Task Generalizable Spatial and Texture Aware Image Downsizing Network (Supplemental material)

Lin Ma[1]
malin_u@126.com
Weiming Li[1]
weiming.li@samsung.com
Hongsheng Li[2]
hsli@ee.cuhk.edu.hk
Qiang Wang[1]
qiang.w@samsung.com
Ji-Yeon Kim[3]
jiyeon31.kim@samsung.com

[1] Samsung Research China - Beijing (SRC-B), Beijing, China
[2] Chinese University of Hong Kong, Hong Kong, China
[3] Samsung Advanced Institute of Technology, Suwon, South Korea

In this material, we first give the details of the proposed DownsizeNet module (Section 1), and then give the details of the implementation on each pipeline (Section 2). We also discuss the relationship between the proposed DownsizeNet and feature map downsizing (Section 3). In Section 4, we discuss the model generalization. In Section 5, we show the results at each milestone of FCN. Finally, we show the downsized samples of RefineDet and FCN (Section 5).

1 The details of the proposed DownsizeNet interpolation method

We propose a new CNN based interpolation method named DownsizeNet which focuses on image downsizing at the image pre-processing stage. Our proposed module is concise and easy to implement. We give the implementation details here.

Besides $2 \sim 4$ convolution layers, and sigmoid layer, there are two special layers, the pooling and the combination (interpolation) layer. For the floating type pooling layer, given a pixel Q on low resolution feature map and its projected pixel Q' on high resolution feature map (Fig. 1), we just select the feature vector v' at P1, and take v' as the feature vector of Q. The projection is the same as bilinear interpolation. Thus, this pooling operation is easy to implement. Here, the P1 is the top left pixel. The vector v' contains the texture around Q'.

For the combination layer, it is similar to the bilinear interpolation. For each pixel Q on the low resolution image, given the four neighbor pixels on the high resolution image and the four corresponding interpolation coefficients, we just need to normalize the interpolation parameters with sum to 1 constraint, and then compute weighted sum as Equation (1) shows.

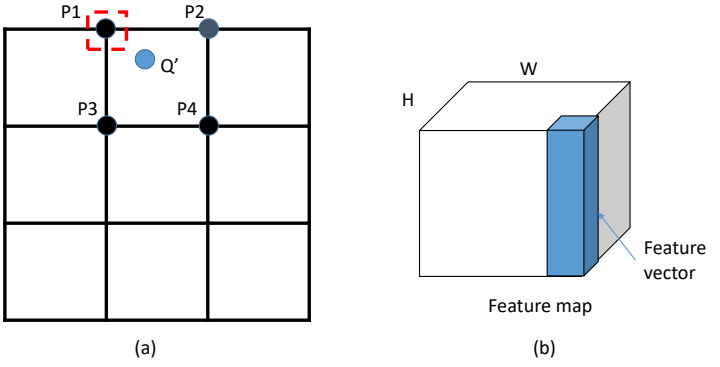

Figure 1: **The floating type pooling.** (a) The corresponding pixels on high resolution feature map about the given pixel Q from low resolution feature map. (b) The feature vector at the pixel.

In another part, we can replace the last sigmoid layer in the coefficient inference subnetwork with softmax, then the sum to 1 operation can be removed in the combination layer.

Thus, the proposed DownsizeNet is easy to implement. Also it can be inserted after the original image conveniently. The users only need to define the size of the low resolution image.

2 The experimental settings for the 7 pipelines

As for the limited and busy GPU resources, the experiments are not done on the same kinds of GPUs. But we guarantee that for each pipeline, the *Bilinear* and ours used the same kinds of and same number of GPUs. Also, we guarantee that *Bilinear* and ours use the same parameter values for the model training and inference. Our experiments are performed based on the source codes of the authors or the widely used source codes in github. The links of the source codes are given in the main manuscript. Next, we present the specific implementation details for each pipeline.

For generality purpose, we use 3 convolutions and 16 channels before the floating pooling layer for each task and these hyper-parameter values are the same if not denoted specifically. That is, the performance of each task is not optimized. We can expect better performance with tuned hyper-parameters.

RefineDet. We use 4 K80 GPUs. For convenient upscaling of arbitrary resolution, we remove the transfer module which needs deconvolution layer. The training parameter values are the same as the source code.

CenterNet. We use 1 P40 GPU. Pretrained model is not used. The image is first warped to 300×300, and then resized to 160×160 with DownsizeNet. We use 5 convolution layers before pooling. The batch size is 20. The model is trained for 140 epochs, and the learning rate milestones are defined as [90, 120] and decay rate is 0.1.

LightHead. We use 1 P40 GPU. We define the batch size as 32. The original code defines max epochs as 40, while we use 80 here. We define the learning rate milestones as [60,] and decay rate is 0.1. We also keep the image ratio (height/width) fixed, and define

Table 1: The mAP of *Bilinear* and ours on LightHead for epoch 70 and 80.

Epoch	Bilinear	Ours
70	50.0	**50.3**
80	50.0	**50.5**

Table 2: The mAP of Bilinear and ours on DETR at four milestones.

Epoch	Bilinear	Ours
1	6.2	**6.6**
150	15.7	**15.9**
175	16.0	**16.2**
200	16.1	**16.3**

$[height, width] \times scale$ as the new height and width. We find that the mAP for Bilinear and Ours have been steady until epoch 80 (Table 1).

DETR. We use 2 V100 GPU. We found that when the image was resized to very small resolution, the training cannot converge very well. Thus, we just use the model trained on larger image resolution as pretrained model (https://dl.fbaipublicfiles.com/detr/detr-r50-e632da11.pth). The pretrained model is downloaded from the github. We define the batch size as 4. We define the max epoch as 200, and set the learning rate milestones as [100,] and decay rate as 0.1. We found that after 175 epochs, the performance became steady for both Bilinear and Ours. Besides, ours consistently outperforms Bilinear at each milestone as Table 2 shows.

FCN. We use 1 K80 GPU, and use 4 convolution layers before pooling. And we use implementation of voc-fcn16s. The original FCN has no data augmentation. We found data augmentation was important for our method. Thus, we add random cropping in data preprocessing. Let h and w be the original height and width, we define,
rand_thres = min(min(h,w)/3,20),
y0 = rand(0, rand_thres)
x0 = rand(0, rand_thres)
y1 = rand(h-rand_thres, h-2)
x1 = rand(w-rand_thres, w-2)
Then the new image is cropped from the region [y0:y1, x0:x1], rand() represents selecting a random integer within the given range. We define the max iteration number as 400K.

DeepLabv3+. We use 1 V100 GPU. We define the max epochs as 200 as suggested by the source code. We define the batch size as 16.

SPNet. We use 1 V100 GPU. We define the max epochs as 180 as suggested by the source code. We define the batch size as 4. To deal with small resolution, we define the $pool\_size$ for $self.strip\_pool1$ and $self.strip\_pool2$ as (10, 6) (original is (20, 12)) in the file 'spnet.py'.

3 Relationship between the proposed DownsizeNet and feature map downsizing

Generally, the feature map downsizing uses strided convolution, max pooling, etc., to downsize the resolution for higher receptive field or higher speed. However, these downsizing

methods lead to integer times downsizing, for example from 200×200 to 100×100, but not 155×155 to 100×100. Thus, these methods cannot be used directly to deal with image downsizing in image preprocessing. In contrast, the proposed DownsizeNet can deal with arbitrary ratio image downsizing effectively with a specifically designed floating pooling layer.

Bilinear interpolation is also used in feature map resizing in some methods. For example, SPNet [3] and DeepLabv3+ [1] uses bilinear interpolation to upscale the feature map to original size to obtain segmentation result. However, the bilinear interpolation operation is the same as used in dealing with image. And the bilinear interpolation only uses relative distance to infer the interpolation coefficients and cannot adapt to the texture or semantic feature variation. Different from the bilinear interpolation method, our interpolation method not only involves the relative distance information but also adapt to the texture variation in interpolating the neighbor image pixels. Besides, our interpolation method can also be used to deal with feature maps. That is, we can replace the bilinear interpolation with our proposed interpolation method to resize the feature maps. But in this paper, we only focus on image downsizing at the image pre-processing stage.

4 Generalization ability of model pretrained on different tasks

In this section, we further discuss the generalization ability of DownsizeNet. We compare the generalization of DownsizeNet and traditional pooling based convolution. With the same setting as in Section 5.3 in the main manuscript (first paragraph), for comparison, we designed a sub-network which only has convolution layers. The comparison sub-network (denoted as *Conv-Down*) has three convolution layers as our sub-network (before pooling), while the last convolution layer output a 3 channel feature map which is directly used in following CNN task. The convolution filter numbers are all set to be 16 as ours. The last convolution layer has stride 2 in Conv-Down. The two downsizing sub-network both are trained on DeepLabv3+, and then used on LightHead. On LightHead, we keep the weights of the pretrained Downsizing module and the VGG backbone fixed, and only train the rest part of the network. From Fig. 2, we can see that ours obtains much better result than *Conv-Down* at all epochs. When the backbone needs to be fixed and only the head part is trained, our DownsizeNet can still obtain good performance. The image is resized from 300×300 to 150×150 in this test.

We have tested the generalization ability of our model in Section 5.3 in the main manuscript. We made two experiments using DeepLabv3+ and LightHead. One test is that we pretrain the DownsizeNet module on DeepLabv3+ and then test on LightHead (denoted as Test A), and the other is that we pretrain the DownsizeNet module on LightHead and then test on DeepLabv3+ (denoted as Test B). We found that in Test A we obtained better result than bilinear interpolation, while in Test B we obtained worse result than bilinear interpolation. We analyze that this difference is mainly caused by the model discrimination ability. Generally, the feature extracted by classification are more discriminative than the feature extracted by regression, and then obtains better generalization ability. For example, in visual tracking the discriminative model [2] separating the foreground samples from background can be considered as classification problem. The discriminative model is relatively more robust to deal with various challenges in new conditions in experiments. That is, the discriminative

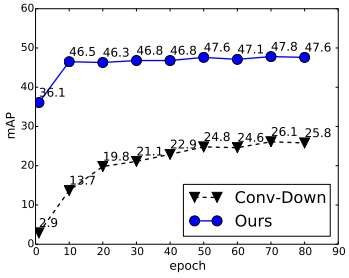

Figure 2: **Test on LightHead about the influences of pretrained downsizing model on Pascal VOC2007.** The downsizing modules are pretrained on DeepLabv3+ with 200 epochs. Results at epoch 1 and $10 \times k, k = 1, 2, ..., 8$ are given.

model is relatively more generalizable. In contrast, the generative model [5] representing the foreground sample distributions can be considered as a regression problem. When the old generative model cannot represent the sample distribution in new conditions or the foreground and background are similar, the tracking tend to fail. That is, the generative model relatively has lower generalization ability than discriminative model. In this paper, DeepLab-v3+ is a classification method and can obtain more discriminative feature. And LightHead contains both classification (for bounding box label) and regression (for bounding box regression). We analyze that the discriminative ability is reduced by the regression task to some extents.

The bounding box regression in detection needs to regress the height, width and center of the box. There can be such conditions where objects of different classes have the same bounding box. In this condition, the regression process just aims to project the samples to the same low dimensional distribution (4D vector here). That is, it does not need to obtain discriminative object representation for different object categories. When the model is used to other tasks, these features of different categories are still entangled with each other. In contrast, segmentation model is more discriminative as a classification task. From this aspect, we can see that the features obtained by classification are more representative of the sample while the features obtained by regression tend to lose the identity of the sample. Thus, we consider that the model pretrained by classification task have higher generalization ability.

5 Results at each milestone of FCN

In Fig. 3, we show the results at each milestone of FCN [4] for 400K iterations. From the figure, we can see that at the four milestones, our method all obtains better performance and the gaps between *Bilinear* and ours are relatively steady. After 300K iterations, the performances for both *Bilinear* and ours have little gains any more.

6 The downsized samples of RefineDet and FCN

We also show some samples of downsized images in RefineDet and FCN (Fig. 4). From the figure, we can see that the difference mainly exists around the object boundary area

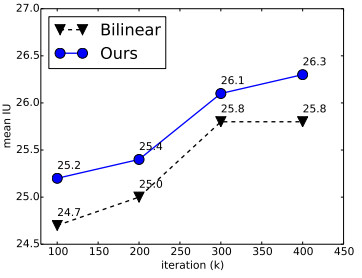

Figure 3: **The results at each milestone of FCN.**

where more image information exists. As the downsized images according to DownsizeNet and bilinear interpolation are both real images and visually similar, it is not easy to see the advantage of DownsizeNet visually from the figure. As our main aim is to optimize the following CNN task performance, the advantage of DownsizeNet is mainly exhibited by the accuracy of CNN tasks. And according to the experimental results on seven pipelines and four datasets, we can see that the CNN task accuracies are improved consistently using the low resolution images downsized by our method.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*, 2018.

[2] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. *ICCV*, 2011.

[3] Qibin Hou, Li Zhang, Ming-Ming Cheng, and Jiashi Feng. Strip pooling: Rethinking spatial pooling for scene parsing. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[5] D. A. Ross, J. Lim, R.S. Lin, and M.H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1):125–141, 2008.

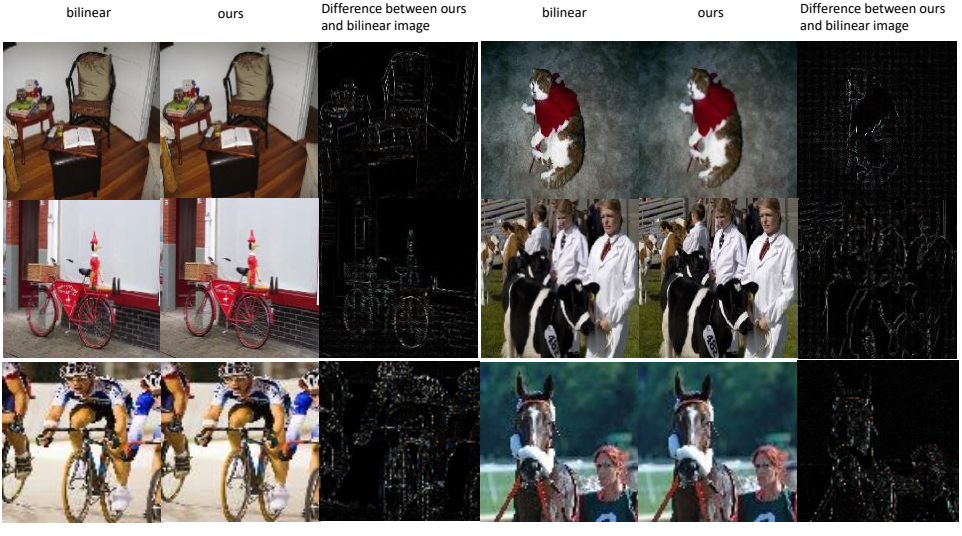

Figure 4: **The output images interpolated with the proposed DownsizeNet.** The top two rows are based on RefineDet, and the last row is based on FCN. The downsized images with bilinear interpolation and the proposed DownsizeNet and the difference between them are presented. The difference image representing the absolute difference between the two images. In the difference image, the min value is assigned 0 and the max value is assigned 255. All other values are projected to the range (0, 255) linearly.