*)

3.1. Notations

For convenience and better understanding, notations that will be used throughout the paper are first defined. For a pooling layer, the input feature map is denoted as X and the output feature map as Y. The j-th element y_j on the output feature map Y is the result of the j-th pooling region R_j on X. $R_j = \{x_1, x_2, ...x_N\}$ contains a set of N input elements on X within the j-th pooling region. For the gradients, assume that the $(l)_{th}$ layer's backward gradient $\frac{\partial E}{\partial y_i}$ is δ_i^l. Given a specific pooling function f, the general pooling is: $Y = f(X)$.

3.2. Definition of $\mathcal{P}$*-Pooling*

In this paper, a new polynomial form of pooling function (denote as $\mathcal{P}$*-pooling*) is proposed as

$$y_j = f_p(R_j) = \frac{\sum_{x_i \in R_j} x_i^{\alpha+1}}{\sum_{x_i \in R_j} x_i^{\alpha}} . \quad (1)$$

The $\mathcal{P}$-*pooling* function has a parameter $\alpha \in [0, +\infty)$, which controls its polynomial order and then functional behaviors. In this work, the specific value of α is dependent on the corresponding input data making it a data-driven parameter. Details about the learning and prediction of α are further elaborated in Section 3.5.

In practice, as most pooling functions in deep network are appended after the convolution and ReLU layers, it can usually be assumed that all elements in the input feature map are non-negative, which ensures that polynomial terms will not generate imaginary numbers. However, for the less common situations where negative input values are inevitable, we propose an approximating form of $\mathcal{P}$-*pooling*. Assume the minimum value of X is x_{min}, then

$$y_j = \hat{f}_p(R_j) = f_p(R_j - x_{min}) + x_{min} \ . \tag{2}$$

Although the approximation function differs slightly from Eq. 1, it shares almost the same properties with the original $\mathcal{P}$-*pooling* function.

3.3. Analysis of Boundary Conditions

$\mathcal{P}$-*pooling* is proposed as a superset representation of standard poolings. This is because max and average poolings are simply special cases of the $\mathcal{P}$-*pooling* function, forming the boundary conditions when $\alpha = 0$ and $\alpha \to +\infty$. By learning α values within $[0, +\infty)$, $\mathcal{P}$-*pooling* behaves as an intermediate function between max and average poolings, inheriting advantages from both to different degrees. The following propositions provide formal descriptions of $\mathcal{P}$-*pooling*'s boundary conditions.

Proposition 1: *$\mathcal{P}$-pooling is equivalent to average pooling when $\alpha = 0$.*

Proposition 2: *$\mathcal{P}$-pooling behaves as max pooling when $\alpha \to +\infty$.*

The mathematical proofs for these propositions are presented in the supplementary file. Note the proofs show that *both the forward and backward* processes of $\mathcal{P}$-*pooling* are identical to those of max and average poolings. This is significant as some other methods, including the L_p norm function (see Section 3.6), ignore the equivalence of the backward process.

3.4. Properties of Non-Linearity and Detail-Sensitivity

As an intermediate form between max and average poolings, $\mathcal{P}$-*pooling* inherits the merits of both standard poolings. Figures 2, 3 provide simple and clear demonstrations on the effectiveness of $\mathcal{P}$-*pooling*'s properties under different conditions and parameters.

The ability for detailed structure preservation in feature space is shown in Figure 2. When $\alpha = 2$, $\mathcal{P}$-*pooling* succeeds in maintaining the information of both white and gray corner points. This illustrates its strong ability to distinguish

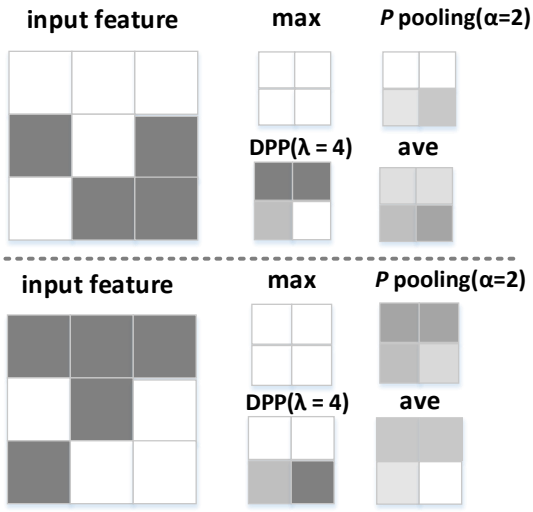

Figure 2. A demonstration on the functionalities of different pooling methods. All poolings are performed with a kernel size of 2 and stride of 1. A brighter color represents a high response value and a darker color stands for a lower response.

different details, behaving similarly to average pooling. In contrast, in Figure 3, the function surface of $\mathcal{P}$-*pooling* when $\alpha = 2$ plotted for 2D input data, has a much more significant curve degree than average pooling. Importantly, the curve of $\mathcal{P}$-*pooling*'s function surface is very close to that of max pooling, indicating these two pooling methods have similar degrees of non-linearity. These demonstrations show that the $\mathcal{P}$-*pooling* is able to unify the superior detail-sensitivity and high non-linearity in a single function.

In Section 3.6, we provide further insight on the merits and special properties of $\mathcal{P}$-*pooling* through comparisons with other recent and relevant pooling functions, as well as down-sampling methods.

3.5. Learning Adaptive Polynomial Pooling Kernels

With α being positive and all input features being non-negative, the $\mathcal{P}$-*pooling* function is differentiable and can participate in the joint end-to-end training with the whole network. Thus we have

$$\begin{aligned}\frac{\partial E}{\partial x_i} &= \sum_j \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial x_i} \\ &= \sum_j \frac{(\alpha+1)x_i^{\alpha} \sum\limits_{x_i \in R_j} x_i^{\alpha} - \alpha x_i^{\alpha-1} \sum\limits_{x_i \in R_j} x_i^{\alpha+1}}{(\sum\limits_{x_i \in R_j} x_i^{\alpha})^2} \delta_j^{l+1} \ ,\end{aligned} \tag{3}$$

$$\begin{aligned}\frac{\partial E}{\partial \alpha} &= \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \alpha} \\ &= \sum_{x_i \in R_j} \frac{ln(x_i)x_i^{\alpha+1} \sum\limits_{i \in R_j} x_i^{\alpha} - ln(x_i)x_i^{\alpha} \sum\limits_{x_i \in R_j} x_i^{\alpha+1}}{(\sum\limits_{x_i \in R_j} x_i^{\alpha})^2} \delta_j^{l+1} \ .\end{aligned} \tag{4}$$

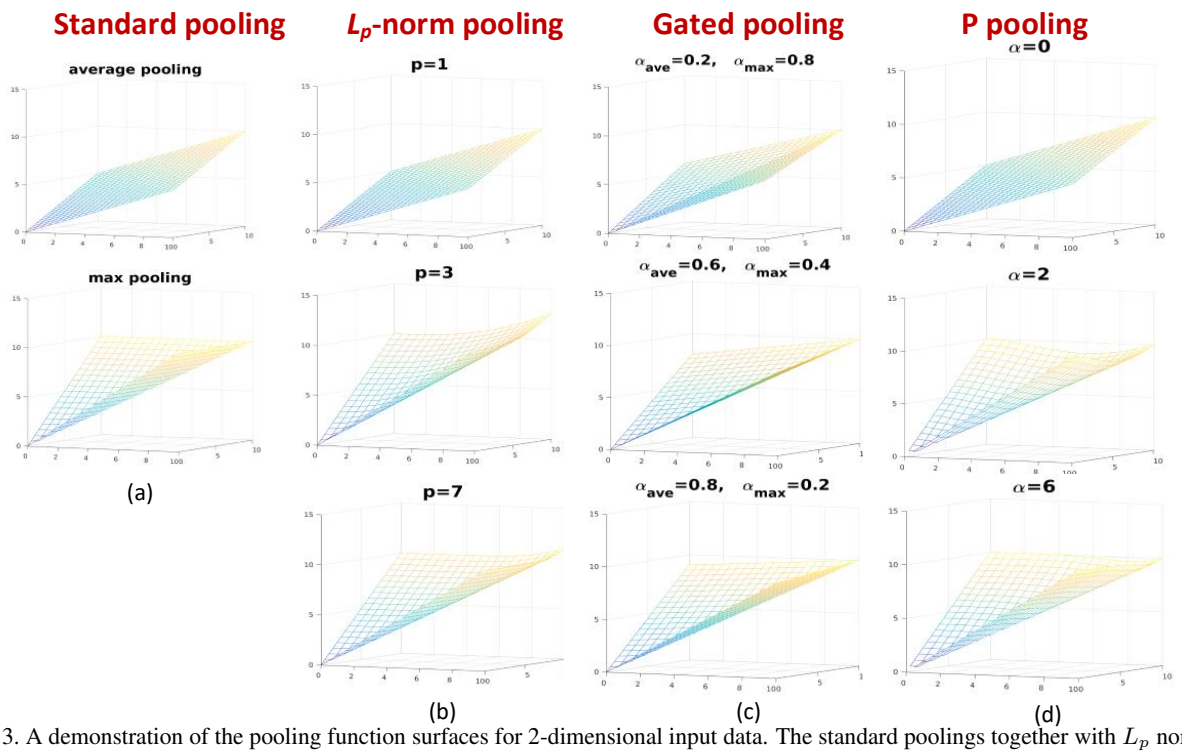

Figure 3. A demonstration of the pooling function surfaces for 2-dimensional input data. The standard poolings together with L_p norm [13], gated pooling [20] and $\mathcal{P}$-*pooling* with different parameters are displayed. The DPP [26] is unnecessary to show as it is always equal to average pooling on 2D data.

To give $\mathcal{P}$-*pooling* the highest flexibility to adjust itself according to specific input data, the values of α for each pooling window are dynamically predicted with a side-branch network. Each side prediction branch takes the feature just before pooling as its input and then outputs a dense α map that has the same size as the pooled feature maps. The j-th α value in a dense α map corresponds to the pooling region R_j and is obtained by

$$\alpha_j = Conv_{side\_branch}(R_j) \ . \tag{5}$$

Figure 1 shows the structure of a $\mathcal{P}$-*pooling* unit. This unit is able to replace pooling layers in most network backbones. Inside the unit, the side-branch is composed of two depth-wise convolutional layers with a pReLU [15] activation. To avoid numerical issues, the output values from the side-branch are clipped to $(0, C]$, where C is a predefined positive constant. The depth-wise convolution disentangles the cross-channel correlations so that α values are solely based on input data from the corresponding regions. It also introduces a very limited number of additional parameters and computational cost. In the VGG-16 model, for example, the side-branches account for less than 0.08% of the overall parameters. The side-branch is totally differentiable. Given the gradients with respect to α, the side-branches can also be trained in an end-to-end fashion.

3.6. Comparative Studies on $\mathcal{P}$-*Pooling*

In the literature for other computer vision tasks, i.e. image classification, there are several pooling and down-sampling methods that address relevant issues. In this section, the merits of $\mathcal{P}$-*pooling* are further elaborated by making comparisons with these methods.

L_p norm pooling (or Minkowski norm) is used in [13, 29, 2] to unify standard poolings and is defined as: $f_{L_p}(R_j) = (\sum_{x_i \in R_j} x_i^p)^{1/p}$. However, compared to $\mathcal{P}$-*pooling*, there are two factors that makes it inappropriate as a pooling alternative in end-to-end training. First, its outputs can be greater than any of the input elements, resulting in a slight shift in the data distribution. This phenomenon can be observed in Figure 3(b), especially when $p = 3$. In addition to this, the imperfections in the backward process make L_p norm unsuitable for end-to-end training. This is not only because $\partial E/\partial x_i$ is not equal to the gradient in average ($p = 1$) or max ($p \to +\infty$) pooling, but also because its gradient will explode when $p = 0$ or 1, making optimization more difficult. In fact, [13] choose to manually select p order values rather than learn to improve the network. The detailed backward formulas are presented in the supplementary file.

Gated Pooling [20] is used to fuse max and average pooling with a linear combination whose weights are generated from input data. Compared with $\mathcal{P}$-*pooling*, gated pooling is a simple solution to finding an intermediate function between max and average pooling The function's complexity is exchanged for its ease to use. As shown in Figure 3, the function surface consists of two planes that fold at different angles making it much less non-linear than $\mathcal{P}$-*pooling*.

Detail-Preserving Pooling [26] borrows the idea from

Method	VOC 2012		Cityscapes		ADE20K	
	VGG	ResNet	VGG	ResNet	VGG	ResNet
Max	53.9*	60.0	57.2*	54.3	25.4*	23.5
Average	52.8	59.6	53.5	52.1	24.7	23.7
Strided-C [29, 17]	50.9	60.3*	56.1	55.9*	23.9	23.9*
All-C [29, 17]	not converge		25.2		not converge	
Gated [20]	53.3	59.5	56.7	53.3	25.1	23.7
DPP [26]	54.2	60.1	57.8	54.4	25.3	24.1
$\mathcal{P}$-pooling (ours)	**55.1**	**61.1**	**58.6**	**56.3**	**25.9**	**24.7**

Table 1. Quantitative comparisons for baselines w.r.t. mIoU on the validation sets of the PASCAL VOC 2012, Cityscapes and ADE20K datasets. For ResNet-based models, *'Strided-C'* and *'All-C'* refer to the same structure. * represents the original settings of the pre-trained network. The full results are shown in the supplementary file.

Detail-Preserving Image Downscaling (DPID) [31] in computer graphics which assumes that prominent pixels contribute more to visual satisfactory on preserving details. Essentially DPP is a weighted average of input elements giving higher importance to more anomalous feature. In contrast to $\mathcal{P}$*-pooling*, the hypothesis of DPP doesn't hold in the segmentation task. This is because each input value in the feature space represents the response of a certain pattern. When the function selects a low value as the pooling result over a region filled with high responses, this can be misleading. As shown in Figure 2, the output of DPP can completely convert the input features where both the white and dark corner points disappear in the result.

Strided Convolutions are used to replace pooling layers in works such as [29, 17]. In [29], the authors propose two different types of strided convolutions: *'Strided-C'* which removes a pooling layer and increase the stride for the previous conv layer and *'All-C'* which replaces poolings with newly initialized stride conv layers. Compared with $\mathcal{P}$*-pooling*, (a) strided convolution is still *rigid* as it applies the same kernels on all spatial regions and thus is *non-responsive* to different data distributions. It is obviously difficult for a single conv layer to acquire non-linearity that is comparable with a max pooling. (b) The newly initialized layers in the pretrained models brought by *'All-C'* can easily cause divergence during further fine-tuning.

4. Experiments

To verify the effectiveness of the proposed $\mathcal{P}$*-pooling*, extensive experiments are conducted on different dataset with various network backbones.

4.1. Comparative Studies

$\mathcal{P}$*-pooling* is compared with several of the most related methods. Experimental settings are as follows.

Datasets: Models are trained and tested on the PASCAL VOC 2012 [10], Cityscapes [8] and ADE20K [38] datasets. These three common segmentation benchmarks cover both object and scene segmentation, which enable us to analyze the properties of $\mathcal{P}$*-pooling* under different conditions. All models are evaluated in terms of their mean-intersection-over-union (mIoU, or mean Jaccard Similarity).

The PASCAL VOC 2012 segmentation dataset is composed of the official segmentation benchmark [10] and extra annotations provided by [14]. There are 10,582 images for training and 1499 images for validation, consisting of 20 foreground object classes and one background class.

Cityscapes [8] is a dataset of street scene images from 50 different European cities. The dataset provides fine-grained pixel-level annotations for 19 categories. The training set has 2975 images and the validation set has 500 images.

ADE20K [38] is a scene parsing dataset which provides dense labels for 150 classes on more than 20K scene images. The categories include a large variety of objects (e.g., person, car, etc.) and stuff (e.g., sky, road, etc.). The validation set consists of 2000 images.

Networks: Pre-trained VGG [28] and ResNet [17] models are used in our experiments. The VGG model is a standard single-path network and is the basis for developing more complex network architectures. For the ResNet model, we mainly focus on comparing the pre-trained strided convolutions. VGG and ResNet models were selected because they are the most common network backbones used in segmentation task e.g., [21, 32, 3, 4, 35, 23]. These models are also very basic so that the conclusions drawn on them can be used to generalize $\mathcal{P}$*-pooling* towards other existing works.

The VGG-16 model [28] is used in the experiment. To build up a segmentation model, the stride in *pool5* layer is removed to enlarge feature maps in higher layers, producing an overall stride of 16. A bilinear interpolation at the end of the network restores resolutions of the predictions. For baseline models, different pooling methods replace the five original max-pooling layers. The ResNet-50 model [17] is also used. In practice, the final global average pooling layer is removed and the overall stride is 32. To build up baseline models and make specific comparisons with pre-trained strided convolutions, the *pool1* layer remains unchanged. All strides in *conv1*, *res3a*, *res4a* and *res5a* stages are removed. Different pooling methods are inserted after the *conv1* layer and *res2c*, *res3d* and *res4f* layers.

Training Settings: Essentially, the training settings for

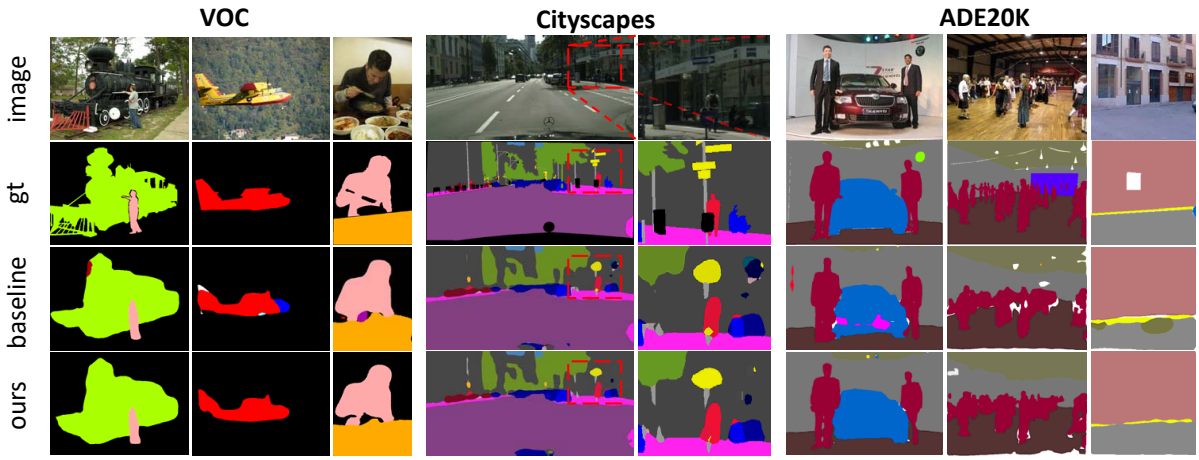

Figure 4. A demonstration of the segmentation results for the VGG-based baseline model (w/ max pooling) and the proposed model (w/ $\mathcal{P}$-*pooling*) on the VOC and Cityscapes datasets. More demonstrations are shown in the supplementary file.

the comparative studies follow the general fine-tuning practice of most fully convolutional networks. Take the *VGG16*-based network trained on the VOC dataset as an example, the learning rate is 2×10^{-7} and the model iterates $10,000$ times and steps at $6,000$. For side-branches, as they are newly initialized layers, their learning rates are multiplied by 3 and input features are scaled up 100 times to speed up their convergence. For initialization, bias terms in the last convolutional layers of each side-branch are set to a large positive value (e.g., 20) enabling the $\mathcal{P}$-*pooling* to begin from max pooling and thus keep the pre-trained model unchanged at the start of fine-tuning. Additionally, due to potential overflow issues, α values are restricted to being smaller than a given constant number, i.e. 35 for the *VGG16*-based network. Our implementation is based on the Caffe library [19]. All experiments are performed on DGX-1 workstations with NVIDIA V100 graphic cards.

Baselines: A series of baseline models are compared to verify the effectiveness of $\mathcal{P}$-*pooling* over the other most relevant pooling methods used in semantic segmentation. The baseline models include standard poolings (max and average poolings), gated pooling [20], DPP [26] and strided convolutions [29, 17]. Among them, we implement the recommended *'gated, layer-wise'* version of gated pooling as described in [20]. The DPP uses its *'sym lite'* version. For the VGG based networks, we implement the strided convolutions strategies *'Strided-C'* and *'All-C'*. For ResNet based networks, the baselines use their vanilla networks.

Evaluation Results: Models trained with baseline pooling methods as well as $\mathcal{P}$-*pooling* are evaluated on the validation sets of PASCAL VOC 2012, Cityscapes and ADE20K. Table 1 and Figure 4 demonstrate the quantitative and qualitative results, respectively. Figure 5 compares the changes at category level performance .

As concluded from Table 1, $\mathcal{P}$-*pooling* outperforms all the baselines. For the VGG-based networks, compared to the max pooling, which is widely used for segmentation, $\mathcal{P}$-*pooling* achieves mIoU improvements of 1.2/1.4/0.5 for

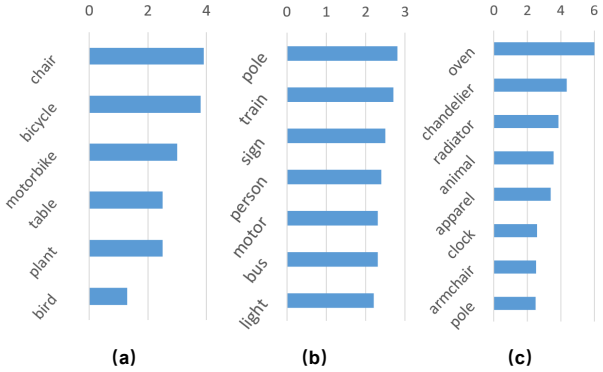

Figure 5. Top positive changes brought by $\mathcal{P}$-*pooling*, comparing to directly fine tuning from original pre-trained VGG on (a) VOC, (b) Cityscapes and (c) ADE20K datasets.

the VOC/Cityscapes/ADE20K datasets, respectively. For ResNet-based networks, $\mathcal{P}$-*pooling* also consistently performs the best, showing 0.8/0.4/0.8 improvement over the default settings. Although ResNet-based networks and VGG-based ones for the Cityscapes and ADE20K datasets, respectively, appear to make less significant improvements, they still shows a notable effect considering the smaller changes obtained by other baselines. Moreover, compared to the recently proposed DPP method, $\mathcal{P}$-*pooling* shows steady advantages in all cases.

Figure 4 demonstrates several segmentation results for all datasets. Overall, using $\mathcal{P}$-*pooling* enable the networks to build up more robust feature representation for details compared to directly fine-tuning on pre-trained models. On the Cityscapes and ADE20K datasets, the $\mathcal{P}$-*pooling* helps to preserve more detailed structures, such as *'poles'*, while the vanilla VGG model simply ignores many. On the VOC dataset, $\mathcal{P}$-*pooling* has a more significant effect on suppressing false positives near object boundaries, indicating a better perception for details.

From Figure 5, it can be concluded that $\mathcal{P}$-*pooling* elevates the performance for many categories on all the VOC, Cityscapes and ADE20K datasets. Specifically, $\mathcal{P}$-*pooling*

Method	$\mathcal{P}$-pooling	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
Deeplab	×	91.2	76.8	33.4	75.2	62.1	67.0	83.2	77.9	79.9	30.5	67.7	49.6	73.2	65.3	70.4	78.4	46.0	71.7	39.8	74.9	53.7	65.1
	✓	91.1	78.1	33.9	75.4	61.2	66.6	83.5	76.2	79.8	31.1	67.7	50.2	72.9	66.5	70.6	78.0	49.0	71.4	41.9	75.2	57.8	65.6
FCN	×	90.4	78.6	32.8	74.4	55.2	65.7	76.8	73.6	76.0	23.1	58.1	30.2	66.3	59.6	62.3	76.6	42.0	67.8	36.9	70.9	54.3	60.6
	✓	91.3	80.0	34.7	75.9	54.7	66.0	81.1	76.1	77.0	27.0	60.1	42.0	65.0	60.7	65.9	78.2	43.5	66.1	32.1	72.5	56.9	62.3
Deeplabv3	×	93.2	80.9	37.2	85.7	66.1	76.5	92.0	82.6	90.9	30.9	86.5	47.7	87.2	85.0	80.2	82.3	51.5	85.5	52.5	82.4	72.0	73.8
	✓	93.3	81.6	38.2	85.9	65.0	77.6	91.9	82.7	91.9	31.5	87.5	48.2	87.0	85.6	83.6	82.4	51.8	86.9	46.1	86.2	73.0	74.2

Table 2. Quantitative evaluation results on PASCAL VOC 2012 test set w.r.t. state-of-the-art models w/ and w/o $\mathcal{P}$-pooling layer. All results are averaged over 5 models that are trained separately to get rid of training fluctuations. The models are evaluated with mean IoU metric.

Method	$\mathcal{P}$-pooling	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
Deeplab	×	91.9	79.7	36.1	76.9	59.9	63.5	83.8	78.5	79.7	27.1	70.1	54	75.1	71.2	79.7	78.4	52.4	77.8	47.1	71.3	58.5	67.27
	✓	92.0	78.7	35.6	77.4	60.7	64.2	82.9	79.8	81.1	28.3	71.3	57.0	75.6	73.2	81.8	78.5	53.9	78.1	46.0	70.1	60.0	67.95
FCN	×	90.7	76.7	36.3	70.1	52.2	61.4	73.7	70.4	74.2	20.0	56.3	38.1	64.7	68.6	74.2	75.6	45.9	68.3	45.7	68.0	49.7	61.01
	✓	91.5	82.6	35.5	67.6	54.5	64.7	76.9	75.2	73.7	24.3	59.7	51.2	63.5	66.8	75.8	77.1	47.7	72.8	43.3	65.3	52.9	62.99
Deeplabv3	×	93.5	87.7	39.2	86.5	69.7	73.7	90.6	85.7	91.5	35.3	83.4	59.5	87.1	84.1	86.0	82.1	56.7	84.3	54.8	82.6	68.6	75.37
	✓	93.8	88.4	39.4	81.9	69.6	73.9	90.5	84.5	92.4	34.9	83.8	63.4	88.4	85.9	83.9	82.1	58.7	83.3	53.5	84.7	71.5	75.64

Table 3. Quantitative evaluation results on PASCAL VOC 2012 validation set w.r.t. state-of-the-art models with and without $\mathcal{P}$-pooling layer. All models are evaluated with mean IoU metric. URLs of the results on VOC evaluation servers are also provided.

Backbones	Training Strategies	Settings
VGG16-20M [3]	Deeplab [3]	16s, Large_FOV
InceptionV2-BN [18]	FCN [23]	8s
Xception-38 [7]	Deeplab v3 [5]	16s, ASPP

Table 4. State-of-the-art models and their specific settings used in the experiment.

has a particularly significant positive effect on small objects or other objects with many detailed structures, such as 'pole' or 'traffic sign' for the Cityscapes dataset, 'chandelier', 'radiator' or 'armchair' for the ADE20K dataset and 'table' or 'plant' in the VOC dataset. For large objects or categories, such as 'wall', 'sidewalk', 'mountain', etc., $\mathcal{P}$*-pooling* still provides improvement, though to a smaller degree.

4.2. Effectiveness on State-of-the-Art Models

The $\mathcal{P}$*-pooling* module is then applied to several state-of-the-art semantic segmentation pipelines to further verify its effectiveness under these more complex settings.

Network: We re-implement three types of state-of-the-art models with their typical settings, which are listed in Table 4. In this experiment, we address $\mathcal{P}$*-pooling*'s performances under the conditions of: (a) more powerful and complicated network backbones, and, more importantly, (b) the other detail-recovery methods adopted by these models.

Among the selected network backbones, VGG16-20 [3] is the truncated version of the standard VGG16 model. The stride of the $pool5$ layer is removed and the number of channels of the fc layers are reduced to $1,024$. The weights of the $fc6$ layers are replaced with dilated 3×3 convolutional kernels. All of these changes minimize the size of model and make the network more suitable for segmentation. InceptionV2-BN [18] and Xception-38 [7] adopt multi-scale feature extraction and fusion mechanisms in order to promote the stability to feature variance, including adding more detail-sensitivity.

The different training strategies contain additional detail-recovery modules that are specially designed for the segmentation task, such as the *Large Field of View* [3], the ASPP module [5] and skip layers [23]. These methods noticeably reduce the inconsistency between extracting high level semantic features in larger receptive fields and keeping low level detailed structure information in local areas. In this experiment we demonstrate that $\mathcal{P}$*-pooling* can further improve a model's detail-sensitivity even when it is applied together with the detail-recovery methods in question.

Experimental Settings: In general, we use the same experimental settings as the original papers. $\mathcal{P}$*-pooling* replaces all down-sampling operations as well as other pooling layers whose stride is 1. To focus the comparisons on down-sampling modules, irrelevant techniques such as pre-training on additional dataset, data augmentation during testing and post processing are removed. As a result, some of the re-implemented models may have inferior performances than the originally reported ones. All the models are evaluated on the PASCAL VOC val/test set and ADE20K val set.

Evaluation Results: Quantitative results are shown in Table 2, 3 and 5. It can be seen that $\mathcal{P}$*-pooling* further improves all models with different training strategies and detail-recovery modules. In particular, $\mathcal{P}$*-pooling* achieves the most significant improvements on the Inceptionv2-based models and increases the mean IoU scores by 1.7/1.9/2.3 on the VOC validation/VOC test/ADE20K validation sets. We also observe smaller but consistent gains for VGG16-20M-based and Xception-38-based models in terms of their overall performances. From the category level comparisons demonstrated in Table 2 and 3, $\mathcal{P}$*-pooling* is especially helpful for segmenting objects that have detailed structures. The performances of all models on certain categories, such as 'table' and 'plant', are improved. In conclusion, even with other detail recovering methods, $\mathcal{P}$*-pooling* nevertheless can enable the networks to preserve more detailed structural information in a complementary way.

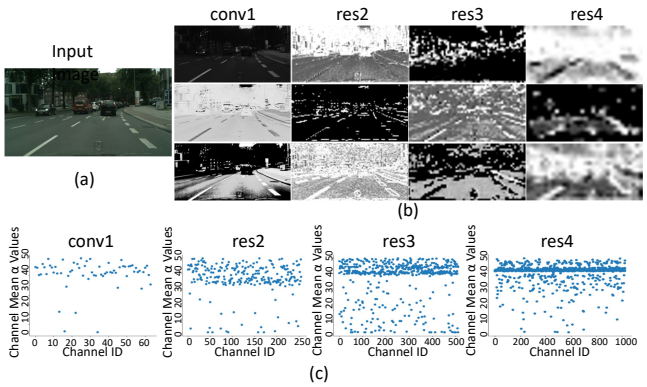

Figure 6. The visualization on α maps from the different stages of the ResNet-based model that is trained on the Cityscapes dataset. (a) input image, (b) α maps in different channels, (c) mean α values in each channel. Brighter pixels on α maps indicate greater α values. For each plot in (b), the x-axis represents the channel's ID and y-axis stands for the mean α value on corresponding channel.

Methods	$\mathcal{P}$-pooling	Pixel Accuracy	mIoU
Deeplab	×	68.45	28.44
	✓	69.22	29.01
FCN	×	66.88	25.81
	✓	68.02	28.09
Deeplabv3	×	73.33	33.93
	✓	73.58	34.70

Table 5. Quantitative evaluation results on ADE20K test set w.r.t. state-of-the-art models with and without $\mathcal{P}$-*pooling* layer. All models are evaluated with pixel wise accuracy and mean IoU metrics.

4.3. Analysis on α Maps

Several examples of α maps predicted from a test image are visualized in Figure 6. The α maps come from each pooling stage of the ResNet-50 model. For each map, α values vary for different positions on the image, indicating the good flexibility of $\mathcal{P}$-*pooling* for adapting to different input data. Specifically, while subtle differences in the α maps can be seen for large objects, the changes are more significant near object edges or around small structures.

When comparing α maps across channels, some channels adopt smaller α values for small details (darker areas) and greater values for large objects (brighter areas) while the other channels behave in the opposite way. This means different channels have various functionalities for extracting features. Figure 6(c) shows the mean α values of each channel. The α values have a wide distribution within the predefined range and show diversity across all channels. In higher levels, more mean values stay around their initialized value. This is because the corresponding input feature maps are sparser than those in lower layers and thus the output of the side-branch in a $\mathcal{P}$-*pooling* unit is also sparse. Then, the bias terms in the last layer of a side-branch dominate the predicted α map.

Method	mIoU
replaced with $\mathcal{P}$-*pooling* in	
pool1	54.9
pool2	54.7
pool3	54.5
pool4	54.8
pool5	54.3
all max pooling	53.9
all $\mathcal{P}$-*pooling*	**55.1**

Table 6. An ablation study on the effectiveness of $\mathcal{P}$-*pooling* in each network stages. All models are evaluated using a VGG-based network on VOC.

4.4. 'Details' Are Not Local

The purpose of $\mathcal{P}$-*pooling* is to add higher detail-sensitivity to a segmentation model. However, *'detail'* does not solely refer to a local area on the input image. Instead, $\mathcal{P}$-*pooling* is able to preserve more information of many factors within one pooling region, such as the values' distribution and the sequential order of input elements, and these information are disentangled from the scales or levels of features. From this perspective, $\mathcal{P}$-*pooling* is effective in each stage of the network. In Table 6, an ablation study is made where only one $\mathcal{P}$-*pooling* layer is used to replace the original *pool1*-*pool5* max pooling one by one. Although all single-stage $\mathcal{P}$-*pooling* models have inferior performances than the *5-p-pooling* model, they nevertheless show noticeable improvement when compared to the 5-max-pooling network. The $\mathcal{P}$-*poolings* in each stage yield similar improvements.

5. Conclusion

In this paper, a new pooling function, *polynomial pooling* ($\mathcal{P}$-*pooling*) was proposed to enhance the detail-sensitivity while preserving the non-linearity of a segmentation network. The proposed $\mathcal{P}$-*pooling* addresses the contradicting purposes of classification models and semantic segmentation, with a function that finds a compromise between max and average pooling. With an adjustable mechanism to provide a combination of detail-sensitivity and non-linearity to arbitrary degrees, $\mathcal{P}$-*pooling* can be dynamically adapted to various data types so as to provide high flexibility inside a network. $\mathcal{P}$-*pooling* is differentiable so as to allow end-to-end training, and it is universally compatible with any pre-trained classification model. Extensive experiments on the VOC, Cityscapes and ADE20k dataset showed the superiority of $\mathcal{P}$-*pooling* over directly fine-tuning the pre-trained models using other pooling methods.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant 61502081, U1536203, 61572493, 61876177), Sichuan Science and Technology Program (No.2019YFG0003, 2018GZDZX0032).

References

[1] G. Bertasius, L. Torresani, S. X. Yu, and J. Shi. Convolutional randomwalk networks for semantic image segmentation. In *CVPR*, 2017.

[2] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010.

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018.

[5] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

[6] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[7] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[9] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

[10] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[11] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, 2016.

[12] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[13] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *ECML*, 2014.

[14] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *PAMI*, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[20] C.-Y. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016.

[21] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

[22] G. Lin, C. Shen, A. van den Hengel, and I. D. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[24] H. Noh, S. Hong, and B. Hann. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[25] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters - improve semantic segmentation by global convolutional network. In *CVPR*, 2017.

[26] F. Saeedan, N. Weber, M. Goesele, and S. Roth. Detail-preserving pooling in deep networks. *arXiv:1804.04076*, 2018.

[27] F. Shen, R. Gan, S. Yan, and G. Zeng. Semantic segmentation via structured patch prediction, context crf and guidance crf. In *CVPR*, 2017.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[29] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR Workshop*, 2015.

[30] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. W. Cottrell. Understanding convolution for semantic segmentation. In *WACV*, 2018.

[31] N. Weber, M. Waechter, S. C. Amend, S. Guthe, and M. Goesele. Rapid, detail-preserving image downscaling. In *TOG*, 2016.

[32] Z. Wei, Y. Sun, J. Wang, H. Lai, and S. Liu. Learning adaptive receptive fields for deep image parsing network. In *CVPR*, 2017.

[33] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv:1511.07122*, 2015.

[34] H. Zhan, K. J. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.

[35] R. Zhang, S. Tang, Y. Zhang, J. Li, and S. Yan. Scale-adaptive convolutions for scene parsing. In *ICCV*, 2017.

[36] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[37] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.

[38] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.