Computational Visual Media
https://doi.org/10.1007/s41095-021-0235-7 Vol. 8, No. 1, March 2022, 165–175

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 We consider semantic image segmentation. Our method is inspired by 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 training comparing to the Monte Carlo integration often used in Bayesian deep learning. It also allows us to train a branch to correct the labels of erroneous pixels. Our method consists of three stages: (i) predict pixel-wise error probability of the initial result, (ii) redetermine new labels for pixels with high error probability, and (iii) fuse the initial result and the redetermined 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 pixel-wise error probabilities. We also introduce a detail branch to focus the training process on the erroneous pixels. We have experimentally validated our method on the Cityscapes and ADE20K datasets. Our model can be easily added to various advanced segmentation networks to improve their performance. Taking DeepLabv3+ as an example, our network can achieve 82.88% of mIoU on Cityscapes testing dataset and 45.73% on ADE20K validation dataset, improving corresponding DeepLabv3+ results by 0.74% and 0.13% respectively.

Keywords erroneous pixel prediction; image segmentation; deep learning

1 State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China. E-mail: L. Gong, gonglx@zju.edu.cn; Yiqun Zhang, zyqlouise@zju.edu.cn; Yunke Zhang, yunkezhang@zju.edu.cn; W. Xu, xww@cad.zju.edu.cn (✉).

2 School of Computing Clemson University, South Carolina, 29634, USA. E-mail: yin5@clemson.edu.

Manuscript received: 2021-01-13; accepted: 2021-03-30

1 Introduction

The goal of semantic image segmentation is to obtain a high-level representation of an image by assigning each pixel 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 [1], Cityscapes [2], and ADE20K [3] have significantly improved the accuracy of image segmentation.

While end-to-end training a DCNN can effectively learn multi-scale features for various vision tasks, the down-sampling operations in the encoder designed to enlarge the receptive field are likely to lose detailed information required for pixel-level image segmentation [4]. Thus, atrous convolution and skip-connections are used to balance down-sampling operations and learning of multi-scale features [5, 6]. It has also been shown that fusing global context and multi-scale features can effectively improve the accuracy of image segmentation [7–9]. However, even with state-of-the-art image segmentation algorithms, we can still see a large number of pixels with wrong labels in regions with indistinct RGB information, at object boundaries and in small-scale objects. We call these erroneous pixels. While hard-mining methods exist that train the network using gradient information back-propagated from the erroneous pixels, these methods rely on ground-truth data to detect erroneous pixels, which is not available during inferencing. The difficulty-aware method in Ref. [10] 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 in one layer are greater than the threshold, which we refer to as *hard* erroneous pixels, are simply accepted as part of the result and overlooked in 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 segmentation accuracy. It runs as a model cascading strategy during inferencing: using an existing image segmentation network as a front-end *semantic branch*, we first predict error pixels in its segmentation result, then redetermine semantic labels for those error pixels, and finally fuse them to obtain the final segmentation result. The difference of our strategy to that of Ref. [10] is that we add an *error-prediction* branch to the network to improve the accuracy of error pixel prediction. 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. Our method can speed up training by modeling error-pixel prediction as a binary classification problem, as an alternative to Monte Carlo integration used to evaluate the objective function in Ref. [11]. It implicitly assumes that the aleatoric uncertainty can be learned through the difference between the segmentation result and the ground-truth labeling in training. To correct the detected erroneous pixels, we employ another independent sub-network, the *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 handle well, 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 [12], which is 0.74% higher than the original network.

2 Related work

In the following, we mainly review image segmentation methods using deep neural networks, which are mostly related to our work. Please see Ref. [13] for a comprehensive survey.

The encoder–decoder is the fully convolutional neural network structure most used for pixel-wise segmentation for high-resolution images [4, 14]. A common technique in DNN-based image segmentation algorithms is to fuse multi-scale features to improve segmentation accuracy. U-Net [5] exploits skip connections to augment high-level features with low-level features in the decoder so as to improve the accuracy of localization, and is widely used in many works [9, 15–17]. ParseNet [18] adopts a simple global branch to add global context, while Refs. [8, 19] use the global feature to guide feature fusion. PSP-Net [7] proposes a pyramid pooling module to aggregate representative context features. Atrous spatial pyramid pooling (ASPP) in Ref. [20] uses atrous convolution filters [6, 21] at multiple dilation rates to capture multi-scale image contexts. In order to handle small objects in the image, EncNet [22] utilizes a context encoding module to explicitly enforce learning of global scene context. A recent contribution [23] proposed HRNet to improve segmentation accuracy; it gradually adds high-to-low resolution subnetworks and fuses the learned multi-scale features in parallel.

Neural architecture search (NAS) is a new method which aims to find the optimal neural architecture and weights simultaneously. Ref. [24] explores the construction of meta-learning techniques for recurrently searching. Ref. [25] introduces auxiliary cells that provide an intermediate supervisory signal for architecture parameterization. Auto-DeepLab [26] proposes a hierarchical architecture search, searching at 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 [27–30]. The cascading structure can also be applied to segmentation. A layer-cascading (LC) method is introduced in Ref. [10], but our network can capture hard erroneous pixels overlooked in LC to further improve the segmentation accuracy.

3 Approach

In the following, we first introduce the overall framework of our method, and then provide details of the error-prediction branch and the detail branch of our network. Training strategy is also described.

3.1 Overview

Figure 1 provides an overview of our method, which consists of three modules: (i) a pre-trained segmentation network, the *semantic branch*, which is used to obtain initial segmentation results and semantic features (see Section 3.2), (ii) error-prediction and detail branches to find erroneous pixels and predict new labels for them respectively (see Sections 3.3 and 3.4), and (iii) a module to combine the initial segmentation result and the newly predicted labels, providing a more accurate segmentation result (see Section 3.5).

More concretely, given an input image $\boldsymbol{I}$ and a segmentation network $f_{\mathrm{sb}}(\cdot)$, we obtain the initial segmentation probability map $\boldsymbol{P}_{\mathrm{sb}} = f_{\mathrm{sb}}(\boldsymbol{I})$. For the i-th pixel, $\boldsymbol{P}_{\mathrm{sb}}^{i} \in \mathbb{R}^{C\times 1}$ gives the probabilities of this pixel belonging to each of C categories. The error-prediction branch $f_{\mathrm{ep}}(\cdot)$ yields a probability map $\boldsymbol{P}_{\mathrm{ep}} = f_{\mathrm{ep}}(\cdot)$ with the same size as the initial result $\boldsymbol{P}_{\mathrm{sb}}$. A pixel with a high probability in $\boldsymbol{P}_{\mathrm{ep}}$ is likely to be wrongly labelled in the initial segmentation. After error prediction, those erroneous pixels should be relabelled. The *detail branch*, denoted $f_{\mathrm{db}}(\cdot)$, is responsible for predicting new labels for erroneous pixels and predicts a new probability map $\boldsymbol{P}_{\mathrm{db}} = f_{\mathrm{db}}(\cdot)$. Finally, labels of erroneous pixels in the initial label map are replaced by the new labels generated by the *detail branch*, giving 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 used DeepLabv3+ [12], PSP-Net [7], and the DPC network [24] as our semantic branch in the following experiments. The pre-trained segmentation network gives the initial segmentation probability map $\boldsymbol{P}_{\mathrm{sb}}$ and corresponding low-level and high-level features that are 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 $\boldsymbol{P}_{\mathrm{ep}}$ in which each pixel value represents the probability that the *semantic branch* prediction is mislabeled. The inputs of this branch consist of (i) the probability map $\boldsymbol{P}_{\mathrm{sb}}$ generated by the semantic branch, (ii) the feature maps from the direct convolution of the input RGB image, and (iii) the feature maps from the semantic branch. We exploit the global attention upsampling (GAU) module from Ref. [8], as illustrated in Fig. 2, to provide channel-wise attention in this branch.

In detail, we firstly apply convolutions to $\boldsymbol{P}_{\mathrm{sb}}$, the probability map output by the semantic branch, and the input RGB image $\boldsymbol{I}$ 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 the 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

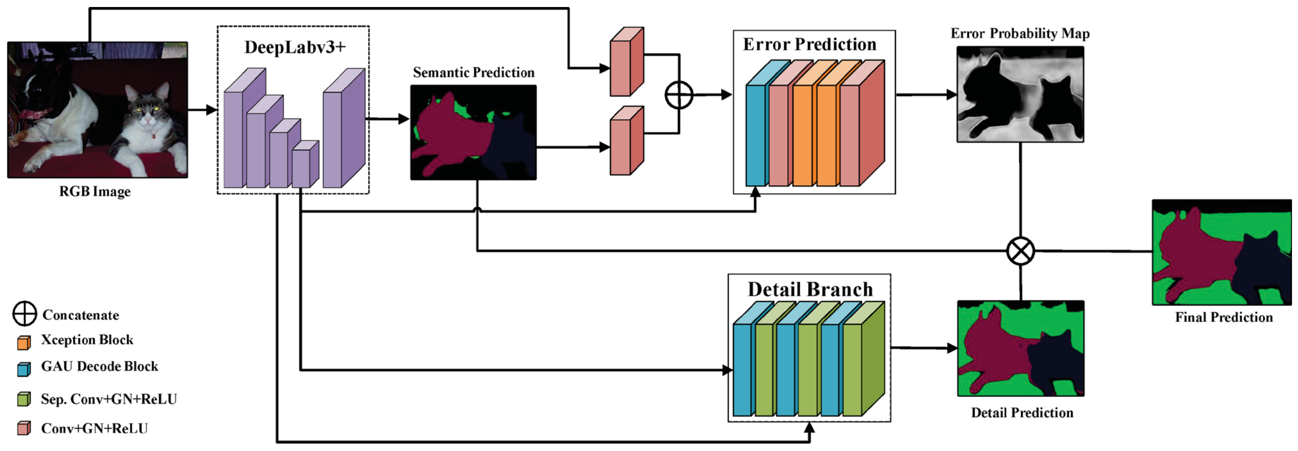


Fig. 1 Architecture of our network. We use a pre-trained segmentation network (for example, DeepLabv3+ [12]) as the semantic branch. We add two branches: the error-prediction branch predicts an error probability map to find error pixels, and the the detail branch predicts the correct labels for the mislabeled pixels.

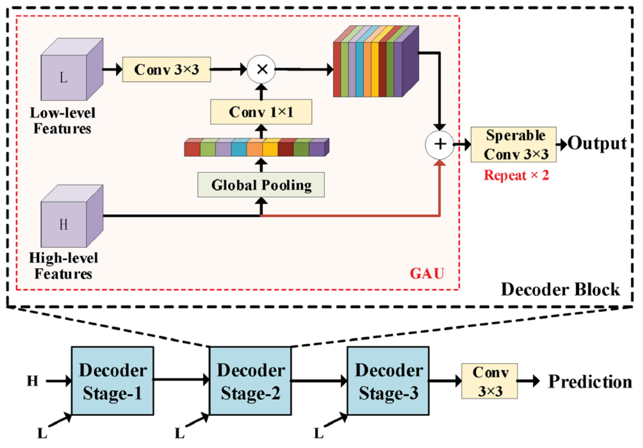


Fig. 2 Detail branch decoder with GAU. "L" and "H" represent low- and high-level features respectively. "Repeat × 2" indicates 2 convolution layers.

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 $\boldsymbol{M}_{\text{err}}$ 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 otherwise. Specifically, the loss function is

$$\boldsymbol{M}^i_{\text{err}} = \begin{cases} 1, & \boldsymbol{S}^i_{\text{sb}} \neq \boldsymbol{S}^i_{\text{gt}} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

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

Since the number of the erroneous pixels is 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 Ref. [10], we define the erroneous pixels with classification scores smaller than a threshold ρ as the "easy" erroneous pixels (i.e., $\max(\boldsymbol{P}^i_{\text{sb}}) \leqslant \rho$). These "easy" erroneous pixels are easy to detect from the input of the initial prediction $\boldsymbol{P}_{\text{sb}}$; (2) "hard" erroneous pixels. The rest erroneous pixels with classification scores larger than ρ are defined as "hard" erroneous pixels (i.e., $\max(\boldsymbol{P}^i_{\text{sb}}) > \rho$). 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

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

where $\boldsymbol{M}^{+\text{e}}_{\text{err}}$ and $\boldsymbol{M}^{+\text{h}}_{\text{err}}$ are the "easy" erroneous pixels and "hard" erroneous pixels respectively. $\boldsymbol{M}^{-}_{\text{err}}$ 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 pixels are likely to be mislabeled by the semantic branch, we wish to correct the errors 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 features from the semantic branch as input, where the low-level features are fed into the corresponding decoder stages using skip connections. Specifically, we use 3 successive decoder blocks as shown in Fig. 2 to build the decoder with GAU.

During training, we require the detail branch to achieve higher accuracy for the erroneous pixels so that it can correct errors in 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 $\boldsymbol{E}_{\text{ep}}$ derived from $\boldsymbol{P}_{\text{ep}}$ is used in the loss function:

$$L_{\text{db}} = -\sum_i \boldsymbol{E}^i_{\text{ep}} \sum_c^C \boldsymbol{S}^{i,c}_{\text{gt}} \log \boldsymbol{P}^{i,c}_{\text{db}} \tag{3}$$

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

$$\boldsymbol{E}^i_{\text{ep}} = \begin{cases} 1, & \boldsymbol{P}^i_{\text{ep}} > t \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

With this binary loss weight, the pixels that are classified as mislabeled, i.e., with probabilities larger than t in $\boldsymbol{P}_{\text{ep}}$, 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 other hard erroneous pixels which are highly likely to be mislabeled are passed to the detail branch.

3.5 Fusion

During the fusion stage, we 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 $\boldsymbol{E}_{\text{ep}}$:

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

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 during inferencing after the fusion step. This is superior to the LC method [10] in which hard erroneous pixels are simply ignored.

4 Training strategy

4.1 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 [31] layers and dropout [32] layers in the semantic branch are always in inferencing mode. We first train the error-prediction branch with the loss function defined in Eq. (2) for 60k iterations. After the error-prediction branch has converged, we fix it and update the detail branch using Eq. (3) for 90k iterations.

4.2 Optimizer and learning rate

We adopt a *poly* learning rate policy similar to Ref. [21] where the initial learning rate is multiplied by $(1 - \text{iter}/\text{max\_iter})^{\text{power}}$ with power $= 0.9$. We then employ Adam [33] as the optimizer during training.

4.3 Group normalization

In general, the performance of the batch normalization layer is related to the batch size. However, in practice, the batch size is constrained by the limited GPU memory. To improve stability during optimization, we adopt group normalization [34] in both error-prediction and detail branches; the channels are divided into 32 groups in our implementation.

4.4 Data augmentation

Following the training protocol of Refs. [7, 12], we randomly crop patches from the image during training, with a crop size of 769 (DeepLabv3+ based model) or 713 (PSP-Net based model) for the Cityscapes dataset, and 513 for the ADE20K dataset. For data augmentation, random scaling (from 0.5 to 2 with a step size of 0.25), random left-right flipping, and random rotation between $-10°$ and $10°$ are applied.

5 Experiments

5.1 Datasets

We evaluated our network on an urban scene dataset, Cityscapes [2], and a diverse scenes dataset, ADE20K [3]. These two datasets provide densely annotated images, which are important to recover segmentation details when training our method. The Cityscapes dataset contains high-quality dense annotations with 19 object classes for 5000 images (2975, 500, and 1525 for the training, validation, and testing sets, respectively) and 20,000 coarsely annotated images. ADE20K is a more challenging dataset with 150 object classes, withe 20,210, 2000, and 3000 images for the training, validation, and testing sets, respectively.

5.2 Evaluation of branches

In this section, we consider experiments to analyze the performance of the proposed branches in our network. We employed DeepLabv3+ as our semantic branch and kept it fixed in the experiments for ease of interpretation. The network was trained using the Cityscapes training set and all outcomes are reported for the validation set.

5.2.1 Error-prediction branch

The error-prediction branch is just a classifier to predict the pixel-wise error probability. We illustrate a predicted error probability map and ground-truth error map in Fig. 3. The ground-truth error map is computed as the difference between the semantic label map output by the semantic branch and the ground-truth.

Given the error probability map, we consider pixels with error probability larger than the threshold t to be erroneous pixels, from $\boldsymbol{E}_{\text{ep}}$ in Eq. (3). Thus, the mean intersection over union (mIoU) between

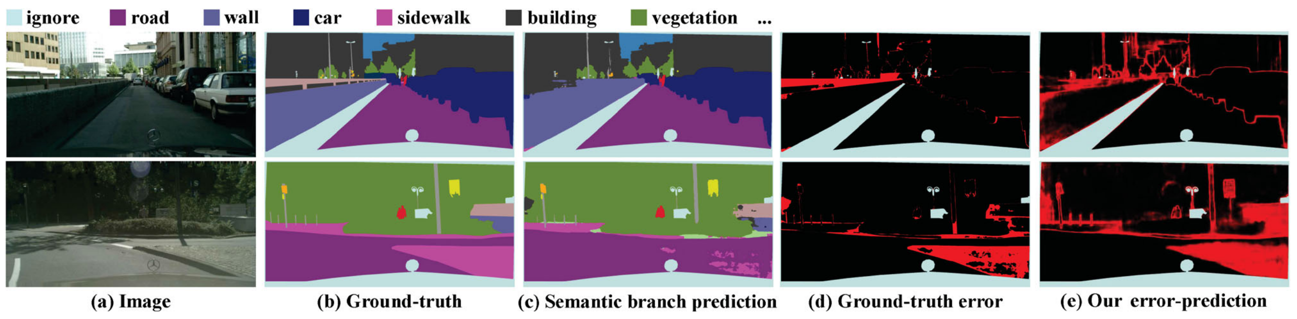


Fig. 3 (a) Input images. (b) Ground-truth semantic label maps provided by the Cityscapes dataset. (c) Semantic segmentation results output by the semantic branch (DeepLabv3+ here). (d) Ground-truth error maps. (e) Error probability maps generated by our error-prediction branch. Red: error probability = 1. Black: error probability = 0. White: unlabeled pixels in the dataset.

the predicted error mask $\boldsymbol{E}_{\text{ep}}$ and ground-truth error mask $\boldsymbol{M}_{\text{err}}$, defined as error-pixels mIoU, can be computed, as reported in the 2nd column in Table 1 for different values of the threshold t. In order to show how many hard erroneous pixels are captured by the error-prediction branch, we also compute the recall values, the percentage of hard erroneous pixels classified as erroneous pixels, and report them in the 3rd column in Table 1 as 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 the LC.

It can be seen that with increasing value of binarization threshold t, the mIoU of the erroneous pixels increases while the hard recall drops. Since a small mIoU indicates that a large number of correct pixels are classified as erroneous pixels, which will distract the subsequent training of the detail branch, we need to balance the mIoU and hard recall so as to achieve high segmentation accuracy. The influence of the choice of different threshold values on the final segmentation accuracy on the Cityscapes validation set is reported in Table 2. As a result of the experiments, we set the binarization threshold to 0.7 for training the detail branch.

Table 1 Error-prediction evaluation

Threshold	Error-pixel mIoU	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%

Table 2 Effect of binarization threshold t

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

5.2.2 Detail branch

The detail branch is trained using the cross-entropy loss for the erroneous pixels predicted by the error-prediction branch. As reported in Table 3, 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%.

The designed detail branch has 3 decoder stages requiring an additional 11.6 MB memory for its parameters, and is more complex than the lightweight decoder of DeepLabv3+. It is thus worthwhile to verify whether the performance gains come from the additional parameters or the cascading of the error-prediction and the detail branch. We thus conducted an experiment to directly replace the original 1-stage decoder in DeepLabv3+ with our detail branch. We trained this network variant, called DeepLabv3+-GAU-Decoder, with the same training strategy as DeepLabv3+, in which the cross-entropy loss is equally weighted over all pixels. Its mIoU (78.89%) is reported in the row for DeepLabv3+-GAU-Decoder in Table 3: it is slightly higher than

Table 3 Quantitative results on Cityscapes validation set

Method	mIoU
DeepLabv3+ [12]	78.79%
DeepLabv3+ [12]-GAU-Decoder	78.89%
DeepLabv3+-DUpsampling [35]	79.06%
LC [10]-GAU-Decoder	79.73%
DeepLabv3+ [12]-Hard-mining	79.37%
DeepLabv3+ [12]-Bagging	79.38%
Ours	**79.90%**

the original DeepLabv3+ network, but still inferior to our overall network.

We also report the mIoU of the network proposed by Tian et al. [35], which improves the decoder of DeepLabv3+ by data-dependent upsampling and improves the mIoU by 0.27% (the 3rd row in Table 3). Thus, increasing the complexity of the decoder is not as effective as our method of allocating resources to erroneous pixels.

5.2.3 Running time

The average running time of our full network is 0.71 s for an input image with resolution 1025×2049 (the output segmentation result is 1/4 of the input resolution), the semantic branch taking 0.38 s, the error-prediction branch 0.05 s, and the detail branch 0.28 s.

5.3 Comparisons to layer-cascading and hard-mining

5.3.1 Layer-cascading

To compare with LC [10], we adopt layer-wise cascading in the decoder of the DeepLabv3+-GAU-Decoder network discussed above; we denote this variant model as LC-GAU-Decoder and illustrate it in Fig. 4. As in LC, stage 1 predicts a segmentation result, and pixels with classification score smaller than a threshold ρ are propagated to stage 2. Stage 2 follows the same propagation procedure. We set $\rho = 0.95$ to be consistent with the definition of the hard erroneous pixels to test how simply discarding hard erroneous pixels in LC influences the segmentation results. As reported in Table 3, our method can outperform the LC-GAU-Decoder by a 0.17% gain.

5.3.2 Hard-mining

For a fair comparison, we employed loss-rank mining from Ref. [36] as a hard-mining method to train the DeepLabv3+-GAU-Decoder network, where the decoder of 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 pixels are ranked in order of descending loss. Only a proportion of pixels with the highest loss

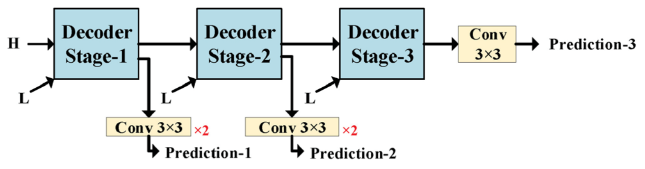


Fig. 4 Layer cascading adapted from GAU decoder.

(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 terms of mIoU: see the 5th row of Table 3.

5.3.3 Error-pixel based fusion vs. bagging

A bagging result was obtained by training the detail branch by setting every pixel as erroneous and then averaging its result with the result from the initial DeepLabv3+ network, which leads to an mIoU of 79.38% (the 6th row in Table 3). Instead of directly averaging the results of the semantic branch and the detail branch, we combine the two branch results guided by error probability, which gives a 0.52% gain with respect to average bagging.

Additional visual comparisons using the methods introduced above are shown in the Electronic Supplementary material (ESM).

5.4 Integration with other segmentation networks

In this section, we report how the error-prediction and detail branch can be cascaded with PSP-Net [7] and the DPC network [24] to improve segmentation accuracy. Specifically, for PSP-Net, we concatenate features generated by pyramid pooling as 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 GAU. The error-pixel mIoU, hard recall, and the final mIoU of segmentation results are reported in Table 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. Various visual results are shown in Fig. 5. Our method achieves more detailed segmentation results for some difficult classes like "pole".

Table 4 Quantitative results of error-prediction and segmentation using different semantic branches on the Cityscapes validation dataset

	DeepLabv3+	PSP [7]	DPC
Our error-pixel mIoU	27.34%	27.10%	27.84%
Hard erroneous pixel ratio	19.14%	19.64%	22.80%
Our hard recall	19.79%	31.04%	21.36%
Original mIoU	78.79%	79.70%	80.31%
Our mIoU	**79.90%**	**80.35%**	**81.22%**

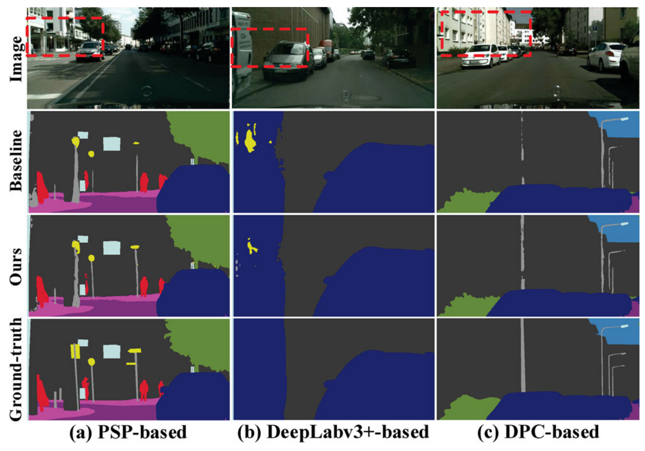


Fig. 5 Visual improvements of our method using different semantic branches on Cityscapes validation dataset. The dashed rectangles highlight the regions where our method can effectively correct the errors in the front end model results.

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

In this section, we further evaluate our method on the Cityscapes benchmark testing dataset (1525 images) and the ADE20K validation dataset (2000 images), which have more images than the Cityscapes validation dataset (500 images) used in the experiments reported so far. The binarization threshold is set to 0.7 below.

5.5.1 *Cityscapes benchmark testing dataset*

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

5.5.2 *ADE20K*

We selected Xception65-DeepLabv3+ as the semantic branch and trained our network using the ADE20K training set. Our network improves the accuracy of Xception65-DeepLabv3+ as reported in Table 6. A qualitative comparison of the segmentation results is shown in Fig. 7.

Table 5 Per-class results on the 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	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+	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	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	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	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	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**

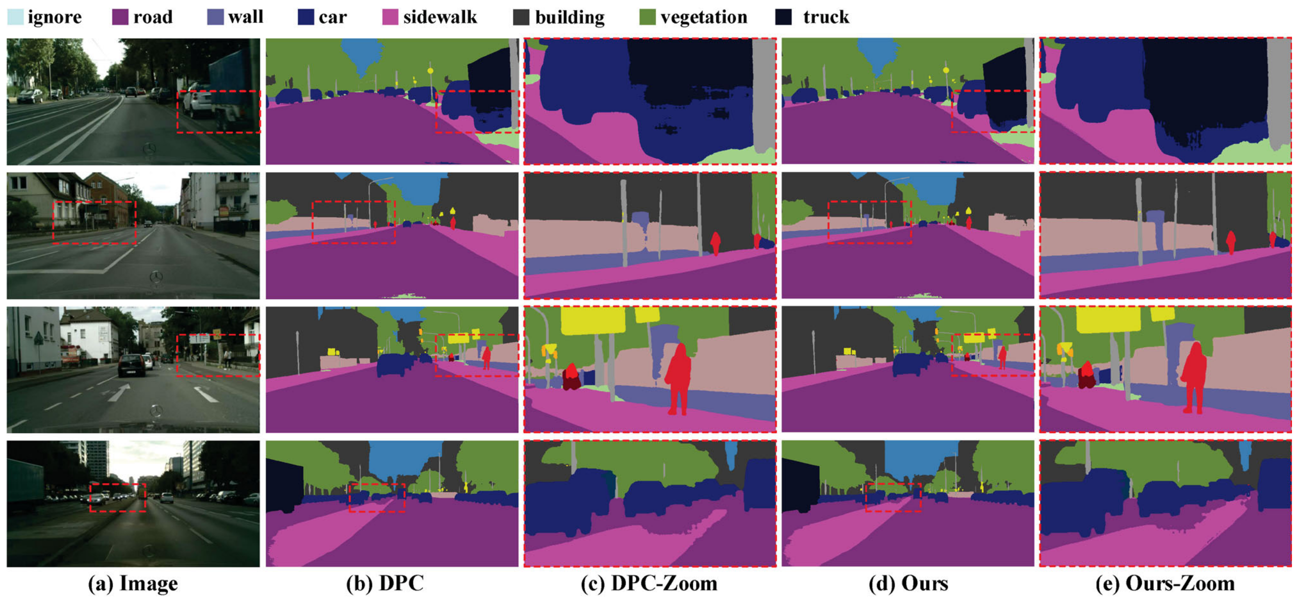


Fig. 6 Visual results selected from the Cityscapes testing dataset. Semantic branch: DPC network [24].

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

Method	mIoU
DeepLabv3+ [12]	43.06%
DeepLabv3+ [12]-MS	45.65%
PSP-ResNet50 [7]	41.68%
PSP-ResNet50-MS	42.78%
DilatedNet [37]	32.31%
CascadeNet [38]	34.90%
Ours(DeepLabv3+)	43.51%
Ours(DeepLabv3+)-MS	**45.73%**

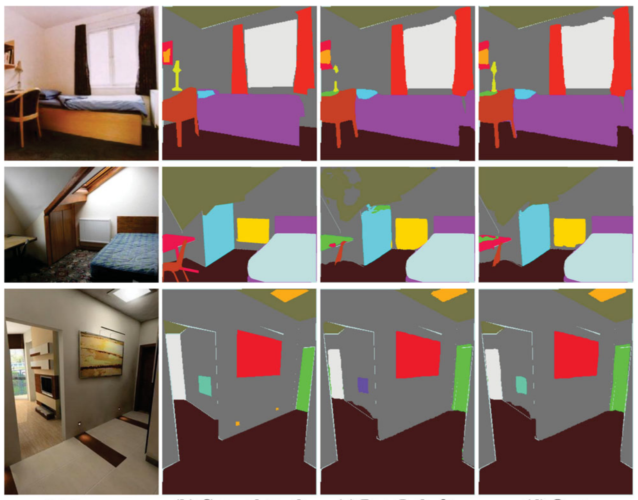


Fig. 7 Visual improvements on the ADE20K validation set.

6 Conclusions

We have proposed a method to improve semantic image segmentation results by predicting erroneous pixels and re-estimating the semantic label for these pixels. Our method can improve the segmentation mIoU for state-of-the-art segmentation networks. The experimental results have demonstrated that cascading error-prediction and detail branches can improve segmentation results. In future, we would like to investigate how to improve the mIoU of the erroneous pixels with attention techniques and layer-wise cascading of error-prediction and image segmentation.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments. Weiwei Xu is partially supported by the National Natural Science Foundation of China (No. 61732016).

Electronic Supplementary Material Supplementary material is available in the online version of this article at `https://doi.org/10.1007/s41095-021-0235-7`.

References

[1] Everingham, M.; Eslami, S. M. A.; van Gool, L.; Williams, C. K. I.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision* Vol. 111, No. 1, 98–136, 2015.

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

[3] Zhou, B. L.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; Torralba, A. Scene parsing through ADE20K dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 633–641, 2017.

[4] Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3431–3440, 2015.

[5] Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention. Lecture Notes in Computer Science, Vol. 9351.* Navab, N.; Hornegger, J.; Wells, W.; Frangi, A. Eds. Springer Cham, 234–241, 2015.

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

[7] Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6230–6239, 2017.

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

[9] Lin, G. S.; Milan, A.; Shen, C. H.; Reid, I. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5168–5177, 2017.

[10] Li, X.; Liu, Z.; Luo, P.; Loy, C. C.; Tang, X. Not all pixels are equal: Dificulty-aware semantic segmentation via deep layer cascade. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6459–6468, 2017.

[11] Kendall, A.; Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In: Proceedings of the 31st Conference on Neural Information Processing Systems, 2017.

[12] Chen, L. C.; Zhu, Y. K.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: *Computer Vision–ECCV 2018. Lecture Notes in Computer Science, Vol. 11211.* Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 833–851, 2018.

[13] Guo, Y. M.; Liu, Y.; Georgiou, T.; Lew, M. S. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval* Vol. 7, No. 2, 87–93, 2018.

[14] Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 12, 2481–2495, 2017.

[15] Ghiasi, G.; Fowlkes, C. C. Laplacian pyramid reconstruction and refinement for semantic segmentation. In: *Computer Vision–ECCV 2016. Lecture Notes in Computer Science, Vol. 9907.* Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 519–534, 2016.

[16] Peng, C.; Zhang, X. Y.; Yu, G.; Luo, G. M.; Sun, J. Large kernel matters—improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1743–1751, 2017.

[17] Ding, H. H.; Jiang, X. D.; Shuai, B.; Liu, A. Q.; Wang, G. Context contrasted feature and gated multi-scale aggregation for scene segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2393–2402, 2018.

[18] Liu, W.; Rabinovich, A.; Berg, A. C. ParseNet: Looking wider to see better. *arXiv preprint* arXiv:1506.04579, 2015.

[19] Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7132–7141, 2018.

[20] Chen, L. C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A. L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 40, No. 4, 834–848, 2018.

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

[22] Zhang, H.; Dana, K., Shi, J. P.; Zhang, Z. Y.; Wang, X. G.; Tyagi, A.; Agrawal, A. Context encoding for semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7151–7160, 2018.

[23] Sun, K.; Xiao, B.; Liu, D.; Wang, J. D. Deep high-resolution representation learning for human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5686–5696, 2019.

[24] Chen, L.-C.; Collins, M.; Zhu, Y.; Papandreou, G.; Zoph, B.; Schrofi, F.; Adam, H.; Shlens, J. Searching for efficient multi-scale architectures for dense image prediction. In: Proceedings of the 32nd Conference on Neural Information Processing Systems, 8713–8724, 2018.

[25] Nekrasov, V.; Chen, H.; Shen, C. H.; Reid, I. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9118–9127, 2019.

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

[27] Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 511–518, 2001.

[28] Lienhart, R.; Maydt, J. An extended set of Haar-like features for rapid object detection. In: Proceedings of the International Conference on Image Processing, 2002.

[29] Pang, J. H.; Sun, W. X.; Ren, J. S.; Yang, C. X.; Yan, Q. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, 878–886, 2017.

[30] Li, H. X.; Lin, Z.; Shen, X. H.; Brandt, J.; Hua, G. A convolutional neural network cascade for face detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5325–5334, 2015.

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

[32] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* Vol. 15, 1929–1958, 2014.

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

[34] Wu, Y.; He, K. Group normalization. In: *Computer Vision–ECCV 2018. Lecture Notes in Computer Science, Vol. 11217.* Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 3–19, 2018.

[35] Tian, Z.; He, T.; Shen, C. H.; Yan, Y. L. Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3121–3130, 2019.

[36] Yu, H.; Zhang, Z. N.; Qin, Z.; Wu, H.; Li, D. S.; Zhao, J.; Lu, X. Loss rank mining: A general hard example mining method for real-time detectors. In: Proceedings of the International Joint Conference on Neural Networks, 2018.

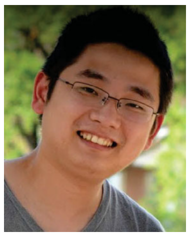
[37] Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint* arXiv:1511.07122, 2015.

[38] Zhou, B.; Zhao, H.; Puig, X.; Xiao, T.; Fidler, S.; Barriuso, A.; Torralba, A. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision* Vol. 127, No. 3, 302–321, 2019.

Lixue Gong received her M.S. degree from the College of Computer Science and Technology, Zhejiang University in 2020, and her 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 her 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 his M.S. degree from Hangzhou Institute of Service Engineering, Hangzhou University in 2018, and his B.S. degree in software engineering from Zhengzhou University in 2015. His research interests include image and video matting and segmentation.



Yin Yang is an associate professor with the School of Computing, Clemson University. Previously, he was a faculty member with the Electrical and Computer Engineering Department of the University of New Mexico. He is still a research faculty member at UNM ECE and CS. He received his Ph.D. degree from the University of Texas at Dallas (with a David Daniel fellowship). He is a recipient of an NSF CRII award (2015) and a 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 other applied areas.

Weiwei Xu is currently a researcher at the State Key Lab of CAD&CG in Zhejiang University. He was a Qianjiang Professor at Hangzhou Normal University and a researcher in the Internet Graphics Group at Microsoft Research Asia from 2005 to 2012. He was a post-doc researcher at Ritsmeikan University in Japan for over one year. He received his Ph.D. degree in computer graphics from Zhejiang University, and B.S. and master degrees in computer science from Hohai University in 1996 and 1999, respectively.