- Learning to Sample
- SampleNet: Differentiable Point Cloud Sampling
- Deep High-Resolution Representation Learning for Visual Recognition
- BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation

Learning to Sample

Oren Dovrat*
Tel-Aviv University
Oren.Dovrat@gmail.com

Itai Lang*
Tel-Aviv University
Itai.Lang83@gmail.com

Shai Avidan
Tel-Aviv University
avidan@eng.tau.ac.il

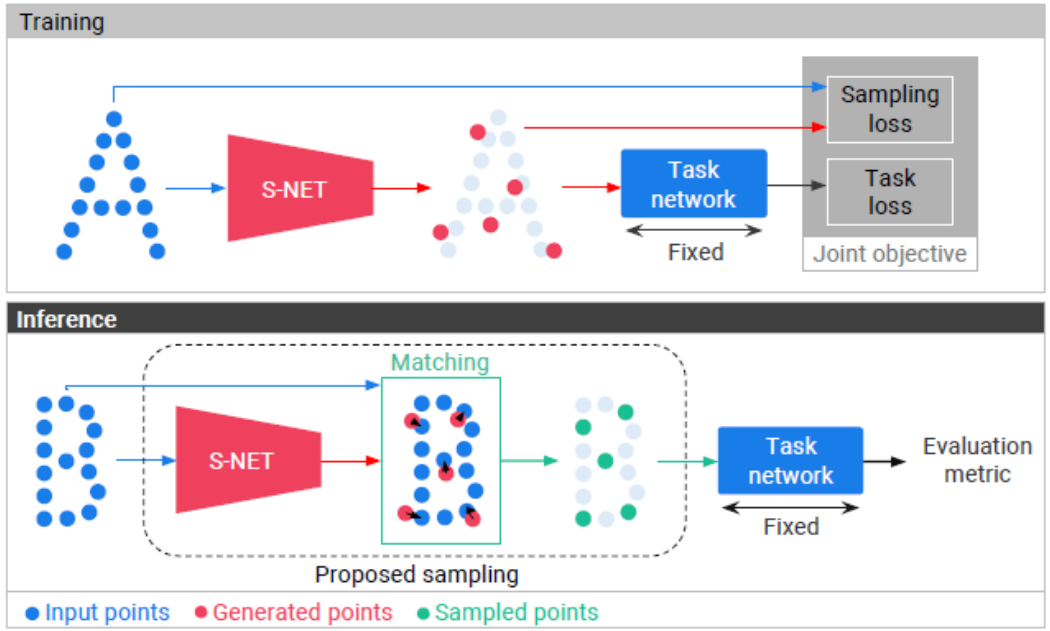


Figure 1. **An illustration of the proposed learned sampling approach.** In the *training* phase, S-NET generates points that are passed to a task network, which was pre-trained and is held fixed. The minimization objective contains the task's loss and a sampling loss. The latter serves as a regularizer and encourages proximity between the input and generated points. At *inference* time, we match the points generated by S-NET with the input point cloud and get a subset of it. Only these points are then fed to the task network for performance evaluation.

Let us denote the generated point set as G and the input point set as P. We construct a sampling regularization loss, composed out of three terms:

$$L_f(G,P) = \frac{1}{|G|}\sum_{g\in G}\min_{p\in P}||g-p||_2^2 \tag{2}$$

$$L_m(G,P) = \max_{g\in G}\min_{p\in P}||g-p||_2^2 \tag{3}$$

$$L_b(G,P) = \frac{1}{|P|}\sum_{p\in P}\min_{g\in G}||p-g||_2^2. \tag{4}$$

L_f and L_m keeps the points in G close to those in P, in the average and worst case, respectively. This is designed to encourage tight matches in the following matching process. We found that mixing average and maximum operations speeds up convergence. L_b ensures that the generated points are well spread over the input points, decreasing the number of collisions in the matching process. The sampling regularization loss is a weighted sum of these three terms:

$$\begin{aligned} L_s(G,P) = L_f(G,P) + \beta L_m(G,P) \\ +(\gamma+\delta|G|)L_b(G,P). \end{aligned} \tag{5}$$

Note that this is a generalization of the Chamfer distance [40], achieved when $\beta=0$, $\gamma=1$ and $\delta=0$.

In addition, we denote L_{task} as the task network loss. The total S-NET loss is:

$$L^{S-NET}(G,P) = L_{task}(G) + \alpha L_s(G,P) \tag{6}$$

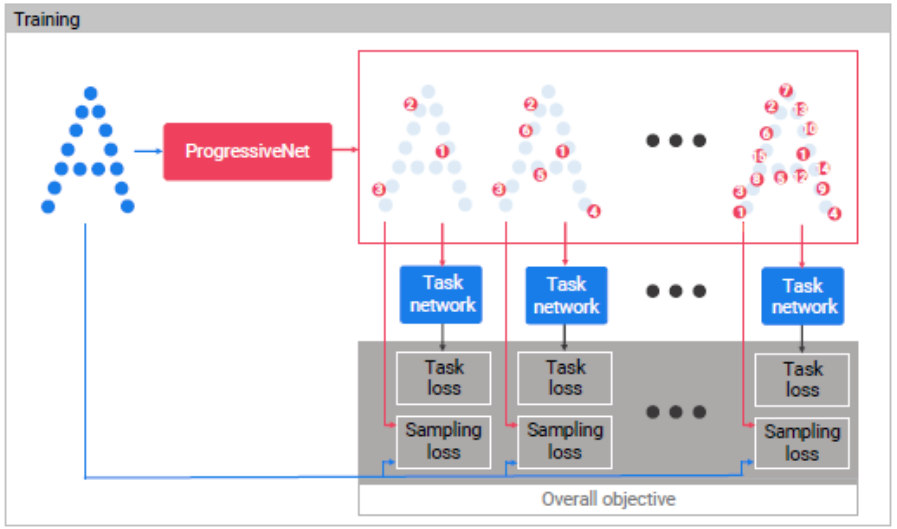


Figure 2. **ProgressiveNet training.** The generated points are divided into groups of increasing size, such that each group is a subset of the following larger group. Each group has corresponding task and sampling losses. The overall objective is the sum of the per-group losses. The task network was pre-trained and is kept fixed.

To train ProgressiveNet (Figure 2) we define a set of sizes $C_s = \{2^1, 2^2, ..., 2^{log_2(n)}\}$. For each size $c \in C_s$ we compute a task loss term and a sampling regularization loss term, such that the total ProgressiveNet's loss becomes:

$$L^{ProgressiveNet}(G, P) = \sum_{c \in C_s} L^{S-NET}(G_c, P)) \quad (8)$$

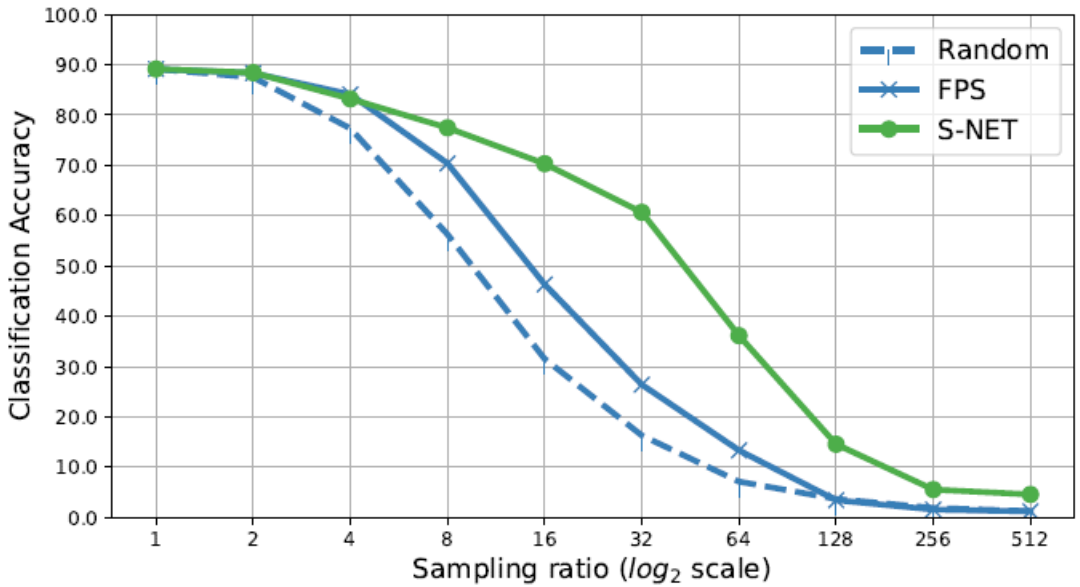


Figure 3. **S-NET for classification.** PointNet was trained on complete point clouds (1024 points) and evaluated on sampled point clouds of the test set using different sampling methods: random, FPS, and S-NET. The accuracy using S-NET is evidently higher.

SampleNet: Differentiable Point Cloud Sampling

Itai Lang
Tel Aviv University
itailang@mail.tau.ac.il

Asaf Manor
Tel Aviv University
asafmanor@mail.tau.ac.il

Shai Avidan
Tel Aviv University
avidan@eng.tau.ac.il

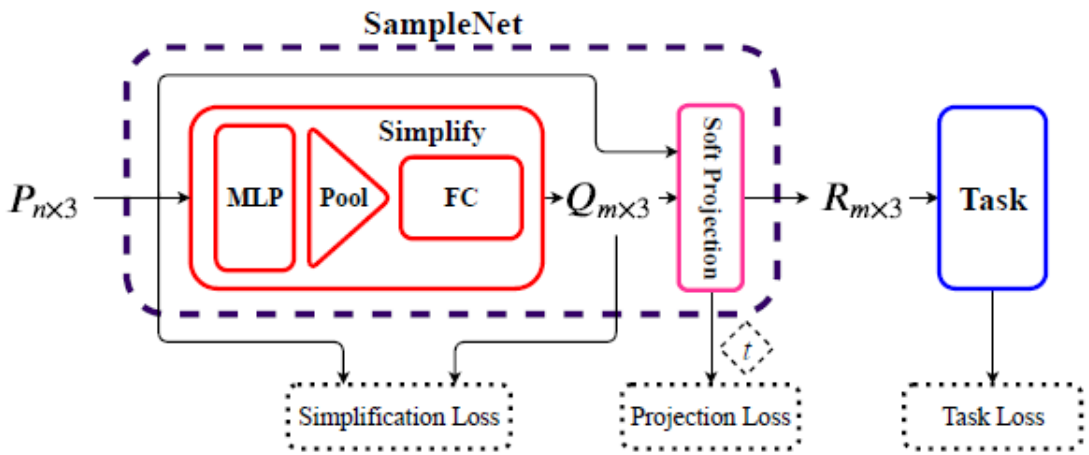


Figure 3. **Training of the proposed sampling method.** The task network trained on complete input point clouds P and kept fixed during the training of our sampling network SampleNet. P is simplified with a neural network to a smaller set Q. Then, Q is softly projected onto P to obtain R, and R is fed to the task network. Subject to the denoted losses, SampleNet is trained to sample points from P that are optimal for the task at hand.

3.2. Project

Instead of optimizing the simplified point cloud for the task, we add the *soft projection* operation. The operation is depicted in Figure 4. Each point $\mathbf{q} \in Q$ is softly projected onto its neighborhood, defined by its k nearest neighbors in the complete point cloud P, to obtain a projected point $\mathbf{r} \in R$. The point $\mathbf{r}$ is a weighted average of original points form P:

$$\mathbf{r} = \sum_{i \in \mathcal{N}_P(\mathbf{q})} w_i \mathbf{p}_i, \tag{8}$$

where $\mathcal{N}_P(\mathbf{q})$ contains the indices of the k nearest neighbors of $\mathbf{q}$ in P. The weights $\{w_i\}$ are determined according to the distance between $\mathbf{q}$ and its neighbors, scaled by a learnable temperature coefficient t:

$$w_i = \frac{e^{-d_i^2/t^2}}{\sum_{j \in \mathcal{N}_P(\mathbf{q})} e^{-d_j^2/t^2}}, \tag{9}$$

The distance is given by $d_i = ||\mathbf{q} - \mathbf{p}_i||_2$.

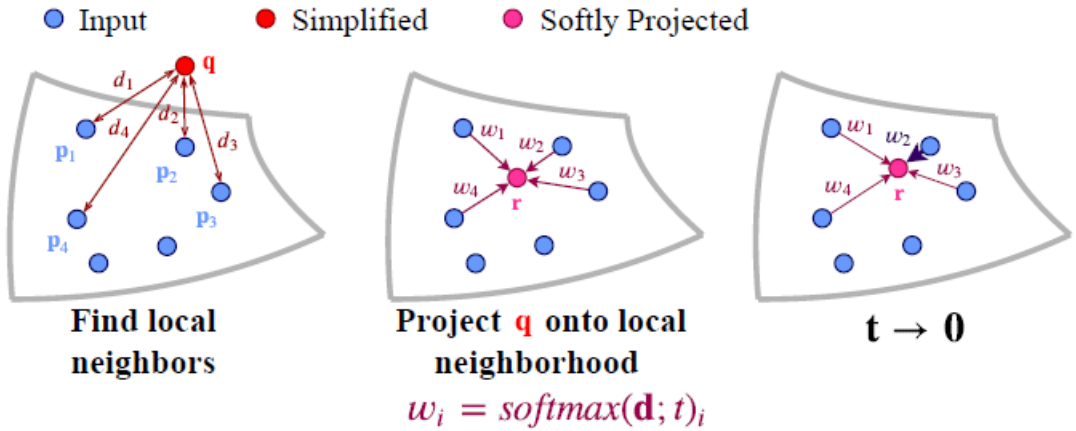


Figure 2. **Illustration of the sampling approximation.** We propose a learned sampling approach for point clouds that employs a differentiable relaxation to nearest neighbor selection. A query point $\mathbf{q}$ (in Red) is projected onto its local neighborhood from the input point cloud (in Blue). A weighted average of the neighbors form a softly projected point $\mathbf{r}$ (in Magenta). During training the weights are optimized to approximated nearest neighbor sampling ($\mathbf{p}_2$ in this example), which occurs at inference time.

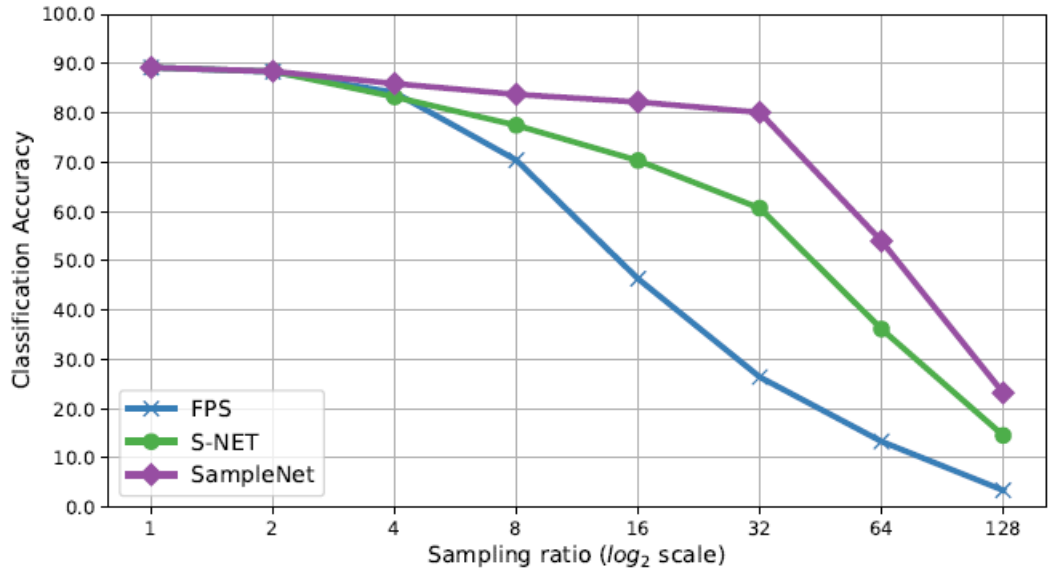


Figure 5. **Classification accuracy with SampleNet.** PointNet is used as the task network and was pre-trained on complete point clouds with 1024 points. The instance classification accuracy is evaluated on sampled point clouds from the test split of ModelNet40. Our sampling method SampleNet outperforms the other sampling alternatives with a large gap.

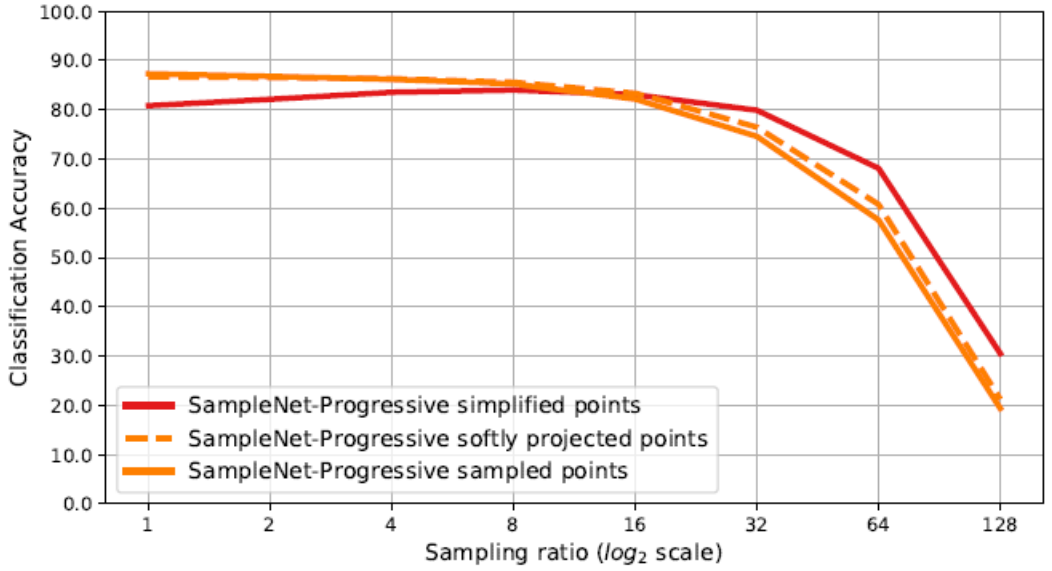


Figure 6. **Classification accuracy with simplified, softly projected, and sampled points.** The instance classification accuracy over the test set of ModelNet40 is measured with simplified, softly projected, and sampled points of SampleNet-Progressive. The accuracy with simplified points is either lower (up to ratio 16) or higher (from ratio 16) than that of the sampled points. On the contrary, the softly projected points closely approximate the accuracy achieved by the sampled points.

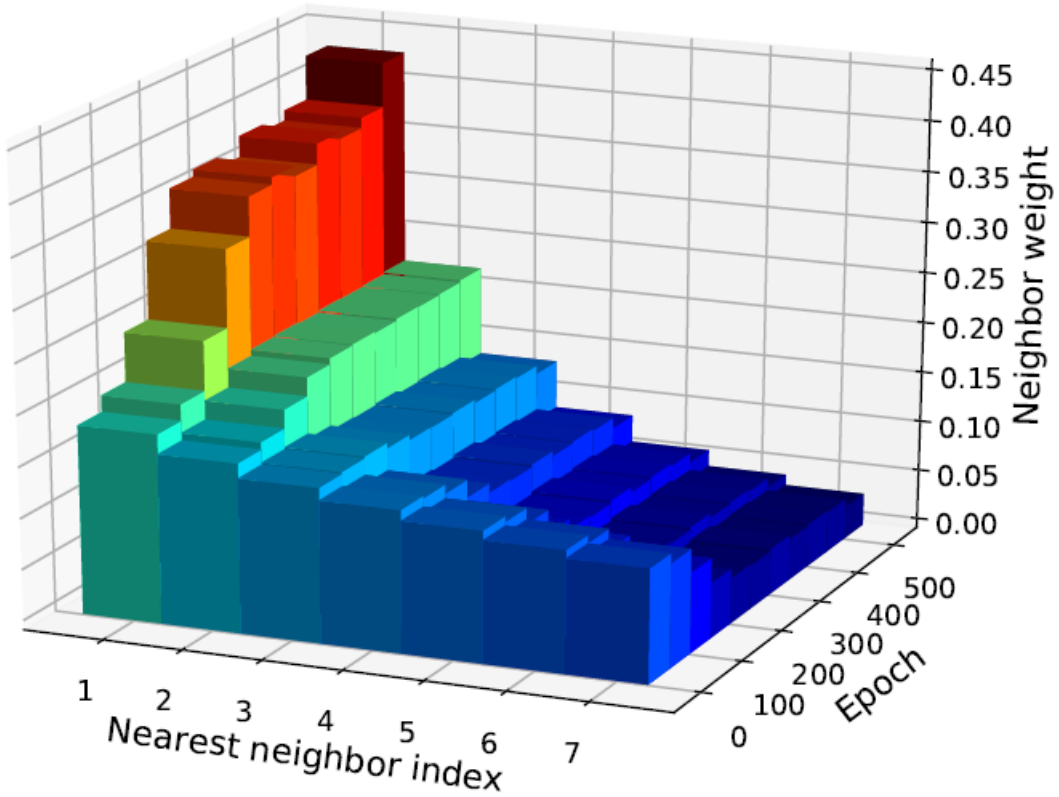
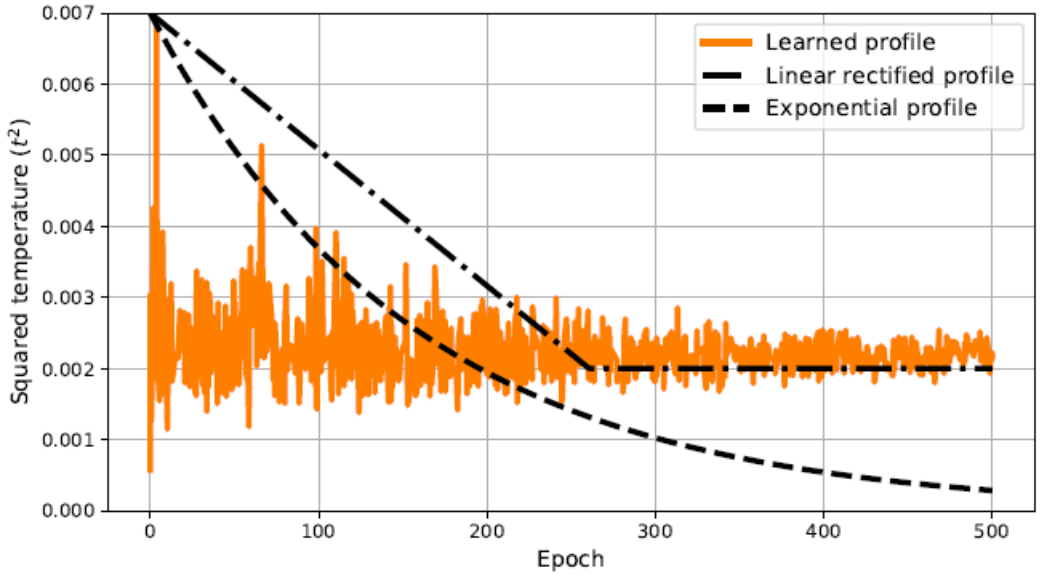


Figure 7. **Evolution of the soft projection weights.** SampleNet is trained to sample 64 points. During training, it is applied to the test split of ModelNet40. The soft projection weights are computed with $k = 7$ neighbors (Equation 9) and averaged over all the examples of the test set. Higher bar with warmer color represents higher weight. As the training progresses, the weight distribution becomes more centered at the close neighbors.



SR	2	4	8	16	32	64	128
FPS	85.6	81.2	68.1	49.4	29.7	16.3	8.6
Con	85.5	75.8	49.6	32.7	17.1	7.0	4.7
Lin	86.7	86.0	85.0	**83.1**	73.7	50.9	20.5
Exp	86.6	85.9	**85.6**	82.0	74.2	55.6	**21.4**
Lrn	**86.8**	**86.2**	85.3	82.2	**74.6**	**57.6**	19.4

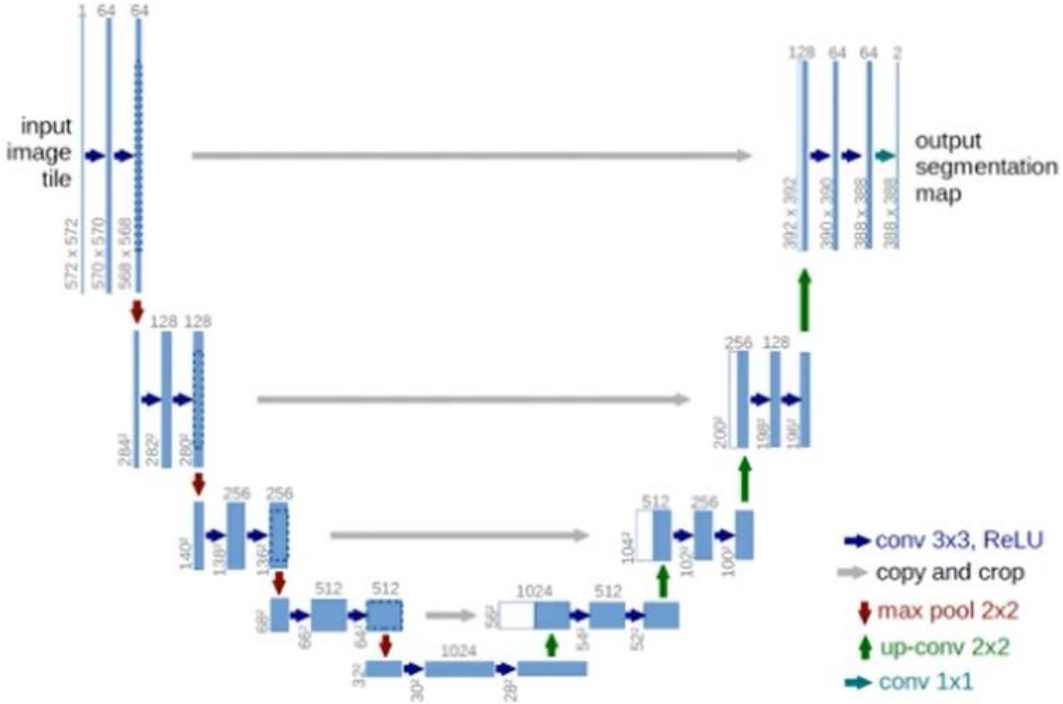
Table 1. **Classification accuracy with different temperature profiles.** SR stands for sampling ratio. Third to last rows correspond to SampleNet-Progressive trained with constant (Con), linear rectified (Lin), exponential (Exp), and learned (Lrn) temperature profile, respectively. SampleNet-Progressive is robust to the decay behavior of the profile. However, if the temperature remains constant, the classification accuracy degrades substantially.

Deep High-Resolution Representation Learning for Visual Recognition

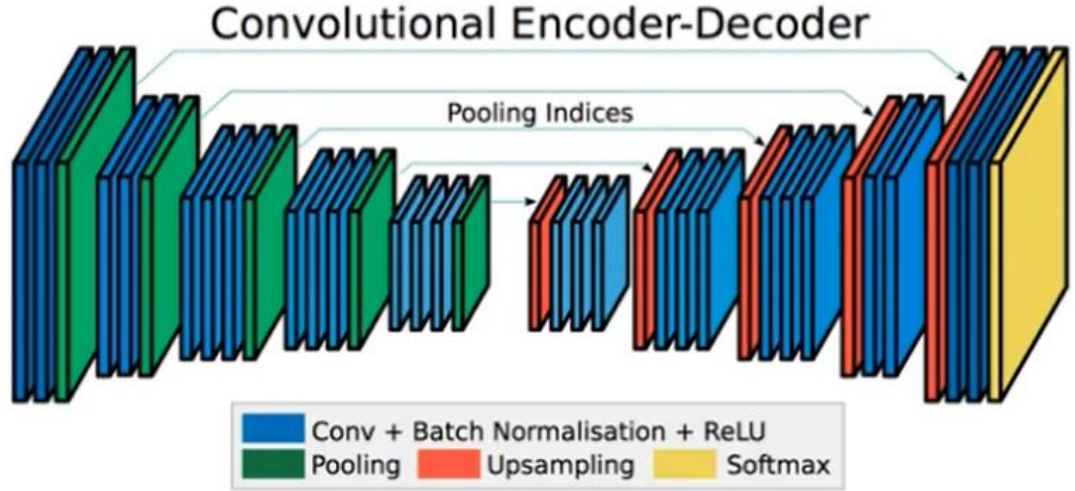
Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao

Abstract—High-resolution representations are essential for position-sensitive vision problems, such as human pose estimation, semantic segmentation, and object detection. Existing state-of-the-art frameworks first encode the input image as a low-resolution representation through a subnetwork that is formed by connecting high-to-low resolution convolutions *in series* (e.g., ResNet, VGGNet), and then recover the high-resolution representation from the encoded low-resolution representation. Instead, our proposed network, named as High-Resolution Network (HRNet), maintains high-resolution representations through the whole process. There are two key characteristics: (i) Connect the high-to-low resolution convolution streams *in parallel*; (ii) Repeatedly exchange the information across resolutions. The benefit is that the resulting representation is semantically richer and spatially more precise. We show the superiority of the proposed HRNet in a wide range of applications, including human pose estimation, semantic segmentation, and object detection, suggesting that the HRNet is a stronger backbone for computer vision problems. All the codes are available at https://github.com/HRNet.

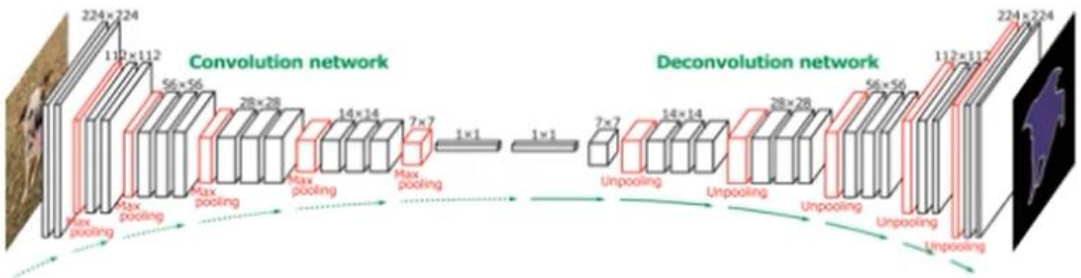
Index Terms—HRNet, high-resolution representations, low-resolution representations, human pose estimation, semantic segmentation, object detection.



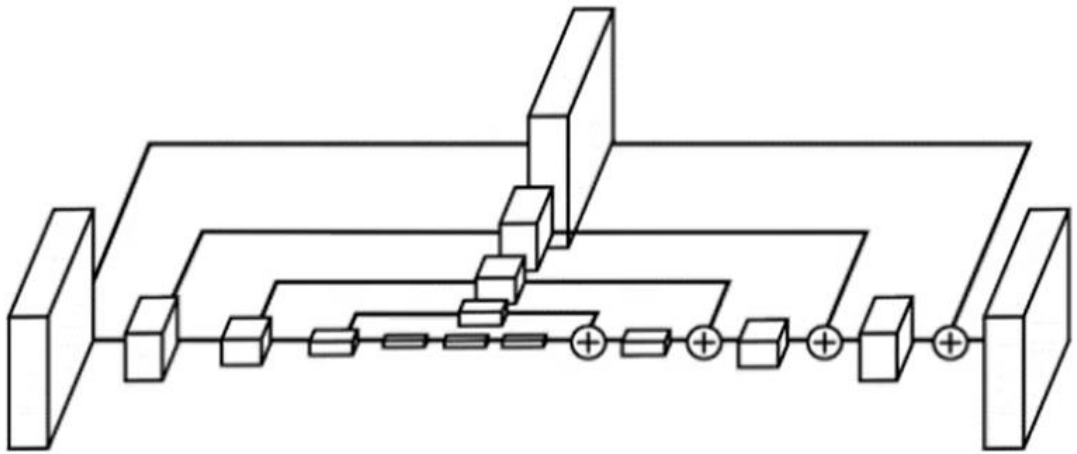
U-Net



SegNet



DeconvNet

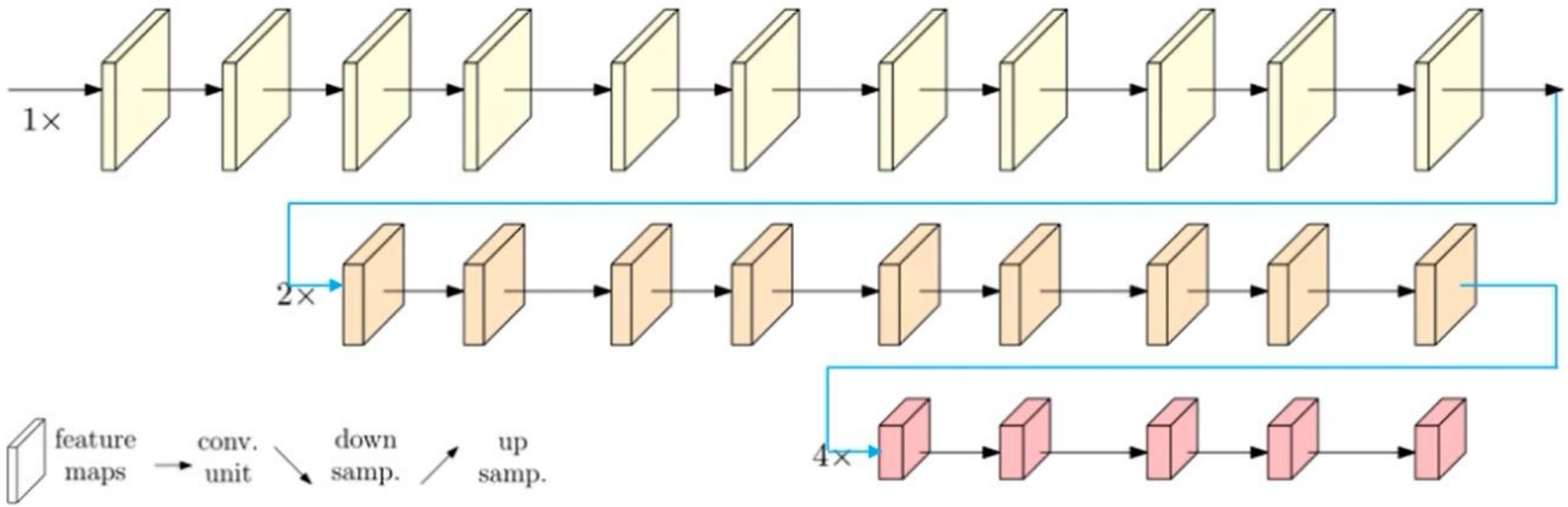


Hourglass

Look different, essentially the same

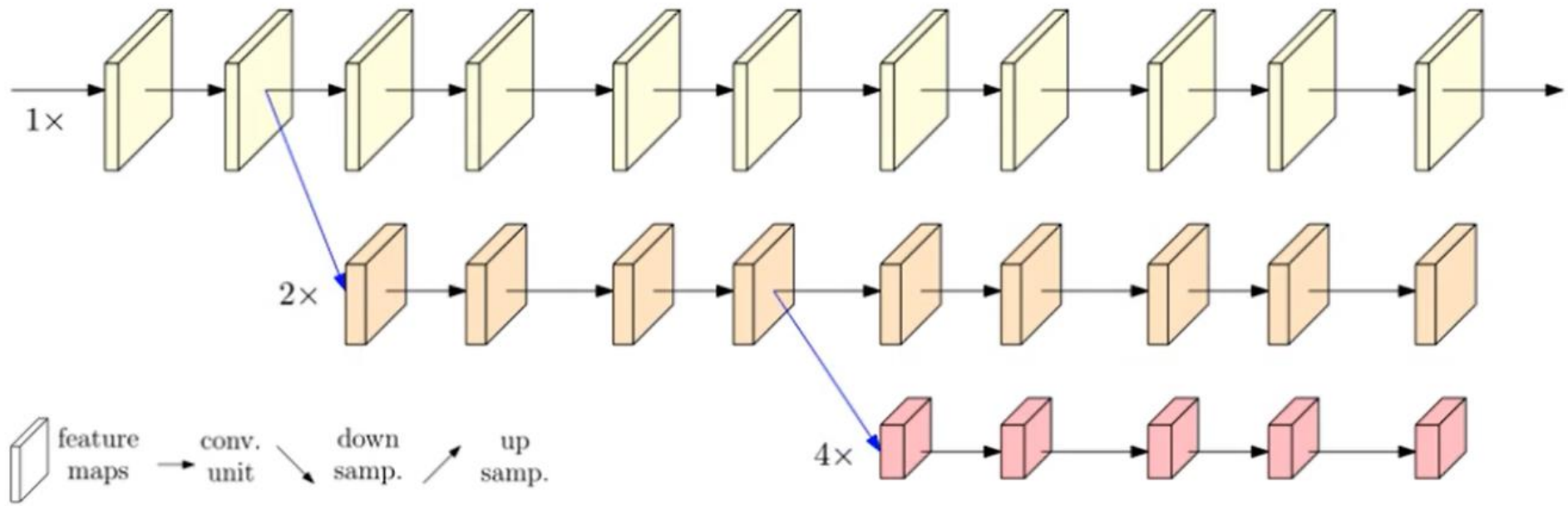
Previous classification networks

Connect multi-resolution convolutions in *series*



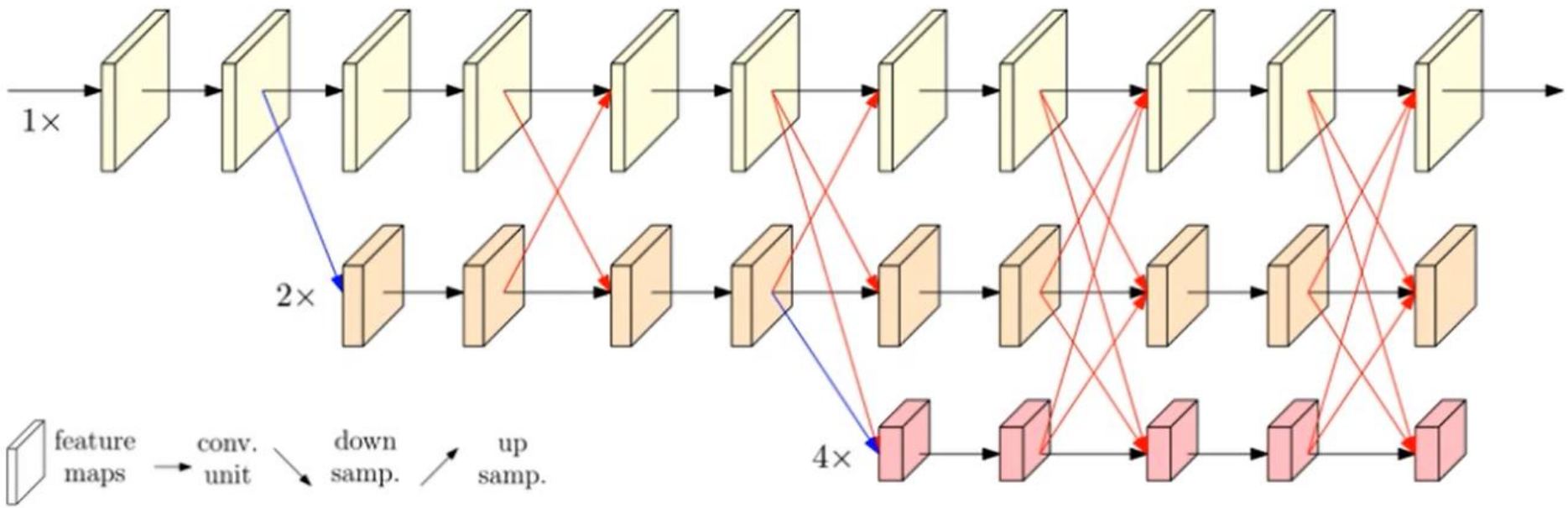
HRNet: high-resolution representation learning

High-resolution networks (HRNet): Connect multi-resolution convolutions in ***parallel***

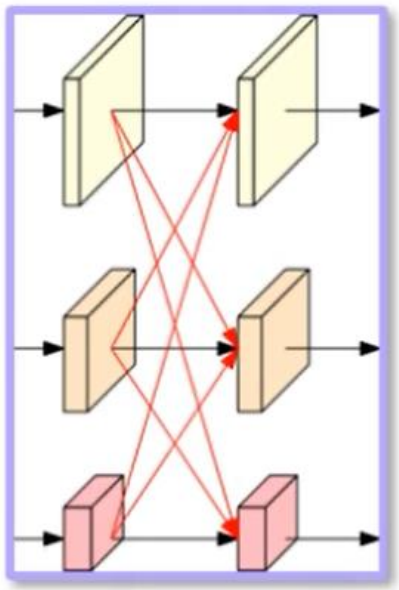


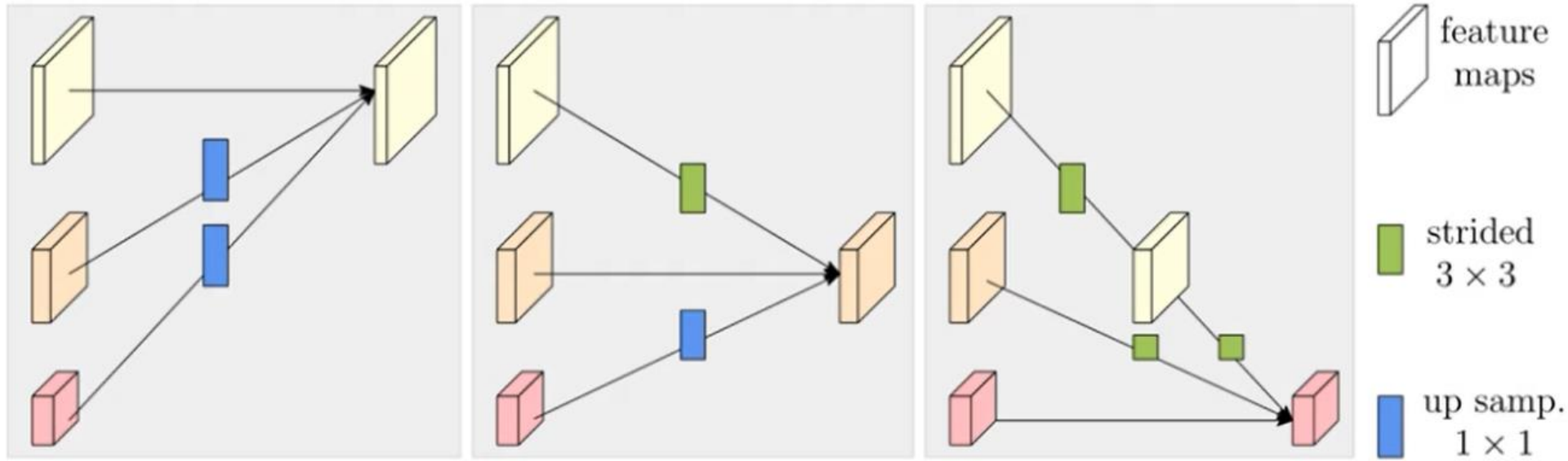
HRNet: high-resolution representation learning

High-resolution networks (HRNet): Connect multi-resolution convolutions in *parallel* with *repeated fusions*



Across-resolution fusion





Down-sample: **stride** − 2 3 × 3

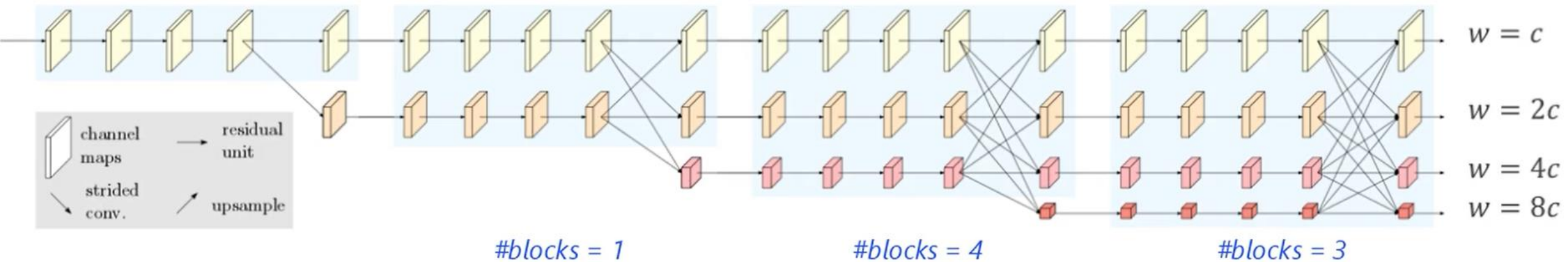
Up-sample: **bilinear** + 1 × 1

Fundamental architecture changes

- Connect high-to-low resolution convolutions in ~~*series*~~ parallel
- Maintain ~~*Recover*~~ high-resolution representations through the whole process ~~*from low-resolution representations*~~
- Repeat fusions across resolutions to strengthen high- & low-resolution representations

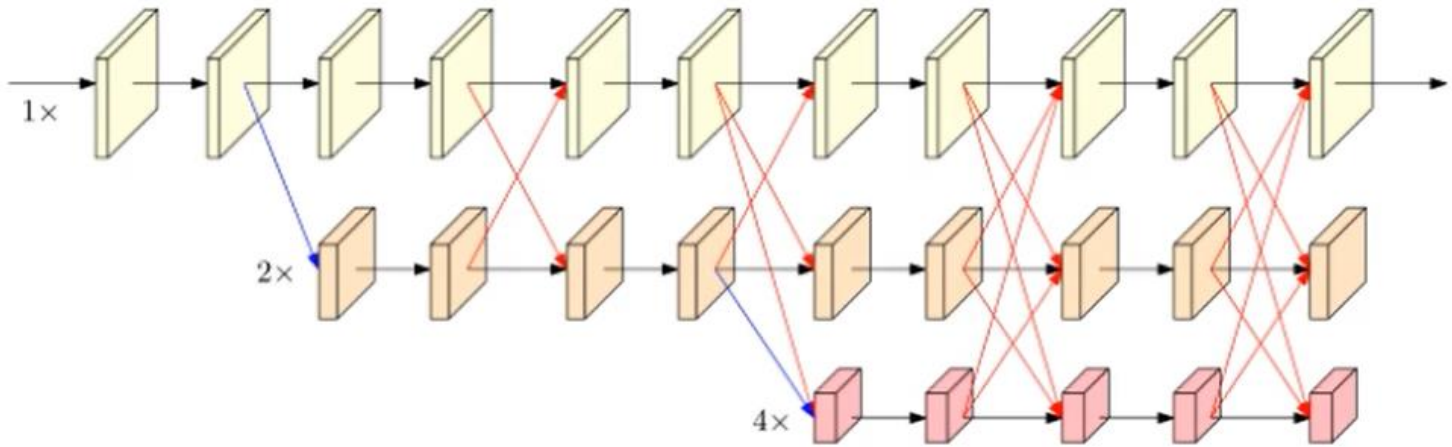
HRNet can learn *high-resolution strong* representations

HRNet instantiation



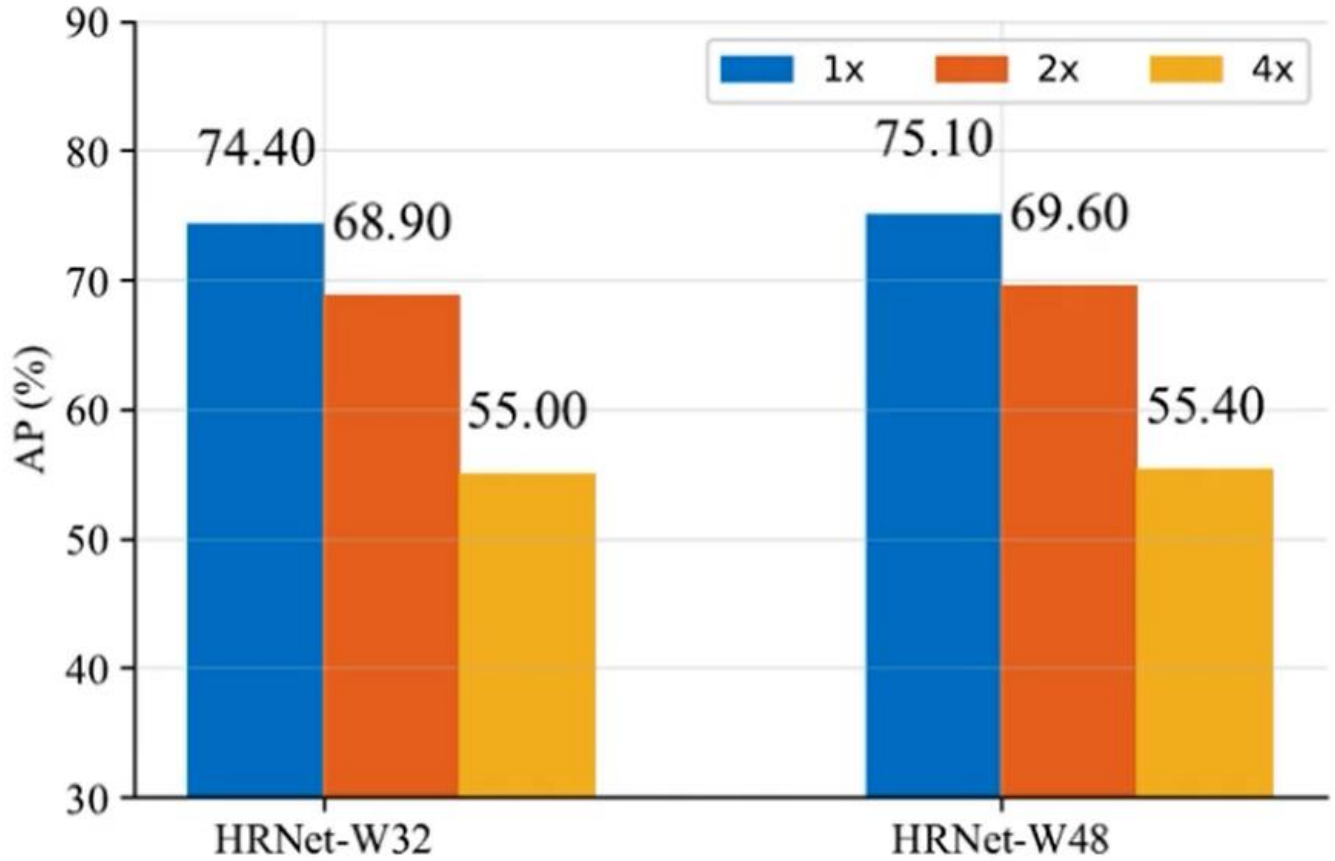
Ablation study: repeated across-resolution fusion

Method	Final exchange	Int. exchange across	Int. exchange within	AP
(a)	✓			70.8
(b)	✓	✓		71.9
(c)	✓	✓	✓	73.4



COCO, train from scratch

Ablation study: high- and low-resolution representations



COCO, train from scratch

Visual recognition applications

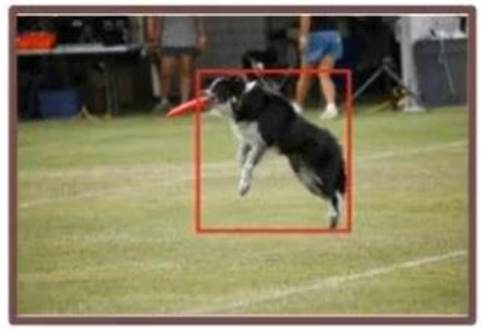




Image classification

Object detection

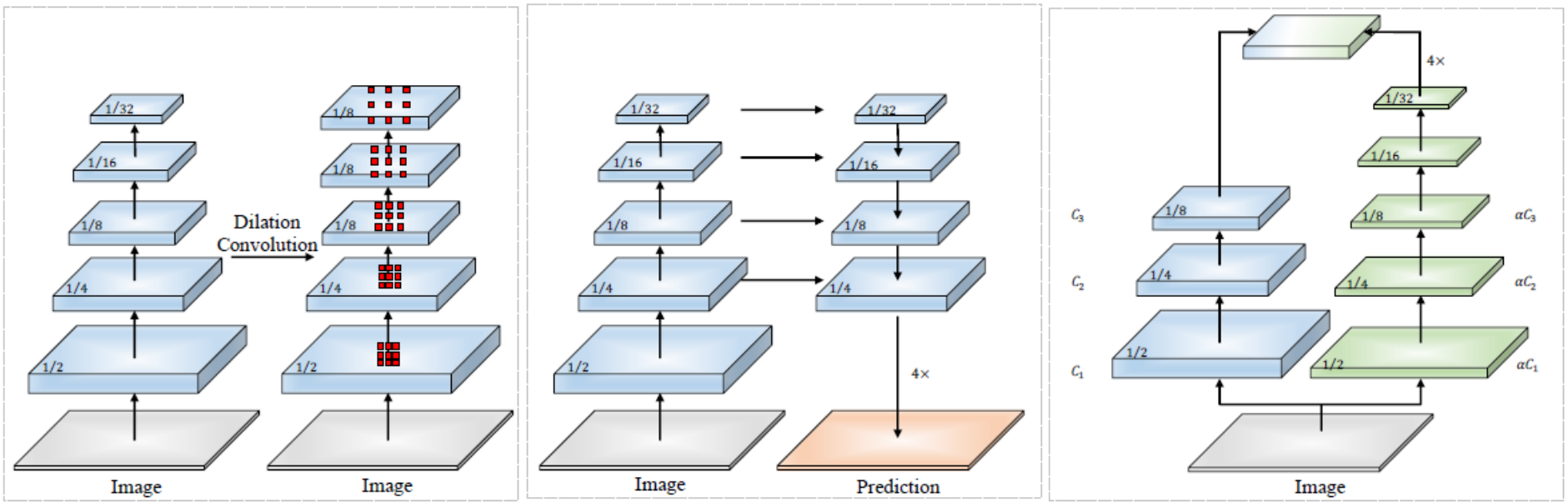
Semantic segmentation

Face alignment

Pose estimation

BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation

Changqian Yu[1,2] · Changxin Gao[1*] · Jingbo Wang[3] · Gang Yu[4] · Chunhua Shen[2] · Nong Sang[1]



(a) Dilation Backbone (b) Encoder-Decoder Backbone (c) Bilateral Segmentation Backbone

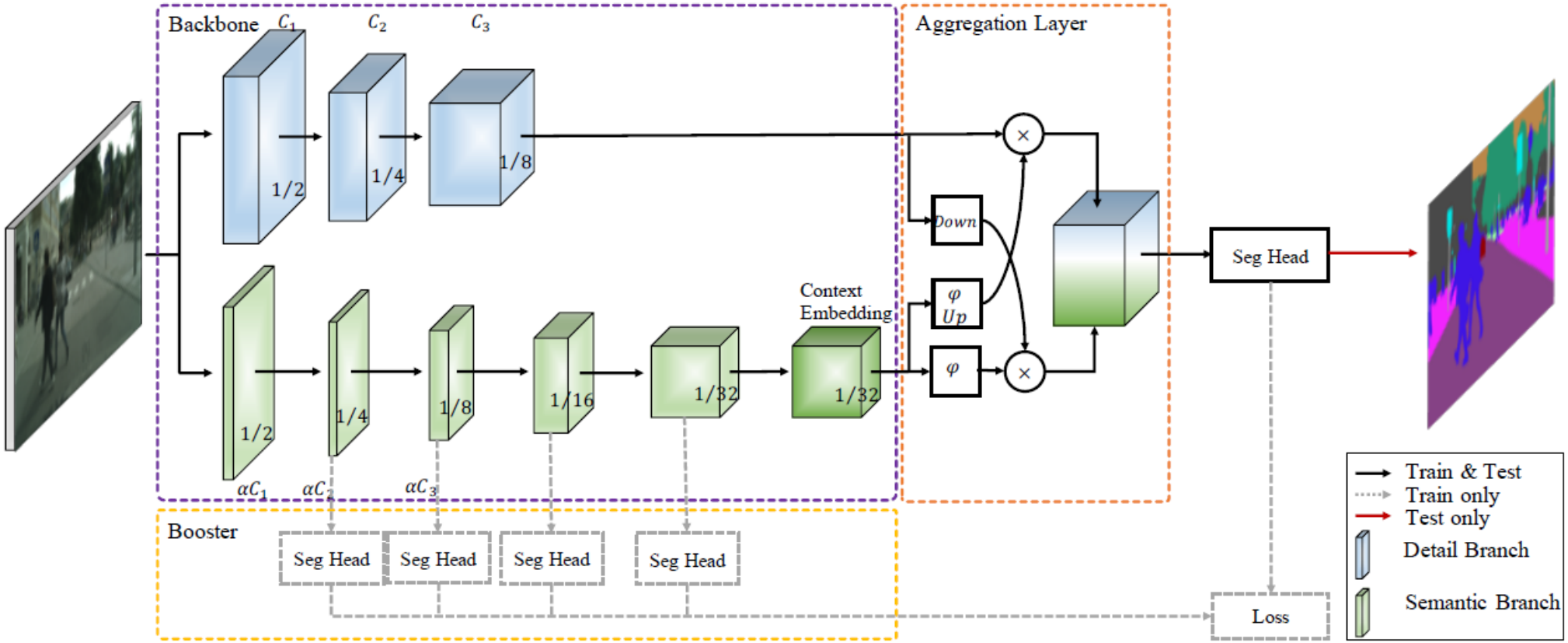
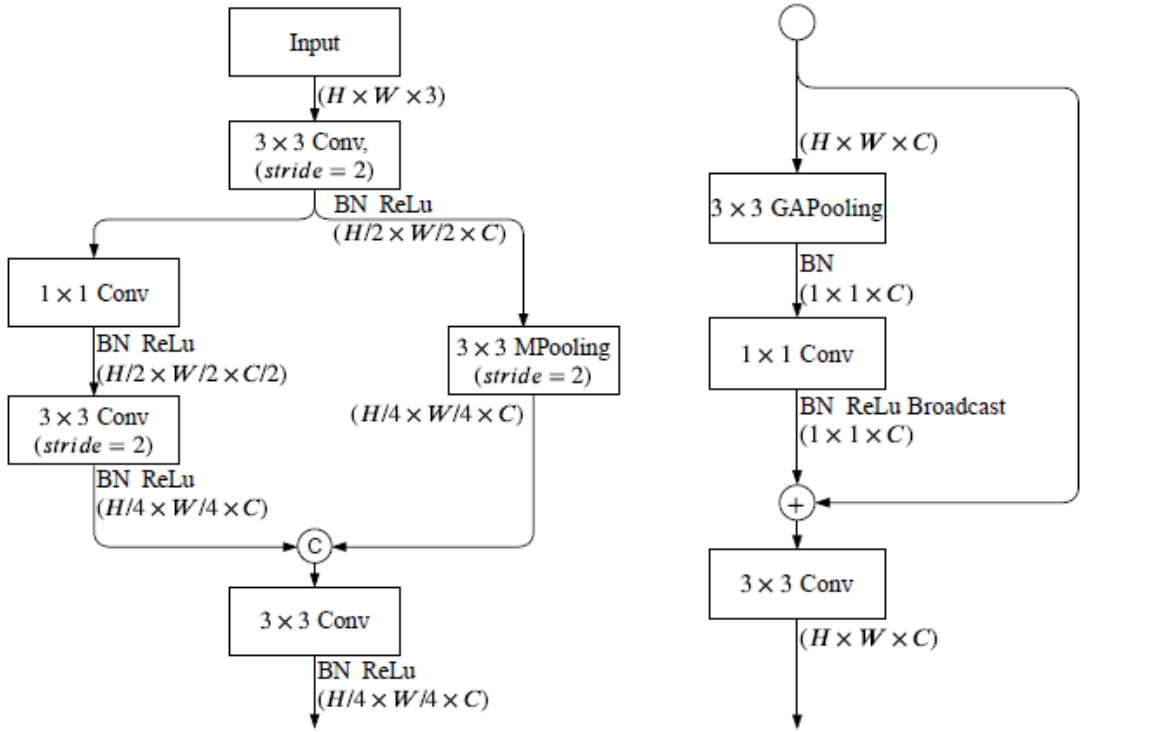
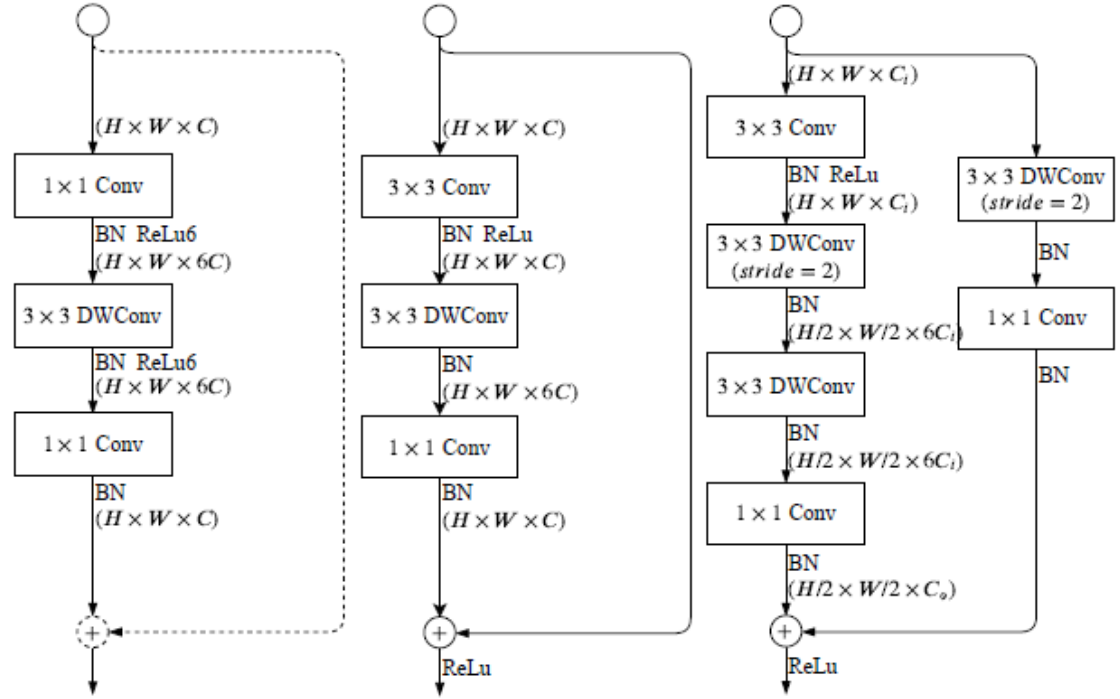


Fig. 3 Overview of the Bilateral Segmentation Network. There are mainly three components: two-pathway backbone in the purple dashed box, the aggregation layer in the orange dashed box, and the booster part in the yellow dashed box. The two-pathway backbone has a Detail Branch (the blue cubes) and a Semantic Branch (the green cubes). The three stages in Detail Branch have C_1, C_2, C_3 channels respectively. The channels of corresponding stages in Semantic Branch can be made lightweight by the factor $\lambda(\lambda < 1)$. The last stage of the Semantic Branch is the output of the Context Embedding Block. Meanwhile, numbers in the cubes are the feature map size ratios to the resolution of the input. In the Aggregation Layer part, we adopt the bilateral aggregation layer. *Down* indicates the downsampling operation, *Up* represents the upsampling operation, φ is the Sigmoid function, and $\bigotimes$ means element-wise product. Besides, in the booster part, we design some auxiliary segmentation heads to improve the segmentation performance without any extra inference cost.



(a) Stem Block (b) Context Embedding Block

Fig. 4 Illustration of Stem Block and Context Embedding Block. (a) is the Stem Block, which adopts a fast-downsampling strategy. This block has two branches with different manners to downsample the feature representation. Then



(a) (b) (c)

Fig. 5 Illustration of Inverted Bottleneck and Gather-and-Expansion Layer. (a) is the mobile inverted bottleneck Conv proposed in MobileNetv2. The dashed shortcut path and summation circle do not exist with the $stride = 2$. (b)(c) are the proposed Gather-and-Expansion Layer. The bottleneck struc-

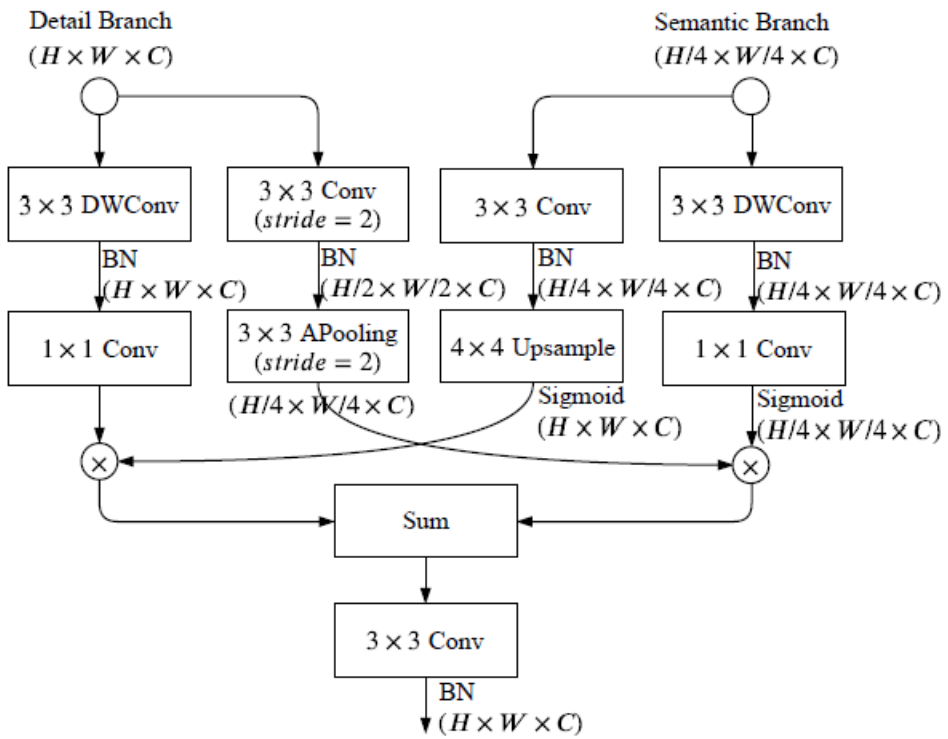


Fig. 6 Detailed design of Bilateral Guided Aggregation Layer. Notation: *Conv* is convolutional operation. *DWConv* is depth-wise convolution. *APooling* is average pooling.

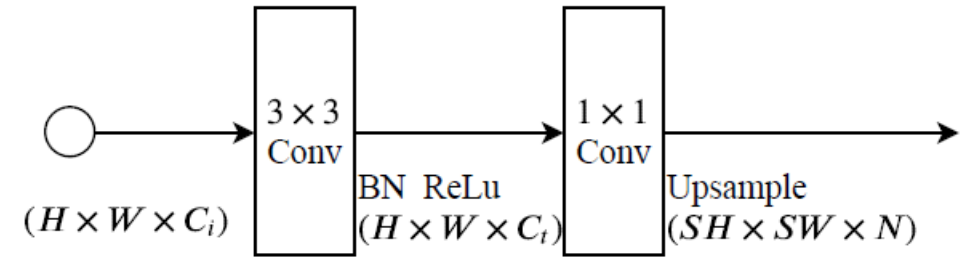


Fig. 7 Detailed design of Segmentation Head in Booster. Notation: *Conv* is convolutional operation. *BN* denotes the batch normalization. *Upsample* means bilinear interpolation. Meanwhile, $1 \times 1, 3 \times 3$ denote the kernel size, $H \times W \times C$ means the tensor shape (height, width, depth), C represents the channel dimension, S denotes the scale ratio of upsampling, and N is the final output dimension.

Table 1 Instantiation of the Detail Branch and Semantic Branch.

Stage	Detail Branch					Semantic Branch						Output Size
	opr	*k*	*c*	*s*	*r*	*opr*	*k*	*c*	*e*	*s*	*r*	
Input												512×1024
S_1	Conv2d	3	64	2	1	Stem	3	16	-	4	1	256×512
	Conv2d	3	64	1	1							256×512
S_2	Conv2d	3	64	2	1							128×256
	Conv2d	3	64	1	2							128×256
S_3	Conv2d	3	128	2	1	GE	3	32	6	2	1	64×128
	Conv2d	3	128	1	2	GE	3	32	6	1	1	64×128
S_4						GE	3	64	6	2	1	32×64
						GE	3	64	6	1	1	32×64
S_5						GE	3	128	6	2	1	16×32
						GE	3	128	6	1	3	16×32
						CE	3	128	-	1	1	16×32

Table 2 Ablations on Cityscapes. We validate the effectiveness of each component step by step.

Detail	Semantic	Aggregation			Booster	OHEM	mIoU(%)	GFLOPs
		Sum	Concate	BGA				
✓							62.35	15.26
	✓						64.68	7.63
✓	✓	✓					68.60	20.77
✓	✓		✓				68.93	21.98
✓	✓			✓			69.67	21.15
✓	✓			✓	✓		73.19	21.15
✓	✓			✓	✓	✓	**73.36**	21.15

Table 3 Ablations on the Semantic Branch design on Cityscapes. We conduct experiments about the channel capacity, the block design, and the expansion ratio of the Semantic Branch. Notation: *GLayer* indicates the Gather Layer, the first 3×3 convolution in GE Layer. *DDWConv* is double depth-wise convolution layer.

	mIoU(%)	GFLOPs
Detail-only	62.35	15.26
$\lambda = 1/2$	69.66	25.84
1/4	**69.67**	21.15
1/8	69.26	19.93
1/16	68.27	19.61

(a) **Channel capacity ratio**: Varying values of λ can control the channel capacity of the first two stages in the Semantic Branch. The channel dimensions of the last two stages are still 64 and 128. Here, we choose $\lambda = 1/4$.

GLayer	DDWConv	Context	mIoU(%)	GFLOPs
✓	✓	✓	**69.67**	21.15
✓	✓		69.01	21.07
✓		✓	68.98	21.15
	✓	✓	66.62	15.78

(b) **Block Analysis**: We specifically design the GE Layer and adopt double depth-wise convolutions when *stride* $= 2$. The second row means we use one 5×5 depth-wise convolution instead of two 3×3 depth-wise convolution. The third row represents we replace the first 3×3 convolution layer of GE Layer with the 1×1 convolution.

	mIoU(%)	GFLOPs
Detail-only	62.35	15.26
$\epsilon = 1$	67.48	17.78
2	68.41	18.45
4	68.78	19.8
6	**69.67**	21.15
8	68.99	22.49

(c) **Expansion ratio**: Varying values of ϵ can affect the representative ability of the Semantic Branch. We choose the $\epsilon = 6$ to make the trade-off between accuracy and computation complexity.

(a) Stage$_1$ (b) Stage$_2$ (c) Stage$_3$

Fig. 8 Examples showing visual explanations for the different stages of the Detail Branch. Following the Grad-CAM (Selvaraju et al., 2017), we visualize the Grad-CAMs of Detail Branch. The visualization shows that Detail Branch can focus on the spatial details, e.g., boundary, gradually.

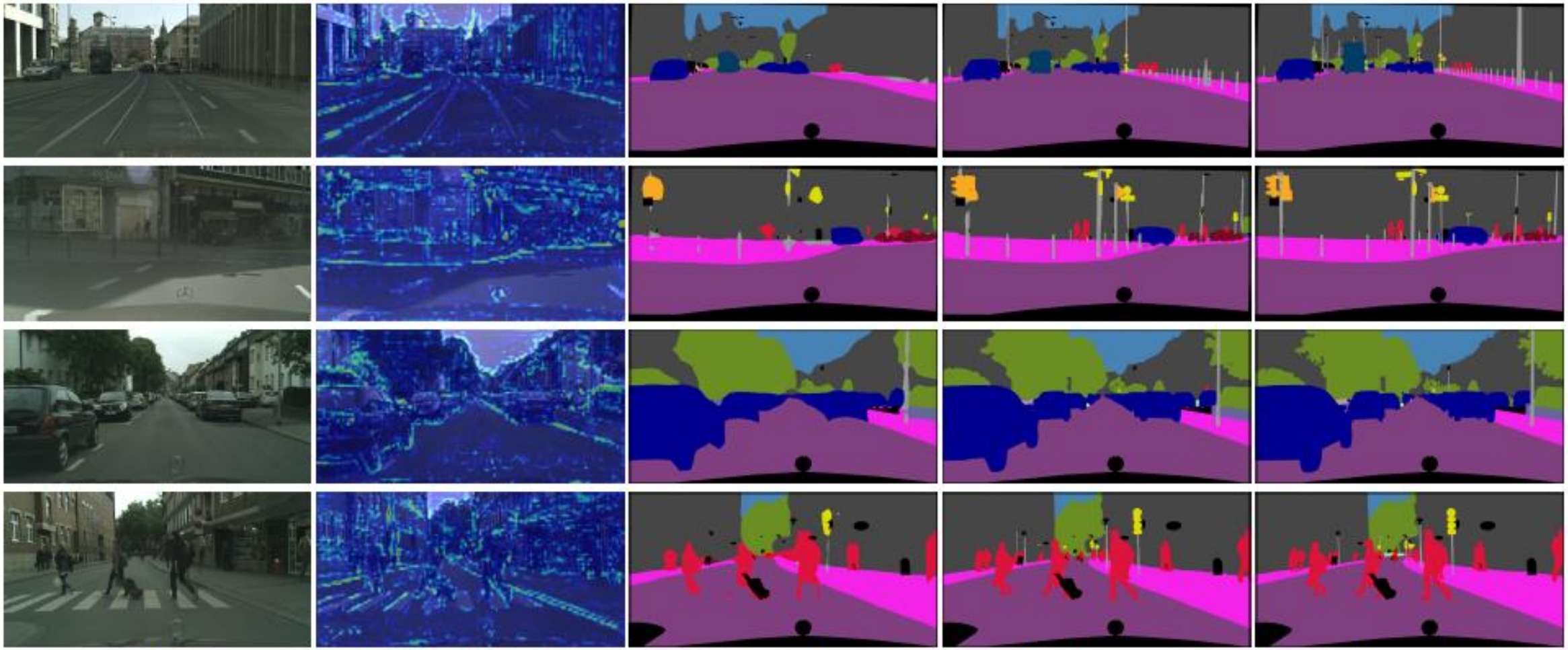


Fig. 9 Visual improvement of the Bilateral Guided Aggregation layer on the Cityscapes *val* set.