ScaleNet - Improve CNNs through Recursively Rescaling Objects

Xingyi Li[1], **Zhongang Qi**[1,2], **Xiaoli Z. Fern**[1], **Li Fuxin**[1]
[1]School of Electrical Engineering and Computer Science, Oregon State University
[2]Applied Research Center, PCG, Tencent
{lixin, xfern, lif}@eecs.oregonstate.edu, zhongangqi@tencent.com

Abstract

Deep networks are often not scale-invariant hence their performance can vary wildly if recognizable objects are at an unseen scale occurring only at testing time. In this paper, we propose ScaleNet, which recursively predicts object scale in a deep learning framework. With an explicit objective to predict the scale of objects in images, ScaleNet enables pretrained deep learning models to identify objects in the scales that are not present in their training sets. By recursively calling ScaleNet, one can generalize to very large scale changes unseen in the training set. To demonstrate the robustness of our proposed framework, we conduct experiments with pretrained as well as fine-tuned classification and detection frameworks on MNIST, CIFAR-10, and MS COCO datasets and results reveal that our proposed framework significantly boosts the performances of deep networks.

Introduction

Deep learning has made significant strides in recent years, achieving extraordinary performance on many tasks in computer vision, such as image recognition (Simonyan and Zisserman 2015; He et al. 2016) and object detection (He et al. 2014; Ren et al. 2015; Dai et al. 2016; He et al. 2017). However, most of the existing approaches are not scale-invariant — their performance can vary wildly in recognizing objects in scales that are not or rarely contained in the training set (Gong et al. 2014). If an image from the training set is down-sampled by half, it is very likely that the pretrained deep model will fail to recognize it. Hence, robustness to scale changes is still an unsolved challenge. To demonstrate this issue more starkly, we conducted a simple experiment on the MNIST dataset by down-sampling the MNIST testing images by half and feeding them to a convolutional neural network (CNN) pretrained on MNIST training set. The classification accuracy drops by an astounding $\mathbf{40}\%$.

One of the most widely applied techniques for tackling this problem is data augmentation, which adds more objects with diverse scales to the training set. However, it is not practical to simulate all possible scales for all the objects in the training set. Besides data augmentation, deformable CNN (Dai et al. 2017) and CapsNet (Sabour, Frosst, and Hinton 2017) claim that they can deal with variant reception fields, but the practical improvements under larger distortions (such as halving the size) are very limited. Some existing techniques attempt to address this problem through integrating side networks with pretrained deep learning models (Lin et al. 2017a; Zhang et al. 2017; Dai et al. 2017; Zhou et al. 2018; Wang et al. 2018). However, none of them formally defines the scale of an object and includes the prediction of scales as an explicit loss function.

In this paper, we propose ScaleNet, which explicitly predicts object scale in a deep learning model. Different from the aforementioned approaches, ScaleNet has an explicit objective to predict the scale of objects in images. Also, our approach does not need to retrain the original object detector or change its network structure. Instead, we aim at rescaling images/objects to fit the pretrained reception fields of the object detector before feeding them to the networks. Thus, the proposed ScaleNet can be applied with any pretrained object classifier/detector to enhance its robustness to scale changes.

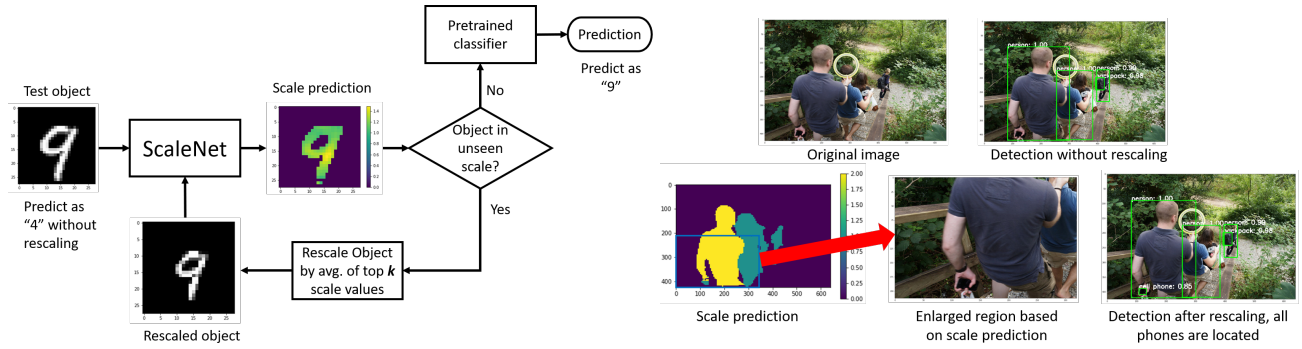
When a new scale is introduced that has never been observed in the training set, ScaleNet may not be able to accurately predict its scale. However, empirically we show that ScaleNet often has sufficient information to predict whether the object should be enlarged or shrinked. Intuitively, this is because the prediction of scales is more likely based on local features such as line thickness and the magnitude of textures, hence more robust than object recognition and affords some extrapolation power. We introduce recursive rescaling to leverage this benefit. Specifically, our approach repeatedly applies ScaleNet and rescales the image/object until the resulting scale falls into the recognizable range for the pretrained classifier/detector.

Our contributions in this paper are as follows:

- We proposed ScaleNet that trains to predict object scales
- By recursively applying ScaleNet and rescaling, pretrained object recognizers can generalize to scales significantly different from those seen in the training set

Related Work

Conventional deep learning based object detection techniques utilize object proposals to arbitrarily specify several



(a) Flowchart of the Recursive ScaleNet

(b) ScaleNet for Detection

Figure 1: (a) Recursive ScaleNet recursively predicts the scale of the object. Even with a scale unseen in training, recursive ScaleNet can gradually bring it to scales that can be recognized by CNN. (b) For detection, we zoom-in on the region that ScaleNet predicts to have a small object, which enables the tiny phone in the hand of the person to be detected after zooming.

different scales to be detected, e.g., Faster RCNN (Ren et al. 2015), R-FCN (Dai et al. 2016), and Mask RCNN (He et al. 2017). To improve the accuracy of multi-scale object detection, many proposed approaches utilize features from intermediate layers of the convolutional neural networks. Some of these approaches combine multi-layer features, e.g., HyperNet (Kong et al. 2016); other approaches predict objects in different scales based on features from different layers, e.g., SSD (Liu et al. 2016), and MS-CNN (Cai et al. 2016). FPN (Lin et al. 2017a) performs both multi-layer features combination and multi-layer prediction.

A standard approach in CNN has been data augmentation (e.g. (Simonyan and Zisserman 2015; He et al. 2014)). This has been shown to improve performance significantly, but it also increases the difficulty to train the model, and is still not robust to scales that are unseen in the training set. Some techniques integrate side-networks with pretrained deep learning models (Lin et al. 2017a; Zhou et al. 2018; Wang et al. 2018; Xu et al. 2018; Zhang et al. 2017; Kim, Kang, and Kim 2018) or attention networks (Wang et al. 2017; Sharma, Kiros, and Salakhutdinov 2015), which output a real-valued attention mask. These side networks are integrated without any explicit losses on scales, and the authors assume that these networks could effectively identify scales of objects during both the training and testing stages.

There has been interest in directly learning transformations (Jaderberg et al. 2015; Lin and Lucey 2017) or deformable filters (Dai et al. 2017) in a deep CNN. Spatial transformer networks (Jaderberg et al. 2015) learn a known transformation by a CNN, which is jointly trained with a classification objective. This approach only learns a global transformation and hence is unable to accommodate the scenarios that multiple objects at different scales are present in the same image. Group invariant (Cohen and Welling 2016; Byravan and Fox 2017) and steerable CNNs (Cohen and Welling 2017) define CNNs on groups, which essentially implement rotated and mirrored CNN filters in practice. However, their filters are still highly localized and do not achieve scale-invariance.

Ke et al. (2017) proposed an architecture that aggregated CNN in 3 different scales, but since the weights on each scale are trained differently, their architecture would not support training on one scale and testing on another. (Bai et al. 2018) proposed a generative adversarial network (GAN) based approach to detect small objects through generating super-resolution images for them. However, this method is not effective for detecting objects in unseen larger scales since those objects already are of sufficient resolution.

Sabour et al. (2017) propose CapsNet with novel units called capsules. They claimed that capsules could encode style information for MNIST digits, and they are robust to various transformations such as scaling, skew, and rotation. However, it is not robust to the very large scale changes in our experiments as shown in Experiments section.

The Zoom-in network (Gao et al. 2018) shares some similarities with our proposed approach in that it zooms into different regions of an image to improve speed and accuracy. A key difference is that the Zoom-in network predefines regions to be zoomed in without knowing if each region actually contains any objects or whether it needs to be rescaled. Predefining the zoom-in regions can be dangerous since some objects can be cut apart which may lead to wrong predictions made by the pretrained detectors. We compared with a predefined zoom-in region approach in Experiments Section. Another critical difference is that their method is not recursive, hence not as robust as our approach for very small objects.

The scale prediction used in our proposed approach shares some similarities with single-image depth prediction. However, the key difference is that an object that is far away in a scene does not necessarily mean that it is small in terms of numbers of pixels. Further, most depth prediction frameworks are specific to indoor scenes (Eigen and Fergus 2015; Eigen, Puhrsch, and Fergus 2014; He, Wang, and Hu 2018).

SNIPER (Singh, Najibi, and Davis 2018) currently leads the performance on MS COCO detection of small objects. It is a data augmentation method that rescales all training images to several predefined resolutions. All COCO training images are rescaled to 480×800, 800×1200, and 1400×2000, respectively, and one independent detector is

trained on each resolution. To predict a test image, it is first rescaled to each predefined resolution, and then fed to the corresponding pretrained detector. Critically, their approach only learns to handle scales used in training and cannot generalize to objects at unseen scales, whereas our approach can generalize to unseen scales.

Proposed Approach

The main idea of this work is to identify the objects whose scales were never encountered in training images, and rescale them to an appropriate scale that can fit the pretrained reception fields of the original object detectors. To achieve this, we first train ScaleNet, which is capable of predicting the object scales at pixel-level within an image. Before feeding a test image to the pretrained object recognizer (classifier/detector), the image is first fed into ScaleNet to predict the scales. The image (or part of the image) is then recursively rescaled until ScaleNet believes its scale falls in the range that the system can recognize. With this recursive rescaling, objects come from unseen scales will gradually be rescaled to known scales that can be recognized by the original object recognizer. After that, we feed the rescaled image to the pretrained detector to generate predictions (Fig.1).

The next two subsections will introduce ScaleNet and the recursive algorithm, respectively.

ScaleNet

The lack of robustness in most deep networks on object scale inspires us to design a network, named ScaleNet, to predict pixel-wise object scales that provide guidance for appropriate rescaling. In order to obtain pixel-wise predictions, we utilize fully convolutional networks (FCNs) as the architecture of ScaleNet. For simple tasks such as recognizing digits in MNIST, a simple network of several convolutional layers with ReLU activation is sufficient to obtain a good performance for scale predictions. For most other tasks with real images, we need FCNs with a more complex architecture.

In this work we utilize U-Net (Ronneberger, Fischer, and Brox 2015) for feature extraction for ScaleNet. The advantage of U-Net is that it combines features from different layers, from low to high level, to provide a fine-grained description for the pixels of the given images. Hence, it is suitable for pixel-wise scale predictions, especially for small objects whose features may not survive downsampling.

Following the definition of scales in MS COCO (Lin et al. 2014), we treat the number of pixels in the segmentation mask as the scale of the object and assign it as the ground truth for all pixels in the object. The loss function in ScaleNet can either be regression-based or classification-based. For regression, we employ mean squared error (MSE) as the loss function. Let $[s_1, s_2]$ be an interval containing all the known scales of a training set. Any object whose predicted scale is smaller than s_1 or larger than s_2 should be zoomed-in or zoomed-out according to its scale prediction.

For classification-based loss, we discretize the scales, e.g., into three bins (small, normal and large). Each pixel is then assigned one of the scale classes if it is part of an object (based on the object size), or assigned to the background class. We consider the focal loss (Lin et al. 2017b) for the resulting classification problem. The focal loss focuses training on a sparse set of hard examples and prevents the vast number of easy examples from overwhelming the learned model during training. It is also capable of dealing with the class-imbalance issue in the training set.

A main discovery in this paper is that ScaleNet does not need to be trained on significantly augmented datasets. Intuitively, local features can be used to indicate the presence of a large/small object, although they may not be sufficient to recognize the object. For example, the boundary of the object can always be detected from the first few layers of the CNN. From the boundaries, a mid-level CNN filter can detect whether another boundary is visible within the spatial scope of the filter, hence recognizing the thickness of the object without recognizing the object. If another boundary is not fully visible, that indicates a thicker and larger object (albeit CNN may not exactly understand how large it is). Similar intuitions apply to texture patterns—if a texture pattern is fully visible within the spatial scope of a CNN filter, the object is probably of a smaller scale, and vice versa if a texture pattern is not fully visible. This intuition is supported by our experiments with ScaleNet. Because of this intuition, we anticipate scale predictions to be more generalizable to previously unseen scales than object recognition.

Recursive-ScaleNet

A key novelty of our proposed algorithm is to utilize recursion to deal with unseen scales that are significantly different from those seen during training. For objects with unseen scales, ScaleNet may not be able to make accurate scale predictions. An important observation is that while the exact scale predictions can be unreliable for out-of-range scales, ScaleNet can still effectively identify whether the object needs to be enlarged or shrank. By recursively rescaling such regions according to the ScaleNet predictions, we can gradually bring the image to the appropriate scale and generalize to large scale changes from the training set.

Algorithm 1 summarizes this recursive procedure in detail. Specifically, given an input image, which can potentially contain multiple objects, ScaleNet is applied to make the pixel-level scale predictions. The predictions are then used to identify regions in the image that contain objects of out-of-range scales, specified by lower bound R_L and upper bound R_H. These regions are cropped by using a bounding box of size W, centered at the center of the pixels with top K out-of-range scales. The cropped regions will then be up-scaled or down-scaled according to the ScaleNet predictions. Each rescaled image region is then stored in the list of images I'_r, and fed into ScaleNet as a new test image to test whether it has out-of-range scales. This process repeats until the predicted scales on all the images in I'_r are within the normal range.

Algorithm 1 can potentially return multiple scaled subimages, each of which can be fed into the object recognition algorithm (either classifier or detector). In object detection experiments, we run object detector on all rescaled subimages from Algorithm 1, and then post process the detector predictions on them by matching the size of the detected ob-

ject and the predicted scale in order to reduce false positives. If the size of the detected object matches the predicted scale from the original image, then we consider the detection to be positive. Otherwise, we would discard the detection as such cropping could often crop parts of larger objects into the cropped bounding box.

Algorithm 1 Recursive-ScaleNet

Input: Testing image I, ScaleNet
Parameters: Stopping scale range $[R_L, R_H]$, rescale window size W, percentage k
Functions: $TopK(x, k)$ obtains the average of top-K values, $rescale(I, scale)$ would rescale the image I with the given scale, $C(c)$ collects all pixels that satisfy the condition c
Output: Rescaled images I'_r

$I' \leftarrow \{I\}$
$I'_r \leftarrow \emptyset$
while $I' \neq \emptyset$ **do**
  for $I_a \in I'$ **do**
    $S \leftarrow ScaleNet(I_a)$
    $P_l \leftarrow C(s_i \in S$ **AND** $s_i > R_H)$
    $P_s \leftarrow C(s_i \in S$ **AND** $s_i < R_L)$
    $I' \leftarrow I' \backslash I_a$
    if $P_l == \emptyset$ **AND** $P_s == \emptyset$ **then**
      $I'_r \leftarrow I'_r \cup \{I_a\}$
      continue
    end if
    while $P_l \neq \emptyset$ **do**
      locate the subimage I_H of size W with most pixels having scales greater than R_H
      $S_{I_H} \leftarrow P_l \cap I_H$
      if $TopK(S_{I_H}) > R_H$ **then**
        $I_H \leftarrow$ rescale$(I_H, TopK(S_{I_H}))$
        $I' \leftarrow I' \cup \{I_H\}$
      else
        $I'_r \leftarrow I'_r \cup \{I_H\}$
      end if
      $S_{I_H} \leftarrow 0$
    end while
    while $P_s \neq \emptyset$ **do**
      locate the subimage I_L of size W with most pixels having scales smaller than R_L
      $S_{I_L} \leftarrow P_s \cap I_L$
      if $TopK(S_{I_L}) < R_L$ **then**
        $I_L \leftarrow$ rescale$(I_L, TopK(S_{I_L}))$
        $I' \leftarrow I' \cup \{I_L\}$
      else
        $I'_r \leftarrow I'_r \cup \{I_L\}$
      end if
      $S_{I_L} \leftarrow 0$
    end while
  end for
end while
return I'_r

For tasks with real images, there are two ways to obtain out-of-range scale predictions. The first method is to refer to the final scale prediction map, and only focus on the classes that unseen small/large objects belong to. For example, if three scale classes, 'Small', 'Normal', 'Large', are defined by a U-Net based *ScaleNet*, one can search for unseen small scale objects from all pixels that are classified as 'Small', which is P_s in Algorithm 1, and vice versa for large objects. The second method is to extract the confidence score map for the unseen scale class, and set a threshold th to exclude all pixels with confidence less than th. Since unseen scale objects are usually predicted with a relatively lower confidence, through adjusting the threshold the second approach is capable of finding more out-of-range objects.

Experiments

To evaluate the performance of the proposed approach, we compare Recursive-ScaleNet with several state-of-the-art approaches on three datasets: MNIST, CIFAR-10, and MS COCO. On MNIST and CIFAR-10, we manually resize the images, in order to examine whether the proposed approach can recognize objects with scales significantly different from ones seen in the training set. On MS COCO, we evaluate the proposed approach on a realistic detector to verify whether it can improve the performance of several state-of-the-art pretrained object detectors in detecting small objects.

MNIST

MNIST contains $60,000$ handwritten digits in the training set and $10,000$ digits in the test set, coming from 10 distinct classes. The original size of each image is 28×28. We utilize a very simple 3 layer convolutional network for ScaleNet, as shown in Table 1. Since there is only one object in every image, we simply rescale the whole image according to ScaleNet prediction. We utilize MSE as the loss function. The batch size is 64 and the optimizer is Stochastic Gradient Descent (SGD) with momentum with learning rate 0.01. The architecture of the pretrained image classifier is a simple 2 layer convolutional network, with 16 and 32 5×5 filters. ReLU and max-pooling layers are utilized after each convolutional layer. The classifier on top consists of two fully connected layers with a ReLU layer in-between.

Layer name	Layer description
Conv1	3×3 conv. 16 w/ ReLU
Conv2	3×3 conv. 32 w/ ReLU
Conv3	5×5 conv. 1

Table 1: The structure of *ScaleNet* for MNIST and CIFAR-10 experiments. Every convolutional layer is followed by a ReLU layer.

Three baselines are used in the experiments. The first one is the original pretrained image classifier. The second one is the Deformable CNN (Dai et al. 2017), which claimed that their deformable convolution filters could adapt to different scales for different objects during test time. The last baseline is CapsNet (Sabour, Frosst, and Hinton 2017) which consists of 'capsules' to encode both the position and style information for each digit. All baselines are tuned to the optimal performance based on validation accuracies.

Detecting large objects The first experiment we conduct is to train with objects of smaller sizes and test on larger objects. To construct the training set, all the MNIST images are rescaled to 14×14, 16×16, and 18×18, respectively. After that, all rescaled images are padded to 28×28 with the object centered.The same rescaling operations are performed on MNIST test set to generate the validation set. We train all baseline models on this new MNIST dataset. *Test accuracies* are measured on the resolutions of 28×28, 56×56, and 84×84, which creates a discrepancy with the training set. To train ScaleNet on the training set, we define 16×16 as the unit scale. All nonzero pixels of 16×16 images are labeled as 1.0. Similarly, 14×14 and 18×18 images are labeled as 0.875 and 1.125, respectively. The R_H in algorithm 1 is set to 1.05, and k is set to 15%.

Table 2 shows that the baselines fail to accommodate significant scale changes, but Recursive-ScaleNet can. Fig. 2 shows the full recursion procedure of Recursive-ScaleNet for a handwritten digit. Before rescaling, it is predicted as '4' by the pretrained classifier. Recursive-ScaleNet gradually rescales it to the known scale and when rescaling stops automatically, the classifier makes the correct prediction.

Approach	Val. acc.	Test acc. 28×28	Test acc. 56×56	Test acc. 84×84
pretrained classifier	99.76%	77.73%	9.35%	9.96%
deformable CNN	98.62%	63.72%	10.92%	9.76%
capsNet	99.56%	67.30%	23.70%	9.65%
Recursive-ScaleNet	96.37%	98.71%	98.70%	98.72%

Table 2: MNIST performance comparison for detecting large objects. Definitions of validation and test set are found in the paper.

Detecting small objects In the next experiment we test the capacity of *ScaleNet* to detect smaller objects never seen in training. We adopt the same data processing operations as the previous experiment to construct the training and validation dataset. The test set consists of 10000 digits in the scale of 8×8. 16×16 is defined as unit scale 1.0 here, so 14×14 and 18×18 are defined as 0.875 and 1.125 , respectively. R_L in algorithm 1 is set to 0.99, and k is set to 15%.

We use the same baselines as the previous setting. The results are presented in Table 3. It demonstrates that our proposed method is also robust for detecting small objects. The test accuracies drop significantly comparing with Table 2, because handwritten digits are heavily distorted under the resolution of 8×8.

Approach	Val. acc.	Test acc.
pretrained classifier	99.76%	57.96%
deformable CNN	98.62%	8.94%
capsNet	99.81%	41.22%
Recursive-ScaleNet	96.37%	77.20%

Table 3: MNIST performance comparison for detecting small objects. Definitions of validation and test set can be found in the paper.

CIFAR-10

In CIFAR-10, there are $60,000$ RGB images in 10 classes in the training set, and $10,000$ images in the test set. The original size of each image is 32×32. The architecture of ScaleNet as well as its parameters are the same as the one applied in MNIST experiment. The architecture of the pretrained image classifier is VGG-11 (Simonyan and Zisserman 2015). To save space, we only demonstrate the experiment for finding large objects on this dataset, and we only compare to the pretrained classifier since it outperforms other baselines in Table 2 and 3. Furthermore, to prove the importance of recursion concept in our algorithm, we also include the classification accuracy by rescaling test images just once.

To construct the training set, 32×32, 38×38, and 44×44 are picked as training scales and they are all padded to the size of 64×64 with the object centered. The test set is built through rescaling original validation images to the size of 96×96. For other parameter settings of *Recursive-ScaleNet*, k equals 10%, and R_H is 1.338. Based on Table 4, it is obvious that *Recursive-ScaleNet* is very robust to detecting objects in unseen larger scales. Furthermore, the accuracy of rescaling images just once is still not ideal, so recursion is necessary for detecting object at a significant different scale.

Approach	Val. acc.	Test acc.
pretrained classifier	79.32%	29.16%
rescaling once	77.31%	43.71%
Recursive-ScaleNet	73.81%	74.22%

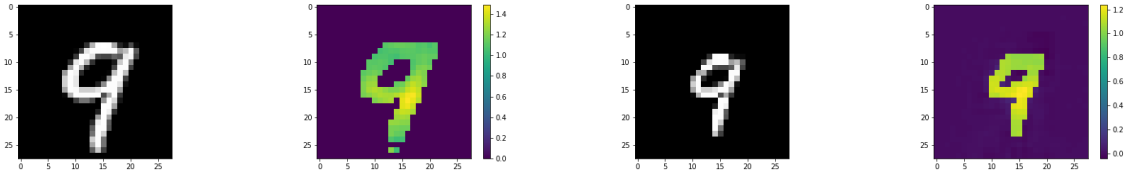
Table 4: Performance comparison for detecting large objects

MS COCO

In COCO, there are about 118K training images containing objects from 80 classes. The validation set contains 5K images, and the test-dev set contains about 22K.

As introduced earlier, the scale of an object is defined as the number of pixels of that object. Object scales are classified into three classes according to MS COCO, which are *small* $(0, 32 \times 32)$, *medium* $(32 \times 32, 96 \times 96)$, and *large* $(96 \times 96, \infty)$, respectively. So, we apply similar rules to modify the output of ScaleNet as classification instead of regression under section . To to address class imbalance, we define object scales as 6 classes instead of 3. These are $\{0$: background class, $1 : (0, 2592)$, $2 : (2592, 2592 \times 8)$, 3 : $(2592 \times 8, 2592 \times 20)$, $4 : (2592 \times 20, 2592 \times 28)$, 5 : $(2592 \times 28, \infty)\}$.

ScaleNet Implementation Details As introduced in Section , the ScaleNet for MS COCO is built upon U-Net (Ronneberger, Fischer, and Brox 2015). The backbone of ScaleNet is VGG-16 (Simonyan and Zisserman 2015). Training images are resized to 512×512 before feeding to ScaleNet. The output of scaleNet is the same size as the input. Focal loss (Lin et al. 2017b) is applied for class imbalance with the parameter $\gamma = 2$. The optimizer is Adam (Kingma and Ba 2014) with $1e^{-4}$ learning rate, and the batch size is 16.

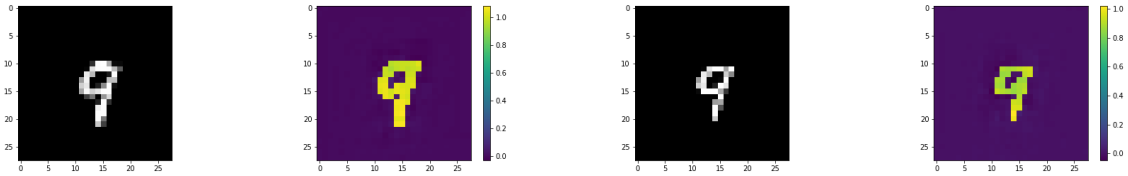


(a) Original image predicted as '4'

(b) Scale Map of (a) avg of top 15% is 1.43

(c) After first rescaling

(d) Scale Map of (c), avg of top 15% is 1.22



(e) After second rescaling

(f) Scale Map of (e) avg of top 15% is 1.07

(g) After Last rescaling predicted as '9'

(h) Scale Map of (g) avg of top 15% is 1.00

Figure 2: Example of the recursion process on MNIST.

Method	Backbone	Avg. Precision, Area:			Avg. Recall, Area:			FPS
		Small	Med.	Large	Small	Med.	Large	
MLKP(Wang et al. 2018)	ResNet-101	10.8	33.4	45.1	15.8	47.8	62.2	NA
STDN513(Zhou et al. 2018)	DenseNet-169	14.4	36.1	43.4	18.3	48.3	57.2	NA
Deformable R-FCN(Dai et al. 2017)	ResNet-101	14.0	37.7	50.3	NA	NA	NA	NA
Faster R-CNN(Ren et al. 2015)	ResNet-101	11.1	38.4	45.2	22.9	51.7	66.1	10.5
Faster R-CNN with Recursive-ScaleNet	ResNet-101	**16.8**	39.8	45.2	**27.4**	54.1	66.8	3.8

Table 5: COCO test-dev 2017 detection results, the training set is trainval35 for all methods, shorter side of images is 600 pixels. FPS (frames per second) is reported based on a single 2080 Ti GPU.

Method	Backbone	Avg. Precision, Area:			Avg. Recall, Area:			FPS
		Small	Med.	Large	Small	Med.	Large	
Deep Regionlets(Xu et al. 2018)	ResNet-101	21.7	43.7	50.9	NA	NA	NA	NA
Learning Region Features(Gu et al. 2018)	ResNet-101-FPN	22.2	43.4	51.6	NA	NA	NA	NA
Mask R-CNN (He et al. 2017)	ResNet-101-FPN	22.9	43.0	51.1	34.2	57.6	68.1	5.2
Mask R-CNN with Grid-Rescale	ResNet-101-FPN	22.7	40.7	51.0	38.8	58.9	68.0	1.0
Mask R-CNN with Rec-ScaleNet	ResNet-101-FPN	24.1	42.3	51.0	38.2	57.7	68.0	1.8
Faster R-CNN (Ren et al. 2015)	ResNet-101-FPN	22.7	42.1	49.8	33.7	56.6	66.4	6.9
Faster R-CNN with Rec-ScaleNet	ResNet-101-FPN	23.8	41.4	49.8	37.4	56.8	66.3	2.5
Faster R-CNN Fine-tuned	ResNet-101-FPN	22.8	42.9	50.4	33.8	57.5	67.8	6.9
Faster R-CNN Fine-tuned with Rec-ScaleNet	ResNet-101-FPN	24.8	42.6	50.4	37.6	57.9	67.7	2.5
SNIPER (soft-NMS)(Singh, Najibi, and Davis 2018)	ResNet-101	24.8	49.8	59.6	38.3	69.4	82.7	2.7
Faster R-CNN Fine-tuned with Rec-ScaleNet (soft-NMS)	ResNet-101-FPN	**25.4**	42.1	50.5	**39.8**	58.5	67.7	2.5

Table 6: COCO test-dev 2017 detection results, the training set is trainval35 for all methods, shorter side of images is 800 pixels.The highest resolution for SNIPER (Singh, Najibi, and Davis 2018) has been rescaled to 800×1280 for fair comparisons. FPS (frames per second) is reported based on a single 2080 Ti GPU.

Pretrained detectors & baselines Multiple pretrained object detectors are experimented, including Faster R-CNN, and Mask R-CNN with backbones of Resnet 101, and Resnet 101-FPN from Facebook Detectron (Massa and Girshick 2018). In order to save running time and memory, we apply the rescaling process up to two times instead of performing the full recursion. To make fair comparisons, we split the competing methods based on their settings of shorter side of images, which are 600, and 800 pixels, respectively. We exclude the comparison with results acquired on higher resolution images, such as a shorter edge of 1000 pixels. Note that using higher resolution images is known

Figure 3: Detection examples for COCO dataset. The first row shows detection results from the pretrained detector. The bottom row shows the results from our proposed method. The confidence threshold is 0.6 for display.

to deliver better performances regardless of the underlying method, due to the higher resolution for small objects.

Small objects detection We focus on detecting small objects for the MS COCO dataset, which is an important topic in computer vision. In this experiment, we do not generate synthetic scales, but focus on improving the performance of existing detectors on the same dataset. To locate small object predictions from *ScaleNet* within an image, which is P_l in Algorithm 1, we focus on the confidence score map for scale 1 $(0, 50\times50)$ of each image associated with a threshold 0.3. Because the output of ScaleNet is classification instead of regression in this experiment, we predefine the rescaling ratio as 2 instead of picking top k percent values from S.

There are a number of post-processing steps for small object detection. Before detection boxes of a sub-image are merged to its original image, all boxes with scale greater than 2 are discarded since the focus of our rescaling is to find small objects only. Plus, all boxes that are close to the edges of the rescale window in Alg. 1 are discarded to avoid predictions on partial objects. Finally, an additional non-maximum suppression (NMS) is performed and the IoU threshold for NMS is the same as the pretrained detector.

The highest resolution of SNIPER (Singh, Najibi, and Davis 2018) is rescaled from 1400×2000 to 800×1280 for fair comparisons. Following SNIPER (Singh, Najibi, and Davis 2018), we apply soft-NMS (Bodla et al. 2017) while merging boxes from sub-images and the performance is reported.

Besides the aforementioned baselines, we add one more baseline that re-scales regions by grid search, named Grid-Rescale. In this approach, each image is divided into 4 equal-sized sub-regions with some overlap. Each sub-region is rescaled to the original image size before feeding to the detector, and the detection boxes from the sub-regions will be merged back through NMS. This style of rescaling is also applied in the Zoom-in network (Gao et al. 2018). Based on Table 5, Grid-Rescale performs even worse than the baseline, the primary reason is that it is much more easier to cut through small objects, and the detector tends to make wrong predictions on those incomplete small objects. On the other hand, regions that are generated from *ScaleNet* could effectively avoid such cases.

The detection performance on the COCO test-dev set is reported in Table 5 and Table 6. *Recursive-ScaleNet* significantly boosts the pretrained object detector and outperforms all other baselines on detecting small objects, on all backbones including faster R-CNN with and without FPN and mask R-CNN, on both average precision and average recall. We found that sometimes CNNs are not capable of processing the rescaled images properly due to not being able to generalize to the pixellated images as a result of upsampling. Thus, fine-tuning on those upsampled images help further improve performance (Faster R-CNN Fine-tuned). However, compared with the baseline that is also fine-tuned on the same images, one can still recognize significant improvements using recursive ScaleNet. The 25.4% average precision on small objects by fine-tuned recursive ScaleNet is the best published performance on images with a short side of 800 pixels, to the best of the knowledge of the authors. Published results that are better than this were achieved on images of higher resolution with a shorter side of $1,000$ pixels or more.

Conclusion

This paper proposes a novel framework that is simple but effective in improving the robustness to scale changes for deep networks. To the best of our knowledge, *ScaleNet* is the first framework that is trained with explicit scale losses, and

the recursive procedure is also introduced for the first time in rescaling or attention approaches. Experiments demonstrate that our proposed framework significantly boosts the scale robustness of CNNs on MNIST and CIFAR-10 datasets, and improves a number of detectors on the average precision and average recall of small objects in the detection task on the MS COCO dataset.

Acknowledgments

This work is partially supported by the National Science Foundation grants CBET-1920945, IIS-1751402, and DARPA contract N66001-17-2-4030.

References

Bai, Y.; Zhang, Y.; Ding, M.; and Ghanem, B. 2018. Sod-mtgan: Small object detection via multi-task generative adversarial network. In *ECCV*.

Bodla, N.; Singh, B.; Chellappa, R.; and Davis, L. S. 2017. Soft-nms – improving object detection with one line of code. In *The IEEE International Conference on Computer Vision (ICCV)*.

Byravan, A., and Fox, D. 2017. Se3-nets: Learning rigid body motion using deep neural networks. In *International Conference on Robot and Automation*.

Cai, Z.; Fan, Q.; Feris, R. S.; and Vasconcelos, N. 2016. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*.

Cohen, T., and Welling, M. 2016. Group equivariant convolutional networks. In *International Conference on Machine Learning*, 2990–2999.

Cohen, T. S., and Welling, M. 2017. Steerable cnns. In *ICLR*.

Dai, J.; Li, Y.; He, K.; and Sun, J. 2016. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *ICCV*.

Eigen, D., and Fergus, R. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*.

Eigen, D.; Puhrsch, C.; and Fergus, R. 2014. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*.

Gao, M.; Yu, R.; Li, A.; Morariu, V. I.; and Davis, L. S. 2018. Dynamic zoom-in network for fast object detection in large images. In *CVPR*.

Gong, Y.; Wang, L.; Guo, R.; and Lazebnik, S. 2014. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*.

Gu, J.; Hu, H.; Wang, L.; Wei, Y.; and Dai, J. 2018. Learning region features for object detection. In *ECCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

He, K.; Gkioxari, G.; Dollar, P.; and Girshick, R. 2017. Mask r-CNN. In *ICCV*.

He, L.; Wang, G.; and Hu, Z. 2018. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *NIPS*, 2017–2025.

Ke, T.-W.; Maire, M.; and Yu, S. X. 2017. Multigrid neural architectures. In *CVPR*.

Kim, Y.; Kang, B.-N.; and Kim, D. 2018. San: Learning relationship between convolutional features for multi-scale object detection. In *ECCV*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Kong, T.; Yao, A.; Chen, Y.; and Sun, F. 2016. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*.

Lin, C.-H., and Lucey, S. 2017. Inverse compositional spatial transformer networks. In *CVPR*.

Lin, T.-Y.; Maire, M.; Belongie, S. J.; Bourdev, L. D.; Girshick, R. B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017a. Feature pyramid networks for object detection. In *CVPR*.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollar, P. 2017b. Focal loss for dense object detection. In *ICCV*.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In Leibe, B.; Matas, J.; Sebe, N.; and Welling, M., eds., *ECCV*.

Massa, F., and Girshick, R. 2018. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark. Accessed: [Insert date here].

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In Navab, N.; Hornegger, J.; Wells, W. M.; and Frangi, A. F., eds., *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *NIPS*.

Sharma, S.; Kiros, R.; and Salakhutdinov, R. 2015. Action recognition using visual attention. In *NIPS Time Series Workshop*.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Singh, B.; Najibi, M.; and Davis, L. S. 2018. Sniper: Efficient multi-scale training. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 9310–9320.

Wang, F.; Jiang, M.; Qian, C.; Yang, S.; Li, C.; Zhang, H.; Wang, X.; and Tang, X. 2017. Residual attention network for image classification. In *CVPR*.

Wang, H.; Wang, Q.; Gao, M.; Li, P.; and Zuo, W. 2018. Multi-scale location-aware kernel representation for object detection. In *CVPR*.

Xu, H.; Lv, X.; Wang, X.; Ren, Z.; Bodla, N.; and Chellappa, R. 2018. Deep regionlets for object detection. In *ECCV*.

Zhang, R.; Tang, S.; Zhang, Y.; Li, J.; and Yan, S. 2017. Scale-adaptive convolutions for scene parsing. *ICCV*.

Zhou, P.; Ni, B.; Geng, C.; Hu, J.; and Xu, Y. 2018. Scale-transferrable object detection. In *CVPR*.