Hire-MLP: Vision MLP via Hierarchical Rearrangement (Supplementary Material)

Jianyuan Guo[1,3*], Yehui Tang[1,2*], Kai Han[1], Xinghao Chen[1],
Han Wu[3], Chao Xu[2], Chang Xu[3†], Yunhe Wang[1†]
[1] Huawei Noah's Ark Lab. [2] School of Artificial Intelligence, Peking University.
[3] School of Computer Science, Faculty of Engineering, University of Sydney.
*Equal contribution. †Corresponding author. Code is available: https://github.com/ggjy/Hire-Wave-MLP.pytorch.
{jianyuan.guo, kai.han, yunhe.wang}@huawei.com, yhtang@pku.edu.cn, c.xu@sydney.edu.au

	Size	Layer	Hire-MLP-Tiny	Hire-MLP-Small	Hire-MLP-Base	Hire-MLP-Large
Stage 1	$\frac{H}{4}\times\frac{W}{4}$	Patch Embed.	$[7\times 7, 64, \text{stride } 4]$	$[7\times 7, 64, \text{stride } 4]$	$[7\times 7, 64, \text{stride } 4]$	$[7\times 7, 96, \text{stride } 4]$
		Hire-MLP Block	$\begin{bmatrix} h=w=4 \\ s=2 \end{bmatrix}\times 2$	$\begin{bmatrix} h=w=4 \\ s=2 \end{bmatrix}\times 3$	$\begin{bmatrix} h=w=4 \\ s=2 \end{bmatrix}\times 4$	$\begin{bmatrix} h=w=4 \\ s=2 \end{bmatrix}\times 4$
Stage 2	$\frac{H}{8}\times\frac{W}{8}$	Patch Embed.	$[3\times 3, 128, \text{stride } 2]$	$[3\times 3, 128, \text{stride } 2]$	$[3\times 3, 128, \text{stride } 2]$	$[3\times 3, 192, \text{stride } 2]$
		Hire-MLP Block	$\begin{bmatrix} h=w=3 \\ s=2 \end{bmatrix}\times 2$	$\begin{bmatrix} h=w=3 \\ s=2 \end{bmatrix}\times 4$	$\begin{bmatrix} h=w=3 \\ s=2 \end{bmatrix}\times 6$	$\begin{bmatrix} h=w=3 \\ s=2 \end{bmatrix}\times 6$
Stage 3	$\frac{H}{16}\times\frac{W}{16}$	Patch Embed.	$[3\times 3, 320, \text{stride } 2]$	$[3\times 3, 320, \text{stride } 2]$	$[3\times 3, 320, \text{stride } 2]$	$[3\times 3, 384, \text{stride } 2]$
		Hire-MLP Block	$\begin{bmatrix} h=w=3 \\ s=1 \end{bmatrix}\times 4$	$\begin{bmatrix} h=w=3 \\ s=1 \end{bmatrix}\times 10$	$\begin{bmatrix} h=w=3 \\ s=1 \end{bmatrix}\times 24$	$\begin{bmatrix} h=w=3 \\ s=1 \end{bmatrix}\times 24$
Stage 4	$\frac{H}{32}\times\frac{W}{32}$	Patch Embed.	$[3\times 3, 512, \text{stride } 2]$	$[3\times 3, 512, \text{stride } 2]$	$[3\times 3, 512, \text{stride } 2]$	$[3\times 3, 768, \text{stride } 2]$
		Hire-MLP Block	$\begin{bmatrix} h=w=2 \\ s=1 \end{bmatrix}\times 2$	$\begin{bmatrix} h=w=2 \\ s=1 \end{bmatrix}\times 3$	$\begin{bmatrix} h=w=2 \\ s=1 \end{bmatrix}\times 3$	$\begin{bmatrix} h=w=2 \\ s=1 \end{bmatrix}\times 3$
# Parameters			17.6M	33.1M	58.1M	95.6M
# FLOPs			2.1G	4.2G	8.1G	13.4G

Table A-1. Detailed architecture specifications about variants of the Hire-MLP for ImageNet classification. Hire-MLP block is shown in bracket with the number of stacked blocks. H and W indicate the height and the width of input images, respectively. h and w indicate the number of tokens in each region in *region partition* (Sec. 3.2) along the height direction and the width direction, respectively. s indicates the step size of shifted tokens in *cross-region rearrangement* (Sec. 3.2). FLOPs is calculated on 224×224 input.

A. Detailed Architectures

Due to space limit, Figure 1 in the main paper only shows the tiny version of Hire-MLP architectures, *i.e.*, Hire-MLP-Tiny. We have also developed diverse variants of Hire-MLP with different memory and computational cost by stacking different numbers of Hire-MLP blocks and expanding corresponding channel dimensions. The detailed architecture specifications are shown in Table A-1, where an input image with the size of 224×224 is assumed for all architectures. For example, the first "Patch Embed." in Hire-MLP-Tiny indicates an overlapping patch embedding layer with the window size of 7, channel dimension of 64, and stride of 4. This operation results in a downsampling of input images by a rate of 4. Furthermore, the parameter h, w, and s in Hire-MLP block represent the number of tokens in each region (Region Partition in Sec. 3.2) along the height direction, along the width direction, and the step size of shifted tokens (Cross-region Rearrangement in Sec. 3.2), respectively.

Specifically, we adopt BN instead of the original LN in the Channel-MLP. The BN and LN achieve similar results (*i.e.*, ± 0.1) on ImageNet in our experiments. We choose BN because "FC-BN" can be merged to "FC" and thus improves the inference speed of our model.

In addition, we provide the PyTorch-like code in Algorithm A-1 associated with the operations used in the hire module.

B. Visualization of Feature Maps

Q4. Down-stream tasks. We evaluate the generalization of Hire-Small trained on ImageNet. Table R3-Q4 reports the performance on CIFAR-10, CIFAR-100, Standford Cars, Flowers-102, and Oxford-IIIT Pets, demonstrating the superiority of Hire-MLP. Final version will discuss more.

In this section, we visualize feature maps generated by the hire module (as is detailed in Figure 1 in the main paper) to further understand the effect of the relative position. Hire-MLP-Small is utilized as the backbone. We compare intermediate features of two cross-region rearrangement manners: ShuffleNet manner and shifted manner. These two manners are also illustrated in Figure B-1. For better visualization, the input image is resized to 1024×1024. Feature maps from the 3rd block in stage 1, the 4th block in stage 2, and the 10th block in stage 3 are shown in Figure B-2, where 12 feature maps are randomly sampled along the channel dimension. We observe that features generated by the shifted manner can capture desired local structures (*e.g.*, edges, lines, and textures) better. Specifically, the structure information is poorly modeled in the first row of "Stage-1 Block-3, ShuffleNet manner", demonstrating that our shifted manner can preserve more information about relative position than ShuffleNet manner.

C. Detailed Experimental Settings

C.1. Object detection on COCO

In order to compare with PVT [21] and CycleMLP [4], we conduct object detection and instance segmentation experiments based on two typical detectors, *i.e.*, RetinaNet [13] and Mask R-CNN [7]. We use AdamW [15] optimizer with a batch size of 4 images per GPU, the initial learning rate is set to 2e-4 and divided by 10 at the 8th and the 11th epoch. The weight decay is set to 0.05. All models are trained for 12 epochs, *i.e.*, "1x" schedule, with single-scale strategy on 8 NVIDIA Tesla V100 GPUs. The input image is resized such that its shorter side has 800 pixels while its longer side does not exceed 1333 pixels during training.

In addition, we utilize another setting to compare with Swin Transformer [14] and AS-MLP [12] on Mask R-CNN [7] and Cascade Mask R-CNN [1]: multi-scale training [19], *i.e.*, the input image is resized such that its shorter side is between 480 and 800 pixels while its longer side does not exceed 1333, and "3x" schedule, *i.e.*, 36 epochs with the learning rate divided by 10 at the 27th and the 33rd epoch.

C.2. Semantic Segmentation on ADE20K

We first follow PVT [21] and CycleMLP [4] and utilize Semantic FPN [9] in mmsegmentation [5] as our base framework. We use AdamW [15] optimizer with a batch size of 2 images per GPU. The initial learning rate is set to 1e-4 with the polynomial decay parameter of 0.9, and the weight decay is set to 0.05. All models are trained for 40K iterations on 8 NVIDIA Tesla V100 GPUs. Input images are randomly resized and cropped to 512×512 at the training phase.

In addition, we follow Swin Transformer [14] and AS-MLP [12] and utilize UperNet [23] as another base framework for fair comparison. The initial learning rate is set to 6e-5 with the linear decay, and the weight decay is set to 0.01. Models are trained with an input of 512×512 on 8 NVIDIA Tesla V100 GPUs with 2 images per GPU for 160K iterations. For data augmentations, we adopt random horizontal flipping, random re-scaling within ratio range [0.5, 2.0], and random photometric distortion. In inference, we report both single-scale results and multi-scale results in Table 10 in the main paper, and resolutions employed in multi-scale testing are [0.5, 0.75, 1.0, 1.25, 1.5]× of that in training phase.

D. More Experiments

D.1. Ablation study on the number of regions.

Table D-1 lists both the ImageNet top-1 accuracy and the ADE20K single-scale mIoU of the ablation study about the number of regions in region partition.

Above Table D-1 shows that the number of tokens in each region significantly influences the final performance. To have a deeper analysis under different input resolutions that may be appropriate on down-stream tasks, we further do experiments for 448×448 input. Table D-2 shows an interesting result that (6,4,4,3) is the best for 448×448 input on ImageNet-1K.

D.2. Transfer Learning

We also evaluate the proposed method on five commonly used transfer learning datasets, including CIFAR10 [11], CIFAR100 [11], Standford Cars [10], Flowers [16], and Oxford-IIIT Pets [17]. We fine-tune the ImageNet pre-trained models on new datasets following [6]. Table D-3 shows the corresponding results.

D.3. Improvement between single-scale and multi-scale on ADE20K.

Table D-4 shows the improvement between single-scale and multi-scale testing of several commonly used backbones and frameworks. We empirically find that models with self-attention mechanism can obtain a larger improvement during multi-scale testing.

D.4. Ablation study on training methods.

We follow [20, 22] to have an ablation highlighting the importance of some tricks for our proposed Hire-MLP. The corresponding results are shown in Table D-5. The top row

Algorithm A-1 Code of the hire module (see Figure 1 in the main paper) in a PyTorch-like style.

```
# We omit the padding of H and W here

# B: batch, C: channel, H: height, W: width
# x: input tensor with shape (B, C, H, W)
# h: number of tokens in height direction region
# w: number of tokens in width direction region
# s: step size of shifted tokens

import torch
import torch.nn as nn

class Mlp_2fc(nn.Module):
    """Implementation of MLP with 1*1 convolution. Input tensor with shape [B, C, H, W]
    """
    def __init__(self, dim, h, w):
        super().__init__()
        self.mlp_h = nn.Sequential(
            nn.Conv2d(dim*h, dim//2, 1, bias=False),
            nn.BatchNorm2d(dim//2),
            nn.ReLU(),
            nn.Conv2d(dim//2, dim*h, 1, bias=True),
        )
        self.mlp_w = nn.Sequential(
            nn.Conv2d(dim*w, dim//2, 1, bias=False),
            nn.BatchNorm2d(dim//2),
            nn.ReLU(),
            nn.Conv2d(dim//2, dim*w, 1, bias=True),
        )

    def forward(self, h_axis, w_axis):
        h_axis = self.mlp_h(h_axis)
        w_axis = self.mlp_w(w_axis)
        return h_axis, w_axis

class Mlp_1fc(nn.Module):
    """Implementation of MLP with 1*1 convolution. Input tensor with shape [B, C, H, W]
    """
    def __init__(self, dim):
        super().__init__()
        self.mlp_c = nn.Conv2d(dim, dim, 1, bias=True)

    def forward(self, x):
        c_axis = self.mlp_c(x)
        return c_axis

# cross-region rearrangement operation
def cross_region_rearrange(x, s):
    h_axis = torch.roll(x, s, -2)
    w_axis = torch.roll(x, s, -1)

# inner-region rearrangement operation
def inner_region_rearrange(h_axis, w_axis, h, w):
    h_axis = h_axis.reshape(B, C, H//h, h, W).permute(0, 1, 3, 2, 4).reshape(B, C*h, H//h, W)
    w_axis = w_axis.reshape(B, C, H, W//w, w).permute(0, 1, 4, 2, 3).reshape(B, C*w, H, W//w)

# inner-region restoration operation
def inner_region_restore(h_axis, w_axis, h, w):
    h_axis = h_axis.reshape(B, C, h, H//h, W).permute(0, 1, 3, 2, 4).reshape(B, C, H, W)
    w_axis = w_axis.reshape(B, C, w, H, W//w).permute(0, 1, 3, 4, 2).reshape(B, C, H, W)

# cross-region restoration operation
def cross_region_restore(h_axis, w_axis, s):
    h_axis = torch.roll(h_axis, -s, -2)
    w_axis = torch.roll(w_axis, -s, -1)

# hire module
def hire_module(x, h, w, s):

    # first two branches
    h_axis, w_axis = cross_region_rearrange(x)
    h_axis, w_axis = inner_region_rearrange(h_axis, w_axis, h, w)
    h_axis, w_axis = Mlp_2fc(h_axis, w_axis)
    h_axis, w_axis = inner_region_restore(h_axis, w_axis, h, w)
    h_axis, w_axis = cross_region_restore(h_axis, w_axis, s)

    # last branch
    c_axis = Mlp_1fc(x)

    x = h_axis + w_axis + c_axis
    return x
```

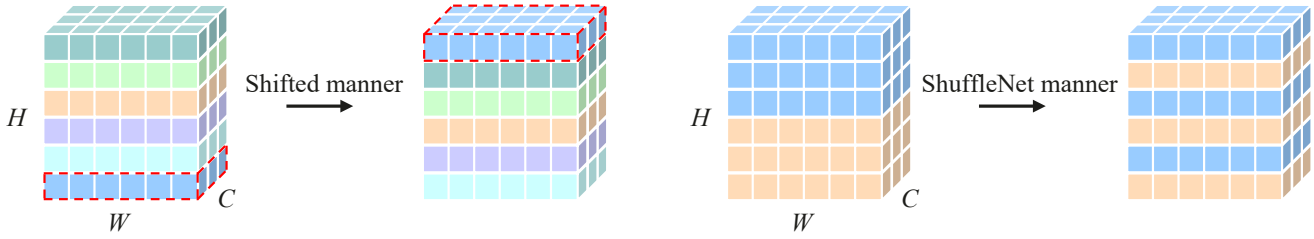

Figure B-1. Illustration of the ShuffleNet manner (group=2) and the shifted manner (s=1) for cross-region rearrangement.

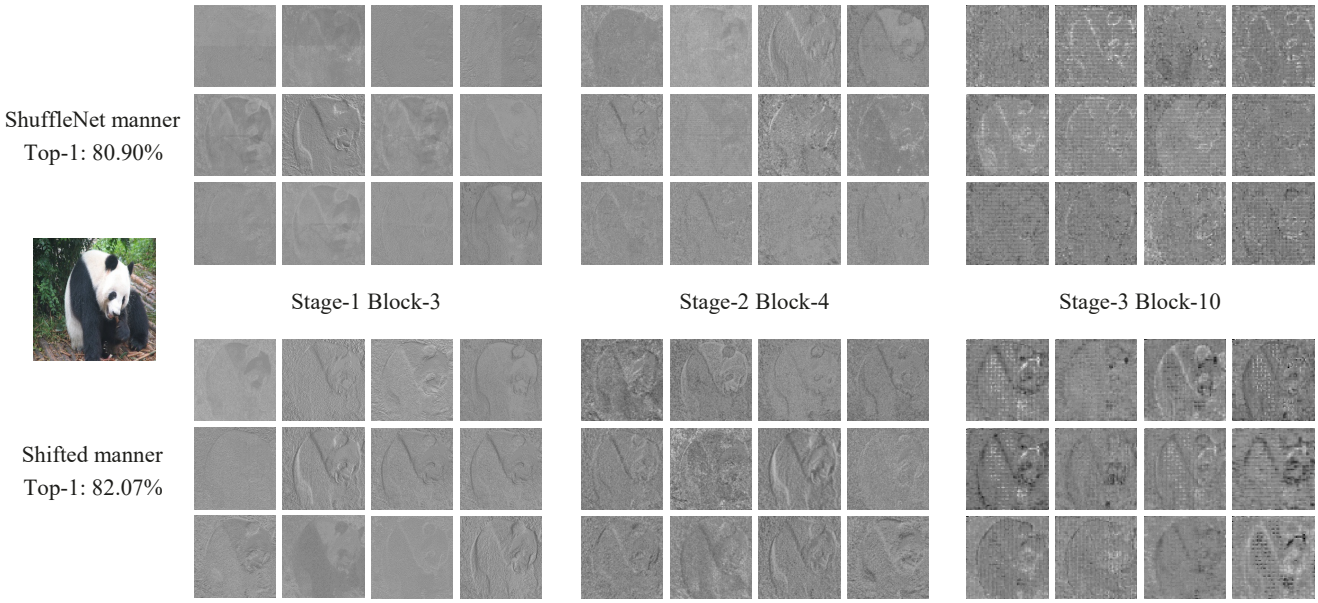

Figure B-2. Visualization of features in the Hire-MLP-Samll (ShuffleNet manner vs. shifted manner) on ImageNet.

corresponds to our default configuration employed for Hire-MLP-Tiny and Hire-MLP-Large. The symbols ✓ and ✗ indicate that we use and do not use the corresponding method, respectively.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. 2

[2] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019. 6

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 6

[4] Shoufa Chen, Enze Xie, Chongjian Ge, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv preprint arXiv:2107.10224*, 2021. 2

[5] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 2

[6] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 2

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2017. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[9] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[10] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In

Num. of h and w	ImageNet Top-1 (%)	ADE20K SS mIoU (%)	Num. of h and w	ImageNet Top-1 (%)	ADE20K SS mIoU (%)
(2, 2, 2, 2)	81.62	44.73	(3, 3, 3, 3)	81.79	45.89
(4, 3, 3, 2)	**82.07**	**45.99**	(4, 4, 4, 4)	81.72	45.26
(5, 4, 3, 3)	81.74	45.50	(6, 4, 3, 3)	81.49	44.83

Table D-1. Ablation study about the number of regions (h and w) in *Region Partition* based on Hire-MLP-Small (ImageNet input size @ 224^2). We report both the ImageNet top-1 accuracy and the ADE20K single-scale mIoU.

Num. of h and w	ImageNet Top-1 (%)	Num. of h and w	ImageNet Top-1 (%)
(2, 2, 2, 2)	80.42	(4, 3, 3, 2)	80.99
(4, 3, 3, 3)	**81.06**	(4, 4, 4, 4)	81.03
(6, 4, 4, 3)	81.27	(8, 6, 6, 3)	81.18

Table D-2. Ablation study on the number of regions (h and w) in *Region Partition* based on Hire-MLP-Tiny (ImageNet input size @ 448^2). We report the ImageNet top-1 accuracy here.

Model	Params/FLOPs	CIFAR-10	CIFAR-100	Cars	Flowers	Pets
T2T-ViT-14	22M/5.2G	97.5†	88.4†	-	-	-
ViP-Small/7	25M/6.9G	98.0†	88.4†	-	-	-
ViT-B/16$_{\uparrow 384}$	86M/17.6G	98.1	87.1	-	89.5	93.8
ResMLP-S24	30M/6.0G	98.7	89.5	89.5	97.9	-
TNT-S$_{\uparrow 384}$	24M/17.3G	98.7	90.1	-	98.8	94.7
Hire-MLP-Small	33M/4.2G	99.0	90.8	93.7	99.0	95.2

Table D-3. Transfer learning results on downstream tasks. † means the setting of ViP/T2T is different. In this setting, Hire-Small achieves 98.7/89.6 on CIFAR10/CIFAR100.

Proceedings of the IEEE international conference on computer vision workshops, 2013. 2

[11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2

[12] Dongze Lian, Zehao Yu, Xing Sun, and Shenghua Gao. As-mlp: An axial shifted mlp architecture for vision. *arXiv preprint arXiv:2107.08391*, 2021. 2

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2017. 2

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 2, 6

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[16] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 2

[17] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, 2012. 2

[18] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 6

[19] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[20] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 2, 6

[21] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 2

[22] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 2

[23] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2, 6

w/o self-attention				w/ self-attention			
Backbone (Framework)	SS mIoU	MS mIoU	△	Backbone (Framework)	SS mIoU	MS mIoU	△
ResNet-50 (UperNet)	42.05	42.78	+0.73	DeiT-B (UperNet)	45.37	47.23	+1.86
ResNet-101 (UperNet)	43.82	44.85	+1.03	ResNet-101 (GCNet)	43.69	45.21	+1.52
ResNeSt-101 (DeepLabV3)	46.47	47.27	+0.80	HRNet-W48 (OCRNet)	39.32	40.80	+1.48
Hire-MLP-Large (UperNet)	48.79	49.87	+1.08	Swin-B (UperNet)	48.13	49.72	+1.59

Table D-4. Improvement between single-scale and multi-scale testing of several commonly used backbones (*e.g.*, ResNet [8], DeiT [20], ResNeSt [25], HRNet [18], and Swin Transformer [14]) and frameworks (*e.g.*, UperNet [23], GCNet [2], OCRNet [24], and DeepLabV3 [3]).

Ablation on ↓	Mix.	Cut.	Era.	DP	Rep.	DO	EMA	Top-1
Hire-Tiny	✓	✓	✓	✗	✓	✗	✗	**79.7**
data augmentation	✓	✗	✓	✗	✓	✗	✗	79.2
	✗	✗	✓	✗	✓	✗	✗	77.7
regularization	✓	✓	✗	✗	✓	✗	✗	79.6
	✓	✓	✓	✗	✗	✗	✗	79.4
	✓	✓	✓	✗	✓	✗	✓	79.2
	✓	✓	✓	✗	✓	**0.1**	✗	79.4
	✓	✓	✓	**0.1**	✓	✗	✗	79.6
Hire-Large	✓	✓	✓	**0.3**	✓	✗	✓	**83.8**
regularization	✓	✓	✓	✗	✓	✗	✓	82.3
	✓	✓	✓	0.3	✓	✗	✗	83.7

Table D-5. Ablation study of Mixup, CutMix, Erasing, Drop Path (DP), Repeat Augmentation, Dropout (DO), and Exponential Moving Average (EMA) on ImageNet @ 224^2.

[24] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, 2020. 6

[25] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 6