# Task Generalizable Spatial and Texture Aware Image Downsizing Network

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

**Abstract**

Nowadays CNN pipelines often downsize input images to a fixed size to use batch normalization efficiently. For the mostly used downsizing method by bilinear interpolation, information loss may occur since only the relative distance is considered to compute the interpolation coefficients. To preserve more image information, we propose a simple yet efficient interpolation method DownsizeNet, which extracts and fuses local texture information into interpolation by a modified CNN network. Specifically, it encodes the relative distance by a map and aligns it spatially with CNN texture features by our specially designed floating type pooling. The DownsizeNet allows end-to-end training by following CNN task and can be embedded in various CNN networks seamlessly with little extra cost. Experimental results on seven architectures of two tasks, including four object detection pipelines and three classical segmentation pipelines and on four datasets (Pascal VOC2007, MS COCO, Pascal VOC2012 Segmentation and Cityscapes) demonstrate that our method consistently reduces accuracy drop than using bilinear interpolation. Further, we also demonstrate that our interpolation module can generalize well to different pipelines without re-training.

## 1 Introduction

Convolutional neural networks (CNNs) are widely used today in various vision tasks such as classification [14, 34], detection [7, 13, 31, 33, 35, 49] and segmentation [18, 21, 41, 50]. To make full use of GPU in CNN model processing and to use batch normalization, images of various resolutions are usually resized to the same resolution in the pipeline with mostly used bilinear interpolation. As higher resolution image tends to obtain higher accuracy by preserving more image information, some models resize the image to higher resolution [6, 33, 48]. However, the computational load greatly increases for high resolution. Thus, low resolution image is still preferred in many conditions, especially in speed and memory demanding applications. We analyse that the interpolation in many methods only consider

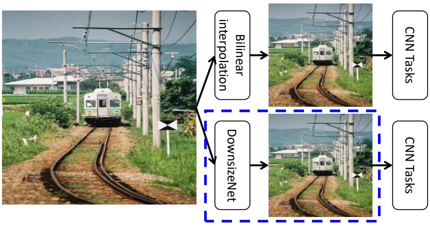


Figure 1: **The image downsizing with the proposed DownsizeNet (fractional resizing ratio).** The figure shows the process of the model with bilinear interpolation and DownsizeNet interpolation respectively. The image is downsized from $300 \times 300$ to $160 \times 160$ in this figure. The DownsizeNet is supervised by following CNN tasks which forms an end-to-end trainable network.

relative distance in inferring the interpolation coefficients [4, 15, 16, 47]. This causes the information loss in image downsizing. In contrast, if the texture is considered in inferring the interpolation coefficients, more texture details or more image information may be preserved. There are some methods which use semantic feature to infer the filter coefficients in feature map downsizing. Zou et al. [52] propose to learn image feature adaptive filter kernels for feature map blurring with a sub-network before max-pooling, and they obtain consistent improvements on several tasks. However, the method [52] only considers the feature map semantic information in inferring the filter kernels. Besides, it is not an interpolation method and cannot deal with arbitrary downsizing scales conditions as in Figure 1.

To solve the image information loss problem, we propose a simple yet efficient interpolation method DownsizeNet which performs interpolation with a CNN sub-network to downsize image at the image pre-processing stage (Figure 1). To our best knowledge, we are the first to introduce texture feature into CNN based interpolation coefficient inference process. A special floating type pooling layer is designed to spatially align the extracted CNN texture feature and the encoded relative distance map. Our interpolation method can retain the pixel position information and also adapt to the local texture variation. Besides, compared to [52] and traditional convolution network, our method can tackle arbitrary resolution scale resizing effectively with the specifically designed pooling layer.

In another part, our model has good generalization ability. This enables the DownsizeNet module to be trained on one task and used on another task directly with fixed weights in interpolation sub-network. In many conditions, for fast implementation with fewer training epochs or when the training set is small, the pre-trained backbone weights, e.g. VGG16 [34], are fixed and only the regression or classification head sub-network is trained. In this condition, the input to the backbone is required to be an image to use the pretrained model, and our proposed DownsizeNet can meet this requirement by keeping the interpolation result being still an image.

Our contributions can be summarized as follows.

- We propose a new interpolation method DownsizeNet aiming to preserve image information in image downsizing with a sub-network interpolation module at the image pre-processing stage. The interpolation sub-network can be easily used in various vision tasks, and has good generalization ability to different pipelines by making the downsizing result retain a real image.
- We introduce texture feature into CNN based interpolation coefficient inference. A special floating type pooling layer is designed to spatially align the CNN texture feature and the encoded relative distance map, and this facilitates the pixel-wise interpo-

lation coefficient inference.

- Experiments on seven pipelines in detection and segmentation tasks demonstrate that our method consistently reduces accuracy drop than widely used bilinear interpolation. Besides, we have demonstrated that our method also outperforms texture only based interpolation, bicubic interpolation and area interpolation.

## 2 Related work

**Image and feature map resizing.** Image resizing is a widely used image processing operation. Traditional methods include bilinear interpolation, bicubic interpolation, etc. These methods only consider the relative distances between the projected pixel and its neighbor pixels while the local texture variation is omitted. In the era of deep learning, the traditional interpolation methods are still widely used in image pre-processing [26, 32] to warp random size images to the same size for batch processing [11, 12, 38]. In the CNN architecture, to reduce the feature map size for speed acceleration, max pooling, average pooling and convolution with larger than 1 integer stride are used. However, they cannot resize the feature map into arbitrary resolution scale for batch processing while our method can. Some researchers find that it is useful to blur the feature map with filters before down-sampling [47, 52]. However, they cannot deal with fractional ratio resizing either. In Meta-SR [16], only the relative distance and scale information are encoded with sub-network for convolution filter inference, while in fact we can also define the relative distance vector with sub-network. Talebi and Milanfar [37] perform traditional bilinear interpolation on extracted convolution feature maps to achieve arbitrary resolution scale resizing. However, the obtained image is not a real-like image and thus their model cannot generalize to other pipelines without retraining as ours. Besides, the bilinear interpolation in their network can be replaced by our interpolation module. In [17, 30], they perform non-uniform downsampling while no interpolation is used and more image information may lose compared with interpolation based methods.

**Super resolution & up-scaling.** As it is noticed that high resolution images tend to obtain higher accuracy in CNNs or have higher visual clarity, many researchers upscale the image resolution [20, 46]. Dong et al. [8] first upscale the low resolution image with cubic interpolation, and then perform convolution on the new image to obtain super resolution image. In [9], the researchers remove the cubic interpolation operation and use deconvolution to upscale the feature map. In some computer vision tasks, upscaling is also widely used, for example in semantic segmentation [29, 42]. Liu et al. [24] performs feature map upscaling with a holistically-guided decoder. Some semantic segmentation methods use bilinear interpolation to upscale feature map to original image size to obtain final segmentation result [4, 15]. Different from the upscaling methods, our method focuses on the information preservation during image downsizing at the image pre-processing stage. Some methods [19, 36, 45] first downscale the image and then upscale the low resolution image for image restoring. They use the difference between restored image and original image as guidance to train the downscaling and upscaling network, and are not designed to optimize various other vision tasks such as detection and segmentation as ours.

**Coordinate operation.** Pixel coordinate is useful information for denoting the position of image element (e.g. edge), especially in position-sensitive tasks such as detection and segmentation. Liu et al. [25] proposes coordinate convolution which embeds the row and column positions of pixels in the convolution operation. AWing [43] finds that the coordinate convolution is effective to encode the spatial relationship between facial landmarks on detected face image patch. Transformer also utilizes the pixel or patch position information

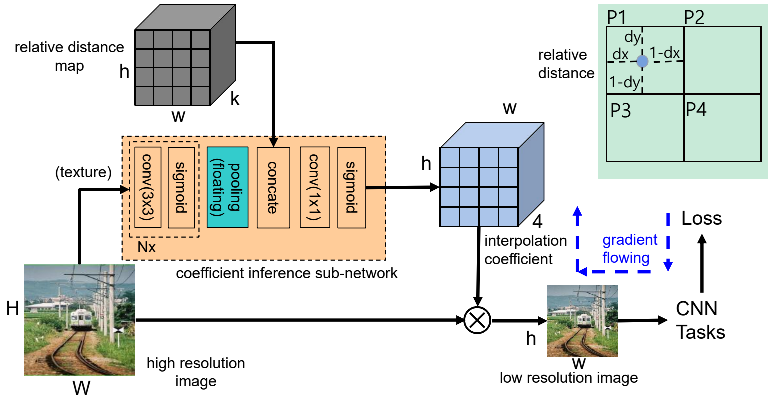


Figure 2: **The flowchart of the proposed DownsizeNet interpolation model.** The relative distance map and the interpolation coefficient map are both pixel-wise. After obtaining the interpolation coefficients (also normalized with sum to 1), the interpolation on four neighbor pixels is performed. The gradient flowing for back propagation is shown as the blue dash line. The relative distance of the projected pixel (blue point) is shown on the top left.

[40]. By embedding the position of patches, the patches can retain the spatial information in the Transformer process [2, 10, 27, 28, 39]. In our method, the relative distance between projected pixel and its neighbor pixels is important to denote the fine object boundary position, and thus this position information is encoded in our method.

# 3 System formulation

Our aim is to provide a fast and effective CNN based interpolation method for image downsizing to replace the hand-crafted bilinear interpolation widely used in CNN pipelines at the image preprocessing stage. There can be other methods to preserve the information in image downsizing, e.g. auto-encoder [44]. However, auto encoder cannot deal with arbitrary downsizing scale as our method. Besides, our model is easy to be implemented and has good generalization ability by retaining the downsized result being still an image.

**Learning for image downsizing**

The proposed interpolation network structure is as shown in Figure 2. We first infer the spatially different interpolation coefficients with a sub-network, and then perform interpolation with these coefficients. The obtained low resolution image is used in the following CNN task. For a specific point $Q$ on the low resolution image, given the pixel intensity $q_i, i = 1, ..., 4$ and interpolation coefficients $a_i, i = 1, ..., 4$ of its four corresponding neighbor pixels on the high resolution image, its pixel intensity is defined as:

$$\begin{aligned} &\widetilde{q} = [q_1, ..., q_4][a_1, ..., a_4]^T. \\ &s.t. \quad \begin{array}{l} \sum_i a_i = 1 \\ 0 \leq a_i \leq 1. \end{array} \end{aligned} \tag{1}$$

Note that the interpolation coefficients are different for different pixels.

The main part of the proposed DownsizeNet is the interpolation coefficient inference sub-network. Different from previous interpolation methods which only use the relative distance between projected pixel and its neighbors [4, 15], we also introduce the texture feature into the coefficient inference process. In this way, the interpolation coefficient can reflect the texture variation. Here, the texture feature for each pixel is extracted by performing convolutions on image patch. As in Figure 2, in this coefficient inference sub-network, we first extract feature map from the high resolution image with $2 \sim 4$ convolution layers with $3 \times 3$ kernels, and then concatenate it with the relative distance map, and then make the new feature map pass a $1 \times 1$ convolution layer and a sigmoid layer. In this way, the pixel-wise fusion of CNN texture and the relative distance is obtained. Then, we can obtain the interpolation coefficient map where sum to 1 constraint is used before interpolation. With sigmoid and sum to 1 normalization, the constraint in Equation (1) is fulfilled [1]. The constraint guarantees that the resized output is still an image. As pre-trained backbone models, e.g. VGG16, are trained on images, this constraint guarantees that the pre-trained backbone models can be fixed for feature extraction to reduce the training time.

Our network is easy to implement. Asides from the traditional convolution and sigmoid layer, there is one special floating point pooling layer to align spatially the texture feature map and the relative distance map. For this layer, for each pixel $Q$ on the low resolution feature map, we project it to the high resolution feature map. Given $P$ as the top left pixel of the four neighbor pixels of the projected pixel, we select the feature vector at $P$ as the feature vector of $Q$. That is, only selection operation is needed in the special pooling layer. This selected feature vector contains the local context information due to the receptive field of the convolution operation.

There can be various ways to define the relative distance map according to the offset $dx$ and $dy$ which represent the offset between the projected pixel and the top left pixel on the high resolution image (top right of Figure 2). That is, the relative distance vector can be defined as $v = [f_1(dx,dy), f_2(dx,dy), ..., f_K(dx,dy)]$ for a specific pixel on the low resolution image where $K$ is the channel number. The $h \times w$ relative distance vectors construct the relative distance map, where $h$ and $w$ are the height and width of the low resolution image respectively. The bilinear interpolation can be considered as a special case of our method. In this condition, we define the coefficient inference sub-network as follows. We define $v = [(1-dx)(1-dy), dx(1-dy), (1-dx)dy, dxdy]$. And define the $1 \times 1$ convolution filters as $g_k = [0, ..., 0, 1, 0, ..., 0], k = 1, ..., 4$, where only the $(C+k)-th$ element is 1 for $g_k$ and $C$ is the channel number of the feature map before pooling. The last sigmoid is removed. Then, our method becomes bilinear interpolation.

**Training the network**

The downsized image can be used for various following CNN tasks such as detection and segmentation. The loss function of DownsizeNet module is the same as the loss function of the following tasks. When having obtained the gradients of the interpolated image, the gradient is back-propagated to the interpolation coefficient inference module. In this way, we obtain an end-to-end trainable system.

# 4 Implementation details

We have tested our method on Linux OS and on GPUs. We tested our method on seven pipelines, including four visual detection models and three semantic segmentation models. Four datasets, i.e. Pascal VOC 2007 [1], MS COCO [23], Pascal VOC 2012 Segmentation [1] and CityScapes [5], are tested in our experiment. We compared our method mainly with bilinear interpolation method (denoted as *Bilinear*). Bicubic interpolation, Area interpolation and texture-only based interpolation are also compared. The same number of neighbor pixels are used for our method and compared methods. As for the limited and busy GPU resources, the experiments are done on different GPUs. But for fair comparison,

[1]The last sigmoid in the coefficient inference subnetwork can be replaced by softmax, then the sum to 1 operation is not needed in the following interpolation operation any more.

Table 1: **The performance (mAP) comparison between Bilinear and ours on four detection pipelines.** RefineDet is implemented with Caffe, and other networks are implemented with PyTorch. $\sim 300$ denotes that one side of the image is resized to 300 while the height and width ratio is fixed.

| Pipeline | Attribute | Dataset | Backbone | Resolution | Bilinear | Ours |
|---|---|---|---|---|---|---|
| RefineDet [48] | One-stage | VOC2007 | VGG16 | $300\times300->160\times160$ | 68.0 | **68.9** |
| CenterNet [51] | Anchor-Free | VOC2007 | ResDCN18 | $300\times300->160\times160$ | 41.0 | **41.9** |
| DETR [3] | Transformer | COCO | ResNet101 | $300\times300->165\times165$ | 16.1 | **16.3** |
| LightHead [22] | Two-stage | VOC2007 | VGG16 | $\sim300->\sim165$ | 50.0 | **50.5** |

we guarantee that for the same pipeline our method and compared methods are performed on the same GPU settings including GPU numbers and types. And all other program settings are the same for our method and compared methods except for the operations specific for the proposed downsizing module. Similar to our method, the compared methods also retain floating type low resolution image. Three convolution layers are used before pooling in the coefficient inference sub-network if not specifically pointed out. Each of the $3\times3$ convolution layers has 16 channels. The relative distance vector is defined as $v=[(1-dx)(1-dy),dx(1-dy),(1-dx)dy,dxdy]$. More implementation details are presented in the supplementary material.

# 5 Experimental results

## 5.1 Experiments on detection models

We test the validity of our method on four detection models, i.e., RefineDet[2] [48], CenterNet[3] [51], LightHead[4] [22] and DETR[5] [3]. The four pipelines represent the four widely used detection strategies. From Table 1, we can see that our method consistently outperforms *Bilinear* by 0.2%∼0.9% mAP. The object detection is sensitive to pixel position. Our method involves relative distance into the interpolation coefficient inference, and then the spatial information can be retained. Besides, our method can adapt to the local image texture variation. And then, our method obtains better results than bilinear interpolation as the table shows. For all the four pipelines, the images are first resized to $\sim 300$ resolution with bilinear, and then resized to $\sim 160$ resolution with DownsizeNet or bilinear interpolation. In another part, as smaller resolution images tend to have lower performance [26, 48] and the image resolution is smaller than the original paper, for example DETR uses larger than $\sim 800$ resolution while it is $165\times165$ here, the performance here is lower than the official reports. All the codes and settings used have been given in the paper. More details are presented in the supplementary material.

Besides, we also tested Bicubic interpolation, Area interpolation and DownsizeNet on RefineDet and VOC2007, and obtained 57.1%, 58.9% and 59.4% mAP respectively. **We are 2.3% and 0.5% higher.** The resolution is reduced from $300\times300$ to $100\times100$. We redesigned the relative distance vector using bicubic coefficients and use $4\times4$ neighbor pixels. In this condition, we have even larger advantage. As more neighbor pixels contain more semantic information, it is more suitable for DownsizeNet. Besides, *Bicubic* and *Area* can also be considered as special cases of our method by carefully defining the relative

[2]https://github.com/sfzhang15/RefineDet
[3]https://github.com/zzzxxxttt/pytorch_simple_CenterNet_45
[4]https://github.com/leowangzi/LightHeadRCNN
[5]https://github.com/facebookresearch/detr

Table 2: **The performance (mean IoU) comparison between Bilinear and ours on three segmentation pipelines.** FCN is implemented with Caffe, and other networks are implemented with PyTorch. SPP: Spatial Pyramid Pooling. For DeepLabv3+, the image is resized to 0.35 times of the original resolution. For SPNet, the image is resized to 0.3 times of the high resolution. The resolutions 513 and 768 are used in the original codes of DeepLabv3+ and SPNet respectively.

| Pipeline | Attribute | Dataset | Backbone | Resolution | Bilinear | Ours |
|---|---|---|---|---|---|---|
| FCN [29] | Fully convolution | VOC2012Seg | VGG16 | Random $->$ $100\times100$ | 25.8 | **26.3** |
| DeepLabv3+ [4] | SPP, encode-decode | Cityscapes | Resnet101 | $513\times513 -> 179\times179$ | 46.4 | **47.3** |
| SPNet [15] | Strip pooling | Cityscapes | Resnet50 | $768\times768 -> 230\times230$ | 58.6 | **59.2** |

Table 3: **Test on LightHead about the generalization ability of the pretrained DownsizeNet model** on Pascal VOC2007 (mAP metric). The interpolation sub-network is pretrained on DeepLabv3+ for 100, 200 and 300 epochs separately. The LightHead is re-trained with 10 epochs. The image is resized to 0.3 times of the high image resolution for both LightHead and DeepLabv3+.

| Training epochs on DeepLabv3+ | Bilinear | Ours |
|---|---|---|
| 100 | 34.7 | **34.8** |
| 200 | 34.7 | **35.1** |
| 300 | 34.7 | **35.2** |

distance vector accordingly.

## 5.2 Experiments on segmentation models

In this part, we test the validity of our method on three classical semantic segmentation models, i.e., FCN[6] [29], DeepLabv3+[7] [4] and SPNet[8] [15]. From Table 2, we can see that our method consistently outperforms *Bilinear* on the three pipelines by 0.5%$\sim$0.9%. In our experiment, we find that with the original FCN code, the performance of our method is slightly lower than *Bilinear*. We analyze that it is because the sample number of VOC 2012 segmentation is not large and we have found no data augmentation in the source code. The data augmentation is important for our method to cover more texture variations. Thus, we augment the training samples by adding random cropping (detailed in supplementary material). With this, our method obtains better results than FCN. In another part, we find that the source code only uses 100K iterations while the performance still goes up after that. Thus, we set the max iteration steps as 400K for FCN. Similar to object detection, semantic segmentation is also position sensitive. Our method can retain the pixel position information effectively. And with the optimized fusion of relative distance map and CNN texture feature by sub-network, our method obtains better results on all the three pipelines.

## 5.3 Generalization ability of DownsizeNet

We also make an experiment to test the generalization ability of our DownsizeNet model. In the downsizing process, we keep the new image as a real image by only inferring the interpolation coefficient values. Then, the DownsizeNet model trained in one task can be used in other tasks. In contrast, the downsizing method using convolution with larger than 1 stride cannot generalize well to other pipelines (more details are in the supplementary material).

[6] https://github.com/shelhamer/fcn.berkeleyvision.org
[7] https://github.com/jfzhang95/pytorch-deeplab-xception
[8] https://github.com/Andrew-Qibin/SPNet

Table 4: **Testing influences of different convolution layer numbers.** The test is made using RefineDet on Pascal VOC2007 dataset.

| Layer number | 1 | 2 | 3 | 4 | 5 | Bilinear |
|---|---|---|---|---|---|---|
| mAP | 68.3 | 68.7 | **68.9** | 68.6 | 68.0 | 68.0 |

To validate the generalization ability of our method, we train the DownsizeNet on DeepLabv3+ [4] (segmentation), and then use this trained DownsizeNet model on LightHead [22] (detection). 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. The image is resized from 300×300 to 150×150. We find that the mAP of LightHead is relatively steady after training for 10 epochs, thus we only tested the performance of the methods at epoch 10 in this test. Also, in this condition we test the generalization of the downsizing module when using only a few epochs training for new tasks for fast implementation. With the same settings as ours, we also train LightHead for 10 epochs with *Bilinear*, and obtain 46.4% mAP which is smaller than ours (46.5% mAP).

In Table 3, we also compare *Bilinear* and ours when pretraining the downsizing modules with different epochs in DeepLabv3+. The image is resized to 0.3 times of the high resolution in Table 3. From Table 3, we can see that when pretraining the proposed DownsizeNet using 100, 200 and 300 epochs, the performance on LightHead varies. When using more training epochs, we can see we have larger advantage than *Bilinear*. As there are random data augmentation in DeepLabv3+, more training epochs can be considered as more training samples of various textures are involved in training. In this way, the model can cover more image texture variations, and then larger advantage is obtained. In another part, our model has better generalization ability on smaller downsizing scale (e.g. ×0.3), where our model can preserve more information than Bilinear due to the context information stored in the interpolation coefficient. We also make a generalization test where we first pretrain the downsizing module on LightHead and then use it on DeepLabv3+. But we find that the performance is not as good as bilinear interpolation. We analyse that it is because DeepLabv3+ can be considered as a classification task and LightHead a regression task (though the bounding box class needs classification), and classification model can extract more discriminative feature which has better generalization ability (such as VGG16 pretrained on classification task [34]).

## 5.4 Ablation study

To investigate the influences of different hyper-parameter settings of the DownsizeNet module, we perform a number of ablation studies based on the detection and segmentation pipelines.

**Influences of different convolution layer numbers.** We test the influences of different convolution layer numbers on RefineDet [48]. Here, we test using $1 \sim 5$ convolution layers before pooling layer in DownsizeNet module. From Table 4, we see that along with the increasing of convolution layer number, the performance of our method first increases and then decreases. The best performance is obtained when the convolution number is 3. More convolution layers represent more nonlinearity and larger receptive field. Thus, from using 1 convolution layer to using 3 convolution layers, the mAP of our method gradually increases. However, when the convolution layer number goes on increasing, the mAP decreases gradually. We analyze that it is because when only considering four neighbor pixels, the DownsizeNet module is easier to meet over-fitting when using more convolution layers.

**Validity of using relative distance.** We test the validity of using relative distance on RefineDet [48]. *Texture* represents our method not using relative distance information in

Table 5: **Testing using relative distance** (mAP metric). The test is made using RefineDet on Pascal VOC2007 dataset. Four convolution layers are used.

| Bilinear | Texture | Ours |
|---|---|---|
| 68.0 | 67.7 | **68.6** |

Table 6: **Testing influences of downsizing scale** on FCN, DeepLabv3+ and RefineDet. VOC2012Seg, Cityscapes and VOC2007 datasets are used respectively. ×0.55 denotes resizing the size to 0.55 times of the high resolution image. ×0.45 and ×0.35 are similar.

| | FCN (mIoU) | | | DeepLabv3+ (mIoU) | | | RefineDet (mAP) | | |
|---|---|---|---|---|---|---|---|---|---|
| | 320×320 | 160×160 | 100×100 | ×0.55 | ×0.45 | ×0.35 | 160×160 | 130×130 | 100×100 |
| Bilinear | 55.3 | 45.3 | 25.8 | 55.2 | 51.9 | 46.4 | 68.0 | 64.2 | 57.7 |
| Ours | 55.3 | 45.5 | 26.3 | 55.3 | 52.5 | 47.3 | 68.9 | 64.7 | 58.8 |

computing the interpolation coefficients. From Table 5, we can see that without relative distance, *Texture* obtains lower performance than ours, and also lower result than *Bilinear*. Relative distance represents where the projected pixel stays on the high resolution image. For tasks sensitive to pixel positions e.g. visual detection, the pixel position information can be retained with the relative distance information. Thus, the performance drops when only using texture feature information. In contrast, by involving both the two factors, our model obtains the best performance.

**Influences of downsizing scale.** We test the influences of downsizing scale on FCN [29] (segmentation), DeepLabv3+ [4] (segmentation) and RefineDet [48] (detection) (Table 6). For FCN, we resize the image from original size to 320×320, 160×160 and 100×100 respectively. We see that the performance drops for both *Bilinear* and ours. But compared with *Bilinear*, our method can still preserve more image information and thus obtains better (or the same) results on all the three image resolutions. In another part, from 320×320 to 100×100, our method outperforms *Bilinear* by 0%, 0.2%, and 0.5% respectively. From this, we can see that when downsizing the image to smaller images, the image information loss is lager while our method can have larger advantage. In Table 6, we can see that DeepLabv3+ has similar result as FCN. For ×0.55, ×0.45 and ×0.35, our method outperforms *Bilinear* by 0.1%, 0.6% and 0.9% respectively on DeepLabv3+ pipeline. We have also tested our method on one visual detection model RefineDet. The image is first resized to 300×300, and then resized to 160×160, 130×130 and 100×100 respectively. We can see that on all the three downsizing scales our method outperforms *Bilinear*. When the image resolution decreases, the performance on both *Bilinear* and ours drops. But our method can have relatively smaller performance drops especially when the image is resized to very small resolution, e.g. 100×100. When downsizing to much smaller resolution, e.g. below ×0.3, the image information loss is relatively much larger as only four neighbor pixels are considered during interpolation. However, with optimized interpolation coefficients which have larger receptive field, our method can preserve more context information and then obtains larger advantage.

## 5.5 Time costs

We also show the time cost of our method and *Bilinear* on RefineDet [48]. The proposed DownsizeNet only uses $2 \sim 4$ convolution layers each of which has 16 channels in this paper. The computational load is small compared with the following CNN architecture (the CNN layers after DownsizeNet). With the setting as in Table 1 and using 1 K80 GPU, bilinear interpolation needs 31.4 ms/image, while ours need 34.2 ms/image, 35.2 ms/image and 37.6 ms/image with 2, 3 and 4 convolution layers respectively. We see that when using 2 convolution layers, our method and *Bilinear* needs approximate time while our method outperforms *Bilinear* by 0.7% mAP. Thus, we can have relatively large performance improvement with little extra time cost.

# 6 Conclusion and discussion

**Conclusion.** In this paper, we propose an interpolation method DownsizeNet which aims to preserve image information in image downsizing at the image pre-processing stage. Besides the relative distance, we also introduce the texture feature information into inferring the interpolation coefficient with the sub-network. Our method can achieve consistent performance improvement than bilinear interpolation while retaining high speed with a few extra convolution layers. Besides, our method has good generalization ability to other pipelines which facilitates the training process on new tasks. However, downsizing the image while retaining high performance is still a challenging task. We wish our work could have some inspirations for more effort to find more effective image information preservation method.

**Discussion.** The DownsizeNet proposed in this paper has the potential to act as a basic module in image resizing process, specifically for image downsizing as in this paper. Also, this proposed module can also be used as an interpolation method in feature map downsizing (or upscaling) and ROI pooling. By introducing the semantic information into inferring the interpolation coefficients, we can expect better performance than only using relative distance. But we mainly focus on the image pre-processing in this paper, and we will make more research about the feature map downsizing in the future.

# References

[1] http://host.robots.ox.ac.uk/pascal/voc/.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv:2102.05095*, 2021.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ECCV*, 2020.

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

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016.

[6] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *NIPS*, 2016.

[7] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. UP-DETR: Unsupervised pre-training for object detection with transformers. *CVPR*, 2021.

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. *ECCV*, 2014.

[9] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. *ECCV*, 2016.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[11] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. *CVPR*, 2019.

[12] Ross Girshick. Fast R-CNN. *ICCV*, 2015.

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[15] Qibin Hou, Li Zhang, Ming-Ming Cheng, and Jiashi Feng. Strip pooling: Rethinking spatial pooling for scene parsing. *CVPR*, 2020.

[16] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution. *CVPR*, 2019.

[17] Chen Jin, Ryutaro Tanno, Thomy Mertzanidou, Eleftheria Panagiotaki, and Daniel C. Alexander. Learning to downsample for segmentation of ultra-high resolution images. *arXiv*, 2021.

[18] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. *CVPR*, 2020.

[19] Heewon Kim, Myungsub Choi, Bee Lim, and Kyoung Mu Lee. Task-aware image downscaling. *ECCV*, 2018.

[20] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CVPR*, 2017.

[21] 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. *ECCV*, 2020.

[22] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head R-CNN: In defense of two-stage object detector. *arXiv:1711.07264*, 2017.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[24] Jianbo Liu, Sijie Ren, Yuanjie Zheng, Xiaogang Wang, and Hongsheng Li. A holistically-guided decoder for deep representation learning with applications to semantic segmentation and object detection. *arXiv:2012.10162*, 2020.

[25] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *NIPS*, 2018.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. *ECCV*, 2016.

[27] 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:2103.14030*, 2021.

[28] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. *CVPR*, 2022.

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

[30] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. *ICCV*, 2019.

[31] Jing Nie, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Enriched feature guided refinement network for object detection. *ICCV*, 2019.

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *TPAMI*, 39(6):1137–1149, 2017.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[35] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: End-to-end object detection with learnable proposals. *CVPR*, 2021.

[36] Wanjie Sun and Zhenzhong Chen. Learned image downscaling for upscaling using content adaptive resampler. *TIP*, 29:4027 – 4040, 2020.

[37] Hossein Talebi and Peyman Milanfar. Learning to resize images for computer vision tasks. *ICCV*, 2021.

[38] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and efficient object detection. *CVPR*, 2020.

[39] Zachary Teed and Jia Deng. Recurrent all-pairs field transforms for optical flow. *ECCV*, 2020.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NIPS*, 2017.

[41] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. *ECCV*, 2020.

[42] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. *CVPR*, 2019.

[43] Xinyao Wang, Liefeng Bo, and Fuxin Li. Adaptive wing loss for robust face alignment via heatmap regression. *ICCV*, 2019.

[44] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, (184):232–242, 2016.

[45] Mingqing Xiao, Shuxin Zheng, Chang Liu, Yaolong Wang, Di He, Guolin Ke, Jiang Bian, Zhouchen Lin, and Tie-Yan Liu. Invertible image rescaling. *ECCV*, 2020.

[46] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. *CVPR*, 2020.

[47] Richard Zhang. Making convolutional networks shift-invariant again. *ICML*, 2019.

[48] Shifeng Zhang, Longyin Wen, Bian Xiao, Lei Zhen, and Stan Z. Li. Single-shot refinement neural network for object detection. *CVPR*, 2018.

[49] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. *CVPR*, 2020.

[50] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *CVPR*, 2021.

[51] Xingyi Zhou, Dequan Wang, and Philipp Krahenbuhl. Objects as points. *arXiv:1904.07850*, 2019.

[52] Xueyan Zou, Fanyi Xiao, Zhiding Yu, and Yong Jae Lee. Delving deeper into anti-aliasing in convnets. *BMVC*, 2020.