Specialize and Fuse: Pyramidal Output Representation for Semantic Segmentation

Chi-Wei Hsiao*[1] Cheng Sun*[1,2] Hwann-Tzong Chen[1,3] Min Sun[1,4]

National Tsing Hua University[1] ASUS AICS Department[2] Aeolus Robotics[3]

Joint Research Center for AI Technology and All Vista Healthcare[4]

{chiweihsiao,chengsun}@gapp.nthu.edu.tw htchen@cs.nthu.edu.tw sunmin@ee.nthu.edu.tw

Abstract

We present a novel pyramidal output representation to ensure parsimony with our "specialize and fuse" process for semantic segmentation. A pyramidal "output" representation consists of coarse-to-fine levels, where each level is "specialize" in a different class distribution (e.g., more stuff than things classes at coarser levels). Two types of pyramidal outputs (i.e., unity and semantic pyramid) are "fused" into the final semantic output, where the unity pyramid indicates unity-cells (i.e., all pixels in such cell share the same semantic label). The process ensures parsimony by predicting a relatively small number of labels for unity-cells (e.g., a large cell of grass) to build the final semantic output. In addition to the "output" representation, we design a coarse-to-fine contextual module to aggregate the "features" representation from different levels. We validate the effectiveness of each key module in our method through comprehensive ablation studies. Finally, our approach achieves state-of-the-art performance on three widely-used semantic segmentation datasets—ADE20K, COCO-Stuff, and Pascal-Context.

1. Introduction

Given an RGB image, semantic segmentation defines semantic labels for all pixels as "output". Recent methods on semantic segmentation widely exploit deep neural networks. One major research direction is designing new contextual modules [4, 5, 6, 13, 27, 28, 29, 34] exploring better "feature" representation in the networks. We argue that leveraging the structure in the "output" representation could open up opportunities orthogonal to the current endeavors. We observe that a large portion of pixels in most images share the same label at a coarse spatial level (*e.g.*, stuff classes like sky and grass, or central region of objects). This observation induces a parsimonious strategy to dynamically predict semantic labels at a coarser level according to the spatial distribution of classes in each input image.

We proposed a novel pyramidal output representation to ensure parsimony with our "specialize and fuse" process (Fig. 1). Firstly, rather than a single-level output, a pyramidal output starting from the coarsest level to the finest level is designed so that each level is learned to "specialize" in a different class distribution (*e.g.*, more stuff than things classes at coarser levels). Specifically, two types of pyramidal output (unity and semantic pyramid) are predicted. Unity pyramid identifies whether a patch of pixels (referred to as a cell) shares the same label (referred to as a unity-cell) (Fig. 1-first row), and semantic pyramid consists of semantic labels at multiple levels (Fig. 1-second row). Finally, the semantic pyramid are "fused" into one single semantic output according to the unity-cells across levels (Fig. 1-bottom panel). Note that our "specialize and fuse" process ensures parsimony by predicting a relatively small number of labels for unity-cells (*e.g.*, a large cell of grass) to build the final semantic output. In addition to the "output" representation, we design a coarse-to-fine contextual module to aggregate the "features" representation from different levels for improving semantic pyramid prediction.

Our main contributions are as follows: *i*) we introduce a pyramidal "output" representation and a "specialize and fuse" process to allow each level to specialize in different class distribution and ensure parsimony; *ii*) we design a contextual module to aggregate the "features" representation from different levels for further improvements; *iii*) we showcase the effectiveness of our method on ADE20K, COCO-Stuff, and Pascal-Context. Our method with both HRNet and ResNet as the backbone can achieve results on par with or better than the recent state-of-the-art methods.

2. Related work

Contextual modules. Context is important to the task of semantic segmentation, with more and more improvements coming from the newly designed context spreading strategy. PSPNet [32] proposes to pool deep features into several small and fixed spatial resolutions to generate global contextual information. Deeplab [2] employs dilated CNN layers with several dilation rates, which helps the model capture different ranges of context. Recently, self-attention

*The authors contribute equally to this paper.

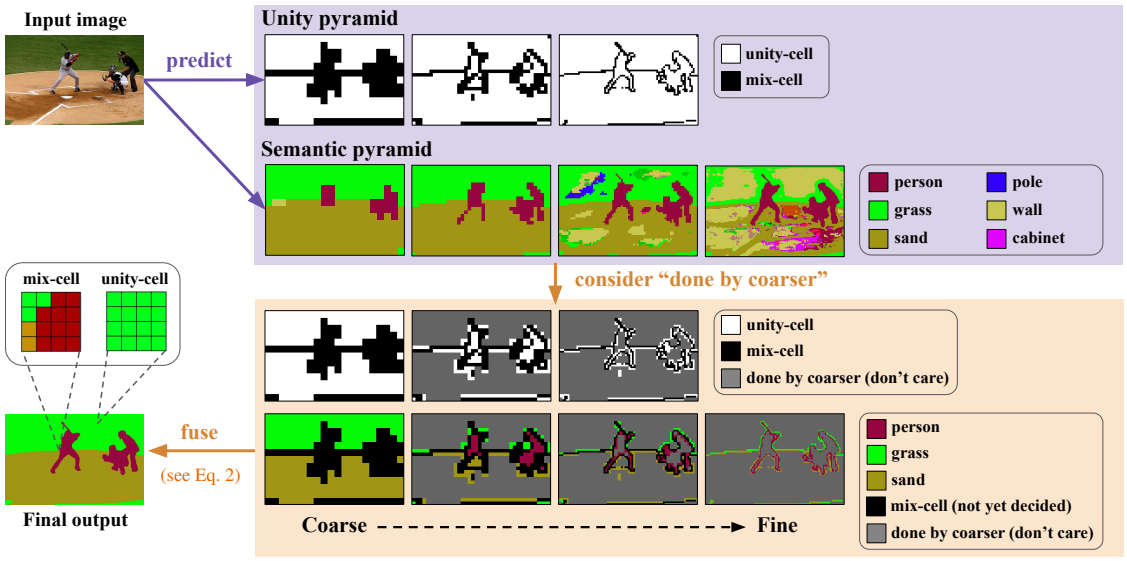


Figure 1: An overview of the "specialize and fuse" approach. We train a neural network to predict two pyramidal outputs: **Unity pyramid** classifies cells into "unity-cell" (*i.e.*, all covered pixels share the same class) or "mix-cell" (*i.e.*, covered pixels contain multiple classes); **Semantic pyramid** predicts the semantic labels at multiple levels. The orange bottom panel illustrates how we fuse the two predicted pyramids into one final semantic output (the bottom-left-most image). Intuitively, a unity-cell at a coarser level indicates that all cells covered by it at finer levels are "done by coarser" and thus can be ignored (colored in grey). Therefore, we acquire the final semantic labels from unity-cells in a coarse-to-fine manner. In other words, mix-cells or "done by coarser" cells are ignored during training (Sec. 3.2) and inference (Sec. 3.3). Our approach achieves parsimony by training the network to predict a relatively small number of semantic labels for unity-cells (*i.e.*, most cells at finer levels are "done by coarser") and enables each pyramid level to specialize in different class distribution.

methods [22, 24] achieve great success in natural language processing and computer vision, with many variants being proposed for semantic segmentation. DANet [6] applies self-attention in spatial dimension and channel dimension. CCNet [13] proposes criss-cross attention in which a pixel attends only to pixels of the same column or row. ANL [34] pools features to a fixed spatial size, which acts as the key and value of attention. OCR [27] pools the context according to a coarse prediction and computes attention between deep features and class centers. CCNet [13], ANL [34], and OCR [27] are able to reduce the computation via specially designed attending strategies while still retain or even improve the performance. Inspired by ANL [34], we design a new coarse-to-fine contextual module in this work, and furthermore, the contextual module is managed to integrate seamlessly with the proposed pyramidal output format.

Hierarchical semantic segmentation prediction. Layer Cascade (LC) [17] predicts three semantic maps of the same resolution using three cascaded sub-networks: Each sub-network passes uncertain pixels to the next sub-network for further prediction, and all of the LC predictions are of the same level. In contrast, our method provides a new output representation that is independent of sub-networks in the backbone, and predicts multi-level semantic maps trained under the principle of parsimony. Besides, unlike LC which simply combines the semantic maps based on the semantic prediction itself, we train a *unity pyramid* with a carefully defined physical meaning to infer with the *semantic pyramid*.

PointRend [14] and QGN [3] are two recent approaches that explore the direction of hierarchical prediction where the final semantic segmentation map is reconstructed in a coarse-to-fine manner instead of a dense prediction from the deep model directly. Both approaches start from the coarsest prediction. PointRend [14] gradually increases the resolution by sampling only uncertain points for the finer prediction, while QGN [3] predicts C+1 classes where the extra "composite" class indicates whether a point would be propagated to a finer level in their SparseConv [8] decoder. Both approaches yield high-resolution prediction (the same as input resolution) with their efficient sparseness design. PointRend [14] achieves slightly better mIoU by refining the prediction to a high spatial resolution; while, QGN [3] focuses on computational efficiency but results in inferior performance. The hierarchical output in previous works [3, 14, 17] entwine with their model's data flow, while our pyramidal output format is more flexible to the model architecture as long as it yields the two pyramids. Besides, rather than results refining and computation saving, the proposed pyramidal output representation with tailored training and fusing procedures (*i.e.*, the proposed *Specialize and Fuse* strategy) is able to achieve state-of-the-art performance.

3. Pyramidal output representation

The overview of our "specialize and fuse" process given two types of pyramidal outputs are shown in Fig. 1. In the following, we first define the semantic and unity pyramids in Sec. 3.1. Then, the "training to specialize" and "fuse in inference" phases are introduced in Sec. 3.2 and Sec. 3.3, respectively.

3.1. Semantic pyramid and unity pyramid

Pyramid structure. We adopt the *coarse-to-fine pyramid* structure to build our pyramidal output format, where a finer level has double resolution than its adjacent coarser level, and all cells (*i.e.*, a patch of pixels) except those in the finest level have exactly four children. Besides, the width and height of an input image should be divisible by those of the coarsest level; otherwise, we resize the input RGB to the nearest divisible.

Notation. We denote the index of the pyramid level by ℓ, where ℓ=1 is the coarsest level and ℓ=L is the finest level (L levels in total). Let s_ℓ denote the spatial stride at the pyramid level ℓ, D the latent dimension of backbone features, and C the number of output classes. As shown in Fig. 2b and Fig. 2c, our model takes a feature tensor $X \in \mathbb{R}^{D\times\frac{H}{s_L}\times\frac{W}{s_L}}$ from the backbone, and predicts a *semantic pyramid* $\left\{\hat{Y}^{(\ell)} \in \mathbb{R}^{C\times\frac{H}{s_\ell}\times\frac{W}{s_\ell}}\right\}_{\ell=1,\dots,L}$ and a *unity pyramid* $\left\{\hat{U}^{(\ell)} \in \mathbb{R}^{\frac{H}{s_\ell}\times\frac{W}{s_\ell}}\right\}_{\ell=1,\dots,L-1}$. The channel dimension of $\hat{U}$ is 1 (binary classification) and is discarded. Note that the output stride of the finest level s_L is the same as the output stride of the backbone feature X in this work for simplicity. The final semantic output fused from two pyramids is denoted as $\hat{Y} \in \mathbb{R}^{C\times\frac{H}{s_L}\times\frac{W}{s_L}}$. The pyramidal ground truths $Y^{(\ell)}, U^{(\ell)}$ for training are derived from the ground truth per-pixel semantic labeling Y.

Pyramidal ground truth. At pyramid level ℓ, each cell is responsible for a patch of $s_\ell \times s_\ell$ pixels in the original image. A cell can be either an unity cell with the same label for all pixels within the cell, or a mix-cell with more than one labels. In the ground truth unity pyramid $U^{(\ell)}$, positive and negative values indicate unity-cells and mix-cells, respectively. In the ground truth semantic pyramid $Y^{(\ell)}$, for a unity-cell, its ground truth semantic label is defined as the shared label by all covered pixels in the original per-pixel ground truth Y, whereas, for a mix-cell, its ground truth semantic label is ill-defined and ignored in computing training loss. Note that a unity-cell at level ℓ implies its child cells are also unity-cells at level $\ell+1$. To avoid redundancy, the children unity-cells (referred to as "done by coarser") are ignored both in computing training loss (Sec. 3.2) and in the fuse-phase (Sec. 3.3).

3.2. *Specialize*—the training phase

Our experiments show that naively training the predicted $\hat{Y}^{(\ell)}, \hat{U}^{(\ell)}$ with their ground-truth counterparts $Y^{(\ell)}, U^{(\ell)}$ is unable to provide any improvement. This setting does not utilize the fact that a large number of pixels belonging to unity-cells are already predicted at a coarser level, and hence the finer level had better not be redundantly trained on those predicted regions. Based on the motivation to encourage parsimony and to train specialized pyramid levels, for those cells whose predecessors in the pyramid structure are already correctly classified as unity-cells (true positives), our training procedure re-labels them as "don't care" on the fly (we refer to such labels as "done by coarser"). With the relabeled ground truths in each mini-batch, the training loss is computed as follows:

$$\begin{aligned}\mathcal{L} =& \frac{1}{L}\sum_{\ell=1}^{L} \mathrm{CE}(\hat{Y}^{(\ell)},\, Y^{(\ell)}_{\text{relabeled}}) \\ &+ \frac{1}{L-1}\sum_{\ell=1}^{L-1} \mathrm{BCE}(\hat{U}^{(\ell)},\, U^{(\ell)}_{\text{relabeled}})\,,\end{aligned} \tag{1}$$

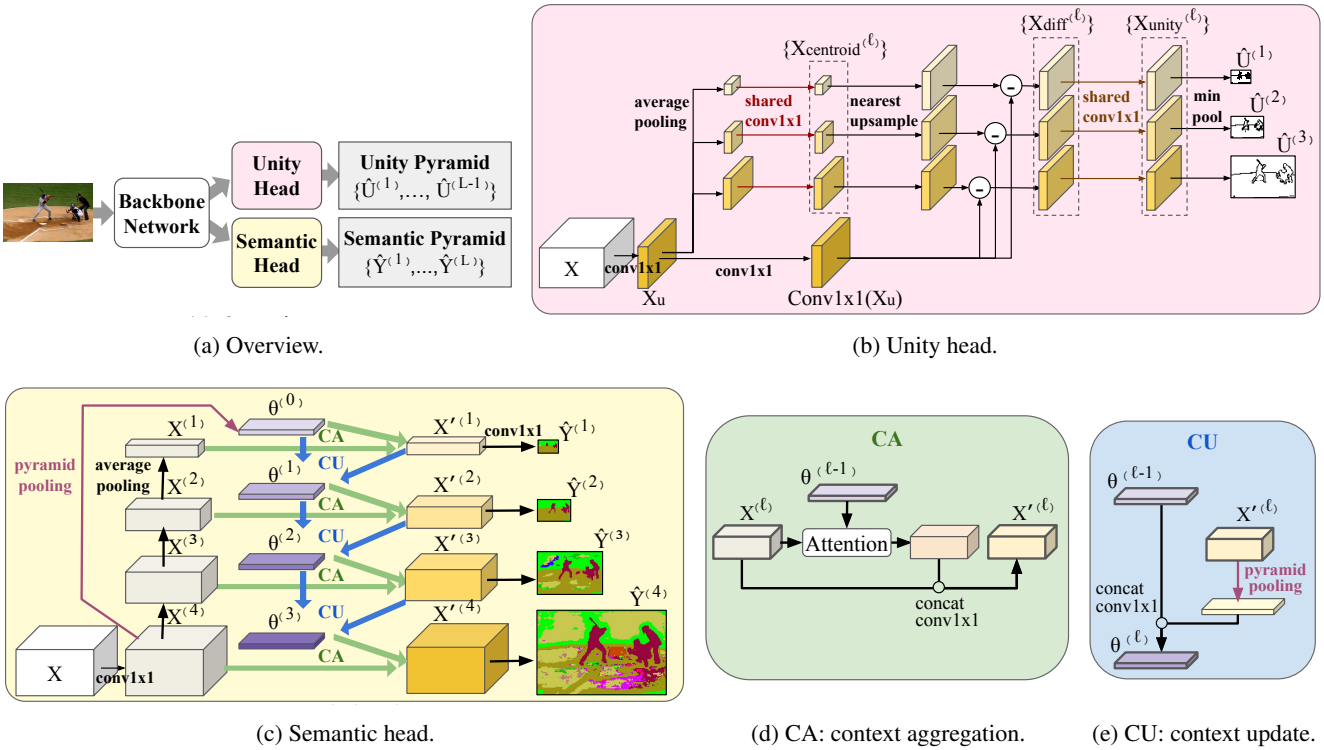
where CE is cross entropy and BCE is binary cross entropy. Note that only $L-1$ levels are predicted in the unity pyramid $\hat{U}^{(\ell)}$ as all cells in the finest level L are assumed to be unity-cells (there is no subsequent finer-level semantic prediction to be considered). We show in the experiments that each level of the semantic pyramid has indeed learned to specialize in characterizing the assigned pixels.

3.3. *Fuse*—the inference phase

During inference, we fuse the two predicted pyramids into one final semantic map $\hat{Y}$. Given the predicted *semantic pyramid* $\hat{Y}^{(\ell)}$ and *unity pyramid* $\hat{U}^{(\ell)}$, the inference procedure refers each pixel to the semantic prediction at the "coarsest" unity-cell as follows.

$$\begin{aligned}\hat{Y} = \sum_{\ell=1}^{L} \Bigg\{ & \mathbb{1}\left[\mathrm{Up}(\hat{U}^{(\ell)}) \geq \tau\right] \odot \mathrm{Up}(\hat{Y}^{(\ell)}) \\ & \odot \bigodot_{1\leq k<\ell} \mathbb{1}\left[\mathrm{Up}(\hat{U}^{(k)}) < \tau\right] \Bigg\},\end{aligned} \tag{2}$$

where $\mathrm{Up}(\cdot)$ upsamples the prediction at level ℓ to the finest level L, τ is a threshold to decide unity-cell (*i.e.*, $\geq \tau$) or mix-cell (*i.e.*, $< \tau$), $\odot$ and $\mathbb{1}[\cdot]$ denote the element-wise multiplication and indicator function, respectively. The 1^{st} line in Eq. 2 selects the semantic labels at unity-cells in level ℓ. To ensure that the "coarsest" unity-cell is selected, the 2^{nd} line in Eq. 2 checks whether all of its preceding cells in levels $1 \sim (\ell-1)$ are mix-cells.



(a) Overview.

(b) Unity head.

(c) Semantic head.

(d) CA: context aggregation.

(e) CU: context update.

Figure 2: An illustration of the neural network architecture. (a) Two additional heads are added to a backbone network for predicting the proposed unity pyramid and semantic pyramid. (b) The unity head. (c) For the semantic head, we design a coarse-to-fine contextual module, which comprises two operations—(d) CA: context aggregation and (e) CU: context update. Note that the $\hat{U}^{(\ell)}$ in (b) and $\hat{Y}^{(\ell)}$ in (c) are raw outputs and will be fused into one to serve as the final prediction (Sec. 3.3).

4. Architectures of pyramid head

The architectures of unity and semantic head are shown in Fig. 2 and detailed in the following sections.

4.1. Unity head

The unity head takes a feature X from the backbone as input and outputs a *unity pyramid* $\hat{U}^{(\ell)}$. The design of unity head (Fig. 2b) follows the idea that the embedding of every pixel within a unity-cell has to be close to the embeddings of the centroid of the cell as they share the same semantic class. First, we employ a 1×1 convolution layer (Conv1x1) to convert X to X_u with reduced channels D_u. Next, we generate centroid embedding of all cells in a pyramid $X^{(\ell)}_{\text{centroid}} \in \mathbb{R}^{D_u \times \frac{H}{s_\ell} \times \frac{W}{s_\ell}}$ by applying average pooling and a shared Conv1x1 to X_u. To measure the difference $X^{(\ell)}_{\text{diff}}$ between the embedding of all pixels within a cell and the centroid embedding of the cell, we upsample each $X^{(\ell)}_{\text{centroid}}$ to the finest level using nearest-neighbor interpolation and subtract it from $\text{Conv1x1}(X_u)$. Subsequently, a shared Conv1x1 followed by a sigmoid function converts $X^{(\ell)}_{\text{diff}} \in \mathbb{R}^{D_u \times \frac{H}{s_L} \times \frac{W}{s_L}}$ to $X^{(\ell)}_{\text{unity}} \in \mathbb{R}^{\frac{H}{s_L} \times \frac{W}{s_L}}$, where each entry yields a probability that it shares the same semantic class with the cell centroid. Reflecting the definition of a unity-cell, we use min pooling to query if the most deviated entry in a cell is similar to the cell centroid, and produce the final unity pyramid $\hat{U}^{(\ell)} \in \mathbb{R}^{\frac{H}{s_\ell} \times \frac{W}{s_\ell}}$.

4.2. Semantic head

Predicting semantic in pyramidal format. We first employ a Conv1x1 layer to project the number of channels from backbone's D to D_s, producing feature $X^{(L)}$. The simplest way to predict the semantic pyramid $\{\hat{Y}^{(\ell)}\}_{\ell=1,\dots,L}$ is directly pooling $X^{(L)}$ to the L desired spatial sizes:

$$X^{(\ell)} = \text{AvgPool}\left(X^{(L)}, 2^{L-\ell}\right), \tag{3}$$

where $\text{AvgPool}(\cdot, k)$ is average pooling with kernel size and stride set to k. Each $X^{(\ell)}$ is then projected from latent dimension D_s to the number of classes C with convolutional layers. We show in our experiment that such a simplest network setting can already achieve promising improvements with the proposed *specialize-and-fuse* strategy.

Coarse-to-fine contextual module. Motivated by the recent success of context spreading strategy for semantic segmentation, we further design a coarse-to-fine contextual

module for our pyramidal output format. Intuitively, we aggregate the contextual information from coarser pyramid levels to assist the prediction at finer pyramid levels. Fig. 2c depicts the proposed coarse-to-fine semantic head, which refines $\{X^{(\ell)}\}_{\ell=1,\dots,L}$ in Eq. (3) with context feature $\{\theta^{(\ell)}\}_{\ell=0,\dots,L-1}$ from coarse to fine level:

$$X'^{(\ell)} = \mathrm{CA}^{(\ell)}\left(X^{(\ell)}, \theta^{(\ell-1)}\right), \tag{4}$$

$$\theta^{(\ell)} = \mathrm{CU}^{(\ell)}\left(X'^{(\ell)}, \theta^{(\ell-1)}\right), \tag{5}$$

where the CA and CU stand for *Context Aggregation module* and *Context Updating module* and will be detailed later. Inspired by ANL's strategy [34] to improve efficiency of contextual module, our initial context feature is

$$\theta^{(0)} = \mathrm{PyramidPool}(X^{(L)}), \tag{6}$$

where $\mathrm{PyramidPool}$ flatten the 1×1, 3×3, 6×6, and 8×8 features generated by spatial pyramid pooling [15].

Iterating ℓ from coarse to fine (from 1 to L), the *Context Aggregation module* refines $X^{(\ell)}$ using $\theta^{(\ell-1)}$ (while $X^{(1)}$ is refined by the initial context feature $\theta^{(0)}$); the *Context Updating module* then updates the context feature $\theta^{(\ell-1)}$ with the refined $X'^{(\ell)}$, forming the new context feature $\theta^{(\ell)}$ which facilitates finer-level semantic prediction by encapsulating the information from the coarsest to the current level.

Once the coarse-to-fine contextual module generates the refined feature pyramid $\{X'^{(\ell)}\}_{\ell=1,\dots,L}$, the semantic pyramid $\{\hat{Y}^{(\ell)}\}_{\ell=1,\dots,L}$ are predicted by

$$\hat{Y}^{(\ell)} = \mathrm{ConvBlock}^{(\ell)}\left(X'^{(\ell)}\right), \tag{7}$$

where $\mathrm{ConvBlock}^{(\ell)}$ consists of $\mathrm{Conv1x1}$, BN, ReLU, and a final $\mathrm{Conv1x1}$ projecting D_s to the number of classes C. In below, we detail the *Context Aggregation module* (CA) and *Context Updating module* (CU).

CA**—Context Aggregation module.** The context aggregation module is illustrated in Fig. 2d. To refine $X^{(\ell)}$, we use attention operation to aggregate coarser-level context encoded in $\theta^{(\ell-1)}$. Specifically, we transform $X^{(\ell)}$ to $X^{(\ell)}_{\text{query}}$, $\theta^{(\ell-1)}$ to $\theta^{(\ell-1)}_{\text{key}}, \theta^{(\ell-1)}_{\text{value}}$ by $\mathrm{Conv1x1}$ layers; then we apply

$$X^{(\ell)}_{\text{att}} = \mathrm{Attention}\left(X^{(\ell)}_{\text{query}}, \theta^{(\ell-1)}_{\text{key}}, \theta^{(\ell-1)}_{\text{value}}\right), \tag{8}$$

$$X'^{(\ell)} = \mathrm{Conv1x1}^{(\ell)}_{\text{agg.}}\left(\mathrm{concat}\left(X^{(\ell)}_{\text{att}}, X^{(\ell)}\right)\right), \tag{9}$$

where the $\mathrm{Attention}$ is the attention operation [22], $\mathrm{Conv1x1}^{(\ell)}_{\text{agg.}}$ consisting of $\mathrm{Conv1x1}, \mathrm{BN}, \mathrm{ReLU}$ projects the concatenated $2D_s$ channels back to D_s.

CU**—Context Updating module.** The context updating module is illustrated in Fig. 2e. To update the context feature $\theta^{(\ell-1)}$ on the refined feature $X'^{(\ell)}$ at level ℓ, we apply

$$\theta^{(\ell)}_{\text{init}} = \mathrm{PyramidPool}\left(X'^{(\ell)}\right), \tag{10}$$

$$\theta^{(\ell)} = \mathrm{Conv1x1}^{(\ell)}_{\text{upd.}}\left(\mathrm{concat}\left(\theta^{(\ell)}_{\text{init}}, \theta^{(\ell-1)}\right)\right), \tag{11}$$

where the $\mathrm{Conv1x1}^{(\ell)}_{\text{upd.}}$ consisting of $\mathrm{Conv1x1}, \mathrm{BN}, \mathrm{ReLU}$ projects the concatenated $2D_s$ channels back to D_s. Note that the new context feature $\theta^{(\ell)}$ remains at the same low spatial resolution as $\theta^{(\ell-1)}$, so the overall coarse-to-fine contextual module runs efficiently.

5. Experiments

We first introduce our implementation details in Sec. 5.1. Then, we report our comparison with state-of-the-art methods on three datasets in Sec. 5.2 and the comparison on computation efficiency in Sec. 5.3. Finally, thorough ablation study and performance analysis are conducted to support the contribution of our designed components in Sec. 5.4 and Sec. 5.5, respectively.

5.1. Implementation detail

5.1.1 Training setting

We mainly follow the training protocol of the public implementation of HRNet-OCR. The SGD optimizer with momentum 0.9 is employed. Data augmentation includes random brightness, random left-right flip, random scaling with factor uniformly sampled from $[0.5, 2.0]$, and finally random crop to a fixed size. The crop size, weight decay, and batch-size are set to $(512 \times 512, 1\text{e}{-4}, 16)$ for all datasets. The base learning rate and the number of epochs are set to $(0.02, 120)$, $(0.001, 110)$, and $(0.001, 200)$ for ADE20K, COCO-Stuff, and Pascal-Context, respectively. The learning rate follows the poly schedule with the power factor set to 0.9.

5.1.2 Backbone setting

We experiment with two backbone networks—HRNet48 [23] and ResNet101 [10]. For simplicity, we ensure both backbones generate features at the same spatial level as the finest pyramid level, *i.e.*, output stride 4 in our experiments.

HRNet48. HRNet [23] provides high-resolution features of output stride 4, so we directly attach our *Unity head* and *Semantic head* to the end of HRNet.

ResNet101. ResNet [10] produces coarse features of output stride 32. To obtain better results, some recent methods [18, 27, 20, 12, 16] employ the dilated version of ResNet with output stride 8. However, we find such modification leads to lower speed and more memory footprint, so we adopt the standard ResNet with a lightweight decoder. We simply

Method	Venue	Backbone	mIoU (%)
CFNet [30]	CVPR2019	ResNet101	44.89
APCNet [9]	CVPR2019	ResNet101	45.38
CCNet [13]	ICCV2019	ResNet101	45.22
ANL [34]	ICCV2019	ResNet101	45.24
ACNet [7]	ICCV2019	ResNet101	45.90
CPNet [26]	CVPR2020	ResNet101	46.27
SPNet [11]	CVPR2020	ResNet101	45.60
QGN [3]	WACV2020	ResNet101	43.91
GFFNet [18]	AAAI2020	ResNet101	45.33
OCR [27]	ECCV2020	ResNet101	45.28
DNL [25]	ECCV2020	ResNet101	45.97
CaCNet [20]	ECCV2020	ResNet101	46.12
ours	-	ResNet101	**47.00**
HRNet [23]	TPAMI2019	HRNet48	44.20
CCNet [13]†	-	HRNet48	45.65
ANL [34]†	-	HRNet48	45.23
OCR [27]	ECCV2020	HRNet48	45.50
DNL [25]	ECCV2020	HRNet48	45.82
ours	-	HRNet48	**47.16**

†Our reproduction by replacing the backbone with HRNet48

Table 1: Comparisons on ADE20K [33] validation set.

Method	Venue	Backbone	Score
PSPNet [32]	CVPR2017	ResNet269	55.38
EncNet [29]	CVPR2018	ResNet101	55.67
ACNet [7]	ICCV2019	ResNet101	55.84
DNL [25]	ECCV2020	ResNet101	56.23
ours	-	ResNet101	**56.67**
DNL [25]	ECCV2020	HRNet48	55.98
ours	-	HRNet48	**58.04**

Table 2: ADE20K [33] official evaluation.

reduce the number of channels of each stage from ResNet to save computation and apply the fusion module [23] as the decoder to form feature of output stride 4. Comparing to the dilated ResNet, our adaptation requires only $0.75\times$ processing time and $0.88\times$ memory footprint (see the supplementary material for details about implementation and computational efficiency). Besides, some recent methods [12, 16] also adopt ASPP [2], which we do not employ, for the ResNet backbone.

5.1.3 *Specialize and Fuse* setting

The backbone features have $D = 720$ channels for both our HRNet48 and ResNet101 experiments, and we set D_u in the unity head to 64 and D_s in the semantic pyramid head to 512. We use the instantiation of $L = 4$ with output strides $\{4, 8, 16, 32\}$ for the semantic pyramid. We set a high threshold $\tau = 0.9$ for the binary classifier in the unity pyramid to suppress false positives for the unity-cell prediction as, intuitively, a false positive always introduces error while a false negative could have the chance to be "remedied" by finer level semantic.

Method	Venue	Backbone	mIoU (%)
SVCNet [5]	CVPR2019	ResNet101	39.6
DANet [6]	CVPR2019	ResNet101	39.7
EMANet [19]	ICCV2019	ResNet101	39.9
ACNet [7]	ICCV2019	ResNet101	40.1
GFFNet [18]	AAAI2020	ResNet101	39.2
OCR [27]	ECCV2020	ResNet101	39.5
CDGCNet [12]	ECCV2020	ResNet101	**40.7**
ours	-	ResNet101	**40.7**
HRNet [23]	TPAMI2019	HRNet48	37.9
CCNet [13]†	-	HRNet48	39.8
ANL [34]†	-	HRNet48	40.6
OCR [27]	ECCV2020	HRNet48	40.6
ours	-	HRNet48	**41.0**

†Our reproduction by replacing the backbone with HRNet48

Table 3: Comparisons on COCO-Stuff [1] test set.

Method	Venue	Backbone	mIoU (%)
CFNet [30]	CVPR2019	ResNet101	54.0
APCNet [9]	CVPR2019	ResNet101	54.7
SVCNet [5]	CVPR2019	ResNet101	53.2
DANet [6]	CVPR2019	ResNet101	52.6
BFP [4]	ICCV2019	ResNet101	53.6
ANL [34]	ICCV2019	ResNet101	52.8
EMANet [19]	ICCV2019	ResNet101	53.1
ACNet [7]	ICCV2019	ResNet101	54.1
DGCNet [31]	BMVC2019	ResNet101	53.7
CPNet [26]	CVPR2020	ResNet101	53.9
SPNet [11]	CVPR2020	ResNet101	54.5
GFFNet [18]	AAAI2020	ResNet101	54.2
OCR [27]	ECCV2020	ResNet101	54.8
DNL [25]	ECCV2020	ResNet101	54.8
CaCNet [20]	ECCV2020	ResNet101	55.4
ours	-	ResNet101	**55.6**
HRNet [23]	TPAMI2019	HRNet48	54.0
OCR [27]	ECCV2020	HRNet48	56.2
DNL [25]	ECCV2020	HRNet48	55.3
ours	-	HRNet48	**57.0**

Table 4: Comparisons on Pascal-Context [21] test set.

5.2. Comparison with state-of-the-arts

Following the literature, we apply multi-scale and left-right flip testing augmentation to report our results.

ADE20K [33]. ADE20k is a dataset with diverse scenes containing 35 stuff and 115 thing classes. The training, validation, and test split contains 20K/2K/3K images, respectively. The results on validation set of ADE20K is shown in Table 1. Our method establishes new state-of-the-art results with both ResNet101 and HRNet48 backbone. In addition, we submit our prediction on hold-out test set to ADE20k's server for official evaluation. The results reported in Table 2 also show our advantage over previous methods.

COCO-Stuff [1]. COCO-Stuff is a challenging dataset with 91 stuff and 80 thing classes. The training and the test sets contain 9K and 1K images, respectively. In Table 3, our approach shows comparable performance to recent state-of-the-art results with ResNet101 backbone and outperforms previous methods with HRNet48 backbone.

Pascal-Context [21]. Pascal-Context is a widely-used dataset for semantic segmentation. It contains 59 classes and one background class, consisting of 4,998 training and 5,105 test images. Results on Pascal-Context test set is presented in Table 4. With both ResNet101 and HRNet48 backbone, our method achieves state-of-the-art results comparing to previous methods with the same backbone.

5.3. Computational efficiency

We compare the testing FPS, training iterations per second, and GPU memory consumption by our in-house implementation in Table B. Our full approach shows similar computational efficiency to the recent efficient variant of self-attention modules [13, 34, 27] for semantic segmentation, meanwhile showing better accuracy (Table 1 and Table 3).

Method	Testing	Training	
	FPS↑	it./sec.↑	Mem.↓
HRNet48 [23]	28	2.6	8.2G
HRNet48 + CCNet [13]	21	1.7	9.9G
HRNet48 + ANL [34]	26	2.2	8.6G
HRNet48 + OCR [27]	24	2.1	9.6G
HRNet48 + ours	24	2.0	8.9G

Table 5: Comparing the model efficiency measured on a GeForce RTX 2080 Ti with image size 512×512. Testing FPS is averaged for processing 50 images. Model training is monitored with a batch size of 4.

5.4. Ablation study

We conduct comprehensive ablation experiments to verify the effectiveness of our proposals. In our ablation experiments, the HRNet32 backbone is employed, and we subsample ADE20K's original training split into 16K/4K for training and validation. We put detailed description and architecture diagram for each experiment in the supplementary material and focus on the comparisons and discussion here.

The effectiveness of the pyramidal output representation. As demonstrated in Table 6, our pyramidal "output" representation (the second row) consistently brings improvement over the standard single-level output (the first row) under various model settings. It gains $+1.65$, $+1.05$ and $+2.31$ mIoU improvement without any contextual module, with ANL module [34] and with the proposed coarse-to-fine contextual module, respectively.

The effectiveness of the contextual module. Our contextual module is designed to accord with the multi-level essence of the proposed pyramidal output. As shown in Table 6, when predicting the standard output (of a single finest level), our coarse-to-fine contextual module and the ANL [34] gain similar improvement over the baseline: $+1.58$ ($40.42 \rightarrow 42.00$) and $+1.60$ ($40.42 \rightarrow 42.02$). However, when predicting the proposed pyramidal output, our coarse-to-fine contextual module achieves a more significant $+2.24$ improvement ($42.07 \rightarrow 44.31$) than appending a single ANL to the backbone $+1.00$ ($42.07 \rightarrow 43.07$). Furthermore, our module also gains more improvement than applying ANL modules at all pyramid level (denoted as ANL-multi) $+1.38$ ($42.07 \rightarrow 43.45$), indicating the effectiveness of our design to aggregate contextual information from coarser pyramid levels.

	Contextual module			
Output format	-	ANL	ANL-multi	ours
Single (standard)	40.42	42.02	-	42.00
Pyramidal (ours)	42.07	43.07	43.45	44.31

Table 6: Ablation study for the main proposals in two aspects—*i)* the pyramidal output representation (the rows) and *ii)* the contextual module (the columns). Mean IoU (%) of each setting is reported for comparison. Here "ANL" is appending a single ANL to the backbone, and "ANL-multi" is applying ANL to each level of our pyramidal "output".

Supervision for specialization of different pyramid output levels. To enforce specialization in training phase, it is important to relabel cells which are "done by coarser" as "don't care". In Table 7, we show the results comparing to other relabeling policies. We can see that naively training all semantic pyramid levels to predict full semantic segmentation map do not achieve any improvement. This is reasonable as it ignores the fact that only a small portion of the cells in finer pyramid level would be activated in testing. A simple fix is explicitly relabeling all descendants of a ground-truth unity-cell as "don't care" during training. By doing so, different semantic pyramid levels now can specialize in the pixels assigned by the oracle. A clear $+1.12$ mIoU improvement is now shown. However, the simple fix still ignores the fact that the unity-cell prediction may produce false negatives where a ground-truth unity-cell is falsely classified as a mix-cell and thus a finer-level semantic prediction is referred in inference phase. As a result, the false negatives might make the train-test distributions inconsistent. Finally, our design of re-labeling the descendants of a true positive unity-cell as "don't care" gains an extra $+0.53$ mIoU improvement.

single output	naive	simple fix	our final
40.42	40.41	41.54	42.07

Table 7: Comparing the relabeling policy in the training procedures for our pyramidal output (the right-most 3 columns). No contextual module is deployed for all experiments.

Does the improvement stem from auxiliary supervision? One may argue that the improvement from our pyramidal output representation stems from the rich supervision at the multi-level rather than our "specialize and fuse" process. To clarify the contribution, we skip the "fuse" phase at inference and use the semantic output at the finest-level as the final output. In this case, the supervision at multi-level are considered as auxiliary supervision. In Table 8, we can see that the auxiliary supervision indeed improves the performance (+0.63). However, the improvement is not comparable to our *specialized and fuse* process (+2.31), which suggests that the main contributor to our superior performance is not multi-level supervision but the *specialized and fuse* strategy.

single output	aux. supervision	specialize & fuse
42.00	42.63 (+0.63)	44.31 (+2.31)

Table 8: How to use coarser-level outputs. The proposed contextual module is deployed for all results.

Number of pyramid levels in our output representation. This work focuses on the setting of L=4 pyramid levels of output strides $\{4, 8, 16, 32\}$ for our pyramidal output representation in all experiments. We also experiment with L=2 levels of output strides $\{4, 32\}$ which yields a slightly worse results (44.31 vs. 44.20 mIoU). Therefore, we stick to the L=4 setting.

5.5. Performance analysis

Does each pyramid level specializes in the selected pixels by unity pyramid? In Table 9, we divide pixels into four groups (the four columns, $\ell \in \{1, 2, 3, 4\}$) according to the assignment from our predicted unity pyramid (Sec. 3.3). For each group, we show the performance of the semantic predictions of the four pyramid levels (the four rows, $\ell' \in \{1, 2, 3, 4\}$). We can see that the mIoU is degraded if a pixel refers to a pyramid level that does not agree with the assignment by the trained unity pyramid ($\ell' \neq \ell$ in each column of Table. 9), indicating that different semantic pyramid levels learn to specialize in predicting the pixels assigned by our predicted unity pyramid.

$\ell' \setminus \ell$	4	3	2	1
4	**33.48**	39.52	42.13	38.61
3	31.63	**41.17**	45.98	44.57
2	27.31	38.13	**46.34**	47.52
1	21.94	28.94	40.05	**48.05**

Table 9: mIoU over a pair of pyramid levels, where ℓ indicates the level of the unity-cell prediction and ℓ' indicates the level of the semantic prediction. When $\ell' = \ell$, the mIoU is higher than other $\ell' \neq \ell$ for every ℓ.

Do different pyramid levels specialize in different classes? To demonstrate our intuition that different pyramid levels have their specializations in different classes, in each row of Fig. 3, we show the per-class IoUs predicted by each semantic level. The results suggest that each $\hat{Y}^{(\ell)}$ learns better for different classes in practice. For instance, $\hat{Y}^{(4)}$ performs better at trafficlight, while $\hat{Y}^{(2)}$ is good at mountain. See supplementary material for more visualizations.

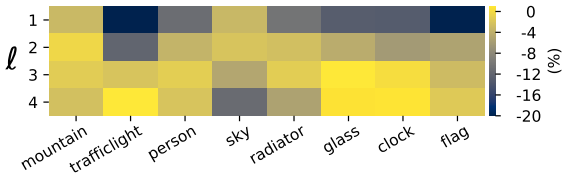


Figure 3: Semantic segmentation performance at each level $\{\hat{Y}^{(\ell)}\}_{\ell=1,\dots,4}$ on different classes. We show the IoU degradation due to using a level $\hat{Y}^{(\ell)}$ versus using the fused $\hat{Y}$.

Qualitative results. We show some qualitative results comparing to the strong baseline, HRNet48-ANL, in Fig. D. See the supplementary material for more examples.

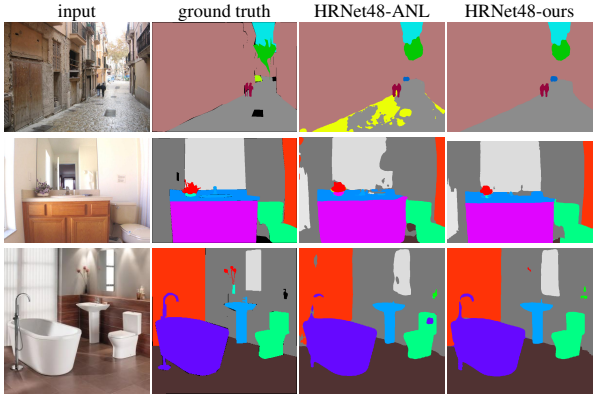


Figure 4: Qualitative results. In the examples, our approach yields better results in the stuff (the 1st row), the large thing (the 2nd row), and the thin part of the object (the 3rd row).

6. Conclusion

We present a novel "output" representation for the task of semantic segmentation. The proposed pyramidal output format and the fusing procedure follow the motivation to assign each pixel to an appropriate pyramid level for better specialization and the parsimony principle. We also present a contextual module, which is efficient and fits the essence of the proposed pyramidal output, improving our performance further. Improvements are shown through extensive experiments. Finally, our performance is on par with or better than the recent state-of-the-art on three widely-used semantic segmentation datasets.

Acknowledgements: This work was supported in part by the MOST, Taiwan under Grants 110-2634-F-001-009 and 110-2634-F-007-016, MOST Joint Research Center for AI Technology and All Vista Healthcare. We thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

References

[1] Holger Caesar, Jasper R. R. Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1209–1218, 2018.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, pages 833–851, 2018.

[3] Kashyap Chitta, José M. Álvarez, and Martial Hebert. Quadtree generating networks: Efficient hierarchical scene parsing with sparse convolutions. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pages 2009–2018, 2020.

[4] Henghui Ding, Xudong Jiang, Ai Qun Liu, Nadia Magnenat-Thalmann, and Gang Wang. Boundary-aware feature propagation for scene segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6818–6828, 2019.

[5] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Semantic correlation promoted shape-variant context for segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8885–8894, 2019.

[6] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3146–3154, 2019.

[7] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for scene parsing. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6747–6756, 2019.

[8] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9224–9232, 2018.

[9] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang, and Yu Qiao. Adaptive pyramid context network for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 7519–7528, 2019.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[11] Qibin Hou, Li Zhang, Ming-Ming Cheng, and Jiashi Feng. Strip pooling: Rethinking spatial pooling for scene parsing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 4002–4011. IEEE, 2020.

[12] Hanzhe Hu, Deyi Ji, Weihao Gan, Shuai Bai, Wei Wu, and Junjie Yan. Class-wise dynamic graph convolution for semantic segmentation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVII*, volume 12362 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2020.

[13] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 603–612, 2019.

[14] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross B. Girshick. Pointrend: Image segmentation as rendering. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9796–9805. IEEE, 2020.

[15] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 2169–2178. IEEE Computer Society, 2006.

[16] Xiangtai Li, Xia Li, Li Zhang, Guangliang Cheng, Jianping Shi, Zhouchen Lin, Shaohua Tan, and Yunhai Tong. Improving semantic segmentation via decoupled body and edge supervision. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVII*, volume 12362 of *Lecture Notes in Computer Science*, pages 435–452. Springer, 2020.

[17] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6459–6468, 2017.

[18] Xiangtai Li, Houlong Zhao, Lei Han, Yunhai Tong, Shaohua Tan, and Kuiyuan Yang. Gated fully fusion for semantic segmentation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11418–11425. AAAI Press, 2020.

[19] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9166–9175, 2019.

[20] Jianbo Liu, Junjun He, Yu Qiao, Jimmy S. Ren, and Hongsheng Li. Learning to predict context-adaptive convolution for semantic segmentation. In Andrea Vedaldi, Horst Bischof,

Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV*, volume 12370 of *Lecture Notes in Computer Science*, pages 769–786. Springer, 2020.

[21] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan L. Yuille. The role of context for object detection and semantic segmentation in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 891–898, 2014.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[23] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2019.

[24] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7794–7803, 2018.

[25] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XV*, volume 12360 of *Lecture Notes in Computer Science*, pages 191–207. Springer, 2020.

[26] Changqian Yu, Jingbo Wang, Changxin Gao, Gang Yu, Chunhua Shen, and Nong Sang. Context prior for scene segmentation. *CoRR*, abs/2004.01547, 2020.

[27] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, volume 12351 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2020.

[28] Fan Zhang, Yanqin Chen, Zhihang Li, Zhibin Hong, Jingtuo Liu, Feifei Ma, Junyu Han, and Errui Ding. Acfnet: Attentional class feature network for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6797–6806, 2019.

[29] Hang Zhang, Kristin J. Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7151–7160, 2018.

[30] Hang Zhang, Han Zhang, Chenguang Wang, and Junyuan Xie. Co-occurrent features in semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 548–557, 2019.

[31] Li Zhang, Xiangtai Li, Anurag Arnab, Kuiyuan Yang, Yunhai Tong, and Philip H. S. Torr. Dual graph convolutional network for semantic segmentation. In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, page 254. BMVA Press, 2019.

[32] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6230–6239, 2017.

[33] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5122–5130, 2017.

[34] Zhen Zhu, Mengdu Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 593–602, 2019.

A. Visualization of the intermediate results

We show the intermediate results of our approach and illustrate how they are fused in Fig. A.

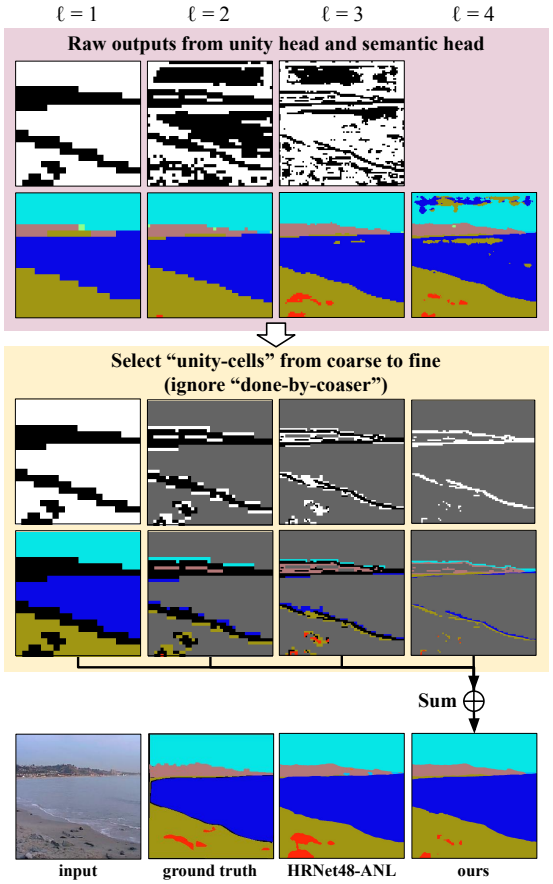


Figure A: An example from the ADE20K validation set showing the intermediate results. **Top**: The raw semantic pyramid outputs and raw unity pyramid outputs. **Middle**: From the coarsest output to the finest output, we select semantic labels only from "unity-cells" but ignore the "done-by-coarser" regions (colored in grey). **Bottom**: Input image, ground truth, HRNet48-ANL prediction, and our final fused prediction.

B. Pyramidal ground truth

Here, we detail how to generate the pyramidal ground truth from standard per-pixel semantic labels when there are "don't care" annotations in the original dataset. For a cell covering $s_\ell \times s_\ell$ pixels, we consider 3 cases for different treatments.

- **All pixels share a semantic class.** In this case, the ground-truth label for the unity pyramid is "unity-cell" (*i.e.*, positive), and the label for the semantic pyramid is the shared class.
- **More than one semantic classes appear.** In this case, the label for the unity pyramid is "mix-cell" (*i.e.*, negative), and the label for the semantic pyramid is "don't care" because it is ambiguous to use one semantic label to represent all underlying pixels of different semantic classes and thus a finer semantic prediction should be referred to.
- **One semantic class and "don't care" appear.** Both the unity and the semantic ground truth are defined as "don't care" as it is ambiguous to determine whether the cell is "unity-cell" or "mix-cell". During the training re-labeling procedure, such a cell is never regarded as true positive, and thus its children cells in the next finer level would never be re-labeled as "done by coarser".

C. Detailed model settings in ablation study

We reorganize the ablation experiment results presented in the main paper into a unified view in Table A and illustrate their network architectures in Fig. B. We label each experiment with an ID (A∼K) and describe the detailed architecture setting below. We call the layers projecting latent dimension D_s to number of classes C as *final projection layer*, which is implemented as Conv1×1 → BN → ReLU → Conv1×1.

- *A* (40.42 %) directly appends the *final projection layer* to $X^{(4)}$.
- *B* (42.02 %) refines $X^{(4)}$ with ANL's APNB block.
- *C* (42.00 %) is our coarse-to-fine contextual module, but only a single finest level semantic prediction is outputted.
- *F* (42.07 %) performs average-pooling on $X^{(4)}$ to get features $X^{(1)}, X^{(2)}, X^{(3)}$ of desired spatial scales, and each of $X^{(1)}, \ldots, X^{(4)}$ has its own *final projection layer*.
- *D* (40.41 %): is a variant of *F* where the training ground-truth is the raw pyramidal ground truth $Y^{(1)}, \ldots, Y^{(4)}$ and $U^{(1)}, \ldots, U^{(3)}$ without the re-labeling procedure to encourage specialization in each pyramid level.
- *E* (41.54 %) is a variant of *F* where the ground-truth unity-cell, instead of the true positive unity-cell, is used for "don't care" re-labeling.
- *G* (43.07 %) refines $X^{(4)}$ of *F* by appending ANL's APNB block to $X^{(4)}$.
- *H* (43.45 %) is similar to *G* but appends APNB block to each of $X'^{(1)}, X'^{(2)}, X'^{(3)}, X'^{(4)}$.

ID	mIoU (%)	Contextual module	Output format	Training procedure
A	40.42	-	single (standard)	-
B	42.02	ANL	single (standard)	-
C	42.00	ours	single (standard)	-
D	40.41	-	pyramidal (ours) {4,8,16,32}	naive
E	41.54	-	pyramidal (ours) {4,8,16,32}	simple fix
F	42.07	-	pyramidal (ours) {4,8,16,32}	our final
G	43.07	ANL	pyramidal (ours) {4,8,16,32}	our final
H	43.45	ANL-multi	pyramidal (ours) {4,8,16,32}	our final
I	**44.31**	ours	pyramidal (ours) {4,8,16,32}	our final
J	42.63	ours	single (standard)	{4,8,16} only for aux.
K	44.20	ours	pyramidal (ours) {4,32}	our final

Table A: The ablation experiment results reorganized in a unified view.

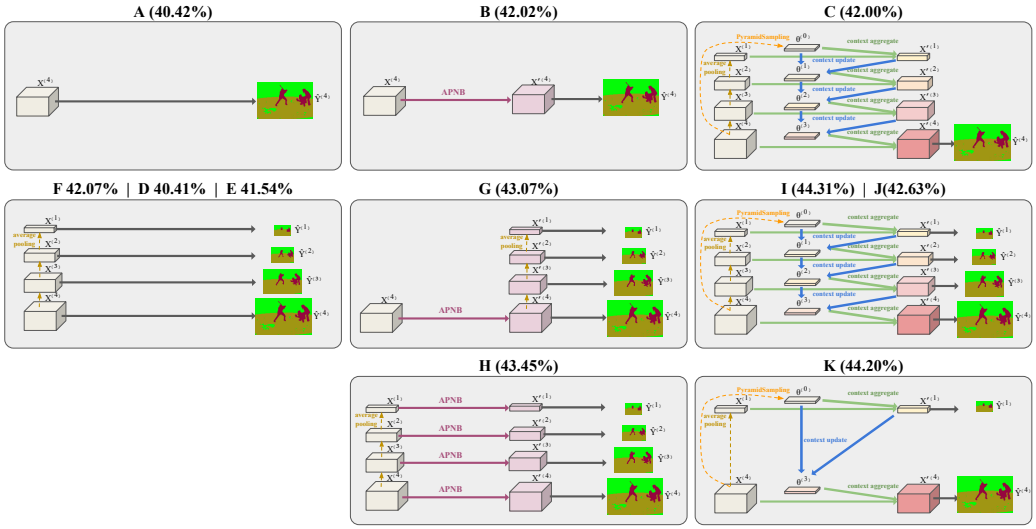


Figure B: The network architectures of the ablation experiments listed in Table A.

- *I* (44.31 %) is the final version of the proposed **Specialize and Fuse**.
- *J* (42.63 %) is similar to *I*, but the non-finest level predictions $\hat{Y}^{(1)}, \ldots, \hat{Y}^{(3)}$ are only used for auxiliary loss (the loss weight is set to 0.4) in the training phase and discarded in the inference phase.
- *K* (44.20 %) is similar to *I* but with less pyramid levels.

D. Architecture of our ResNet-decoder

The standard ResNet produces coarse features of output stride 32. To obtain better results, recent state-of-the-art methods employ the dilated version of ResNet, which generates features of output stride 8. However, we find such a modification leads to a lower speed and more memory footprint as shown in Table B (the 2^{nd} row), so we adopt the standard ResNet with a lightweight decoder instead. Specifically, we use the features from the four stages of a standard ResNet, which respectively have output strides $4, 8, 16, 32$, and $256, 512, 1024, 2048$ latent channels. To reduce the computational cost, the numbers of latent channels are decreased from $[256, 512, 1024, 2048]$ to $[48, 96, 192, 384]$ with $\mathrm{Conv3x3}$ layers. We then fuse the four features by applying the last stage of HRNet, which produces a high-resolution feature for the following semantic segmentation model head. The comparison of computational efficiency between ResNet variants is shown in Table B. Note that the constructed ResNet-decoder runs faster and consumes less GPU memory than ResNet101-dilated-os8. Moreover, the overall computational efficiency of adding our *specialize and fuse* head upon the ResNet-decoder is still better than that of ResNet101-dilated-os8 (which is the optimal bound for many recent ResNet-based state-of-the-art methods).

Method	Testing FPS ↑	Training Memory ↓
ResNet101	82	4.5G
ResNet101-dilated-os8 †	24	9.6G
ResNet101-decoder	32	8.4G
ResNet101-decoder + our head	26	8.9G

†Optimal bound for many recent works built upon dilated ResNet

Table B: Comparing the model efficiency measured on a GeForce RTX 2080 Ti with image size 512×512. FPS is averaged for processing 50 images. GPU memory consumption in training is monitored with a batch size of 4.

E. Do different pyramid levels specialize in different classes?

To demonstrate our intuition that different pyramid levels have their specializations in different classes, we show the per-class IoUs predicted by each level of the semantic pyramid for the 150 classes in the ADE20K dataset in Fig. C. For clearer visualization, we show the IoU difference between the prediction of a single level $\hat{Y}^{(\ell)}$ and the final fused $\hat{Y}$ instead of the original IoU of $\hat{Y}^{(\ell)}$, and we clip the values in the heatmaps to the range from -15% to 1%. A few entries in the heatmaps are larger than 0, which means that $\hat{Y}^{(\ell)}$ performs better than the fused $\hat{Y}$ on that class; it also implies that improving the performance of unity pyramid would provide opportunities for further enhancement. In Fig. C, the coarser pyramid levels (*e.g.* $\ell = 1$) generally look inferior to the finer levels, which could be caused by the fact that the single-level evaluation setting disadvantage the coarser levels since the per-class IoUs for each level are evaluated on the same per-pixel scale. However, this does not conflict with our intention to demonstrate that each pyramid level specializes in different classes. In addition, the finest level performs significantly worst for some classes (*e.g.*, sky, water, tower) since it is not trained to predict the central parts of these large-area classes. Some visual results presented in Fig. A provide more cues about this.

F. Qualitative results

In Fig. D, we show some visual comparisons of results between our approach and a baseline method—ANL with HRNet48 backbone.

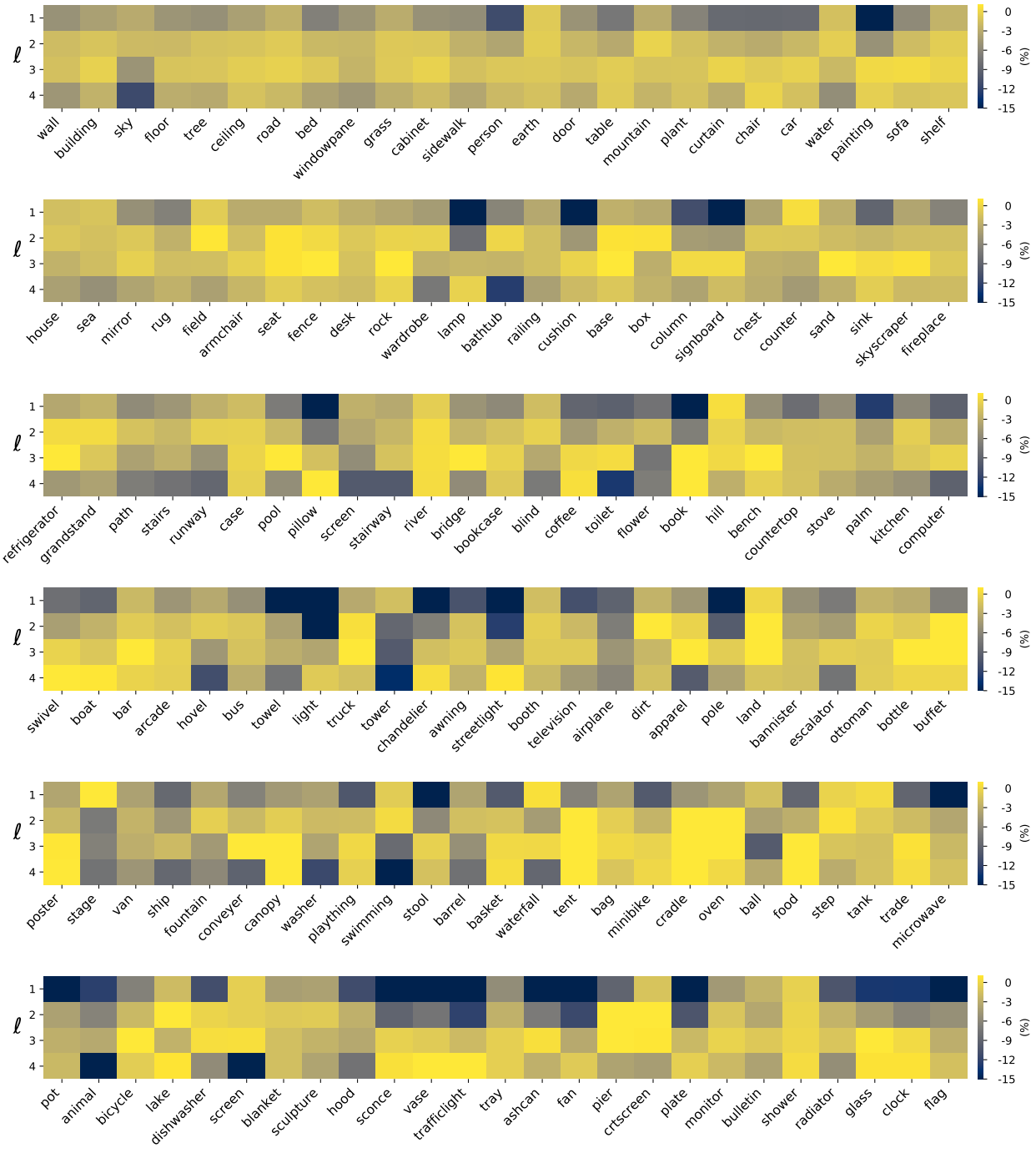


Figure C: Semantic segmentation performance at each level $\{\hat{Y}^{(\ell)}\}_{\ell=1,\ldots,4}$ on different classes. We show the IoU difference between using prediction of a level $\hat{Y}^{(\ell)}$ and using the fused prediction $\hat{Y}$. A brighter cell indicates that the pyramid level is more specialized in the class.

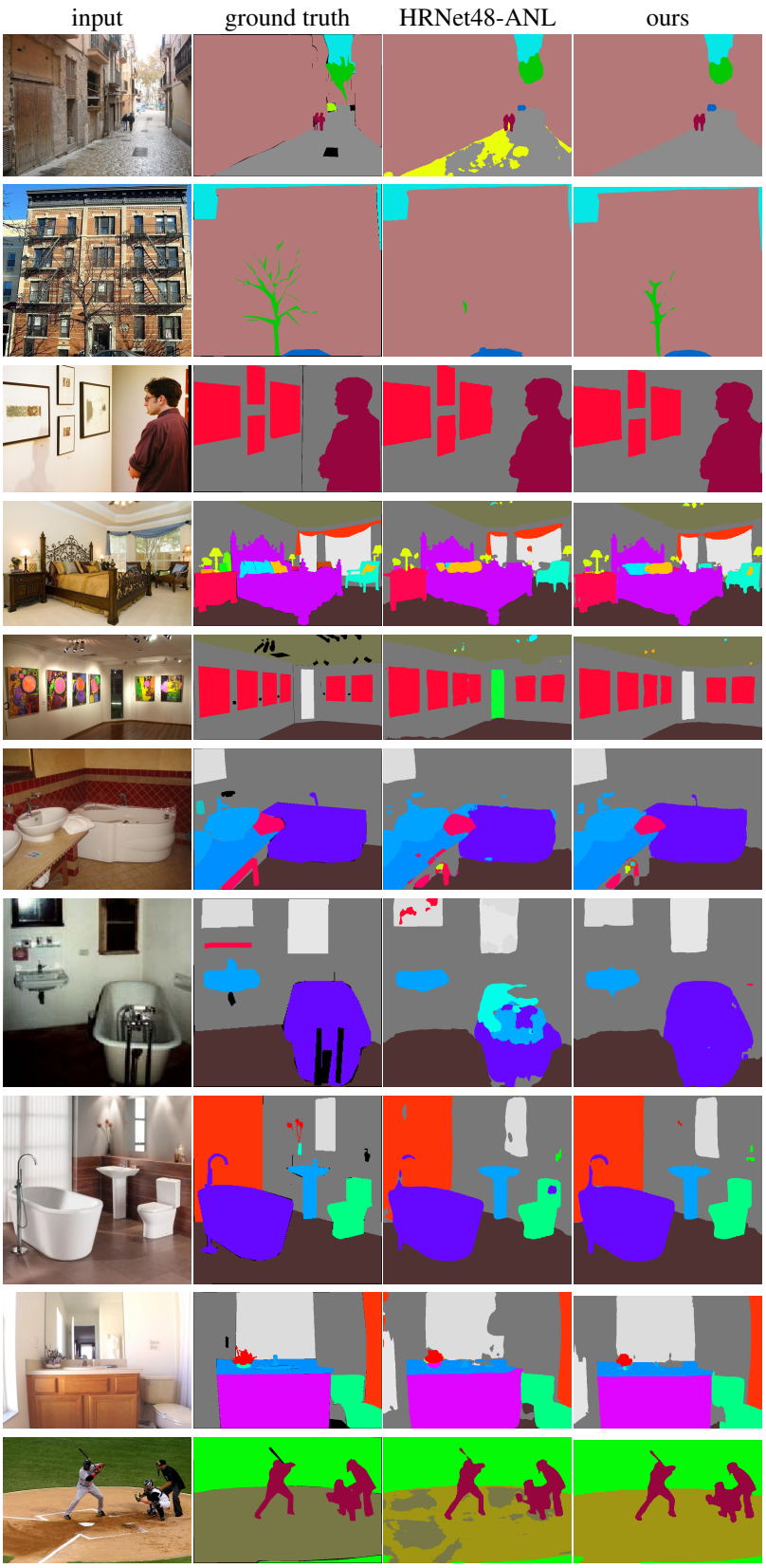


Figure D: Qualitative comparison. In the above examples, predictions by our approach are more consistent within an instance and also provide finer details.