Computational Visual Media
DOI 10.1007/s41095-xxx-xxxx-x Vol. x, No. x, month year, xx–xx

Research Article

Erroneous Pixel Prediction for Semantic Image Segmentation

Lixue Gong[1], Yiqun Zhang[1], Yunke Zhang[1],Yin Yang[2] and Weiwei Xu[1](✉)

Abstract Our method is inspired by the Bayesian deep learning which improves image segmentation accuracy by modeling the uncertainty of the network output. In contrast to uncertainty, our method directly learns to predict the erroneous pixels of a segmentation network, which is modeled as a binary classification problem. It can speed up the training comparing to the Monte Carlo integration often used in the Bayesian deep learning. It also allows us to train a branch to correct the labels of erroneous pixels. Our method consists of three stages: 1) predict pixel-wise error probability of the initial result, 2) re-estimate new labels for the pixels with high error probability, 3) fuse the initial result and the re-estimated result with respect to the error probability. We formulate the error-pixel prediction problem as a classification task and employ an *error-prediction branch* in the network to predict the pixel-wise error probabilities. We also introduce another network branch called *detail branch*. This branch is designed such that the training process is focused on the erroneous pixels. We experimentally validate our method on Cityscapes and ADE20K dataset. Our model can be easily attached to various advanced segmentation networks to improve performance. Taking the segmentation results from DeepLabv3+ as the initial segmentation result, our network can achieve 82.88% of mIoU on Cityscapes testing dataset and 45.73% on ADE20K validation dataset, which is 0.74% and 0.13% higher than DeepLabv3+.

Keywords erroneous pixel prediction; image segmentation; deep learning.

1 the State Key Laboratory of CAD & CG at Zhejiang University, 310058, China. E-mail: L. Gong, gonglx@zju.edu.cn; W. Xu, xww@cad.zju.edu.cn(✉).

2 School of Computing Clemson University, South Carolina, 29634, U.S..

Manuscript received: 2020-12-31; accepted: 2021-01-30.

1 Introduction

The goal of semantic image segmentation is to obtain a high-level representation of an image by assigning each pixel in the image a semantic class label. Semantic image segmentation can be used in video surveillance, medical imaging, autonomous driving, etc. Recently, Deep Convolutional Neural Networks (DCNN) trained on large scale image segmentation datasets, such as PASCAL VOC 2012 [10], Cityscapes [8] and ADE20K [39], have significantly improved the accuracy of image segmentation.

While the end-to-end training of DCNN can effectively learn the multi-scale features for various vision tasks, the down-sampling operations in the encoder designed to enlarge the reception fields are likely to lose the detail information required in pixel-level image segmentation [24]. Thus, the atrous convolution and skip-connections are proposed to balance between the down-sampling operations and the learning of multi-scale features [6, 28]. It has also been shown that the fusion of the global context and multi-scale features is effective to improve the accuracy of image segmentation [18, 21, 38]. However, even with the state-of-the-art image segmentation algorithms, we can still notice a large number of pixels with wrong labels located at regions with in-distinctive RGB information, object boundaries and small-scale objects. These pixels are denoted as erroneous pixels hereafter. While there exist hard-mining methods that train the network according to the gradient information back-propagated from the erroneous pixels, these methods rely on the ground-truth data to detect erroneous pixels, which is not available during the inference. The difficulty-aware method in [19] is a Layer-Cascading method (LC) that focuses on those pixels whose largest label probabilities are less than a threshold in a layer-by-layer manner. However, the erroneous pixels whose largest label probabilities at one layer are greater than the threshold, also called "hard" erroneous pixels, are simply accepted as the result and overlooked in the subsequent layers.

In this paper, we study how to learn to predict the erroneous pixels for a segmentation network so that a cascaded detailed branch can be used to handle erroneous pixels to improve the segmentation accuracy. It runs as a model cascading strategy in the inference: with an existing image segmentation network as a front-end *semantic branch*, we first predict error pixels in its segmentation result, then re-estimates semantic labels for those error pixels, and finally fuse them to obtain the final segmentation result. The difference of our strategy to [19] is that we add a branch into the network to improve the accuracy of error pixel prediction, denoted as *error-prediction branch*. Thus, it is possible, in our method, to predict the overlooked hard erroneous pixels as erroneous pixels to be corrected. The error-pixel prediction is similar to uncertainty modeling in Bayesian deep learning for computer vision tasks. Our method can speed up the training by modeling the error-pixel prediction as a binary classification problem, while the Monte Carlo integration is used to evaluate the objective function in [15]. It implicitly assumes the aleatoric uncertainty can be learned through the difference between the segmentation result and the ground-truth labeling in the training. To correct the detected erroneous pixels, we employ another independent sub-network(called *detail branch*) trained to focus on the segmentation of such pixels.

Since using an independent branch to learn to predict the erroneous pixels does not affect the pixels that the front-end segmentation network can well handle, the *error-prediction branch* and *detail branch* can be used to improve the accuracy of a variety of segmentation networks due to its cascading design. Our network trained on Cityscapes can achieve mIoU at 82.88% on the testing dataset when using DeepLabv3+ as the semantic branch [7], which is 0.74% higher than the original network.

2 Related work

In the following, we mainly review the image segmentation methods using deep neural networks, which are mostly related to our work. Please also refer to [12] for a comprehensive survey.

The encoder-decoder structure is the mostly used fully convolutional neural network structure to generate pixel-wise segmentation results for high-resolution images [2, 24], and a common technique in DNN-based image segmentation algorithms is to fuse multi-scale features to improve the segmentation accuracy. U-Net [28] exploits the skip connections to augment the high-level features with low-level features in the decoder so as to improve the accuracy of localization, which is widely used in many following works [9, 11, 21, 27]. ParseNet [23] adopts a simple global branch to add global context, while [13, 18] use the global feature as a guidance for feature fusion. PSP-Net [38] proposes Pyramid Pooling Module to aggregate more representative context features. Atrous Spatial Pyramid Pooling (ASPP) in [5] uses atrous convolution filters [4, 6] at multiple dilation rates to capture multi-scale image context. In order to handle small objects in the image, EncNet [37] utilizes a context encoding module to explicitly enforce the learning of global scene context. A recent contribution [30] proposed HRNet to improve the segmentation accuracy. The HRNet gradually adds high-to-low resolution subnetworks and fuse the learned multi-scale feature in parallel.

Neural Architecture Search (NAS) method is a new method with the purpose for searching the optimal neural architectures and weights simultaneously. [3] explores the construction of meta-learning techniques for recurrently searching. [25] introduces auxiliary cells that provide an intermediate supervisory signal for architecture parameterization. Auto-DeepLab [22] proposes a hierarchical architecture search space, that is, searching in cell level and network level.

Our work is also related to the popular cascading structure used in computer vision. In object detection, successive classifiers are combined in a cascading structure, which allows the background regions of an image to be quickly discarded while spending more computation on promising regions [17, 20, 26, 33]. For the segmentation task, the cascading structure can also be applied. A Layer-Cascading (LC) method is introduced in [19], while our network is possible to capture hard erroneous pixels overlooked in LC to further improve the segmentation accuracy.

3 Our approach

In the following, we firstly introduce the overall framework of our method, then provide the details about the error-prediction branch and the detail branch of our network. The training strategies are also elaborated.

3.1 Approach Overview

Fig. 1 illustrates an overview of our method, which consists of three modules: 1) A pre-trained segmentation network, namely *semantic branch*, is used to obtain initial segmentation results and semantic

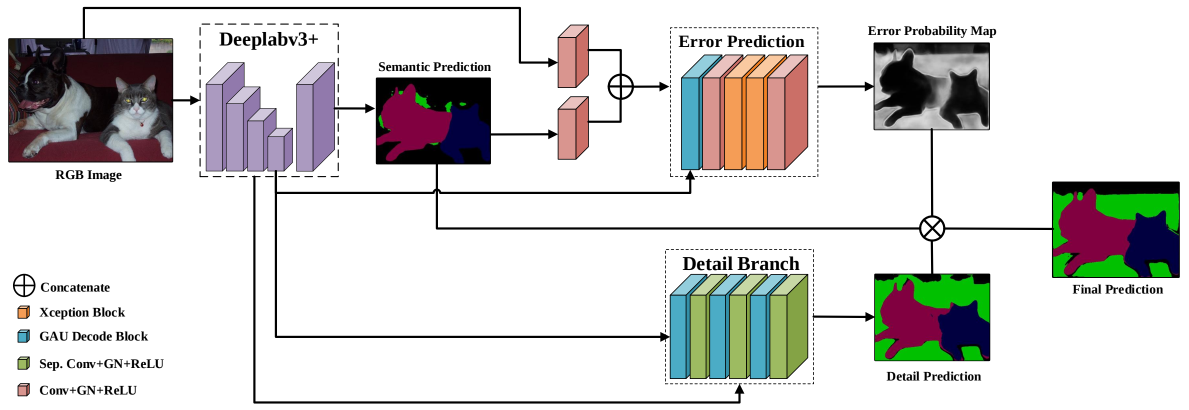


Fig. 1 The architecture of our network. We directly use a pre-trained segmentation network (for example, DeepLabv3+ [7]) as the semantic branch. Then we exploit two branches: the error-prediction branch predicts an error probability map to find out error pixels; the detail branch predicts the correct labels for the mislabeled pixels.

features in Section 3.2. 2) Then, we exploit two branches, error-prediction and detail branches to find out erroneous pixels and predict new labels for erroneous pixels respectively in Section 3.3 and 3.4. After aggregating the initial segmentation result and the newly predicted result from our method, we finally achieve a more accurate segmentation result.

More concretely, given an input image $\mathbf{I}$ and a segmentation network $f_{\rm sb}(\cdot)$, we can obtain the initial segmentation probability map $\mathbf{P}_{\rm sb} = f_{\rm sb}(\mathbf{I})$. For the i-th pixel, $\mathbf{P}^i_{\rm sb} \in \mathbb{R}^{C\times 1}$ indicates the probabilities of this pixel belonging to C categories, respectively. And then the error-prediction branch $f_{\rm ep}(\cdot)$ yields a probability map $\mathbf{P}_{\rm ep} = f_{\rm ep}(\cdot)$ with the same size as the initial result $\mathbf{P}_{\rm sb}$. A pixel with a high probability in $\mathbf{P}_{\rm ep}$ is likely to be wrongly labelled in the initial segmentation. After error prediction, it is expected that those erroneous pixels should be relabelled. The *detail branch*, denoted as $f_{\rm db}(\cdot)$, is responsible to predict new labels for erroneous pixels and predicts a new probability map $\mathbf{P}_{\rm db} = f_{\rm db}(\cdot)$. In the end, the labels of erroneous pixels in the initial label map will be replaced by the new labels generated by the *detail branch* so that we get a more reliable semantic segmentation result.

3.2 Semantic Branch

We directly use a pre-trained segmentation network as the semantic branch. More concretely, we mainly choose DeepLabv3+ [7], PSP-Net [38] and DPC network [3] as our semantic branch in the following experiments. From the pre-trained segmentation network, we can obtain the initial segmentation probability map $\mathbf{P}_{\rm sb}$ and the corresponding low-level and high-level features that will be used in the training of the error-prediction branch and the detail branch.

3.3 Error-prediction Branch

The error-prediction branch aims to predict whether the initial labels given by the semantic branch are erroneous. Specifically, this branch predicts a probability map $\mathbf{P}_{\rm ep}$ in which each pixel value represents the probability that the *semantic branch* prediction is mislabeled. The inputs of this branch consist of (1) the probability map $\mathbf{P}_{\rm sb}$ generated by the *semantic branch*; (2) the feature maps from the direct convolution of the input RGB image; (3) the feature maps from the *semantic branch*. We exploit the Global Attention Upsampling (GAU) module from [18], as illustrated in Fig. 2, to provide channel-wise attention in this branch.

In detail, we firstly apply convolutions to $\mathbf{P}_{\rm sb}$, the probability map output by the semantic branch, and the input RGB image $\mathbf{I}$ separately. The obtained features are then concatenated as the input low-level features. Afterwards, we use the high-level features from the *semantic branch* as the input to GAU. For example, the features generated by ASPP in DeepLabv3+ and Pyramid Pooling Module in PSP-Net are used as the high-level features input to the error-prediction branch.

The loss function for this branch is formulated as a pixel-wise cross-entropy loss to classify each pixel as mislabelled or not, which is a binary classification problem. The ground-truth error map $\mathbf{M}_{\rm err}$ for training is obtained by checking whether the initial segmentation from the semantic branch is inconsistent with ground-truth or not. We use 1 to denote a mislabelled pixel and 0 for the rest. Specifically, the loss function can be written into:

$$\mathbf{M}_{\mathrm{err}}^{i} = \begin{cases} 1, & \mathbf{S}_{\mathrm{sb}}^{i} \neq \mathbf{S}_{\mathrm{gt}}^{i}. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $\mathbf{S}_{\mathrm{sb}}^{i}$ and $\mathbf{S}_{\mathrm{gt}}^{i}$ are the predicted semantic label and the ground-truth semantic label of pixel i, respectively.

Since the number of the erroneous pixels are usually much smaller than the number of correct pixels, we adopt a balanced version cross-entropy to deal with the imbalance in training data. In addition, the erroneous pixels are categorized into two types with different weights counted into the cross-entropy loss: (1) "easy" erroneous pixels. Inspired by [19], we define the erroneous pixels with classification scores smaller than a threshold ρ as the "easy" erroneous pixels (i.e., $\max(\mathbf{P}_{\mathrm{sb}}^{i}) \leq \rho$). These "easy" erroneous pixels are easy to detect from the input of the initial prediction $\mathbf{P}_{\mathrm{sb}}$; (2) "hard" erroneous pixels. The rest erroneous pixels with classification scores larger than ρ are defined as "hard" erroneous pixels (i.e., $\max(\mathbf{P}_{\mathrm{sb}}^{i}) > \rho$). These pixels are misclassified with high confidence which are hard to detect. Hence we add a larger loss weight to the "hard" erroneous pixels. In summary, the balanced cross-entropy loss is formulated as:

$$\begin{aligned} L_{\mathrm{ep}} = & - w_1 \sum_{i \in \mathbf{M}_{\mathrm{err}}^{+e}} \log \mathbf{P}_{\mathrm{ep}}^{i} - w_2 \sum_{i \in \mathbf{M}_{\mathrm{err}}^{+h}} \log \mathbf{P}_{\mathrm{ep}}^{i} \\ & - w_3 \sum_{i \in \mathbf{M}_{\mathrm{err}}^{-}} \log(1 - \mathbf{P}_{\mathrm{ep}}^{i}) \end{aligned} \tag{2}$$

where $\mathbf{M}_{\mathrm{err}}^{+e}$ and $\mathbf{M}_{\mathrm{err}}^{+h}$ are the "easy" erroneous pixels and "hard" erroneous pixels. $\mathbf{M}_{\mathrm{err}}^{-}$ are the negatively labelled pixels. We set $w_1 = 1.0$ and $w_2 = 1.5$. The value of weight w_3 is 0.04 on average, which can be computed according to the proportion of the erroneous pixels for an image.

3.4 Detail Branch

Once we know which pixel is likely to be mislabeled by the semantic branch, we would like to correct the error with the detail branch. Thus, the detail branch is trained to predict the correct labels for the mislabeled pixels. This branch is designed to be a decoder branch to obtain a pixel-wise segmentation result using the features from the semantic branch as input, where the low-level features are fed into the corresponding decoder stages using the skip connections. Specifically, we use 3 successive decoder blocks as shown in Fig. 2 to build the decoder with GAU.

During the training, we require the detail branch to achieve higher accuracy for the erroneous pixels so that it can correct errors of the initial segmentation results from the semantic branch. To this end, we design the loss function to enforce the training to focus on the erroneous pixels captured by the error-prediction branch. Specifically, a pixel-wise weight $\mathbf{E}_{ep}$ derived from $\mathbf{P}_{ep}$ is used in the loss function:

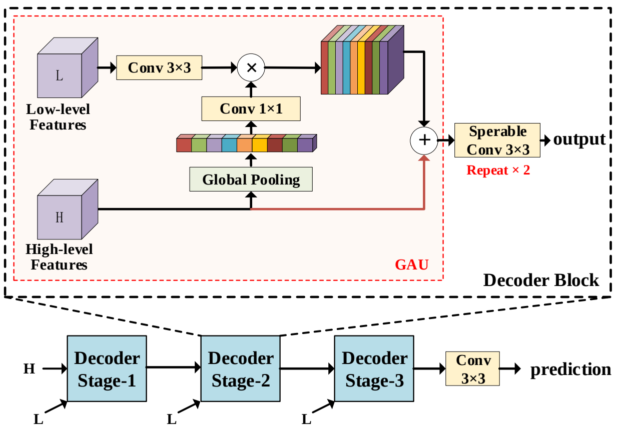


Fig. 2 Detail branch decoder with GAU. "L" denotes the low-level features and "H" denotes the high-level features. "Repeat×2" means 2 convolution layers.

$$L_{\mathrm{db}} = - \sum_{i} \mathbf{E}_{\mathrm{ep}}^{i} \sum_{c}^{C} \mathbf{S}_{\mathrm{gt}}^{i,c} \log \mathbf{P}_{\mathrm{db}}^{i,c} \tag{3}$$

where $\mathbf{P}_{\mathrm{db}}^{i,c}$ is the probability of the i-th pixel belonging to c-th category and $\mathbf{S}_{\mathrm{gt}}^{i,c}$ equals to 1 if the i-th pixel belongs to c-th category, and equals to 0 otherwise. The pixel-wise loss weight $\mathbf{E}_{\mathrm{ep}}$ is a binary map generated from the probability map $\mathbf{P}_{\mathrm{ep}}$ which is predicted by the error-prediction branch with a binarization threshold t:

$$\mathbf{E}_{\mathrm{ep}}^{i} = \begin{cases} 1, & \mathbf{P}_{\mathrm{ep}}^{i} > t \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

With this binary loss weight, the pixels that are classified to be mislabeled, i.e., with probabilities larger than t in $\mathbf{P}_{\mathrm{ep}}$, will contribute to the loss. Thus, our network is also designed as a cascading architecture: the semantic branch is able to classify most of the easy erroneous pixels correctly, and the rest hard erroneous pixels which are highly likely to be mislabeled are passed to the detail branch.

Fusion: In this stage, We need to combine the segmentation results from the semantic branch and the detail branch. The final segmentation result is computed as a pixel-wise linear combination according to the binary error mask $\mathbf{E}_{\mathrm{ep}}$:

$$\mathbf{P}_{\mathrm{f}} = \mathbf{E}_{\mathrm{ep}} \cdot \mathbf{P}_{\mathrm{db}} + (1 - \mathbf{E}_{\mathrm{ep}}) \cdot \mathbf{P}_{\mathrm{sb}} \tag{5}$$

Since the hard erroneous pixels are also trained as erroneous pixels in the error-prediction branch, they can also be corrected in the detail branch if they are correctly classified as erroneous pixels in the inference after the fusion step. It is superior to LC method in [19] where the hard erroneous pixels are simply ignored.

4 Training Strategy

Branch training: Because our method aims to improve a given segmentation network, we keep the semantic branch fixed during the whole training procedure, i.e., the parameters are frozen and both Batch Normalization [14] layers and Dropout [29] layers in the semantic branch are always in inference mode. We first train the error-prediction branch with the loss function defined in Eq. (2) for 60K iterations. After the error-prediction branch is converged, we fix it and update the detail branch according to Eq. (3) for 90K iterations.
Optimizer and learning rate: We adopt a "poly" learning rate policy similar to [4] where the initial learning rate is multiplied by $(1-\frac{iter}{max\_iter})^{power}$ with $power = 0.9$. And we employ Adam [16] as the optimizer during training.
Group Normalization: In general, the performance of Batch Normalization layer is related to the batch size. However, in practice, the batch size is constrained by the limited GPU memory. To improve the stability in the optimization, we adopt Group Normalization [34] in both the error-prediction and the detail branch, in which the channels are divided into 32 groups in our implementation.
Data augmentation: Following the training protocol of [7, 38], we randomly crop patches from the image during training and the crop size is set to 769 (DeepLabv3+ based model) or 713 (PSP-Net based model) on Cityscapes dataset and 513 on ADE20K dataset. For data augmentation, random scaling (from 0.5 to 2 with the step of 0.25), random left-right flipping and random rotation between -10 and 10 degrees are applied.

5 Experiment Results

We evaluate our network on the urban scene dataset Cityscapes [8] and diverse scenes dataset ADE20K [39]. These two datasets provide densely annotated images, which are important for the training of our method to recover segmentation details. Cityscapes dataset contains high-quality dense annotations of 5000 images with 19 object classes(2975, 500 and 1525 for the training set, the validation set, and the testing set, respectively) and 20000 coarsely annotated images. ADE20K is a more challenging dataset with 150 object classes. There are 20210, 2000, and 3000 images for the training set, validation set and testing set, respectively.

Tab. 1 Error prediction evaluation

Threshold	Error-pixels mIoU(%)	Our hard recall (%)
0.1	14.80	79.16
0.2	17.19	70.21
0.3	19.13	61.72
0.4	20.98	52.70
0.5	22.89	42.67
0.6	24.97	31.41
0.7	27.34	19.78
0.8	29.92	9.09
0.9	30.20	3.32

Tab. 2 Ablation study on binarization threshold t

t	0.5	0.6	0.7	0.8
mIoU(%)	79.60	79.61	**79.90**	79.66

5.1 Evaluation of Branches

In this section, we conduct experiments to analyze the performance of the proposed branches in our network. We employ DeepLabv3+ as our semantic branch and keep it fixed in the experiments for the purpose of clarity. The network is trained using the training set of Cityscapes and all the performance statistics are reported on the validation set.
Error-prediction branch: The *error-prediction branch* is just a classifier to predict the pixel-wise error probability. We illustrate the predicted error probability map and ground-truth error map in Fig. 3. The ground-truth error map is computed by the difference between the semantic label map output by the *semantic branch* and the ground-truth.

Given the error probability map, we consider the pixels with the error probability larger than the threshold t as erroneous pixels, the same way as the $\mathbf{E}_{\mathrm{ep}}$ in Eq. (3).Thus, the mean intersection over union (mIoU) between the predicted error mask $\mathbf{E}_{\mathrm{ep}}$ and ground truth error mask $\mathbf{M}_{\mathrm{err}}$, defined as error-pixels mIoU, can be computed, as reported in the second column in Table 1 with different values of the threshold t. In order to show how many hard erroneous pixels are captured by the designed error-prediction branch, we also compute the recall values, the percentage of hard erroneous pixels classified to be the erroneous pixels, and report them in the third column in Tab. 1, named hard recall. The hard erroneous pixels are those mislabelled pixels whose largest class probability is larger than $\rho = 0.95$, which is consistent with the definition in LC.

It can be seen that with increasing value of the binarization threshold t, the mIoU of the erroneous pixels increases while the hard recall drops. Since a

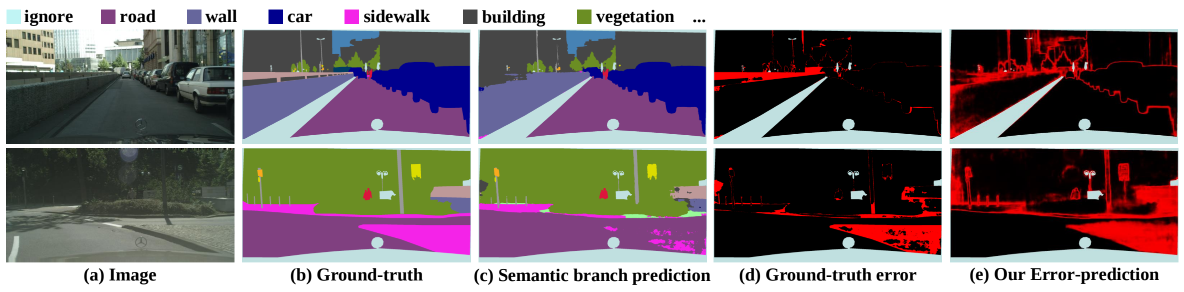


Fig. 3 (a) Input images. (b) The ground truth semantic label maps provided by Cityscapes dataset. (c) The semantic segmentation results output by the semantic branch (DeepLabv3+ in this case). (d) The ground-truth error maps. (e) The error probability maps generated by our error-prediction branch. The error probability with value 1 is colored in red, and 0 in black. White pixels indicate the unlabeled pixels in the dataset.

Tab. 3 Quantitative results on Cityscapes validation set

method	mIoU(%)
DeepLabv3+ [7]	78.79
DeepLabv3+ [7]-GAU-Decoder	78.89
DeepLabv3+-DUpsampling [31]	79.06
LC [19]-GAU-Decoder	79.73
DeepLabv3+ [7]-Hard-mining	79.37
DeepLabv3+ [7]-Bagging	79.38
Ours	**79.90**

small mIoU indicates that a large number of correct pixels are classified to be erroneous pixels which will lead the subsequent training of the detail branch to be distracted, we need to balance between the mIoU and hard recall so as to achieve high segmentation accuracy. The choice of different threshold values and its influence on the final segmentation accuracy on the Cityscapes validation set is reported in Tab. 2. According to the experiments, we set the binarization threshold to 0.7 to continue the training of detail branch.

Detail Branch: The detail branch is trained using the cross-entropy loss at the erroneous pixels predicted by the error-prediction branch. As reported in Tab. 3, the fusion of the segmentation results from the detail branch and the semantic branch can improve the mIoU of DeepLabv3+, used as the semantic branch, by 1.11%.

Since the designed detail branch has 3 decoder stages with additional 11.6 MB parameters, which is more complex than the lightweight decoder of DeepLabv3+, it is worthwhile to verify whether the gains of performance comes from the additional parameters or the cascading of the error-prediction and the detail branch. We thus conduct an experiment to directly replace the original 1-stage decoder in DeepLabv3+ with our detail branch. We train this network variant, called DeepLabv3+-GAU-Decoder, with the same training strategy as DeepLabv3+, where the

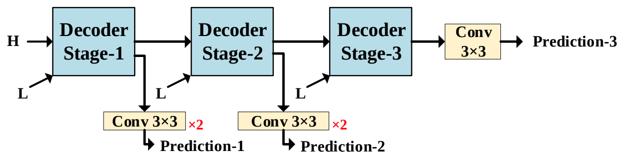


Fig. 4 Layer Cascading adapted from GAU decoder.

cross-entropy loss is equally weighted over all pixels. Its mIoU is reported in the row of DeepLabv3+-GAU-Decoder in Tab. 3, which is 78.89%. It is slightly higher than the original DeepLabv3+ network, but still inferior to our network.

We also report the mIoU of the network proposed by Tian *et al.*. [31], which improves the decoder of DeepLabv3+ with a data-dependent upsampling and improve the mIoU by 0.27% (the third row in Tab. 3). Thus, increasing the complexity of the decoder is not as effective as our method to allocate resource on erroneous pixels.

Running time: The average running time of our full network is 0.71 second. For an input image of the resolution 1025×2049 (the output segmentation result is 1/4 of the input resolution), it takes 0.38 second on average at the semantic branch DeepLabv3+ network, 0.05 second at the error-prediction branch, and 0.28 second at the detail branch.

5.2 Comparison with Layer-cascading and Hard-mining

Layer-cascading (LC): To compare with LC [19], we adopt layer-wise cascading in the decoder of the DeepLabv3+-GAU-Decoder network discussed above and denote such variant model as LC-GAU-Decoder as illustrated in Fig. 4. Similar to LC, the stage-1 predicts a segmentation result, and the pixels with the classification score smaller than a threshold ρ are propagated to stage-2. The stage-2 follows the same propagation procedure. We also set $\rho = 0.95$ to be

Tab. 4 Quantitative results of error-prediction and segmentation with different semantic branches on Cityscapes validation dataset.

	Deeplabv3+ [7]	PSP [38]	DPC [3]
Our error-pixel mIoU(%)	27.34	27.10	27.84
Hard erroneous pixel ratio(%)	19.14	19.64	22.8
Our hard recall	19.79	31.04	21.36
Original mIoU(%)	78.79	79.70	80.31
Ours mIoU(%)	**79.90**	**80.35**	**81.22**

consistent with the definition of the hard erroneous pixels to test how the simply discarding of hard erroneous pixels in LC influences the segmentation results. As reported in Tab. 3, our method can outperform LC-GAU-Decoder at a 0.17% gain.

Hard-mining: For a fair comparison, we employ the loss-rank mining in [36] as a hard-mining method to train the DeepLabv3+-GAU-Decoder network, where the decoder of the DeepLabv3+ is replaced by our proposed decoder in the detail branch. In this method, the cross-entropy loss is calculated for each pixel and then all the pixels are ranked in loss-descent order. Only a certain percentage of pixels with the highest loss (20% in our experiment) contribute to the training process. Although the hard-mining method enhances the training of hard examples, it is still inferior to our network in the statistics of mIoU. The performance of the hard-mining strategy is reported in the 5th row of Tab. 3.

Error-pixel induced fusion vs. Bagging: The bagging result is obtained by training the detail branch by setting every pixel as erroneous pixels and then average its result with the result from the initial DeepLabv3+ network, which leads to mIoU 79.38% (the 6th row in Tab. 3). Instead of directly averaging the results of the semantic branch and the detail branch, we ensemble the two branch results under the guidance of error probability, it gets 0.52% gain with respect to the average bagging. Additional visual comparisons among the methods introduced above are shown in the supplemental materials.

5.3 Integration with Other Segmentation Networks

In this section, we report how the error-prediction and detail branch can be cascaded with PSP-Net [38] and DPC network [3] to improve the segmentation accuracy. Specifically, for PSP-Net, we concatenate the features generated by the pyramid pooling as the high-level features to provide global attention for the error-prediction branch. For DPC, we use the features generated by the dense prediction cell as the input to

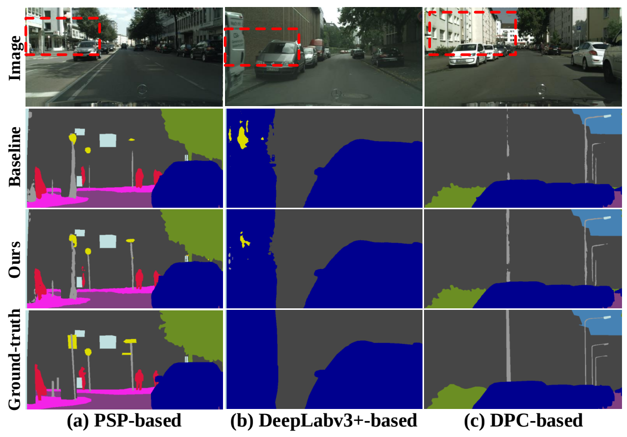


Fig. 5 Visual improvements of our method using different semantic branches on Cityscapes validation dataset. The dashed rectangles highlight the regions where our method can effectively correct the errors in the front end model results.

GAU. The error-pixel mIou, hard recall and the final mIoU of the segmentation results are reported in Tab. 4. The binarization threshold is again set to $t = 0.7$, and the threshold for "hard" erroneous pixels is set to 0.95.

The results suggest that our approach can correct errors and boost mIoU for various advanced segmentation models. Some visual results are shown in Fig. 5. Our method achieves more detailed segmentation results for some "hard" classes like "pole".

5.4 Comparison with other state-of-the-art methods

In this section, we further evaluate our method on Cityscapes benchmark testing dataset (1525 images) and ADE20K validation dataset (2000 images), which have a larger number of testing images than Cityscapes validation dataset (500 images) used in the experiments reported in the last two sections. The binarization threshold is set to 0.7 in all the following experiments.

Cityscapes benchmark testing dataset: We exploit the state-of-the-art method Xception71-DPC as the semantic branch, and train the detail branch on *trainval_fine* set because the finely annotated images in this set can provide valid training data for segmentation details. Our proposed method achieves the mIoU of 82.88% on the test set, as reported in Tab. 5. It improves the DeepLabv3+ by 1.69% and the original DPC network by 0.22%. More visual results are illustrated in Fig. 6.

ADE20K: We select Xception65-DeepLabv3+ as the semantic branch and train our network using ADE20K training set. Our network can improve the accuracy of Xception65-DeepLabv3+ as reported in Tab. 6. The

Tab. 5 Per-class results on Cityscapes testing set.

	road	sidewalk	build.	wall	fence	pole	t.light	t.sigh	veg.	terrain	sky	person	rider	car	truck	bus	train	m.bike	bicycle	mIoU(%)
PSPNet [38]	98.68	86.92	93.47	58.39	63.68	67.67	76.12	80.47	93.64	72.20	95.30	86.83	71.91	96.21	77.70	91.51	83.64	70.80	77.54	81.19
DeepLabv3+ [7]	98.69	87.04	93.91	59.47	63.73	71.39	78.16	82.15	93.96	73.04	95.84	87.95	73.26	96.41	78.02	90.91	83.91	73.84	78.88	82.14
GFF-Net [1]	98.74	87.20	93.91	59.64	64.32	71.52	78.31	82.23	94.00	72.59	95.94	88.20	73.94	96.45	79.83	92.16	84.70	71.53	78.84	82.32
SSMA [32]	98.67	86.88	93.61	57.85	63.43	68.94	77.15	81.14	93.86	73.06	95.32	87.43	73.78	96.36	81.14	93.49	89.95	73.54	78.34	82.31
DPC [3]	98.69	87.12	93.78	57.72	63.53	71.04	78.04	82.09	94.00	73.31	95.44	88.22	74.46	96.47	81.17	93.30	89.03	74.13	78.99	82.66
DRN [41]	98.83	87.72	93.97	65.08	64.20	70.08	77.39	81.59	93.92	73.45	95.81	88.00	74.90	96.46	80.84	92.14	88.47	72.05	78.76	82.83
ours	98.71	87.27	93.81	57.91	64.78	72.06	78.83	82.29	94.07	73.82	95.45	88.54	74.83	96.41	81.36	92.85	88.34	74.69	79.46	**82.88**

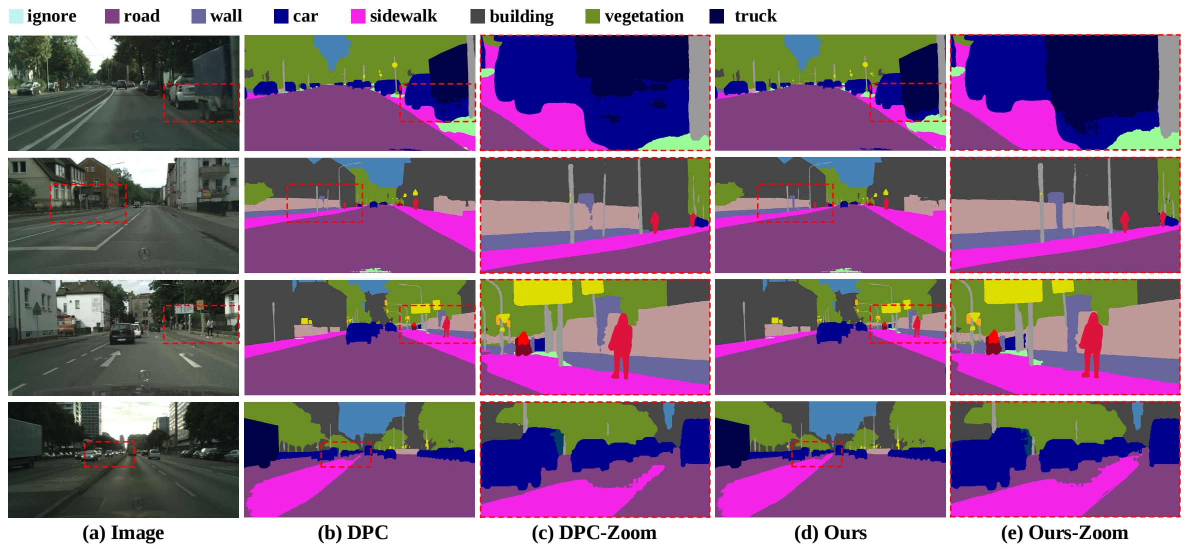


Fig. 6 Visual results selected from Cityscapes testing dataset. Semantic branch: DPC network[3].

Tab. 6 Quantitative results on ADE20K validation set. "MS" means multi-scale inference.

Method	mIoU(%)
DeepLabv3+ [7]	43.06
DeepLabv3+ [7]-MS	45.65
PSP-ResNet50 [38]	41.68
PSP-ResNet50-MS	42.78
DilatedNet [35]	32.31
CascadeNet [40]	34.90
Ours(DeepLabv3+)	43.51
Ours(DeepLabv3+)-MS	**45.73**

qualitative comparisons of the segmentation results are shown in Fig. 7.

6 Conclusions

We propose a method to improve semantic image segmentation result by predicting erroneous pixels and re-estimating semantic label for erroneous pixels. Our method can improve the segmentation mIoU for a given state-of-the-art segmentation network. The experiments results have demonstrated the cascading of error-prediction and detail branch can improve the segmentation results. In the future, we would like to investigate how to improve the mIoU of the erroneous pixels with attention techniques and the layer-wise cascading of error-prediction and image segmentation.

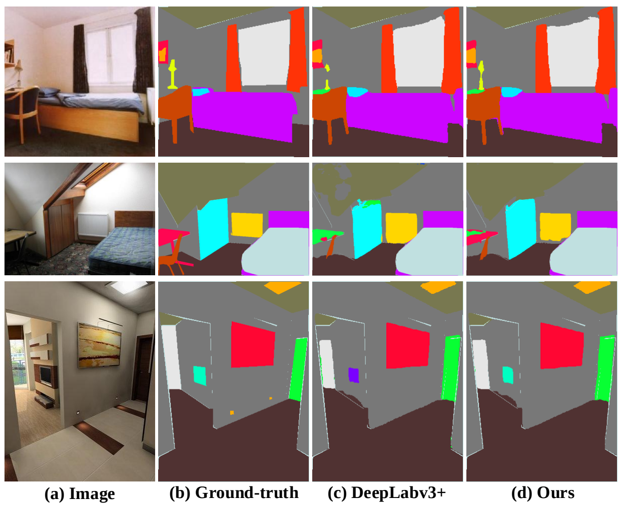


Fig. 7 Visual improvements on ADE20K validation set.

Acknowledgements

We would like to thank anonymous reviewers for their constructive comments. Weiwei Xu is partially supported by the NSFC grant (No 61732016).

References

[1] M. Amirul Islam, M. Rochan, N. D. Bruce, and Y. Wang. Gated feedback refinement network for dense image labeling. In *CVPR*, pages 3751–3759, 2017.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *PAMI*, 39(12):2481–2495, 2017.

[3] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NIPS*, pages 8713–8724, 2018.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[5] 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. *PAMI*, 40(4):834–848, 2018.

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

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 801–818, 2018.

[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*, pages 3213–3223, 2016.

[9] H. Ding, X. Jiang, B. Shuai, A. Qun Liu, and G. Wang. Context contrasted feature and gated multi-scale aggregation for scene segmentation. In *CVPR*, pages 2393–2402, 2018.

[10] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015.

[11] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, pages 519–534. Springer, 2016.

[12] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew. A review of semantic segmentation using deep neural networks. *IJMIR*, 7(2):87–93, Jun 2018.

[13] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018.

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeuroIPS*, June 2017.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, pages 5325–5334, 2015.

[18] H. Li, P. Xiong, J. An, and L. Wang. Pyramid attention network for semantic segmentation. *arXiv preprint arXiv:1805.10180*, 2018.

[19] X. Li, Z. Liu, P. Luo, C. Change Loy, and X. Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *CVPR*, pages 3193–3202, 2017.

[20] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *ICIP*, volume 1, pages I–I. IEEE, 2002.

[21] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, pages 1925–1934, 2017.

[22] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019.

[23] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[25] V. Nekrasov, H. Chen, C. Shen, and I. Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9126–9135, 2019.

[26] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV*, pages 887–895, 2017.

[27] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *CVPR*, pages 4353–4361, 2017.

[28] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*,

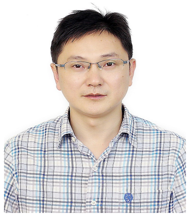
15(1):1929–1958, 2014.
[30] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.
[31] Z. Tian, T. He, C. Shen, and Y. Yan. Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation. In *CVPR*, June 2019.
[32] A. Valada, R. Mohan, and W. Burgard. Self-supervised model adaptation for multimodal semantic segmentation. *arXiv preprint arXiv:1808.03833*, 2018.
[33] P. Viola, M. Jones, et al. Rapid object detection using a boosted cascade of simple features. 1:511–518, 2001.
[34] Y. Wu and K. He. Group normalization. In *ECCV*, pages 3–19, 2018.
[35] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
[36] H. Yu, Z. Zhang, Z. Qin, H. Wu, D. Li, J. Zhao, and X. Lu. Loss rank mining: A general hard example mining method for real-time detectors. In *IJCNN*, pages 1–8. IEEE, 2018.
[37] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. In *CVPR*, pages 7151–7160, 2018.
[38] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017.
[39] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017.
[40] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 127(3):302–321, 2019.
[41] Y. Zhuang, F. Yang, L. Tao, C. Ma, Z. Zhang, Y. Li, H. Jia, X. Xie, and W. Gao. Dense relation network: Learning consistent and context-aware representation for semantic image segmentation. In *ICIP*, pages 3698–3702. IEEE, 2018.

Lixue Gong received the M.S degree from the College of Computer Science And Technology, Zhejiang University in 2020, the B.S degree in digital media technology from Zhejiang University in 2017. Her research interests include image segmentation, image matting and video enhancement.

Yiqun Zhang is currently a student in the College of Computer Science And Technology, Zhejiang University. She received the B.S degree in digital media technology from Zhejiang University in 2019. Her research interests include image generation and segmentation.

Yunke Zhang is currently a Ph.D candidate at Zhejiang University. He received the M.S degree from the Hangzhou Institute of Service Engineering, Hangzhou University in 2018, the B.S degree in Software Engineering from the Zhengzhou University in 2015. His research interests include image, video matting and segmentation.

Yin Yang is an Associate Professor with School of Computing, Clemson University. Before joining Clemson, he was a faculty member with Electrical and Computer Engineering Department of the University of New Mexico. He is still a research faculty with UNM ECE and CS. He received his Ph.D. degree at the University of Texas at Dallas (with David Daniel fellowship). He is a recipient of the NSF CRII award (2015) and CAREER award (2019). His research aims to develop efficient and customized computing methods for challenging problems in Graphics , Animation, Machine Learning , Vision, Visualization, Simulation, HCI, Robotics, Medicine, and many other applied areas.



Weiwei Xu is currently a researcher at state key lab of CAD&CG in Zhejiang university. He was a Qianjiang Professor at Hangzhou Normal University and a researcher in Internet Graphics Group at Microsoft Research Asia from 2005 to 2012. He was a post-doc researcher at Ritsmeikan university in Japan for more than one year. He received Ph.D. Degree in Computer Graphics from Zhejiang University, Hangzhou, and B.S. Degree and Master Degree in Computer Science from Hohai University in 1996 and 1999 respectively.